С. В. Веретехина,
В. Л. Симонов,
О. Л. Мнацаканян

Модели, методы, алгоритмы и программные решения вычислительных машин, комплексов и систем

Учебник

DirectMEDIA

С. В. Веретехина, В. Л. Симонов,
О. Л. Мнацаканян

Модели, методы, алгоритмы и программные решения вычислительных машин, комплексов и систем

Учебник

Издание второе

DirectMEDIA
Москва
Берлин
2021

УДК 004.7(075.8)
ББК 32.97я73
В31

Рецензенты:
В. Э Вольфенгаген — д-р техн. наук, проф, проф. каф. кибернетики Национального исследовательского ядерного института «МИФИ» (г. Москва);
П. Н. Миронов — канд. техн. наук, начальник отдела аппаратно-программных разработок АО «Государственный научно-исследовательский институт приборостроения» (г. Москва)

Веретехина, С. В.

В31 Модели, методы, алгоритмы и программные решения вычислительных машин, комплексов и систем : учебник / С. В. Веретехина, В. Л. Симонов, О. Л. Мнацаканян. — Изд. 2-е, доп. — Москва ; Берлин : Директ-Медиа, 2021. — 306 с.

ISBN 978-5-4499-1937-3

Учебник предназначен для изучения принципов работы микроЭВМ IBM на основе микропроцессоров, логического программирования, программирования микроконтроллерной базы на основе платформы Arduino. Раздел лабораторных работ содержит описание создания дополненной реальности в конструкторе (AR) ELIGOVISION TOOLBOX (EV TOOLBOX). В учебнике проводится ознакомление с интеллектуальными системами и технологиями в инженерии знаний. Описано моделирование работ с основными объектами, процессами и явлениями, связанными с интеллектуальными информационными системами и использованием методов их научного исследования. Дополнительно прилагаются практические задания и методы их решения с использованием алгоритмов Дейкстры, Флойда — Уоршелла, Хаффмана.

Учебник разработан с учетом профессиональных стандартов, сопряженных с профессиональной деятельностью выпускника: «Программист», «Администратор баз данных», «Специалист по информационным системам», «Руководитель проектов в области информационных технологий», «Специалист по тестированию в области информационных технологий», «Технический писатель», «Системный администратор информационно-коммуникационных систем», «Специалист по администрированию сетевых устройств информационно-коммуникационных систем».

Предназначен для бакалавров направлений подготовки 09.03.01 «Информатика и вычислительная техника»; 09.03.02 «Информационные системы и технологии»; 09.03.04 «Программная инженерия», с учетом требования профессиональных стандартов.

УДК 004.7(075.8)
ББК 32.97я73

ISBN 978-5-4499-1937-3

Оглавление

1. Принципы построения ПЭВМ

ПЭВМ (Персональная Электронная Вычислительная Машина) строится по принципу соединения составных частей через общую системную магистраль. Общая системная магистраль - набор шин, по которым передаются данные, места нахождения этих данных (адреса) и управляющие сигналы.

Структура универсальной ПЭВМ показана на рисунке 1.1.

ПЭВМ состоит их следующих блоков:

- микропроцессор (МП);
- оперативное запоминающее устройство (ОЗУ);
- постоянное запоминающее устройство (ПЗУ);
- сопряжения устройств (контроллеры).

Рисунок 1.1 – Структура универсальной ПЭВМ

Назначение ПЭВМ - обрабатывать данные по команде пользователя. В универсальной ПЭВМ через блоки сопряжения подключаются устройства ввода-вывода информации: дисплеи, печатающие устройства, графопостроители, клавиатуры, устройства внешней памяти для хранения программ и данных на магнитных дисках и т. д.

1.1. ВЗАИМОДЕЙСТВИЕ УСТРОЙСТВ ПО СИСТЕМНОЙ МАГИСТРАЛИ

Рассмотрим процесс взаимодействия между элементами по системной магистрали ПЭВМ.

К системной магистрали подсоединены разные блоки: МП, ОЗУ, ПЗУ, порты контроллеров устройств ввода-вывода.

Для возможности взаимодействия в каждый момент времени к системной магистрали подсоединен один ведущий блок, (например МП) и один ведомый блок. Блоки ПЭВМ построены на тристабильной логике (тристабильные логические микросхемы - микросхемы, которые могут переводить свои выводы в три устойчивых состояния: логического нуля, логической единицы и высокоимпедансное).

Все ведомые блоки имеют уникальные адреса. Ведущий блок всегда начинает общение с выдачи на системную магистраль адреса информации об адресе. Все ведомые блоки, чей адрес не совпадает с выставленным,

переводят свои выводы в высокоимпедансное состояние, то есть, отключаются.

Блок, чей адрес совпадает с запросом ведущего блока, выдает на магистраль данных содержащиеся по этому адресу данные или принимает посланные ведущим блока. Для указания направления обмена информацией и моменты приема информации вырабатываются **управляющие сигналы**.

Адресные и управляющие сигналы всегда направлены от ведущего блока, передача данных может быть в обоих направлениях (двунаправленная), но есть ведомые блоки, которые только принимают информацию.

Принцип реализации IBM PC AT *(первая форма реализации общей системной магистрали ПЭВМ)* заключается в том, что для всех этих сигналов применяют параллельно расположенные проводники. Общую системную магистраль разделяют на три части: магистраль адреса, магистраль данных и магистраль управления (рис. 1.2). Общее число проводников магистралей определяется длинами слов адреса (разрядностью магистрали адреса) и данных (разрядностью магистрали данных).

Принцип реализации IBM PC XT *(вторая форма реализации общей системной магистрали ПЭВМ*) заключается в том, что сигналы адреса и данных передаются одни за другими, разделенными по времени (мультиплексированная по времени системная магистраль) [1].

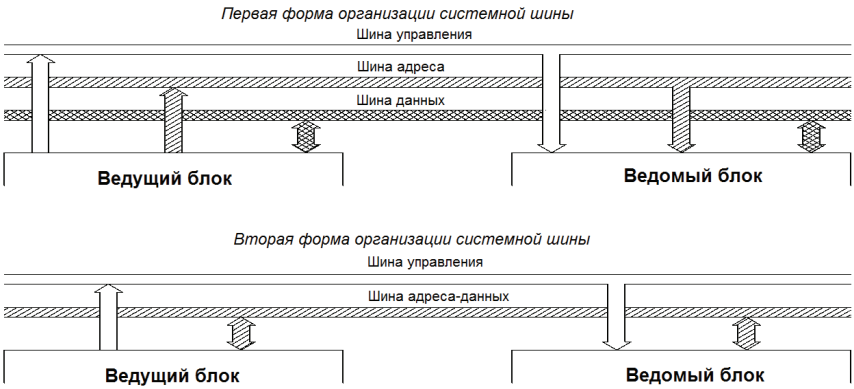


Рисунок 1.2 – Формы организации общей системной магистрали

Во время выполнения операции записи (передача информации к ведомому блока) ведущий блок выставляет адрес и данные. Ведомый блок через необходимое для получения адреса и протекания внутренних процессов время (время обращения) принимает информацию с магистрали данных.

1.2. ЦЕНТРАЛЬНЫЙ ПРОЦЕССОР

Устройство центрального процессора показано на рисунке 1.3. Тракт данных состоит из регистров (обычно от 1 до 32), арифметико-логического устройства (АЛУ) и нескольких соединительных шин. Содержимое регистров поступает во входные регистры АЛУ, которые на рисунке 1.3 обозначены буквами А и В. В них находятся входные данные АЛУ, пока АЛУ проводит вычисления [2].

АЛУ выполняет сложение, вычитание и другие простые операции над входными данными и помещает результат в выходной регистр. Содержимое этого выходного регистра может записываться обратно в один из регистров или сохраняться в памяти, если это необходимо. Не во всех архитектурах есть регистры А, В и выходные регистры. На рисунке 1.3 представлена операция сложения, но АЛУ может выполнять и другие операции.

Команды можно разделить на две группы: команды типа регистр-память и типа регистр-регистр.

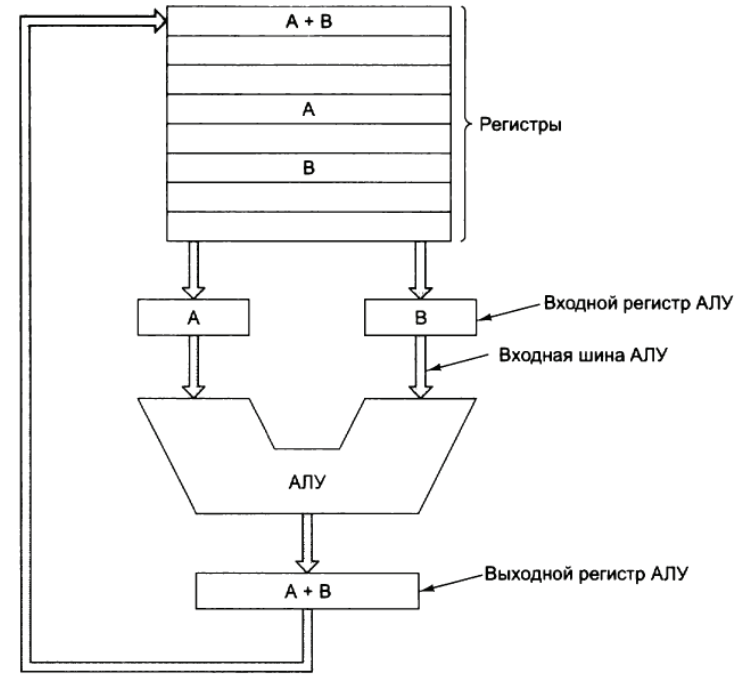


Рисунок 1.3 – Устройство центрального процессора

Команды первого типа вызывают слова из памяти, помещают их в регистры, где они используются в качестве входных данных АЛУ.

Слова – это такие элементы данных, которые перемещаются между памятью и регистрами, размер слова обычно соответствует разрядности регистра данных. Так, у 16-разрядных микропроцессоров 8086 и 8088 слово имеет длину 16 бит, у 32-разрядных микропроцессоров – 32 бита, а у 64-разрядных микропроцессоров – 64 бита. Словом может быть целое число.

Другие команды этого типа помещают регистры обратно в память.

Команды второго типа вызывают два операнда из регистров, помещают их во входные регистры АЛУ, выполняют над ними какую-нибудь арифметическую или логическую операцию и переносят результат обратно в один из регистров.

Этот процесс называют **циклом тракта данных**. В какой-то степени он определяет, что может делать машина. Современные компьютеры оснащаются несколькими АЛУ, работающими параллельно и специализирующимися на разных функциях. Чем быстрее происходит цикл тракта данных, тем быстрее компьютер работает.

Выполнение команд

Центральный процессор выполняет каждую команду за несколько шагов. Он делает следующее:

Шаг 1. Вызывает очередную команду из памяти и помещает ее в регистр команд.

Шаг 2. После декодирования текущей команды (иногда и после её выполнения) меняет положение счетчика команд, который после этого указывает на следующую команду.

Шаг 3. Определяет тип вызванной команды.

Шаг 4. Если команда использует слово из памяти, определяет, где находится это слово.

Шаг 5. Помещает слово, если это необходимо, в регистр центрального процессора (бывают команды, которые требуют загрузки из памяти целого множества слов и их обработки в рамках единственной команды).

Шаг 6. Выполняет команду.

Шаг 7. Переходит к Шагу 1, чтобы начать выполнение следующей команды.

Такая последовательность шагов (выборка – декодирование – исполнение) является основой работы всех компьютеров.

Описание работы центрального процессора можно представить в виде программы. В листинге 1.1 приведена такая программа-интерпретатор на языке Java. В описываемом компьютере есть два регистра: счетчик команд (PC) с адресом предыдущей команды и аккумулятор (AC), в котором хранятся результаты арифметических операций. Кроме того, имеются внутренние регистры, в которых хранится текущая команда (instr), тип текущей команды (instr_type), адрес операнда команды (data_loc) и сам операнд (data). Каждая команда содержит один адрес ячейки памяти. В

ячейке памяти хранится операнд – например, фрагмент данных, который нужно добавить в аккумулятор.

Листинг 1.1. Интерпретатор для простого компьютера (на языке Jawa)

```
Public class Interp{
  static int PC;          //PC содержит адрес следующей команды
  static int AC;          //Аккумулятор, регистр для арифметики
  static int instr;       //Регистр для текущей команды
  static int instr_type;  //Тип команды (код операции)
  static int data_loc;    //Адрес данных или -1, если его нет
  static int data;        // Текущий операнд
  static boolean run_bit = true;  //Бит, который можно сбросить,
                                  //чтобы остановить машину
  public static void interpret(int memory[], int starting_address{
//эта процедура интерпретирует программы для простой машины,
//которая содержит команды только с одним операндом из
//памяти. Машина имеет регистр AC (аккумулятор). Он
//используется для арифметичесих действий – например,
//команда ADD суммирует число из памяти с AC. Интерпретатор
//работает до тех пор, пока не будет выполнена команда
//HALT, вследствие чего бит run_bit поменяет значение на
//false. Машина состоит из блока памяти, счетчика команд, бита
//run bit и аккумулятора AC. Входные параметры представляют собой
//копию содержимого памяти и начальный адрес.

  PC-starting_address;
  while (run_bit) {
      instr = memoru[PC]; //вызывает следующую команду в instr
      PC = PC+1;          //Увеличивает значение счетчика команд
      instr_type = get_instr_type(instr); //определяет тип команды
      data_loc = find_data(instr, instr_type); //находит данные (-1,

                                              //если данных нет)
      if(data_loc>=0) //Если data_loc=-1, значит операнда нет
      data=memory[data_loc];//вызов данных
      rxecute(instr_type,data);//выполнение команды
                  }
              }
 private static int get_instr_type(int addr){…}
 private static int find_data(int instr, int type){…}
 private static void execute(int type, int data){…}
}
```

Сам факт того, что можно написать программу, имитирующую работу центрального процессора, показывает, что программа не обязательно должна выполняться реальным процессором (устройством). Напротив, вызывать из памяти, определять тип команд и выполнять эти команды может другая программа. Такая программа называется **интерпретатором**.

Эквивалентность аппаратных процессоров и интерпретаторов имеет важные последствия для организации компьютера и проектирования компьютерных систем. После того как разработчик выбрал машинный язык (Я) для нового компьютера, он должен решить, разрабатывать ли ему процессор, который будет выполнять программы на языке Я, или написать специальную программу для интерпретации программ на том же языке. Если он решить написать интерпретатор, ему потребуется разработать аппаратное обеспечение для исполнения этого интерпретатора. Возможны такие гибридные конструкции, когда часть команд выполняется аппаратным обеспечением, а часть интерпретируется.

Интерпретатор разбивает команды на более мелкие (элементарные). В результате машина, предназначенная для исполнения интерпретатора, может быть гораздо проще по строению и дешевле, чем процессор, выполняющий программы без интерпретации. Такая экономия особенно важна при большом количестве сложных команд с различными параметрами. В сущности, экономия проистекает из самого факта замены аппаратного обеспечения программой (интерпретатором), тогда как создание копий программного продукта обходится дешевле, чем создание копий аппаратных элементов.

Первые компьютеры поддерживали небольшое количество команд, и эти команды были простыми. Однако разработка более мощных компьютеров привела, помимо всего прочего, к появлению более сложных команд. Вскоре разработчики поняли, что при наличии сложных команд программы выполняются быстрее, хотя выполнение каждой отдельной команды занимает больше времени. (В качестве примеров таких сложных команд можно назвать выполнение операций с плавающей точкой, обеспечение доступа к элементам массива ит.п.) Если обнаруживалос, что пара тех или иных команд часто выполняется последовательно, нередко вводилась новая команда, заменяющая эти две.

Сложные команды оказались лучше ещё и потому, что некоторые операции иногда перекрывались. Подобные операции могли выполняться параллельно, но для этого требовалась дополнительная аппаратура. Для дорогих компьютеров с высокой производительностью приобретение такого дополнительного аппаратного обеспечения было вполне оправданным. Таким образом, у дорогих компьютеров было гораздо больше команд, чем у дешевых. Однако растущая стоимость разработки и требования совместимости команд привели к тому, что сложные команды стали использоваться и в дешевых компьютерах, хотя там во главу угла ставилась стоимость, а не быстродействие.

К концу 50-х годов компания IBM, которая лидировала тогда на компьютерном рынке, решила, что производство семейства компьютеров, каждый из которых выполняет одни и те же команды, выгоднее и для самой компании, и для покупателей. Чтобы охарактеризовать этот уровень совместимости, компания IBM ввела термин **архитектура**. Новое семейство компьютеров должно было иметь единую архитектуру и много разных моделей, отличающихся по цене и скорости, но «умеющих» выполнять одни и те же программы. Но как построить дешевый компьютер, который сможет выполнять все сложные команды, предназначенные для высокоэффективных дорогостоящих машин?

Решением проблемы стала интерпретация. Эта технология, впервые предложенная Уилксом в 1951 году, позволяла разрабатывать простые дешевые компьютеры, которые, тем не менее, могли выполнять болшое количество команд. В результате компания IBM создала архитектуру System/360, семейство совместимых компьютеров, различающееся по цене и производительности почти на три порядка. Аппаратное обеспечение, позволяющее работать без интерпретации, использовалось только в самых дорогих моделях.

Простые компьютеры с интерпретаторами команд имели свои достоинства. Наиболее важными среди них являлись:

- возможность исправлять неправильно реализованные команды «на месте» или даже компенсировать ошибки аппаратного обеспечения на уровне аппаратного обеспечения;
- возможность добавлять новые команды при минимальных затратах, причем при необходимости уже после покупки компьютера;
- возможность (благодаря структурированной организации) разработки, проверки и документирования сложных команд.

В 70-е годы компьютерный рынок быстро разрастался, новые компьютеры могли выполнять все больше и больше функций. Вследствие повышенного спроса на дешевые компьютеры предпочтение отдавалось компьютерам с интерпретаторами. Возможность разрабатывать аппаратное обеспечение с интерпретатором для определенного набора команд привела к появлению дешевых процессоров. Полупроводниковые технологии быстро развивались, низкая стоимость брала верх над высокой производительностью, и интерпретаторы стали применяться при разработке компьютеров все шире и шире. Интерпретация использовалась практически во всех компьютерах, выпущенных в 70-е годы, от мини компьютеров до самых больших машин.

К концу 70-х годов интерпретаторы стали применяться практически во всех моделях, кроме самых дорогих машин с очень высокой производительностью (например, Cray-1 и компьютеров серии Control Data Cyber). Интерпретаторы обеспечивали реализацию сложных команд без использования дорогостоящей аппаратуры, поэтому разработчики могли

вводить все более и более сложные команды, а также (и даже в особенности) расширять способы определения операндов.

Эта тенденция достигла своего апогея в компьютере VAX, разработанном компанией DEC; у него было несколько сот команд и более 200 способов определения операндов в каждой команде. К несчастью, архитектура VAX с самого начала ориентировалась на интерпретацию, а производительности уделялось мало внимания. Это привело к появлению большого количества второстепенных команд, которые сложно было выполнять непосредственно. Данное упущение стало фатальным как для VAX, так и для его производителя (компании DEC). Компания Compaq купила DEC в 1998 году (правда, тремя годами позже сама компания Compaq вошла в структуру Hewlett-Packard).

Хотя самые первые 8-разрядные микропроцессоры были очень простыми и поддерживали небольшой набор команд, к концу 70-х годов даже они стали разрабатываться с ориентацией на интерпретаторы. В этот период основной проблемой для разработчиков стала возрастающая сложность микропроцессоров. Главное преимущество интерпретации заключалось в том, что можно было разработать очень простой процессор, а все самое сложное реализовать с помощью интерпретатора. Таким образом, вместо разработки сложной аппаратуры требовалась разработка сложного программного обеспечения.

Успех системы Motorola 68000 с большим набором интерпретируемых команд и одновременный провал компьютера Zilog Z8000, у которого был столь же обширный набор команд, но не было интерпретатора, продемонстрировали все преимущества интерпретации при разработке новых машин. Этот успех был довольно неожиданным, учитывая, что компьютер Z80 (предшественник Zilog Z8000) пользовался большей популярностью, чем Motorola 6800 (предшественник Motorola 68000). Конечно, важную роль сдесь сыграли и другие факторы – например то, что компания Motorola много лет занималась производством микросхем, а торговая марка Zilog принадлежала Exxon – крупной нефтяной компании. Еще один фактор в пользу интерпретации – существование быстродействующих постоянных запоминающих устройств для хранения интерпретаторов (так называемых командных ПЗУ). Предположим, что для выполнения обычной интерпретируемой команды интерпретатору компьютера Motorola 68000 нужно выполнить 10 команд (они называются **микрокомандами**), по 100 нс на каждую, и произвести два обращения к оперативной памяти, по 500 нс на каждое. Общее время выполнения команды составит, следовательно, 2000 нс – всего лишь в два раза больше, чем в лучшем случае заняло бы непосредственное выполнение этой команды без интерпретации. А если бы не было специального быстродействующего постоянного запоминающего устройства, выполнение этой команды заняло бы целых 6000 нс. Шестикратное возрастание времени выполнения вынести намного сложнее.

1.3. МИКРОСХЕМЫ ПАМЯТИ

Преимущество памяти, изображенной на рисунке 1.4, состоит в том, что подобная структура применима при разработке памяти большого объема. На рисунке показана схема 4X3 (для 4-х слотов по 3 бита каждое). Чтобы расширить её до размеров 4x8, нужно добавить ещё 5 колонок триггеров по 4 триггера в каждой, а также 5 входных и 5 выходных линий. Чтобы перейти от схемы 4x3 к схеме 8x3, требуется добавить ещё четыре ряда триггеров по три триггера в каждом, а также адресную линию A_2. При такой структуре число слов в памяти должно быть степенью двойки для максимальной эффективности, а число битов в слове может быть любым. Технология изготовления интегральных схем идеально соответствует регулярной структуре микросхем памяти. С развитием технологии число битов, которое можно разместить в одной микросхеме, постоянно растет, обычно в два раза каждые 18 месяцев (закон Мура). С появлением больших микросхем маленькие микросхемы не всегда сразу устаревают, поскольку всегда существует компромисс между ёмкостью, быстродействием, мощностью, ценой и удобством сопряжения. Обычно самые большие современные микросхемы пользуются огромным спросом и, следовательно, стоят дороже в расчете за один бит, чем микросхемы небольшого размера.

При любом объёме памяти существуют несколько вариантов организации микросхемы. На рисунке 1.4 показаны две возможные структуры микросхемы ёмкостью 4 Мбит: 512 К x 8 и 4096 К x 1 (размеры микросхемы памяти обычно даются в битах, а не в байтах, поэтому здесь мы будем придерживаться этого соглашения). На рисунке 1.4 а можно видеть 19 адресных линий для обращения к одному из 2^{19} байт и 8 линий данных для загрузки или хранения выбранного байта.

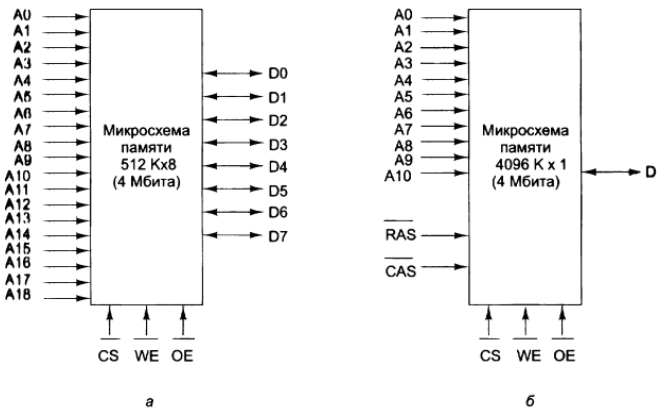


Рисунок 1.4 – Два способа организации памяти объемом 4 Мбит

Сделаем небольшое замечание по поводу терминологии. На одних выводах высокое напряжение вызывает какое-либо действие, на других остается низкое напряжение. Чтобы избежать путаницы, мы будем употреблять термин **установить сигнал**, когда вызывается какое-то действие, вместо того чтобы говорить, что напряжение повышается или понижается. Таким образом, для одних выводов установка сигнала означает установку единицы, для других – установку нуля. Названия выводов, которые устанавливаются в 0, содержат сверху черту. То есть сигнал CS – это единица, сигнал $\overline{CS}$ – ноль. Противоположный термин **сбросить**.

А теперь вернемся к нашей микросхеме. Поскольку обычно компьютер содержит много микросхем памяти, нужен сигнал для выбора необходимой микросхемы, такой, чтобы нужная нам микросхема реагировала на вызов, а остальные нет. Сигнал $\overline{CS}$ (Chip Select – выбор элемента памяти) используется именно для этой цели. Он устанавливается, чтобы запустить микросхему. Кроме того, нужен способ, чтобы отличать считывания от записи. Сигнал $\overline{WE}$ (Write Enable – разрешение записи) указывает на то, что данные должны записываться, а не считываться. Наконец, сигнал $\overline{OE}$ (Output Enable – разрешение вывода) устанавливается для выдачи выходных сигналов. Когда этого сигнала нет, выход отсоединяется от остальной части схемы.

На рисунке 1.4, б используется другая схема адресации. Микросхема представляет собой матрицу размером 2048 x 2048 однобитных ячеек, что составляет 4 Мбит. Чтобы обратить к микросхеме, сначала нужно выбрать строку. Для этого 11-разрядный номер этой строки подается на адресные выводы. Затем устанавливается сигнал $\overline{RAS}$ (Row Address Strobe – строб адреса строки). После этого на адресные выводы подается номер столбца и устанавливается сигнал CAS (Column Address Strobe – строб адреса столбца). Микросхема реагирует на сигнал, принимая или выдавая один бит данных.

Большие микросхемы памяти часто производятся в виде матриц размером m x n, обращение к которым происходит по строкам и столбцам. Такая организация памяти сокращает число необходимых выводов, но, с другой стороны, замедляет обращение к микросхеме, поскольку требуется два цикла адресации: один для строки, другой для столбца. Потеря скорости отчасти компенсируется тем, что в некоторых микросхемах возможна передача адреса строки с последующей передачей нескольких адресов столбцов для обращения к последовательным битам строки.

Много лет назад самые большие микросхемы памяти обычно были устроены так, как показано на рисунке 1.4, б. Поскольку размер слов увеличился от 8 до 32 бит и выше, использовать подобные микросхемы стало неудобно. Чтобы из микросхем 4096 К x 1 построить память с 32-разрядными словами, требуется 32 микросхемы, работающие параллельно. Эти 32 микросхемы имеют общий объем по крайней мере 16 Мбайт. Если

использовать микросхемы 512 К х 8, то потребуется всего 4 микросхемы, но при этом объем памяти составит 2 Мбайт. Чтобы не возиться с 32 микросхемами, большинство производителей выпускают семейства микросхем с длиной слов 4, 8 и 16 бит. Ситуация с 64-разрядными словами, естественно, ещё хуже.

Примеры современных микросхем объемом 512 Мбит показаны на рисунке 1.5. В каждой такой микросхеме содержится четыре внутренних банка памяти по 128 Мбит; соответственно, для определения банка требуются две линии выбора банка. На микросхеме 32 М х 16, показанной на рисунке 3.30, а, 13 линий выделено для сигналов RAS, 10 для сигналов CAS и 2 линии для выбора банка. Взятые в целом, 25 сигналов обеспечивают возможность адресации 2^{25} внутренних 16-разрядных ячеек. На микросхеме 128 М х 4, изображенной на рисунке 1.5, б, для сигналов RAS выделено 13 линий, для CAS – 12 линий, для выбора банка – 2 линии.

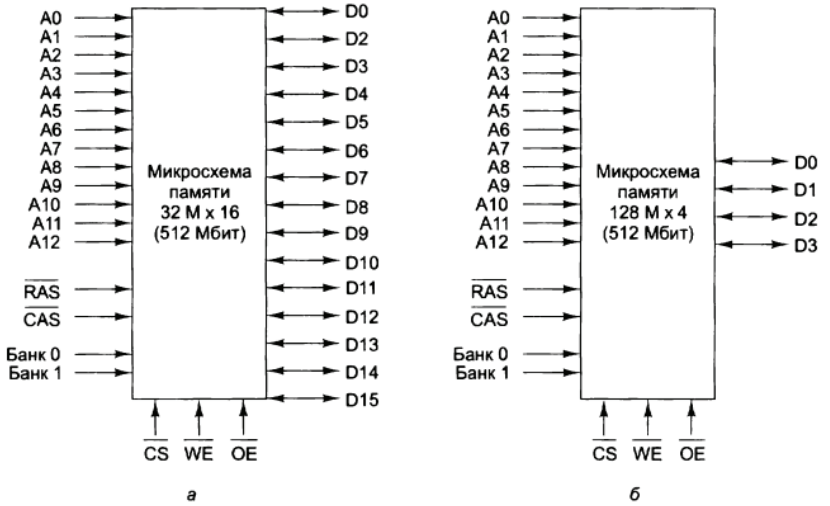


Рисунок 1.5 – Два способа организации микросхемы памяти объемом 512 Мбит

Таким образом, 27 сигналов делают возможной адресацию любой из 2^{25} внутренних 4-разрядных ячеек. Количества строк и столбцов в микросхемах определяются на основании инженерных факторов. Матрица не обязательно должна быть квадратной.

Эти примеры наглядно демонстрируют значимость двух не связанных друг с другом аспектов конструирования микросхем памяти. Первый касается ширины выхода (в битах) – иначе говоря, количества битов (1, 4, 8, 16 и пр.) в выходном сигнале. Второй аспект заключается в способе представления битов адреса; здесь есть два варианта: во-первых, биты

адресов могут быть представлены одновременно на разных выводах, во-вторых, может быть последовательное представление строк и столбцов – так, как показано на рисунке 1.5. Прежде чем приступать к проектированию микросхемы, специалист должен определиться с обоими аспектами.

1.3.1. Оперативное запоминающее устройство (ОЗУ)

Память, которая позволяет записывать и считывать информацию – называется ОЗУ (оперативное запоминающее устройство), или **RAM** (Random Access Memory – **оперативная память**). ОЗУ предназначено для хранения программ и промежуточных данных, используемых программами пользователя. Известны микросхемы ОЗУ статического (StaticRAM) и динамического (DynamicRAM) типа [3].

Микросхемы ОЗУ статического типа основаны на транзисторных триггерах. После записи данных в такие микросхемы, сохраняют их до тех пор, пока не будет выключено питание, не появится сигнал сброса или они не будут перезаписаны на новые. Статическое ОЗУ работает очень быстро. Обычно время доступа составляет несколько наносекунд. По этой причине статическое ОЗУ часто используется в качестве кэш-памяти второго уровня.

Микросхемы ОЗУ динамического типа основаны на миниатюрных конденсаторах. После записи данных такие микросхемы сохраняют их только несколько микросекунд после записи, что требует постоянного обновления информации, записанной в них (регенерации), что является их недостатком. Однако микросхемы ОЗУ динамического типа имеют значительно большую емкость и меньшую стоимость, что и обусловило их широкое распространение. Для микросхем, использующих примерно одну и ту же технологию, емкость микросхем динамических ОЗУ по грубым оценкам в 4-8 раз превышает емкость микросхем статических ОЗУ, но последние имеют в 8-16 раз меньшую длительность цикла и большую стоимость. По этим причинам в ОЗУ ПЭВМ, произведенной после 1985 г., использовались микросхемы динамических ОЗУ (для построения кэш-памяти при этом применяются микросхемы статических ОЗУ).

Существует несколько типов динамического ОЗУ. Самый древний тип, который все ещё используется, - **FPM** (Fast Page Mode – быстрый постраничный режим). Это ОЗУ представляет собой матрицу битов. Аппаратное обеспечение представляет адрес строки, а затем – адреса столбцов (мы описывали этот процесс, когда говорили об устройстве памяти, показанном на рисунке 1.4, б). Благодаря явно передаваемым сигналам память работает асинхронно по отношению к главному тактовому генератору системы.

FPM постепенно заместилось памятью EDO (Extended Data Output – память с расширенными возможностями вывода), которая позволяла обращаться к памяти ещё до того, как закончилось предыдущее обращение.

Такой конвейерный режим, хотя и не ускоряет доступ к памяти, повышает пропускную способность, позволяя получить больше слов в секунду.

Память типа FRM и EDO сохраняла актуальность в те времена, когда продолжительность цикла работы микросхем памяти не превышала 12 нс. Впоследствии, с увеличением быстродействия процессоров, сформировалась потребность в более быстрых микросхемах памяти, и тогда на смену асинхронным режимам FPM и EDO пришли синхронные динамические ОЗУ (Synchronous DRAM, SDRAM). Синхронное динамическое ОЗУ управляется от главного системного тактового генератора. Данное устройство представляет собой гибрид статического и динамического ОЗУ. Основное преимущество синхронного динамического ОЗУ состоит в том, что оно исключает зависимость микросхемы памяти от управляющих сигналов. ЦП сообщает памяти, сколько циклов следует выполнить, а затем запускает её. Каждый цикл на выходе даёт 4, 8 или 16 бит в зависимости от количества выходных строк. Устранение зависимости от управляющих сигналов приводит к ускорению передачи данных между ЦП и памятью.

Следующим этапом в развитии памяти SDRAM стала память DDR (Double Data Rate – передача данных с двойной скоростью). Эта технология предусматривает вывод данных как на фронте, так и на спаде импульса, вследствие чего скорость передачи увеличивается вдвое. Интерфейсы памяти DDR4 обеспечивают дополнительный прирост производительности по сравнению с DDR за счет повышения скорости шины памяти до 2666 МГц. На момент издания книги самые быстрые микросхемы DDR4 могли выдавать данные на скорости 21300 Мб/с (KTH-PL426S8/8G DDR4).

При записи отдельных байтов каждый байт располагается в ОЗУ по своему адресу. Слово при размещении в памяти занимает несколько смежных байтов. Слово характеризуется не всеми адресами занятых байтов, а только одним – адресом младшего байта слова.

1.3.2. Постоянное запоминающее устройство (ПЗУ)

Постоянное запоминающее устройство (ROM - Read Only Memory) предназначено для хранения служебных программ и подпрограмм, предназначенных для обеспечения возможности функционирования ПЭВМ. В настоящее время в ПЗУ записываются программа первоначального тестирования POST (Power On Self Test) и подпрограммы Базовой Системы Ввода Вывода (BIOS), обеспечивающие взаимодействие операционной системы и прикладных программ пользователя с ресурсами ПЭВМ.

Масочные ПЗУ (ROM). Запись в такие микросхемы ПЗУ осуществляется в процессе производства - хранящиеся данные определяются применяемой маской.

Разработчик микросхемы сообщает спецификации содержимого ПЗУ фирме-изготовителю. Поскольку заказ оказывается выгодным только для

партии в десятки тысяч микросхем, разработчик должен быть полностью уверен в том, что данные и программы безошибочны и не потребуют изменений.

Программируемые ПЗУ (Programmable ROM) с плавкими перемычками. Внутри микросхемы ПЗУ находится матрица из нихромовых или поликремниевых перемычек, которые можно расплавить, подав импульс тока с соответствующими параметрами. Программирование занимает значительное время, но сам прибор (программатор) оказывается простым и относительно недорогим. Довольно часто опытные образцы микропроцессорных систем поставляются с ППЗУ, которые после выявления ошибок и при переходе к массовому выпуску заменяются на ПЗУ [4].

Стираемые программируемые ПЗУ (Erasable PROM). Данные микросхемы ПЗУ обеспечивают многократные стирание их содержимого и программирование. В корпусе микросхемы ПЗУ сделано “окно”, через которое на матрицу запоминающих элементов может попадать свет. При экспонировании ультрафиолетовым светом в течение нескольких минут хранимые данные стираются. После этого микросхему ПЗУ можно снова запрограммировать.

После этого в микросхему с помощью программатора импульсами тока можно записать новую информацию (процесс программирования длится несколько минут).

Стираемые ППЗУ удобно применять при мелкосерийном производстве и на этапе проектирования.

Стираемые программируемые ПЗУ обычно устроены так же, как статические ОЗУ. Например, микросхема 27С040 имеет структуру, которая показана на рисунке 1.5, а, а такая структура типична для статического ОЗУ. Интересно, что подобные «древние» микросхемы не вымирают. Становятся дешевле и используются в бюджетных продуктах, для которых критична стоимость. Сейчас одиночные микросхемы 27С040 можно купить дешевле $3, а при большом размере партии они обойдутся значительно дешевле.

Электрически стираемые программируемые ПЗУ (Electronically EPROM, **EEPROM).** В данных микросхемах ПЗУ содержимое можно стереть электрическими импульсами; при этом микросхемы ПЗУ не нужно вынимать из гнезд.

В то же время самые большие электронно перепрограммируемые ПЗУ в 64 раза меньше обычных стираемых ПЗУ, и работают они в два раза медленнее. Электронно перепрограммируемые ПЗУ не могут конкурировать с динамическими и статическими ОЗУ, поскольку работают в 10 раз медленнее, их емкость в 100 раз меньше, и они стоят гораздо дороже. Они используются только в тех ситуациях, когда необходимо сохранять информацию при выключении питания.

Более современный тип электронно-перепрограммируемого ПЗУ – **флеш-память**. В отличие от стираемого ПЗУ, которое стирается под воздействием ультрафиолетовых лучей, и от электронно-перепрограммироуемого ПЗУ, которое стирается по байтам, флеш-память стирается и записывается блоками. Многие изготовители производят небольшие печатные платы, содержащие до 64 Гбайт флеш-памяти. Они используются для хранения изображений в цифровых камерах и для других целей. Флеш-память постепенно начинает вытеснять диски, что будет грандиозным шагом вперед, учитывая время доступа в 50 нс. Флеш-память обеспечивает лучшее время доступа при более низком энергопотреблении; с другой стороны, стоимость одного бита флеш-памяти существенно выше, чем у дисков. Краткое описание различных типов памяти дано в таблице 1.1.

Таблица 1.1 – Характеристики различных типов памяти

Тип запоминающего устройства	Категория	Стирание информации	Изменение информации	Необходимость питания	Применение
SRAM	Чтение и запись	Электрическое	Да	Да	Кэш-память второго уровня
DRAM	Чтение и запись	Электрическое	Да	Да	Основная память (старые модели)
SDRAM	Чтение и запись	Электрическое	Да	Да	Основная память (старые модели)
ROM	Только чтение	Невозможно	Нет	Нет	Устройства большого объема
PROM	Только чтение	Невозможно	Нет	Нет	Устройства небольшого объема
EPROM	Преимущественно чтение	Ультрафиолетовый свет	Нет	Нет	Построение прототипов устройств
EEPROM	Преимущественно чтение	Электрическое	Да	Нет	Построение прототипов устройств
Флеш-память	Чтение и запись	Электрическое	Да	Нет	Цифровые камеры

1.4. МИКРОСХЕМЫ ПРОЦЕССОРОВ

Вооружившись информацией о микросхемах, тактовых генераторах и микросхемах памяти, мы можем сложить все составные части вместе и начать изучение целых систем [2]. В этом разделе сначала мы рассмотрим процессоры на цифровом логическом уровне, включая цоколевку (то есть значения сигналов на различных выводах). Поскольку центральные процессоры тесно связаны с шинами, которые они используют, мы также кратко изложим основные принципы разработки шин. В следующих разделах приводятся подробные примеры центральных процессоров, их шин и взаимодействий между ними.

Все современные процессоры помещаются на одной микросхеме, благодаря чему их взаимодействия с остальными частями системы становятся четко определенными. Каждая микросхема процессора содержит набор выводов, через которые происходит обмен информацией с внешним миром. Одни выводы передают сигналы от центрального процессора, другие принимают сигналы от других компонентов, третьи делают то и другое. Изучив функции всех выводов, мы сможем узнать, как процессор взаимодействует с памятью и устройствами ввода-вывода на цифровом логическом уровне.

Выводы микросхемы центрального процессора можно подразделить на три типа: адресные, информационные и управляющие. Эти выводы связаны с соответствующими выводами на микросхемах памяти и микросхемах устройств ввода-вывода через набор параллельных выводов (так называемую шину). Чтобы вызвать команду, центральный процессор сначала посылает в память адрес этой команды по адресным выводам. Затем он задействует одну или несколько линий управления, чтобы сообщить памяти, что ему нужно (например, прочитать слово). Память выдаст ответ, помещая требуемое слово на информационные выводы процессора и посылая сигнал о том, что это сделано. Когда центральный процессор получает этот сигнал, он считывает слово и выполняет вызванную команду.

Команда может требовать чтения или записи слов, содержащих данные. В этом случае весь процесс повторяется для каждого дополнительного слова. Как происходит процесс чтения и записи, мы подробно рассмотрим далее. А пока важно понять, что центральный процессор обменивается информацией с памятью и устройствами ввода-вывода, подавая сигналы на выводы и принимая сигналы на входы. Другого способа обмена информацией не существует.

Число адресных выводов и число информационных выводов – два ключевых параметра, которые определяют производительность процессора. Микросхема, содержащая m адресных выводов, может обращаться к 2^m ячейкам памяти. Обычно m равно 16,32 или 64. Микросхема, содержащая n информационных выводов, может считывать или записывать n-разрядное слово за одну операцию. Обычно n равно 8, 32

или 64. Центральному процессору с 8 информационными выводами понадобится 4 операции, чтобы считать 32-разрядное слово, тогда как процессор, имеющий 32 информационных вывода, может сделать ту же работу в рамках одной операции. Следовательно, микросхема с 32 информационными выводами работает гораздо быстрее, но и стоит гораздо дороже.

Помимо адресных и информационных выводов, каждый процессор содержит управляющие выводы. Эти выводы позволяют регулировать и синхронизировать поток данных к процессору и от него, а также выполнять другие функции. Все процессоры содержат выводы для питания (обычно +1,2 В или +1,5 В), заземления и синхронизирующего сигнала (меандра). Остальные выводы разнятся от процессора к процессору. Тем не менее управляющие выводы можно разделить на несколько основных категорий:

-управление шиной;
-прерывания;
-арбитраж шины;
-сигналы процессора;
-состояние;
-разное.

Далее мы кратко опишем каждую из этих категорий, а когда мы будем рассматривать микросхемы Intel Core i7, TI OMAP4430 и Atmel ATmega168, дадим более подробную информацию. Схема типичного центрального процессора, в котором используются эти типы сигналов, изображена на рисунке 1.6.

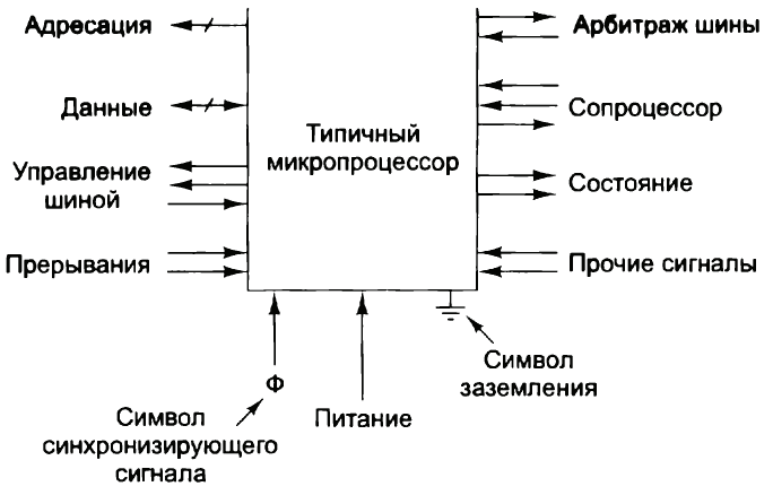


Рисунок 1.6 – Цоколёвка типичного центрального процессора

Выводы управления шиной по большей части представляют собой выходы из центрального процессора в шину (и, следовательно, входы микросхемы памяти и микросхем устройств ввода-вывода). Они позволяют сообщить, что процессор хочет считать информацию из памяти или записать информацию в память или сделать что-нибудь ещё.

Выводы прерывания – это входы из устройств ввода-вывода в процессор. В большинстве систем процессор может дать сигнал устройству ввода-вывода начать операцию, а затем приступить к какому-нибудь другому действию, пока устройство ввода-вывода выполняет свою работу. Когда устройство ввода-вывода её завершит, контроллер ввода-вывода посылает сигнал на один из выводов прерывания, чтобы прервать работу процессора и заставить его обслужить устройство ввода-вывода (например, проверить ошибки ввода-вывода). Некоторые процессоры содержат вывод для подтверждения сигнала прерывания.

Выводы арбитража шины нужны для регулировки потока информации в шине, то есть для исключения таких ситуаций, когда два устройства пытаются воспользоваться шиной одновременно. В плане арбитража центральный процессор считается просто одним из устройств.

Некоторые центральные процессоры могут работать с различными сопроцессорами (например, с графическими процессорами, процессорами для обработки вещественных данных и т.п.). Чтобы обеспечить обмен информацией между процессором и сопроцессором, используются специальные выводы.

Помимо этих выводов, у некоторых процессоров есть дополнительные выводы. Одни из них выдают или принимают информацию о состоянии, другие нужны для перезагрузки компьютера, третьи призваны обеспечить совместимость со старыми микросхемами устройств ввода-вывода.

Рассмотрим процессоры Intel Core i7, TI OMAP4430, Atmel ATmega128 и 1887ВЕ7Т на уровне аппаратного обеспечения.

1.4.1. Процессор Intel Core i7

Core i7 прямой потомок процессора 8088, который использовался в первой модели IBM PC [2]. Презентация Core i7 состоялась в ноябре 2008 года. Публике было представлено четырехпроцессорное ЦПУ с 731 млн транзисторов, частотой до 3,2 ГГц и шириной строки 45 нанометра. Понятие «ширина строки» обозначает ширину проводников между транзисторами (и одновременно определяет размер самих транзисторов). Чем меньше эта величина, тем больше транзисторов умещается на одной микросхеме. По сути, закон Мура прогнозирует способность инженеров к дальнейшему уменьшению ширины строки. Помимо прочего, уменьшение этой величины позволяет повысить тактовую частоту. Для сравнения, диаметр человеческого волоса составляет 20-100 мкм (причем светлые волосы тоньше темных).

Исходный выпуск архитектуры Core i7 базировался на архитектуре «Nahalem», однако новые версии Core i7 строятся на базе более новой архитектуры «Sandy-Bridge». Термином «архитектура» в этом контексте обозначается внутренняя организация центрального процессора, которой часто присваивается кодовое название. Обычно проектировщики компьютерных архитектур – люди серьезные, но иногда они придумывают для своих проектов очень остроумные кодовые названия. Например, архитектуры серии AMD K должны были разрушить позиции Intel на рынке процессоров для настольных систем, казавшиеся неуязвимыми. Для процессоров серии К было выбрано кодовое название «Kryptonite» название единственного вещества, которое могло повредить Супермену, остроумно намекало на доминирование Intel.

Новая версия Core i7 на базе архитектуры «Sandy-Bridge» увеличилась до 1,16 млрд. транзисторов. Она работает на скорости 3,5 ГГц с шириной строки 32 нанометра. Хотя Core i7 очень сильно отличается от процессора 8088 с его 20 000 транзисторов, он полностью совместим с 8088 и может выполнять двоичные программы, написанные для 8088 (не говоря уже о программах для всех процессоров, появившихся между Core i7 и 8088).

С точки зрения программного обеспечения Core i7 представляет собой 64-разрядную машину. Он поддерживает ту же стандартную промышленную архитектуру (ISA), что и процессоры 80386, 80486, Pentium, Pentium II, Pentium Pro, Pentium III и Pentium 4, включая те же регистры, те же команды и такую же встроенную систему обработки значений с плавающей точкой стандарта IEEE 754. Помимо этого, в Core i7 имеются новые команды, предназначенные в первую очередь для криптографических операций.

Core i7 является многоядерным процессором; таким образом, кремниевая подложка содержит несколько процессоров. Он продается с разным числом внутренних процессоров – от 2 до 6 (причем в ближайшем будущем их число должно увеличиться). Если программисты пишут параллельную программу с использованием потоков и блокировок, организация параллельного выполнения на нескольких процессорах обеспечит существенный выигрыш по скорости, поддерживается технология **гиперпоточности**, позволяющая нескольким аппаратным потокам быть активными одновременно. Гиперпоточность позволяет осуществлять аппаратное переключение потоков во время очень коротких задержек (например, промахов кэша). Программное переключение потоков может происходить только во время очень длинных задержек (например, сбоев страниц), поскольку для его реализации требуются сотни тактов.

На уровне микроархитектуры Core i7 базируется на архитектуре своих предшественников Core 2 и Core 2 Duo. Процессор Core i7 может выполнять до четырех команд одновременно, что позволяет рассматривать его как 4-кратную суперскалярную машину.

В процессорах Core i7 используется трехуровневый кэш. Каждый процессор Core i7 имеет 32-килобайтный кэш данных первого уровня (L1) и 32-килобайтный кэш команд первого уровня. У каждого ядра также имеется собственный 256-килобайтный кэш второго уровня (L2). Кэш второго уровня унифицирован, то есть позволяет хранить комбинацию команд и данных. Все ядра совместно используют один унифицированный кэш третьего уровня (L3), размер которого составляет от 4 до 15 Мбайт в зависимости от модели процессора. Трехуровневое кэширование значительно улучшает производительность процессора, но за счет возрастания стоимости кремниевых компонентов, так как у процессоров Core i7 общий объем КЭШа на одной подложке не может превышать 17 Мбайт.

Поскольку все микросхемы Core i7 содержат несколько процессоров с собственными КЭШами данных, при изменении одним из процессоров слова, размещенного в его приватном КЭШе, могут возникать трудности. Если, предположим, другой процессор попытается считать это слово из памяти, он получит устаревшее значение, поскольку между изменением слова и его записью в память проходит некоторое время. В целях поддержания согласованности данных в памяти каждый ЦП в мультипроцессорной системе **следит** за шиной памяти на предмет поиска запросов на кэшированные слова. В случае обнаружения подобного рода запроса процессор предоставляет необходимые данные до того, как память передаст их другим потребителям.

В системах с процессором Core i7 используются две внешние шины, обе они синхронные. Шина памяти DDR3 служит для доступа к главному динамическому ОЗУ; шина PCI Express – для взаимодействия с устройствами ввода-вывода. Высокопроизводительные версии Core i7 содержат несколько шин памяти и PCI Express, а также порт QPI (Quick Path Interconnect). Порт QPI связывает процессор с внешним мультипроцессорным соединением, что открывает возможность построения систем, в которых установлено более шести процессоров. Порт QPI отправляет и получает **запросы когерентного КЭШа**, а также другие управляюшие сообщения для мультипроцессорных систем – например, межпроцессорные прерывания.

Основная проблема Core i7, как и у всех современных процессоров для настольных систем, заключается в объемах потребляемой мощности и выделяемого тепла. Чтобы избежать повреждения кремниевых компонентов, необходимо отводить тепло от процессора сразу же после его образования. Процессоры Core i7 в зависимости от частоты и модели потребляют от 17 до 1502 Вт. Поэтому Intel пребывает в постоянном поиске новых решений, которые позволили бы урегулировать проблему тепловыделения.

Микросхемы Core i7 поставляются в квадратном корпусе LGA с длинной стороны 37,5 мм. На нижней плоскости микросхемы находится 1155

площадок, из которых 286 используются для подачи питания, а 360 заземляются в целях шумоподавления. Площадки размещены в виде матрицы 40x40, причем её центральный сегмент 17x25 не заполнен. Кроме того, по периметру ассиметрично пропущены еще 20 площадок, за счет чего исключается возможность неправильной установки микросхемы в гнезде. Физическую компоновку Core i7 иллюстрирует рис. 1.7.

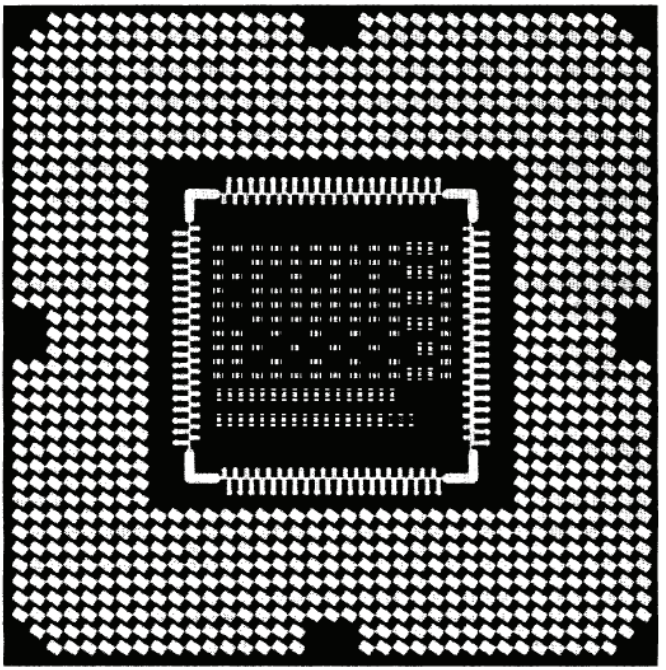


Рисунок 1.7 – Компоновка Core i7

Микросхема снабжена креплением для радиатора, который рассеивает тепло, и вентилятора, который охлаждает процессор.

В соответствии с законами физики всё, что выделяет большое количество тепла, должно потреблять большое количество энергии. В случае с портативным компьютером, который работает от батареи с ограниченным зарядом, потребление большого количества энергии нежелательно. Чтобы решить эту проблему, компания Intel нашла способ переводить центральный процессор в режим по снижению энергопотребления (состояние «сна»), если он не выполняет никаких действий, и вообще отключать его (вводить в состояние «глубокого сна»), если есть вероятность, что он не будет выполнять никаких действий некоторое

время. Всего предусмотрено пять различных состояний – от полной активности до глубокого сна. В промежуточных состояниях одни функции работают (например, функция слежения КЭШа, обработка прерываний), другие – отключаются. В состоянии глубокого сна значения кэш-памяти и регистров сохраняются, а тактовый генератор и все внутренние блоки отключаются. Выход из «глубокого сна» происходит по специальному аппаратному сигналу.

Цоколевка процессора Core i7

Из 1155 контактов Core i7 для сигналов используются 447, для питания (с различным напряжением) – 286, для «земли» – 360; ещё 62 зарезервированы на будущее. Для некоторых логических сигналов требуются два и более выводов (например, для запроса адреса памяти), поэтому существует только 131 вариант сигналов. Цоколевка Core i7 в несколько упрощенном виде представлена на рис.1.8. С левой стороны рисунка показано 5 основных групп сигналов шины памяти; с правой стороны расположены прочие сигналы.

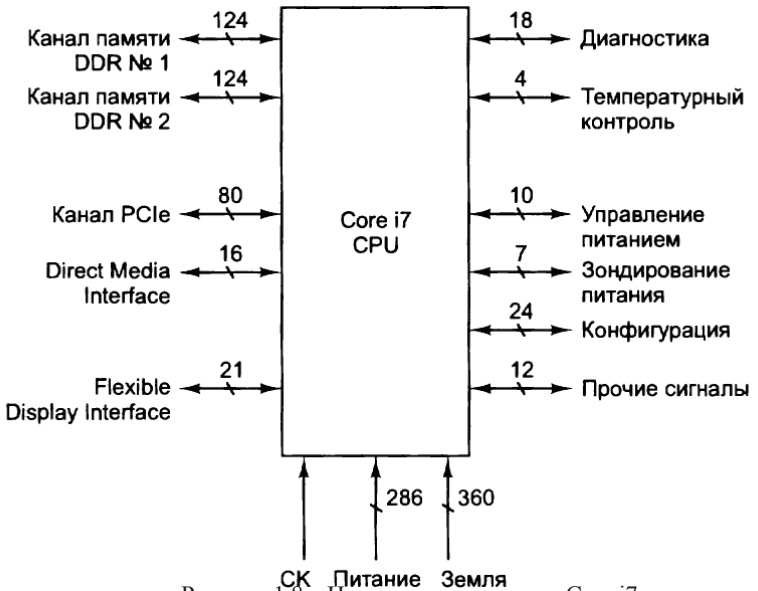


Рисунок 1.8 – Цоколевка процессора Core i7

Рассмотрим различные типы сигналов, начиная с сигналов шины. Первые два сигнала шины используются для взаимодействия с DDR3 – совместимой динамической памятью. Группа сигналов предоставляет банку динамической памяти адрес, данные, управляющую информацию и синхронизацию. Core i7 поддерживает два независимых канала DDR3, работающих на частоте шины 666 МГц с передачей данных по фронту и по

спаду сигнала; таким образом, возможно до 1333 млн взаимодействий в секунду. Интерфейс DDR3 является 64-разрядным, то есть два интерфейса DDR3 совместными усилиями ежесекундно предоставляют программам до 20 Гбайт данных.

Третья группа используется интерфейсом PCI, предназначенным для прямой связи периферийных устройств с центральным процессором Core i7. PCI Express высокоскоростной последовательный интерфейс, в котором каждый последовательный канал образует «тракт» обмена данными с периферийными устройствами. Core i7 поддерживает интерфейс x16, то есть может одновременно использовать 16 трактов с совокупной пропускной способностью 16 Гбайт/с. Хотя канал является последовательным, через PCI Express передаются самые разнообразные команды, включая команды чтения с устройства, записи, прерывания и настройки конфигурации.

Следующая группа сигналов образует интерфейс DMI (Direct Media Interface), используемый для связи процессора Core i7 с комплектным чипсетом. Интерфейс DMI схож с PCI Express, хотя и работает на половине скорости последнего, поскольку четыре тракта могут обеспечить скорость передачи данных только до 2,5 Гбайт/с. Чипсет содержит полнофункциональную поддержку дополнительных периферийных интерфейсов, обычно необходимую для высокопроизводительных систем с многочисленными устройствами ввода-вывода. Чипсет Core i7 состоит из микросхем P67 и ICH10. Микросхема P67 обеспечивает поддержку интерфейса SATA, USB, аудио, PCIe и флеш-памяти. Микросхема ICH10 обеспечивает поддержку наследных интерфейсов, включая интерфейс PCI и функциональность контроллера прерываний 8259A. Кроме того, ICH10 содержит много других схем: часы реального времени, таймеры событий и контроллеры прямого доступа к памяти (DMA). Существование таких микросхем значительно упрощает сборку полноценного персонального компьютера.

Core i7 может осуществлять прерывания тем же способом, что и 8088 (это требуется в целях совместимости), или использовать новую систему прерывания с устройством **APIC** (Advanced Programmable Interrupt Controller – **усовершенствованный программируемый контроллер прерываний**). Core i7 может действовать на любом из нескольких предустановленных напряжений, но процессор должен знать, на каком именно напряжении ему предстоит работать. Сигналы управления питанием используются для автоматического выбора напряжения источника питания, оповещения процессора о стабильности питания и ряда других родственных операций. С их же помощью осуществляется переход в различные состояния сна, которые, естественно, являются одними из инструментов управления питанием.

Несмотря на сложный механизм управления питанием, температура Core i7 может достигать очень высоких значений. Группа сигналов температурного контроля позволяют процессору оповещать окружающие

устройства об опасности перегрева. Сюда относится, например, сигнал, который выдается центральным процессором, если его внутренняя температура превышает 130°C (266°F). Хотя если температура центрального процессора превышает 130°C, он уже, вероятно, мечтает о выходе на пенсию и добросовестной службе в качестве нагревателя.

Впрочем, даже на таких запредельных температурах вам не придется беспокоиться о безопасности Core i7. Если внутренние датчики обнаруживают, что процессор вскоре перегреется, они запускают **терморегуляцию** – механизм, быстро снижающий выделение тепла за счет того, что процессор работает только на каждом N-ом такте. Чем выше значение N, тем сильнее замедляется процессор и тем быстрее он остывает. Конечно, за терморегуляцию приходиться расплачиваться снижением производительности системы. До изобретения терморегуляции в случае недостаточно эффективного охлаждения процессор перегорал. Доказательства этих «мрачных времен» температурного контроля можно найти на YouTube (поищите по строке «exploding CPU»). Видеоролик поддельный, но проблема настоящая.

Сигнал «Группа сигналов тактовой частоты» отвечает за определение частоты системной шины. Группа диагностических сигналов предназначена для тестирования и отладки систем согласно стандарту IEEE 1149.1 JTAG. Группа сигналов инициализации обслуживает загрузку (запуск) системы.

Сигнал СК используется процессором для генерирования различных тактовых импульсов с частотой, кратной или дробной по отношению к частоте системного генератора. Для этого применяется устройство, называемое **системой автоподстройки по задержке**, или **DLL** (Deley-Locked Loop).

Группа диагностических сигналов предназначена для тестирования и отладки систем согласно стандарту IEEE 1149.1 JTAG (Joint Test Action Group). Наконец, в группу «прочих сигналов» отнесены разнородные сигналы, используемые для разных специальных целей.

Конвейерный режим шины памяти процессора Core i7

Современные процессоры, такие как Core i7, предъявляют жесткие требования к динамической памяти. Они работают гораздо быстрее, чем медленная динамическая память может выдавать значения, причем эта проблема усугубляется, когда несколько процессов выдают одновременно запросы. Чтобы процессор не простаивал, необходима максимально возможная производительность памяти. По этой причине шина памяти процессора Core i7 DDR3 работает в конвейерном режиме, когда в шине происходят одновременно до 4-х операций..

Чтобы конвейерный режим стал возможным, запросы к памяти Core i7 состоят из трех этапов:

1). Фаза активизации (ACT) памяти «открывает» строку динамической памяти, делая ее готовой для последующих обращений.

2). В фазе чтения (READ) или записи (WRITE) могут происходить обращения к отдельным словам открытой строки динамической памяти или к последовательным словам текущей строки динамической памяти с использованием пакетного режима.

3). Фаза предзаряда (PCHRG) «закрывает» текущую строку динамической памяти и готовит память к следующей команде активизации.

Конвейерная работа с памятью процессора Core i7 основана на том, что динамическая память DDR3 на микросхеме состоит из нескольких банков. Банк представляет собой блок динамической памяти, к которому процессор может обращаться параллельно с другими банками, даже находящимися на той же микросхеме. Типичная микросхема динамической памяти DDR3 содержит до восьми банков. Впрочем, спецификация интерфейса DDR3 разрешает не более трех параллельных обращений для одного канала DDR3. Временная диаграмма на рисунке 1.9 показывает, как Core i7 выдает 3 обращения к трем разным банкам динамической памяти. Обращения полностью перекрываются, так что операции чтения на микросхеме динамической памяти выполняются параллельно. Связь между командами и последующими операциями на временной диаграмме обозначается стрелками.

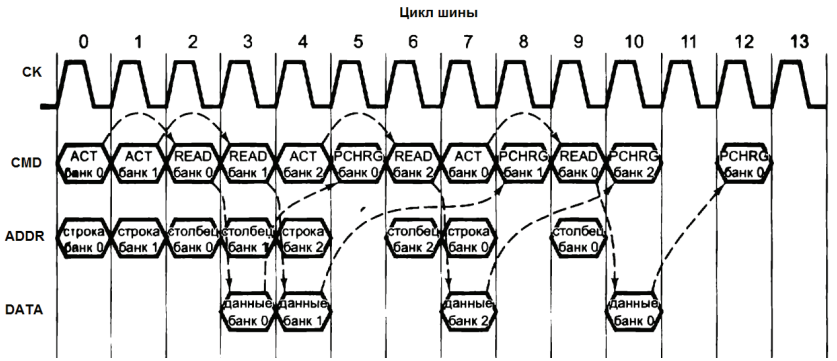


Рисунок 1.9 – Конвейерные обращения в памяти через интерфейс DDR3 процессора Core i7

Как видно из рис.1.9, интерфейс памяти DDR3 имеет четыре основных сигнальных канала: синхронизация шины (СК), команда шины (CMD), адрес (ADDR) и данные (DATA). Сигнал синхронизации шины СК управляет всей работой шины. Командный сигнал CMD указывает, какая операция запрашивается у динамической памяти. Команда ACT задает адрес строки динамической памяти, открытой сигналом ADDR. При выполнении команды READ адрес столбца динамической памяти задается

с использованием сигналов ADDR, а динамическая память выдает прочитанное значение спустя фиксированное время через сигналы DATA.

Наконец, команда PCHRG указывает банк, к которому применяется операция предзаряда, через сигналы ADDR. В нашем примере команда ACT должна предшествовать первой команде READ для того же банка на два цикла шины DDR3, а данные выдаются через один цикл после команды READ. Кроме того, операция PCHRG должна произойти по крайней мере на два цикла позже последней операции READ с тем же банком динамической памяти.

Параллелизм запросов памяти проявляется в перекрытии запросов READ к разным банкам динамической памяти. Первые два обращения READ к банкам 0 и 1 полностью перекрываются, производя результаты в циклах шины 3 и 4 соответственно. Обращение к банку 2 частично перекрывается с первым обращением к банку 1, и наконец, второе чтение из банка 0 частично перекрывается с обращением к банку 2.

Как Core i7 узнает, когда следует ожидать возвращения данных команды READ и когда можно выдавать новый запрос к памяти? Для этого он осуществляет полное моделирование внутренней деятельности каждой подключенной микросхемы DDR3. Соответственно он ожидает возвращения данных в правильно выбранном цикле и знает, что операцию предзаряда не следует начинать раньше чем через два цикла после последней операции чтения. Core i7 может прогнозировать все эти события, потому что интерфейс памяти DDR3 работает синхронно, так что все операции занимают четко определенное количество тактов шины DDR3. Даже при наличии всей этой информации простроение высокопроизводительного, полностью конвейерного интерфейса памяти DDR3 – нетривиальная задача, требующая применения многочисленных внутренних таймеров и детекторов конфликтов для реализации эффективной обработки запросов.

Компьютеры с процессорами Intel Core 10-го поколения с технологией Intel Thermal Velocity Boost (Intel TVB) и тактовой частотой до 5,3 ГГц, интеллектуальной кэш-памятью Intel емкостью 20 МБ, высокопроизводительными инструментами для оверлокинга, Ethernet адаптером Intel I225 и другими компонентами обеспечивают геймерам и творческим работникам преимущества высочайшей производительности [5].

Процессоры Intel® Core™ 10-го поколения для мобильных ПК имеют тактовую частоту до 5,3 ГГц2 в режиме Turbo, 8 ядер, 16 МБ кэш-памяти Intel® Smart Cache и поддержку оверклокинга3, а также поддерживают технологии Intel® Wi-Fi 6 (Gig+), Intel® Thermal Velocity Boost (Intel® TVB) и Intel® Dynamic Tuning Technology (Intel® DTT) для постоянной оптимизации работы системы. На базе этих процессоров работают самые производительные портативные системы для геймеров и создателей контента.

На основе максимально достижимой тактовой частоты 5,3 ГГц в режиме Turbo процессора Intel® Core™ i9-10980HK, что превосходит все другие продукты для мобильных ПК, доступные по состоянию на апрель 2020 года. Включает использование технологии Intel® Thermal Velocity Boost (Intel® TVB).

В апреле 2020 года компания Intel представила свои новые процессоры Comet Lake-S.

Comet Lake-S – это настольные процессоры, использующие всё ту же старую архитектуру и 14-нанометровый техпроцесс.

Основных изменений в Comet Lake два: новый сокет LGA 1200 (рис.1.10) и флагманские модели с 10 ядрами. Новый сокет «пригодится», когда выйдут процессоры Rocket Lake.

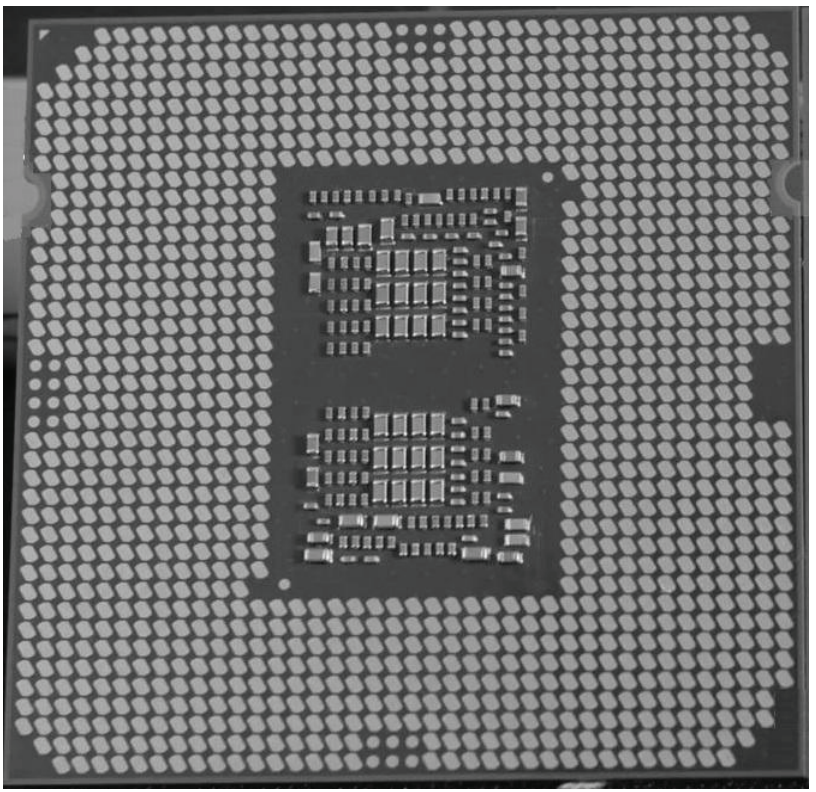


Рисунок 1.10 – Сокет LGA 1200

Наибольший интерес вызывает 10-ядерный флагман Core i9-10900K (количество ядер/потоков – 10/20, частоты – 3,7-5,2-5,3 ГГц), который будет напрямую конкурировать с 12-ядерным Ryzen 9 3900X, так как их цены очень близки. На стороне новинки Intel более высокие максимальные частоты, а в активе модели AMD имеются два дополнительных ядра и меньшее энергопотребление.

Все процессоры, кроме линейки Celeron, поддерживают Hyper-Threading, это значит, что производительность некоторых моделей относительно их предшественников существенно вырастет именно за счёт этой особенности. Стоит отметить поддержку контроллера Intel Ethernet Connection I225, обеспечивающего скорость до 2,5 Гбит/с, а также поддержку Wi-Fi 6 AX201.

1.4.2. Однокристальная система Texas Instruments OMAP5430

В качестве второго примера процессора возьмем однокристальную систему Texas Instruments (TI) OMAP5430. OMAP5430 реализует набор команд ARM, а основной областью его применения являются мобильные и встроенные системы: смартфоны, планшетные компьютеры, интернет-гаджеты. Однокристальная система включает широкий диапазон таких устройств, чтобы при объединении её с физической периферией (сенсорным экраном, флеш-памятью) формировалось полноценное компьютерное устройство.

Пятое поколение OMAP базируется на двухъядерном CPU ARM Cortex-A15 с дополнительными двумя ядрами ARM Cortex-M4, избавляющими ядра A15 от задач, не требующих большой вычислительной производительности, что способствует увеличению энергоэффективности, двух графических ядрах PowerVR SGX544MP и выделенном 2D графическом ускорителе от Vivante, многоканальной дисплейной подсистеме и процессоре цифровой обработки сигналов. Они поддерживают 20 и 24 мегапиксельные камеры для фронтальной и задней 3D HD видеокамер соответственно. Также поддерживается до 8 Гбайт двухканальной DDR3 памяти, работа с четырьмя 3D дисплеями, интерфейс 3D HDMI версии 1.4, 3 порта USB2.0 и SATA 2.0.

Также однокристальная система OMAP5430 содержит широкий спектр периферийных интерфейсов, включая сенсорные экраны и контроллеры клавиатуры, интерфейсы динамической и флеш-памяти, USB и HDMI. Фирма Texas Instruments опубликовала планы развития серии процессоров OMAP. В будущих архитектурах будет больше всего – больше ядер APM, графических процессоров и разнообразных периферийных устройств.

Трудно сравнивать CISC-микросхему (такую, как Core i7) с RISC-микросхемой (такой, как OMAP5430) на основании одной лишь тактовой частоты. Например, у двух ядер ARMv7 пиковая скорость выполнения достигает четырех команд на такт, в этом OMAP5430 почти сравнивается со суперскалярными процессорами у Core i7. Однако Core i7 быстрее

выполняет программы, потому что он содержит до восьми процессоров, тактовая частота которых в 2,5 раза выше (5,1 ГГц), чем у OMAP5430. Кажется, что OMAP5430 напоминает улитку, которая пытается обогнать ящерицу Core i7, однако улитка расходует намного меньше энергии и первой придет к финишу – особенно если заряд батареи у ящерицы не достаточно большой.

1.4.3. Микроконтроллер Atmel ATmega128

Core i7 и OMAP5430 – высокопроизводительные процессоры, разработанные для создания быстродействующих вычислительных устройств (Core i7 предназначен для настольных компьютеров, OMAP5430 – для мобильных систем). Наряду с ними существуют и другие компьютеры, даже более многочисленные – так называемые «встроенные системы» или микроконтроллеры.

Множество электронных устройств стоимостью более 100 долларов может содержать микроконтроллер. Телевизоры, сотовые телефоны, электронные секретари, микроволновые печи, лазерные принтеры, системы охранной сигнализации, слуховые аппараты, электронные игры и многие другие устройства управляются компьютером. Они приобрели популярность не из-за высокой производительности, а из-за низкой стоимости микроконтроллера.

Одним из популярных микроконтроллеров является ATmega128. ATmega128 – это универсальная микросхема, к которой очень легко подключить другие устройства. Её физическая компоновка показана на рисунке 1.11.

ATmega128 обычно поставляется в корпусе с 64 выводами. Особенность этой микросхемы по сравнению с двумя предыдущими примерами: у неё нет адресных линий и линий данных. Это связано с тем, что микросхема не предназначена для подключения к памяти – только к устройствам. Вся память (статическая и флэш-память) содержится в самой микросхеме, поэтому необходимость в отдельных адресных линиях и линиях данных отпадает (рис.1.12).

С точки зрения внутренней архитектуры ATmega128, как и OMAP5430 представляет собой однокристальную систему с широким набором внутренних устройств и памяти.

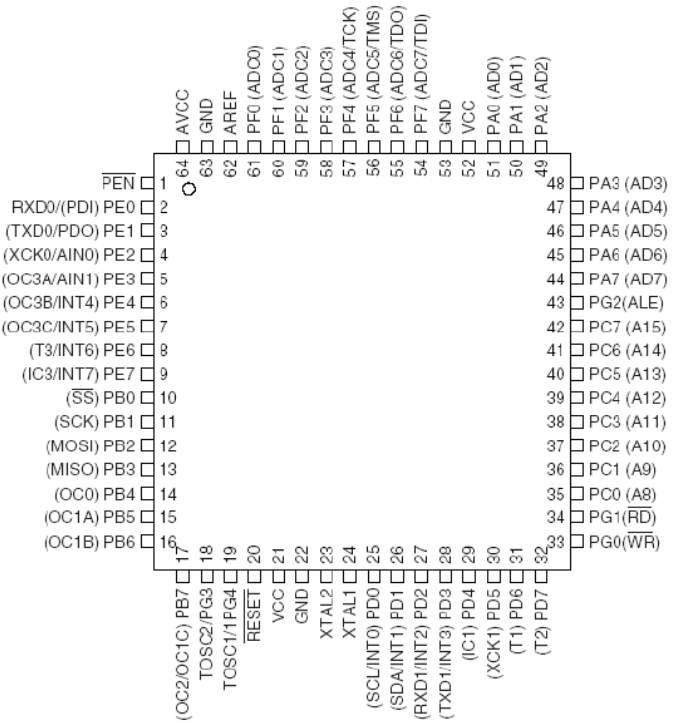


Рисунок 1.11 – Физическая компоновка микросхемы ATmega128

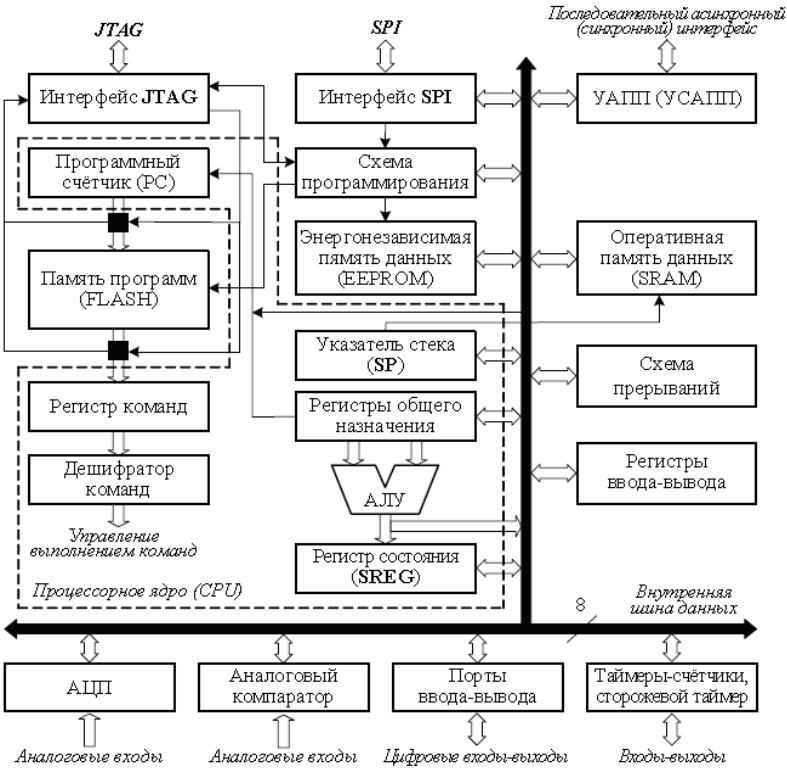


Рисунок 1.12 – Архитектура микроконтроллеров семейства AVR

Вместо адресных линий и линий данных ATmega128 содержит 53 порта цифрового ввода-вывода, 8 линий портов A, B, C, D, E и F и 5 линий порта G. Линии цифрового ввода-вывода предназначены для подключения периферийных устройств ввода-вывода, есть возможность программно задать режим работы каждой линии (ввод или вывод). При использовании, например, в инкубаторе одна цифровая линия может получать ввод от датчика температуры, другая – выдавать выходной сигнал для включения и выключения обогревателя, еще одна линия включает вентилятор для увеличения влажности.

ATmega 128 также содержит: 128 кбайт программируемой флеш-памяти, 4 кбайт ППЗУ, 4 кбайт статического ОЗУ, 32 универсальных рабочих регистра, счетчик реального времени (RTC), четыре гибких таймера-счетчика с режимами сравнения и ШИМ, 2 УСАП, двухпроводной последовательный интерфейс, 8-канальный 10-разрядный АЦП,

программируемый сторожевой таймер с внутренним генератором, последовательный порт SPI, интерфейс JTAG совместимый со стандартом IEEE 1149.1, который также используется для доступа к встроенной системе отладки и для программирования, и шесть программно выбираемых режимов уменьшения мощности. Режим холостого хода останавливает ЦПУ, но при этом поддерживая работу статического ОЗУ, таймеров-счетчиков, SPI-порта и системы прерываний.

Микроконтроллер производится по технологии высокоплотной энергонезависимой памяти компании Atmel.

ATmega 128 поддерживается полным набором программных и аппаратных средств для проектирования, в том числе: Си-компиляторы, макроассемблеры, программные отладчики/симуляторы, внутрисистемные эмуляторы и оценочные наборы.

1.4.4. Микроконтроллер 1887ВЕ7Т

Микроконтроллер 1887ВЕ7Т разработан в АО «НИИЭТ» (г. Воронеж) и является аналогом ATmega128 [6]. Данный микроконтроллер предназначен для использования в военной технике. Внешний вид микроконтроллера показан на рисунке 1.13.

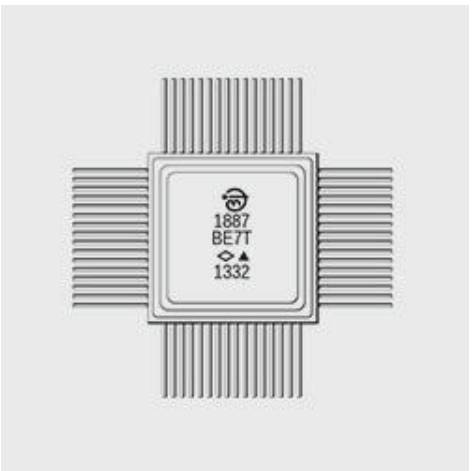


Рисунок 1.13 – Микроконтроллер 1887ВЕ7Т

Микроконтроллер имеет 8-разрядную RISC архитектура, тактовую частоту 16 МГц, напряжение питания 5В±10%, встроенный двухцикловый умножитель 8х8 бит.

Устройства памяти:

- внутрисистемная флеш-память программ типа EEPROM 128 Кбайт с поддержкой режима самопрограммирования;
- ЭСППЗУ данных 4 Кбайт;

- ОЗУ 4 Кбайт;
- объем адресуемой внешней памяти данных 64 Кбайт;
- программируемая защита кода программы.

Интерфейсы и порты:

- 53 программируемых линий ввода/вывода;
- 2 программируемых последовательных порта USART;
- последовательный периферийный интерфейс SPI с возможностью внутрисхемого программирования;
- байт-ориентированный двухпроводной последовательный интерфейс TWI;
- интерфейс JTAG.

Периферийные устройства

- два 8-битных таймера/счетчика с раздельными предделителями, режимами сравнения и захвата;
- счетчик реального времени с отдельным генератором;
- два 8-разрядных канала ШИМ;
- шесть каналов ШИМ с программируемым разрешением от 2 до 16 бит;
- встроенный 10-битный 8-канальный АЦП;
- встроенный аналоговый компаратор;
- встроенный калиброванный RC-генератор;
- программируемый сторожевой таймер со встроенным генератором.

Специальные возможности

- количество режимов пониженного энергопотребления – 6;
- сброс при подаче питания и программируемая схема сброса при снижении напряжения питания.

Области применения:

- встроенные цифровые системы управления и обработки данных;
- авионика;
- средства радиосвязи и РЭБ;
- РЛС наземного, авиационного и морского базирования;
- промышленные и специальные системы автоматики.

Структурная схема микроконтроллера 1887ВЕ7Т показана на рисунке 1.14.

Перечень отличий 1887ВЕ7Т от Atmega128 приведен в таблице 1.2.

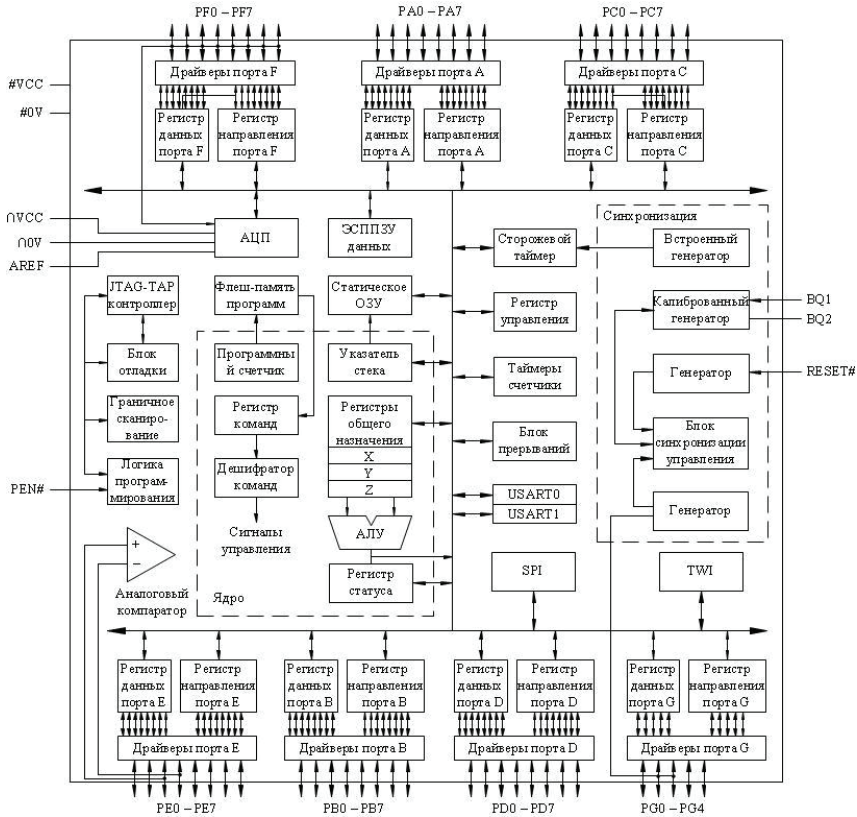


Рисунок 1.14 – Структурная схема микроконтроллера 1887ВЕ7Т

Таблица 1.2 – Перечень отличий ИМС от Atmega128

№	Описание	Комментарий
1	Тип использованной памяти: EEPROM	Стертое значение памяти - 00
2	Размер буфера страницы области RWW: 128 слов	
3	Размер буфера страницы области NRWW: 32 слова	
4	Длительность команды ChipErase	1,5 сек

Несмотря на эти отличия микроконтроллер может быть успешно запрограммирован с использованием среды AVRstudio посредсвом

программаторов STK500, STK600, AVRISP, JTAGICEII или среды Atmel Studio 6.

Для программирования данного микроконтроллера также можно использовать программно-аппаратные средства разработки АО «НИИЭТ» (рисунок 1.15).

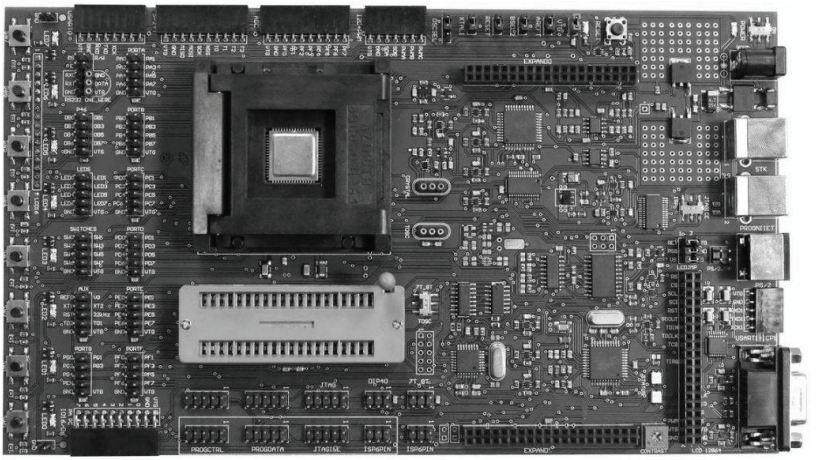


Рисунок 1.15 – Макетно-отладочная плата КФДЛ.301411.243 и USB-программатор КФДЛ.301411.247

Функционально-технические характеристики:

– USB интерфейс для связи с ПК;

– внутрисистемный последовательный интерфейс программирования ISP;

– высоковольтный параллельный интерфейс программирования;

– JTAG программатор и внутрисхемный отладчик;

– регулируемый источник питания 0,7 – 6,0 В;

– регулируемый источник опорного напряжения 0,7 – 6,0 В;

– кварцевый генератор в диапазоне 1 – 24 МГц;

– программируемый тактовый генератор в диапазоне 14,06 Гц – 3,686 МГц;

– контактное устройство для 64 -выводного корпуса 4203.64-2;

– контактное устройство для 40-выводного DIP-корпуса;

– разъемы портов ввода-вывода;

– расширительные разъемы и разъемы интерфейсов USART, TWI, SPI, One Wire, LCD12864, LCD1604;

– пользовательские светодиоды и кнопки;

– совместимость с программной средой КФДЛ.301411.247.

1.5. КОМПЬЮТЕРНЫЕ ШИНЫ

Шина – это несколько проводников, соединяющих несколько устройств [2]. Шины можно разделить на категории в соответствии с выполняемыми функциями. Они могут быть внутренними по отношению к процессору и служить для передачи данных в АЛУ и из АЛУ, а могут быть внешними по отношению к процессору и связывать процессор с памятью или устройствами ввода-вывода. Каждый тип шины обладает определенными свойствами и к каждому из них предъявляются определенные требования. В этом разделе мы сосредоточимся на шинах, которые связывают центральный процессор с памятью и устройствами ввода-вывода.

Первые персональные компьютеры имели одну внешнюю шину, которая называлась **системной**. Она состояла из нескольких медных проводов (от 50 до 100), которые встраивались в материнскую плату. На материнской плате на одинаковых расстояниях друг от друга находились разъемы для микросхем памяти и устройств ввода-вывода. Современные персональные компьютеры обычно содержат специальную шину между центральным процессором и памятью и по крайней мере еще одну шину для устройств ввода-вывода. На рисунке 1.16 изображена система с одной шиной памяти и одной шиной ввода-вывода.

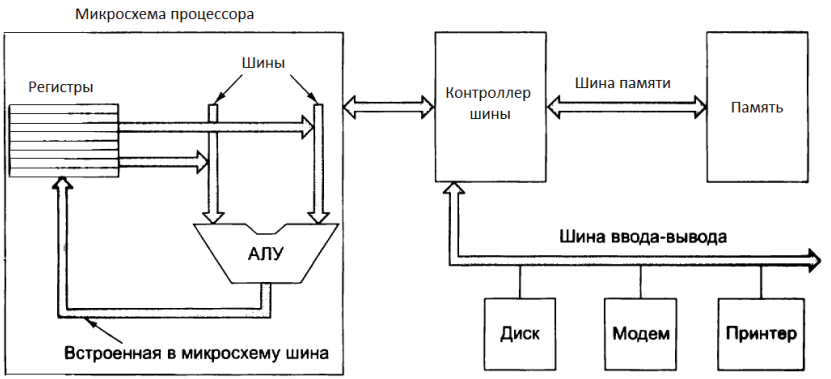


Рисунок 1.16 – Компьютерная система с несколькими шинами

В литературе шины обычно изображаются в виде жирных стрелок, как показано на этом рисунке. Разница между жирной стрелкой и нежирной стрелкой, через которую проходит короткая диагональная линия с указанием числа битов, небольшая. Когда тип всех битов одинаков, например все адресные или все информационные, рисуется обычная стрелка. Когда включаются адресные линии, линии данных и управления, используется жирная стрелка.

Хотя разработчики процессоров могут использовать любой тип шины для микросхемы, должны быть введены четкие правила о том, как работает шина; и все устройства, связанные с шиной, должны подчиняться этим правилам, чтобы платы, которые выпускаются сторонними производителями, подходили к системной шине. Эти правила называются **протоколом шины**. Кроме того, должны существовать определенные технические требования, чтобы платы от сторонних производителей подходили к направляющим для печатных плат и имели разъемы, соответствующие материнской плате механически, с точки зрения напряжений, синхронизации и т. д. Некоторые шины не имеют механических спецификаций, потому что они спроектированы для использования только с интегральными схемами – например, для соединения компонентов в однокристальных системах (SoC).

Существует целый ряд шин, широко используемых в компьютерном мире, например: Omnibus (PDP-8), Unibus (PDP-11), Multibus (8086), VME (оборудование для физической лаборатории), IBM PC (PC/XT), ISA (PC/AT), EISA (80386), Microchannal (PS/2), Nubus (Macintosh), PCI (различные персональные компьютеры), SCSI (различные персональные компьютеры и рабочие станции), Universal Serial Bus (современные персональные компьютеры), FireWire (бытовая электроника). Может быть, все стало бы намного проще, если бы все шины, кроме одной или двух, исчезли с поверхности земли. К сожалению, стандартизация в этой области кажется очень маловероятной, поскольку во все эти несовместимые системы уже вложено слишком много средств.

Начнем с того, как работают шины. Некоторые устройства, соединенные с шиной, являются активными и могут инициировать передачу информации по шине, тогда как другие являются пассивными и ждут запросов. Активное устройство называется **задающим**, пассивное – **подчиненным**. Когда центральный процессор требует от контроллера диска считать или записать блок информации, центральный процессор действует как задающее устройство, а контроллер диска – как подчиненное. Контроллер диска может действовать как задающее устройство, когда он командует памяти принять слова, которые считал с диска. Несколько типичных комбинаций задающего и подчиненного устройств перечислены в таблице 1.3. Память ни при каких обстоятельствах не может быть задающим устройством.

Таблица 1.3 – Примеры задающих и подчиненных устройств

Задающее устройство	Подчиненное устройство	Пример
Центральный процессор	Память	Вызов команд и данных
Центральный процессор	Устройство ввода-вывода	Инициализация передачи данных
Центральный процессор	Сопроцессор	Передача команды от процессора к сопроцессору
Устройство ввода-вывода	Память	Прямой доступ к памяти
Сопроцессор	Центральный процессор	Вызов сопроцессором операндов из центрального процессора

Двоичным сигналам, которые выдают устройства компьютера, часто не хватает мощности для активизации шины, особенно если она достаточно длинная и если к ней присоединено много устройств. По этой причине большинство задающих устройств шины обычно связаны с ней через микросхему, которая называется **драйвером шины** и по существу является цифровым усилителем. Сходным образом большинство подчиненных устройств связаны с шиной **приемником шины**. Для устройств, которые могут быть и задающим, и подчиненным устройством, используется **приемопередатчик**, или **трансивер**, **шины**.

Эти микросхемы, предназначенные для взаимодействия с шиной, часто являются устройствами с тремя состояниями, что дает им возможность отсоединяться, когда они не нужны. Иногда они подключаются через **открытый коллектор**, что дает сходный эффект. Когда одно или несколько устройств на открытом коллекторе требуют доступа к шине в одно и то же время, результатом является булева операция ИЛИ над всеми этими сигналами. Такое соглашение называется монтажным ИЛИ. В большинстве шин одни линии являются устройствами с тремя состояниями, а другие, которым требуется свойство монтажного ИЛИ, - открытым коллектором.

Как и процессор, шина имеет адресные, информационные линии и управляющие линии. Тем не менее между выводами процессора и сигналами шины может не быть взаимно однозначного соответствия. Например, некоторые процессоры содержат три вывода, которые выдают сигнал чтения из памяти или записи в память, чтения устройства с ввода-вывода, записи на устройство ввода-вывода или выполнения какой либо другой операции. Обычная шина может содержать одну линию для чтения из памяти, вторую – для записи в память, третью – для чтения с устройства ввода-вывода, четвертую – для записи на устройство ввода-вывода, и т. д. Тогда связывать процессор с такой шиной должна микросхема-кодер,

призванная преобразовывать 3-разрядный кодированный сигнал в отдельные сигналы, которые будут управлять линиями шины.

Проектирование и принципы действия шин – это достаточно сложные вопросы и по этому поводу написан ряд книг. Принципиальными вопросами в разработке являются ширина шины, синхронизация шины, арбитраж шины и функционирование шины. Все эти параметры существенно влияют на пропускную способность шины. В следующих подразделах мы рассмотрим каждый из них.

1.5.1. Ширина шины

Ширина (количество адресных линий) шины – самый очевидный параметр при проектировании [2]. Чем больше адресных линий содержит шина, тем к большему объему памяти может обращаться процессор. Если шина содержит n адресных линий, тогда процессор может использовать ее для обращения к 2^n различным ячейкам памяти. Для памяти большой ёмкости необходимо много адресных линий. Вроде бы всё просто.

Проблема заключается в том, что для широких шин требуется больше проводов, чем для узких. Они занимают больше физического пространства (например, на материнской плате) и для них нужны разъемы большего размера. Все эти факторы делают шину дорогостоящей. Следовательно, необходим компромисс между максимальным объемом доступной памяти и стоимостью системы. Система с шиной, содержащей 64 адресные линии, и памятью в 2^{32} байт будет стоить дороже, чем система с шиной 32 адресные линии, и такой же памятью в 2^{32} байт. Дальнейшее расширение не бесплатное.

Многие разработчики систем оказались недальновидными, что привело к неприятным последствиям. Первая модель IBM PC содержала процессор 8088 и 20-раззядную адресную шину (рис. 1.17,а). Шина позволяла обращаться к 1 Мбайт памяти.

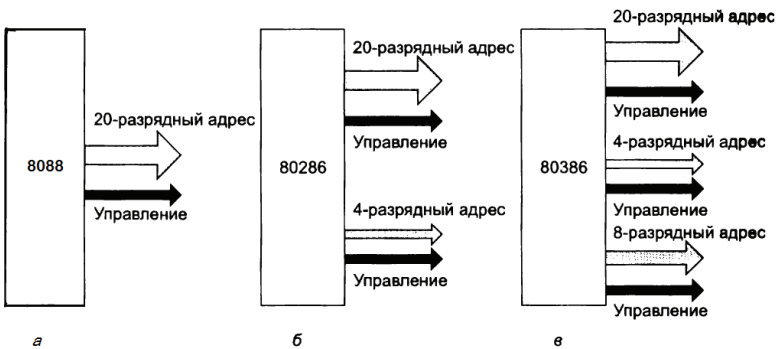


Рисунок 1.17 – Расширение адресной шины с течением времени

Когда появился следующий процессор (80286), компания Intel решила увеличить адресное пространство до 16 Мбайт, поэтому пришлось добавить ещё 4 линии (не нарушая изначальные 20 по причинам совместимости с предыдущими версиями), как показано на рисунке 1.17,б. К сожалению, пришлось также добавить управляющие линии для новых адресных линий. Когда появился процессор 80386, было добавлено еще 8 адресных линий и, естественно, несколько управляющих линий, как показано на рисунке 1.17,в. В результате получилась шина EISA. Однако архитектура получилась куда более запутанной, чем если бы с самого начала использовались 32 линии.

С течением времени увеличивается не только число адресных линий, но и число информационных линий, хотя это происходит по другой причине. Можно увеличить пропускную способность шины двумя способами: сократить время цикла шины (сделать большее количество передач в секунду) или увеличить ширину шины данных (то есть увеличить количество битов, передаваемых за цикл). Можно повысить скорость работы шины, но сделать это довольно сложно, поскольку сигналы на разных линиях передаются с разной скоростью (это явление называется **расфазировка шины**). Чем быстрее работает шина, тем больше расфазировка.

При увеличении скорости работы шины возникает ещё одна проблема: в этом случае она становится несовместимой с предыдущими версиями. Старые платы, разработанные для более медленной шины, не могут работать с новой. Такая ситуация не выгодна для владельцев и производителей старых плат. Поэтому обычно для увеличения производительности просто добавляются новые линии, как показано на рисунке 1.17. Как вы понимаете, в этом есть тоже свои недостатки. Компьютер IBM PC и его преемники, например, начали с 8 информационных линий, затем перешли к 16, потом к 32 линиям, и все это в одной и той же шине.

Чтобы обойти эту проблему, разработчики иногда отдают предпочтение **мультиплексной шине**. В этой шине нет разделения на адресные и информационные линии. В ней может быть, например, 32 линии и для адресов и для данных. Сначала эти линии используются для адресов, затем – для данных. Чтобы записать информацию в память, нужно сначала передать в память адрес, а потом – данные. В случае с отдельными линиями адреса и данные могут передаваться вместе. Объединение линий сокращает ширину и стоимость шины, но система работает при этом медленнее. Поэтому проектировщикам приходится взвешивать все за и против, прежде чем сделать выбор.

1.5.2. Синхронизация шины

Шины можно разделить на две категории в зависимости от их синхронизации [2]. **Синхронная шина** содержит линию, которая

запускается кварцевым генератором. Сигнал на этой линии представляет собой меандр с частотой обычно от 5 до133 МГц. Любое действие шины занимает целое число так называемых **циклов шины**. **Асинхронная шина** не содержит задающего генератора. Циклы шины могут быть произвольными и не обязательно одинаковыми для всех пар устройств.

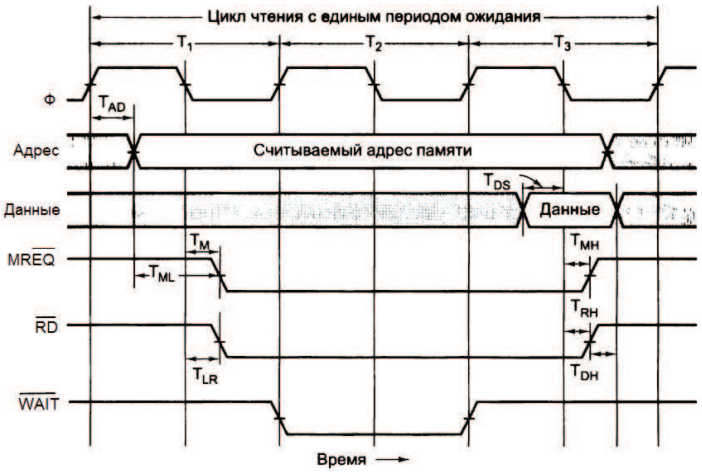
Синхронные шины

В качестве примера того, как работает асинхронная шина, рассмотрим временную диаграмму на рис.1.18. В этом примере мы будем использовать задающий генератор на 100 МГц, который дает цикл шины в 10 нс. Хотя может показаться, что шина работает медленно по сравнению с процессорами на 3 ГГц и выше. О причинах такой низкой скорости современных шин уже рассказывалось: к ним можно отнести такие технические проблемы, как расфазировка шины и необходимость обеспечения совместимости.

В своем примере мы предполагаем, что считывание информации из памяти занимает 15 нс с момента установки адреса. Как мы скоро увидим, для чтения слова понадобится три цикла шины. Первый цикл начинается на фронте отрезка T_1, а третий заканчивается на фронте отрезка T_4, как показано на рисунке 1.18. Отметим, что ни один из фронтов и спадов не нарисован вертикальным, потому что ни один электрический сигнал не может изменять свое значение за нулевое время. В нашем примере мы предполагаем, что для изменения сигнала требуется 1 нс. Генератор и линии адреса и данных, а также линии MREQ, RD, WAIT показаны в том же масштабе времени.

Начало T_1 определяется фронтом генератора. За время T_1 центральный процессор помещает адрес нужного слова на адресные линии. Поскольку адрес представляет собой не одно значение (в отличие от генератора), мы не можем показать его в виде одной линии на схеме. Вместо этого мы показали его в виде двух линий с пересечениями там, где этот адрес меняется.

После того как у адресных линий появляется возможность приобрести новое значение, устанавливаются сигналы MREQ и RD. Первый указывает, что осуществляется доступ к памяти, а не к устройству ввода-вывода, а второй – что осуществляется чтение, а не запись. Поскольку после установки адреса считывание информации из памяти занимает 15 нс (часть первого цикла), память не может передать требуемые данные за период T_2. Чтобы центральный процессор не ожидал поступления данных, память устанавливает сигнал WAIT в начале отрезка T_2. Это означает ввод **периодов ожидания** (дополнительных циклов шины) до тех пор, пока память не сбросит сигнал WAIT. В нашем примере вводится один период ожидания (T_2), поскольку память работает слишком медленно. В начале отрезка T_3, когда есть уверенность в том, что память получит данные в течение текущего цикла, сигнал WAIT сбрасывается.



а

Символ	Значение	Минимум (нс)	Максимум (нс)
T_{AD}	Задержка выдачи адреса		4
T_{ML}	Промежуток между стабилизацией адреса и установкой сигнала MREQ	2	
T_{M}	Промежуток между спадом синхронизирующего сигнала в цикле T_1 и установкой сигнала MREQ		3
T_{LR}	Промежуток между спадом синхронизирующего сигнала в цикле T_1 и установкой сигнала RD		3
T_{DS}	Период передачи данных до спада синхронизирующего сигнала	2	
T_{MH}	Промежуток между спадом синхронизирующего сигнала в цикле T_3 и сбросом сигнала MREQ		3
T_{RH}	Промежуток между спадом синхронизирующего сигнала в цикле T_3 и сбросом сигнала RD		3
T_{DH}	Период продолжения передачи данных с момента сброса сигнала RD	0	

б

Рисунок 1.18 – Временная диаграмма процесса считывания на синхронной шине (а); некоторые временные характеристики процесса считывания на синхронной шине (б)

Во время первой половины отрезка T_3 память помещает данные на информационные линии. На спаде отрезка T_3 центральный процессор стробирует (то есть считывает) информационные линии, сохраняя их значения во внутреннем регистре. Считав данные, центральный процессор сбрасывает сигналы MREQ и RD. В случае необходимости на следующем фронте может начаться ещё один цикл памяти. Эта последовательность может повторяться бесконечно.

Сигналы управления могут задаваться низким или высоким напряжением. Что является более удобным в каждом конкретном случае, должен решать разработчик, хотя, по существу, выбор произволен. Такую свободу выбора можно назвать «аппаратным» аналогом ситуации, при которой программист может представить свободные дисковые блоки в битовом отображении как в виде нулей, так и в виде единиц.

Пропускная способность шина PCIe версии 1.0 составляет 250 Мбайт/с.

В таблице 1.4 приведены данные по пропускной способности шины PCIe в зависимости от версии шины.

Таблица 1.4 – Пропускная способность PCIe

Год выпуска	Версия	Скорость передачи линии (ГТ/с)	Пропускная способность на x линий (Гбайт/с)				
			x1	x2	X4	X8	X16
2002	1.0	2,5	0,25	0,5	1,0	2,0	4,0
2007	2.0	5	0,5	1,0	2,0	4,0	8,0
2010	3.0	8	0,98	1,97	3,94	7,88	15,8
2017	4.0	16	1,969	3,94	7,88	15,75	31,5
2019	5.0	32	3,938	7,88	15,75	31,51	63,02
2021	6.0	64	7,877	15,75	31,51	63,02	126,03

Кроме PCE Express, существует ещё несколько высокоскоростных стандартизованных последовательных интерфейсов, таких как: RapidlO, StarFabric, HyperTransport, InfiniBand.

Также существуют специализированные шины для соединения северного и южного мостов, разработанные на базе физического протокола PCIe, но с другими логическими протоколами. В платформах Intel используется шина DMI, а в системах AMD с чипсетом AMD Fusion – шина UMI.

Асинхронные шины

Хотя использовать синхронные шины благодаря дискретным временным интервалам достаточно удобно, здесь всё же есть некоторые проблемы. Например, если процессор и память способны закончить передачу за 3,1 цикла, они вынуждены продлить её до 4,0 цикла, поскольку неполные циклы запрещены.

Ещё хуже то, что если однажды был выбран определенный цикл шины и в соответствии с ним разработана память и карта ввода-вывода, то в

будущем трудно делать технологические усовершенствования. Например, предположим, что через несколько лет после выпуска системы, изображенной на рисунке 1.18, появилась новая память с временем доступа не в 15, а в 8 нс. Это время позволяет избавиться от периода ожидания и увеличить скорость работы машины. А теперь представим, что появилась память с временем доступа в 4 нс. При этом улучшения производительности уже не будет, поскольку в данной разработке минимальное время чтения – 2 цикла.

Если синхронная шина соединяет ряд устройств, одни из которых работают быстро, а другие медленно, шина подстраивается под самое медленное устройство, а более быстрые не могут использовать свой потенциал полностью.

По этой причине были разработаны асинхронные шины, то есть шины без задающего генератора (рис. 1.19). Работа асинхронной шины не привязывается к генератору. Когда задающее устройство устанавливает адрес, сигнал MREQ и RD или любой другой требуемый сигнал, он выдает специальный синхронизирующий сигнал MSYN (Master SYNcronization). Когда подчиненное устройство получает этот сигнал, оно начинает выполнять свою работу настолько быстро, насколько это возможно. Когда работа заканчивается, подчиненное устройство выдает сигнал SSYN (Slave SYNchronization).

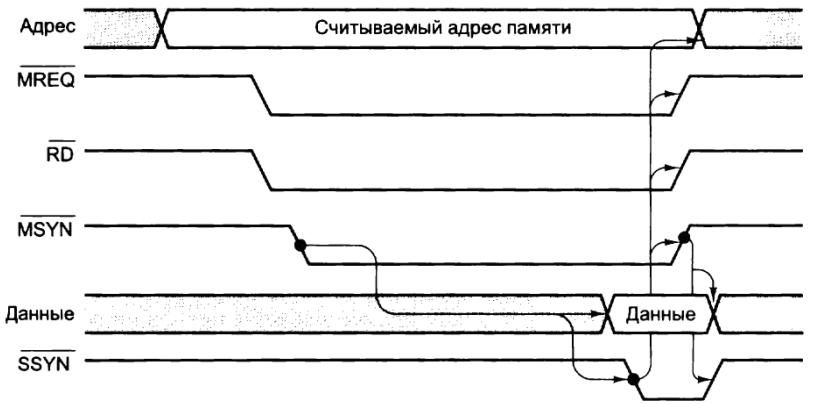


Рисунок 1.19 – Работа асинхронной шины

Сигнал SSYNсообщает задающему устройству, что данные доступны. Он фиксирует их, а затем сбрасывает адресные линии вместе с сигналами MREQ, RD и MSYN. Сброс сигнала MSYN означает для подчиненного устройства, что цикл закончен, поэтому устройство сбрасывает сигнал SSYN, и все возвращается к первоначальному состоянию, когда все сигналы сброшены.

Стрелочки на временных диаграммах асинхронных шин (а иногда и синхронных шин) показывают причину и следствие какого-либо действия (см. рис. 1.19). Установка сигнала MSYN приводит к включению информационных линий, а также к установке сигнала SSYN.

Установка сигнала SSYN, в свою очередь, вызывает отключение всех адресных линий, а также линий MREQ, RD и MSYN. Наконец, сброс сигнала MSYN вызывает сброс сигнала SSYN, и на этом процесс считывания заканчивается.

Набор таких взаимообусловленных сигналов называется **полным квитированием**. Здесь в сущности, наблюдается 4 события:

1). Установка сигнала MSYN.
2). Установка сигнала SSYN в ответ на сигнал MSYN.
3). Сброс сигнала MSYN в ответ на сигнал SSYN.
4). Сброс сигнала SSYN в ответ на сброс сигнала MSYN.

Разумеется, взаимообусловленность сигналов не является синхронной. Каждое событие вызывается предыдущим событием, а не импульсами генератора. Если какая то пара устройств (задающее и подчинённое) работает медленно, это никак не влияет на другую пару устройств, которая может работать гораздо быстрее.

Преимущества асинхронной шины очевидны, хотя на самом деле большинство шин являются синхронными. Дело в том, что синхронную систему построить проще, чем асинхронную. Центральный процессор просто выдает сигналы, а память просто реагирует на них. Здесь нет никакой причинно-следственной связи, и если компоненты выбраны удачно, все работает и без квитирования. Кроме того, в разработку синхронных шин вложено очень много ресурсов.

1.5.3. Арбитраж шины

До этого момента мы неявно предполагали, что существует только одно задающее устройство шины – центральный процессор [2]. В действительности микросхемы ввода-вывода могут становиться задающими устройствами при считывании информации из памяти и записи информации в память. Кроме того, они могут вызывать прерывания. Сопроцессоры также могут становиться задающими устройствами шины. Возникает вопрос: «Что происходит, когда задающим устройством шины становятся два или более устройств одновременно?». Чтобы предотвратить хаос, который может при этом возникнуть, нужен специальный механизм – так называемый **арбитраж шины**.

Арбитраж может быть централизованным или децентрализованным. Рассмотрим сначала централизованный арбитраж. Простой пример централизованного арбитража показан на рисунке 1.20, а. В данном примере один орбитр шины определяет, чья очередь следующая. Часто механизм арбитража встраивается в микросхему процессора, но иногда используется отдельная микросхема. Шина содержит одну линию запроса

(монтажное ИЛИ), которая может запускаться одним или несколькими устройствами в любое время. Арбитр не может определить, сколько устройств запрашивают шину. Он может определить только факт наличия или отсутствия запросов.

Когда орбитр обнаруживает запрос шины, он устанавливает линию предоставления шины. Эта линия последовательно связывает все устройства ввода-вывода (как в ёлочной гирлянде). Когда физически ближайшее к арбитру устройство получает сигнал предоставления шины, это устиройство проверяет, нет ли запроса шины. Если запрос есть, устройство пользуется шиной, но не распространяет сигнал предоставления дальше по линии. Если запроса нет, устройство передает сигнал предоставления шины следующему устройству. Это устройство тоже проверяет, есть ли запрос, и действует соответствующим образом в зависимости от наличия или отсутствия запроса. Передача сигнала предоставления шины продолжается до тех пор, пока какое-нибудь устройство не воспользуется предоставленной шиной. Такая система называется **системой последовательного опроса**. При этом приоритеты устройств зависят от того, насколько близка они находятся к арбитру. Ближайшее к арбитру устройство обладает наивысшим приоритетом.

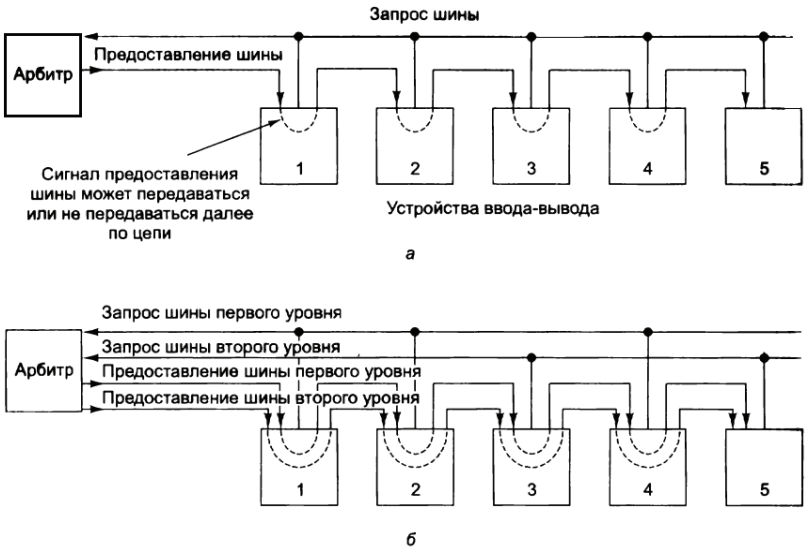


Рисунок 1.20 – Одноуровневый централизованный арбитраж шины с последовательным опросом (а); двухуровневый централизованный арбитраж (б)

Чтобы приоритеты устройств не зависели от расстояния до орбитра, в некоторых шинах поддерживается несколько уровней приоритета. На каждом уровне приоритета есть линия запроса шины и линия предоставления шины. На рис.1.20, б изображено 2 уровня (хотя в действительности шины обычно поддерживают 4, 8 или 16 уровней). Каждое устройство связано с одним из уровней запроса шины, причём, чем выше уровень приоритета, тем больше устройств привязано к этому уровню. На рис.1.20, б можно видеть, что устройства 1,2 и 4 обладают приоритетом уровня 1, а устройства 3 и 5 – приоритетом уровня 2.

Если одновременно запрашивается несколько уровней приоритета, арбитр предоставляет шину самому высокому уровню. Среди устройств одинакового приоритета реализуется система последовательного опроса. На рис.1.20, б видно, что в случае конфликта устройство 2 «побеждает» устройство 4, а устройство 4 «побеждает» устройство 3. Устройство 5 имеет низший приоритет, поскольку оно находится в самом конце самого нижнего уровня.

Следует заметить, что с технической точки зрения линия предоставления шины уровня 2 не обязательно должна последовательно связывать устройства 1 и 2, поскольку они не могут посылать на неё запросы. Однако гораздо проще провести все линии предоставления шины через все устройства, чем соединять устройства особым образом в зависимости от их приоритетов.

Некоторые арбитры содержат третью линию, которая устанавливается, как только устройство принимает сигнал предоставления шины, и получает шину в свое распоряжение. Как только эта линия подтверждения приема устанавливается, линии запроса и предоставления шины могут быть сброшены. В результате другие устройства могут запрашивать шину, пока первое устройство её использует. К тому моменту, когда закончится текущая передача, следующее задающее устройство уже будет выбрано. Это устройство может начать работу, как только будет сброшена линия подтверждения приема. С этого момента начинается следующий цикл арбитража. Такая структура требует дополнительной линии и большого количества логических схем в каждом устройстве, но зато при этом циклы шины используются рациональнее.

В системах, где память связана с главной шиной, центральный процессор должен конкурировать со всеми устройствами ввода-вывода практически на каждом цикле шины. Чтобы разрешить эту проблему, можно предоставить центральному процессору самый низкий приоритет. При этом шина будет предоставляться процессору только в том случае, если она не нужна ни одному другому устройству. Центральный процессор всегда может подождать, а устройства ввода-вывода должны получить доступ к шине как можно быстрее, чтобы не потерять данные. Например, диски, вращающиеся с высокой скоростью, не могут ждать. Во многих современных компьютерах для решения этой проблемы память помещается

на одну шину, а устройства ввода-вывода – на другую, поэтому им не приходится прерывать работу, чтобы предоставить доступ к шине.

Возможен также децентрализованный арбитраж шины. Например, компьютер может содержать 16 приоритетных линий запроса шины. Когда устройству нужна шина, оно устанавливает свою линию запроса. Все устройства отслеживают все линии запроса, поэтому в конце каждого цикла шины каждое устройство может определить, обладает ли оно в данный момент наивысшим приоритетом и, следовательно, разрешено ли ему пользоваться шиной в следующем цикле. Такой метод требует большого количества линий, но зато избавляет от потенциальных затрат ресурсов на использование арбитра. В этом случае число устройств ограничивается числом линий запроса.

При другом типе децентрализованного арбитража используются только три линии независимо от того, сколько устройств имеется в наличии (рис.1.21). Первая линия – монтажное ИЛИ. Она требуется для запроса шины. Вторая линия называется BUSY и означает занятость. Она запускается текущим задающим устройством шины. Третья линия служит для арбитража шины. Она последовательно соединяет все устройства. Начало цепи связано с источником питания с напряжением 5 В.

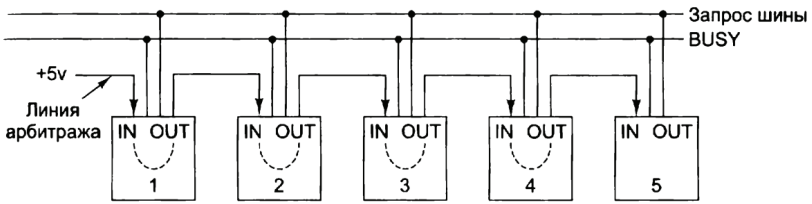


Рисунок 1.21 – Децентрализованный арбитраж шины

Когда шина не требуется ни одному из устройств, линия арбитража передает сигнал всем устройствам. Чтобы получить доступ к шине, устройство сначала проверяет, свободна ли шина и установлен ли сигнал арбитража IN. Если сигнал IN не установлен, устройство не может стать задающим устройством шины. В этом случае оно сбрасывает сигнал OUT. Если сигнал IN установлен, устройство также сбрасывает сигнал OUT, в результате чего следующее устройство не получает сигнала IN и, в свою очередь, сбрасывает сигнал OUT. Следовательно, все следующие по цепи устройства не получают сигнал IN и сбрасывают сигнал OUT. В результате остается только одно устройство, у которого сигнал IN установлен. А СИГНАЛ out СБРОШЕН. Оно становится задающим устройством шины, устанавливает линию BUSY и сигнал OUT, после чего начинает передачу данных.

Немного поразмыслив, можно обнаружить, что среди всех устройств, которым нужна шина, доступ к шине получает самое левое. Такая система напоминает ситему последовательного опроса, только в данном случае нет арбитра, поэтому она стоит дешевле и работает быстрее. К тому же не возникает проблем со сбоями арбитра.

1.5.4. Принцип работы шины

До этого момента мы обсуждали только обычные циклы шины, когда задающее устройство (обычно центральный процессор) считывает информацию из подчиненного устройства (обычно из памяти) или записывает в него информацию. Однако существуют ещё несколько типов циклов шины [2]. Давайте рассмотрим некоторые из них.

Обычно за раз передается одно слово. При использовании кэш-памяти (то есть 16 последовательных 64-разрядных слов). Однако часто передача блоками может быть более эффективна, чем такая последовательная передача информации. Когда начинается чтение блока, задающее устройство сообщает подчиненному устройству, сколько слов нужно передать (например, помещая общее число слов на информационные линии в период T_1). Вместо того чтобы выдать в ответ одно слово, задающее устройство выдает одно слово в течение каждого цикла до тех пор, пока не будет передано требуемое количество слов. На рис.1.22 изображена такая же схема, как и на рис.1.18, только с дополнительным сигналом BLOCK, который указывает, что запрашивается передача блока. В данном примере считывание блока из четырёх слов занимает 6 циклов вместо 12-ти.

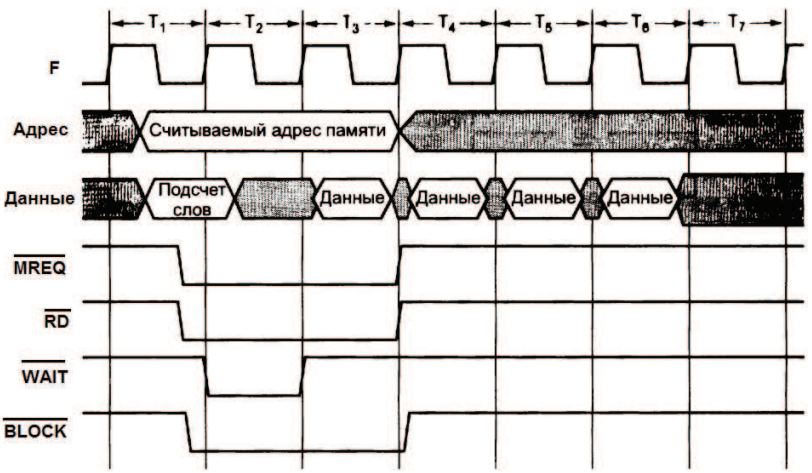


Рисунок 1.22 – Передача блока данных

Существуют также другие типы циклов шины. Например, если речь идет о системах с двумя или несколькими центральными процессорами на одной шине, нужно быть уверенным, что в конкретный момент только один центральный процессор может использовать определенную структуру данных в памяти. Чтобы упорядочить этот процесс, в памяти должна содержаться переменная, которая принимает значение 0, когда центральный процессор использует структуру данных, и 1, когда структура данных не используется. Если центральному процессору нужно получить доступ к структуре данных, он должен считать переменную, и если она равна 0, придать ей значение 1. Проблема заключается в том, что два центральных процессора могут считывать переменную на последовательных циклах шины. Если каждый процессор обнаружит, что переменная равна 0, а затем поменяет значение переменной на 1, как будто только он один использует эту структуру данных, то такая последовательность событий приведет к хаосу.

Чтобы не допустить подобную ситуацию, в мультипроцессорных системах предусмотрен специальный цикл шины, который дает возможность любому процессору считать слово из памяти, проверить и изменить его, а затем записать обратно в память; весь этот процесс происходит без освобождения шины. Такой цикл не дает возможности другим центральным процессорам использовать шину и, следовательно, мешать работе первого процессора.

Ещё один важный цикл шины – цикл обработки прерываний. Когда центральный процессор командует устройству ввода-вывода произвести какое-то действие, он ожидает прерывания после завершения работы. Для сигнала прерывания нужна шина.

Поскольку может сложиться ситуация, когда несколько устройств одновременно захотят выполнить прерывание, здесь имеют место те же проблемы разрешения конфликтных ситуаций, что и в обычных циклах шины. Чтобы избежать таких проблем, нужно каждому устройству приписать определенный приоритет и для распределения приоритетов поддерживать централизованный арбитраж. Для этих целей существует стандартный, широко используемый интерфейс прерываний. В компьютерах IBM PC и последующих моделях для этого служит микросхема Intel 8259A. Она изображена на рисунке 1.23.

До восьми контроллеров ввода-вывода могут быть непосредственно связаны с восемью входами IR_x (Interrupt Request – запрос прерывания) микросхемы 8259A. Когда любое из этих устройств решит произвести прерывание, оно запускает свою линию входа.

При активизации одного или нескольких входов контроллер 8259A выдает сигнал INT (INTerrupt – прерывание), который подается на соответствующий вход центрального процессора.

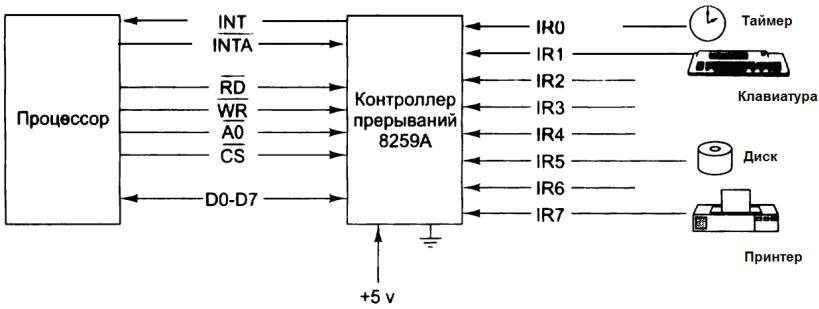


Рисунок 1.23 – Контроллер прерываний 8259A

Если центральный процессор способен обработать прерывание, он посылает микросхеме 8259A импульс через вывод INTA (INTerrupt Acknowledge – подтверждение прерывания).В этот момент микросхема 8259A должна определить, на какой именно вход поступил сигнал прерывания. Для этого она помещает номер входа на информационную шину. Эта операция требует особого цикла шины. Центральный процессор использует этот номер для обращения к таблице указателей, которую называют таблицей **векторов прерываний**, чтобы найти адрес процедуры обработки этого прерывания.

Микросхема 8259A содержит несколько регистров, которые центральный процессор может считывать и записывать, используя обычные циклы шины и выводы RD (ReaD – чтение), WR (Write – запись), CS (Chip Select – выбор элемента памяти) и A0. Когда программное обеспечение обработало прерывание и готово получить следующее, оно записывает специальный код в один из регистров, который вызывает сброс сигнала INT микросхемой 8259A, если не появляется другое прерывание. Регистры также могут записываться для того, чтобы перевести микросхему 8259A в один из нескольких режимов, и для выполнения некоторых других функций.

При наличии более 8 устройств ввода-вывода, микросхемы 8259A могут соединяться каскадом. В самой экстремальной ситуации все 8 входов могут быть связаны с выходами ещё 8 микросхем 8259A, соединяя до 64 устройств ввода-вывода в двухступенчатую систему обработки прерываний. Контроллер-концентратор ввода-вывода Intel ICH10 I/O, одна из микросхем чипсета Core i7, содержат два контроллера прерываний 8259A. Таким образом, ICH10 имеет 15 внешних прерываний – на 1 меньше 16 прерываний двух контроллеров 8259A, так как одно из прерываний используется для каскадного подключения второго контроллера 8259A. Микросхема 8259A содержит несколько выводов для

каскадного соединения, но мы их опустили ради простоты. В наши дни 8259A является составной частью другой микросхемы.

Хотя приведенное описание никоим образом не исчерпывает всех вопросов разработки шин, оно дает достаточно информации для общего понимания принципов работы шины и принципов взаимодействия с шиной центрального процессора. Теперь мы перейдем от общего к частному и рассмотрим несколько конкретных примеров процессоров и их шин.

1.6. СИСТЕМЫ RISC И CISC

Во второй половине 70-х годов разработчики экспериментировали со с сложными командами, которые возникли благодаря интерпретации. Разработчики пытались уменьшить разрыв между тем, что компьютеры способны делать, и тем, что требуют языки высокого уровня. Тогда никто не думал о разработке более простых машин, так же как сейчас мало кто занимается разработкой менее мощных электронных таблиц, сетей, веб-серверов и прикладных программ.

В компании IBM этой тенденции противостояла группа разработчиков во главе с Джоном Коком (John Cocke); они пытались воплотить идеи Сеймура Крея, создав экспериментальный высокоэффективный мини-компьютер 801. Компания IBM не занималась сбытом этой машины, результаты эксперимента были опубликованы только через несколько лет [Radin, 1982], но весть быстро разнеслась по свету, и другие производители тоже занялись разработкой подобных архитектур.

В 1980 году группа разработчиков в университете Беркли во главе с Дэвидом Паттерсоном (David Patterson) и Карло Секвином (Carlo Sequin) начала разработку не ориентированных на интерпретацию процессоров VLSI [Patterson, Patterson and Sequin, 1982]. Для обозначения этого понятия они придумали термин RISC, а новый процессор назвали RISC I, вслед за которым вскоре был выпущен RISC II. В 1981 году, Джон Хеннеси (John Hennesy) разработал и выпустил другую микросхему, которую он назвал MIPS [Hennesy, 1984]. Эти две микросхемы развились в коммерчески важные продукты SPARC и MIPS соответственно.

Новые процессоры были несовместимы с существующей продукцией, и разработчики вправе были включать туда новые наборы команд, которые могли бы повысить общую производительность системы. Основное внимание уделялось простым командам, которые могли быстро выполняться. Но вскоре разработчики осознали, что ключом к высокой производительности компьютера является разработка команд, которые можно быстро *запускать*. Не так важно, как долго выполняется та или иная команда, важнее то, сколько команд в секунду может быть запущено.

Когда разрабатывались эти простые процессоры, внимание привлекло относительно небольшое количество команд - около 50 (число команд в компьютерах VAX производства DEC и больших компьютерах производства IBM в то время составляло от 200 до 300). Компьютер **RISC**

(Reduced Instruction Set Computer – **компьютер с сокращенным набором команд**) – противопоставлялся системе **CISC** (Complex Instruction Set Computer – **компьютер с полным набором команд**) – намек на компьютер VAX, который был распространен в университетской среде. Сегодня мало кто считает, что количество команд в наборе так уж важно, но понятия сохранились до сих пор.

С того момента началось разделение на сторонников RISC и CISC систем. По мнению первых, наилучший способ разработки компьютеров – включение туда небольшого количества простых команд, каждая из которых выполняется за один цикл чтения данных, то есть производит над парой регистров арифметическую или логическую операцию и помещает результат обратно в регистр. Они утверждали, что даже если системе RISC приходится выполнять 4 или 5 команд вместо одной, которую выполняет CISC, RISC все равно выигрывает в скорости, так как RISC-команды выполняются в 10 раз быстрее (поскольку они не интерпретируются). К этому времени быстродействие основной памяти приблизилось к быстродействию специальных командных ПЗУ, поэтому недостатки интерпретации были налицо, что еще более поднимало популярность компьютеров RISC.

Учитывая преимущества RISC в плане производительности, можно предположить, что на рынке такие компьютеры, как UltraSPARC компании Sun, должны доминировать над компьютерами CISC (Intel Pentium и т.д.). Но этого не произошло. Причины две.

Первая: компьютеры RISC несовместимы с другими моделями компьютеров, а многие компании вложили миллиарды долларов в программное обеспечение для продукции Intel.

Вторая: компания Intel сумела воплотить те же идеи в архитектуре CISC. Процессоры Intel, начиная с процессора 486, содержат RISC-ядро, которое выполняет самые простые (и обычно самые часто используемые) команды за один цикл такта данных, а по обычной технологии CISC интерпретируются более сложные команды. В результате обычные команды выполняются быстро, а более сложные и редкие – медленно. Хотя при таком «гибридном» подходе производительность ниже, чем в архитектуре RISC, новая архитектура CISC имеет ряд преимуществ, поскольку позволяет использовать старое программное обеспечение без изменений, то есть, соблюдается принцип преемственности.

1.7. ПРИНЦИПЫ ПРОЕКТИРОВАНИЯ СОВРЕМЕННЫХ КОМПЬЮТЕРОВ

Прошло уже более сорока лет с тех пор, как были сконструированы первые компьютеры RISC, однако некоторые принципы их функционирования можно перенять, учитывая современное состояние технологии разработки аппаратного обеспечения [2]. Если происходит очень резкое изменение в технологии (например, новый процесс производства делает время обращения к памяти в 10 раз меньше, чем время

обращения к центральному процессору), меняются все условия. Поэтому разработчики всегда должны учитывать возможные технологические изменения, которые могли бы повлиять на баланс между компонентами компьютера.

Существует ряд принципов разработки, иногда называемых принципами RISC, которым по возможности стараются следовать производители универсальных процессоров. Из-за некоторых внешних ограничений, например требования совместимости с другими машинами, приходится время от времени идти на компромисс, но эти принципы – цель, к которой стремится большинство разработчиков.

- *Все команды должны выполняться непосредственно аппаратным обеспечением.* То есть обычные команды выполняются напрямую, без интерпретации команд. Устранение уровня интерпретации повышает скорость выполнения большинства команд. В компьютерах типа CISC более сложные команды могут разбиваться на несколько шагов, которые затем выполняются как последовательность микрокоманд. Эта дополнительная операция снижает быстродействие машины, но может использоваться для редко применяемых команд.

- *Компьютер должен запускать как можно больше команд в секунду.* В современных компьютерах используются много различных способов повышения производительности, главный из которых – запуск как можно большего количества команд в секунду. В конце концов, если процессор сможет запустить 500 млн команд в секунду, то эго производительность составляет 500 MIPS, сколько бы времени ни занимало выполнение этих команд.

(**MIPS** – сокращенное от Millions of Instructions Per Second – миллионов команд в секунду). Этот принцип предполагает, что параллелизм должен стать важным фактором повышения производительности, поскольку запустить на выполнение большое количество команд за короткий промежуток времени можно только в том случае, если есть возможность одновременного выполнения нескольких команд.

Хотя команды любой программы всегда располагаются в памяти в определенном порядке, компьютер изменить порядок их запуска (так как необходимые ресурсы памяти могут быть заняты) и (или) завершения. Конечно, если команда 1 устанавливает значение в регистр, а команда 2 использует этот регистр, нужно действовать с особой осторожностью, чтобы команда 2 не считала значение из регистра раньше, чем оно там окажется. Чтобы не допускать подобных ощибок, необходимо хранить в памяти большое количество дополнительной информации, но благодаря возможности выполнять несколько команд одновременно производительность все равно оказывается выше.

- *Команды должны легко декодироваться*. Предел количества запускаемых в секунду команд зависит от темпа декодирования отдельных команд. Декодирование команд позволяет определить, какие ресурсы им

необходимы и какие действия нужно выполнить. Все, что способствует упрощению этого процесса, полезно. Например, можно использовать единообразные команды с фиксированной длиной и с небольшим количеством полей. Чем меньше разных форматов команд, тем лучше.

- К памяти должны обращаться только команды загрузки и сохранения. Один из самых простых способов разбить операцию на отдельные шаги – сделать так, чтобы операнды большей части команд брались из регистров и возвращались туда же. Операция перемещения операндов из памяти в регистры и обратно может осуществляться в разных командах. Поскольку доступ к памяти занимает много времени, длительность которой невозможно спрогнозировать, выполнение этих команд могут взять на себя другие команды, единственное назначение которых – перемещение операндов между регистрами и памятью. То есть к памяти должны обращаться только команды загрузки и сохранения (LOAD и STORE).

- Регистров должно быть много. Поскольку доступ к памяти происходит относительно медленно, в компьютере должно быть много регистров (по крайней мере 32). Если слово было однажды загружено из памяти, при наличии большого числа регистров оно может содержаться в регистре до тех пор, пока не потребуется. Возвращение слова из регистра в память и новая загрузка этого же слова в регистр нежелательны. Лучший способ избежать излишних перемещений – наличие достаточного количества регистров.

1.8. ПАРАЛЛЕЛИЗМ НА УРОВНЕ КОМАНД

Разработчики компьютеров стремятся к тому, чтобы повысить производительность своих машин. Один из способов заставить процессоры работать быстрее – повышение их тактовой частоты, однако при этом существуют некоторые технические ограничения на то, что можно сделать методом «грубой силы» на данный момент. Поэтому большинство проектировщиков для повышения производительности при данной тактовой частоте процессора используют параллелизм (выполнение двух или более операций одновременно).

Существует две основные формы параллелизма: параллелизм на уровне команд и параллелизм на уровне процессоров [2]. В первом случае параллелизм реализуется за счет выпуска большого количества команд каждую секунду. Во втором случае над одним заданием работают одновременно несколько процессоров. Каждый подход имеет свои преимущества.

1.8.1. Конвейеры

Главным препятствием высокой скорости выполнения команд является необходимость их загрузки из памяти. Для разрешения этой проблемы можно вызывать команды из памяти заранее и хранить в специальном наборе регистров. Эта идея использовалась еще в 1959 году при разработке компьютера Stretch компании IBM, а набор регистров был назван **буфером**

выборки с упреждением. Таким образом, когда требовалась определенная команда, она вызывалась прямо из буфера, а обращения к памяти не происходило.

В действительности при выборке с упреждением команда обрабатывается за два шага: сначала происходит выборка команды, а затем ее выполнение. Дальнейшим развитием этой стратегии стала концепция **конвейера**. При использовании конвейера команда обрабатывается уже не за два, а за большее количество шагов, каждый из которых реализуется определенным аппаратным компонентом, причем все эти компоненты могут работать параллельно.

На рисунке 1.24 изображен конвейер из 5 блоков, которые называются **ступенями**. Первая ступень (блок С1) вызывает команду из памяти и помещает ее в буфер, где она хранится до тех пор, пока не потребуется. Вторая ступень (блок С2) декодирует эту команду, определяя ее тип и тип ее операндов. Третья ступень (блок С3) определяет местонахождение операндов и вызывает их из регистров или из памяти. Четвертая ступень (блок С4) выполняет команду, обычно проводя операнды через тракт данных. И наконец, блок С5 записывает результат обратно в нужный регистр.

Рисунок 1.24 – Пятиступенчатый конвеер

Конвейеры позволяют добиться компромисса между **временем запаздывания** (время выполнения одной команды) и **пропускной способностью процессора** (количество команд, выполняемых процессором в секунду).

1.8.2. Суперскалярные архитектуры

Один конвейер – хорошо, а два – еще лучше. Одна из возможных схем процессора с двумя конвейерами показана на рисунке 1.25. В ее основе лежит конвейер, изображенный на рисунке 1.24. Здесь общий блок выборки команд вызывает из памяти сразу по две команды и помещает каждую из них в один из конвейеров. Каждый конвейер содержит АЛУ для параллельных операций. Чтобы выполняться параллельно, две команды не должны конфликтовать из-за ресурсов (например, регистров) и ни одна из них не должна зависеть от результата выполнения другой. Как и в случае с одним конвейером, либо компилятор должен гарантировать отсутствие нештатных ситуаций (когда, например, аппаратура не обеспечивает проверку команд на совместимость и при обработке таких команд выдает некорректный результат), либо конфликты должны выявляться и устраняться дополнительным оборудованием непосредственно в ходе выполнения команд.

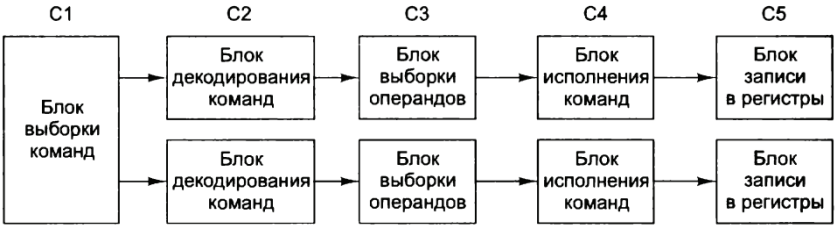


Рисунок 1.25 – Сдвоенный пятиступенчатый конвейер с общим блоком выборки команд

Сначала конвейеры (как сдвоенные, так и обычные) использовались только в RISC-компьютерах. У процессора 386 было параллельное функционирование отдельных блоков. Этот механизм стал прообразом 5-ступенчатого конвейера микропроцессора 486. Процессор 486 имел один пятиступенчатый конвейер, а Pentium – два таких конвейера. Похожая схема изображена на рисунке 3, но разделение функций между второй и третьей ступенями (они назывались декодер 1 и декодер 2) было немного другим. Главный конвейер (u-конвейер) мог выполнять произвольные команды. Второй конвейер (v-конвейер) мог выполнять только простые команды с целыми числами, а также одну простую команду с плавающей точкой (FXCH).

Проверка совместимости команд для параллельного выполнения осуществляется по жестким правилам. Если команды, входящие в пару, были сложными или несовместимыми, выполнялась только одна из них (в u-конвейере). Оставшаяся вторая команда сопоставлялась со следующей командой. Команды всегда выполнялись по порядку. Специальные компиляторы для процессора Pentium объединяли совместимые команды в пары и могли генерировать программы, выполняющиеся быстрее, чем в предыдущих версиях. Измерения показали, что программы, в которых применяются операции с целыми числами, при той же тактовой частоте на Pentium выполняются вдвое быстрее, чем на 486 [Pountain,1993]. Выигрыш в скорости достигался благодаря второму конвейеру.

Переход к четырем конвейерам возможен, но требует громоздкого аппаратного обеспечения. Вместо этого используется другой подход. Основная идея – один конвейер с болшим количеством функциональных блоков, как показано на рисунке 4. Intel Core, к примеру, имеет сходную структуру (подробно мы рассмотрим ее в XXX). В 1987 году для обозначения этого подхода был введен термин **суперскалярная архитектура** [Agerwala and Cocke, 1987]. Однако подобная идея нашла воплощение еще более 50 лет назад в компьютере CDC 6600. Этот компьютер вызывал команду из памяти каждые 100 нс и помещал ее в один из 10 функциональных блоков для параллельного выполнения. Пока

команды выполнялись, центральный процессор вызывал следующую команду.

Со временем определение «суперскалярности» несколько изменилось. Теперь «суперскалярными» называют процессоры, способные запускать несколько команд (зачастую от четырех до шести) за один тактовый цикл. Естественно, при передаче всех этих команд в суперскалярном процессоре должно быть несколько функциональных блоков. Поскольку в процессорах этого типа, как правило, предусматривается один конвейер, его устройство обычно соответствует рисунку 1.26.

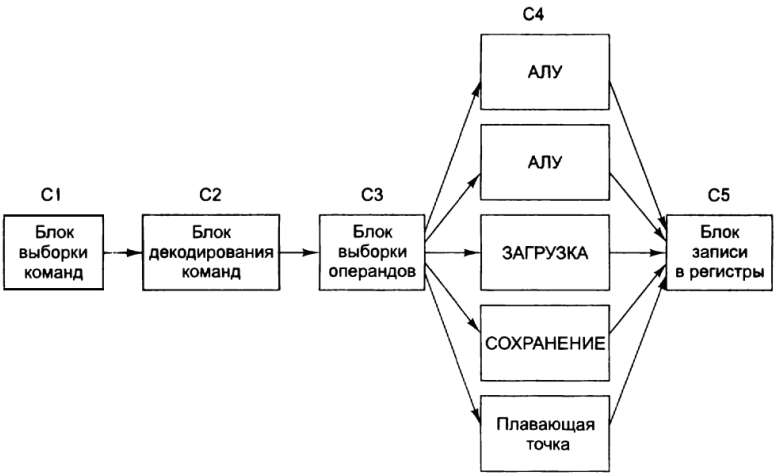


Рисунок 1.26 – Суперскалярный процессор с пятью функциональными блоками

В соответствие с этим определением компьютер 6600 формально не был суперскалярным с технической точки зрения – ведь за один тактовый цикл в нем запускалось не больше одной команды. Однако при этом был достигнут отличный результат – команды запускались быстрее, чем исполнялись. На самом деле, разница в производительности между ЦП с циклом 100 нс, передающим за этот период по одной команде четырем функциональным блокам, и ЦП с циклом в 400 нс, запускающим за это время четыре команды, трудноуловима. В обоих процессорах соблюдается принцип превышения скорости запуска над скоростью управления; при этом рабочая нагрузка распределяется между несколькими функциональными блоками.

Отметим, что на выходе ступени 3 команды появляются значительно быстрее, чем ступень 4 способна их обрабатывать. Если бы на выходе ступени 3 команды появлялись каждые10 нс, а все функциональные блоки

делали свою работу также за 10 нс, то на ступени 4 всегда функционировал бы только один блок, что сделало бы саму идею конвейера бессмысленной. В действительности большинству функциональных блоков ступени 4 (точнее, обоим блокам доступа к памяти и блоку выполнения операций с плавающей точкой) для обработки команды требуется значительно больше времени, чем занимает один цикл. Как видно из рисунка 4, на ступени 4 может быть несколько АЛУ.

1.9. ПАРАЛЛЕЛИЗМ НА УРОВНЕ ПРОЦЕССОРОВ

Спрос на компьютеры, работающие все с более и более высокой скоростью, не прекращается [2]. Астрономы хотят выяснить, что произошло в первую микросекунду после Большого взрыва, экономисты хотят смоделировать всю мировую экономику, подростки хотят играть в трехмерные интерактивные игры со своими виртуальными друзьями через Интернет. Быстродействие процессоров растет, но у них постоянно возникают проблемы со скоростью передачи информации, поскольку скорость распространения электромагнитных волн в медных проводах и света в оптико-волоконных кабелях по-прежнему остается равной 20 см/нс независимо от того, насколько умны инженеры компании Intel. Кроме того, чем быстрее работает процессор, тем сильнее он нагреется, поэтому возникает задача защиты его от перегрева.

Параллелизм на уровне команд в определенной степени помогает, но конвейеры и суперскалярная архитектура обычно повышают скорость работы всего лишь в 5-10 раз. Чтобы увеличить производительность в 50, 100 и более раз, нужно создавать компьютеры с несколькими процессорами. Ознакомимся с устройством таких компьютеров.

1.9.1 Матричные компьютеры

Многие задачи в физических и технических науках предполагают использование циклов, массивов или других упорядоченных структур [2]. Часто одни и те же вычисления многократно повторяются с разными наборами данных. Упорядоченность и структурированность программ, предназначенных для выполнения такого рода вычислений, очень удобны в плане ускорения вычислений за счет параллельной обработки команд. Существует две схемы ускоренного выполнения больших научных программ: SIMD-процессоры и векторные процессоры. Хотя между этими схемами много общего, как ни парадоксально, первую обычно представляют как параллельный компьютер, а вторую – как добавление расширения параллельного вычислителя.

Компьютеры с распараллеливанием по данным нашли много успешных применений благодаря своей выдающейся эффективности. Они могут обеспечивать существенную вычислительную мощность с меньшим количеством транзисторов по сравнению с альтернативными решениями. Гордон Мур (автор закона Мура) также известен своим замечанием, что кремний стоит около 1 миллиарда долларов за акр (4047 квадратных

метров). Таким образом, чем больше вычислительной мощи удастся выжать из этого акра кремния, тем больше денег компьютерная компания заработает на его продаже. Компьютеры с распараллеливанием по данным являются одним из самых эффективных средств для «выжимания» производительности из кремния. Так как все процессоры выполняют одну инструкцию, системе необходим только один «мозг», управляющий компьютером. Соответственно, процессору нужна одна ступень выборки команд, одна ступень декодирования и один блок управляющей логики. Так достигается существенная экономия, которая дает параллельным компьютерам большое преимущество перед другими процессорами – при условии, что выполняемые программы имеют упорядоченную структуру с большой степенью параллелизма.

SIMD-процессор (Single Instruction-stream Multiple Data-stream – один поток команд с несколькими потоками данных) состоит из большого числа сходных процессоров, которые выполняют одну и ту же последовательность команд применительно к разным наборам данных. Первым в мире таким процессором был IAC IV (университет Иллинойс) [Bouknight et al., 1972]. Первоначально предполагалось сконструировать машину, состоящую из четырех квадрантов, каждый из которых содержал матрицу размером 8x8 из блоков процессор/память. Для каждого квадранта имелся один управляющий блок. Он рассылал команды, которые выполнялись всеми процессорами одновременно, при этом каждый процессор использовал собственные данные из собственной памяти. Из-за очень высокой стоимости был построен только один такой квадрант, но он мог выполнять 50 млн операций с плавающей точкой в секунду. Если бы при создании машины использовалось четыре квадранта, она могла бы выполнять 1 млрд операций с плавающей точкой в секунду, и вычислительные возможности такой машины в два раза превышали бы возможности компьютеров всего мира.

Современные графические процессоры (GPU) широко используют SIMD-обработку для обеспечения высокой вычислительной мощности при относительно небольшом количестве транзисторов. Обработка графики отлично подходит для SIMD-процессоров, потому что большинство алгоритмов имеет четкую структуру с повторением операций для пикселов, вершин, текстур и ребер. На рисунке 5 изображен SIMD-процессор ядра графического процессора Nvidia Fermi/ Он содержит до 16 потоковых мультипроцессоров (SM, Stream Multiprocessor) SIMD, каждый из которых содержит 32 процессора SIMD. За каждый цикл планировщик выбирает два потока для выполнения на процессоре SIMD. Затем следующая команда каждого потока выполняется на процессорах SIMD (до 16, что при отсутствии достаточного параллелизма данных используется меньшее количество процессоров). Если каждый поток способен выполнить 16 операций за цикл, полностью загруженное ядро графического процессора Fermi с 32 SM будет выполнять целых 512 операций за цикл. Это весьма

впечатляющее достижение, если учесть, что четырехядерный процессор общего назначения того же размера с трудом достигает 1/32 такой вычислительной мощи.

С точки зрения программиста **векторный процессор** (vector processor) очень похож на SIMD-процессор. Он также чрезвычайно эффективен при выполнении последовательности операций над парами элементов данных.

Однако в отличие от SIMD-процессора, все операции сложения выполняются в одном блоке суммирования, который имеет конвейерную структуру. Компания Cray Research, основателем которой был Сеймур Крей, выпустила множество моделей векторных процессоров, начиная с модели Cray-1 (1974).

Оба типа процессоров работают с массивами данных. Оба они выполняют одни и те же команды, которые, например, попарно складывают элементы двух векторов. Однако если у SIMD-процессора столько же суммирующих устройств, сколько элементов в массиве, векторный процессор содержит **векторный регистр**, состоящий из набора традиционных регистров. Эти регистры загружаются из памяти единственной командой, которая фактически делает это последовательно. Команда сложения попарно складывает элементы двух таких векторов, загружая их из двух векторных регистров в суммирующее устройство с конвейерной структурой.

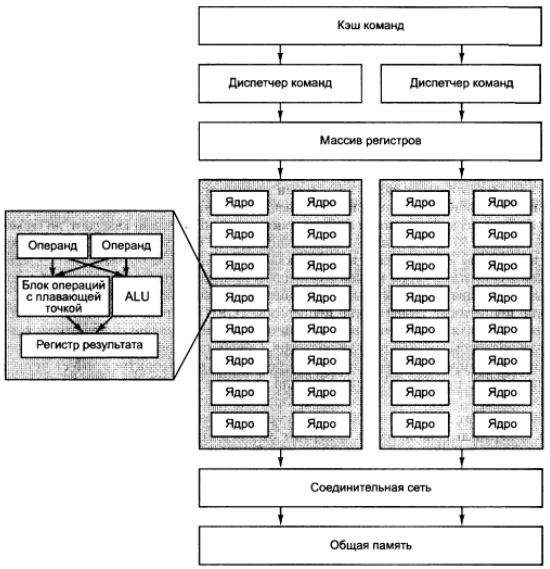


Рисунок 1.27 – SIMD-ядро графического процессора Fermi

В результате из суммирующего устройства выходит другой вектор, который либо перемещается в векторный регистр, либо сразу используется в качестве операнда при выполнении другой операции с векторами. Команды SSE (Streaming SIMD Extension) в архитектуре Intel Core используют эту модель расширения для ускорения вычислений с высокой степенью упорядоченности – например, обработки мультимедийных и научных данных. В этом отношении компьютер ILLIAC IV можно считать одним из прародителей процессора Intel Core.

1.9.2. Мультипроцессоры

Элементы процессора, распараллеленного по данным, связаны между собой, поскольку их работу контролирует единый блок управления [2]. Система из нескольких параллельных процессоров, имеющих общую память, называется **мультипроцессором**. Поскольку каждый процессор может записывать информацию в любую часть памяти и считывать информацию из любой части памяти, чтобы не допустить каких-либо нестыковок, их работа должна согласовываться программным обеспечением. В ситуации, когда два или несколько процессоров имеют возможность тесного взаимодействия, а именно так происходит в случае с мультипроцессорами, эти процессоры называют сильно связанными (Tightly coupled).

Возможны разные способы воплощения этой идеи. Самый простой из них – соединение единственной шиной нескольких процессоров и общей памяти. Схема мультипроцессора показана на рисунке 1.28.

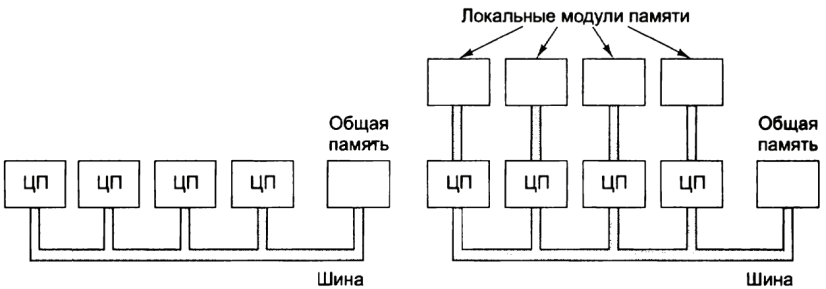


Рисунок 1.28 – Мультипроцессор с единственной шиной и общей памятью (слева) и мультипроцессор с собственной локальной памятью для каждого процессора (справа)

Естественно, при наличии большого числа быстродействующих процессоров, которые постоянно пытаются получить доступ к памяти через одну и ту же шину, будут возникать конфликты. Чтобы решить эту проблему и повысить производительность компьютера, разработаны различные схемы. Одна из них изображена на рисунке 1.28 справа. В таком компьютере каждый процессор имеет собственную локальную память,

недоступную для других процессоров. Эта память используется для тех программ и данных, которые не нужно разделять между несколькими процессорами. При обращении к локальной памяти основная шина не используется, и, таким образом, объем передаваемой по ней информации становится меньше. Возможны и другие варианты решения проблемы (например, кэширование – см. ниже).

Мультипроцессоры имеют преимущество перед другими видами параллельных компьютеров, поскольку с единой общей памятью очень легко работать. Например, представим, что программа ищет раковые клетки на сделанном через микроскоп снимке ткани. Фотография в цифровом виде может храниться в общей памяти, при этом каждый процессор будет обследовать какую-нибудь определенную область фотографии. Поскольку каждый процессор имеет доступ к общей памяти, обследование клетки, расположенной сразу в нескольких областях, не представляет трудностей.

1.9.3. Мультикомпьютеры

Мультипроцессоры с небольшим числом процессоров (≤256) разрабатывать достаточно просто, а вот создание больших мультипроцессоров представляет определенные трудности [2]. Сложность заключается в том, чтобы связать все процессоры общей памятью. Поэтому многие разработчики просто отказались от идеи разделения памяти и стали создавать системы без общей памяти, состоящие из большого числа взаимосвязанных компьютеров, у каждого из которых имеется собственная память. Такие системы называются мультикомпьютерами. В них процессоры являются слабо связанными, в противоположность сильно связанным процессорам в мультипроцессорных системах.

Процессоры мультикомпьютера отправляют друг другу сообщения (что-то вроде электронной почты, но гораздо быстрее). Каждый компьютер не обязательно соединять со всеми другими, поэтому обычно в качестве топологий используются двух- и трехмерные решетки, а также деревья и кольца. Хотя на пути до места назначения сообщения проходят через один или несколько промежуточных компьютеров, время передачи занимает всего несколько микросекунд. Уже работают мультикомпьтеры, содержащие до 250 000 процессоров – например, Blue Gene/P фирмы IBM [2].

Поскольку мультипроцессоры легче программировать, а мультикомпьютеры – конструировать, появилась идея создания гибридных систем, сочетающих в себе достоинства обоих технологий. Такие компьютеры представляют иллюзию общей памяти. При этом в действительности она не существует.

1.10. ИЕРАРХИЧЕСКАЯ СТРУКТУРА ПАМЯТИ

Иерархическая структура памяти является традиционным решением проблемы хранения больших объемов данных (рисунок 1.29) [2]. На самом

верху иерархии находятся регистры процессора. Доступ к регистрам осуществляется быстрее всего. Дальше идет кэш-память, объем которой сейчас составляет от 32 Кбайт до нескольких мегабайт. Затем следует основная память, объем которой в настоящее время лежит в диапазоне от 1 Гбайт до сотен гигабайт. Затем идут магнитные диски и твердотельные накопители для долгосрочного хранения данных. Нижний уровень иерархии занимают накопители на магнитной ленте и оптические диски для хранения архивов.

Рисунок 1.29 – Пятиуровневая организация памяти

По мере продвижения сверху вниз по иерархии меняются три параметра. Во-первых, увеличивается время доступа. Доступ к регистрам занимает несколько наносекунд, доступ к кэш-памяти – немного больше, доступ к основной памяти – несколько десятков наносекунд. Дальше идет большой разрыв: доступ к дискам происходит по крайней мере в 10 раз медленнее для твердотельных дисков и в сотни раз медленнее для магнитных дисков. Время доступа к магнитным лентам и оптическим дискам вообще может измеряться в секундах (поскольку их накопители информации еще нужно взять и поместить в соответствующее устройство).

Во-вторых, растет объем памяти. Регистры могут содержать в лучшем случае 128 байт, кэш-память – десятки мегабайт, основная память – гигабайты, магнитные диски – терабайты. Магнитные ленты и оптические диски хранятся автономно от компьютера, поэтому их совокупный объем ограничивается только финансовыми возможностями владельца.

В третьих, увеличивается количество битов, которые вы получаете за один доллар. Стоимость объема основной памяти измеряется в долларах за мегабайт, твердотельных накопителей – в долларах за гигабайт, магнитных дисков и лент – в центах за гигабайт или еще дешевле.

1.11. ВВОД-ВЫВОД

Компьютерная система состоит из трех основных компонентов: центрального процессора, памяти (основной и вспомогательной) устройств ввода-вывода (принтеров, сканеров и модемов). Рассмотрим как устройства ввода-вывода соединяются с остальными компонентами системы [2].

1.11.1. Шины

Большинство персональных компьютеров и рабочих станций имеют физическую структуру, сходную с показанной на рисунке 1.30. Обычно устройство представляет собой металлический корпус с большой интегральной схемой на дне, которая называется **материнской платой** (системной платой). Материнская плата содержит микросхему процессора, несколько разъемов для модулей DIMM и различные вспомогательные микросхемы. Еще на материнской плате располагаются шина (она тянется вдоль платы) и несколько разъемов для подсоединения устройств ввода-вывода.

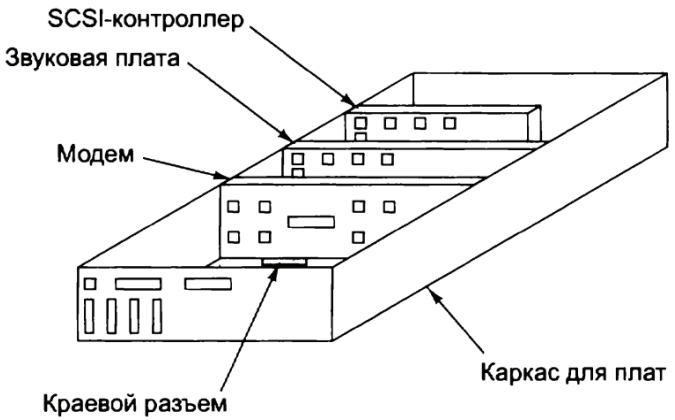


Рисунок 1.30 – Физическая структура персонального компьютера

Логическую структуру обычного персонального компьютера иллюстрирует рисунок 1.31. У данного компьютера имеется одна шина для соединения центрального процессора, памяти и устройств ввода-вывода; однако большинство систем имеют две и более шин. Каждое устройство ввода-вывода состоит из двух частей: одна объединяет большую часть электроники и называется контроллером, а другая представляет собой само устройство ввода-вывода, например дисковод.

Контроллер обычно располагается на плате, которая вставляется в свободный разъем. Исключение представляют собой контроллеры устройств, являющихся неотъемлемыми составными частями компьютера (например, клавиатуры), которые иногда располагаются на материнской плате. Хотя дисплей (монитор) и нельзя назвать дополнительным устройством, соответствующий контроллер иногда располагается на встроенной плате, чтобы пользователь мог по желанию выбирать платы с графическими ускорителями или без них, устанавливать дополнительную память и т. д. Контроллер связывается с самим устройством кабелем, который соединяется с разъемом на задней стороне корпуса.

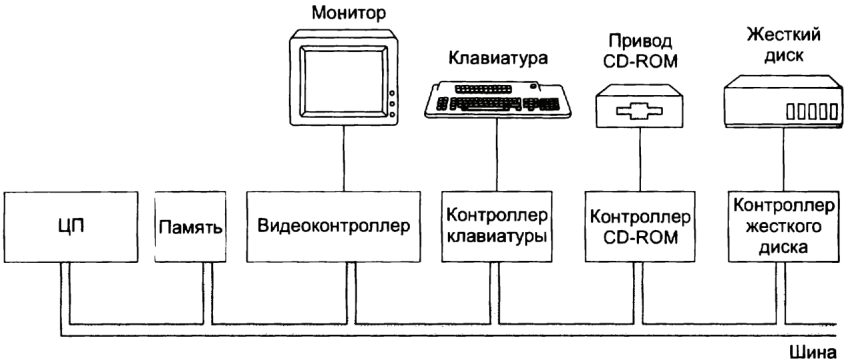


Рисунок 1.31 – Логическая структура обычного персонального компьютера

Контроллер управляет своим устройством ввода-вывода и для этого регулирует доступ к шине. Например, если программа запрашивает данные с диска, она посылает команду контроллеру диска, который затем отправляет диску команду поиска и другие команды. После нахождения соответствующей дорожки и сектора диск начинает передавать контроллеру данные в виде потока битов. Задача контроллера состоит в том, чтобы разбить поток битов на фрагменты и записывать каждый такой фрагмент по мере накопления битов для него в память. Отдельный фрагмент обычно представляет собой одно или несколько слов. Если контроллер считывает данные из памяти или записывает их в память без участия центрального процессора, то говорят, что осуществляется **прямой доступ к памяти** (Direct Memory Access, **DMA**). Когда передача данных заканчивается, контроллер выдает **прерывание**, вынуждая центральный процессор приостановить работу текущей программы и начать выполнение особой процедуры. Эта процедура называется **программой обработки прерываний** и нужна она для того, чтобы проверить, нет ли ошибок, в случае их обнаружения произвести необходимые действия и сообщить операционной системе, что процесс ввода-вывода завершен. Когда

программа обработки прерывания завершится, процессор возобновляет работу программы, которая была приостановлена в момент прерывания.

Шина используется не только контроллерами ввода-вывода, но и процессором для передачи команд и данных. А что происходит, если процессор и контроллер ввода-вывода хотят получить доступ к шине одновременно? В этом случае особая микросхема, которая называется **арбитром шины**, решает, чья очередь первая. Обычно предпочтение отдается устройствам ввода-вывода, поскольку работу дисков и других движущихся устройств нельзя прерывать, так как это может привести к потере данных. Когда ни одного устройства ввода-вывода не функционирует, центральный процессор может полностью распоряжаться шиной для взаимодействия с памятью. Однако если работает какое-нибудь устройство ввода-вывода, оно будет запрашивать доступ к шине и получать его каждый раз, когда ему это необходимо. Этот процесс, который притормаживает работу компьютера, называется захватом цикла памяти (cycle stealing).

Описанная структура успешно использовалась в первых персональных компьютерах, поскольку все их компоненты работали примерно с одинаковой скоростью. Однако как только центральные процессоры, память и устройства ввода-вывода стали работать быстрее, возникла проблема: шина перестала справляться с нагрузкой. В случае закрытых систем, таких как инженерные рабочие станции, решением проблемы стала разработка для следующей модели машины новой шины с более высокой скоростью передачи данных. Поскольку в закрытых системах никто никогда не переносил устройства ввода-вывода со старой модели на новую, такой подход работал успешно.

Однако в мире персональных компьютеров большая часть пользователей, изменяя свой компьютер новой моделью, никак не рассчитывает одновременно отказываться от своих старых и привычных принтера, сканера и модема. Кроме того, существовала целая отрасль промышленности, выпускавшая широкий спектр устройств ввода-вывода для компьютеров IBM PC, и производители этих устройств совершенно не были заинтересованы в том, чтобы начинать все свои разработки заново. Компания IBM узнала об этом на горьком опыте, выпустив после линейки IBM PC линейку PS/2. У компьютеров PS/2 была новая шина с более высокой скоростью передачи данных, но большинство производителей клонов продолжали использовать старую шину PC, которая сейчас называется шиной ISA (Indastry Standart Architecture – стандартная промышленная архитектура). Большинство производителей дисков и устройств ввода-вывода также продолжали выпускать контроллеры для старой модели, и компания IBM оказалась в весьма неприятной ситуации: на тот момент она оказалась единственным производителем персональных компьютеров, несовместимых с линейкой IBM. В конце концов компания была вынуждена вернуться к производству компьютеров на основе шины

ISA. В наши дни шина ISA встречается разве что в самых древних системах и в музеях компьютерной техники, так как на смену ей пришли новые, более быстрые стандарты архитектуры шин. Отметим, что аббревиатура ISA также расшифровывается как Instruction Set Architecture (архитектура набора команд), если речь идет об уровнях иерархии команд.

1.11.2. Шины PCI и PCIe

Хотя влияние рынка было направлено на то, чтобы старая шина оставалась неизменной, быстрее она работать не стала, и нужно было что-то предпринять. В результате другие компании начали производить компьютеры с несколькими шинами, одной из которых была либо прежняя шина ISA, либо шина EISA (Extended ISA – **расширенная стандартная промышленная архитектура**), как и ISA совместимая со старыми устройствами ввода-вывода. Что касается другой шины, то в настоящее время самой популярной моделью является шина PSI (Peripheral Component Interconnect – взаимодействие периферийных компонентов), разработанная компанией Intel, которая решила открыть всю связанную с шиной техническую информацию, чтобы сторонние производители (в том числе конкуренты компании) могли разрабатывать соответствующие устройства.

Существует много различных конфигураций шины PCI. Наиболее типичная из них показана на рисунке 1.32. В такой конфигурации центральный процессор взаимодействует с контроллером памяти по выделенному высокоскоростному соединению. Таким образом, контроллер соединяется с памятью непосредственно, то есть передача данных между центральным процессором и памятью происходит не через шину PCI. Другие периферийные устройства подсоединяются прямо к шине PCI. Машина такого типа обычно содержит 2 или 3 пустых разъема PCI, чтобы покупатели имели возможность подключать карты PCI (для новых периферийных устройств).

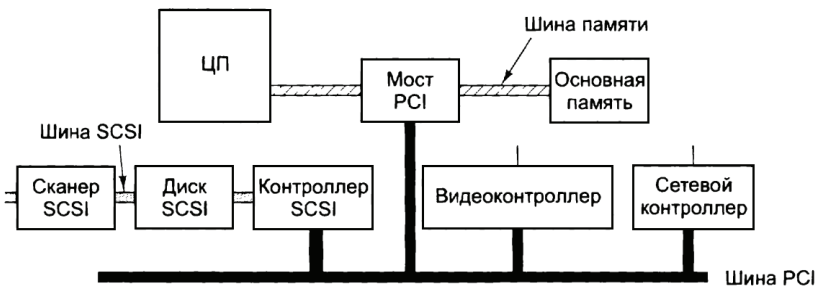


Рисунок 1.32 – Современный персональный компьютер с шиной PCI. Контроллер SCSI является PCI-устройством

Как бы быстро ни работало компьютерное оборудование, найдется много людей, которым оно покажется слишком медленным. Такая судьба постигла и шину PCI, которая была заменена шиной **PCI Express**

(сокращенно **PCIe**). Многие современные компьютеры поддерживают обе шины, благодаря чему пользователи могут подключать новые, быстрые устройства к шине PCIe, а старые, более медленные – к шине PCI.

Если шина PCI представляла собой обновленную версию старой шины ISA с более высокой скоростью и разрядностью параллельно передаваемых данных, PCIe представляет кардинальное изменение по сравнению с шиной PCI. Собственно, это вообще не шина, а одноранговая сеть, использующая разрядно-последовательные линии и коммутацию пакетов. У нее больше от Интернета, чем от традиционных щин. Архитектура PCIe изображена на рисунке 1.33.

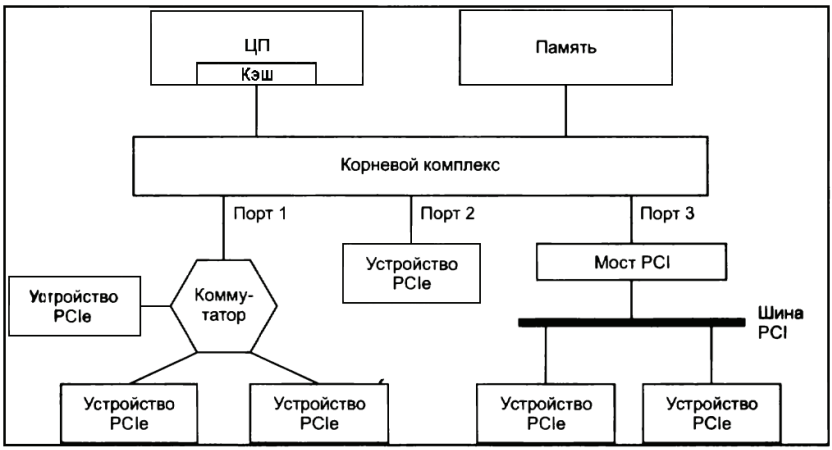


Рисунок 1.33 – Архитектура системы PCIe с тремя портами PCI

Некоторые особенности шины PCIe сразу бросаются в глаза. Во-первых, соединения между устройствами являются последовательными, то есть имеют разрядность в один бит вместо 8, 16, 32 или 64 бит. Хотя казалось бы, 64-разрядное соединение обладает более высокой пропускной способностью, на практике различия во времени распространения 64-разрядной информации, называемые **расфазировкой**, заставляют использовать относительно низкие скорости передачи данных. По последовательному соединению данные передаются на значительно более высокой скорости, что более чем компенсирует потерю параллелизма. Шины PCI работают на максимальной тактовой частоте 66 МГц. При передаче 64 бит за такт скорость передачи данных составляет 528 Мбайт/с. При тактовой частоте 8 Гбит/с, даже в случае последовательной передачи, скорость передачи по шине PCIe составляет 1 Гбайт/с. Кроме того, обмен данными между устройством и корневым комплексом или коммутатором не ограничивается одной проводной парой. Устройство может иметь до 32 проводных пар, называемых трактами (lanes) или дорожками. Тракты

работают несинхронно, поэтому расфазировка в данном случае несущественна. На большинстве материнских плат имеется 16-трактовый разъем для графической карты, что для PCIe 3.0 обеспечивает пропускную способность в 16 Гбайт/с – примерно в 30 раз больше, чем у графических карт PCI. Такая пропускная способность необходима для приложений, требования которых постоянно растут – например, трехмерной графики.

Во-вторых, все взаимодействия являются одноранговыми. Когда процессор хочет обратиться к устройству, он отправляет этому устройству пакет и обычно получает ответ. Пакет проходит через корневой комплекс на материнской плате, а затем передается устройству – как правило, через коммутатор (или для устройств PCI – через мост PCI). Переход от системы, в которой все устройства взаимодействуют с общей шиной, к системе с одноранговыми взаимодействиями, соответствует направлению развития Ethernet (популярная технология локальных сетей), которая в исходном варианте тоже использовала широковещательный канал, но в наши дни используется на одноранговых взаимодействиях с использованием коммутаторов.

1.12. ТЕРМИНАЛЫ

В настоящее время существуют множество разнообразных устройств ввода-вывода [2]. Мы коснемся только наиболее распространенных из них. Терминалы компьютера состоят из двух частей: клавиатуры и монитора. В мэйнфреймах эти части объединены в одно устройство и связаны с самим мэйнфреймом обычным или телефонным проводом. В авиакомпаниях, банках и различных отраслях промышленности, где работают мэйнфреймы, эти устройства до сих пор широко распространены. В мире персональных компьютеров клавиатура и монитор – независимые устройства, однако технологически клавиатура и монитор мэйнфрейма ничем не отличаются от соответствующих устройств ПК.

1.12.1. Клавиатуры

Клавиатура предназначена для ввода алфавитно-цифровой информации и управления работой ПЭВМ.

Существуют несколько видов клавиатур [2]. У первых компьютеров IBM PC под каждой клавишей находился переключатель, который давал ощутимую отдачу и щелкал при нажатии клавиши. Сегодня механический контакт с печатной платой при нажатии клавиш происходит лишь у самых дешевых клавиатур. У клавиатур получше между клавишами и печатной платой располагается слой эластичного материала (особого типа резины). Под каждой клавишей находится небольшой купол, который прогибается в случае нажатия клавиши. Проводящий материал, находящийся внутри купола, замыкает схему. У некоторых клавиатур под каждой клавишей находится магнит, который при нажатии клавиши проходит через катушку и таким образом вызывает электрический ток. Используются и другие методы, как механические, так и электромагнитные.

В персональных компьютерах при нажатии клавиши происходит процедура прерывания и запускается программа обработки прерывания (эта программа является частью программного обеспечения операционной системы). Программа обработки прерывания считывает содержимое аппаратного регистра в контроллер клавиатуры, чтобы получить номер нажатой клавиши (от 1 до 102). Когда клавиша отпускается, происходит второе прерывание. Так, если пользователь нажимает клавишу SHIFT, затем нажимает и отпускает клавишу M, а после этого отпускает клавишу SHIFT, операционная система понимает, что ему нужна прописная, а не строчная буква M. Обработка нажатий клавиш SHIFT, CTRL и ALT в сочетании с другими клавишами выполняется только программно (сюда же относится известное сочетание клавиш CTRL+ALT+DEL, которое используется для перезагрузки всех компьютеров IBM PC и их клонов).

Принцип организации клавиатуры показан на рис. 1.34.

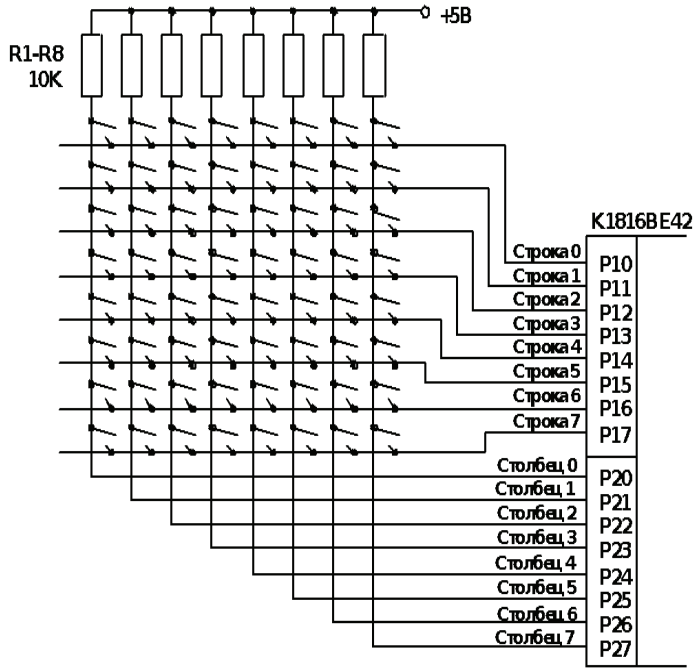


Рисунок 1.34 – Организация клавиатуры

При нажатии клавиши соответствующая строка и столбец замыкаются, образуя соединение. Контроллер клавиатуры определяет номер строки и столбца, в которых произошло замыкание и выдает код, соответствующий

нажатой клавише. Этот процесс называется сканированием клавиатуры. Сканирование производится следующим образом: выходной порт выдает сигнал логического нуля в строку 0 и сигналы логической единицы во все остальные строки. Затем через входной порт производится считывание состояния столбцов. Если ни на одном из столбцов нет сигнала логического 0, проверяется строка 1 (т. е. выдается сигнал логического нуля в строку 1) и так далее.

При обнаружении сигнала логического 0 на одном из столбцов, определяется нажатая клавиша, номер строки которой соответствует выводимой комбинации сигналов, а номер столбца - по вводимой комбинации сигналов. Затем в соответствии с позицией нажатой клавиши формируется код нажатой клавиши.

Контроллер клавиатуры осуществляет сканирование клавиатуры, обработку дребезга при нажатии и отпускании клавиш, буферизацию кодов, поддержание передачи данных с системным устройством в последовательном коде с обеспечением требуемого протокола связи, а также выполняет автотестирование при включении электропитания или по требованию системного устройства.

1.12.2. Сенсорные экраны

Хотя клавиатуры еще не собираются отправляться вслед за механическими пишущими машинками, в области компьютерного ввода появилась новая технология сенсорных экранов [2]. Хотя эти устройства вышли на массовый рынок только с выходом Apple iPhone в 2007 году, появились они намного раньше. Первый сенсорный экран был разработан в фирме Royal Radar Establishment в Мэлверне, Великобритания, в 1965 году. Даже характерные жесты масштабирования сведением/разведением пальцев, так широко разрекламированные для iPhone, были изобретены в ходе работы, проводившейся в университете Торонто в 1982 году. С тех пор исследователи разработали и вывели на рынок много разных технологий.

Сенсорные устройства делятся на прозрачные и непрозрачные. Типичное непрозрачное сенсорное устройство – сенсорная панель (тачпад) на ноутбуке. Типичное прозрачное устройство – экран смартфона или планшетного компьютера. Мы ограничимся рассмотрением устройств второго типа, которые обычно называются **сенсорными экранами**. Основные разновидности сенсорных экранов – инфракрасные, резистивные и емкостные.

Принцип работы инфракрасных экранов основан на размещении инфракрасных передатчиков (скажем, инфракрасных светодиодов или лазеров) на левом и верхнем краях оправы, с детекторами на правом и нижнем краях. Когда палец, стилус или любой непрозрачный объект блокирует один или несколько лучей сетки, соответствующий детектор обнаруживает исчезновение сигнала. Оборудование устройства может сообщить операционной системе, какой из лучей был заблокирован: по

этим данным вычисляются координаты (x,y) пальца или стилуса. Эта технология появилась уже давно, она до сих пор используется в интерактивных киосках и других областях, но в мобильных устройствах она не применяется.

Другая старая технология изготовления сенсорных экранов – **резистивная** – состоит из двух слоев. Верхний гибкий слой содержит большое количество горизонтальных проводников. В находящейся под ним мембране проходят вертикальные проводники. Когда палец или другой объект нажимает на экран, один из проводников верхней панели соприкасается (или проходит близко) к перпендикулярным проводникам нижней панели. Электроника устройства позволяет определить, в какой области было произведено нажатие. Резистивные экраны очень дешевы, они широко применяются в областях, критичных по цене.

Обе технологии хорошо работают при нажатии одним пальцем, но при использовании двух пальцев возникают проблемы. Для объяснения сути проблемы мы воспользуемся терминологией инфракрасного сенсорного экрана, но у резистивных экранов возникает та же проблема. Представьте, что два пальца нажимают на экран в точках (3,3) и (8,8). В результате прерываются вертикальные лучи x=3 и x=8, как и горизонтальные лучи y=3 и y=8.

Теперь рассмотрим другую ситуацию: пользователь нажимает на экран в точках (3,8) и (8,3) – противоположных углах прямоугольника с углами (3,3), (8,3), (8,8) и (3,8). При этом блокируются те же самые лучи, а программа не может определить, с какой из двух ситуаций она имеет дело. Эта проблема называется **двоением**.

Для обнаружения одновременных нажатий несколькими пальцами (свойство, необходимое для распознавания жестов сведения/разведения) потребовалась новая технология. В большинстве смартфонов и планшетных компьютерах (но не на цифровых камерах и других устройствах!) чаще всего используются **проекционно-емкостные** сенсорные экраны. Они тоже делятся на несколько разновидностей, наиболее распространенной из которых является **взаимно-емкостная**. Все сенсорные экраны, способные одновременно распознавать две и более точки контакта, называются **мультитач**-экранами. Давайте в общих чертах посмотрим, как они работают.

Для читателей, забывших школьный курс физики: конденсатор – устройство, способное накапливать электрический заряд. Простой конденсатор состоит из двух электродов в форме пластин, разделенных слоем диэлектрика. В современных сенсорных экранах сетка тонких «проводов», проходящих вертикально, отделяется от горизонтальной сетки тонким изолирующим слоем. Когда палец прикасается к экрану, он изменяет емкость всех затронутых пересечений (возможно, находящихся далеко друг от друга). Это изменение можно измерить. Чтобы убедиться в том, что современные сенсорные экраны отличаются от старых

инфракрасных и резистивных экранов, попробуйте прикоснуться к экрану ручкой, карандашом, скрепкой или пальцем в перчатке – вы увидите, что ничего не происходит. Тело человека хорошо накапливает электрический заряд – каждый, кто в сухой холодный день вытирал ноги о коврик, а потом прикасался к металлической дверной ручке, знает это на собственном опыте. Пластмассовые, деревянные и металлические инструменты не могут сравниться с человеком в отношении своей емкости.

«Проводники» в сенсорном экране не похожи на обычные медные провода из обычных электрических устройств – они бы закрывали свет от экрана. Вместо них используются тонкие (обычно 50 микрон) полоски прозрачного резистивного сплава оксида индия и оксида олова, прикрепленные к обратным сторонам тонкой стеклянной панели. В совокупности они образуют конденсатор. В некоторых новых конструкциях диэлектрическая стеклянная панель заменяется тонким слоем диоксида кремния (песка!). В любом случае конденсаторы защищаются от грязи и царапин стеклянной пластиной, образующей поверхность экрана. Чем тоньше стеклянная пластина, тем чувствительней экран, но и тем меньше прочность устройства.

В процессе работы устройства напряжение подается поеременно на горизонтальные и вертикальные проводники, в то время как с других проводников читаются значения напряжения, изменившиеся под воздействием емкости пересечения. Эта операция повторяется много раз за секунду, а координаты точки прикосновения передаются драйверу устройства в виде потока пар (x,y). Дальнейшая обработка (например, определение простого нажатия, жестов сведения/разведения или скольжения) выполняется операционной системой. Если вы используете все 10 пальцев, да еще позовете друга на помощь, операционной системе придется изрядно поломать голову, но мультитач-оборудование справится со своей задачей.

1.12.3. Мониторы

В первых компьютерных мониторах использовались **электронно-лучевые трубки** (**ЭЛТ**), как в старых телевизорах [2]. Они были слишком громоздкими и тяжелыми для использования в портативных компьютерах, поэтому для экранов портативных компьютеров требовалась совершенно другая технология. Развитие плоских (flat-panel) мониторов позволило реализовать компактный форм-фактор, необходимый для ноутбуков, к тому же эти устройства использовали меньше энергии. В наши дни преимущества плоских экранов привели практически к полному вымиранию ЭЛТ-мониторов.

Самой распространенной технологией плоских мониторов является жидкокристаллический дисплей. Соответствующая технология чрезвычайно сложна, имеет несколько вариантов воплощения и быстро меняется, тем не менее мы постараемся сделать ее описание по возможности кратким и простым.

Жидкие кристаллы представляют собой вязкие органические молекулы, которые двигаются как молекулы жидкостей, но при этом имеют структуру, как у кристалла. Они были открыты австрийским ботаником Рейницером (Rheinitzer) в 1888 году и впервые стали применяться при изготовлении разнообразных дисплеев (для калькуляторов, часов и т.п.) в 1960 году. Когда молекулы расположены в одну линию, оптические качества кристалла зависят от направления и поляризации воздействующего света. При использовании электрического поля линия молекул, а следовательно, и оптические свойства, меняются. Если воздействовать лучом света на жидкий кристалл, интенсивность света, исходящего из самого жидкого кристалла, может контролироваться с помощью электричества. Это свойство используется при создании индикаторных дисплеев.

Экран жидкокристаллического дисплей состоит из двух стеклянных параллельно расположенных пластин, между которыми находится герметичное пространство с жидким кристаллом. К обеим пластинам присоединяются прозрачные электроды. Искусственный или естественный свет за задней пластиной освещают экран изнутри. Электроды, подведенные к пластинам, используют для того, чтобы создать электрические поля в жидком кристалле. На различные части экрана воздействует разное напряжение, что и позволяет строить изображение. К передней и задней пластинам экрана приклеиваются поляроиды, поскольку технологически дисплей требует поляризованного света. Общая структура показана на рисунке 1.35а

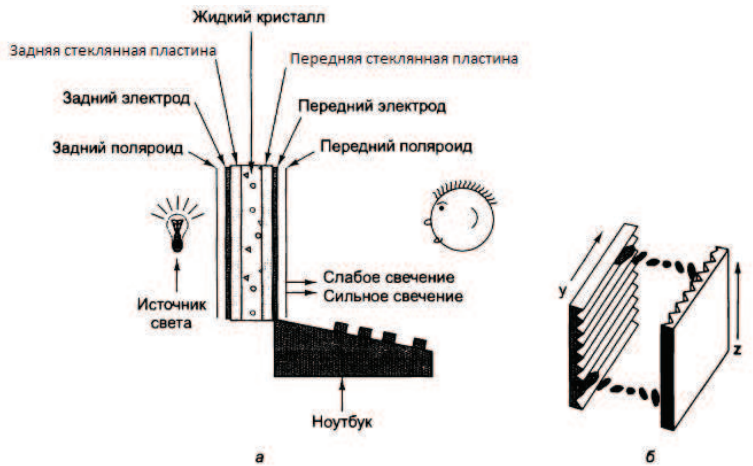


Рисунок 1.35 – Структура экрана на жидких кристаллах (а); желобки на передней и задней пластинах, расположенные перпендикулярно друг другу (б)

В настоящее время используются различные типы жидкокристаллических дисплеев, но мы рассмотрим только один из них – дисплей со скрученным нематиком (Twisted Nematic, TN). В этом дисплее на задней пластине находятся крошечные горизонтальные желобки, а на передней – крошечные вертикальные желобки, как показано на рис 1.35, б. При отсутствии электрического поля малекулы направляются к этим желобкам. Так как они (желобки) расположены перпендикулярно друг к другу, молекулы жидкого кристалла оказываются скрученными на 90°.

На задней пластине дисплея находится горизонтальный поляроид. Он пропускает только горизонтально поляризованный свет. На передней пластине дисплея находится вертикальный поляроид. Он пропускает только вертикально поляризованный свет. Если бы между пластинами не было жидкого кристалла горизонтально поляризованный свет, пропущенный поляроидом на задней пластине, блокировался бы поляроидом на передней пластине, что делало бы экран полностью черным.

Однако крученная кристаллическая структура молекул, сквозь которую проходит свет, меняет плоскость поляризации света. При отсутствии электрического поля весь жидкокристаллический экран светится с однородной яркостью. Если подавать напряжение к определенным частям пластины, скрученная структура разрушается, блокирую прохождение света в этих частях.

Для подачи напряжения обычно используется два подхода. В дешевом **пассивном матричном индикаторе** на обоих электродах провода располагаются параллельно друг другу. Например, на дисплее размером 640x480 электрод задней пластины содержит 680 вертикальных проводов, а электрод передней пластины – 480 горизонтальных проводов. Если подавать напряжение на один из вертикальных проводов, а затем посылать импульсы на один из горизонтальных, можно изменить напряжение в определенной позиции пиксела и, таким образом, сделать нужную точку темной. Если то же самое повторить со следующим пикселом и т.д., можно получить темную строку развертки, аналогичную строкам в электронно-лучевых трубках. Обычно изображение на экране перерисовывается 60 раз в секунду, чтобы создавалось впечатление постоянной картинки (так же, как в электронно-лучевых трубках).

Второй подход – применение **активного матричного индикатора**. Он стоит гораздо дороже, чем пассивный, но зато дает изображение лучшего качества, что является большим преимуществом. Вместо двух наборов перпендикулярно расположенных проводов у активного матричного индикатора на одном из электродов имеется крошечный переключатель в каждой позиции пиксела. Меняя состояние переключателей, можно создавать на экране произвольную комбинацию напряжений в зависимости от комбинации битов. Эти переключатели называются **тонкопленочными транзисторами** (Thin Film Transistor, TFT), а плоские экраны, в которых они используются, - **TFT – дисплеями**. На основе технологии TFT теперь

производится подавляющее большинство ноутбуков и автономных жидкокристаллических мониторов.

До сих пор мы описывали, как работают монохромные мониторы. Что касается цветных мониторов, достаточно сказать, что они работают на основе тех же общих принципов, что и монохромные, но детали гораздо сложнее. Чтобы разделить белый цвет на красный, зеленый и синий, в каждой позиции пиксела используются оптические фильтры, поэтому эти цвета могут отображаться независимо друг от друга. Из сочетания этих трех основных цветов можно получить любой цвет.

Технология изготовления IPS или SFT матрицы представляет собой более развитую и модернизированную TFT. Угол обзора увеличен и у лучших изделий достигает 178 градусов, а цветовой охват практически идентичен естественному.

Виды IPS матрицы:

-H-IPS – повышает контраст изображения и снижает время отклика.

-AS-IPS – основное качество заключается в повышении контрастности.

-H-IPS A-TW – H-IPS с технологией «True White», которая улучшает белый цвет и его оттенки.

-AFFS – увеличение напряженности электрического поля для больших углов обзора и яркости.

PLS матрица – доработанная, с целью снижения себестоимости и оптимизации времени отклика, версия IPS. Разработана концерном Самсунг и является аналогом H-IPS, AH-IPS, которые запатентованы другими разработчиками.

Сегодня используется технология органических светодиодов OLED (Organic Light Emitting Diode). Такие экраны состоят из слоев электрически заряженных органических молекул, помещенных между двумя электродами. Изменения напряжения заставляют молекулы переходить на более высокие энергетические состояния. При возвращении к нормальному состоянию они излучают свет.

Для мобильных устройств применяется AMOLED (Active Matrix Organic Light-Emitting Diode)- комбинация LED и TFT.

Достоинства:

-маленький вес и габариты;

-низкое энергопотребление;

-неограниченные геометрические формы;

-не нужна подсветка специальной лампой;

-мгновенный отклик матрицы;

-контрастность превышает все известные альтернативные технологии;

-возможность создания гибких экранов;

-температурный диапазон шире, чем у других экранов.

Недостатки:

-маленький срок службы диодов определенного цвета;

-невозможность создания долговечных полноценных дисплеев;
-очень высокая цена, даже по сравнению с IPS.

1.12. ВИДЕОПАМЯТЬ

Обновление картинки на экранах ЭЛТ и TFT-мониторов производится от 60 до 100 раз в секунду; для этого используется видеопамять, размещенная на плате контроллера дисплея [2]. Видеопамять содержит одну или несколько битовых карт, представляющих выводимое на экран изображение. Если, скажем, на экране имеется 1920x1080 элементов изображения (**пикселов**), значит, в видеопамяти содержится 1920x1080 значений, по одному на каждый пиксел. В целях быстрого переключения с одного изображения на другое в памяти может размещаться несколько таких карт.

В современных дисплеях каждый пиксел представлен 3-байтным значением RGB, которое определяет интенсивность красного (Red), зеленого (Green) и синего (Blue) компонентов изображения (мощные профессиональные мониторы используют 10 и более бит на цвет). Как известно, любой цвет можно представить путем линейной суперпозиции трех упомянутых базовых цветов.

Если в видеопамяти хранится информация о 1920x1080 пикселах, причем на каждый из них выделяется по 3 байта, общий объем этих данных составляет около 6,2 Мбайт; поэтому на любые манипуляции таким изображением уходит довольно много процессорного времени. По этой причине в некоторых компьютерах для определения цвета используются 8-разрядные числа. Такое число представляет собой индекс аппаратной таблицы (так называемой **цветовой палитры**), состоящей из 256 значений RGB (24-разрядных). Это решение, известное под названием **индексированного цвета**, позволяет на 2/3 сократить объем данных, хранящихся в видеопамяти. В то же время, при изменении индексированного цвета в каждый конкретный момент на экран не может выводиться более 256 цветов. Как правило, для каждого окна формируется индивидуальная битовая карта, а это значит, что при наличии одной аппаратной палитры из всех присутствующих на экране окон корректно визуализируется только одно. Также применяются палитры с 2^{16} элементами, но в этом случае выигрыш по занимаемой памяти составляет всего 1/3.

Для вывода растровых (то есть сформированных на основе битовых карт) изображений требуется большая пропускная способность. К примеру, для воспроизведения одного кадра полноцветных мультимедийных данных в полноэкранном формате на дисплее размером 1920x1080 необходимо скопировать в видеопамять 6,2 Мбайт. Если учесть, что полноценный видеофильм выводится со скоростью 25 кадров в секунду, общая скорость передачи данных должна составлять 155 Мбайт/с. Такую пропускную

способность не способна обеспечить даже первоначальная версия шины PCI (132 Мбайт/с), но шина PCIe легко справляется с ней.

1.14. МЫШИ

Время идет, и за компьютер садятся те, кто разбирается в нем все меньше и меньше [2]. Компьютеры серии ENIAC использовались только теми, кто их разрабатывал. В 50-е годы с компьютерами работали лишь высококвалифицированные программисты. Сейчас многие из тех, кто работает за компьютером, не знают (и не хотят знать) ни как функционирует компьютер, ни как он программируется.

Много лет назад у большинства компьютеров был интерфейс командной строки, в которой набирались различные команды. Поскольку многие неспециалисты считали такие интерфейсы недружественными или даже враждебными, компьютерные фирмы разработали специальные интерфейсы с возможностью указания определенной позиции на экране с помощью специального устройства (как в Macintosh и Windows), которым чаще всего является мышь.

Мышь – это устройство в маленьком пластиковом корпусе, располагающееся на столе рядом с клавиатурой. Если двигать мышь по столу, указатель на экране тоже будет двигаться, что дает возможность навести его на тот или иной элемент экрана. У мыши есть одна, две или три кнопки, нажатие которых дает возможность пользователям выбирать пункты меню. Было очень много споров по поводу того, сколько кнопок должно быть у мыши. Начинающим пользователям достаточно было одной кнопки (в этом случае перепутать кнопки не возможно), но их более опытные коллеги предпочитали несколько кнопок, чтобы можно было на экране выполнять сложные действия.

Существует три типа мышей: механические, оптомеханические и оптические. У мышей первого типа снизу располагаются резиновые колесики, оси которых расположены перпендикулярно друг другу. Если мышь передвигается в вертикальном направлении, то вращается одно колесо, а если в горизонтальном, то другое. Каждое колесико приводит в действие резистор (потенциометр). Если измерить изменения сопротивления, можно узнать, на сколько провернулось колесико, и таким образом вычислить, на какое расстояние передвинулась мышь в каждом направлении. Такие мыши практически полностью были вытеснены следующей моделью, в которой вместо колес используется шарик, слегка выступающий снизу (рис. 1.36).

Следующий тип – оптомеханическая мышь. У неё есть шарик, который вращает два колесика, расположенные перпендикулярно друг к другу. Колесики связаны с кодировщиками. В каждом кодировщике имеются прорези, через которые проходит свет. Когда мышь двигается, колесики вращаются, и световые импульсы воздействуют на детекторы каждый раз, когда между светодиодом и детектором появляется прорезь. Число воспринятых детектором импульсов пропорционально расстоянию.

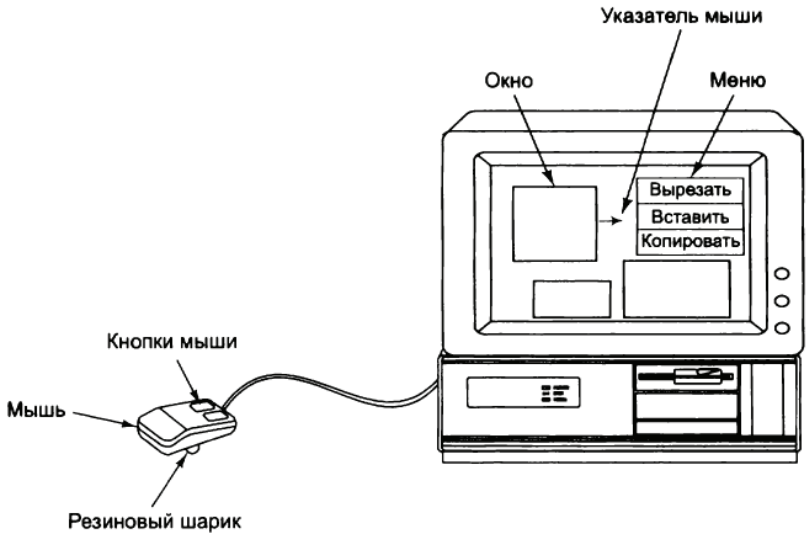


Рисунок 1.36 – Использование мыши для выбора пункта меню

Следующий тип – оптическая мышь. У неё нет ни колес, ни шарика. Вместо этого в нижней части мыши располагается светодиод и фотодетектор. Первые модели оптических мышей перемещались по поверхности особого пластикового коврика, который содержит прямоугольную решетку с линиями, близко расположенными друг к другу. Когда мышь двигается по решетке, фотодетектор воспринимает пересечения линий за счет изменения в количестве света, отражаемого от светодиода. Электронное устройство внутри мыши подсчитывает количество пересеченных линий в каждом направлении. Современные оптические мыши содержат светодиод, освещающий неоднородности нижележащей поверхности, и крошечную видеокамеру, которая снимает изображение (как правило, размером 18x18 пикселов) до 1000 раз в секунду. Сравнение соседних изображений определяет, как далеко переместилась мышь. Некоторые оптические мыши используют для освещения лазер вместо светодиода. Они обеспечивают большую точность, но и стоят дороже.

Хотя мыши могут быть устроены по разному, обычно используется следующая схема: компьютеру передаётся последовательность из 3 байт каждый раз, когда мышь проходит определенное минимальное расстояние (например, 0,01 дюйма). Обычно эти характеристики передаются в последовательном потоке битов. Первый байт содержит целое число,

которое указывает, на какое расстояние переместилась мышь в направлении X с прошлого раза. Второй байт содержит ту же информацию для направления У. Третий байт указывает на текущее состояние кнопок мыши. Иногда для каждой координаты используются 2 байта.

Низкоуровневое программное обеспечение принимает эту информацию по мере поступления и преобразует относительные движения, передаваемые мышью, в абсолютную позицию. Затем оно отображает стрелочку на экране в позиции, соответствующей расположению мыши. Если указать стрелочкой на определенный элемент экрана и щелкнуть кнопкой мыши, компьютер может вычислить, какой именно элемент на экране выбран.

1.15. ШЛЕМЫ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ

Наголовный дисплей, также распространенный вариант шлем виртуальной реальности или очки виртуальной реальности – устройство, позволяющее частично погрузиться в мир виртуальной реальности, создающее зрительный и акустический эффект присутствия в заданном управляющим устройством (компьютером) пространстве [7]. Представляет собой конструкцию, надеваемую на голову, снабженную видеоэкраном и акустической системой.

Наголовный дисплей создает объемную картинку, демонстрируя два изображения – по одному для каждого глаза. Кроме того, он может содержать гироскопический или инфракрасный датчик положения головы.

Типичные шлемы (очки) виртуальной реальности используют один или два дисплея с линзами и, иногда с зеркалами. В качестве дисплеев могут использоваться миниатюрные электронно-лучевые приборы, ЖК-дисплеи, LCos-проекторы, органические светодиоды. Иногда могут использоваться несколько микродисплеев для увеличения поля зрения. В некоторых системах используются дисплеи обычного смартфона.

Часто наголовные дисплеи делят на два класса по способности комбинировать искусственное изображение с реальным:

-Большинство дисплеев может отображать лишь искусственное (виртуальное) изображение.

-Некоторые дисплеи позволяют комбинировать реальное и виртуальное изображение, реализуя дополненную реальность или смешанную реальность. Комбинирование может происходить за счет полупрозрачных зеркал или с помощью видеокамер, снимающих реальность, и использование этого видеопотока при генерации изображения.

Дисплеи виртуальной реальности могут использоваться для стереоскопического отображения информации при работе с системами автоматизированного проектирования, при ремонте сложных систем. Применяются в хирургии для изучения томографических снимков (компьютерная томография, магнитно-резонансная томография).

3 ноября 2011 года компания Recon Instruments выпустила MOD Live – первый в мире интерактивный дисплей для горнолыжных очков под управлением Android. На дисплей можно выводить скорость, высоту, дистанцию, время в прыжке, его длину и высоту, местоположение по GPS, температуру и многое другое.

Наголовный дисплей виртуальной реальности позволяет разместить стажёра в ситуации, которая слишком дорога или слишком опасна для повторения в реальной жизни. Обучение охватывает широкий спектр тренировок от вождения, прыжков с парашютом, сварки, полётов и тренировок солдат до подготовки медицинских процедур. Тем не менее ряд нежелательных симптомов был вызван длительным использованием определённых типов дисплеев на голове

1.16. 3D СКАНЕРЫ

3D-сканер – периферийное устройство, анализирующее форму предмета и на основе полученных данных создающее его 3D-модель.

Технология трехмерного сканирования появилась всего несколько десятилетий назад, в конце 20-го века. Первый работающий прототип появился в 60-х годах. Конечно, тогда он не мог похвастаться широким спектром возможностей, однако это был настоящий 3D-сканер, неплохо справляющийся с основной функцией [8].

В середине 80-х годов сканирующие устройства усовершенствовали. Их начали дополнять лазерами, источниками белого света и затемнения. Благодаря этому удалось улучшить «захват» исследуемых объектов. В этот период появляются контактные датчики. С их помощью оцифровывалась поверхность твердых предметов, которые не отличались сложной формой. Чтобы усовершенствовать оборудование, разработчикам пришлось позаимствовать ряд оптических технологий из военной промышленности.

Применение 3D-сканеров было интересно не только конструкторам дизайн-студий, автомобильных концернов, но и работникам киноиндустрии. В 80-х - 2000-х годах разные компании выпускали свои модели оборудования: Head Scanner, 3D-сканер REPLICA и другие. С тех времен агрегаты изменились, усовершенствовались, стали более мобильными и функциональными.

Устройство 3d сканера занимается детальным исследованием физических объектов, после чего воссоздаются их точные модели в цифровом формате. Современные агрегаты могут быть стационарными или мобильными. В качестве подсветки применяется лазер или особая лампа (их использование увеличивает точность измерений).

Принцип действия 3D-сканера показан на рисунках 1.37 и 1.38.

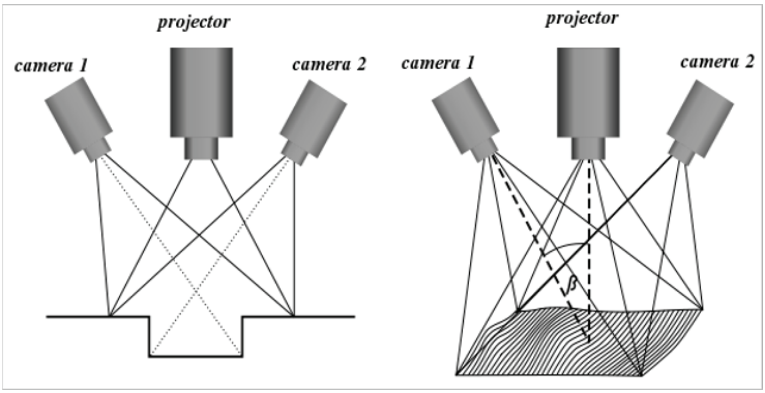


Рисунок 1.37 – Принцип действия 3D-сканера

Принцип работы 3D-сканера определяется технологией сканирования. При помощи подсветки и встроенных камер аппарат измеряет расстояние до объекта с разных ракурсов. Затем сопоставляются картинки, передаваемые камерами. После тщательного анализа всех полученных данных, на экране отображается готовая цифровая трехмерная модель. Если устройство 3D-сканера основано на работе лазерного луча, то с его помощью измеряются расстояния в заданных точках. На основе этих сведений выводятся координаты.

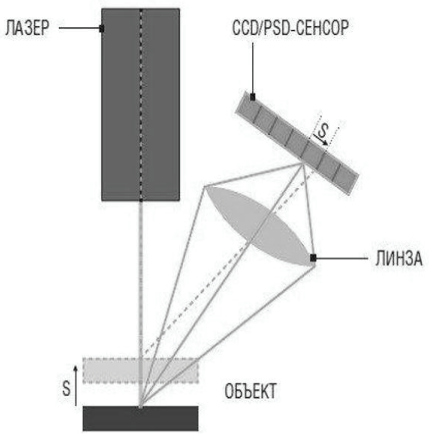


Рисунок 1.38 – Принцип работы 3D-сканера

Выделяются два основных **метода** сканирования:

1) **Контактный**. Устройство зондирует предмет посредством физического контакта, пока объект находится на прецизионной

поверочной плите. Контактный 3D-сканер отличается сверхточностью работы. Правда, при сканировании можно повредить или изменить форму объекта.

2) **Бесконтактный**. Применяется излучение или особый свет (ультазвук, рентгеновские лучи, лазер). В данном случае предмет сканируется через отражение светового потока.

Технология сканирования бывает:

1) **Лазерная**. Функционирование устройств основывается на принципе работы лазерных дальномеров. Лазерные 3D-сканеры характеризуются точностью получаемой трехмерной модели. Правда их применение затруднительно в условиях подвижности объекта. Это больше 3D-сканер для помещения. Сканирование человека 3D-сканером лазерного типа практически невозможно.
2) **Оптическая**. В данном случае применяется специальный лазер второго класса безопасности. Оптический 3D-сканер отличается большой скоростью сканирования. Его использование исключает любое искажение, даже если объект будет двигаться. Также нет необходимости в нанесении отражающих меток. Правда такие устройства не подходят для исследования зеркальных, прозрачных или блестящих изделий. Зато это отличный вариант 3D-сканера человека.

Промышленный 3D-сканер полезен в:

-инженерии;
-медицине;
-производстве;
-дизайне;
-киноиндустрии;
-сфере создания компьютерных игр.

Особое внимание хотелось уделить ультразвуковому 3D-сканеру. Он является настоящей находкой для современной медицины. Устройства снабжаются энергетическими, цветными, тканевыми, нерперывноволновыми и импульсными допплерами. Данный агрегат характеризуется высочайшей разрешающей способностью, поэтому популярен в маммологии, акушерстве, урологии, исследовании сосудов и мышечных тканей, эхокардиографии, неонаталогии, педиатрии.

По принципу работы устройства также отличаются. Рынок предлагает стационарный или переносной, то есть ручной 3В-сканер. В качестве сенсора во втором случае используется координатно-чувствительный детектор или аппарат с зарядовой связью. Данный агрегат чрезвычайно удобен тем, что его можно свободно перемещать. Портативный 3D-сканер идеально подходит для сканирования труднодоступных мест или крупногабаритных объектов. Измерение можно проводить под любыми углами, вокруг или под исследуемыми предметами.

Устройства используются совместно с разным оборудованием. Это может быть не только 3D-сканер для 3D-принтера, но и 3D-сканер для ipad. Современные производители подобных агрегатов выпускают мобильные устройства, которые работают не только со стационарными компьютерами, но и с планшетами или даже смартфонами. Кроме этого существуют специальные программы, с помощью которых обычные телефоны превращаются в сканеры. К примеру, можно найти 3D-сканер для андроид. Он поможет конструировать уникальные детали, проводить быстрое прототипирование и оцифровку объектов.

1.17. ПРИНТЕРЫ

Рано или поздно пользователю потребуется напечатать созданный документ или страницу, полученную из Интернета, поэтому компьютеры могут быть оснащены принтером. В этом разделе мы опишем некоторые наиболее распространенные типы монохромных (то есть черно-белых) и цветных принтеров.

1.17.1. Лазерные принтеры

Вероятно, самым удивительным изобретением в области печатных технологий со времен Йоганна Гуттенберга (Johann Gutenberg), который изобрел подвижную литеру в XV веке, является лазерный принтер [2]. Это устройство сочетает хорошее качество печати, универсальность, высокую скорость работы и умеренную стоимость. В лазерных принтерах используется почти та же технология, что и в фотокопировальных устройствах. Многие компании производят устройства, совмещающие свойства копировальной машины, принтера и иногда факса.

Схематически устройство принтера показано на рисунке 1.39 [2]. Главной частью этого принтера является вращающийся барабан (в некоторых более дорогостоящих системах вместо барабана используется лента). Перед печатью каждого листа барабан получает напряжение около 1000 вольт и окружается фоточувствительным материалом. Свет лазера проходит вдоль барабана (по длине), почти как пучок электронов в электронно-лучевой трубке, только вместо напряжения для сканирования барабана используется вращающееся восьмиугольное зеркало. Луч света модулируется, в результате получается набор темных и светлых участков. Участки, на которые воздействует луч, теряют свой электрический заряд.

После того как нарисована строка точек, барабан немного поворачивается для следующего создания строки. В итоге первая строка точек достигает резервуара с тонером (электростатическим черным порошком). Тонер притягивается к заряженным точкам, и так формируется визуальное изображение строки.

Через некоторое время барабан с тонером прижимается к бумаге, оставляя на ней отпечаток изображения. Затем лист проходит через нагретые валики, и изображение закрепляется.

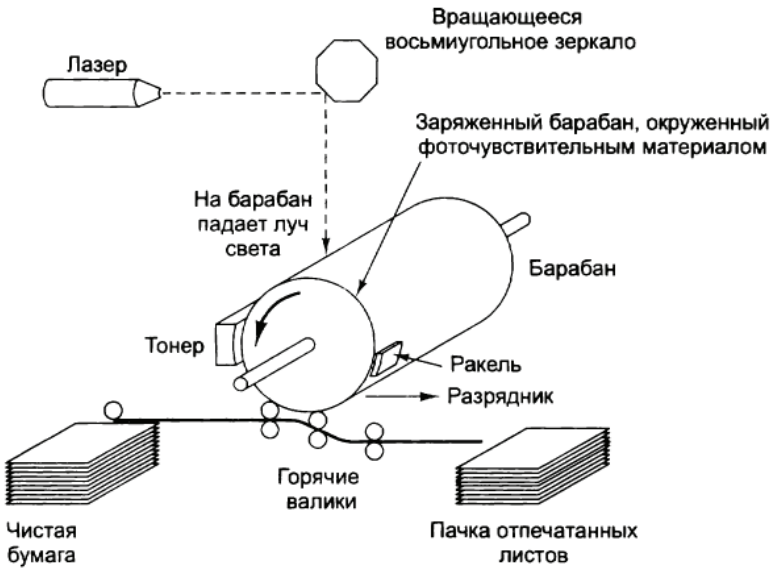


Рисунок 1.39 – Схема работы лазерного принтера

После этого барабан разряжается и остатки тонера счищаются с него. Теперь он готов к печати следующей страницы.

Едва ли нужно говорить, что этот процесс представляет собой чрезвычайно сложную комбинацию приемов, требующих знания физики, химии, механика и оптики. Впрочем, некоторые фирмы предлагают готовые модули, называемые блоками печати (print engines). Производители лазерных принтеров дополняют блоки печати собственной электроникой и программным обеспечением. Электроника лазерных принтеров состоит из быстродействующего процессора и нескольких мегабайтов памяти для хранения полного изображения в битовой форме и различных шрифтов, одни из которых встроены, а другие загружаются из памяти. Большинство принтеров получают команды, описывающие печатаемую страницу (в противоположность принтерам, получаемым изображения в битовой форме от центрального процессора). Эти команды обычно даются на языке PCL от HP или PostScript от Adobe – полноценных, хотя и специализированных, языках программирования.

Лазерные принтеры с разрешающей способностью 600 dpi и выше могут не читать черно-белые фотографии, но технология при этом гораздо сложнее, чем может показаться на первый взгляд. Рассмотрим фотографию, отсканированную с разрешающей способностью 600 dpi, которую нужно напечатать на принтере с такой же разрешающей способностью (600 dpi).

Сканированное изображение содержит 600 х 600 пикселов на дюйм, каждый пиксел характеризуется определенной градацией серого цвета от 0 (белый цвет) до 255 (черный цвет). Принтер может печатать с разрешающей способностью 600 dpi, но каждый напечатанный пиксел может быть либо черного цвета (когда есть тонер), либо белого цвета (когда нет тонера). Градации серого печататься не могут.

При печати таких изображений имеет место так называемая **обработка полутонов** (как при печати серийных плакатов). Изображение разбивается на ячейки, каждая по 6х6 пикселов. Каждая ячейка может содержать от 0 до 36 черных пикселов. Человеческому глазу ячейка с большим количеством черных пикселов кажется темнее, чем ячейка с небольшим количеством черных пикселов. Серые тона в диапазоне от 0 до 255 передаются следующим образом. Этот диапазон делится на 37 зон. Серые тона от 0 до 6 расположены в зоне 0, от 7 до 13 – в зоне 1, и т.д. (зона 36 немного меньше, чем другие, потому что 256 на 37 без остатка не делится). Когда встречаются тона зона 0, ячейка оставляется белой, как показано на рисунке 1.40, а. Тона зоны 1 передаются одним пикселом в ячейке. Тона зоны 2 – двумя пикселами в ячейке, как показано на рисунке 1.40,б. Изображения серых тонов других зон показаны на рисунке 1.40, в-е.

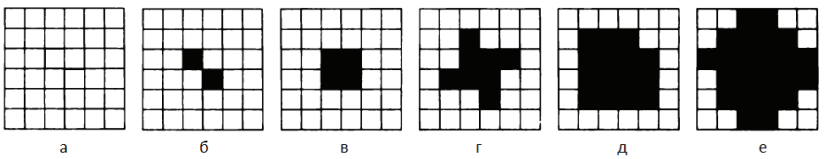


Рисунок 1.40 – Изображение серых полутонов различных зон: 0-6 (а); 14-20 (б); 28-34 (в); 56-62 (г); 105-111 (д); 161-167 (в)

Если фотография отсканирована с разрешающей способностью 600 dpi, после подобной обработки полутонов фактическая разрешающая способность напечатонного изображения снижается до 100 ячеек на дюйм. Данная величина называется градацией полутонов и меряется в lpi (lines per inch – строк на дюйм).

1.17.2. Цветные принтеры

Хотя лазерные принтеры чаще всего являются монохромными, цветные принтеры получают все более широкое распространение, поэтому о цветной печати тоже стоит сказать пару слов (причем все сказанное также относится к струйным и другим принтерам) [2]. Цветные изображения могут строиться двумя способами: испусканием света и отражением света. Испускание света имеет место, например, при создании изображений в электронно-лучевых мониторах. В данном случае изображение строится путем аддитивного наложения трех основных цветов: красного, зеленого и синего.

Отраженный свет используется при создании цветных фотографий и картинок в глянцевых журналах. В этом случае поглощается свет с определенной длиной волны, а остальной свет отражается. Такие изображения создаются путем субтрактивного наложения трех основных цветов: голубого (красный полностью поглощен), жёлтого (синий полностью поглощен) и сиреневого (зеленый полностью поглощен). Теоретически путем смешения голубых, жёлтых и сиреневых чернил можно получить любой цвет. Но на практике очень сложно получить такие чернила, которые полностью поглощали бы весь свет и в результате давали черный цвет. По этой причине практически во всех цветных печатающих устройствах используются чернила четырёх цветов: голубого (Cyan), сиреневого (Magenta), желтого (Yellow) и черного (black). Такая цветовая модель называется CMYK (из слова «black» берется последняя буква, чтобы отличать его от слова «blue» в модели RGB). Мониторы, напротив, для создания цветного изображения используют испускаемый свет и наложение красного, зеленого и синего цветов.

Полный набор цветов, который может производить монитор или принтер, называется **цветовой шкалой**. Не существует такого устройства, которое полностью передавало бы цвета окружающего нас мира. В лучшем случае устройство дает всего 256 степеней интенсивности каждого цвета, и в итоге получается только 16 777 216 различных цветов. Несовершенство технологий ещё больше сокращает это число, а оставшиеся цвета не дают полного цветового спектра. Кроме того, цветовосприятие связано не только с физическими свойствами света, но и с работой «палочек» и «колбочек» в сетчатке глаза.

Из всего этого следует, что превратить красивое цветное изображение, которое замечательно смотрится на экране, в идентичное печатное изображение очень сложно. Среди основных проблем можно назвать следующие:

- цветные мониторы используют поглощенный свет; цветные принтеры – отраженный;
- электронно-лучевая трубка дает 256 оттенков каждого цвета, цветные принтеры должны обеспечивать обработку полутонов;
- мониторы имеют темный фон; бумага – светлый;
- цветовая модель RGB монитора и модель CMYK принтера отличаются друг от друга.

Чтобы цветные печатные изображения соответствовали реальной действительности (или хотя бы изображениям на экране), необходима калибровка оборудования, сложное программное обеспечение и компетентность пользователя.

1.17.3. Струйные принтеры

Всем удобно использовать недорогие струйные принтеры. В таком принтере подвижная печатающая головка содержит картридж с чернилами. Она двигается горизонтально над бумагой, а чернила в это время

выпрыскиваются из крошечных сопел. Объем одной порции чернил приблизительно равен один пиколитр. Для наглядности уточним, что в одной капле воды может уместиться около 100 миллионов таких порций.

Струйные принтеры бывают двух типов: пьезоэлектрические и термографические. В пьезоэлектрических струйных принтерах рядом с чернильной камерой устанавливается специальный кристалл. При подаче на этот кристалл напряжения он деформируется, в результате из форсунки выпускаются чернила. Чем выше напряжение, тем больше выходная порция чернил, причем управление этим процессом производится программно.

В термографических (пузырьковых) струйных принтерах в каждой форсунке устанавливается небольшой резистор. При подаче напряжения резистор быстро нагревается, доводит температуру чернил до точки кипения, в результате последние превращаются в пузырьки газа. Поскольку объем пузырька больше объема простых чернил, в форсунке создается повышенное давление, под влиянием которого чернила распыляются на бумагу. Затем форсунка охлаждается, и в результате снижения давления внутри форсунки в неё из картриджа подается новая порция чернил. Скорость работы принтера в рамках этой схемы ограничена временными циклами кипения/охлаждения. Размер всех формируемых чернильных капель одинаков, причем, как правило, он уступает аналогичному показателю пьезоэлектрических принтеров.

Струйные принтеры обычно имеют разрешающую способность от 1200 dpi (dots per inch – точек на дюйм) до 4800 dpi. Они достаточно дешевы, работают бесшумно, однако отличаются низкой скоростью печати и дороговизной картриджей. Качество печати хорошее – если распечатать фотографию с высоким разрешением на ведущей модели любой линейки струйных принтеров, результат будет не отличить от обычной фотографии 8 x 10.

Для получения лучших результатов должны использоваться особые чернила и особая бумага. Существуют два вида чернил. **Чернила на основе красителя** состоят из красителей, растворенных в жидкой среде. Они дают яркие цвета и легко вытекают из картриджа. Главным недостатком таких чернил является то, что они быстро выгорают под воздействием ультрафиолетовых лучей, которые содержатся в солнечном свете. **Чернила на основе пигмента** содержат твердые частицы пигмента, погруженные в жидкость. Жидкость испаряется с бумаги, а пигмент остается. Чернила не выгорают, но зато дают не такие яркие краски, как чернила на основе красителя. Кроме того, частицы пигмента часто засоряют выпускные отверстия картриджей, поэтому их нужно периодически чистить. Для печати фотографий необходима мелованная или глянцевая бумага. Эти особые виды бумаги созданы специально для того, чтобы удерживать капельки чернил и не давать им растекаться.

1.17.4. Термографические принтеры

Термографические принтеры содержат небольшую печатающую головку с множеством игольчатых элементов. При прохождении электрического тока иглы очень быстро нагреваются. Над печатающей головкой проходит специальная термочувствительная бумага, и в тех местах, где находятся нагретые иглы, появляются точки. В сущности, термографический принтер работает по принципу старого матричного принтера, в котором контакты через красящую ленту оставляли точки на бумаге. Термографические принтеры широко применяются для печати чеков в магазинах, банкоматах, автоматизированных заправках и т.д.

1.17.5. 3D принтеры

3D-принтер это станок с числовым программным управлением, использующий метод послойной печати детали [9].

Если для печати в обычном принтере используют тонер, то в 3D-принтере пользуются различными видами пластика, нейлоном, металлической пудрой, стеклянным порошком, строительными смесями и другими материалами [10]. Основа данной технологии – послойное построение твердых моделей. Этот способ идеально подходит для предметов различной сложности: от обычных детских игрушек до всевозможных элементов, используемых, например, в протезировании.

Принцип работы данной техники:

-создание компьютерной модели будущего объекта;

-деление полученного шаблона на множество поперечных слоев с помощью специального ПО;

-постепенное наращивание по направлению от основания вверх жидкого, порошкообразного, другого материала с последующим соединением (сплавлением) его в объект нужной формы.

Существует несколько технологий такой печати, отличающихся техникой работы, свойствами используемого исходного материала, используемым ПО:

1) Экструзионная печать. Суть этого метода заключается в воздействии экструдера на расходный материал. Он нагревает сырье до определенной температуры, затем выдавливая его через сопло, формирует изделие или его фрагменты. В роли расходников выступают различные виды полимеров.
2) Поршковая. Данная технология включает:
   –струйную печать, основанную на нанесении связующих материалов на тонкие слои порошка, с последующей пропиткой полимерами или воском;
   –выборочное или прямое спекание тонких слоев порошка с помощью лазера;
   –электронно-лучевое, лазерное сплавление – вместо спекания в местах соприкосновения с лазером происходит плавление порошка.

3) Ламинирование. Этот способ позволит значительно удешевить стоимость полученных изделий, так как использует в качестве сырья бумагу, листы из тонкого металла и пластика.
4) Фотополимеризация. Данная технология основана на использовании жидких фотополимерных смол, затвердевающих под влиянием ультрафиолетового света.

В основном принтеры трехмерной печати состоят из одинаковых деталей и по устройству похожи на обычные принтеры. Главное отличие – очевидное: 3D-принтер печатает в трех плоскостях, и кроме ширины и высоты появляется глубина.

Вот из каких деталей состоит 3D-принтер, не считая корпуса:

-экструдер, или печатающая головка – разогревает поверхность, с помощью системы захвата отмеряет точное количество материала и выдавливает полужидкий пластик, который подается в виде нитей;

-рабочий стол (его еще называют рабочей платформой или поверхностью для печати) – на нем принтер формирует детали и выращивает изделия;

-линейный и шаговый двигатели – приводят в движение детали, отвечают за точность и скорость печати;

-фиксаторы – датчики, которые определяют координаты печати и ограничивают подвижные детали. Нужны, чтобы принтер не выходил за пределы рабочего стола, и делают печать более аккуратной;

-рама – соединяет все элементы принтера [11].

На рисунке 1.41 показана схема 3D-принтера.

Все это управляется компьютером.

3D-принтеры стали незаменимыми в строительстве, медицине, пищевой промышленности, их используют во многих дизайнерских студиях.

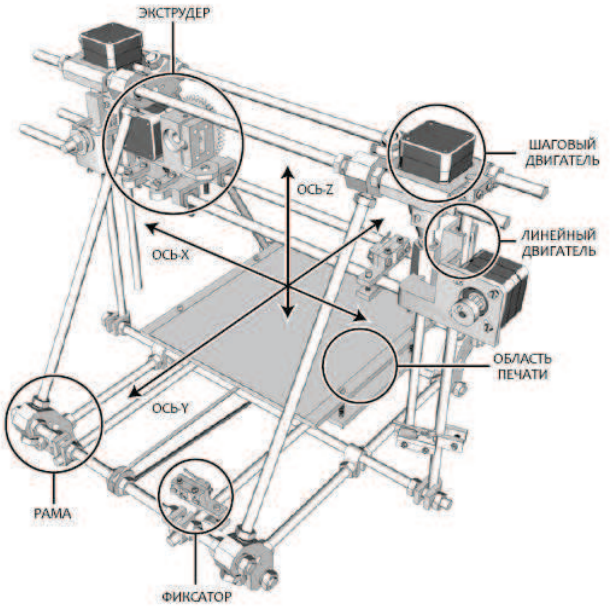


Рисунок 1.41 – Схема 3D-принтера

ВОПРОСЫ К ГЛАВЕ 1

1. Первая и вторая форма реализации общей системной магистрали ПЭВМ.
2. Суть программы интерпретатора.
3. Назначение и особенности применения ОЗУ.
4. Назначение и особенности применения ПЗУ.
5. В чем заключается технология гиперпоточности.
6. Особенности применения микроконтроллера 1887ВЕ7Т.
7. Ширина шины.
8. Централизованный арбитраж шины.
9. Децентрализованный арбитраж шины.
10. Особенности системы RISC.
11. Особенности системы CISC.
12. Принципы RISC.
13. Принцип конвейера.
14. Иерархическая структура памяти.
15. Какие методы сканирования существуют.
16. Какие существуют технологии сканирования.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Франке К.* Введение в микро-ЭВМ [Текст]: пер. с нем. / К. Франке. – М.: Энергоатомиздат, 1988. – 160 с.

2. *Таненбаум Э.* Архитектура компьютера [Текст] / Э. Таненбаум, Т. Остин. – 6-е изд. – СПб.: Питер, 2013. – 816 с.

3. *Лебедев О.Н.* Микросхемы памяти и их применение [Текст] / О.Н. Лебедев. – М.: Радио и связь, 1990. – 160 с.

4. *Тули М.* Справочное пособие по цифровой электронике [Текст]: пер. с англ. / М. Тули. – М.: Энергоатомиздат, 1990. – 176 с.

5. https://www.intel.ru

6. https://niiet.ru

7. https://ru.wikipedia.org

8. https://make-3d.ru

9. https://ru.wikipedia.org

10. https://technopanorama.ru

11. https://losprinters.ru

2. Логическое программирование

2.1. ОСНОВНЫЕ ПОНЯТИЯ

2.1.1 Алгоритмы и алгоритмические языки

По своей внутренней структуре алгоритмические языки подразделяются на процедурные и логические (декларативные). В процедурных языках алгоритм решения задачи записывается в виде последовательности шагов, каждый из которых приближает нас к требуемому результату, т.е. процесс решения задачи представляется в явной, прозрачной форме с использованием языковых конструкций.

В логических языках данные и знания описываются в виде набора формальных конструкций, представляющих собой с математической точки зрения систему аксиом. При этом любая задача формулируется как теорема, которую необходимо доказать. Сама процедура доказательства осуществляется автоматически, т.е. явное описание алгоритма решения задачи отсутствует.

Известный математик Пойа [3] определяет задачу следующим образом: “Задача предполагает необходимость сознательного поиска соответствующего средства для достижения непосредственно недоступной цели”. Задача может рассматриваться как цель в определенных условиях или как разделение известного и неизвестного. Задача определяется тремя компонентами: исходными данными, требуемым результатом и решением. Существует два весьма общих типа задач: задачи на нахождение и задачи на доказательство.

Задачи первого типа, назовем их расчетными, состоят в нахождении неизвестного заранее объекта, удовлетворяющего условиям, связывающим его с исходными данными. Этот объект может принадлежать к самым разнообразным категориям, определяющим множество конкретных объектов, а условие задачи выделяет их подмножество. Каждый объект, принадлежащий этому подмножеству, называется решением. Таким образом, в задачах первого типа мы имеем дело с определением решения задачи не как процедуры, а как результата. Исходными данными и конечными результатами решения таких задач, как правило, являются наборы числовых величин. Программы для решения задач первого типа составляются на процедурных алгоритмических языках.

В задачах на доказательство объект определен и задан в виде заключения. Решить задачу на доказательство - это найти подтверждение истинности или ложности того, что заключение следует из исходных посылок (данных). Таким образом, в задачах на доказательство решение представляет собой последовательность действий, позволяющих перейти от посылок к заключению, а поиск решения - это процесс нахождения этой последовательности. Исходные данные и результаты решения носят не числовой, а описательный, качественный характер. К задачам такого рода относятся задачи идентификации, прогнозирования, интерпретация и т.д.

Программы для решения таких задач составляются на алгоритмических языках логического (декларативного) типа.

Для решения тех задач, которые требуют анализа сложной структурированной информации, используются автоматизированные и экспертные системы. Системы подобного рода предполагают обязательное участие специалиста, эксперта, как на этапе их разработки, формализации знаний, так и в процессе решения конкретных задач, которые, как правило, носят качественный характер. Такие системы разрабатываются с применением как логических, так и процедурных языков программирования.

Знания - это информация, имеющая внутреннюю структуру, или же организованная по какому либо принципу. Знания представляют собой совокупность сведений, образующих целостное описание, соответствующее некоторому уровню осведомленности о рассматриваемом вопросе, предмете, проблеме.

Знания можно классифицировать различными способами, например как декларативные, процедурные, эвристические, экспертные, прагматические и т. д.

Формализованные знания всегда описывают некоторую предметную область - совокупность реальных или абстрактных объектов (сущностей), связей и отношений между этими объектами, а также процедур преобразования этих объектов при решении задач, возникающих в данной предметной области.

На концептуальном уровне наиболее распространены модели представления знаний в виде систем продукций, семантических сетей, фреймов.

Система продукций представляет собой совокупность логических выражений типа $A \rightarrow B$ *(если A то B, причина - следствие), выполнение которых задается с помощью стратегии управления выводом.*

Сеть - это пятерка H = <A, B, P, P1, C>, где A - множество вершин, B - множество имен вершин, P - множество дуг, соединяющих пары вершин, P1 - множество отмеченных дуг, C - множество имен дуг.

Семантическая сеть – модель, формально описываемая как сеть, в вершинах которой находятся информационные единицы (объекты), а дуги характеризуют отношения и связи между ними.

Широкое распространение получила фреймовая модель М. Минского, в которой под фреймом понимается структура данных для представления некоторого концептуального объекта. Информация, относящаяся к фрейму, содержится в его слотах, все фреймы взаимосвязаны и образуют единую систему, включающую в себя декларативные и процедурные знания.

2.2. ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ И ТЕОРИИ НЕЧЕТКИХ МНОЖЕСТВ

2.2.1 Исчисление предикатов первого порядка

Предикат - специальный знак, отражающий определенное отношение между конечным множеством сущностей (аргументов).

С формальной точки зрения предикатом называется пропозиционная функция P(x1,x2,...xn), определенная на индивидных переменных x1,x2,....xn, область значений которой составляют утверждения истинные или ложные. Синтаксис логики предикатов включает в себя переменные xi, yi, ... ri, индивиды (константы) xio, yio, ... rio, функциональные символы Fi, предикатные символы Ii,Rj,...Ak,Pl, символы логических операторов ~, ∨, ∧, →, ¬, кванторов существования ∃ и всеобщности ∀.

Всякая переменная или индивид (константа) есть терм ti. Выражение F(t1,t2,..,ti,..tn) также терм, если t1,..tn, являются термами. Предикат P(t1,..tn), где - t1,..tn термы, является элементарной формулой.

Выражение является правильно построенной формулой (ППФ), если это элементарная формула или формула вида ¬Pi, Pi ~ Pj , Pi ∨ Pj , Pi ∧ Pj, Pi → Pj, где Pi и Pj - ППФ. Если t - переменная в Pi и Pj, то ∃Pi и ∀P - ППФ, при этом переменная называется связанной квантором, Pi и Pj - области действия квантора.

Интерпретация формулы P логики предикатов состоит из непустой предметной области M и указания значений всех констант, функцинальных и предикатных символов, встречающихся в P.

При этом каждой константе ставится в соответствие некоторый элемент из M, каждому n-местному функциональному символу ставится в соответствие отображение Mn в M, каждому n-местному предикатному символу ставится в соответствие отображение Mn в {1,0}.

Если G и H - предикаты, то значения истинности для ППФ обычно задают табличным способом.

G	H	¬ G	G∧H	G∨H	G→H	G~H
1	1	0	1	1	1	1
1	0	0	0	1	0	0
0	1	1	0	1	1	0
0	0	1	0	0	1	1

Формула (∀t)G получает значение 1, если G получает значение 1 для каждого t из M, в противном случае она получает значение 0.

Формула (∃t)G получает значение 1, если G получает значение 1 хотя бы для одного t из M, в противном случае она получает значение 0.

Синтаксис классической логики предикатов первого порядка (КЛП) описывает двухзначную логику.

2.2.2. Прикладное исчисление нечетких предикатов

Пусть M- множество, x- элемент M, тогда нечеткое подмножество *A* множества M определяется как множество упорядоченных пар {(x,μA(x)}, ∀x∈M, где μA(x) - характеристическая функция принадлежности, принимающая свои значения во вполне упорядоченном множестве E, которая указывает степень или уровень принадлежности элемента x подмножеству A. Множество E называется множеством принадлежности. Если E={0,1, то нечеткое подмножество A рассматривается как обычное подмножество.

Пример представления μ(x) для нечеткого числа 965 приведен на рис. 2.1, при этом на рис. 2.1а функция μ(x) имеет вид нормального распределения, а на рис. 2.1б - кусочно-линейной аппроксимации.

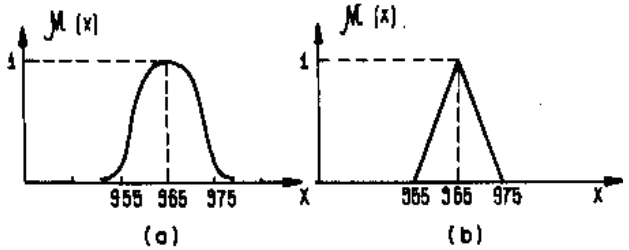


Рисунок 2.1 – Пример представления нечеткого числа

Определение 1. Переменную z будем называть нечеткой, если она определена на множестве M и область ее значений составляет множество нечетких подмножеств {Az}.

Переменная в обычном понимании является частным случаем нечеткой переменной, когда каждое нечеткое подмножество состоит из одного элемента.

Определение 2. Нечетким предикатом называется функция G(z1,z2,...zn), определенная на нечетких переменных z1,z2,...zn, область значений которой составляют утверждения, истинность которых оценивается величинами из отрезка (0,1).

Определение 3. Интерпретация формулы G исчисления нечетких предикатов включает в себя непустое множество предметной области M и указания значений констант, функциональных и предикатных символов, встречающихся в G.

Каждой константе ставится в соответствие некоторое нечеткое подмножество из M с характеристической функцией принадлежности μ(x), при этом $0 \le \mu(x) \le 1$.

Каждому n-местному функциональному символу ставится в соответствие отображение из Mn в множество подмножеств {MF∈M и

множество их характеристических функций принадлежности {μF}, при этом $0 \le \mu Fi \le 1$.

Каждому n-местному предикатному символу ставится в соответствие отображение Mn в множество значений функции истинности {μP}, $0 \le \mu Pi \le 1$.

Формуле G, предикатному символу P ставится в соответствие множество внешних значений функции истинности {μe}, $0 \le \mu ei \le 1$.

Всякая нечеткая переменная или константа есть терм t . Выражение F(t1,t2,...tn) также терм, если t1,t2,...tn- термы. Нечеткий предикат P(t1,t2,...,tn), где t1,t2,...tn, - термы, является элементарной формулой.

Определение 4. Значение функции истинности элементарной формулы равно алгебраическому произведению значений характеристических функций принадлежности термов и внешнего значения истинности нечеткого предиката.

Определение 5. Для двух формул G и F определим нечеткие логические операции следующим образом:

$G \wedge F = \min(G,F)$
$G \vee F = \max(G,F)$
$\neg G = 1 - G$
$G \rightarrow F = \neg G \vee F$
$G \sim F = (\neg G \vee F) \wedge (G \wedge \neg F)$

Теорема 1. Для значений истинности μu ППФ прикладного логического исчисления справедливо $0 \le \mu u \le 1$.

Определение 6. Две формулы G и F эквивалентны (записывается как G=F) тогда и только тогда, когда значения истинности G и F совпадают при любой интерпретации.

Определение 7. Формула в прикладном логическом исчислении находится в предваренной нормальной форме (ПНФ) тогда и только тогда, когда она имеет вид (χ1x1)....(χnxn)(F), где χ={∀,∃}, а F- формула, не содержащая кванторов, (χ1x1)...(χnxn) называется префиксом, F-матрицей формулы.

В связи с тем, что формулы в прикладном логическом исчислении сохраняют основные свойства классического исчисления предикатов, сохраняются и известные схемы их преобразований. Для преобразования формулы в ПНФ необходимо выполнить следующую последовательность операций:

Известно, что матрицу F формулы G можно преобразовать в дизъюнктивную (ДНФ) и конъюнктивную (КНФ) нормальные формы, используя эквивалентные формулы.

В практических приложениях используется, как правило, бескванторная форма представления формул.

2.2.3. Дедуктивные вопросы прикладного исчисления нечетких предикатов

Будем рассматривать декларативные знания как систему аксиом прикладного логического исчисления. Если временно отвлечься от процедуральных и информационных сторон прикладной теории, то задачи представляются утверждениями (теоремами), которые необходимо доказать или опровергнуть.

Определение 8. Формула G прикладного исчисления выполнима (непротиворечива) тогда и только тогда, когда не существует интерпретации I, в которой значение истинности G не равно нулю.

Определение 9. Формула прикладного исчисления невыполнима (противоречива) тогда и только тогда, когда не существует интерпретации I, в которой значение истинности G не равно нулю.

Определение 10. Формула G прикладного исчисления общезначима тогда и только тогда, когда значение истинности не равно нулю при всех возможных интерпретациях.

Определение 11. Формула G есть логическое следствие формул F1, F2,..Fn тогда и только тогда, когда для каждой интерпретации I, если F1 ∧ F2 ... ∧ Fn имеет отличное от нуля значение истинности, то и G имеет значение истинности, отличное от нуля в I.

Задачей доказательства теорем будем считать выяснение вопроса логического следования некоторой формулы G из данного множества формул {F1, F2, ... Fn}, то есть выяснение общезначимости формулы (F1 ∧ F2 ∧... ∧ Fn) → G.

Черчем было доказано, что не существует общего метода установления общезначимости даже для формул исчисления предикатов первого порядка. Однако из теоремы Эрбрана [2,6] следует, что если формула общезначима, то существует процедура проверки ее общезначимости.

Практически удобнее определять невыполнимость, а не общезначимость, поэтому будем рассматривать формулу ¬((F1 ∧ F2 ∧...∧Fn) → G), которая эквивалентна формуле F1∧F2 ∧ ...∧¬Gn, невыполнимость последней и доказывают. Для установления невыполнимости необходимо доказать, что не существует такой интерпретации, при которой одновременно истинны F1,F2,...Fn,¬G. Процедура приведет к успеху в том случае, если G следует из F1, F,...Fn, иначе она может продолжаться бесконечно.

В прикладной теории G представляет собой качественную задачу, а (F1,F2,...Fn)- систему логических выражений, описывающих декларативные знания, а также условия решения задачи. Для установления невыполнимости формулы F1∧F2∧...∧Fn∧¬G в прикладном исчислении нечетких предикатов необходимо доказать, что не существует такой интерпретации, при которой формулы F1,F2...Fn,¬G одновременно имеют значения истинности, отличные от нуля.

Существует ряд методов доказательства теорем в исчислении предикатов первого порядка, наиболее известные из них базируются на принципе резолюций Робинсона [2]. Для применения принципа логические выражения (формулы) приводятся к предваренной нормальной форме (ПНФ), то есть они представляются в виде (χ1x1)...(χnxn)F, где χi∈{∀,∃, а F- формула в конъюнктивной нормальной форме (КНФ). Для преобразования используются известные тождества логики предикатов, затем скулемизацией исключаются кванторы существования ∃, опускаются кванторы всеобщности ∀, знаки конъюнкции ∧ заменяются на запятые, в результате мы имеем множество дизъюнктов S.

В исчислении предикатов первого порядка подстановкой называется конечное множество γ вида γ={t1/x1,.. ti/xi...tn/xn}, где xi- переменная, ti-терм. Обозначим формулу G, для которой все вхождения xi переменных заменены на ti, как Gγ. Множество {Gγ} называется унифицируемым, если существует подстановка, называемая унификатором, такая, что G1γ=G2γ=...=Gnγ. Существует алгоритм унификации, который всегда находит наиболее общий унификатор за конечное число шагов.

В прикладном логическом исчислении xi u ti - термы, причем если xi константа, то и ti константа. В общем случае xi и ti представляют собой множества нечетных подмножеств с функциями принадлежности {μti(ti)} и {μxi(xi). Тогда подстановкой называется конечное множество α пар вида α = {(t1/x1,{μx1(t1)*μt1(t1)/μx1(x1)}),...(tn/xn,{μxn(tn)*μtn(tn)/μxn(xn)})}.

Множество {Gα} называется унифицируемым, если существует подстановка α, при которой значения истинности каждой формулы из {Gα} отличны от нуля. Следствием такого определения является то, что ни одно из произведений μxi(ti)*μti(ti) не должно быть равно нулю.

Если при подстановке α хотя бы одно произведение равно нулю, то определим такую подстановку αv как вырожденную, при этом она представляет собой множество подмножеств αv={{r1/c1,μu(r1)/μu(c1),μx1(r1)/μx1(c1),...μxm(r1)/μxm(c1)}...{rn/cn,μu(rn)/μu(cn), μx1(rn)/μx1(cn),..μxm(rm)/ μxm(cn)}}.

Вырожденная подстановка описывает замену константы cj на константу rj, при этом значение функции истинности дизъюнкта μu(cj) заменяется на μu(rj), а функции принадлежности нечетких переменных xi μxi(cj) на μxi(rj).

Ясно, что при вырожденной подстановке неопределенность знаний возрастает. Это может выразиться либо в уменьшении значения истинности дизъюнкта μu(rj), либо в расширении областей, где для характеристических функций принадлежности нечетких переменных 0< μxi(rj) <1. Характер изменений зависит от существа решаемых задач. Чаще всего уменьшение значения μu(rj) нецелесообразно, более разумным представляется сохранение значений μu(rj) даже за счет большого расширения областей 0<μxi(rj)<1. Источником информации о характере изменений является та часть знаний, которая описывается как закономерности.

Еще один тип подстановок, необходимый для решения задач прикладного характера, назовем подстановкой поглощения. Такие подстановки необходимы при решении задач с арифметическими изменениями количественных величин. Определим ее следующим образом: αp={((t1-x1)/x1,{(μt1(t1)-μx1(t1))/μx1(x1)}),...((tn-xn) /xn,{(μtn(tn)-μxn(tn))/ μxn(xn)})}.

Если произведение 0<μxi(ti)*μti(ti)<1 для всех ti, преобразуем его к виду max(μxi(ti)*μti(ti))(1/(max(μxi(tti)*μti(tti))*μxi(ti), а значение μ=max(μxi(ti)*μti(ti)) будем называть внешним значением функции принадлежности для подстановки α.

Будем называть литерой формулу или ее отрицание, дизъюнктом - дизъюнкцию литер.

Для логики предикатов первого порядка понятие резольвенты вводится следующим образом.

Пусть C1 и C2 - два дизъюнкта, не имеющие общих переменных, L1 и L2 - две литеры в C1 и C2 соответственно. Если L1 и ¬L2 имеют наиболее общий унификатор γ, то дизъюнкт C0Υ=(C1Υ - L1Υ) ∨ (C2Υ - L2Υ) называется резольвентой C1 и C2. Литеры L1 и L2 называются отрезаемыми литерами.

Определим резольвенту в исчислении нечетких предикатов.

Пусть C1 и C2 - два дизъюнкта, не имеющие общих переменных, L1 и L2 - две литеры с функциями истинности μL1 и μL2 в C1 и C2. Если L1 и ¬L2 имеют наиболее общий унификатор α, то дизъюнкт C0α=[μL1*μL2](C1α -L1α)∨(C2α-L2α) будем называть резольвентой C1 И C2 в исчислении нечетких предикатов.

Теорема 3. Пусть даны два дизъюнкта C1 и C2, не имеющие общих переменных, и их функции истинности μ1 и μ2. Тогда для функции истинности μ0 резольвенты C0 дизъюнктов C1 и C2, полученной подстановкой α, справедливо неравенство μ0 ≥μ1α*μ2α.

Пусть согласно определению резольвенты C1α=Lα∨C11α, C2α=¬Lα∨C12α, C0=C11α∨C12α, тогда μ1α = мах(μLα,μ11α), μ2α = мах(1-μLα,μ12α), μ0 = мах(μ11α,μ12α) ≥мах(μLα,μ 11α)*мах(1-μLα,μ12α), т.к. 0≤μ≤1, т.е. μ 0≥ μ1α*μ2α, что и требовалось доказать.

Теорема 4. Пусть даны два дизъюнкта C1 и C2, не имеющие общих переменных, их функции истинности μ1 и μ2, и существует подстановка α, такая, что C1α = Lα∨ C11α, C2α = ¬Lα∨C2α, резольвента C0 = C11α∨C12α. Если хотя бы одна из функций истинности μ11α или μ12α≥0.5, то для функции истинности μ0 резольвенты C0 справедливо неравенство μ0≥ min(μ1α,μ2α).

Перепишем неравенство в следующем вид: max(μ11α,μ11α)≥min (max(μLα, μ11α), max(1- μLα, μ12α)). Пусть μ11α ≥ 0.5, и если μLα ≥ μ11α, тогда max(1-μLα, μ12α)≤0.5, и если μLα<μ11α, тогда max(μLα,μ11α)=μ11α и неравенство также справедливо. Аналогично доказывается случай μ12α≥ 0.5.

В системе знаний, как правило, используется информация со значениями функций истинности, большими 0.5 и близкими к 1, поэтому теорема 4 имеет важное практическое значение для решения качественных задач, т.к. служит обоснованием более высокого значения функций истинности результатов по сравнению с теоремой 3.

Резолюцией называется правило вывода, генерирующее резольвенты из множества дизъюнктов S. Объединение S с множеством всех резольвент, т.е. с множеством выводов, которые могут быть получены из S, обозначим D(S). Тогда D0(S) = S, Du+1(S) = D(Du(S)).

Хотя определение невыполнимости в исчислении нечетких предикатов и в классической логике предикатов первого порядка отличается, теорема Робинсона о полноте формулируется аналогично.

Теорема 5. Если S- произвольное конечное множество дизъюнктов, то S невыполнимо тогда и только тогда, когда Du(S) содержит для некоторого $u \geq 0$ пустой дизъюнкт.

Построенная на базе этой теоремы процедура опровержения может привести к следующим результатам.

1. Du(S) содержит пустой дизъюнкт, т.е. множество S невыполнимо.
2. Du+1(S) совпадает с Du(S), S выполнимо.
3. Процедура продолжается бесконечно, а при компьютерной реализации до исчерпывания ресурсов времени или памяти, и может трактоваться как недостаток информации.

Для того чтобы эффективно получать доказательства, т.е. выводить из противоречивого множества дизъюнктов пустой дизъюнкт, при порождении револьвент могут быть использованы различные стратегии, например семантическая резолюция, лок-резолюция, линейная резолюция.

Фактически метод резолюций- это правило вывода, которое может быть использовано для порождения новых дизъюнктов из уже имеющихся. Неограниченное применение резолюции может порождать много лишних и ненужных дизъюнктов наряду с полезными, и в результате этого потребует больших ресурсов машинного времени и памяти. Поэтому для получения эффективных процедур доказательства теорем мы должны не допустить порождения бесполезных дизъюнктов, используя определенные стратегии резолюции.

2.3. ПРОГРАММИРОВАНИЕ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ ПРОЛОГ

Основная идея языка PROLOG (PROgramming in LOGic). состоит в том, чтобы описать условия задачи, а решение ее получить как результат некоего рутинного процесса. Обычно условия задачи представляет собой множество формул специального вида в языке логики предикатов первого порядка, одна из формул выделена и называется целью, остальные называются посылками. Тогда упомянутый процесс состоит в построении

доказательства цели из посылок в исчислении, единственным правилом вывода которого является правило резолюции.

2.3.1. Основные понятия языка ПРОЛОГ

Существует довольно много различных версий языка ПРОЛОГ, для персональных компьютеров наиболее наглядной и распространенной является версия Turbo Prolog (Турбо-Пролог) компании Borland International, которую мы и будем рассматривать в дальнейшем (6).

Базовое понятие языка – это предикат, или отношение между объектами (доменами). Имена объектов записываются с маленькой буквы, если это константы, и с большой, если это переменные.

Отдельно выделенный предикат называется фактом, например,

likes(john,camera).
likes(tom,computer).
likes(kathy,X).

Первые два предиката likes описывают отношение объектов john и tom к объектам camera и computer, а третий предикат likes описывают отношение объекта kathy к неизвестному объекту.

Любая программа, написанная на Турбо-Прологе, состоит из 5 разделов, представленных в таблице 2.1.

Таблица 2.1 – Программа

```
/*-------------------------------------------------*/
/*                                                 */
/*                  Комментарии                    */
/*-------------------------------------------------*/
 domains
        ‹описание доменов›
 database
        ‹описание предикатов динамической базы данных›
 predicates
        ‹описание предикатов›
 goal
        ‹целевое утверждение›
 clauses
        ‹утверждения›


/*-------------------------------------------------*/
/*                  Комментарии                    */
/*-------------------------------------------------*/
```

Раздел domains содержит содержит определения доменов, которые описывают различные классы объектов. Имеется 6 встроенных типов доменов, представленных в таблице 2.2.

Таблица 2.2 – Типы доменов

Тип данных	Ключевое слово	Диапазон значений	Примеры использования
Символы	Char	Все возможные символы	“a”,”b”,”#”,”B”, “13”,”%”
Целые числа	Integer	От –32768 до 32767	-63, 84, 2349, 32763
Действительные числа	Real	От +1E –307 до +1E308	-42769, 8324, 360, 093, 1.25E23, 5.15E-9
Строки	String	Последовательность символов (не более 250)	“Today”,”123”, “just_a_reminder”
Символические имена	Symbol	1)Последова-тельность букв, цифр и подчер-ков; первый символ - строч-ная буква 2)Последова-тельность символов, заключенная в кавычки	pay _check, school_day, flower, “Stars and Stripes”, ”Singing in the rain”
Файл	File	Допустимое в DOS имя файла	Mail.txt, BIRDS.DBA

Для рассмотренных ранее фактов в разделах **domains** и **predicates** должны появиться такие описания:

```
domains
person, things = symbol
predicates
likes(person,things)
```

Можно не описывать в разделе domains домены person, things, хотя это и придает программе ясный содержательный смысл. Тогда в разделе predicates необходимо сделать следующее описание

```
predicates
likes(symbol, symbol)
```

Раздел **database** содержит утверждения динамической базы данных (может быть опущен).

Раздел **predicates** служит для описания используемых предикатов.

Раздел **goal** содержит формулировку задачи, записанную как внутренняя цель (чаще задачи формулируются как внешние цели).

В разделе **clauses** содержаться данные и знания, описывающие предметную область (факты и правила).

Кроме фактов для описания знаний используются правила, которые с математической точки зрения представляют систему продукций, каждая из которых представлена в виде хорновского дизъюнкта вида

Bn ← A1∧A2∧…∧Am.

Согласно синтаксису Пролога, правила состоят из головы (предикат **Bn**) и тела (предикаты **A1∧A2∧…∧Am**). Тело записывается в виде последовательности предикатов, отделенных друг от друга запятыми, а символ ← заменяется на :-, т.е. правило имеет вид **Bn :- A1,A2,…Am.**

С точки зрения математической логики предикат **Bn** истинен, если при каком-либо сочетании значений доменов истинны все предикаты **A1∧A2∧…∧Am.**

Все знания на языке Пролог состоят из набора фактов и правил. Каждая задача формулируется, как цель (goal), которую необходимо достигнуть. Цель имеет такую же структуру, как факты или правила. Работа Пролог-программы состоит в достижении цели. Если цель представляет собой правило, то она (голова правила) может быть достигнута лишь в том случае, если последовательно достигнуты все подцели (тело правила).

Для достижения цели используется метод резолюций, который реализован в языке с помощью средств, называемых механизмами сопоставления и отката.

Суть метода сопоставления состоит в том, что предикаты сравниваются. Сопоставимы два предиката, у которых одно и тоже имя и одинаковые константы в тех же позициях. Константа сопоставима с переменной, при этом после сопоставления переменная принимает значение константы. Для достижения цели при сопоставлении предиката с правилом производится последовательное сопоставление с каждым предикатом тела правила слева направо.

Например, определенные страны Европы имеют общие между собой границы, в то время как другие их не имеют. Предикатом для предоставления этого отношения служит

border (country, country)

Тот факт, что Германия в Франция имеют общую границу, можно представить в виде утверждения

border (“France”, “Germany”)

Франция с Германией имеют общую границу, так же как и Франция с Испанией, и Франция с Италией.

Шесть утверждений задают все возможные пары четырех выбранных европейских стран:

```
euro_ pair (“France”, “Germany”)
euro_ pair (“France”, “Spain”)
euro_ pair (“France”, “Italy”)
euro_ pair (“Germany”, “Spain”)
euro_ pair (“Germany”, “Italy”)
euro_ pair (“Spain”, “Italy”)
```

Утверждения для стран с общей границей выглядят так:

```
border ( “France”,  “Germany”)
border ( “France”,  “Spain”)
border ( “France”,  “Italy”)
```

Предположим теперь, что вы хотите определить, какие из стран не имеют общей границы. Отрицание предиката border задается при помощи предиката not:

```
not (border (country1, country2))
```

Этот предикат, используемый как цель, выдает все пары не граничащих друг с другом стран.

```
/*Программа “Пары стран Европы” */
/*Указание: Цель внутренняя          */

domains

country = symbol

predicates

euro_pair (country, country)
border(country, country)
find_non_border_pair

goal

find_non_border_pair
```

```
clauses

/* факты*/

euro_ pair (“France”, “Germany”).
euro_ pair (“France”, “Spain”).
euro_ pair (“France”, “Italy”).
euro_ pair (“Germany”, “Spain”).
euro_ pair (“Germany”, “Italy”).
euro_ pair (“Spain”, “Italy”).

border ( “France”,  “Germany”).
border ( “France”,  “Spain”).
border ( “France”,  “Italy”).

/*правила*/

find_non_ border_ pair:-
euro_pair (X,Y)
hot (border X,Y)
write (X,” -“, Y).
```

Механизм отката состоит в том, что если при попытке достижения цели текущее сопоставление с предикатом оказалось неудачным, то происходит возврат на один шаг назад и осуществляется следующая попытка достижения цели с помощью последовательности всех возможных сопоставлений, пока все варианты сопоставления не будут исчерпаны и получен либо положительный (достигнут конец правила), либо отрицательный результат. Для управления процессом отката предусмотрены два встроенных предиката: fail (неудача), и cut, или ! (отсечение).

Очень часто в программах необходимо выполнить одну и ту же задачу несколько раз. В программах на Турбо - Прологе повторяющиеся операции обычно выполняются при помощи правил, которые используют откат и рекурсию одновременно.

Правила Турбо - Пролога, выполняющие повторения, используют откат, а правила, выполняющие рекурсию, используют самовызов.

Вид правила, выполняющего повторение, следующий:

```
repetitive_ rule:-          /*правило повторения*/
  <предикаты и правила>
fail.                       /*неудача*/
```

Конструкция <предикаты и правила> в теле правила обозначает предикаты, содержащие несколько утверждений, а также правила, определенные в программе. Встроенный предикат fail (неудача) вызывает откат, так что предикаты и правила выполняются еще раз.

Вид правила, выполняющего рекурсию, следующий:

```
recursive_rule:-                /*правило рекурсии*/
        <предикаты и правила>
recursive_rule.
```

Последним правилом в теле данного правила является само правило recursive_rule. Правила рекурсии содержат в теле правила сами себя.

Правила повтора и рекурсии могут обеспечить одинаковый результат, хотя алгоритмы их выполнения различны. Каждый из них в конкретной ситуации имеет свои преимущества.

Правило повтора, определяемого пользователем, имеет следующий вид:

```
repeat.             /*повторить*/
repeat :- repeat.
```

Первый repeat является утверждением, объявляющим предикат

repeat истинным. Первый repeat не создает подцелей, поэтому данное правило всегда успешно. Однако, поскольку имеется еще один вариант для этого правила, то указатель отката устанавливается на первый repeat. Второй repeat - это правило которое использует само себя как компоненту (третий repeat). Второй repeat вызывает третий repeat, и этот вызов вычисляется успешно, так как первый repeat удовлетворяет подцели repeat. Следовательно, правило repeat всегда успешно. Предикат repeat будет вычисляться успешно при каждой новой попытке его вызвать после отката. Факт в правиле будет использоваться для выполнения всех подцелей программы. Таким образом, repeat - это рекурсивное правило, которое никогда не бывает не успешным .

Правило repeat широко используется в качестве компоненты других правил. Примером этого может служить программа Эхо, которая считывает строку, введенную с клавиатуры, и дублирует ее на экран.

Если пользователь введет stop, то программа завершается.

```
/*Программа:  Эхо                                                    /*
/*Назначение:  Демонстрация  использования  рекурсии методом,  */
/*                  определенным  пользователем                      /*

domains

        name = symbol

predicates

       write_message
       repeat
       do_echo
       check(name)

goal

     write_message,
     do_echo.

clauses

repeat.
repeat :- repeat.

write_message :-
        nl, write(“ Введите, пожалуйста, имена”) ,nl,
write("Я повторю  их”) , nl,
write(“Чтобы остановить  меня, введите stop”), nl,nl.
do_echo :-
     repeat ,
     readln(Name),
     write(Name),nl,
     check(Name),!.

check(stop) :-
      nl, write(“  -OK, bye!”).


check(_) :-fail.
```

```
/* Программа: Факториал!                                          */
/*Назначение: Демонстрация использования рекурсии для             */
/*            процедура вычисления факториала N! Поло-            */
/*            жительного числа N.  Процедура использу-            */
/*            ет предикат cut для запрещения отката               */
/*Пример:     7!=7*6*5*4*3*2*1=5040                               */


domains

        number, product=integer

predicates
        factorial(number, product)

goal

        factorial(7,Result),

        write(" 7!=  ", Result), nl.
clauses

factorial(1,1) : - !.
factorial(Number, Result) : -
          Next_number=Number-1,
           factorial (Next_number, Partial_factorial),
           Result=Number*Partial_factorial.



/*Программа:  Отгадай число                                      */
/*Назначение:  Демонстрация простого варианта игры               */
/*            "Отгадай число"                                    */
/*Указание:   Запустите программу. Цель содержится в             */
/*            программе                                          */


domains

predicates

        play_the_game
        give_info
        play_it
```

```
      generate_rand(integer)
      play_it_sam(integer)
      test_and_tell(integer,integer)
      say_you_got_it_right(integer)
      say_too_big
      say_too_small

goal

      play_the_game

clauses

/* цель является правилом                              */
       play_the_game:-
       give _info,
                              play_it

 /* выдача  информации пользователю об игре */

give _info : -
              make window(1,7,7,"",0,0,25,80),
              make window(2,7,7,"A  NUMBER Guessing Game",
                          2,20,22,45),
             nl,write(" This is a number guessing game."),
             nl,write("I shall think  of an integer number"),
             nl,write("between i and 100. You make a guess"),
             nl,write("and type in your guess. If your gues"),
             nl,write("is correct, I shall say so. If not,"),
             nl,write("too small."), nl,
             nl,write("When you are ready, press the space bar."),
             nl, readchar (_),
             clearwidow.

/* выполнение игры                                        */
play_it: -
          generate_rand(A),
          play_it_sam(A).

/* генерализация  случайного числа                        */

generate_rand(X):-
              random(R),
              X=1+R*100,
```

```
            nl,write("I have thought of a number."),
             nl,write("Now, it is your turn!"), nl.

/* запрос к пользователю                                    */

play_it_sam(X):-
          nl,write("Type in your guess."),
          nl,readint(G),
          test_and_tell(X,G),
          play_it_sam(X).

/*проверка и выдача результата                              */

test_and_tell(X,G):-
               X=G,
                Say_you_got_it_right(X).

Test_and_tell(X,G):-
              X>G,
              Say_too_big.

Test_and_tell(X,G):-
               X<G,
                Say_too_small.

/* выдача сообщений                                         */

say_too_big:-
          nl,write("Your guess is too big."),
          nl,write("Try a smaller number.").

say_too_small:-
          nl,write("Your guess is too small."),
          nl,write("Try a bigger number.").

say_you_got_it_right(X):-
          nl,write("You got it right."),
          nl,write(" It is ",X,"."),
           nl,write("          Good bye!"),
           nl,write("Press the space bar."),
           nl,readchar(_),
           exit.

/* конец программы                                          */
```

2.3.2. Главное меню системы Турбо-Пролог

Главное меню Турбо-Пролога включает в себя 7 опций (команд) в верхней части экрана. Первая буква названия каждой из команд выделена при помощи увеличенной яркости. Выделение означает, что для задания команды достаточно нажать лишь первую букву ее названия.

Run Compile Edit Options Files Setup Quit

Editor	Load
Line1 Col1 Indent Insert WELCOM	Save
Predicates	File name:
Hello	WELCOME.PRO
Goal	Rename
Hello.	File Name
Clauses	Module list
Hello:-	Zap file in editor
Write("Welcome to Turbo	Erase
Prolog!"), nl.	Operating system
Message	Trace
Compiling WELCOM.PRO	
Hello	
Compilation successful	

Рисунок 2.2 – Главное меню

Команды относятся к 7 функциям, описываемым в меню Турбо-Пролога:
Запуск программы на счет (Run)
Трансляция программы (Compile)
Редактирование текста программы (Edit)
Задание опций компилятора (Options)
Работа с файлами (Files)
Настройка системы в соответствии с индивидуальными потребностями (Setup)
Выход из системы (Quit)

Существует два способа задания команд. Первый требует нажатия клавиши, соответствующей первой букве названия выбранной команды. Так, для выбора команды Edit необходимо нажать E. Для окончания работы с командой используется клавиша Esc. Второй способ состоит в перемещении по меню при помощи стрелок; переход к работе с выбранной командой осуществляется нажатием Enter.

Главное меню содержит четыре окна. В левом верхнем углу располагается окно редактора Турбо-Пролога (Editor), в правом верхнем углу – окно диалога (Dialog), в левом нижнем – окно сообщений (Message), в правом нижнем – окно трассировки (Trace).

Верхняя строка окна редактора содержит информацию о высвечиваемом в этом окне файле (смотри рис.3.1). Line1 и Col1 свидетельствуют о том, что курсор в настоящий момент располагается в первой позиции первой строки. Значения этих индикаторов строки и позиций меняется вслед за изменением положения курсора. Надпись Indent сигнализирует о том, что включен режим автоматического выравнивания строк, а надпись Insert – о том, что задан режим вставки. WORK.PRO является заданным по умолчанию именем рабочего файла; .PRO есть заданное по умолчанию расширение для файлов, содержащих программы на Турбо-Прологе. Если вы наберете в редакторе какой-либо текст и запишите его на диск без изменения имени файла, то файл с вашим текстом получит имя WORK.PRO.

2.3.3. Основные режимы работы системы

Перейдите при помощи стрелки к команде главного меню Edit и нажмите клавишу Enter (либо просто введите латинскую букву E). При этом в левом верхнем углу окна Editor появится мерцающая черточка – курсор редактора. Теперь редактор готов принять вводимый вами с клавиатуры текст. Набейте текст программы WELCOME.PRO:

```
predicates
        hello
  goal
        hello.
  clauses
        hello :-
           write ("Welcome to Turbo Prolog"),
            nl.
```

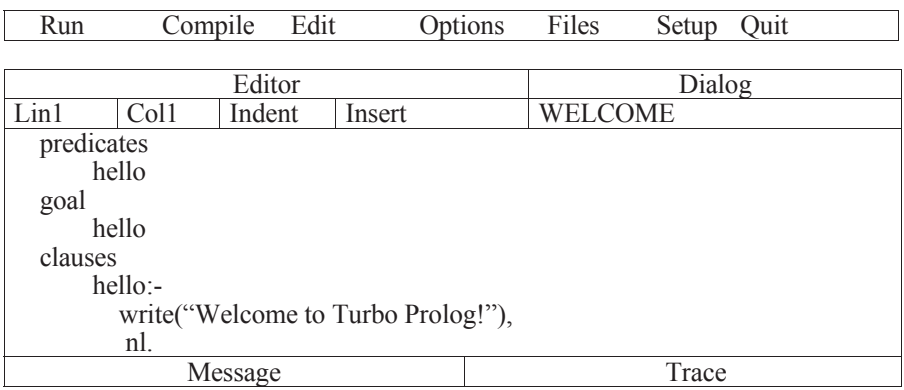


Рисунок 2.3 – Программа WELCOME.PRO в окне редактора

После ввода программы экран компьютера должен выглядеть так, как это показано на рис.2.3. Для того чтобы запустить ее, сначала требуется нажать клавишу Esc. Курсор редактора при этом исчезнет, а курсор главного меню укажет на команду Edit. Далее следует задать команду Run. Результат работы программы - в окне диалога Dialog (рис.2.4).

Первая строка в окне сообщений указывает на то, что началась трансляция программы WELCOME.PRO. Трансляция задается автоматически при задании команды Run, т.е. нет необходимости прибегать к помощи специальной команды Compile.

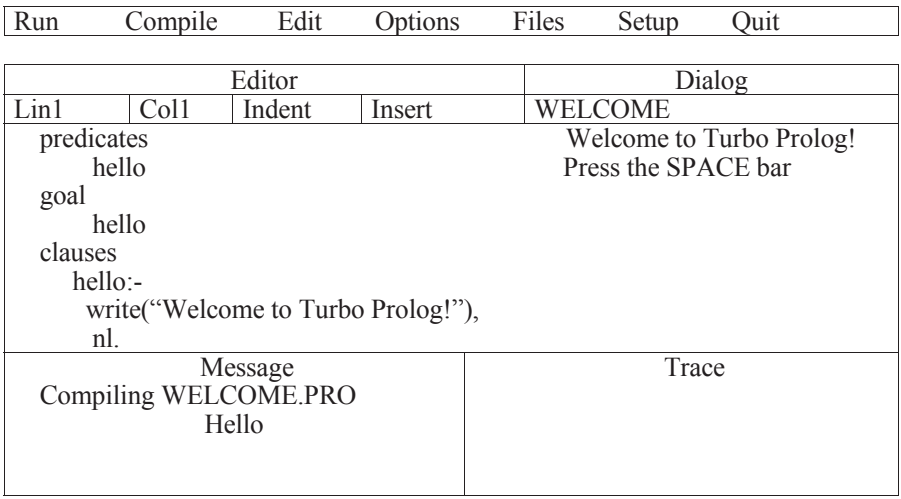


Рисунок 2.4 – Результат трансляции и запуска программы WELCOME.PRO.

Если в рабочий файл были внесены какие либо изменения, система спросит, нужно ли записывать новый исправленный вариант файла. Для обозначения положительного ответа необходимо нажать клавишу Y.

Для того чтобы сохранить программу, необходимо выйти из редактора, нажав клавишу Esc, а затем выбрать команду Files и подкоманду Save во вновь появившемся меню. В результате этих действий на экране возникает окно, в котором будет высвечено либо заданное по умолчанию имя файла (как, например, WORK.PRO), либо то имя, которое вы присвоили файлу. Имя файла можно оставить без изменений, а можно отредактировать. В нашем случае следует ввести имя WELCOME.PRO, а затем нажать ENTER.

Для того чтобы просмотреть каталог файлов какой либо директории, необходимо выбрать в главном меню команду Files и команду Directory в появившемся меню команды. На экране возникает окно, в котором перечисляются все файлы текущей директории PRO. Если вы хотите увидеть каталог другой директории, то следует ввести путь доступа к этой

директории, а затем нажать клавишу Enter. В ответ система попросит задать маску интересующих вас файлов (File mask). По умолчанию стоит маска *.PRO.

После того, как вы нажмете Enter, в окне появятся имена всех файлов директории, удовлетворяющие заданной маске.

Для того чтобы загрузить в окно редактора уже существующий файл, требуется выбрать команду Files главного меню и подкоманду Load в меню Files. Если в ответ на запрос имени файла нажать клавишу Enter, то на экране в специальном окне будет высвечен перечень файлов директории PRO. Теперь, используя четыре стрелки, можно указать имя интересующего вас файла, и после этого нажать клавишу Enter. Если же вы решили набрать имя файла с клавиатуры, нет необходимости указывать его расширение, так как по умолчанию считается, что файл имеет расширение .PRO.

Работая с редактором Турбо-Пролога, можно получить информацию о любой из команд, для этого требуется нажать функциональную клавишу F1, на экране появляется меню подсказки Help. Если вы выберете первую опцию из предлагаемого списка, то на экране возникнет окно Help. Окно демонстрирует краткий перечень команд редактора и другую полезную информацию о редакторе.

Предикатная конструкция, называемая целью, используется для запуска процесса выполнения программы. Турбо – Пролог пытается сопоставить цель с фактами и правилами программы. Если цель является фактом, таким, как likes (mary,apples), то Турбо – Пролог отвечает TRUE (истина) или FALSE (ложь); если цель содержит переменные, то Турбо – Пролог выдает либо те их значения, которые приводят к решению, если оно существует, либо сообщение NO SOLUTIONS (решений нет).

Турбо – Пролог использует как внутренние цели, которые содержатся в программе, так и внешние цели, которые вводятся с клавиатуры после запуска программы. Таблица, расположенная ниже, иллюстрирует ситуацию, которую можно рассматривать как «Приглашение для ввода внешней цели».

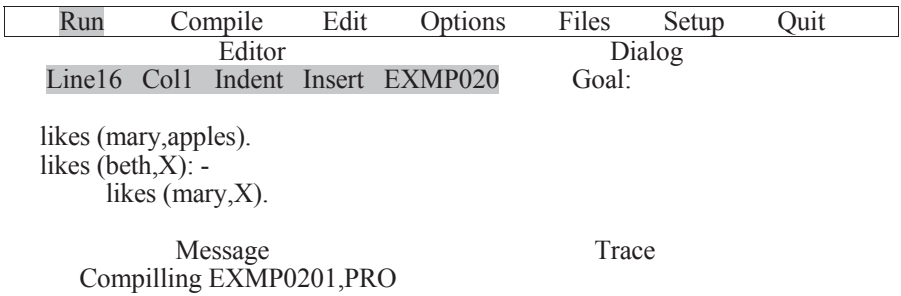


Рисунок 2.5 – Приглашение для ввода внешней цели.

Турбо-Пролог поддерживает связанные объекты, называемые списками. Список – это упорядоченный набор объектов, следующих друг за другом. Составляющие списка связаны между собой, поэтому с ними можно работать и как с группой (списком в целом), и как с индивидуальными объектами (элементами списка).

Турбо-Пролог позволяет выполнять со списком целый ряд операций. Их перечень включает:

- доступ к объектам списка,
- проверку на принадлежность к списку,
- разделение списка на два,
- слияние двух списков,
- сортировку элементов списка в порядке возрастания или убывания.

2.3.4. Работа со списками

Списки используются при создании баз знаний (баз данных), экспертных систем, словарей и т.д. Список является набором объектов одного и того же доменного типа. Объектами списка могут быть целые числа, действительные числа, символы, символьные строки и структуры. Порядок расположения элементов является отличительной чертой списка; те же самые элементы, упорядоченные иным способом, представляют уже совсем другой список. Порядок играет важную роль в процессе сопоставления.

Совокупность элементов списка заключается в квадратные скобки ([]), а друг от друга элементы отделяются запятыми.

Примеры списков:

[1, 2, 3, 6, 9, 3, 4,]

[3, 2, 4, 6, 1.1, 2, 64, 100, 2]

[«YESTERDAY», «TODAY», «TOMORROW»]

Элементами первого списка являются целые числа. Элементами второго – действительные числа, третьего – символьные строки, т. е. конкретные значения символов.

Объекты списка называются элементами списка. Список может содержать произвольное число элементов, единственным ограничением является лишь объем оперативной памяти.

Турбо-Пролог требует, чтобы все элементы списка принадлежали к одному и тому же типу. Количество элементов в списке называется его длиной. Длина списка {“MADONNA”, “AND”, “CHILD”} равна 3. Длина списка [4.50, 3.50, 6.25, 2.9,100.15] равна 5. Список может содержать всего один элемент и даже не содержать элементов. Список, не содержащий элементов, называется пустым или нулевым списком.

Непустой список можно рассматривать как состоящий из двух частей: голова и хвост. Голова является элементом списка, хвост есть список сам по себе. Если список состоит из одного элемента, то его можно разделить

на голову, которой будет этот самый единственный элемент, и хвост, являющийся пустым списком. В списке

{4.50,3.50,6.25,2.9,100.15}

головой является значение 4.50, а хвостом – список

{3.50,6.25,2.9,100.15}

Этот список в свою очередь имеет и голову, и хвост. Голова – это значение 3.50, хвост – список

{6.25,2.9,100.15}

Таблица 2.3 – Головы и хвосты различных списков

Список	Голова	Хвост
{1,2,3,4,5}	1	{2,3,4,5}
{6.9,4.3,8.4,1.2}	6.9	{4.3,8.4,1.2}
{cat,dog,horse}	Cat	{dog,horse}
{’S’,’K’,’Y’}	‘S’	{‘K’,’Y’}
{”PIG”}	“PIG”	{}
{}	Не определена	Не определен

Для того, чтобы использовать в программе список, необходимо записать предикат списка. Ниже приведены примеры таких предикатов:

num {[1,2,3,6,9,3,4]}

realnum {[3.2,4.6,1.1,2.64,100.2]}

time {[“yesterdey”, “today”, “tomorrow”]}

В этих выражениях num, realnum и time представляют предикаты списков. Предикатам списков обычно присваиваются имена, которые характеризуют либо тип элемента (num), либо сами данные (time).

Введение списков в программу отражается на трех ее разделах. Домен списка должен быть описан в разделе domains, а работающий со списком предикат – в разделе predicates. Наконец, нужно ввести сам список; т.е. в программе: либо в разделе clauses, либо в разделе goal.

Каждый элемент приведенного ниже списка обозначает одну из птиц.

```
birds( [“sparrow”, “robin”, “mockingbird”,
        “thunderbird”, “bald eagle”] ).
```

Если этот список необходимо использовать в программе, то следует описать домен элементов списка. Отличительной особенностью описания списка является наличие звездочки (*) после имени домена элементов. Так, запись

bird_name *

указывает на то, что это домен списка, элементами которого являются bird_name*, т.е. запись bird_name* следует понимать как список, состоящий из элементов домена bird_name.

Описание в разделе domains, следовательно, может выглядеть либо как

bird_list = bird-name*

```
bird_name = symbol
```

либо как

```
bird_list    = symbol*
```

В разделе predicates требуется присутствие имени предиката, а за ним – заключенного в круглые скобки имени домена.

```
birds(bird_list)
birds( [“sparrow”, “robin”, “mockingbird”,
         “thunderbird”, “bald eagle”] ).

/*Программа:  Списки                                    */
/*Назначение:  работа со списками                        */

domains

      bird_list = bird_name*
                  bird_name = symbol

      number_list = number*
                    number = integer
predicates

         birds(bird_list)
         score(number_list)

clauses

         birds( [“sparrow”,
                 “robin”,
                 “mosckingbird”,
                 “thunderbird”,
                 “bald eagle”] ).

Score ([56,87,63,89,91,62,85]).

/*****              конец программы              *****/
```

Эта программа создавалась в расчете на следующие внешние запросы:

```
birds(A11).
birds([-,-,-,B,_]).
birds([B1,B2,_,_,_]).
```

score(A11).
score([F,S,T,_,_,_,_]).

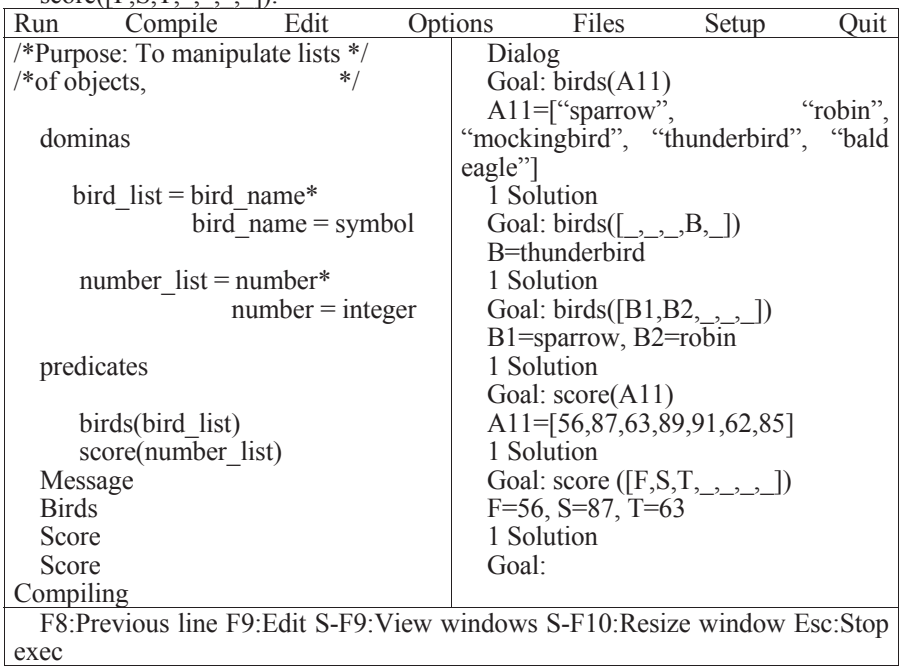


Рисунок 2.6 – Выдача программы «Списки» при задании различных внешних целей

Заметим, что свободная переменная A11 представляет весь список в целом. Он рассматривается при этом как некое целое, элементы играют роль частей этого целого.

Что касается второй цели, birds ([_,_,_,B,_]), то процесс сопоставления начинается с первого элемента. Первые три переменные в целевом утверждении являются анонимными. При сопоставлении это обстоятельство, однако, роли не играет. Переменной B присваивается значение thunderbird. В этом процессе используется внутренняя связь элементов. В результате удовлетворения цели появляется строка B=thunderbird.

Третья цель, birds ([B1, B2,_,_,_]), запрашивает первые два элемента списка. На выходе можно будет увидеть B1=sparrow, B2=robin, т.е. значения первого и второго элементов списка.

Обратимся теперь ко второй части программы, имеющей дело со списком целых чисел. В случае со score (All) переменной All присваивается весь список из 7 элементов, а выдача будет выглядеть так:

All=[56,87,63,89,91,62,85].

Пятая цель – это score ([S1,_,_,S4,_,S6,_]). На выходе будем иметь S1=56, S4=89, S6=62. Так же как и в случае с третьей целью, внутренне связанные элементы будут выбираться селективно в соответствии с порядком в списке.

```
/*Программа:     Элементы                                          */
/* Назначение: Поиск нужного элемента в списке.                    */
domains
number_list =number *
                  number=integer

member_list=member *
                 member=symbol

predicates

find_it(number,number_list)
find_it(member,member_list)

clauses

find_it(Head,[Head/_]).
find_it(Head,[_/Tail]):-
             Find_it(Head,Tail).

/*****          конец программы                    *****/


/*программа: Деление списка                          */
/*назначение: Разделение списка на два.              */
domains
middle = integer
list = integer *
          predicates
split(middle, list, list, list)
clauses
split(middle, [Head | Tail ], [Head | L1], L2):-
head <= Middle,
split(middle, Tail, L1, L2).
split(middle, [Head | Tail], L1, [Head | L2]):-
split(middle, Tail, L1, L2),
head > Middle
split( _ , [],[],[]).
          /*****          конец программы                *****/
```

ВОПРОСЫ К ГЛАВЕ 2

1. Определение задачи.
2. Тип задачи на нахождение.
3. Тип задачи на доказательство.
4. Определение знания.
5. Классификация знания.
6. Определение предиката.
7. Определение нечеткой переменной.
8. Определение нечеткого предиката.
9. Основная идея языка Пролог.
10. Разделы языка Пролог.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Серов В.В.* Логическое представление нечетких знаний и его применение для решения прикладных задач качественного характера [Текст] / В.В. Серов. – М.: РосЗИТЛП, 2001. – 108 с.

2. *Robinson G.* A machine–oriented logic based on the resolution principle [Text] / G. Robinson. – J. ACM, 1965. – V. 12. – № 1. – P. 23–41; *Робинсон Дж.* Машинно–ориентированный логический базис в принципе резолюций [Текст]: пер. с англ.: кибернетический сб. / Дж. Робинсон // Новая серия. – № 7. – М.: Мир, 1970.

3. *Polya G.* Mathematics and plausidle reasoning [Text]: 2 vols. / G. Polya. – Princeton, 1954; *Полйя Дж.* Математика и правдоподобные рассуждения [Текст]: пер. с англ. / Дж. Полйя. – М.: Наука, 1975.

4. *Клоксин У.* Программирование на языке Пролог [Текст] / У. Клоксин, К. Меллиш. – М.: Мир, 1987.

5. *Ин Ц.* Использование Турбо – Пролога [Текст] / Ц. Ин, Д. Соломон; пер. с англ. Д.Ю. Буланже, О.Л. Кондратьева. – М.: Мир, 1993. – 598 с.

6. *Чень Ч.* Математическая логика и автоматическое доказательство теорем [Текст] / Ч. Чень, Р. Ли; пер. с англ. под ред. С.Ю. Маслова. – М.: Наука, 1983.

3. Реализация обучения программированию на базе программно-аппаратных средств в соответствии с требованиями новых образовательных стандартов (ФГОС 3++)

При обучении студентов по направлениям подготовки «Информатика и вычислительная техника» (ИВТ), «Информационные системы и технологии» (ИСТ), «Программная инженерия» (ПИН) и ряда других, в соответствии с ФГОС 3++ [1], требуется освоение ряда компетенций, а именно (рассмотрим на примере ИВТ):

- ОПК-5. Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем;
- ОПК-7. Способен участвовать в настройке и наладке программно-аппаратных комплексов;
- ОПК-8. Способен разрабатывать алгоритмы и программы, пригодные для практического применения;
- ОПК-9. Способен осваивать методики использования программных средств для решения практических задач.

Кроме того, необходимо упомянуть профессиональные компетенции, соответствующие профессиональным стандартам, таким, как:

- «Администратор баз данных» (код 06.011);
- «Специалист по информационным системам» (код 06.015);
- «Технический писатель (специалист по технической документации в области информационных технологий)» (код 06.019)

и другие [1].

Указанное непосредственно относится к таким областям профессиональное деятельности, в которых выпускники могут осуществлять свою профессиональную деятельность, а именно:

- 06 - связь, информационные и коммуникационные технологии (в сфере проектирования, разработки, внедрения и эксплуатации средств вычислительной техники и информационных систем, управления их жизненным циклом), а также к некоторым сквозным видам профессиональной деятельности, в том числе:
- 40.011 - специалист по научно-исследовательским и опытно-конструкторским разработкам;
- 40.030 - регулировщик радиоэлектронной аппаратуры и приборов;
- 40.067 - слесарь-наладчик контрольно-измерительных приборов и автоматики;
- 40.158 - специалист в области контрольно-измерительных приборов и автоматики;

Кроме того, необходимо упомянуть следующие виды сквозной профессиональной деятельности, упомянутые в образовательном стандарте:

- 40.178 - специалист в области проектирования автоматизированных систем управления технологическими процессами;

- 40.083 - специалист по автоматизированному проектированию технологических процессов.

Более подробно с руководящими документами можно ознакомиться в [1].

Ярким примером отличной организации учебного процесса для освоения учащимися указанных выше компетенций является созданный в РГСУ "Технопарк равных возможностей" [2].

Освоение указанных компетенций, с учетом требований стандарта к привитию навыков проектной деятельности обучающихся (в стандарте [1] проектная деятельность упомянута несколько раз, а именно «Разработка и реализация проектов (УК-2); подготовка к «проектному типу профессиональной деятельности», и т.д.), достаточно удобно осуществлять с помощью практического программирования, т.е. выполнения реальных проектов на основе распространенной программно-аппаратной базы [4-6].

Настоящая работа посвящена краткому описанию практических занятий по дисциплинам, связанным с практическим программированием и электроникой.

3.1. ПРОГРАММИРОВАНИЕ. ЗНАКОМСТВО С МИКРОКОНТРОЛЛЕРНОЙ БАЗОЙ НА ОСНОВЕ ПЛАТФОРМЫ «ARDUINO»

В настоящем разделе представлено программирование (начальные навыки) с использованием широко распространенной микроэлектронной базы, а именно «Arduino».

Микроконтроллер – вычислительное устройство, предназначенное для управления другими устройствами. Это сборка на основе одной или нескольких микросхем. Собственно в микроконтроллере имеются процессор, память (ПЗУ и ОЗУ), периферийные устройства, что обеспечивает их работу и взаимодействие с внешним миром с помощью специальной микропрограммы, которая храниться внутри микроконтроллера. Таким образом, микроконтроллерные устройства применяются почти во всей бытовой технике, автомобилях, телевизорах, промышленных роботах, а также в военных радиолокаторах.

Arduino Uno – платформа (на базе микроконтроллера ATmega328P) для разработки разнообразных автоматических и автоматизированных устройств. На Arduino Uno предусмотрено всё необходимое для удобной работы: 14 цифровых входов/выходов (6 из них могут использоваться в качестве ШИМ-выходов), 6 аналоговых входов, кварцевый резонатор на 16 МГц, разъём USB, разъём питания, разъём для внутрисхемного программирования (ICSP) и кнопка сброса, см. рисунок 3.1.

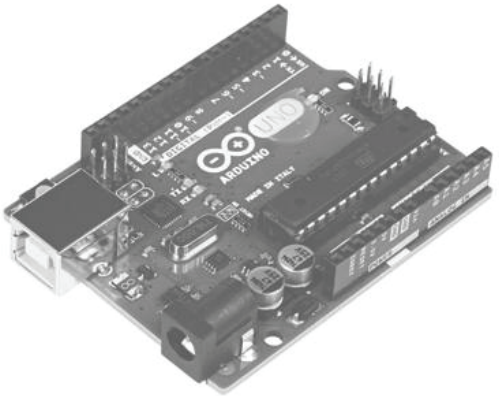


Рисунок 3.1 – Плата «Arduino Uno»

Проверка работоспособности подсоединения Arduino к компьютеру осуществляется следующим образом:

- соединить компьютер с Arduino с помощью кабеля USB;
- запустить Arduino IDE файлом arduino.exe (для ОС Windows);
- открыть нижеприведенный код (см. рисунок 3.2);
- дождаться в нижней части окна сообщения о завершении загрузки, после чего на плате Arduino начнет мигать светодиод около пина номер 13.

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH); // светодиод включен
  delay(1000);            // интервал одна секунда
  digitalWrite(13, LOW);  // светодиод выключен
  delay(1000);            // интервал одна секунда
}
```

Рисунок 3.2 – Проверочный код для платы «Arduino Uno»

3.1.1. Датчики информации, используемые при построении схем

Для построения схем в подавляющем большинстве случаев необходимы датчики информации. Использование датчиков информации можно отнести к мехатронике. Мехатроника находится на стыке нескольких наук, то есть объединяются механическое и электронное проектирование систем

автоматизации и управления разнообразными устройствами. При этом используется программирование, включая визуальное, онлайн и т.д., наряду с классическим. Одним из наиболее важных и полезных разделов, который могут охватить учащиеся в курсе введения в мехатронику – понимание как принципов работы, так и практического применения датчиков. Во всех видах промышленности используются разнообразные датчики. В частности, в автомобильной промышленности при измерениях положения распределителя, педалей и клапанов используются преобразователи магнитного поля. Оптические датчики используются на сборочных конвейерах в машинном производстве для бесконтактного считывания положения. В транспортных контейнерах устанавливаются пленочные пьезодатчики для регистрации вибрации товара.

При организации практических работ были выбраны наиболее широко используемые на практике датчики, используемые для регистрации уровня освещенности, температуры, давления, перемещения, нажимные кнопки, потенциометры, датчики загазованности, присутствия, ультрафиолетового света и другие [3].

Перечень датчиков представлен ниже:

- поворотный потенциометр для измерения положения;
- линейный потенциометр;
- датчики давления;
- датчики нажатия;
- термисторы;
- датчик загазованности;
- датчики для измерения больших расстояний: звуковые и инфракрасные;
- микровыключатель;
- нажимная кнопка;
- энкодер;
- датчики ускорения (акселерометры);
- гироскопы;
- тензодатчик для измерения деформации;
- пленочный пьезодатчик для измерения вибраций;
- оптический сканер отпечатков пальцев;
- сенсорная кнопка (модуль);
- некоторые другие датчики.

Необходимо отметить, что многие из датчиков информации не могут использоваться напрямую вследствие достаточно сложного закона преобразования информации с собственно датчика, а также в силу ряда иных причин. Поэтому такие датчики выпускаются совместно со специализированными микросхемами и устройствами, конструктивно с ними объединенными. Поэтому такие датчики называют «модулями».

После обработки информации микроконтроллерным устройством (в частности, Arduino), ее необходимо выводить в приемлемом для потребителя виде (например, буквенно-цифровом, на линейные шкалы, в

виде светоизлучающих индикаторов различного цвета). Для этой цели используются устройства вывода информации, к которым в данном учебном курсе можно отнести:

- светодиоды монохромные;
- светодиоды RGB-типа;
- светодиодные линейки;
- светодиодные модули;
- светодиодные ленты;
- светодиодные матрицы;
- жидкокристаллические дисплеи текстовые;
- жидкокристаллические дисплеи графические;
- TFT-экраны (мониторы) различных размеров;
- цветные сенсорные дисплеи различных типов;
- OLED-дисплеи;
- излучатели звуковые пьезоэлектрического типа;
- излучатели звуковые динамического типа;
- семисегментный индикатор;
- модуль семисегментных индикаторов (четыре совмещенных семисегментных индикатора);
- а также ряд других.

3.1.2. Пример построения схем с использованием датчиков информации

В качестве примеров приведем реализацию устройства, регистрирующего такие распространенные параметры, как освещенность и температура.

Устройство регистрации освещенности и температуры

В датчике освещенности используется фоторезистор. Фоторезистор (или LDR) типа VT90N2 - компонент, меняющий сопротивление в зависимости от количества падающего на него света. В полной темноте он имеет максимальное сопротивление в сотни килом или единицы мегаом, а по мере роста освещённости сопротивление уменьшается до десятков и единиц килоом. На его основе достаточно просто создать схему, которая бы выводила данные об уровне освещённости в виде аналогового сигнала. Для этого достаточно сделать делитель напряжения, где одним из резисторов будет фоторезистор, а вторым - резистор на 100 кОм.

Характеристики фоторезистора:

- теневое сопротивление: 2 МОм;
- сопротивление при 200 люкс: 500 Ом.

Датчик температуры. Здесь используется сенсор тип TMP36. Особенности и преимущества:

- откалиброваны в градусах по шкале Кельвина;
- низкое рабочее напряжение питания (от +2.7 В до +5.5 В);
- погрешность в диапазоне температур ±2°C;

- работают при токах от 0,4 до 5 мА;
- характеристики гарантируются для диапазона от -40 °С до +125 °С;
- рабочий потребляемый ток менее 50 мкА.

Имеется возможность измерить температуру также с помощью сенсора тип LM335. Особенности и преимущества:

- откалиброваны в градусах по шкале Кельвина;
- первоначальная точность 1 °С;
- работают при токах от 0,4 до 5 мА;
- характеристики гарантируются для диапазона от -40 °С до +100 °С;
- кратковременные перегрузки до 200 °С.

На датчике температуры TMP36 (или LM335) имеется 3 пина, связанных с питанием, землей и с аналоговым пином, на котором появляется напряжение в соответствии с температурой измерении.

Структурная схема объекта проектирования приведена на рисунке 3.3. Данные схемы спроектированы магистрантом МАИ Чан Шон Хунг, под руководством автора, в ходе курсового проектирования обучающегося.

Рисунок 3.3 – Структурная схема объекта проектирования

Назначение блоков структурной схемы следующее.

Микроконтроллер – это микроконтроллерная плата (например, Arduino), предназначенная для управления всей работы проектируемой системы. В зависимости от того, как он будет запрограммирован на считывание сигналов с датчиков и подачу сигналов на управляющие устройства, так и будет функционировать вся схема.

Блок датчиков представляет собой датчик освещенности и датчик температуры TMP36.

Блок светодиодов – элементы стандартного оборудования освещения.

3.1.3. Моделирование в среде Tinkercad.com

Схемы моделирования представлены на рисунках 3.4 – 3.7.

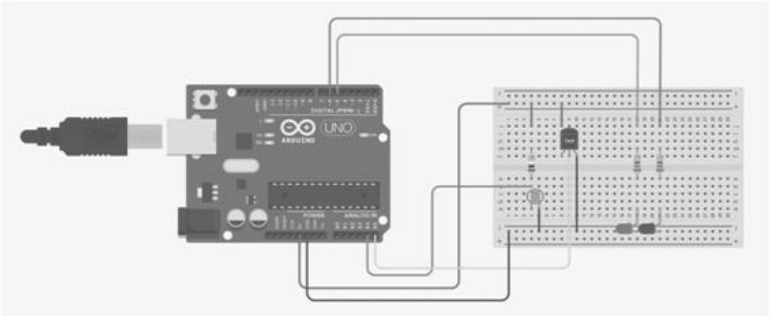


Рисунок 3.4 – Схема моделировании в среде Tinkercad.com

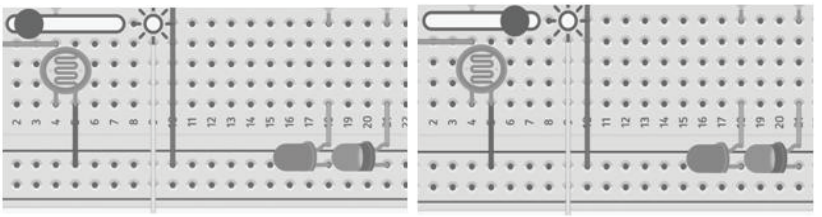


Рисунок 3.5 – Схема моделирования. Горит правый светодиод в случае, когда слишком яркое или слишком слабое освещение

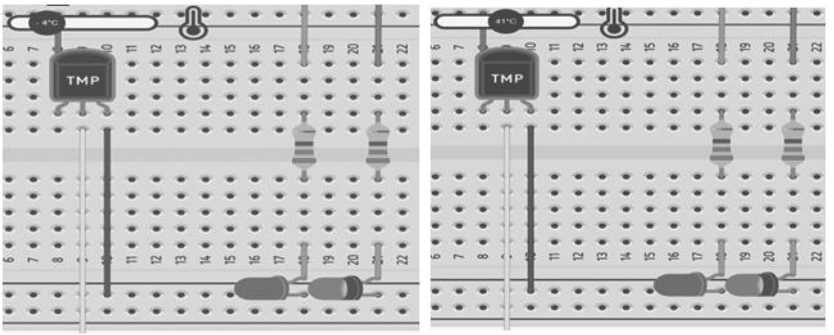


Рисунок 3.6 – Горит правый, когда слишком высокая или слишком низкая температура

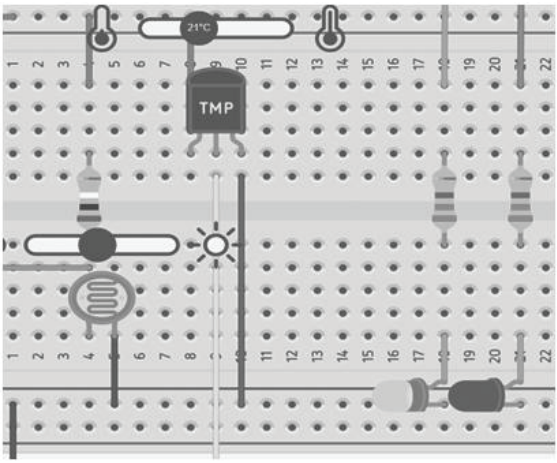


Рисунок 3.7 – Горит правый светодиод - когда нормальные условия (не слишком яркое и не слишком слабое освещение, не слишком высокая и не слишком низкая температура)

На рисунках 3.8, 3.9 представлено реальное устройство (т.е. собранное на практике).

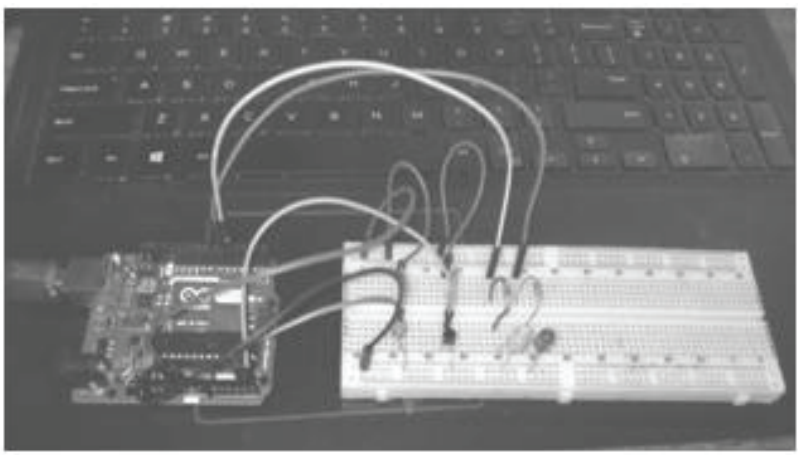


Рисунок 3.8 – Реальное моделирование – подключенные друг к другу компьютер, плата Arduino и датчики информации

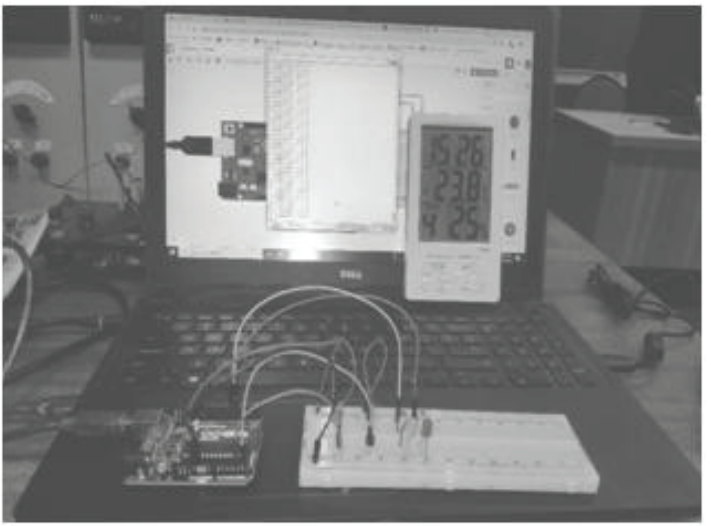


Рисунок 3.9 – Компьютер с подключенным устройством

С помощью термометра можно проверять точность обработки сигналов на практике, рисунок 3.10.

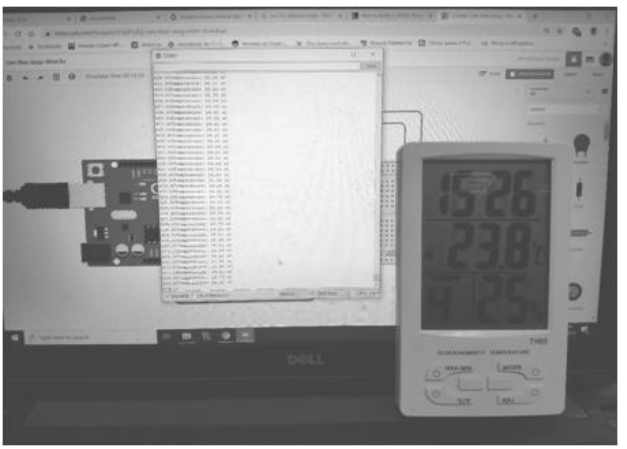


Рисунок 3.10 – Сравнение температуры по калиброванному термометру и по разработанному устройству (программный вывод в терминал)

Программный код в среде моделирования TinkerCAD.com представлен на рисунке 3.11.

```
int termPin = A5; // Аналоговый пин с термистором
int ledPin = 6;   // Цифровой пин с красным светодиодом
int lightPin = A4; // Аналоговый пин с фоторезистором
int ledPin2 = 5;   // Цифровой пин с зеленым светодиодом

float sensor = 0; // Инициализируем начальные значения
float celsius = 0;

void setup()
{
   pinMode(ledPin, OUTPUT);
   pinMode(ledPin2, OUTPUT);
   Serial.begin(9600);
}
void loop()
{
   sensor = analogRead(termPin);
   celsius = sensor*0.4881657-49.7633;
        Serial.print("Temperature: ");
        Serial.print(celsius,2);
        Serial.println(" oC");
   // Считываем показания термистора и сравниваем
 if (celsius>5&&celsius<33&&analogRead(lightPin)>6&&analogRead(lightPin)<20) {
   //если ненормальные условия – горит «красный»
   //когда слишком яркое и слишком слабое освещение;
   //слишком большая или слишком низкая температура.
      digitalWrite(ledPin2, HIGH);
      digitalWrite(ledPin, LOW);
   } else {
    //когда нормальные условия - горит «зеленый»
      digitalWrite(ledPin, HIGH);
      digitalWrite(ledPin2, LOW);
   }
   delay (100);
}
```

Рисунок 3.11 – Программный код в среде моделирования TinkerCAD

Программный код реальной системы несколько отличается от схемы моделирования в TinkerCAD вследствие замены нескольких электронных компонентов, принципиально не влияющих на работу устройства. Программный код реальной системы представлен на рисунке 3.12.

```
int termPin = A5; // Аналоговый пин с термистором
int ledPin = 6;   // Цифровой пин с красным светодиодом
int lightPin = A4; // Аналоговый пин с фоторезистором
int ledPin2 = 5;   // Цифровой пин с зеленым светодиодом

float sensor = 0; // Инициализируем начальные значения
float celsius = 0;

void setup()
{
   pinMode(ledPin, OUTPUT);
   pinMode(ledPin2, OUTPUT);
   Serial.begin(9600);
}

void loop()
{
   sensor = analogRead(termPin);
   celsius = (sensor/1024)*5000/10-273.15;
  Serial.print("Temperature: ");
  Serial.print(celsius,2);
  Serial.println(" oC");
    // Считываем показания термистора и сравниваем
     if
(celsius>5&&celsius<33&&analogRead(lightPin)>100&&analogRead(lightPin)<300) {
   //если ненормальные условия – горит «красный»
   //когда слишком яркое и слишком слабое освещение;
   //слишком большая или слишком низкая температура.
      digitalWrite(ledPin2, HIGH);
      digitalWrite(ledPin, LOW);
   } else {
    //когда нормальные условия - горит «зеленый»
      digitalWrite(ledPin, HIGH);
      digitalWrite(ledPin2, LOW);
   }
   delay (100);
}
```

Рисунок 3.12 – Программный код реальной системы

ВОПРОСЫ К ГЛАВЕ 3

1. Приведите основные характеристики микроконтроллерной платформы Arduino Uno. Какими возможностями обладает данная платформа?
2. Что такое ШИМ-выходы микроконтроллерной платформы Arduino Uno? Какие возможности предоставляет ШИМ?
3. Какие имеются модификации микроконтроллерной платформы Arduino? Приведите характеристики некоторых из них.
4. Охарактеризуйте датчики информации, которые можно использовать для ввода информации в микроконтроллерную платформу Arduino.
5. Какие функции выполняют библиотеки при работе с датчиками информации? Приведите примеры.
6. Охарактеризуйте устройства вывода информации, применяемые совместно с микроконтроллерной платформой Arduino Uno.
7. Объясните принцип функционирования датчика освещенности на основе фоторезистора.
8. Как можно изменить чувствительность датчика освещенности? Назовите пути возможных изменений как в аппаратной, так и в программной частях устройства (датчик включен по схеме делителя напряжения).
9. Измените аппаратную и программную части устройства для датчика освещенности так, чтобы логика срабатывания (и, соответственно, характеристика датчика) изменилась бы на обратную, однако в целом устройство бы функционировало по-прежнему.
10. Какие имеются онлайн-среды для моделирования схем на основе микроконтроллерной платформы Arduino Uno? Приведите примеры и охарактеризуйте каждую из сред.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Приказ Министерства образования и науки РФ от 19 сентября 2017 г. № 929 «Об утверждении федерального государственного образовательного стандарта высшего образования – бакалавриат по направлению подготовки 09.03.01 Информатика и вычислительная техника» [Электронный ресурс]. – URL: http://fgosvo.ru/uploadfiles/FGOS%20VO%203++/Bak/090301_B_3_12102017.pdf. – URL: http://fgosvo.ru/fgosvo/151/150/24/9 (дата обращения: 10.01.2020).
2. Сайт, посвященный технопарку РГСУ «Технопарк равных возможностей» [Электронный ресурс]. – URL: http://technopark.rgsu.net/ (дата обращения: 10.01.2020).
3. Сайт, посвященный разработке электронных схем и программированию, фирма «Амперка» [Электронный ресурс]. – URL: http://amperka.ru (дата обращения: 10.01.2020).

4. *Ерпелев А.В.* Использование свободного ПО в учебном процессе на примере разработки устройств «умного дома» с использованием микроконтроллеров Arduino и Iskra JS [Текст] / А.В. Ерпелев, В.Л. Симонов, А.П. Рубанкова // Четырнадцатая конференция «Свободное программное обеспечение в высшей школе»: материалы конференции. – М.: МАКС Пресс, 2019. – С. 34–36.

5. *Симонов В.Л.* Проектирование студентами высших учебных заведений реальных устройств при изучении ряда дисциплин, связанных с обработкой информации и данных [Текст] / В.Л. Симонов, М.М. Аметова, Н.А. Хмыров [и др.] // Информационные технологии в образовании: материалы науч.-практ. конф. – Саратов, 2017. – С. 296–298.

6. *Бакин М.А.* Применение технологий умного дома в интернатах и других организациях социальной сферы (тезисы доклада) [Текст] / М.А. Бакин, Е.Б. Венина, А.М. Дербышева [и др.] // Сборник докладов II Всероссийского форума научной молодежи «Богатство России» (Москва, 10–11 декабря 2018 г.); ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)». – М.: Изд-во МГТУ им. Н.Э. Баумана, 2019. – 251 с. – С. 156–157. – ISBN 978-5-7038-5090-9; [Электронный ресурс]. – URL: http://bogatstvo.bmstu.ru/?page_id=143 (дата обращения: 10.01.2020).

4. Лабораторные работы в конструкторе (AR) ELIGOVISION TOOLBOX (EV TOOLBOX)

4.1. ПОНЯТИЕ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ В КОНСТРУКТОРЕ (AR) ELIGOVISION TOOLBOX (EV TOOLBOX)

Дополненная реальность (AR) - это интерактивная технология, которая позволяет дорисовывать "поверх" изображения с камеры 3d модели таким образом, что создается впечатление, что они непосредственно находятся в реальном мире.

EV Toolbox - простой и удобный программный продукт, позволяющий 3D дизайнерам и программистам создавать собственные презентации на основе технологии дополненной и виртуальной реальности.

EV Toolbox - это набор программ для создания и просмотра презентаций с дополненной реальностью. EV Toolbox состоит из двух программ: EV Studio и prEView. С помощью EV Studio создается проект, в котором задается сценарий презентации. Проект из EV Studio можно экспортировать как eva-пакет. Проигрыватель prEView позволяет просматривать eva-пакет.

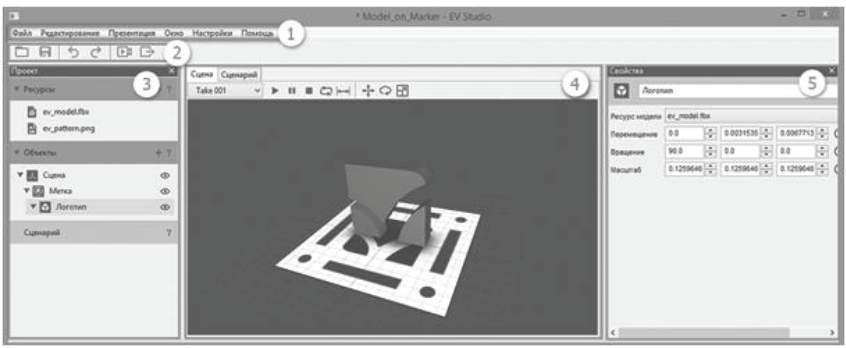
4.2. ИНТЕРФЕЙС EV TOOLBOX

4.2.1. Экран приветствия

Экран, который появляется при запуске приложения EV Studio со списком основных действий.

действие	описание
НОВЫЙ ПРОЕКТ	создать новый проект
ОТКРЫТЬ ПРОЕКТ	открыть ранее созданный проект
НЕДАВНИЕ ПРОЕКТЫ	открыть ранее созданный проект из списка ниже
ПРИМЕРЫ ПРОЕКТОВ	открыть пример проекта из списка ниже
ПОМОЩЬ	перейти на страницу документации
ВИДЕОУРОКИ	Перейти на канал EligoVision Toolbox в youtube с обучающими видео
РАСПЕЧАТАТЬ МЕТКИ	открыть pdf файл с изображением меток используемых в примерах
GOOGLE PLAY	перейти на страницу google play с примерами приложений созданных в EV Studio

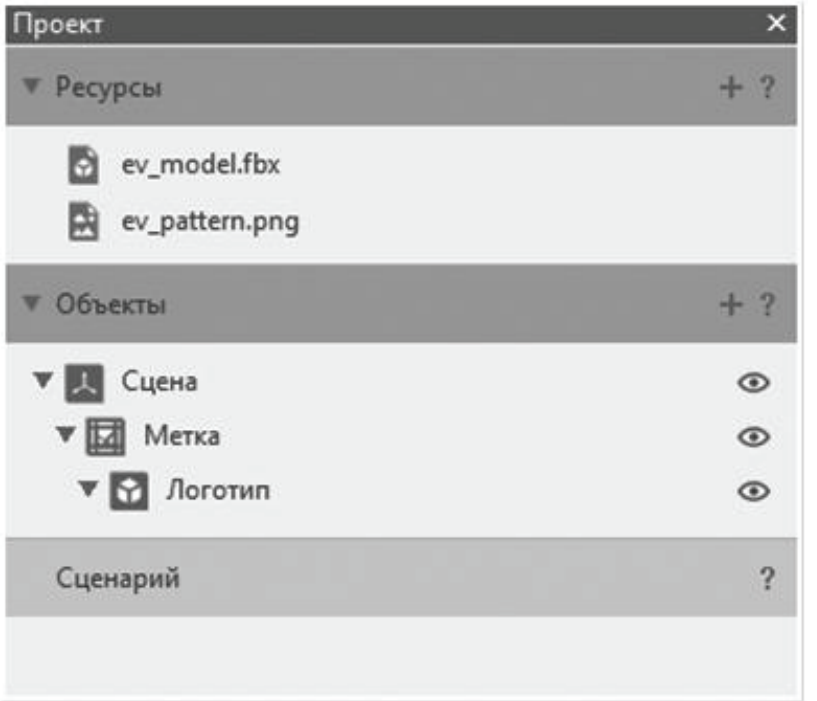
4.2.2. Основной экран



1. Меню
2. Панель инструментов
3. Панель "Проект"
4. Рабочее пространство
5. Свойства

Окна 3, 5 можно располагать по своему усмотрению перемещая и закрепляя на новом месте или за пределами окна EV Studio

4.2.3. Панель «Проект»



Панель проектов состоит из двух вкладок:

- **Ресурсы**, в которой отображается список ресурсов, добавленных в проект. Подробнее о ресурсах можно узнать в разделе "Ресурсы"
- **Объекты**, в которой отображается список объектов добавленных в проект. Объекты отображаются в иерархическом порядке. Объекты можно перемещать в структуре презентации. Чтобы скрыть объект нажмите на иконку "глаз" (объект будет скрыт при просмотре презентации).

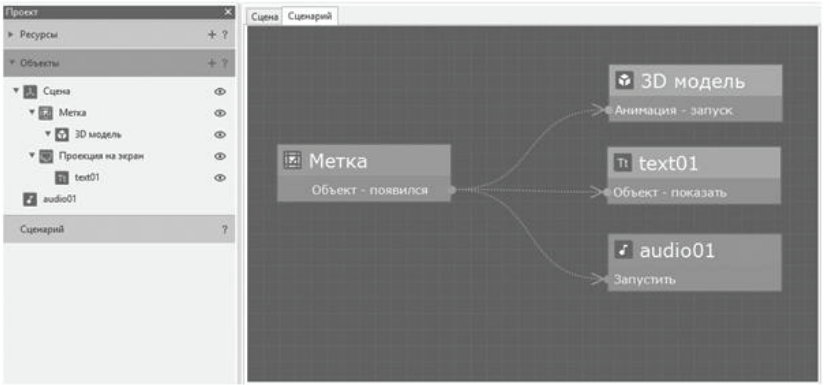
Вкладку можно скрывать/раскрывать, нажатием на серую плашку с названием вкладки. Нажмите "+" чтобы добавить элемент, нажмите "?" чтобы получить справку по вкладке. Чтобы быстро присвоить ресурс объекту, перетащите его на объект.

Вкладка	Описание
Ресурсы	Список ресурсов добавленных в проект.
Объекты	Список объектов добавленных в проект. Объекты отображаются в иерархическом порядке. Чтобы скрыть объект нажмите на иконку "глаз".

4.2.4. Сценарий

Во вкладке "Сценарий" рабочего пространства отображается схема взаимодействия событий и действий объектов. Окно представляет собой поле сценария на которое добавляются блоки объектов, состоящих из списка событий и действий для данного объекта. Событие соединяется с действием кривой линией. Блоки можно свободно перемещать по полю сценария, зажав левую кнопку мыши. Поле можно масштабировать, вращая колесико мыши, а также перемещаться по полю, зажав третью кнопку мыши(колесико).

- **Ctrl+1** - задать 100% масштаб
- **Ctrl+2** - задать масштаб так, чтобы в окно поместилось все поле, занятое блоками и соединениями.



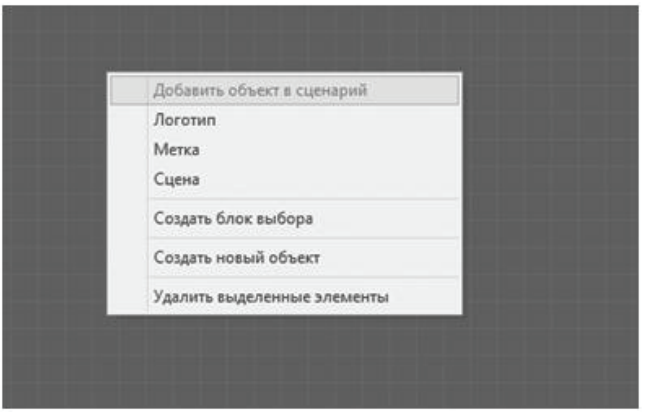
Пример простого сценария. При появлении в кадре метки и распознания ее, запускается анимация в модели, которая находится на метке, и появляется надпись в проекции на экран.

4.2.5. События, действия и свойства объектов

- **Действие** - это команда выполнить некоторую операцию. Например, у объекта "Аудио" действие "проиграть" означает, запустить проигрывание аудиофайла. Или при выполнении действия "объект - скрыть" у объекта "Модель" - модель исчезнет.
- **Событие** - это сигнал программы о том, что с объектом что-то произошло. Например, у объекта текст событие "нажатие" происходит, когда пользователь нажимает на изображение. Или у объекта "Метка" событие "скрылся" происходит, когда камера потеряла метку (перестала распознаваться). События нужны, чтобы указать, когда выполнить действие. У действий может быть событие о его запуске. Например, у объекта "Модель" событие "анимация началась" произойдет тогда, когда будет выполнено действие "анимация - запуск".

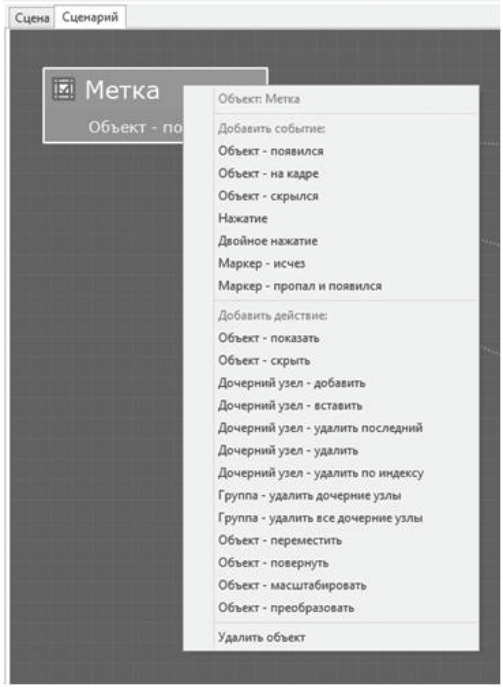
- **Свойства** - это дополнительные параметры действий и событий объекта. Например, у объекта "модель" у действия "анимация- запуск" имеется параметр "Название", который позволяет выбрать какую именно анимацию запустить.

4.2.6. Создание нового блока объекта



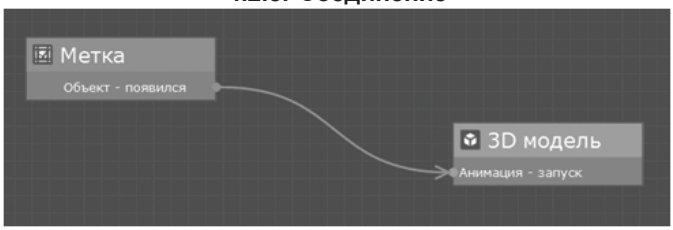
Чтобы создать новый блок нажмите правую кнопку мыши и выберите объект из списка. В списке отображаются все объекты проекта, а также блок выбора.

4.2.7. Добавление событий и действий объекта



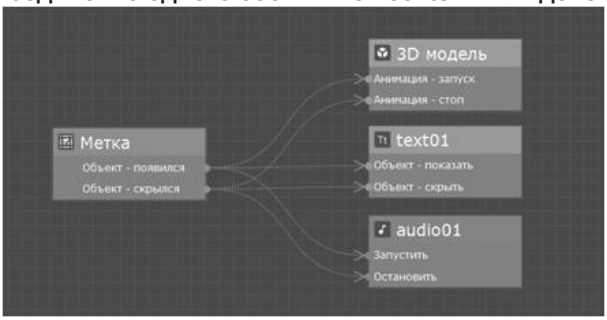
Чтобы добавить в блок событие или действие нажмите правую кнопку мыши на блоке и выберите событие или действие из списка.

4.2.8. Соединение



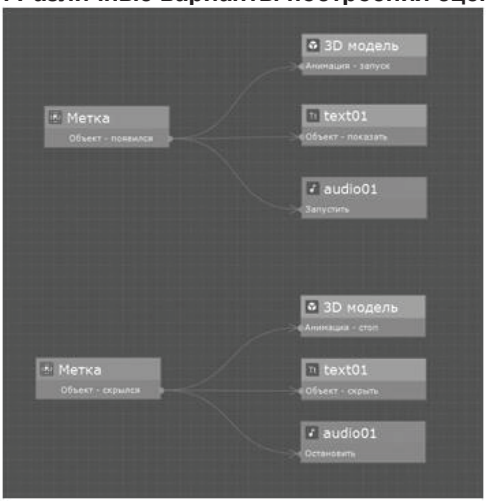
Чтобы соединить событие с действием нажмите дважды левой кнопкой мыши на кружочек справа от события и протяните линию до нужного вам действия.

4.2.9. Соединение одного события с несколькими действиями



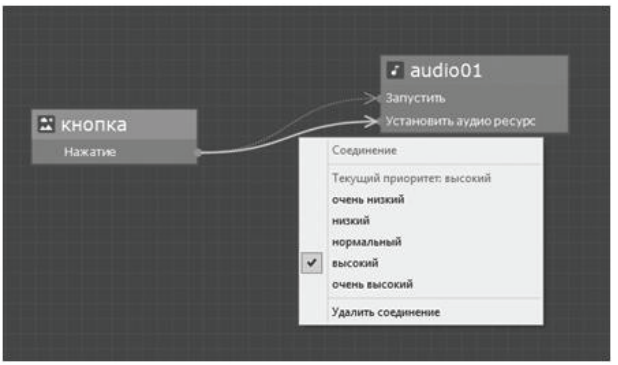
Одно событие можно соединить с несколькими действиями.

4.2.10. Различные варианты построения сценария



Для одного объекта возможно создавать несколько блоков. При создании сложных сценариев с большим количеством взаимодействующих блоков, для удобства восприятия лучше разделять блоки объекта. На двух изображениях выше сценарии идентичны, во втором случае блоки сгруппированы по событию метки.

4.2.11. Приоритеты соединений

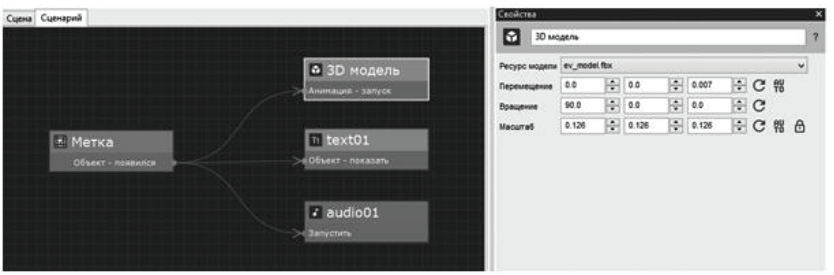


У соединений можно выставить приоритеты, нажав правую кнопку мыши на линии. Это нужно в случаях, когда важна очередность выполнения действий от одного события. Например, как на изображении выше. В данном примере необходимо чтобы по нажатию на кнопку изменился ресурс у объекта audio, а затем запустилось проигрывание нового ресурса. Для этого у действия "изменить ресурс" установлен высокий приоритет. Если выполнить эти действия в обратном порядке, проигрывание нового ресурса не произойдет.

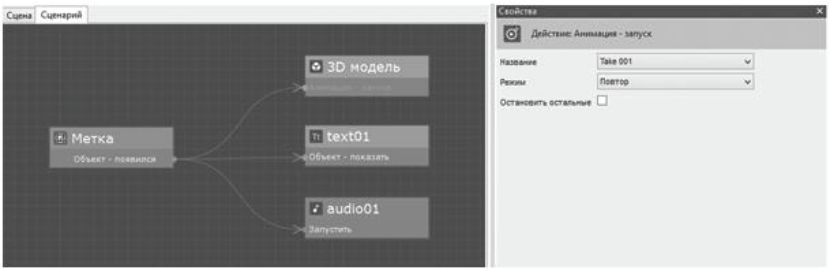
По умолчанию у соединений нормальный приоритет. Если у различных действий одинаковый приоритет, то очередность выполнения действий случайна.

В большинстве случаев очередность выполнения не важна, так как все действия выполняются очень быстро (за один кадр).

4.2.12. Свойства



При выделении блока с объектом в окне "Свойства" отображаются параметры объекта. Аналогично тому если выделить объект в окне "Проект".



При выделении события или действия в блоке объекта в окне "Свойства" отображаются параметры события или действия если таковые имеются.

4.3. ПОДДЕРЖКА ФОРМАТА FBX

Из формата FBX в EV Studio экспортируется:

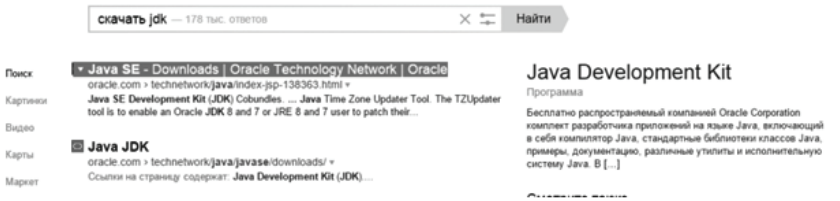
- **Геометрия.**
- **Текстуры** в форматах **.png, .bmp, .jpg, .jpeg, .tiff, tif** (rgb, rgba, 8bit). Поддерживаются карты диффуза, спекуляра, нормалей, отражения, маски и композитные карты, экспортируемые из 3DS Max материала стандарт.
- **Анимация**. Поддерживается анимация трансформации (сдвиг,поворот, масштабирование), скелетная анимация, морф анимация, анимационные клипы (takes, экспорт из Game Exporter 3DS Max и Maya)
- **Источники Света** типа Omni и Target максимум 3. Если в моделях нет источников, то используется источник из камеры.

4.4. НАСТРОЙКА СРЕДЫ ОКРУЖЕНИЯ ANDROID

Для экспорта в Android необходимо настроить среду окружения Android. Для этого необходимо установить Java development kit (JDK) и Android software development kit (Android SDK).

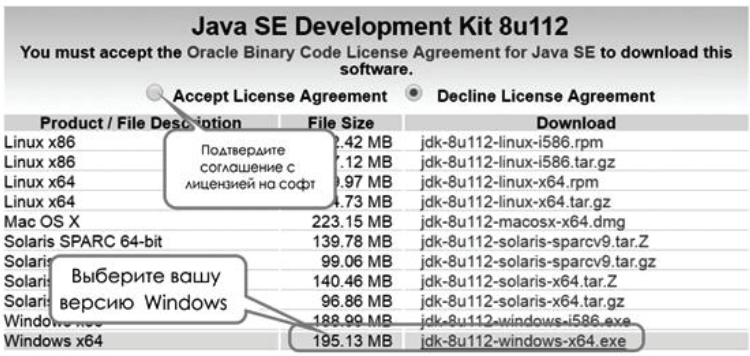
4.4.1. Установка JDK

- Первый же результат в поисковой системе выдаст нужный сайт

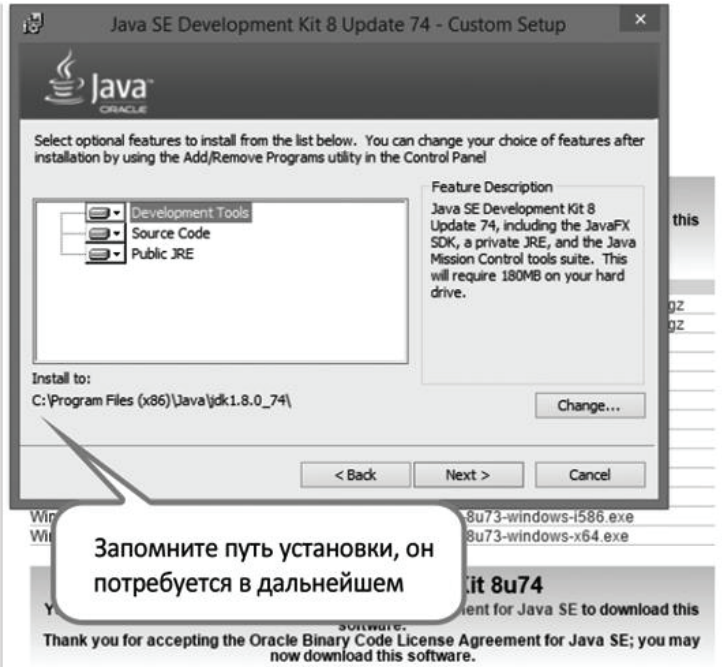


- Нажмите кнопку как на изображении

- в новом окне скачайте JDK



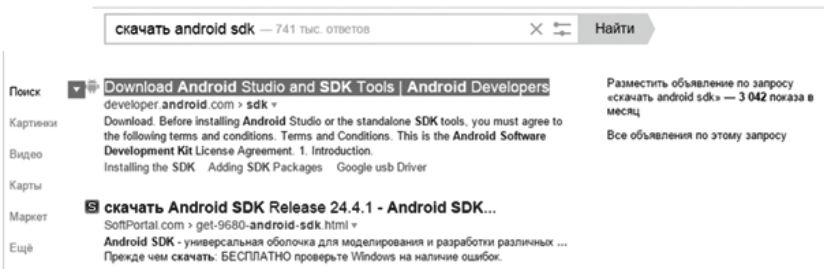
- запустите установку и дождитесь окончания



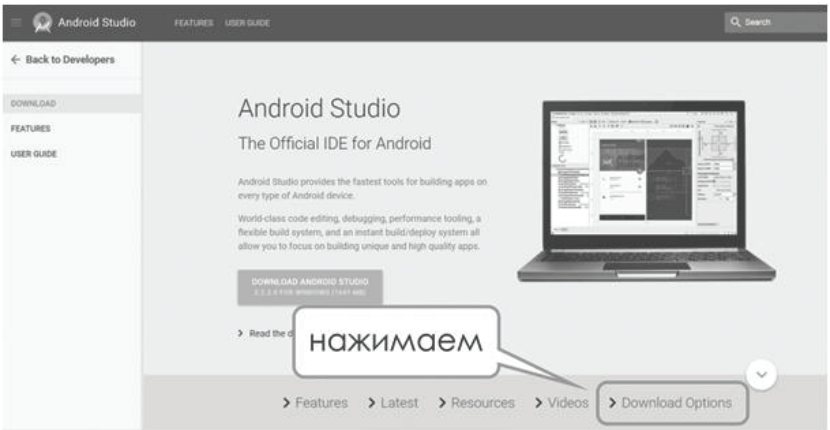
- После полной установки закройте окно установщика

4.4.2. Установка Android SDK

- перейдите на сайт developer.android.com/studio. Его легко найти по запросу:



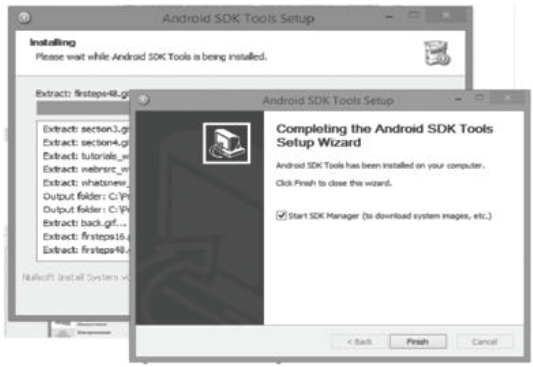
- нажмите на кнопку как на изображении



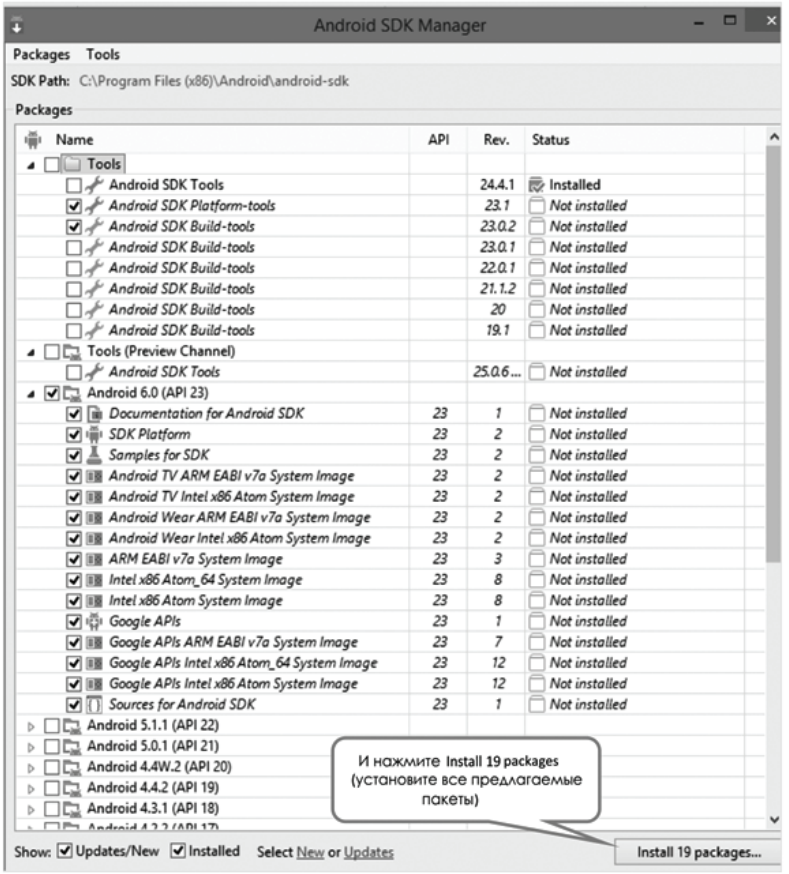
- в разделе Get just the command line tools выберите installer...

- дождитесь окончания установки

- после окончания установки запустите SDK



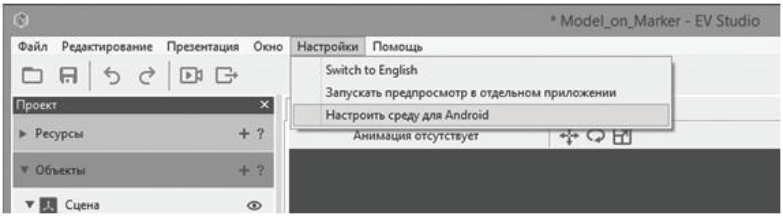
- manager
- в окне "Android SDK manager" поставте галочки как на изображении и нажмите кнопку install...



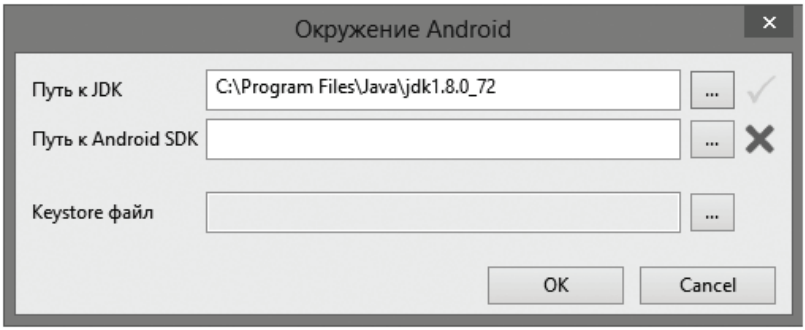
- дождитесь окончания установки, которая может занять длительное время

Указание путей к JDK и Android SDK в окне "Окружение Android"

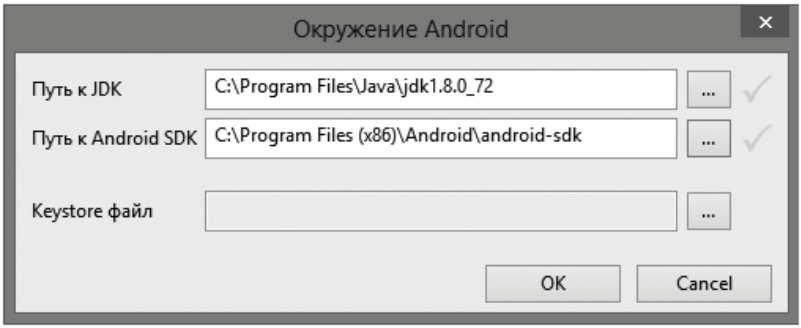
- после установки JDK и Android SDK в окне "Окружение Android", в которое можно прейти выбрав в меню > настройки > настроить среду для Android



● укажите путь до папки с JDK (по умолчанию устанавливается в C:/Program Files/Java/jdk...)



● укажите путь до папки с Android SDK (по умолчанию устанавливается в C:Files (x86)-sdk)



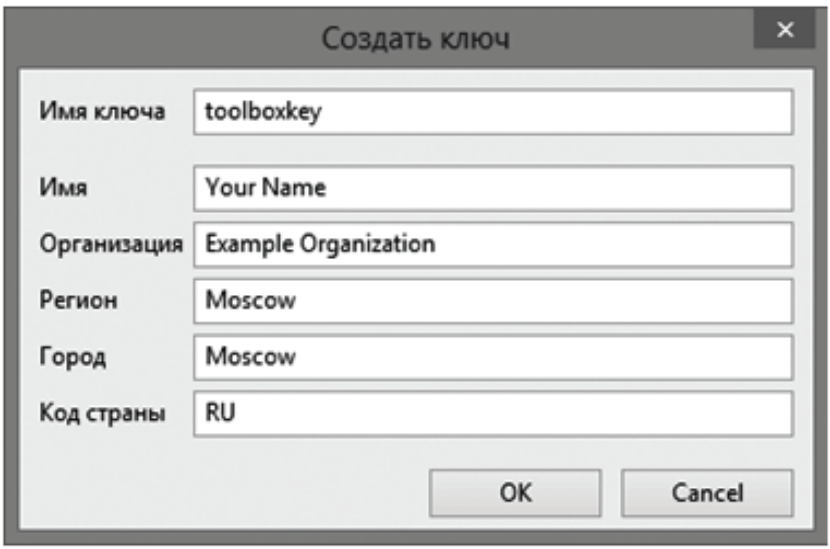
● если все указано верно, то крестики сменятся на галочки, и можно будет экспортировать приложения для android.

Keystore

Keystore - файл android.keystore, который хранится на диске и содержит сертификат вашего приложения. Keystore уникален для каждого разработчика. Система Android требует, чтобы все установленные приложения имели Keystore. Сертификат необходим для идентификации автора приложений.

Когда разработчик обновляет приложение или добавляет новое в в Google Play под своим именем, необходимо чтобы у нового приложения был тот же Keystore, иначе загрузить приложение не удастся. Важно знать, что уникальный Keystore генерируется только один раз. Если удалить EV Studio, или Keystore файл, повторное его создание с такими же данными сгенерирует другой Keystore. Если вы загрузили приложение в Google Play в первый раз, и планируете обновления, советуем скопировать и сохранить keystore файл в надежном месте. В окне "Окружение Android" указано местонахождение Keystore файла на диске(по умолчанию C:Studio.keystore). Если вы хотите использовать свой Keystore в новой EV Studio скопируйте его в указанное место на диске.

Окно "Создать ключ"



В окне "Создать ключ" указываются личные данные на основе которых сгенерируется Keystore.

ЛАБОРАТОРНАЯ РАБОТА № 1

Создание проекта: «Планеты земной группы».

Цель работы: разработать проект дополненной реальности (AR приложение) для мобильного устройства в конструкторе проектов дополненной реальности EV Toolbox.

Результат: архив проекта с вложенными локальными ресурсами, готовый к экспорту в apk файл.

Дано: 3 трехмерные модели планет земной группы с анимацией.

Все прочие данные для проекта (тексты, изображения, видео, аудио и проч.) необходимо создать (найти и скачать) самостоятельно, при необходимости конвертировав их в требуемые форматы.

Особенность: маркерная технология распознавания. Все 3 объекта собраны в один проект.

Порядок выполнения проекта

1). Создать 3 разные метки для планет Земля, Меркурий и Марс.

2). Разместить на каждой из меток соответствующую 3D модель планеты.

3). Вывести на экране при запуске проекта подсказку-инструкцию, объясняющую пользователю, что нужно сделать (это может быть текстовый блок или изображение). Этот блок должен скрыться, как только в области камеры появится любая из меток.

4). Разместить на экране (проекция на экран) кнопку с текстом.

5). При нажатии на кнопку запустить анимацию вращения каждой планеты

6). После запуска анимации скрыть первую кнопку и показать сопроводительную информацию – интересные данные о планете на экране. Это может быть видео, текстовый блок, изображение или аудиофайл – можно все сразу или по очереди, главное, чтобы это было удобно для пользователя.

7). После просмотра информационного блока разместить на экране кнопку с приглашением ответить на проверочный вопрос.

8). Придумать один вопрос и два варианта ответа по каждой планете. При нажатии кнопки «тест» вывести на экран вопрос и варианты ответа (текстовые блоки, можно 3D текст).

9). По выбору одного из вариантов ответа показать на экране результат – правильно или неправильно.

10). В случае если выбран правильный ответ – предложить изучить следующую из планет земной группы.

Важно: необходимо учитывать тот факт, что приложение будет создаваться под мобильное устройство, т.е. под экраны смартфонов (текст должен быть читаемым).

Пример выполнения лабораторной работы.
Создание проекта: «Планеты земной группы».

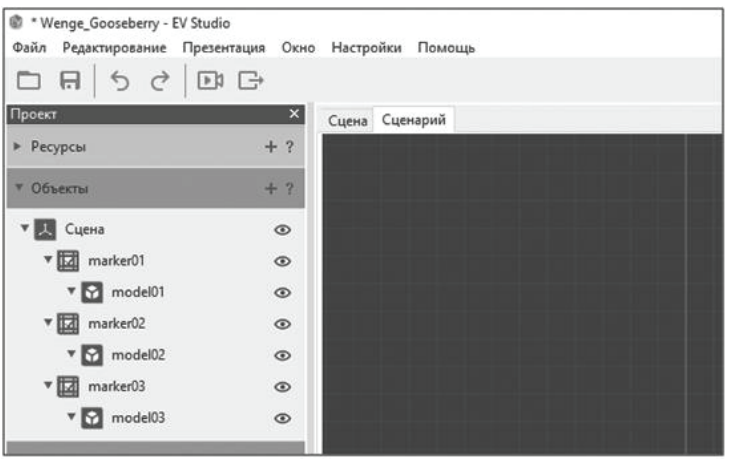
1. Для создания проекта «Планеты земной группы» необходимо подготовить 3 маркера, требуется его печатная версия и файл с изображением его внутренней части (паттерна).

Запустить приложение EV Studio и открыть **НОВЫЙ ПРОЕКТ**

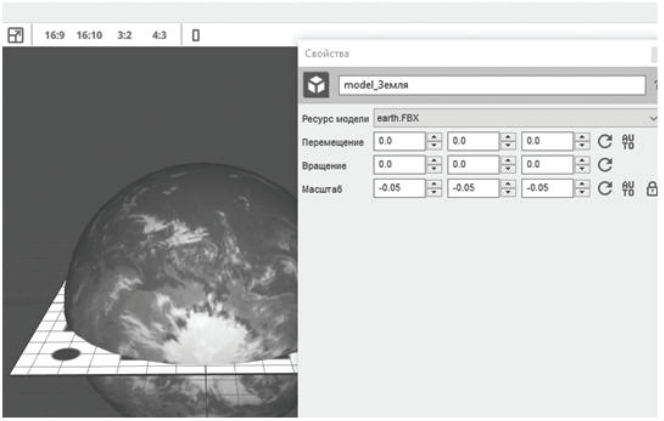
2. На панель **Проект**, в раздел **Ресурсы** необходимо добавить нужные ресурсы (три паттерна и 3D Модели с изображением Земли, Марса и Меркурия)

3. В раздел **Объекты** добавить три объекта Метка и три объекта Модель

4. Объекты расположить в иерархическом порядке.

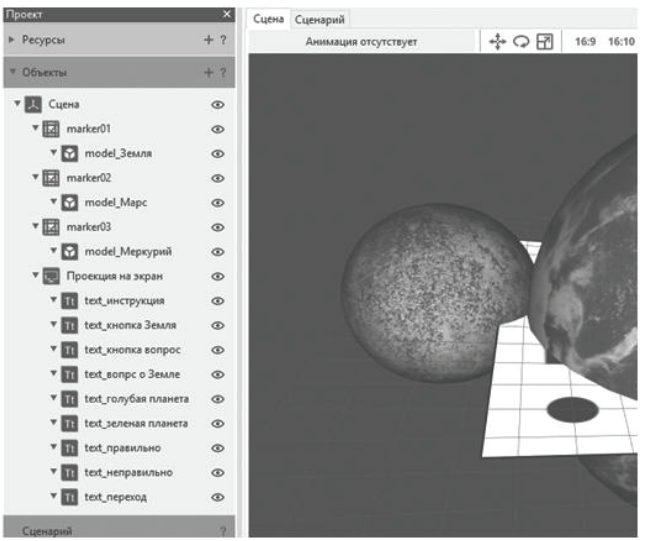


5. В разделе Свойства переименовать название моделей (для удобства) и определить ресурс моделей. Так же настроить масштаб объектов.



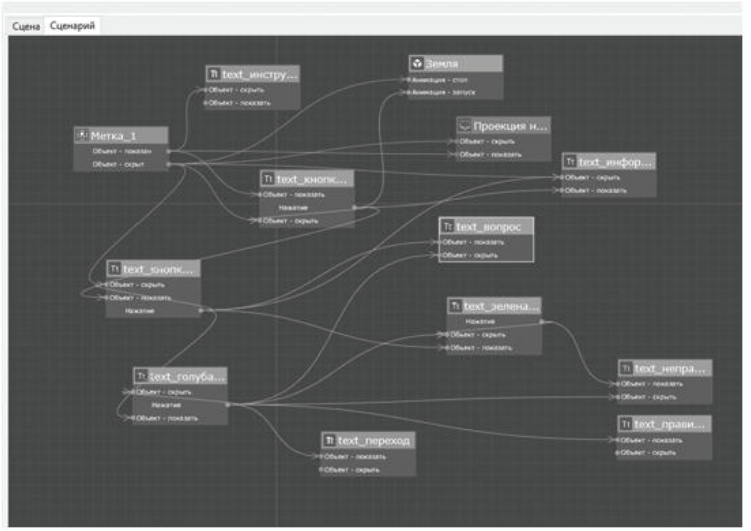
6. В раздел **Объекты** добавить объект **Проекция на экран**.

7. В раздел Объекты добавить объекты Текст, Текст 3D, Видео или Изображение (на усмотрение, в зависимости от того, как вы хотите представить информацию). В разделе Свойства дать соответствующее название каждому объекту, ввести нужный текст, а также положение, размер шрифта, цвет текста и т.д.



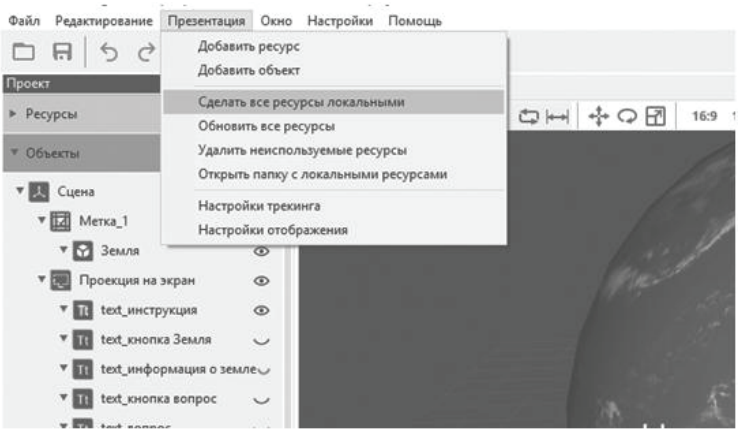
8. Необходимо использовать "глаз", чтобы скрыть объекты (объект будет скрыт при просмотре презентации).

9. Перейти на вкладку Сценарий рабочего пространства, и добавить необходимые блоки объектов. Отобразить схему взаимодействия событий и действий объектов в нужном порядке. Примеры схем:



10. Аналогично выполнить задание для планет Меркурий и Марс.

Примечание: перед сохранением проекта, необходимо зайти в меню **Презентация** и нажать **Сделать все ресурсы локальными**



После завершения проекта необходимо настроить среду Android для экспорта файлов apk. (см. руководство пользователя EV TOOLBOX, раздел настройка среды окружения android).

ВОПРОСЫ К ГЛАВЕ 4

1. Понятие дополненной реальности.
2. Маркерная технология распознавания.
3. Безмаркерная технология распознавания.
4. Предназначение вкладок «Ресурсы».
5. Предназначение вкладок «Объекты».
6. Что отображается в «Сценарии» рабочего пространства?
7. Пояснить команды события, действия и свойства объектов.
8. Алгоритм создания нового блока объекта.
9. Каким образом происходит добавление событий и действий объекта?
10. Предназначение объекта «Переключатель».
11. Как происходит соединение одного события с несколькими действиями?
12. Какие приоритеты соединений объектов можно выделить?
13. Поддержка формата FBX. Экспорт формата FBX в EV Studio.
14. Алгоритм настройки среды окружения ANDROID.
15. Для каких целей выполняется действие «Сделать все ресурсы локальными»?

5. Интеллектуальные системы и технологии в инженерии знаний

5.1. ВВЕДЕНИЕ

Термин **«искусственный интеллект»** (artificial intelligence) был предложен в 1956 году. Слово intelligence означает «умение рассуждать разумно», а вовсе не «интеллект», для которого есть термин intellect.

Искусственный интеллект (ИИ) – это одно из направлений информатики, целью которого является разработка аппаратно-программных средств, позволяющих пользователю-непрограммисту ставить и решать свои, традиционно считающиеся интеллектуальными, задачи, общаясь с компьютером на ограниченном подмножестве естественного языка.

ИИ занимается изучением разумного поведения (у людей, животных и машин) и пытается найти способы моделирования подобного поведения в любом типе искусственно созданного механизма. Несмотря на то что термину больше полувека, единого определения его не существует.

Материалы учебного пособия, в соответствии с ФГОС ВО по направлению подготовки 09.04.01 – «ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА» (УРОВЕНЬ МАГИСТРАТУРЫ), направлены на формирование следующих компетенций:

- **ОК-1** – способность совершенствовать и развивать свой интеллектуальный и общекультурный уровень;

- **ОК-2** – способностью понимать роль науки в развитии цивилизации, соотношение науки и техники, иметь представление о связанных с ними современных социальных и этических проблемах, понимать ценность научной рациональности и ее исторических типов;

- **ОК-4** – способность заниматься научными исследованиями.

Искусственный интеллект всегда был междисциплинарной наукой, являясь одновременно и наукой, и искусством, и техникой, и психологией. Методы искусственного интеллекта разнообразны. Они активно заимствуются из других наук, адаптируются и изменяются под решаемую задачу. Для создания интеллектуальной системы необходимо привлекать специалистов из прикладной области, поэтому в рамках искусственного интеллекта сотрудничают лингвисты, нейрофизиологи, психологи, экономисты, информатики, программисты и т.д.

5.2. ПОНЯТИЕ ИНТЕЛЛЕКТУАЛЬНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Интеллектуальная система (ИС) – автоматизированная система, основанная на знаниях, или комплекс программных, лингвистических и логико-математических средств для реализации основной задачи – осуществления поддержки деятельности человека и поиска информации в режиме продвинутого диалога на естественном языке.

Кроме того, информационно-вычислительными системами с интеллектуальной поддержкой для решения сложных задач называют те системы, в которых логическая обработка информации превалирует над вычислительной.

Таким образом, любая информационная система, решающая интеллектуальную задачу или использующая методы искусственного интеллекта, относится к интеллектуальным.

Исследователи, работающие в этом направлении, надеются достичь такого понимания механизмов интеллекта, при котором можно будет составлять компьютерные программы с человеческим или более высоким уровнем интеллекта. Общий подход состоит в разработке методов решения задач, для которых отсутствуют формальные алгоритмы: понимание естественного языка, обучение, доказательство теорем, распознавание сложных образов и т.д. Теоретические исследования направлены на изучение интеллектуальных процессов и создание соответствующих математических моделей. Экспериментальные работы ведутся путем составления компьютерных программ и создания машин, решающих частные интеллектуальные задачи или разумно ведущих себя в заданной ситуации. Систематические исследования в области искусственного интеллекта начались лишь с появлением цифрового компьютера. Первая научная статья по искусственному интеллекту была опубликована в 1950 А. Тьюрингом.

5.3. НАПРАВЛЕНИЯ ИССЛЕДОВАНИЙ В ОБЛАСТИ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Развитие интеллектуальных систем на современном этапе идет в соответствии с тремя направлениями исследований.

Первое направление объектом исследований рассматривает структуру и механизмы работы мозга человека, а конечной целью – раскрытие тайн мышления. Необходимыми этапами исследований в этом направлении являются построение моделей интеллектуальной деятельности на основе психофизиологических данных.

Второе направление в качестве объекта исследования рассматривает искусственную интеллектуальную систему. Здесь речь идет о моделировании интеллектуальной деятельности с помощью

вычислительных машин. Целью работ в этом направлении является создание программного обеспечения, позволяющего решать некоторые виды интеллектуальных задач так же, как их решил бы человек.

Третье направление ориентировано на создание человеко-машинных, или, как еще говорят – интерактивных, интеллектуальных систем. Важнейшими проблемами в этих исследованиях является организация **семантически безупречного диалога между человеком и такой системой.**

5.4. КЛАССИФИКАЦИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Для интеллектуальных информационных систем характерны следующие признаки:

- развитые коммуникативные способности;
- умение решать сложные плохо формализуемые задачи;
- способность к самообучению;
- адаптивность.

Коммуникативные способности ИС характеризуют способ взаимодействия (интерфейса) конечного пользователя с системой, в частности возможность формулирования произвольного запроса в диалоге с ИС на языке, максимально приближенном к естественному.

Сложные плохо формализуемые задачи – это задачи, которые требуют построения оригинального алгоритма решения в зависимости от конкретной ситуации, для которой могут быть характерны неопределенность и динамичность исходных данных и знаний.

Способность к самообучению – это возможность автоматического извлечения знаний для решения задач из накопленного опыта конкретных ситуаций.

Адаптивность – способность к развитию системы в соответствии с объективными изменениями модели проблемной области. В соответствии с перечисленными признаками ИС делятся на (данная классификация – одна из возможных) (рисунок 5.1):

- системы с коммутативными способностями (с интеллектуальным интерфейсом);
- экспертные системы (системы для решения сложных задач);
- самообучающиеся системы (системы, способные к самообучению);
- адаптивные системы (адаптивные информационные системы).

Рисунок 5.1 – Классификация интеллектуальных информационных систем по типам систем

Интеллектуальные базы данных отличаются от обычных баз данных возможностью выборки по запросу необходимой информации, которая может явно не храниться, а выводиться из имеющейся в базе данных.

Естественно-языковой интерфейс предполагает трансляцию естественно-языковых конструкций на внутримашинный уровень представления знаний. Для этого необходимо решать задачи морфологического, синтаксического и семантического анализа и синтеза высказываний на естественном языке. Так, морфологический анализ предполагает распознавание и проверку правильности написания слов по словарям, синтаксический контроль – разложение входных сообщений на отдельные компоненты (определение структуры) с проверкой соответствия грамматическим правилам внутреннего представления знаний и выявления недостающих частей и, наконец, семантический анализ – установление смысловой правильности синтаксических конструкций. Синтез

высказываний решает обратную задачу преобразования внутреннего представления информации в естественно-языковое.

Естественно-языковой интерфейс используется для:

- доступа к интеллектуальным базам данных;
- контекстного поиска документальной текстовой информации;
- голосового ввода команд в системах управления;
- машинного перевода с иностранных языков.

Гипертекстовые системы предназначены для реализации поиска по ключевым словам в базах текстовой информации. Интеллектуальные гипертекстовые системы отличаются возможностью более сложной семантической организации ключевых слов, которая отражает различные смысловые отношения терминов. Таким образом, механизм поиска работает прежде всего с базой знаний ключевых слов, а уже затем непосредственно с текстом. В более широком плане сказанное распространяется и на поиск мультимедийной информации, включающей, помимо текстовой, и цифровую информацию.

Системы контекстной помощи можно рассматривать как частный случай интеллектуальных гипертекстовых и естественно-языковых систем. В отличие от обычных систем помощи, навязывающих пользователю схему поиска требуемой информации, в системах контекстной помощи пользователь описывает проблему (ситуацию), а система с помощью дополнительного диалога ее конкретизирует, и сама выполняет поиск относящихся к ситуации рекомендаций. Такие системы относятся к классу систем распространения знаний (Knowledge Publishing) и создаются как приложение к системам документации (например, технической документации по эксплуатации товаров).

Системы когнитивной графики позволяют осуществлять интерфейс пользователя с ИС с помощью графических образов, которые генерируются в соответствии с происходящими событиями. Такие системы используются в мониторинге и управлении оперативными процессами. Графические образы в наглядном и интегрированном виде описывают множество параметров изучаемой ситуации. Например, состояние сложного управляемого объекта отображается в виде человеческого лица, на котором каждая черта отвечает за какой-либо параметр, а общее выражение лица дает интегрированную характеристику ситуации. Системы когнитивной графики широко используются также в обучающих и тренажерных системах на основе использования принципов виртуальной реальности, когда графические образы моделируют ситуации, в которых обучаемому необходимо принимать решения и выполнять определенные действия.

Экспертные системы предназначены для решения задач на основе накапливаемой базы знаний, отражающей опыт работы экспертов в рассматриваемой проблемной области.

Многоагентные системы – это динамические системы, для которых характерна интеграция в базе знаний нескольких разнородных источников знаний, обменивающихся между собой получаемыми результатами на динамической основе.

Для многоагентных систем характерны следующие особенности:

- проведение альтернативных рассуждений на основе использования различных источников знаний с механизмом устранения противоречий;
- распределенное решение проблем, которые разбиваются на параллельно решаемые подпроблемы, соответствующие самостоятельным источникам знаний;
- применение множества стратегий работы механизма вывода заключений в зависимости от типа решаемой проблемы;
- обработка больших массивов данных, содержащихся в базе данных;
- использование различных математических моделей и внешних процедур, хранимых в базе моделей;
- способность прерывания решения задач в связи с необходимостью получения дополнительных данных и знаний от пользователей, моделей, параллельно решаемых подпроблем.

Самообучающиеся системы основаны на методах автоматической классификации примеров ситуаций реальной практики.

Характерными признаками самообучающихся систем являются:

- самообучающиеся системы «с учителем», когда для каждого примера задается в явном виде значение признака его принадлежности некоторому классу ситуаций (классообразующего признака);
- самообучающиеся системы «без учителя», когда по степени близости значений признаков классификации система сама выделяет классы ситуаций.

Индуктивные системы используют обобщение примеров по принципу от частного к общему. Процесс классификации примеров осуществляется следующим образом:

1. Выбирается признак классификации из множества заданных (либо последовательно, либо по какому-либо правилу, например, в соответствии с максимальным числом получаемых подмножеств примеров).
2. По значению выбранного признака множество примеров разбивается на подмножества.
3. Выполняется проверка, принадлежит ли каждое образовавшееся подмножество примеров одному подклассу.
4. Если какое-то подмножество примеров принадлежит одному подклассу, то есть у всех примеров подмножества совпадает значение классообразующего признака, то процесс классификации заканчивается (при этом остальные признаки классификации не рассматриваются).
5. Для подмножеств примеров с несовпадающим значением классообразующего признака процесс классификации продолжается,

начиная с пункта 1 (каждое подмножество примеров становится классифицируемым множеством).

Нейронные сети представляют собой устройства параллельных вычислений, состоящие из множества взаимодействующих простых процессоров. Каждый процессор такой сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам.

В экспертных системах, *основанных на прецедентах* (аналогиях), база знаний содержит описания не обобщенных ситуаций, а собственно сами ситуации или прецеденты.

Поиск решения проблемы в экспертных системах, основанных на прецедентах, сводится к поиску по аналогии (то есть абдуктивный вывод от частного к частному).

В отличие от интеллектуальной базы данных, *информационное хранилище* представляет собой хранилище извлеченной значимой информации из оперативной базы данных, которое предназначено для оперативного ситуационного анализа данных (реализации **OLAP-технологии**).

Типичными задачами оперативного ситуационного анализа являются:

- определение профиля потребителей конкретных объектов хранения;
- предсказание изменений объектов хранения во времени;
- анализ зависимостей признаков ситуаций (корреляционный анализ).

Адаптивная информационная система – это информационная система, которая изменяет свою структуру в соответствии с изменением модели проблемной области.

При этом:

- адаптивная информационная система должна в каждый момент времени адекватно поддерживать организацию бизнес-процессов;
- адаптивная информационная система должна проводить адаптацию всякий раз, как возникает потребность в реорганизации бизнес-процессов;
- реконструкция информационной системы должна проводиться быстро и с минимальными затратами.

Ядром адаптивной информационной системы является постоянно развиваемая модель проблемной области (предприятия), поддерживаемая в специальной базе знаний – репозитории. На основе ядра осуществляется генерация или конфигурация программного обеспечения. Таким образом, проектирование и адаптация ИС сводится, прежде всего, к построению модели проблемной области и ее своевременной корректировке.

Так как нет общепринятого определения, четкую единую классификацию интеллектуальных информационных систем дать затруднительно.

Если рассматривать интеллектуальные информационные системы с точки зрения решаемой задачи, то можно выделить системы управления и справочные системы, системы компьютерной лингвистики, системы

распознавания, игровые системы и системы создания интеллектуальных информационных систем (рисунок 5.2)

Рисунок 5.2 – Классификация интеллектуальных информационных систем по решаемым задачам

При этом системы могут решать не одну, а несколько задач или в процессе решения одной задачи решать и ряд других. Например, при обучении иностранному языку система может решать задачи распознавания речи обучаемого, тестировать, отвечать на вопросы, переводить тексты с одного языка на другой и поддерживать естественно-языковой интерфейс работы.

Если классифицировать интеллектуальные информационные системы по критерию «используемые методы», то они делятся на жесткие, мягкие и гибридные (рисунок 5.3).

Рисунок 5.3 – Классификация интеллектуальных информационных систем по методам

Мягкие вычисления (Soft Computing) – это сложная компьютерная методология, основанная на нечеткой логике, генетических вычислениях, нейрокомпьютинге и вероятностных вычислениях.

Жесткие вычисления – традиционные компьютерные вычисления (не мягкие).

Гибридные системы – системы, использующие более чем одну компьютерную технологию (в случае интеллектуальных систем – технологии искусственного интеллекта).

Возможны и другие классификации, например, выделяют системы общего назначения и специализированные системы (рисунок 5.4).

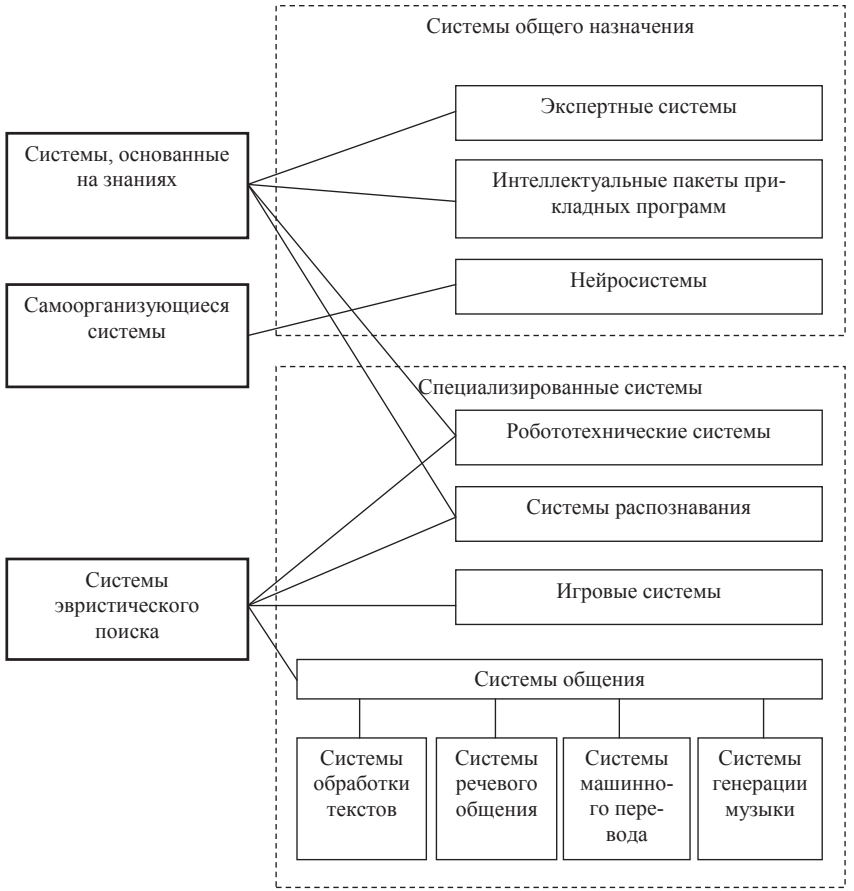


Рисунок 5.4 – Классификация интеллектуальных систем по назначению

Кроме того, эта схема отражает еще один вариант классификации по методам: системы, использующие методы представления знаний, самоорганизующиеся системы и системы, созданные с помощью эвристического программирования. Также в этой классификации системы генерации музыки отнесены к системам общения.

К интеллектуальным системам общего назначения относятся системы, которые не только исполняют заданные процедуры, но на основе метапроцедур поиска генерируют и исполняют процедуры решения новых конкретных задач.

Специализированные интеллектуальные системы выполняют решение фиксированного набора задач, предопределенного при проектировании системы.

Отсутствие четкой классификации также объясняется многообразием интеллектуальных задач и интеллектуальных методов, кроме того, искусственный интеллект – активно развивающаяся наука, в которой новые прикладные области осваиваются ежедневно.

5.5. ПОНЯТИЕ ИНТЕЛЛЕКТУАЛЬНОЙ ИНФОРМАЦИОННОЙ ТЕХНОЛОГИИ

Интеллектуальные информационные технологии (ИИТ) (англ. Intellectual information technology, IIT) – это информационные технологии, помогающие человеку ускорить анализ политической, экономической, социальной и технической ситуации, а также - синтез управленческих решений. При этом используемые методы не обязательно должны быть логически непротиворечивы или копировать процессы человеческого мышления.

Использование ИИТ на практике подразумевает учет специфики проблемной области, которая может характеризоваться следующим набором признаков:

- качество и оперативность принятия решений;
- нечеткость целей и институциональных границ;
- множественность субъектов, участвующих в решении проблемы;
- хаотичность, флюктуируемость и квантованность поведения среды;
- множественность взаимовлияющих друг на друга факторов;
- слабая формализуемость, уникальность, нестереотипность ситуаций;
- латентность, скрытость, неявность информации;
- девиантность реализации планов, значимость малых действий;
- парадоксальность логики решений и др.

ИИТ формируются при создании информационных систем и информационных технологий для повышения эффективности принятия решений в условиях, связанных с возникновением проблемных ситуаций. В этом случае любая жизненная или деловая ситуация – от выбора партнера по жизни до социального конфликта - описывается в виде некоторой познавательной модели (когнитивной схемы, архетипа, фрейма и пр.), которая впоследствии используется в качестве основания для построения и проведения моделирования, в том числе - компьютерного.

Гносеологический фундамент ИИТ наиболее явно видится в работах Канта, Гегеля, Гуссерля. Явную историю ИИТ удобно начать с середины XX века, когда появился термин «Искусственный интеллект». История ИИТ начинается с середины 1970-х годов и связывается с совместным практическим применением интеллектуальных информационных систем,

систем искусственного интеллекта, систем поддержки решений и информационных систем. Эта история связана также с развитием трех научных направлений: компьютерной философии, компьютерной психологии и продвинутой компьютерной науки (англ. Advanced computer science). С организационно-технологической стороны ИИТ дополняются прогрессом в создании: ситуационных центров, информационно-аналитических систем. Программно-математическое обеспечение составляют эволюционные вычисления и генетические алгоритмы, системы поддержки общения человека с компьютером на естественном языке, когнитивное моделирование, системы автоматического тематического рубрицирования документов, системы стратегического планирования, инструментарий технического и фундаментального анализа финансовых рынков, системы менеджмента качества, системы управления интеллектуальной собственностью и др.

С середины 1940-х вплоть до ранних 1970-х гг. создание ИИТ рассматривалось преимущественно в рамках логического решения задач. Этот период развития ИИТ характеризуется сравнительно большой определенностью и низкой динамичностью объекта управления. Вместе с тем уже в 1943 году появились «продукции Поста» и методы решения некорректных (обратных) задач на метризуемых пространствах, а в 1947 году для моделирования сложных экономических ситуаций активно начали использоваться методы причинного нелогического вывода, которые позже легли в основу методов системной динамики, немонотонных вычислений, когнитивного моделирования.

Создание центров управления полетами, организация штабных работ с применением средств визуализации и автоматизации, зарубежные публикации на тему создания специальных ситуационных центров вдохновили в 1970-е годы инженеров на создание ситуационных комнат для совершенствования управления крупными социальными и институциональными системами. В создании таких комнат и интеллектуальных технологий больше внимания стало придаваться средствам визуализации, диалоговым системам, помогающим использовать базы знаний и модели для решения плохо структурированных проблем.

В середине 1970-х годов на основе ИИТ в корпоративном мире начинают развиваться системы поддержки решений для эффективного управления ресурсами, осуществления контроллинга. Ряд замечательных практических идей и результатов, например, связанных с теорией нейронных сетей, многоагентных и активных систем, оптических и голографических процессоров, появилось именно в это время. Тот период можно отметить успехами в создании всеобъемлющих моделей ситуационного управления регионами в периоды кризисов. Его характеризует вера в практически неограниченные возможности искусственного интеллекта.

В середине 1980-х годов был отмечен крах иллюзий относительно неограниченных возможностей успешной формализации процессов мышления с помощью систем логической обработки естественного языка. Вместе с тем появились интеллектуальные технологии для ограниченной поддержки исследовательской и профессиональной деятельности лиц, принимающих решения. Практическое применение получили подходы, основанные на использовании достоверного и правдоподобного вывода, немонотонных логик и нечетких систем, лингвистических процессоров. Тогда же появилась явная потребность в оптических и квантовых вычислениях – для решения многомерных и слабо распараллеливающихся задач. Видимые успехи появились в сфере обработки текстов естественного языка, высококачественного поиска документов, слежения за динамичными объектами управления, решения задач распознавания образов, имитационного моделирования, статистической обработки данных, решения транспортных задач, построения нечетких контроллеров.

В конце 1980-х внимание разработчиков ИИТ все больше акцентируется на исследовании адаптивных свойств информационных систем, учитывающих умственную активность человека при осуществлении речевых актов, дискурса и принятии решений.

С начала 1990-х ИИТ все активней используются в стратегическом менеджменте, управлении ресурсами, реинжиниринге, создании ситуационных центров. Все более заметно внедряются интеллектуальные информационные технологии аналитической обработки больших массивов информации, технологии поддержки решений. В 1990-х годах в совокупности и взаимосвязи развиваются: экспертные системы реального времени, интеллектуальные агенты, активные системы, достоверный и правдоподобный вывод, эволюционные и квантовые вычисления, когнитивные модели, ситуационные центры и пр. Эксклюзивное место в развитии ИИТ с середины 1990-х заняла разработка необходимых условий конвергентности (сходимости) процессов управления, поиска информации и синтеза управленческих решений, направленных на обеспечение необходимых условий устойчивой сходимости этих процессов к намечаемым целям.

С 2000 года начал приобретать новое звучание процесс электронизации деятельности органов власти, бизнеса и населения. Концепция электронной демократии, предполагающая: осуществление гражданского контроля, проведение выборов и референдумов, поддержку процессов самоорганизации населения, обеспечение возможности участия населения в принятии государственных решений, расширение технологической возможности обмена мнениями – также предусматривает расширение возможностей интеллектуальных информационных технологий. Концепции электронной коммерции, включающие: маркетинг, управление корпоративными ресурсами, повышение качества продукции и услуг, расширение доступа к капиталу, электронные торги, развитие инноваций,

поддержку процессов самоорганизации бизнеса – не могла не активизировать работы по дальнейшему развитию систем поддержки решений с помощью ИИТ.

5.6. АРХИТЕКТУРА ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

5.6.1. Свойства знаний

Данные – это информация, полученная в результате наблюдений или измерений отдельных свойств (атрибутов), характеризующих объекты, процессы и явления предметной области.

Знания *(с точки зрения представления знаний в интеллектуальных системах)* – это связи и закономерности предметной области (принципы, модели, законы), полученные в результате практической деятельности и профессионального опыта, позволяющего специалистам ставить и решать задачи в данной области.

Знания от данных отличаются рядом **свойств**:

- внутренняя интерпретируемость;
- структурированность;
- связность;
- семантическая метрика;
- активность.

Внутренняя интерпретируемость. Данные, хранящиеся в памяти или на внешних носителях, лишены имен, таким образом, отсутствует возможность их однозначной идентификации системой. Данные может идентифицировать лишь программа, извлекающая их по определенному алгоритму. При переходе к знаниям в память вводится дополнительная информация (атрибуты: фамилия, год рождения, специальность, стаж). Атрибуты могут играть роль имен. По ним можно осуществлять поиск нужной информации.

Структурированность. Информационные единицы должны обладать гибкой структурой. Иначе говоря, должна существовать возможность произвольного установления между отдельными информационными единицами отношений типа «часть – целое», «род – вид» или «элемент – класс».

Связность. Между информационными единицами должна быть предусмотрена возможность установления связей различного типа. Семантика отношений может носить декларативный или процедурный характер. Например, две и более информационные единицы могут быть связаны отношением «одновременно», две информационные единицы – отношением «причина – следствие» или «быть рядом».

Семантическая метрика. На множестве информационных единиц в некоторых случаях полезно задавать отношение, характеризующее их ситуационную близость, то есть силу ассоциативной связи. Его можно было бы назвать отношением релевантности для информационных единиц. Оно дает возможность выделять в информационной базе некоторые

типовые ситуации (например, «покупка», «регулирование движения на перекрестке»). Отношение релевантности при работе с информационными единицами позволяет находить знания, близкие к уже найденным.

Активность. Все вычислительные процессы инициируются командами, а данные используются этими командами лишь в случае необходимости. Иначе говоря, данные пассивны, а команды активны.

Знания позволяют адаптироваться и действовать в реальной действительности. Существует огромное множество различных знаний, начиная от рецепта приготовления омлета до квантовой физики.

5.6.2. Классификация знаний

Знания можно классифицировать по нескольким критериям (рисунок 5.5)

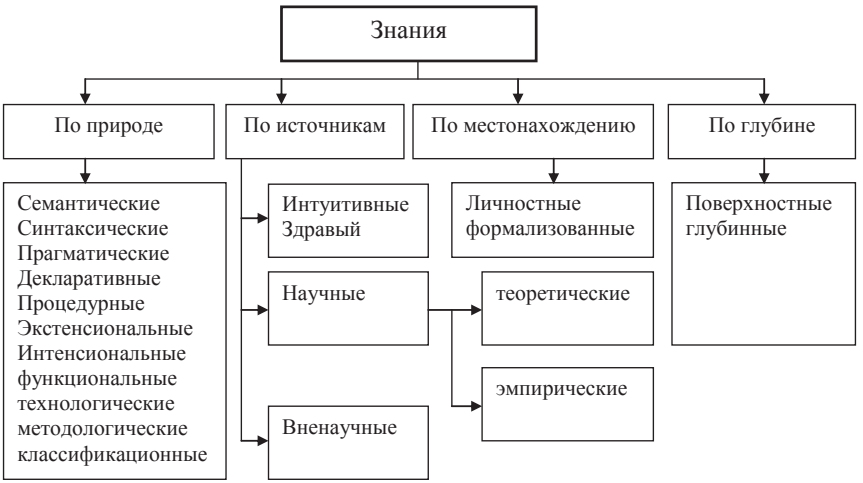


Рисунок 5.5 – Классификация знаний

Знание *синтаксического* типа характеризует синтаксическую структуру потока информации, которая не зависит от смысла и содержания используемых при этом понятий, то есть интеллектуальную систему не образует.

Семантическое знание рассматривается как структура, образующая текущий контекст. Оно содержит информацию, непосредственно связанную с текущими значениями и смыслом описываемых понятий, и предопределяет состояние связей данных в информационной базе.

Прагматическое знание предопределяет наиболее вероятные связи, описывающие данные с точки зрения решаемой задачи (обобщенный или

«объективный» контекст), например, с учетом действующих в данной задаче специфических критериев и соглашений.

Декларативные знания содержат в себе представление о структуре понятий. Эти знания приближены к данным, фактам. Например, высшее учебное заведение есть совокупность факультетов, а каждый факультет в свою очередь есть совокупность кафедр.

Процедурные знания имеют активную природу. Они определяют представления о средствах и путях получения новых знаний, проверке знаний. Это алгоритмы разного рода. С развитием информатики все большая часть знаний сосредотачивалась в структурах данных (таблицы, списки, абстрактные типы данных), то есть увеличивалась роль декларативных.

Существенными для понимания природы знаний являются способы определения понятий. Один из широко применяемых способов основан на идее интенсионала и экстенсионала.

Интенсионал понятия – это определение его через соотнесение с понятием более высокого уровня абстракции с указанием специфических свойств.

Экстенсионал понятия – это определение понятия через перечисление его конкретных примеров, то есть понятий более низкого уровня абстракции. Интенсионалы формируют знания об объектах, в то время как экстенсионалы объединяют данные.

Отсюда *интенсиональные* знания – это знания о предметной области, которые отражают факты, закономерности, свойства и характеристики, справедливые для любых ситуаций, которые могут возникнуть в этой предметной области.

Экстенсиональные знания – это знания о предметной области, отражающие факты, закономерности, свойства и характеристики, типичные для конкретных ситуаций или классов однотипных ситуаций, которые могут возникнуть в этой области.

Функциональные знания – это знания о выполняемых функциях отдельных предметов и о применении их в реальной действительности.

Технологические знания – специализированные знания, обеспечивающие поддержание технологических параметров производства; производственный опыт и навыки, используемые при решении повседневных производственных вопросов. Это может быть знание последовательности операций или знание технологической цепочки, позволяющие достигать поставленные цели в соответствии с принятой технологией.

Методологические знания – знания о методах преобразования действительности, научные знания о построении эффективной деятельности. К методологическим знаниям относят знание целей, форм и

направлений развития теории, методов и способов эффективного преобразования практики.

Классификационные знания применяются главным образом в науке, являются обобщенными, системными знаниями. Пример – система элементов Д. И. Менделеева.

Интуиция – это вид знания, специфика которого обусловлена способом его приобретения. Это знание, не нуждающееся в доказательстве и воспринимаемое как достоверное. По способу получения интуиция – это прямое усмотрение объективной связи вещей, не опирающееся на доказательство (интуиция есть усмотрение внутренним зрением; от лат. intueri – созерцать).

Под *здравым смыслом* понимают знания, позволяющие принимать правильные решения и делать правильные предположения, основываясь на логическом мышлении и накопленном опыте. В этом значении термин зачастую акцентирует внимание на способности человеческого разума противостоять предрассудкам, заблуждениям, мистификациям.

Научные знания в любом случае должны быть основанными на эмпирической или теоретической доказательной основе.

Теоретические знания – абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов. Теоретический уровень научного знания предполагает установление законов, дающих возможность идеализированного восприятия, описания и объяснения эмпирических ситуаций, то есть познания сущности явлений. Теоретические законы имеют более строгий, формальный характер по сравнению с эмпирическими. Термины описания теоретического знания относятся к идеализированным, абстрактным объектам. Подобные объекты невозможно подвергнуть непосредственной экспериментальной проверке.

Эмпирические знания получают в результате применения эмпирических методов познания: наблюдения, измерения, эксперимента. Это знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области. Эмпирические знания, как правило, констатируют качественные и количественные характеристики объектов и явлений. Эмпирические законы часто носят вероятностный характер и не являются строгими.

Вненаучные знания могут быть различными. *Паранормальные* знания – знания, несовместимые с имеющимся гносеологическим стандартом. Широкий класс *паранаучного* (пара от греч. около, при) знания включает в себя учения или размышления о феноменах, объяснение которых не является убедительным с точки зрения критериев научности. *Лженаучные* знания – сознательно эксплуатирующие домыслы и предрассудки. В качестве симптомов лженауки выделяют малограмотный пафос, принципиальную нетерпимость к опровергающим доводам, а также

претенциозность. Лженаучные знания сосуществуют с научными знаниями.

Личностные (неявные, скрытые) знания – это знания людей, полученные из практики и опыта.

Формализованные (явные) знания – знания, содержащиеся в документах, на компакт-дисках, в персональных компьютерах, в Интернете, в базах знаний, в экспертных системах. Формализованные знания объективизируются знаковыми средствами языка, охватывают те знания, о которых мы знаем, их можно записать, сообщить другим.

5.6.3. Базы знаний

Перечисленные ниже пять особенностей информационных единиц определяют ту грань, за которой данные превращаются в знания, а базы данных перерастают в **базы знаний**.

База знаний (БЗ) – основа любой интеллектуальной системы, где знания описаны на некотором языке представления знаний, приближенном к естественному. Сегодня знания приобрели чисто декларативную форму, то есть знаниями считаются предложения, записанные на языках представления знаний, приближенных к естественному языку и понятных неспециалистам.

Внутренняя интерпретируемость. Каждая информационная единица должна иметь уникальное имя, по которому ИС находит ее, а также отвечает на запросы, в которых это имя упомянуто. Когда данные, хранящиеся в памяти, были лишены имен, то отсутствовала возможность их идентификации системой. Данные могла идентифицировать лишь программа, извлекающая их из памяти по указанию программиста, написавшего программу. Что скрывается за тем или иным двоичным кодом машинного слова, системе было неизвестно.

Таблица 5.1 – Данные о сотрудниках предприятия

Фамилия	Год рождения	Специальность	Стаж, число лет
Попов	1965	Слесарь	5
Сидоров	1946	Токарь	20
Иванов	1925	Токарь	30
Петров	1937	Сантехник	25

Если, например, в память компьютера нужно было записать сведения о сотрудниках учреждения, представленные в таблице 5.1, то без внутренней интерпретации в память компьютера была бы занесена совокупность из четырех машинных слов, соответствующих строкам этой таблицы. При этом информация о том, какими группами двоичных разрядов в этих машинных словах закодированы сведения о специалистах, у системы отсутствуют. Они известны лишь программисту, который использует данные таблице 5.1 для решения возникающих у него задач.

При переходе к знаниям в память компьютера вводится информация о некоторой протоструктуре информационных единиц. В рассматриваемом примере она представляет собой специальное машинное слово, в котором указано, в каких разрядах хранятся сведения о фамилиях, годах рождения, специальностях и стажах. При этом должны быть заданы специальные словари, в которых перечислены имеющиеся в памяти системы фамилии, года рождения, специальности и продолжительности стажа. Все эти атрибуты могут играть роль имен для тех машинных слов, которые соответствуют строкам таблицы. По ним можно осуществлять поиск нужной информации. Каждая строка таблицы будет экземпляром протоструктуры. В настоящее время СУБД обеспечивают реализацию внутренней интерпретируемости всех информационных единиц, хранящихся в базе данных.

Структурированность. Информационные единицы должны обладать гибкой структурой. Для них должен выполняться "принцип матрешки", т.е. рекурсивная вложенность одних информационных единиц в другие. Каждая информационная единица может быть включена в состав любой другой, и из каждой информационной единицы можно выделить некоторые составляющие ее информационные единицы. Другими словами, должна существовать возможность произвольного установления между отдельными информационными единицами отношений типа "часть - целое", "род - вид" или "элемент - класс".

Связность. В информационной базе между информационными единицами должна быть предусмотрена возможность установления связей различного типа. Прежде всего эти связи могут характеризовать отношения между информационными единицами. Семантика отношений может носить декларативный или процедурный характер. Например, две или более информационные единицы могут быть связаны отношением "одновременно", две информационные единицы - отношением "причина - следствие" или отношением "быть рядом". Приведенные отношения характеризуют декларативные знания. Если между двумя информационными единицами установлено отношение "аргумент - функция", то оно характеризует процедурное знание, связанное с вычислением определенных функций. Далее будем различать отношения структуризации, функциональные отношения, каузальные отношения и семантические отношения. С помощью первых задаются иерархии

информационных единиц, вторые несут процедурную информацию, позволяющую находить (вычислять) одни информационные единицы через другие, третьи задают причинно-следственные связи, четвертые соответствуют всем остальным отношениям.

Между информационными единицами могут устанавливаться и иные связи, например, определяющие порядок выбора информационных единиц из памяти или указывающие на то, что две информационные единицы несовместимы друг с другом в одном описании.

Перечисленные три особенности знаний позволяют ввести общую модель представления знаний, которую можно назвать семантической сетью, представляющей собой иерархическую сеть, в вершинах которой находятся информационные единицы. Эти единицы снабжены индивидуальными именами. Дуги семантической сети соответствуют различным связям между информационными единицами.

Семантическая метрика. На множестве информационных единиц в некоторых случаях полезно задавать отношение, характеризующее ситуационную близость информационных единиц, т.е. силу ассоциативной связи между информационными единицами. Его можно было бы назвать отношением релевантности для информационных единиц. Такое отношение дает возможность выделять в информационной базе некоторые типовые ситуации. Отношение релевантности при работе с информационными единицами позволяет находить знания, близкие к уже найденным.

Активность. Все процессы, протекающие в ЭВМ, инициируются командами, а данные используются этими командами лишь в случае необходимости. Для ИС эта ситуация не приемлема. Как и у человека,

в ИС актуализации тех или иных действий способствуют знания, имеющиеся в системе. Таким образом, выполнение программ в ИС должно инициироваться текущим состоянием информационной базы. Появление в базе фактов или описаний событий, установление связей может стать источником активности системы.

Совокупность средств, обеспечивающих работу с знаниями, образует **Систему Управления Базой Знаний (СУБЗ).** В настоящее время не существует баз знаний, в которых в полной мере были бы реализованы внутренняя интерпретируемость, структуризация, связность, введена семантическая мера и обеспечена активность знаний.

5.6.4. Архитектура интеллектуальных систем

Архитектура интеллектуальных систем включает три комплекса вычислительных средств (рисунок 5.6).

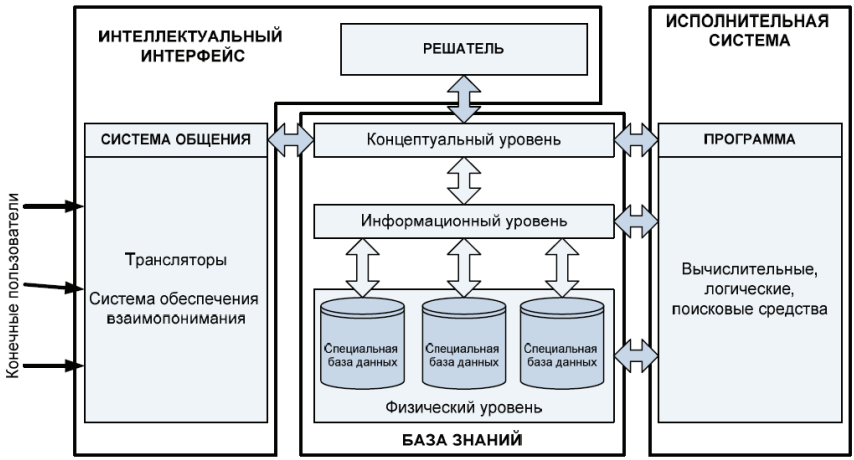


Рисунок 5.6 – Архитектура интеллектуальных систем

Первый комплекс представляет собой совокупность средств, выполняющих программы (исполнительную систему), спроектированных с позиций эффективного решения задач, имеет в ряде случаев проблемную ориентацию.

Второй комплекс представляет собой совокупность средств интеллектуального интерфейса, имеющих гибкую структуру, которая обеспечивает возможность адаптации в широком спектре интересов конечных пользователей.

Третьим комплексом средств, с помощью которых организуется взаимодействие первых двух, является база знаний, обеспечивающая использование вычислительными средствами первых двух комплексов целостной и независимой от обрабатывающих программ системы знаний о проблемной среде. Исполнительная система объединяет всю совокупность средств, обеспечивающих выполнение сформированной программы.

Интеллектуальный интерфейс – система программных и аппаратных средств, обеспечивающих для конечного пользователя использование компьютера для решения задач, которые возникают в среде его профессиональной деятельности либо без посредников, либо с незначительной их помощью.

БЗ занимает центральное положение по отношению к остальным компонентам вычислительной системы. В целом, через БЗ осуществляется интеграция средств вычислительной системы, участвующих в решении задач.

5.7. ПРИМЕНЕНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ И ТЕХНОЛОГИЙ В ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

5.7.1 Организация диалога между человеком и интеллектуальной системой

Диалоговые системы, основанные на распознавании рукописного текста

Рукописный ввод символов (рисунок 5.7) может по праву считаться одним из самых удобных способов набора текста наравне с оперированием виртуальной клавиатурой. Тот же голосовой набор можно применить далеко не всегда, а в случае с рукописным методом всё обстоит намного проще.

Рисунок 5.7 – Система распознавания рукописного ввода PenReader для планшетов, смартфонов и прочих мобильных устройств на платформе Android

Рассмотрим, в чем заключается различие между двумя формами представления одного и того же текста: рукописной и печатной. При этом могут исследоваться и сравниваться как сам процесс формирования текста, так и его результаты, т.е. уже сформированные тексты.

При исследовании уже сформированных текстов обнаруживается, что главное отличие рукописного текста от печатного состоит в значительно большей степени вариабельности начертаний одной и той же буквы разными людьми и одним и тем же человеком в различных состояниях, чем при воспроизведении тех же букв на различных пишущих машинках и принтерах.

Почерком будем называть систему индивидуальных особенностей начертания и динамики воспроизведения букв, слов и предложений вручную различными людьми или на различных устройствах печати.

В рукописной форме начертание букв является индивидуальным для каждого человека и зависит также от его состояния, хотя, конечно, в начертаниях каждой конкретной буквы всеми людьми безусловно есть и нечто общее, что и позволяет идентифицировать ее именно как данную букву при чтении.

К индивидуальным особенностям рукописного начертания букв отнесено 13 шкал с десятками градаций в каждой.

На современных компьютерах основным устройством ввода текстовой информации является клавиатура. Результат ввода текста в компьютер с точки зрения начертания букв, слов и предложений не имеет особых индивидуальных особенностей (если не считать частот использования различных шрифтов, кеглей, жирностей, подчеркиваний и других эффектов, изменяющих вид текста). Поэтому необходимо ввести понятие ***клавиатурного почерка****, под которым будем понимать систему индивидуальных особенностей начертаний и динамики воспроизведения букв, слов и предложений на клавиатуре.*

Таким образом, *любой текст содержит не только ту информацию, для передачи которой его собственно и создавали, но и информацию о самом авторе этого текста и о технических средствах и технологии его создания.*

Существует целая наука – ***"Психографология"***, которая ставит своей задачей получение максимального количества информации об авторах текстов на основе изучения индивидуальных особенностей их почерка.

В настоящее время в России действует институт графологии. На сайте этого института http://graphology.boom.ru можно познакомится с тем, что такое графология, с ее историей и задачами, которые она позволяет решать сегодня. Графологическое исследование имеет значительное преимущество перед простым тестированием или собеседованием, поскольку нет необходимости информировать человека, чей почерк подвергается изучению о производимых исследованиях.

Но текст представляет собой не просто совокупность букв, а сложную иерархическую структуру, в которой буквы образуют лишь фундамент пирамиды, а на более высоких ее уровнях находятся слова, предложения, и другие части текстов различных размеров, обладающие относительной целостностью и самостоятельностью (абзацы, параграфы, главы, части, книги).

Понятие почерка акцентирует внимание именно на начертании и динамике воспроизведения ***букв и слов***. При этом в понятие почерка не входят индивидуальные особенности текстов, обнаруживаемые на более высоких уровнях иерархической организации текстов, например: частоты употребления тех или иных слов и словосочетаний, средние длины

предложений и абзацев, и т.п. Но именно эти индивидуальные особенности текстов исследуются и используются при ***атрибуции*** анонимных и псевдонимных текстов (определении их вероятного авторства) и датировки.

Соответственно и текст может представлять для читателя интерес по крайней мере с трех точек зрения:

1. Как источник информации о том, о чем говорит автор, т.е. о предмете изложения.
2. Как источник информации о самом авторе.
3. Как источник информации о предмете изложения и об авторе. В этом смысле читать А.С. Пушкина в рукописи может быть значительно интереснее, чем взяв томик с полки. Это объясняется просто: в томике есть лишь сам ***результат*** работы поэта и выхолощена вся информация о ***процессе***, т.е. о самом поэте, содержащаяся в почерке, способе размещения текста на листе, порядке и динамике его формирования, различных вариантах и ассоциациях, возникавших в процессе создания произведения.

Таким образом, система, оснащенная интеллектуальным интерфейсом, может вести по-разному в зависимости от результатов идентификации пользователя, его профессионального уровня и текущего психофизиологического состояния.

Рассмотрим подробнее некоторые вопросы идентификации пользователей по клавиатурному почерку.

Проблемы идентификации и аутентификации пользователей компьютеров являются актуальными в связи с все большим распространением компьютерных преступлений. Использование для идентификации клавиатурного почерка является одним из направлений биометрических методов идентификации личности.

Подобные системы не обеспечивают такую же точность распознавания, как системы идентификации по отпечаткам пальцев или по рисунку радужной оболочки глаз, но имеют то преимущество, что система может быть полностью скрыта от пользователя, т. е. он может даже не подозревать о наличии такой системы контроля доступа.

Аутентификация – это проверка, действительно ли пользователь является тем, за кого себя выдает. При этом пользователь должен предварительно сообщить о себе идентификационную информацию: свое имя и пароль, соответствующий названному имени.

Идентификация – это установление его личности.

И идентификация, и аутентификация являются типичными задачами распознавания образов, которое может проводиться по заранее определенной или произвольной последовательности нажатий клавиш.

При вводе информации пользователь последовательно нажимает и отпускает клавиши, соответствующие вводимому тексту. При этом для каждой нажимаемой клавиши можно фиксировать моменты нажатия и отпускания.

На современных компьютерах на следующую клавишу можно нажимать до отпускания предыдущих, т.е. символ помещается в буфер клавиатуры только по нажатию клавиши, тогда как аппаратные прерывания от клавиатуры возникают и при нажатии, и при отпускании клавиши.

Основной характеристикой клавиатурного почерка следует считать временные интервалы между различными моментами ввода текста:

- между нажатиями клавиш;
- между отпусканиями клавиш;
- между нажатием и отпусканием одной клавиши;
- между отпусканием предыдущей и нажатием следующей клавиши.

Кроме того, могут учитываться производные от временных интервалов *вторичные показатели*, например, такие как скорость и ускорение ввода.

В литературе описано четыре математических подхода к решению задачи распознавания клавиатурного почерка пользователя:

- статистический;
- вероятностно-статистический;
- на базе теории распознавания образов и нечеткой логики;
- на основе нейросетевых алгоритмов.

В настоящее время возможна разработка интеллектуальных ***высоконадежных*** интерфейсов, обеспечивающих решение этих и ряда других задач идентификации и прогнозирования состояния оператора **в *режиме реального времени*** *непосредственно в процессе его работы с системой.*

При этом система в своей работе будет гибко учитывать текущее и прогнозируемое состояние оператора, что может проявляться в адаптации как алгоритмов работы, так и вида, и содержания интерфейса.

Эти работы дополняют возможности **заблаговременного** отбора операторов, обладающих свойствами, необходимыми для высоко ответственных работ в экстремальных ситуациях.

Диалоговые системы, основанные на распознавании речи

Распознавание речи *– процесс преобразования речевого сигнала в цифровую информацию (напр., текстовые данные). Обратной задачей является синтез речи.*

Можно выделить следующие области применения:

- голосовое управление;
- голосовой набор в различной технике (мобильники, компьютеры, и пр.);
- голосовой ввод текстовых сообщений в смартфонах и прочих мобильных компьютерах;
- голосовой поиск;
- голосовая почта.

Первое устройство для распознавания речи появилось в 1952 году, оно могло распознавать произнесённые человеком цифры. В 1964 году на

ярмарке компьютерных технологий в Нью-Йорке было представлено устройство IBM Shoebox.

Коммерческие программы по распознаванию речи появились в начале девяностых годов. Обычно их используют люди, которые из-за травмы руки не в состоянии набирать большое количество текста. Эти программы (например, Dragon NaturallySpeaking, VoiceNavigator, "Горыныч") переводят голос пользователя в текст, таким образом, разгружая его руки. Надёжность перевода у таких программ достаточно высока, и с годами она постепенно улучшается.

Увеличение вычислительных мощностей мобильных устройств позволило для них создать программы с функцией распознавания речи и так называемых виртуальных помощников (рисунок 5.8).

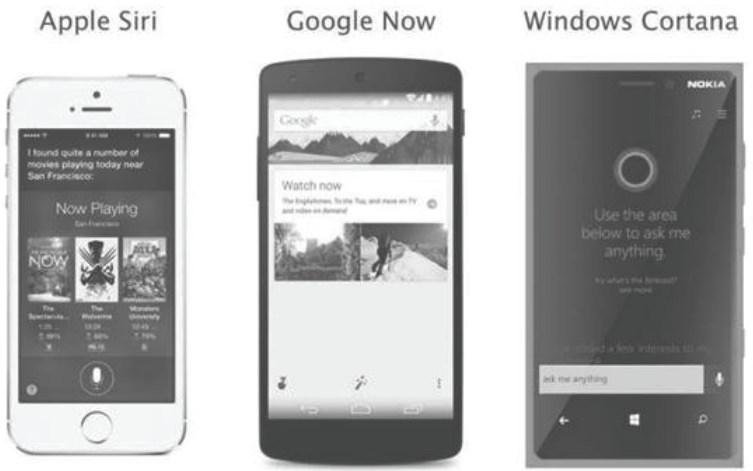


Рисунок 5.8 – Виртуальные помощники

Среди таких программ безусловно выделяются:

- ***Microsoft Cortana (Microsoft);***
- ***Google Now (Google);***
- ***Siri (Apple).***

Эти программы позволяет работать со многими приложениями при помощи голоса. Например, можно набрать нужный номер абонента, включить воспроизведение музыки в плеере, создать новый документ, произвести поиск нужного объекта в сети Интернет.

Siri (Apple) – ‘то персональный помощник, который работает по принципу вопрос-ответ и использует обработку естественной речи. Siri

задает вопросы и может быть полностью персонализирована. Есть возможность выбрать мужской или женский голос. Ориентируется в контексте вашей речи. Эволюционирует из iPhone в iPhone, становится все более интеллектуальной.

Google Now (Google) – голосовой помощник, впервые появившийся в 2012 году и получивший титул "Инновация года". Использует обработку естественного языка для ответов на вопросы, создания рекомендаций, открытия приложений, работы в сети и множества других функций. Подтягивает информацию из запросов в хроме, опираясь на режим дня, данных из календаря, местоположения, анализируя письма, персонализировать можно и вручную. Имеет интерфейс карточек. Доступен для скачивания и на iOS устройствах.

Microsoft Cortana (Microsoft) – виртуальный помощник с искусственным интеллектом. Появилась в общем доступе 14 апреля 2014 года. Cortana получила своё имя в честь персонажа серии компьютерных игр Halo, её голос также принадлежит героине игры – виртуальную помощницу озвучила актриса Джен Тейлор. До Cortana у Windows смартфонов была Loise. Ей можно дать доступ к вашим личным данным, таким как электронная почта, адресная книга, история поисков в сети и т. п. – все эти данные она будет использовать для упреждения ваших нужд. Cortana заменит стандартную поисковую систему и будет вызываться нажатием кнопки «Поиск».

Интеллектуальные речевые решения, позволяющие автоматически синтезировать и распознавать человеческую речь, являются следующей ступенью развития интерактивных голосовых систем (IVR). Использование интерактивного телефонного приложения в настоящее время не веяние моды, а жизненная необходимость. Снижение нагрузки на операторов контакт-центров и секретарей, сокращение расходов на оплату труда и повышение производительности систем обслуживания - вот только некоторые преимущества, доказывающие целесообразность подобных решений.

Прогресс, однако, не стоит на месте и в последнее время в телефонных интерактивных приложениях все чаще стали использоваться системы автоматического распознавания и синтеза речи. В этом случае общение с голосовым порталом становится более естественным, так как выбор в нем может быть осуществлен не только с помощью тонового набора, но и с помощью голосовых команд. При этом системы распознавания являются независимыми от дикторов, то есть распознают голос любого человека.

Следующим шагом технологий распознавания речи можно считать развитие так называемых Silent Speech Interfaces (SSI) (Интерфейсов Безмолвного Доступа). Эти системы обработки речи базируются на получении и обработке речевых сигналов на ранней стадии артикулирования. Данный этап развития распознавания речи вызван двумя существенными недостатками современных систем распознавания:

чрезмерная чувствительность к шумам, а также необходимость четкой и ясной речи при обращении к системе распознавания. Подход, основанный на SSI, заключается в том, чтобы использовать новые сенсоры, не подверженные влиянию шумов в качестве дополнения к обработанным акустическим сигналам.

На сегодняшний день существует два типа систем распознавания речи – работающие «на клиенте» (client-based) и по принципу «клиент-сервер» (client-server). При использовании клиент-серверной технологии речевая команда вводится на устройстве пользователя и через Интернет передается на удаленный сервер, где обрабатывается и возвращается на устройство в виде команды (Google Voice, Vlingo, пр.); ввиду большого количества пользователей сервера система распознавания получает большую базу для обучения. Первый вариант работает на иных математических алгоритмах и встречается редко (Speereo Software) - в этом случае команда вводится на устройстве пользователя и обрабатывается в нем же. Плюс обработки «на клиенте» в мобильности, независимости от наличия связи и работы удаленного оборудования. Так, система, работающая «на клиенте» кажется надежнее, но ограничивается, порой, мощностью устройства на стороне пользователя.

Сейчас применяется также технология SIND (без привязки к голосу конкретного человека).

Системы с биологической обратной связью

Системами с биологической обратной связью (БОС) *будем называть системы, поведение которых зависит от психофизиологического (биологического) состояния пользователя.* Это означает, что в состав систем с БОС в качестве подсистем входят информационно-измерительные системы и системы искусственного интеллекта.

Съем информации о состоянии пользователя осуществляется с помощью контактных и/или дистанционных датчиков в режиме реального времени с применением транспьютерных или обычных карт (плат) с аналого-цифровыми преобразователями (АЦП).

При этом информация может сниматься по большому количеству каналов – показателей (количество которых обычно кратно степеням двойки), подавляющее большинство которых обычно являются ***несознаваемыми*** *для пользователя.* Это является весьма существенным обстоятельством, т.к. означает, что системы БОС позволяют вывести на уровень сознания обычно ранее не осознаваемую информацию о состоянии своего организма, т.е. расширить область осознаваемого. А это значит, что у человека появляются условия, обеспечивающие возможность сознательного управления своими состояниями, ранее не управляемыми на сознательном уровне, что является важным эволюционным достижением технократической цивилизации.

Передача информации от блока съема информации к АЦП-карте может также осуществляться либо по проводной связи, либо дистанционно с использованием каналов инфракрасной или радиосвязи.

Приведем три примера применения подобных систем:

1). Мониторинг состояния сотрудников на конвейере с целью обеспечения высокого качества продукции.

2). Компьютерные тренажеры, основанные на БОС, для обучения больных с функциональными нарушениями управлению своим состоянием.

3). Компьютерные игры с БОС.

Известно, что одной из основных причин производственного брака является ухудшение состояния сотрудников. Но сотрудники не всегда могут вовремя заметить это ухудшение, т.к. самооценка (самочувствие) обычно запаздывает по времени за моментом объективного ухудшения состояния. Поэтому является актуальным своевременное обнаружение объективного ухудшения параметров и адекватное реагирование на него.

С помощью систем БОС это достигается тем, что:

1). Каждому сотруднику одевается на руку браслет с компактным устройством диагностики ряда параметров, например, таких, как:

- частота и наполнение пульса;
- кожно-гальваническая реакция;
- температура;
- давление;
- пототделение.

2). Это же устройство и периодически передает значения данных параметров на компьютер по радиоканалу.

3). Параметры от каждого сотрудника накапливаются в базе данных системы мониторинга на сервере, а также анализируются в режиме реального времени с учетом текущего состояния и динамики, в т. ч. вторичных (расчетных) показателей.

4). Когда параметры выходят за пределы коридора "нормы" или по их совокупности может быть поставлен диагноз, – сотрудник оперативно снимается с рабочего места и заменяется другим из резерва, а затем, при наличии показаний, направляется на лечение.

Некоторыми процессами в своем организме мы не можем управлять не потому, что у нас нет рычагов управления, а лишь потому, что мы их не знаем, не имеем навыков их использования и не знаем результатов их применения. Но ключевой проблемой, без решения которой невозможно управление, является отсутствие быстрого и надежного, адекватного по содержанию канала обратной связи.

Все эти проблемы снимаются системами БОС (рисунок 5.9):

- на экран компьютера в наглядной и легко интерпретируемой форме в режиме реального времени выводится информация о состоянии какой-либо подсистемы организма, например, об уровне pH (кислотности) в желудке;

• в качестве рычагов управления пациенту предлагается применить метод визуализации тех или иных образов, которые сообщаются врачом;

• когда пациент ярко зрительно представляет заданные образы, то при этом он обнаруживает, что кривая кислотности на экране начинает ползти вверх или вниз в прямом соответствии с тем, что именно он себе представляет.

Рисунок 5.9 – Система с биологической обратной связью

Через пару недель подобных тренировок, проводимых по 15-20 минут через день пациент приобретает такой уровень навыков управления ранее не осознаваемыми процессами в своем организме, которых Хатха-йоги добиваются за многие годы упорных тренировок под руководством профессиональных опытных и ответственных наставников (Гуру). Причем скоро пациент начинает понимать, когда необходимо повысить или понизить кислотность и без компьютера с системой БОС и может делать это прямо в той обстановке, в которой возникла такая необходимость. Столь высокая эффективность метода БОС объясняется высокой скоростью, наглядностью и адекватностью обратной связи, что является одним из основных факторов, влияющих на эффективность формирования навыков управления своим состоянием.

Имеется информация, что такими методами могут лечиться или облегчаться многие заболевания, вплоть до диабета, причем не только на стадии функциональных нарушений, но даже и при наличии органических изменений.

В последнее время появляется все больше компьютерных игр, включающих элементы БОС. При этом от психофизиологического

состояния игрока может зависеть, например, и развитие сценария, и точность прицеливания при использовании оптического прицела.

В этих играх часто создаются ситуации, в которых человеку нужно быстро принимать и реализовать решения, при этом цена ошибки, а значит и психическая напряженность, и волнение игрока, постоянно увеличиваются. Этим самым создается экстремальная ситуация, напряженность которой все больше возрастает. В этих условиях лучших результатов достигает тот, у кого "крепче нервы", кто лучше может управлять собой в экстремальных ситуациях.

Поэтому игры с элементами БОС можно считать своего рода тренажерами по формированию и совершенствованию навыков адекватного поведения в экстремальных ситуациях.

Здесь необходимо отметить один очень существенный момент. В обычной реальности развитие событий зависит не непосредственно от нашего психофизиологического состояния, а лишь от того, как оно проявляется в наших ***действиях***. В случае же виртуальной реальности развитие сценария игры может зависеть ***непосредственно*** от состояния игрока. Таким образом, *в виртуальной реальности само сознательное (произвольное) или несознательное (непроизвольное) изменение нашего состояния по сути дела является действием.* Аналогичная ситуация в обычной реальности может иметь место при высших формах сознания и проявлении сверхспособностей.

Системы с семантическим резонансом. Компьютерные Ψ-технологии и интеллектуальный подсознательный интерфейс

Системами с семантическим резонансом (рисунок 5.10) будем называть системы, поведение которых зависит от состояния сознания пользователя и его психологической реакции на смысловые стимулы. Это означает, что в состав систем с семантическим резонансом, также как и систем с БОС, в качестве подсистем входят информационно-измерительные системы и системы искусственного интеллекта, аналогично осуществляется и съём информации о состоянии пользователя.

Рисунок 5.10 – Психоэмоциональный комплекс БОС

Различие между системами с БОС и с семантическим резонансом состоит в том, что в первом случае набор снимаемых параметров и методы их математической обработки определяются необходимостью идентификации биологического состояния пользователя, тогда как во втором – его реакции на смысловые стимулы (раздражители).

В частности, имеется возможность по наличию в электроэнцефалограмме так называемых вызванных потенциалов установить реакцию человека на стимул: заинтересовался он или нет.

Здесь принципиально важно, что *вызванные потенциалы после предъявления стимула по времени возникают гораздо раньше, чем его осознание.*

Из этого следует ряд важных выводов:

1. Если *это осознание не наступает по каким-либо причинам, то вызванные потенциалы все равно с очень высокой достоверностью позволяют прогнозировать ту реакцию, которая была бы у человека, если бы информация о стимуле проникла в его сознание* (причинами, по которым зрительный образ стимула может не успеть сформироваться и проникнуть в сознание пользователя, могут быть, например, его очень сильную зашумленность, фрагментарность или слишком короткое время его предъявления).

2. *Реакция на стимул на уровне вызванных потенциалов не подвергается критическому анализу и корректировке на уровне сознания, т.е. является гораздо более "искренней" и "откровенной", адекватной и достоверной, чем сознательные ответы на опросник с тем же самым стимульным материалом* (сознательные ответы зависят от мотивации, коньюктуры и массы других обстоятельств).

3. Для получения информации о подсознательной реакции пользователя на стимульный материал он может предъявляться в значительно более высоком темпе, чем при сознательном тестировании.

4. При подсознательном тестировании пользователь может даже не знать о том, что оно проводится.

Все это в совокупности означает, что системы с семантическим резонансом позволяют получить и вывести на уровень сознания обычно ранее не осознаваемую адекватную информацию о состоянии своего сознания, систем мотивации, целеполагания, ценностей и т.д., т.е. расширить область осознаваемого. Это позволяет создать качественно более благоприятные условия для управления состоянием сознания, чем ранее, что является важным эволюционным достижением технократической цивилизации.

Системы с семантическим резонансом могут эффективно использоваться в ряде направлений:

- психологическое и профессиональное тестирование, подбор персонала, в т.ч. для действий в специальных условиях и в измененных формах сознания;
- модификация сознания, систем мотиваций, целеполагания, ценностей и др. (компьютерное нейролингвистическое программирование: "компьютерные НЛП-технологии");
- компьютерные игры с системами семантической обратной связи.

Системы виртуальной реальности. Эффекты присутствия, деперсонализации и модификация сознания пользователя

Виртуальная реальность (ВР) – модельная трехмерная (3D) окружающая среда, создаваемая компьютерными средствами и реалистично реагирующая на взаимодействие с пользователями (рисунок 5.11).

Рисунок 5.11 – Система виртуальной реальности для управления флотом

Технической базой систем виртуальной реальности являются современные мощные персональные компьютеры и программное обеспечение высококачественной трехмерной визуализации и анимации. В качестве устройств ввода-вывода информации в системах ВР применяются виртуальные шлемы с дисплеями (HMD), в частности шлемы со стереоскопическими очками, и устройства 3D-ввода, например, мышь с пространственно управляемым курсором или "цифровые перчатки", которые обеспечивают тактильную обратную связь с пользователем.

Совершенствование систем виртуальной реальности приводит ко все большей изоляции пользователя от обычной реальности, т.к. все больше каналов взаимодействия пользователя с окружающей средой замыкаются не на обычную, а на виртуальную среду – виртуальную реальность, которая, при этом, становится все более и более функционально-замкнутой и самодостаточной.

Создание систем ВР является закономерным следствием процесса совершенствования компьютерных систем отображения информации и интерфейса управления.

При обычной работе на компьютере монитор занимает не более 20% поля зрения пользователя. Системы ВР перекрывают все поле зрения.

Обычные мониторы не являются стереоскопическими, т.е. не создают объемного изображения. Правда, в последнее время появились разработки, которые, позволяют преодолеть это ограничение (достаточно сделать поиск

в yandex.ru по запросу "Стереоскопический монитор"). Системы ВР изначально были стереоскопическими. Звуковое сопровождение, в том числе со стерео и квадро-звуком, сегодня уже стали стандартом. В системах ВР человек не слышит ничего, кроме звуков этой виртуальной реальности.

В некоторых моделях систем виртуальной реальности пользователи имеют возможность восприятия изменяющейся перспективы и видят объекты с разных точек наблюдения, как если бы они сами находились и перемещались внутри модели.

Если пользователь располагает более развитыми ***(погруженными)*** устройствами ввода, например, такими, как цифровые перчатки и виртуальные шлемы, то модель может даже надлежащим образом реагировать на такие ***действия*** пользователя, как поворот головы или движение глаз.

Необходимо отметить, что в настоящее время системы виртуальной реальности развиваются очень быстрыми темпами и явно выражена тенденция проникновения технологий виртуальной реальности в стандартные компьютерные технологии широкого применения.

Развитие этих и других подобных средств привело к появлению качественно новых эффектов, которые ранее не наблюдались или наблюдались в очень малой степени:

- эффект присутствия пользователя в виртуальной реальности;
- эффект деперсонализации и модификации самосознания и сознания пользователя в виртуальной реальности.

Эффект присутствия *– это создаваемая для пользователя иллюзия его присутствия в смоделированной компьютером среде, при этом создается полное впечатление "присутствия" в виртуальной среде, очень сходное с ощущением присутствия в обычном "реальном" мире.*

При этом виртуальная среда начинает осознаваться как реальная, а о реальной среде пользователь на время как бы совершенно или почти полностью "забывает". При этом технические особенности интерфейса также вытесняются из сознания, т.е. мы не замечаем этот интерфейс примерно так же, как собственное физическое тело или глаза, когда смотрим на захватывающий сюжет. Таким образом, реальная среда ***замещается*** виртуальной средой.

Исследования показывают, что для возникновения и силы эффекта присутствия определяющую роль играет реалистичность движения различных объектов в виртуальной реальности, а также убедительность реагирования объектов виртуальной реальности при ***взаимодействии*** с ними виртуального тела пользователя или других виртуальных объектов. В то же время, как это ни странно, естественность вида объектов виртуальной среды играет сравнительно меньшую роль.

Системы виртуальной реальности уже в настоящее время широко применяется во многих сферах жизни.

Одними из первых технологии виртуальной реальности были применены НАСА США для тренировки пилотов космических челноков и военных самолетов, при отработке приемов посадки, дозаправки в воздухе и т.п.

Самолет-невидимка "Стелс" вообще управляется пилотом, практически находящемся в виртуальной реальности.

Из виртуальной реальности человек управляет роботом, выполняющим опасную или тонкую работу.

Технология Motion Capture, позволяет дистанционно "снять" движения с человека и присвоить их его трехмерной модели, что широко применяется для создания компьютерных игр и анимации рисованных персонажей в фильмах.

Особенно эффективно применение виртуальной реальности в рекламе, особенно в Интернет-рекламе на стадии информирования и убеждения.

С использованием виртуальной реальности можно показывать различные помещения, например, совершить виртуальную экскурсию по музею, учебному заведению, дому, коттеджу или местности (прогулка по Парижу от туристической фирмы).

Во всех этих приложениях важно, что в отличие от трехмерной графики, виртуальная реальность обеспечивает *эффект присутствия и* ***личного участия*** *пользователя в наблюдаемых им событиях.*

Сегодня уже для всех вполне очевидно, что виртуальная реальность может с успехом использоваться для развлечений, ведь *она помогает представить себя в другой* ***роли*** *и* ***в другом обличии***. Однако в действительности этот эффект связан с модификацией "Образа Я", т.е. сознания и самосознания пользователя. Это значит, что последствия этого в действительности значительно серьезнее, чем обычно представляют, и далеко выходит за рамки собственно развлечений.

Как показано автором в ряде работ, приведенных на сайте http://Lc.kubagro.ru, форма сознания и самосознания человека определяются тем, как он осознает себя и окружающее, т.е. тем:

- что он осознает, как объективное, субъективное и несуществующее;
- с чем он отождествляет себя и что осознает, как объекты окружающий среды.

Очевидно, что разработчики новейших компьютерных технологий совершенно неожиданно вторглись в абсолютно новую для себя сферу исследования ***измененных форм сознания***, и далеко идущие системные последствия этого ими, как и вообще научным сообществом, пока еще очень мало осознаны.

Еще в 1079-1981 годах автором и Л.А. Бакурадзе были оформлены заявки на изобретение компьютерной системы, выполняющей все трудовые функции физического тела, обеспечивающую управление с использованием дистанционного мысленного воздействия, т.е.

микротелекинеза. Телекинез представляет собой управление физическими объектами путем воздействия на них непосредственно с высших планов без использования физического тела, т.е. *тем же способом, с помощью которого любой человек, осознает он это или нет, управляет своим физическим телом.* Были предложены технические и программные решения и инженерно – психологические методики. Система предлагалась адаптивной, т.е. автоматически настраивающейся на индивидуальные особенности, "почерк" оператора и его состояние сознания, с плавным переключением на дистанционные каналы при повышении их надежности (которая измерялась автоматически) и могла одновременно с выполнением основной работы выступать в качестве тренажера. Человек, начиная работу с системой в обычной форме сознания с использованием традиционных каналов (интерфейса), имея мгновенную адекватную по форме и содержанию ***обратную связь*** об эффективности своего телекинетического воздействия, должен быстро переходить в форму сознания, оптимальную для использования телекинеза в качестве управляющего воздействия.

С учетом вышесказанного, предлагается следующее определение виртуальной реальности.

Система ВР – это система, обеспечивающая:

1). ***Генерацию полиперцептивной модели реальности*** в соответствии с математической моделью этой реальности, реализованной в программной системе.

2). ***Погружение пользователя в модель реальности*** путем подачи на все или основные его перцептивные каналы – органы восприятия, программно-управляемых по величине и содержанию воздействий: зрительного, слухового, тактильного, термического, вкусового и обонятельного и других.

3). ***Управление системой*** путем использования виртуального ***"образа Я" пользователя*** и виртуальных ***органов управления*** системой (интерфейса), на которые он воздействует, представляющие собой ***зависящую от пользователя часть*** модели реальности.

4). ***Реалистичную реакцию*** моделируемой реальности на виртуальное воздействие и управление со стороны пользователя.

5). Разрыв отождествления пользователя со своим "Образом Я" из обычной реальности ***(деперсонализация)***, и отождествление себя с "виртуальным образом Я", генерируемым системой виртуальной реальности ***(модификация сознания и самосознания пользователя).***

6). ***Эффект присутствия*** пользователя в моделируемой реальности в своем "виртуальном образе Я", т.е. эффект *личного участия пользователя в наблюдаемых виртуальных событиях.*

7). ***Положительные результаты применения критериев реальности,*** т.е. функциональную замкнутость и самодостаточность виртуальной реальности, вследствие чего *никакими действиями внутри виртуальной реальности, осуществляемыми над ее объектами, в т.ч. объектами*

виртуального интерфейса, с помощью своего виртуального тела, невозможно установить, "истинная" эта реальность или виртуальная.

В этой связи вспоминается ставший уже классическим первый фильм "Матрица", в котором Морфей, обращаясь к Нео, произносит свою знаменитую фразу: *"Сейчас я покажу тебе, как выглядит* ***окончательная истинная*** *реальность".* Эта фраза сразу вызвала у меня массу ассоциаций и вопросов, в частности:

1. А каковы критерии реальности?
2. А вдруг и эта реальность, которую Морфей назвал окончательной, истинной, в действительности является не более, чем симулятором следующего иерархического уровня, так сказать более фундаментальным симулятором?

Здесь возникает сложный мировоззренческий вопрос о том, возможно ли ***хотя бы в принципе*** находясь в виртуальной реальности, не выходя за ее пределы установить, что ты находишься именно в виртуальной, а не истинной реальности, или это возможно сделать только *задним числом*, после выхода из виртуальной реальности и перехода в истинную реальность?

Итак, каковы же критерии реальности?

По нашему мнению, прежде всего это ***самосогласованность*** реальности, т.е. получение одной и той же информации качественно различными способами и по различным каналом связи (принцип наблюдаемости):

- согласованность реальности самой с собой ***во времени***;
- согласованность и взаимное подтверждение информации от различных органов восприятия, которые обычно реагируют на различные формы материи и часто являются парными (зрение, слух, обоняние) и расположенными **в *различных точках пространства.***

Например, мы не только что-то видим, но и слышим, и осязаем, и можем попробовать его на вкус и ощутить запах и все эти восприятия ОТ ***РАЗЛИЧНЫХ*** *ОРГАНОВ ЧУВСТВ* ***соответствуют*** *друг другу и означают, что перед нами некий определенный объект, а не галлюцинация или визуализация. Согласованная и взаимно подтверждающая информация с различных органов чувств, в соответствии с принципом наблюдаемости, также может рассматриваться как повышающая достоверность и адекватность восприятия.*

В современных компьютерных играх мы не только видим довольно качественную визуализацию, но и соответствующее реалистичное звуковое сопровождение. А в системах виртуальной реальности – визуализация стереоскопическая (то, что мы видим РАЗНЫМИ глазами как бы с разных точек в ПРОСТРАНСТВЕ, также взаимно подтверждается), а также появляется тактильный канал с обратной связью, который позволяет ощутить даже твердость, вес и температуру моделируемого в виртуальной реальности объекта. Все это вместе уже

создает на столько высокую степень реалистичности, что может возникнуть эффект присутствия в виртуальной реальности, деперсонализация и отождествление с измененным образом Я, моделируемым в виртуальной реальности (переход в измененную форму сознания).

Представим, что эти сформулированные критерии реальности не выполняются, т.е. нарушается ее самосогласованность. По-видимому, как своего рода "сбои" и различные ***"нарушения физических законов"*** и несогласованности в виртуальной реальности:

- "зацикливание" событий, как на заезженной пластинке, т.е. их многократное повторное осуществление без каких-либо изменений (пример: повторный проход черной кошки, с характерной остановкой и поворотом головы, в дверном проеме в "Матрице");
- прохождение сквозь стены;
- полеты и очень длинные прыжки, а также телепортация в своем "реальном" теле;
- действия в другом темпе времени, т.е. эффект замедления внешнего времени, соответствующий аналогичному ускорению внутреннего времени;
- действия в другом масштабе пространства, "увеличение" и "уменьшение" размеров, наблюдение мега и микроструктуры материи;
- видение сквозь стены, видение на больших расстояниях (в т.ч. с увеличением "как в телескоп"), видение прошлого и будущего;
- телекинез, пирокинез, психосинтез, левитация и т.п.;
- одновременное нахождение в нескольких местах.

Нетрудно заметить, что все эти проявления весьма напоминают так называемые *"паранормальные явления"*, которые традиционно связывают с *сверхвозможностями* человека, т.е. с его возможностями при высших формах сознания.

Эти явления хотя и редко, но все же наблюдаются в нашем мире, что может указывать на то, что наша "истинная реальность" в определенной мере возможно является виртуальной, по крайней мере в большей степени, чем ранее предполагалось.

Вспомним известные в физике принципы относительности Галилея и Эйнштейна:

1. Никакими экспериментами внутри замкнутой системы невозможно отличить состояние покоя от состояния равномерного и прямолинейного движения (Галилей). Следовательно, покоящаяся система отсчета ***физически эквивалентна*** системе отсчета, движущейся равномерно и прямолинейно под действием сил инерции.

2. Никакими экспериментами внутри ограниченной по размерам замкнутой системы невозможно установить, движется она под действием сил гравитации или по инерции (Эйнштейн). Следовательно, система

отсчета, движущаяся в поле сил тяготения ***физически эквивалентна*** системе отсчета, движущейся под действием сил инерции.

Легко заметить, что формулировка 7-го пункта в определении системы виртуальной реальности весьма сходна с формулировками принципов относительности Галилея и Эйнштейна: *никакими действиями внутри виртуальной реальности, осуществляемыми над ее объектами, в т.ч. объектами виртуального интерфейса, с помощью своего виртуального тела, невозможно установить, "истинная" эта реальность или виртуальная.*

Следовательно, **виртуальная система отсчета, локализованная в полнофункциональной виртуальной реальности *полностью физически эквивалентна* физической системе отсчета, локализованной в "истинной реальности".**

Учитывая эту аналогию, принцип, предложенный автором, назовем принципом относительности или *принципом эквивалентности виртуальной и истинной реальности.*

Системы с дистанционным телекинетическим интерфейсом

В 1981 году Л.А. Бакурадзе и Е.В. Луценко были оформлены заявки на изобретение компьютерной системы, выполняющей все трудовые функции физического тела, обеспечивающую управление с использованием дистанционного мысленного воздействия, т.е. микротелекинеза.

Телекинез представляет собой управление физическими объектами путем воздействия на них непосредственно с высших планов без использования физического тела, т.е. тем же способом, с помощью которого любой человек, осознает он это или нет, управляет своим физическим телом. К подобным системам могут быть отнесены системы с жестовым управлением (рисунок 5.12).

Рисунок 5.12 – Жестовое управление бортовой навигационной системой на автомобиле BMW

К настоящему времени предложены различные технические и программные решения и инженерно-психологические методики. Системы являются адаптивными, т.е. автоматически настраиваются на индивидуальные особенности, "почерк" оператора и его состояние сознания, ***с плавным переключением на дистанционные каналы при повышении их надежности*** (которая измерялась автоматически) и могла одновременно с выполнением основной работы выступать в качестве ***тренажера для овладения высшими формами сознания.***

Человек, начиная работу с системой в обычной форме сознания с использованием традиционных каналов (интерфейса), имея мгновенную адекватную по форме и содержанию обратную связь об эффективности своего телекинетического воздействия, должен быстро переходить в одну из высших форм сознания, оптимальную для использования телекинеза в качестве управляющего воздействия.

5.8. РАЗРАБОТКА СЛОЖНЫХ ПРЕДМЕТНО-ОРИЕНТИРОВАННЫХ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ НА ОСНОВЕ ЕСТЕСТВЕННО-ЯЗЫКОВОГО ИНТЕРФЕЙСА

Естественно-языковой интерфейс (ЕЯИ) – разновидность пользовательского интерфейса, который принимает запросы на естественном языке, а также, возможно, использует ЕЯ и для вывода информации (реакции системы на запрос пользователя).

На рисунке 5.13 приведены основные составляющие ЕЯИ и взаимосвязи между ними.

Рисунок 5.13 – Основные составляющие ЕЯ-интерфейсов и их взаимосвязи

5.8.1. Сравнительный анализ ЕЯ-интерфейсов и традиционных интерфейсов к структурированным источникам данных

В противоположность ЕЯ-интерфейсам, нетрадиционным с точки зрения распространенности, существуют другие виды пользовательских интерфейсов к структурированным источникам данных (СИД), которые можно назвать традиционными. Среди них:

- интерфейсы с формальным языком запросов
- интерфейсы с графическим построением запросов
- интерфейсы, основанные на заполнении форм запросов

Предполагается, что рассматриваемые здесь виды интерфейсов по своему предназначению ограничены только получением информации из базы данных, это предположение сделано в силу ограничения на ЕЯ-

интерфейсы областью запросов, поскольку область занесения данных и их модификации с помощью естественно-языковых оболочек является отдельной большой темой для рассмотрения.

В интерфейсах *с формальным языком запросов* пользователь, для того, чтобы правильно задать запрос, должен, во-первых, знать синтаксис языка запросов (например, SQL), а во-вторых, представлять устройство конкретного структурированного источника данных (например, реляционную схему базы данных). При работе с этим типом интерфейсов пользователь должен обладать достаточно высокой квалификацией. Опыт показывает, что такой необходимой квалификацией обладают лишь специалисты, проектирующие и создающие информационные системы, и сам термин "пользователь" с учетом современных тенденций здесь не совсем адекватен. Очевидно, ЕЯ-интерфейсы обладают большей гибкостью - один и тот же запрос обычно можно формулировать различными способами. Что немаловажно, ЕЯ-интерфейсы, как правило, обладают системой понятий – описанием предметной области, которая находится выше логического уровня хранения данных. Это позволяет абстрагироваться от деталей устройства той или иной базы данных, как на уровне структуры, так и на уровне содержимого.

Средства *графического построения запросов*, которыми снабжаются многие "настольные" СУБД (например, MS Access, MS FoxPro), безусловно, обладают большим удобством – по крайней мере пользователь не должен держать в голове названия таблиц, полей и конструкции языка. Однако для работы с такими средствами необходим опыт и представление некоторых понятий, относящихся скорее к математике (например, термин связывания таблиц в реляционной алгебре), а не к предметной области, и иногда достаточно утомительные действия по заполнению форм. Так, в базе данных Microsoft Access для того, чтобы сформулировать выражение AVG(PERSONNEL.SALARY), эквивалентный ЕЯ-фразе "средняя зарплата", требуется около 15 нажатий мышью. Неподготовленный пользователь обычно пасует перед системами, требующими сложных действий. Как и в случае интерфейсов с формальным языком, пользователь должен представлять устройство базы данных. По сути, эти средства позволяют графически создавать формальные запросы, и не случайно они обычно позволяют редактировать пользователям полученный формальный запрос.

Интерфейсы, основанные на *заполнении форм* запросов, являются более дружественными, по сравнению с формальными языками. Сама метафора формы и ее заполнения подразумевает, что пользователь сразу видит набор критериев и параметров поиска, а иногда и список возможных значений полей формы, это сводит к минимуму ошибки при вводе запроса. От предыдущего метода построения пользовательских интерфейсов данный отличается тем, что как правило, все необходимые запросы уже написаны разработчиком интерфейса, и пользователь, чтобы получить ответ, должен

просто вставить недостающие значения. Отличие заключается также в том, что задавая значения формы, пользователь обычно не выбирает, какие атрибуты данного класса объектов будут в результате, а сам список доступных классов (в реляционной базе - таблиц) ограничен множеством построенных форм. Так работают многие современные коммерческие приложения, работающие с базами данных - пользователю информация в системе доступна в виде нескольких типовых "срезов" информационного пространства. К недостатку систем, основанных на таком подходе, как и в предыдущем, также следует отнести необходимость наличия у пользователя опыта работы с подобными системами, а также необходимость создания форм, что требует дополнительных усилий программиста для создания интерфейса.

Преимущества ЕЯИ достаточно очевидны:

- минимальная предварительная подготовка пользователя. Естественный язык является наиболее привычным и удобным средством коммуникации, и именно в силу этого с ростом эффективности ЕЯ-систем, он, безусловно, будет вытеснять другие виды интерфейсов к СИД, традиционные в данный момент.
- простота задания запросов на ЕЯ. Во многих случаях запрос на ЕЯ получается гораздо короче языка на формальном языке, поскольку ЕЯ-представление более емко, ведь в самой структуре языка содержится понятийная база, которую отражает структура источника данных. Зачастую сложность этой структуры отражается на сложности запроса на формальном языке.
- большая скорость создания произвольного запроса (отсутствует стадия формального задания запроса). Как правило, пользователь сразу может сформулировать

корректное ЕЯ-представление запроса, поскольку такое представление является самым естественным для человека, тогда как построение запроса на формальном языке, даже с помощью вспомогательных средств, таит множество ошибок, зачастую исправить которые можно, только проанализировав результат запроса.

- более высокий уровень модели предметной области. Традиционные интерфейсы обычно не обладают моделью предметной области как таковой, и в лучшем случае скрывают от пользователя искусственные средства и особенности структуры, присущие конкретному типу СИД (такие, как связи по идентификаторам между таблицами в реляционных базах данных или синтаксис XML).

Более подробно рассмотрим недостатки ЕЯ-интерфейса по сравнению с другими типами интерфейсов.

- неоднозначность естественного языка приводит к множественности смыслов. Специфика естественного языка такова, что часто запрос может иметь несколько смыслов, о которых пользователь в момент задания запроса не предполагает. Формальные же языки лишены проблемы

неоднозначности. Это свойство ЕЯ приводит к усложнению ЕЯИ и методов анализа, в противном случае ЕЯИ получается слишком примитивным для реального использования.

- недостаточная надежность анализаторов ЕЯ-запросов может привести к неправильному пониманию. Современные ЕЯ-интерфейсы далеко не всегда позволяют диагностировать причины неудач понимания. Причины этих неудач могут быть как в лингвистической сфере, так и в концептуальной. Например, запрос к кадровой базе данных "Кто получает больше Иванова" может привести к непониманию, если ЕЯ-интерфейс не умеет распознавать вложенные запросы (а в данном случае надо сначала получить значение зарплаты Иванова, а затем сравнить с ней зарплату сотрудников). Это случай лингвистической проблемы. Второй пример - "Как зовут жен сотрудников?" может привести к неудаче понимания, если ЕЯ-интерфейс не поймет, что имя супруга/супруги - это реальный атрибут сотрудника, но отсутствующий в данной базе данных. В данном случае налицо будет концептуальная проблема - ЕЯ-интерфейс должен уметь отличать реальную предметную область, которую имеет в виду пользователь, задавая ЕЯ-запрос, от той ее части или трансформации, которая представлена в данном источнике данных.
- пользователь может иметь завышенные или заниженные ожидания от ЕЯ-интерфейса. Сравнительный анализ типов пользовательских интерфейсов (основанных на формах, с формальным языком запросов, графические) показывает, что в целях построения ЕЯ-интерфейсов превалирует желание максимально приблизить интерфейс к потребностям неподготовленного пользователя. Это несколько поднимает планку требований к дружественности и надежности ЕЯ-интерфейсов, поскольку пользователь, впервые столкнувшись с системой, понимающей естественный язык, слабо представляет, насколько интеллектуальна система. При этом ожидания к степени понимания ЕЯ может отличаться от реальных способностей системы в обе стороны
- пользователь может спрашивать систему о том, чего она "не знает", а может "по привычке" использовать простейшие шаблонные формулировки запросов. В других же типах интерфейсов к СИД рамки того, что пользователь может делать с помощью интерфейса, видны, как правило, сразу.

Поскольку характеристики ЕЯИ и систем для их построения могут существенно различаться, то преимущества и недостатки ЕЯИ по сравнению с другими типами интерфейсов к СИД можно выделить довольно схематично, только на качественном уровне. Для сравнения подходов к построению ЕЯИ введем метрику показателей, характеризующих качество ЕЯИ к структурированным источникам данных.

5.8.2. Критерии качества ЕЯ-интерфейсов

Для сравнительного анализа подходов к созданию ЕЯ-интерфейсов рассмотрим такую качественную интегральную характеристику, как надежность. Под *надежностью* здесь понимается способность ЕЯ-интерфейса правильно понимать намерения пользователя по получению информации из источника, при условии, что пользователь корректно выразил потребности в виде ЕЯ-запроса. Надежность отражает правильность принципов, лежащих в методе ЕЯ-анализа, а также правильность (корректность) построения ЕЯИ к конкретному СИД.

Любой ЕЯИ имеет некоторое пространство правильно понимаемых запросов. Чем больше это пространство, тем большей *полнотой* обладает ЕЯИ. Полнота – характеристика, тесно связанная с *гибкостью* интерфейса. Поскольку пространство ЕЯ-запросов весьма неоднородно, следует говорить о различных типах запросов, т.е. групп запросов, имеющих сходное строение. Гибкость - показатель того, насколько разнообразные типы запросов может понимать ЕЯИ. Речь в основном идет о так называемых "трудных" типах запросов, в числе которых - вложенные, эллипсис, анафорические.

Другой важной характеристикой является *дружественность* интерфейса, которую можно определить как меру того, насколько ЕЯ-интерфейс удобен в работе, насколько корректно он может сообщать о проблемах понимания, может ли он помогать в переформулировке неберущихся запросов и т.д.

5.8.3. Критерии стоимости построения и сопровождения ЕЯ-интерфейса

Вышеперечисленные характеристики входят в оценки качества ЕЯ-интерфейса. Важным критерием при сравнении ЕЯ-интерфейсов является также трудоемкость его создания, то есть необходимое количество усилий (времени), требуемых для его построения. Ранние ЕЯ-интерфейсы создавались для каждой базы данных отдельно, и, разумеется, их стоимость была очень большой. Все эти системы были экспериментальными. Усугубляло проблему также то, что до конца 70-х годов не было единого универсального формального языка запросов к базам данных. Ранние системы понимания ЕЯ-запросов к СУБД были непортируемыми на другие базы данных, и зачастую лингвистическое ядро не отделялось от предметно-ориентированных настроек.

Современные промышленные системы построения ЕЯ-интерфейсов обладают достаточно высокой степенью портируемости, что, безусловно, снижает стоимость построения ЕЯ-интерфейса. Лингвистическое ядро является универсальным элементом, словарь содержит универсальную лексику, используемую во многих ЕЯ-интерфейсах, модели предметной области могут содержать шаблоны, общие для нескольких предметных областей и т.д. Зачастую используется метафора "фабрики и изделия",

изделием выступает ЕЯ-интерфейс, который собирается из готовых компонентов, которые настраиваются под конкретную базу данных.

Следует отметить, однако, что вопрос портирования на другие языки является открытым. Подавляющее большинство исследований проведено для английского языка, некоторые особенности которого изначально заложили в пути исследований мину замедленного действия - первоначально огромное количество усилий были потрачены на анализ синтаксиса. Сейчас можно сказать, что эти усилия не оправдали себя.

На трудоемкость создания ЕЯ-интерфейса влияет также необходимая квалификация настройщика ЕЯ-интерфейса. Для систем, требующих навыков лингвиста, трудоемкость построения ЕЯ-интерфейса больше, чем для систем, где для построения интерфейса требуется просто описать предметную область по некоторым предопределенным шаблонам и отобразить ее на схему базы данных, и дело здесь не только в стоимости труда лингвиста и инженера знаний или специалиста в области баз данных. Системы, требующие подстроек на уровне лингвистического ядра, являются более гибкими, поскольку позволяют разрешать проблемы понимания ЕЯ-запросов написанием соответствующих "заплаток", однако работы по написанию таких "заплаток" являются настолько сложными, требуют такого уровня понимания принципов машинного анализа ЕЯ в целом, что настройка ЕЯ-интерфейса на уровне лингвистического процессора зачастую возможна только авторами системы построения ЕЯ-интерфейса. Впрочем, сложность подстройки ядра очень сильно зависит от принципов анализа, используемого при написании инструментария, открытости ядра и т.д.

5.8.4. Вопросы портируемости

Коснемся теперь характеристики, присущей системам построения ЕЯ-интерфейсов – портируемость компонентов анализа. Существует несколько видов портируемости на:

1) другую предметную область,
2) другой язык,
3) другую СУБД,
4) другие платформы и языки программирования.

Портируемость на другую предметную область (ПО). Обычно, если ЕЯ-интерфейс портируется на другую ПО, систему необходимо "научить" словам и концептам, используемым в новой предметной области. Также надо отобразить модель ПО (МПО) и модель БД (МБД). Портируемость на другую предметную область может проводиться различными специалистами, и механизмы портирования могут быть ориентированы на:

- программиста: в некоторых системах часть кода (обычно небольшая и четко определенная) должна быть переписана;

• инженера знаний: конфигурация системы без программирования, построением схем предметной области и/или введением и описанием новых концептов ПО;

• администратора СУБД: БД-центрическое описание предметной области, например, описание некоторых важных для ПО ЕЯИ характеристик для каждой таблицы и поля БД;

• конечного пользователя: в предположении, что настройка ЕЯИ - постоянный процесс, ЕЯ снабжается средствами для настройки методом введения определений концептов на уровне ЕЯ и другими пригодными для конечного пользователя методами.

Портируемость на другой естественный язык. Подавляющее большинство ЕЯ-интерфейсов к БД были разработаны для английского языка. Переориентация ЕЯ-интерфейса на другой язык является для большинства систем большой проблемой, вследствие активного использования законов языка на уровне морфологии, синтаксиса, существенно отличающихся от языка к языку. Иногда настройка на язык возможна переписыванием синтаксических и семантических правил, а также новым наполнением словаря. В описываемой работе переносимость на другой язык является достаточно высокой, что обусловлено использованием преимущественно семантической информации при анализе.

Портируемость на другую СУБД. Система является портируемой на другую СУБД, если она позволяет перенос ЕЯ-интерфейса на другую СУБД. В случае, если ЕЯ-интерфейс генерирует запросы в распространенном языке (например, SQL), он может быть легко портирован на другую СУБД, если она поддерживает этот язык. Если язык запросов ЕЯ-интерфейса не поддерживается новой СУБД, ЕЯ-интерфейс может быть портирован переписыванием компонента генерации формального языка, однако это справедливо только для систем, в архитектуру которых включен промежуточный уровень запроса. В противном случае для портирования необходимо существенно переписать ЕЯ-интерфейс.

Портируемость на другую платформу. Некоторые ЕЯ-интерфейсы могли исполняться только на дорогих исследовательских машинах, используемых экзотические языки программирования (например, Lisp-машины), что делает их практически неприменимыми вреальных приложениях. Появление недорогих мощных компьютеров и повсеместная доступность языков, ориентированных на приложения искусственного интеллекта, как Lisp, Prolog, OCAML и Haskell, делают доступными ЕЯ-bynthatqcs, написанными на этих языках.

5.8.5. Основные составные части ЕЯ-интерфейсов

Кратко рассмотрим основные части ЕЯ-интерфейсов и их взаимосвязи. Прежде всего следует выделить из интерфейса *анализатор* ЕЯ как

компонент, реализующий тот или иной метод анализа естественного языка, и от принципов построения которого зависит архитектура системы и основные характеристики интерфейсов на основе данного компонента.

Работа анализатора заключается в построении *внутреннего представления* входного ЕЯ-текста либо запроса, обычно в виде некоторой структуры, например, синтаксического дерева, семантической сети, фреймовой структуры и т.д. Предшествующим этапом для процесса анализа является *лексический анализ (пред-анализ)*, который преобразует входной текст как последовательность символов, в цепочку лексем, поступающей на вход анализатора.

Необходимым компонентом работы анализатора является *словарь*, который содержит слова и фразы, обычно с привязкой к ним определенной информации, связанной с семантикой, морфологией и т.д., в зависимости от подхода анализа ЕЯ. Еще одним важным компонентом многих систем является *модель предметной области*, структура которой варьируется в очень больших пределах от системы к системе.

Для построения запроса на *формальном языке* источника данных используется *модель источника данных*, отражающая основную структуру СИД, ее части, существенные для данного ЕЯ-интерфейса.

Для перевода запроса из внутреннего представления системы в формальный язык источника данных предназначен процесс *генерации формального запроса.* Некоторые системы имеют также модуль синтеза ЕЯ, который может применяться для генерации естественно-языкового представления запроса, например, для верификации понимания запроса системой, а также для генерации уточняющих вопросов [8].

Модель предметной области в некоторых системах дополняется *базой знаний* со средствами вывода новых знаний.

5.9. РАБОТЫ С ОСНОВНЫМИ ОБЪЕКТАМИ, ПРОЦЕССАМИ И ЯВЛЕНИЯМИ, СВЯЗАННЫМИ С ИНТЕЛЛЕКТУАЛЬНЫМИ СИСТЕМАМИ, И ИСПОЛЬЗОВАНИЕ МЕТОДОВ ИХ НАУЧНОГО ИССЛЕДОВАНИЯ

В настоящее время научные исследования направлены на изучение и построение сложных, больших и слабо формализуемых технических, экологических, экономических, политических и социальных проблем, порождаемых процессом развития цивилизации. По мнению некоторых ученых, только использование всего потенциала знаний, накопленных человеком и создаваемых его интеллектом, позволяет успешно решать возникающие проблемы и находить пути адаптации человека к новым условиям его жизни при развитии цивилизации.

Интеллектуальные системы и носители интеллекта традиционно находили применение в различных системах управления, включая ручное и полуавтоматическое управление. С момента начала исследований по искусственному интеллекту понималось создание вычислительных систем,

обладающих свойствами имитации творческих процессов, логических выводов, восприятие естественно-язычных запросов и команд, аккумуляции знаний в компьютере. В качестве начальных научных направлений исследований новой информационной технологии можно выделить работы по интеллектуальным информационно-поисковым системам, обеспечивающим в процессе диалога человека с компьютером пользователей непрограммистов с базами данных и знаний на профессиональных языках пользователей близких к естественному языку.

Структура систем интеллектуального управления

Построение структуры системы интеллектуального управления связано в первую очередь с построением модели системы, в которой должны быть определены как традиционные элементы системы управления, так и модели обработки знаний, реализуемые интеллектуальной системой. В интеллектуальной системе управления новыми элементами по сравнению с традиционной системой управления являются все интеллектуальные преобразования или элементы управления знаниями, которые связаны с реализацией искусственного интеллекта, т.е. с использованием технологий экспертных систем, базы знаний, принятия решений, ассоциативной памяти, нечеткой логики, семиотических сетей, управления структурной динамикой и т.п.

Анализируя принятые структуры систем управления с решающими устройствами [22] можно и для обобщенной интеллектуальной системы использовать аналогичную структуру (рисунок 5.14), которая взаимодействует с внешней средой и в процессе ее получения от нее необходимой информации формирует цель действия и анализирует воздействия на систему (физические и информационные).

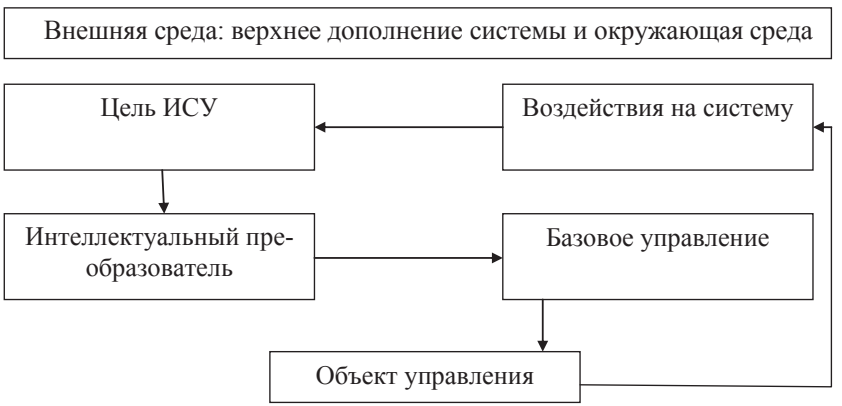


Рисунок 5.14 – Обобщенная схема системы интеллектуального управления

Определяющими элементами системы управления в этом случае являются: интеллектуальный преобразователь и базовая система управления.

Можно заметить, что использование интеллекта человека строиться на основе рассмотренной структуры, когда человек участвует в управлении в качестве интеллектуального преобразователя, согласованного с внешней средой через специализированные датчики и реализующий воздействие на систему управления через ручку управления или интерфейс взаимодействия с компьютером. В практике управления подвижными объектами такие системы получили специальное название. Системы управления подвижными объектами (в авиации, в космической технике, автомобиле и других транспортных средствах) получили название систем полуавтоматического управления, когда используется способность человека наблюдать и оценивать ситуации, возникающие при движении объектов, и формировать непрерывное управление ими.

В системах ручного управления на человека – оператора возлагаются дополнительные функции по управлению движением подвижным объектом, связанные с тем фактом, что он осуществляет полностью функции базового управления и воздействует на органы управления.

В ручных системах управления человек – оператор рассматривается как звено управления, формирующее закон и программу управления.

В автоматизированных системах управления (автоматизированным управлении космическим кораблем, атомной электростанцией и других подобных системах управления) человек – оператор оказывает воздействие на базовую автоматическую систему через вычислительные системы, что адекватно может быть описано предлагаемой структурой.

В случае использования в системе управления искусственного интеллекта в качестве интеллектуального преобразователя реализуются [18]:

- экспертные системы [11, 15];
- ситуационное управление [12,15];
- управление структурной динамикой сложных технологических [3] и другие интеллектуальные системы и их элементы.

Интересным примером использования интеллектуального преобразователя в системе управления является использование динамической экспертной системы.

Математическая модель интеллектуальной системы управления состоит из трех частей:

- интеллектуального преобразователя (экспертной системы, включающей базы данных и знаний);
- объекта управления;

• управляющее устройства системы (вычислительных и преобразующих и исполнительных устройств).

Интеллектуальный преобразователь представляет из себя логико-преобразующее устройство, которое преобразовывает информацию о внешней среде и объекте управления трансформирует в сигналы Y, в сигналы воздействия на управляющие устройства системы [18]. Математическая модель интеллектуального преобразователя оператором вида

$$Y = F(x, u, w, p, z), \quad (5.1)$$

где: F(.) – некоторый оператор интеллектуального преобразования, характеризующий структуру или работу интеллектуального преобразователя,

x – вектор состояния системы управления, u – вектор управления,

w – вектор воздействий внешней среды, p- вектор сигналов цели, z-вектор параметров объекта.

Объект управления в достаточно общем случае описывается уравнениями вида:

$$x = f(x, u, w, z, t),\ y = C(x),\ x(t0) = x0,\ t \geq t0, \quad (5.2)$$

где:

f(.) – вектор – функция, описывающая объект управления,

C(.) – заданная функция выходных сигналов,

t – координата времени,

y- вектор выхода или измерений.

Управляющее устройства системы (вычислительных и преобразующих и исполнительных устройств) формируют управляющие воздействия на объект управления u из множества его возможных значений в соответствие с решаемой задачей для достижения сформированной интеллектуальным преобразователем цели.

Для формирования воздействий на систему управления объектом в интеллектуальном преобразователе используется блок принятия решения, который может быть рассмотрен как самостоятельный элемент. Блок принятия решений формируется на основе теории принятия решений [25 … 27].

5.9.1. Модели принятия решения в условиях конфликта

Основные этапы процесса принятия решения согласно теории принятия решения декомпозируются на следующие этапы [27]:

• определение цели решения возникшей проблемы;

- выбор наиболее предпочтительного варианта действий, ведущего к достижению цели;
- реализация решения (выбранного варианта действия).

Определение цели решения возникшей проблемы реализуется в блоке интеллектуального преобразователя, получающего и обрабатывающего информацию о внешней среде с системы датчиков.

В условиях конфликта цель может зависеть от имеющихся ресурсов и факторов, которые образуют *проблемную ситуацию*, т.е. ситуацию принятия решения в условиях конфликта. Способ действия для управления объектом в процессе принятия решения называют *стратегиями*, а результат, к которому может привести выбранная стратегия, называют *исходом*. Условия конфликта порождают факторы, воздействующие на стратегию и соответственно на управление, реализуемое интеллектуальной системой. С точки зрения наличия информации об условиях конфликта факторы разделяются на две группы:

- *определенные* (фиксированные) факторы, значения которых известны;
- *неопределенные* факторы, о которых априорно не известно, какое значение они примут.

В зависимости от происхождения неопределенные факторы делятся на *случайные* и *неопределенные нестохастического характера*, состоящие из *природных* и *стратегических*. Математическая модель принятия решений формируется с учетом всех факторов и имеющейся о них информации. Упрощенная модель принятия решения в этом случае может быть описана следующей системой

$$D0 = < Y, G, U, L, J, \Omega >, \qquad (5.3)$$

где:

Y - множество исходов (результатов);

G - модель предпочтений исходов (принимаемых решений);

U - множество стратегий принятия решений,

L - множество возможных значений неопределенных факторов;

J - функция, определяющая взаимосвязь неопределенного фактора и исход, получаемый в результате принятого решения;

Ω - вся остальная информация о принимаемом решении в формализованном виде (сведения о конфликте, предпочтения других лиц, участвующих в конфликте и др.).

Удобство использования модели (5.3) в условиях конфликта определяется тем положением, что она позволяет просто и наглядно связать значения неопределенных факторов и стратегий с управлением, реализуемым интеллектуальной системой. Множества Y, G, U, L и функция J формально задают компоненты принимаемого решения и определяют связь с системой управления через понятия критерия и показателей эффективности системы. В теории управления наиболее часто отношения предпочтения описываются с помощью специальных функций: показателей

качества и критериев. Под *показателем* W качества или эффективности системы управления понимается мера степени соответствия реального результата управления Y требуемому Yтр для достижения цели и получения оценок или измерений интенсивности исходов. Под *критерием* К понимается правило, позволяющее сопоставлять принимаемые решения и стратегии с точки зрения выбранных показателей оценок исходов. Критерии вводятся на основе определенной концепции рационального поведения интеллектуальной системы [25]: пригодности, оптимизации и адаптивности.

Более общей моделью принятия решения в условиях конфликта по сравнению с (5.3) является модель динамической системы интеллектуального управления на основе моделей теории игр [28, 29] и современного аппарата функционального анализа [30].

Теория игр как раздел математики в настоящее время стал теорией математических моделей [29]. При этом под *конфликтом* понимается явление, которое связано с ответом на вопросы:

- Кто и как в этом явлении участвует?
- Каковы возможные исходы конфликта?
- Кто в этих конфликтах заинтересован?

В чем состоит заинтересованность участников конфликта? Для описания конфликта вводится понятия:

- *коалиций действия* RД, объединяющее множество игроков (участников конфликта) по их действиям;
- *коалиций интересов* RД объединяющих игроков по интересам;
- *стратегий*, характеризующих решения коалиций К (действий RД и коалиций интересов RИ, которые в зависимости от типа игры могут иметь одних и тех же игроков или образовываться из разных игроков);
- *отношений предпочтения* G, как абстрактного бинарного отношения на множестве всех стратегий (нередко отношения предпочтения задаются функцией выигрыша WK и тогда коалиция К, если она предпочитает ситуацию х ситуации у, то обозначают через отношение предпочтение в виде х GK у или при использовании функции выигрыша, если WK (х)WK (у), то в виде х WK у).

Тогда формальное описание конфликта состоит в задании системы

$$\Gamma = < RД , SRД, S, RИ, GRИ >, \qquad (5.4)$$

где: SRД- множество стратегий коалиции действия; GRИ – множество отношений коалиции интересов.

В работе [30], как и в ряде других исследованиях по общей системной теории управления, предложено рассматривать динамические системы управления состоящие из объекта управления, характеризуемого некоторым множеством состояний, и регулятора, под которым понимается математическая модель, состоящую из элемента, обеспечивающего оценку состояния объекта, и элемента, формирующего управление.

Для описания динамической системы используется математическая модель совокупности элементов, удовлетворяющих следующим основным аксиомам:

A. Заданы множества моментов времени T, множество состояний системы X. Множество значений входных воздействий U, непустое множество их допустимых значений

$$W = \{\omega: T \rightarrow U\}, \tag{5.5}$$

множество значений выходных величин Y и множество их допустимых значений

$$U = \{\gamma: T \times X \rightarrow Y\}. \tag{5.6}$$

B. Множество T есть некоторое упорядоченное подмножество множества вещественных чисел (направление времени).

C. Существует *переходная функция*

$$\Phi = \{\phi: T \times T \times X \times \Omega \rightarrow X\}, \tag{5.7}$$

где состояния системы $x(t) = \phi[t, t0, x(t0), \omega] \in X$.

D. Задано входное отображение

$$U = \{\gamma: T \times X \rightarrow Y\}. \tag{5.8}$$

Приведенная модель в наиболее общем виде динамического объекта или системы изображена на рисунке 5.15.

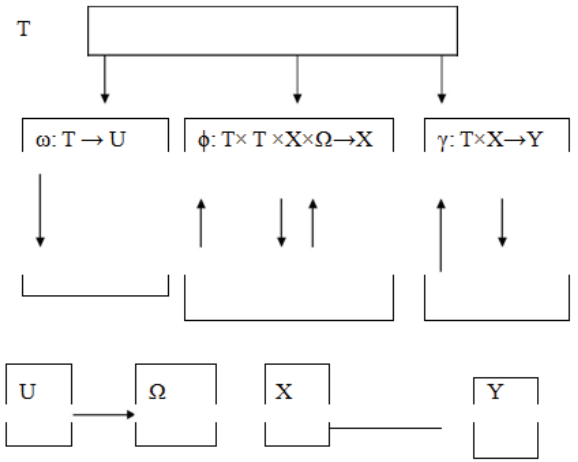


Рисунок 5.15 – Множественная схема динамической системы

Для описания динамической системы используются дополнительные термины. Так, состояние системы x в момент времени t, или пара элементов множества T×X называется *событием* (или *фазой*) динамической системы. Множество T×X называется *пространством событий*, или *фазовым пространством*. В том случае, когда множество выходных воздействий используется для управления, то оно называется управлением. Управление переводит систему из одного состояние в некоторое другое. При этом система находится в движении, описывая в пространстве состояния *траекторию*. В рассмотренной формализации динамическая система рассматривается как объект управления.

Для реализации систему управления в классической теории систем выделяют объект управления и регулятор управления, состоящий в простейшем случае из элемента оценки состояния системы и элемента, формирующего закон управления. Так как объект управления нами уже рассмотрен как динамическая система, то введем понятия *закона управления* и *оценки состояния* системы.

Законом управления принято называть отображение

$$k : T \times X \rightarrow U, \qquad (5.9)$$

где величины $u(t) = k[t; x(t)]$ управления, принадлежащие множеству U.

Для реализации управления необходимо знание переменных состояния системы x(t), что требует операции определения обратного отображения

$$\gamma\text{-}1 : Y \rightarrow X \qquad (5.10)$$

и тогда координаты системы определяются из условия

$$x(t) = \gamma\text{-}1[y(t)]. \qquad (5.11)$$

Кроме того, для оценки состояния системы необходимо оценивать точность определения x(t), т.е. получать оценку состояния x0(t).

Структурно – логическая схема системы управления, состоящая из объекта управления и регулятора, показана на рисунке 5.16.

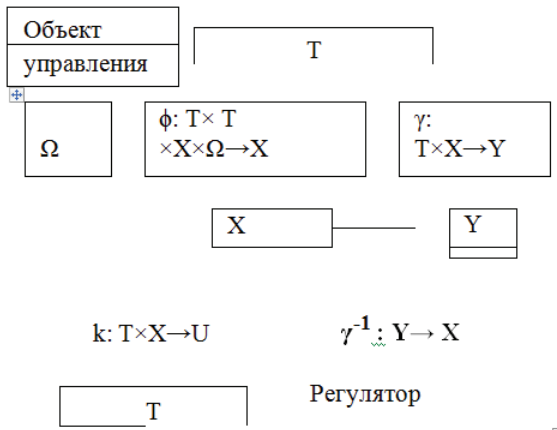


Рисунок 5.16 – Структурно-логическая схема блока основного управления системы

Для учета факторов воздействия внешней среды и конфликта необходимо провести учет воздействий этих объектов. При этом факторы внешней среды и условий конфликта необходимо рассматривать и описывать с позиций динамической системы.

При принятии решения в системе интеллектуального управления возникает необходимость объединения интеллектуального преобразователя, включающего блок принятия решения, и динамическую систему управления.

5.9.2. Определение оптимальной интеллектуальной системы принятия решения и управления в условиях конфликта

Понятие оптимальности получило широкое распространение в теории управления и теории принятия решений. Под оптимальной системой автоматического управления понимают «наилучшую» систему, которую выбирают из множества систем по принятому показателю качества системы или эффективности ее функционирования. При этом система является оптимальной, если она обеспечивает экстремум принятого показателя. В зависимости от конкретного вида показателя понятие оптимальности связывается с критерием, т.е. с минимумом или максимумом показателя. Если, например, в качестве показателя используется средняя квадратическая ошибка системы, то оптимальной системой считают ту, которая реализует критерий – минимумом средней квадратической ошибки. Если же в качестве показателя используется вероятность невыхода ошибки системы из заданных допусков, то оптимальной считают ту систему, которая реализует критерий – максимум невыхода ошибки системы из заданных допусков.

Принятие решений в условиях определенности (фиксированности) факторов и в случае, когда неопределенность является случайной и о факторе имеется полная априорная информация, то оптимальность решения определяется аналогично тому, как это делается в системах автоматического управления.

В условиях стратегической (поведенческой) неопределенности, которая появляется в условиях конфликта, понятие оптимальности принимаемого решения значительно труднее поддается формализации. Теория математических моделей принятия оптимальных решений составляет значительную часть науки, которая получила название *исследования операций*. Особое место в исследовании операций занимает раздел, занимающейся теорией математических моделей принятия оптимальных решений в условиях конфликта, который получил название *теории игр*. Теории игр, как теория математических моделей базируется на использовании формальных, знаковых моделей для описания конфликта, а также использует формальные средства их анализа. В теории игр успешно были реализованы определения понятия конфликта и принятия решения. В тоже время понятие оптимальности удалось в настоящее время реализовать только для части игр. Поэтому при анализе систем принятия решений и управления мы ограничимся рассмотрением только тех конфликтов, которые описываются моделями бескоалиционных (множества коалиций действия и интересов совпадают, определены функции выигрыша) и антагонистических (число игроков равно двум, а значения их функций выигрыша в любой ситуации равны и противоположны по знаку) игр. Для бескоалиционных и антагонистических игр условия оптимальности основываются на понятии равновесия.

Вопрос об оптимальности управления и принятия решений в интеллектуальных системах усложняется и тем фактом, что для реализации системы используются экспертные знания, которые не всегда удается реализовать. Эта трудность преодолевается за счет допущения, что в системе реализуется интеллектуальный датчик, который однозначно определяет информацию о действиях другой стороны за счет физических измерений действия противоположных игроков, воздействующих на систему управления и принятия решения в условиях конфликта. Фактически в интеллектуальной системе реализуется рациональное управление и принятие решения. Достоверность и степень приближения к точному оптимальному управлению и принятию решения определяется той информацией, которую получает, обрабатывает и выдает интеллектуальная система, или интеллектуальный датчик. При этом в результате ошибок интеллектуального датчика, связанных с использованием экспертных знаний, появляются риски, связанные с ошибочным формированием цели принятия решения и управления, определением показателей и критериев оценки эффективности работы системы, выбором лучшей системы. Для уменьшения рисков в интеллектуальной системе используются методы обучения и самообучения, которые в настоящее время начинают разрабатываться в интеллектуальных системах и которые всегда присутствовали в биологическом интеллекте [17 … 21].

Решение задач синтеза систем оптимального управления и принятия решения в условиях конфликта традиционно связаны с математическими методами отыскания экстремумов функций и функционалов, нахождения равновесных состояний, что позволяет решать задачи синтеза систем управления и принятия решений в практически важных случаях.

Рассмотрение структуры систем интеллектуального принятия решения и управления в условиях конфликта (рисунок 5.17) показывает, что система состоит из трех основных элементов (подсистем), каждый из которых имеет свои модели и специфические методы исследования:

- интеллектуальная подсистема;
- подсистема принятия решения;
- подсистема управления объектом и объект управления.

Рисунок 5.17 – Структура интеллектуальной системы принятия решения и управления в условиях конфликта, порождаемого конфликтующими системами

Не смотря на свою сложность и методологические особенности элементов интеллектуальных систем принятия решения и управления в условиях конфликта при определении оптимальной системы, удается выделить следующие основные задачи:

- построение модели условий конфликтного взаимодействия оптимизируемой системы с другими системами, изучение и описание информации о воздействиях на систему;
- формирование интеллектуальной подсистемы, включающей датчики, базу знаний, блоки экспертизы, оценки состояния, формирования целей, выбор показателей и критериев оптимизации системы и других элементов интеллектуальной деятельности;
- принятие решений, обеспечивающих оптимальное противодействие одной или нескольким конфликтующим системам;
- определение и математическое описание классов допустимых систем управления объектом управления;
- отыскание экстремума критерия или точек равновесия в игровых задачах, а также соответствующих им характеристик.

Формирование модели условий конфликтного взаимодействия оптимизируемой системы с другими системами основывается в простейшем случае с позиции теории игр как конфликт двух игроков (А, В). При этом конфликт рассматривается как операция, в которой игроки имеют различные цели и реализуют свою деятельность и выбирают свои стратегии в соответствии со своими целями (рисунок 5.18) [1, 24 … 26].

Исследование операции проводится всегда с точки зрения одного игрока, а в рассматриваемом случае это проводится с позиции интеллектуальной системы принятия решения и управления (для определенности примем, что в конфликте – это игрок А).

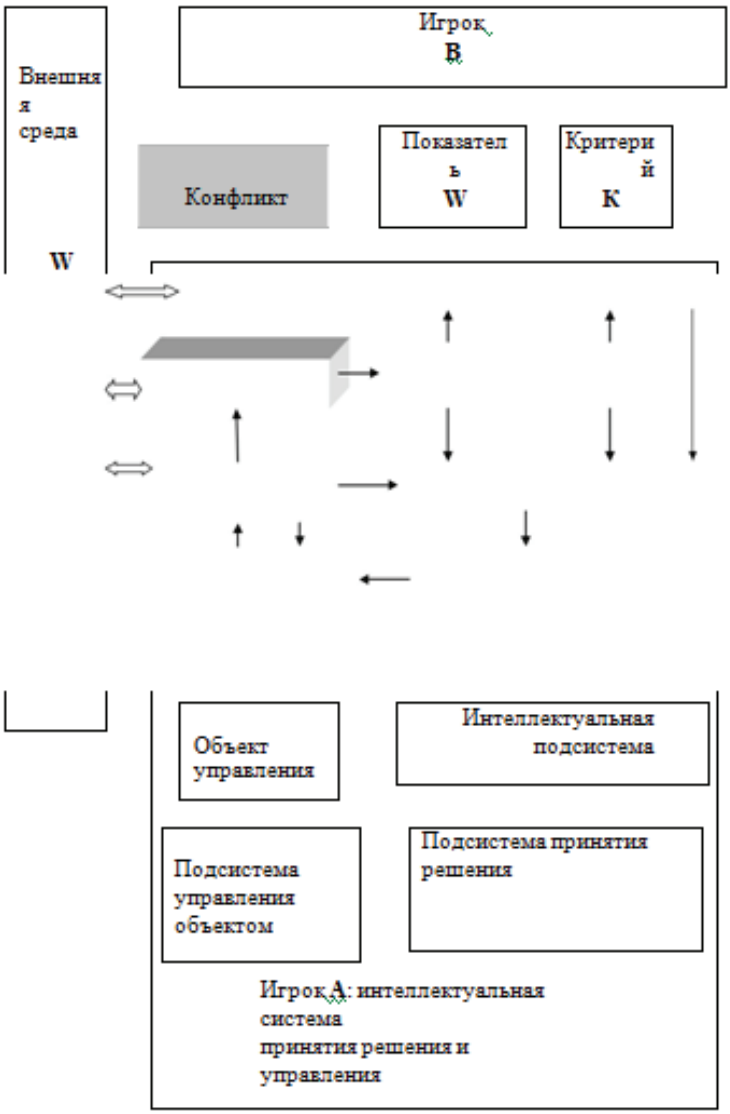


Рисунок 5.18 – Общая схема конфликта как операции двух систем с разными целями

Будем считать, что эффект достижения цели определяется векторными показателем W = (WA, WB) и критерием К = (КА, КВ), связанные с целями поведения сторон. При этом в зависимости от наличия датчиков получения информации о целях действия игроков предполагается, что идентификация целей позволяет интеллектуальной системе определять критерии и показатели, используемые игроками для идентификации показателей и критериев игроков.

Вопросы формирование интеллектуальной подсистемы, включающей датчики, базу знаний, блоки экспертизы, оценки состояния, формирования целей, выбор показателей и критериев оптимизации системы и других элементов интеллектуальной деятельности были рассмотрены в первом разделе лекции, когда анализировались интеллектуальные системы и их элементы.

Принятие решений, обеспечивающих оптимальное противодействие одной или нескольким конфликтующим системам конфликтующим системам, основывается в условиях конфликта на использовании моделей, методов и алгоритмов принятия оптимальных решений в теории игр. В рассматриваемом нами случае мы ограничиваемся матричными антагонистическими (игры двух лиц с нулевой суммой), которые имеют развитый инструментарий решения прикладных задач

Определение и математическое описание классов допустимых систем управления объектом управления основывается на теории систем управления [1, 31]. В основе рассматриваемой задачи оптимизации системы управления объектом используется подход, основанный на использовании корреляционной теории статистической оптимизации систем, разработанной профессором Н.И. Андреевым [22, 32]. В его работах разработаны методы исследования сложных линейных и нелинейных динамических систем, подверженных случайным воздействиям. При постановке задач определения оптимальных динамических систем по статистическим критериям (по минимуму средней квадратической ошибки системы, максимуму вероятности невыхода ошибки системы из заданных допусков и др.) им разработана методология учета ограничений типа неравенств, накладываемые на функции управления и вектора. Большое внимание им уделено исследованию динамических систем с заданной структурой, так как для прикладных задач эти системы играют, с одной стороны, важную роль, а, с другой стороны, приводят к значительному усложнению математических задач оптимизации этого класса систем. В его работах исследовано ряд новых задач, относящихся к оценке параметров системы, фильтру Калмана, синтезу нелинейных динамических систем, построению адаптивных систем оценивания и управления, развитию методов принятия статистических решений, определению оптимальных систем управления с решающими устройствами.

Проиллюстрируем проведенный анализ постановки задачи определения оптимальной интеллектуальной системы принятия решения и управления в условиях конфликта на простом примере, в иной постановке задачи оптимизации.

Пример. Рассмотрим задачу определения интеллектуальной системы принятия решения и управления, на вход которой поступает сигнал, представляющего сумму полезного сигнала G(t) и помехи Z(t). При этом будем предполагать, что слежение за полезным сигналом осуществляет интеллектуальная система управления (игрок А), а помеху формирует противоположная сторона (игрок В), которая выбирает характеристики такими, чтобы ошибки следящей системы были большими. В результате возникает конфликт, который часто рассматривается в теории игр. Для наглядности и простоты изложения будем полагать, что каждая из сторон имеет две стратегии управления и по этой причине можно ограничить решение задачи на основе матричной и биматричной игр.

Пусть помеха представляет собой нормально распределенный белый шум с интенсивностями (α1, α2), выбором величины которых управляет игрок А. Полезный сигнал G = G0 числовая случайная величина, принимающая значения β0 и -β0 с вероятностью 0,5, величина которых формируется игроком А и может принимать значения (β01, β02).

Желаемая выходная величина формируемой системы слежения - полезный сигнал. Система выбирается на классе стационарных систем, время регулирования которых не превосходит время Т.

В качестве показателя качества системы принимается суммарная ошибка системы ошибка системы D∑, которая определяется биноминального распределения полезного сигнала D1 и случайным характером помехи D2. Запишем ошибки системы в следующем виде:

$$D_1 = \int_0^T \int_0^T \delta(\tau_1 - \tau_2)\omega(\tau_1)\,\omega(\tau_2)\,d\tau_1 d\tau_2,$$

где: δ(τ1- τ2) – дельта-функция, принимающая нулевое значение, если (τ1- τ2)≠0, и равна ∞, если(τ1- τ2)=0 ;

ω(τ1) – весовая функция при переменной τ1; ω(τ2) – весовая функция при переменной τ2 ;

$$D_2 = (\beta_0)^2 \{\int_0^T \omega(\tau)\, d\tau_1 - 1\}^2;$$

$$D\Sigma = D_1 + D_2.$$

Базовая подсистема управления будет описываться весовой функцией, параметры которой зависят от решения игровой задачи и выбора

подсистемой интеллектуального управления цели системы и соответствующих показателей и критериев оптимизации.

Тогда для иллюстрационного примем Т =1 и выражение для весовой функции можно записать в виде

$$\omega_{IJ}(\tau) = \beta^2_{0J} (\alpha_i + \beta^2_{0J})^{-1},$$

где управление реализуется в виде весовых функций в зависимости от стратегий игроков (В - максимизирует дисперсию ошибки, а А - минимизирует ошибку слежения).

В этом случае дисперсии для принятой весовой функции записываются в следующем виде, учитывающем стратегии игроков

$$D_1^{IJ} = \alpha_i (\alpha_i \beta^{-2}_{0J} + 1)^{-2},$$
$$D_2^{IJ} = \alpha_i^{\,2} \beta^2_{0J} (\alpha_i + \beta^2_{0J})^{-2}.$$

Интеллектуальная подсистема В, наблюдая с помощью своих информационных и физических датчиков за игроком А, определяет его действия и формирует свою стратегию движения в каждой конкретной ситуации. Возникает задача ситуационного управления движением, когда решения и управления должны осуществляться в соответствие с той ситуацией, которая складывается в текущий момент конфликта.

Одной из сложных задач в этом случае является задача определения типа антагонизма между игроками и определения целей действия систем, выяснения их интересов. В практике конфликтных ситуаций о действиях другой стороны или игрока, как правило, используется информация, получаемая через информационные сообщения, телекоммуникационные сети и путем сбора различными способами конфиденциальной информации. При реализации интеллектуальной экспертной системы используются эксперты и их оценки, но и возможен аналитический подход к решению идентификации намерений и цели противоположной стороны на основе анализа конфликта. Если реализуется интеллектуальная подсистема на основе ситуационного управления, то возможна декомпозиция общей задачи на две части, когда интеллектуальная подсистема идентифицирует конкретную ситуацию, а управление в конкретной ситуации формируется на основе решения игровых задач для определения неопределенных стратегий, или неопределенных параметров противодействующей стороны. Вместе с тем интеллектуальная подсистема должна определять тип модели игры в конфликте. Возникает вопрос, как это может реализовать интеллектуальная подсистема.

Покажем возможность оценки действий противника на основе обработки информации по решению задачи с помощью теории игр. Для этого

проанализируем результаты принятия решения при реализации матричной антагонистической игры. Вначале рассмотрим влияние конфликта на управление на примере единичной ситуации, когда время управления Т фиксировано и единственное, а затем рассмотрим условную модель ситуационного управления на основе сценария, в котором достижение цели обеспечивается несколькими ситуациями управлениями в разные моменты времени Tv.

Примем, что игроки могут реализовать по две стратегии, т.е. конфликт описывается матрицей размерности 2x2, где (β201 = 1, β202= 2) значение стратегий игрока В и (α1 = 1, α2 = 2) значение стратегий игрока А. В этом случае при принятых исходных данных матрицы выигрышей можно записать в следующем виде:

при анализе системы по дисперсии D2

Стратегии В \ А	β^2_{01} **= 1**	β^2_{02}**= 2**	Min по j	**Max min =**
α_1 = 1	0,5	0, 667	0,5	**0,5**
α_2= 2	0,333	0,4	0,4	
Max по i	0,5	0,667		
Min max =	***0,5***			

при анализе системы по дисперсии **D_1**

Стратегии В \ А	β^2_{01} = 1	β^2_{02}= 2	Min по j	**Max min =**
α_1 = 1	0,25	0,011	0,011	
α_2 = 2	0,011	0,08	0,08	**0,08**
Max по i	0,011	0,08		
Min max =	***0,08***			

при анализе системы по дисперсии D1

Стратегии B \ A	β^2_{01} = 1	β^2_{02}= 2	Min по j	**Max min =**
α_1 = 1	0,25	0,011	0,011	
α_2 = 2	0,011	0,08	0,08	**0,08**
Max по i	0,011	0,08		
Min max =	***0,08***			

при анализе системы по дисперсии D_Σ

Стратегии B \ A	β^2_{01} = 1	β^2_{02}= 2	Min по j	**Max min =**
	0,75	0,678	0,678	
α_2 = 2	0,344	0,48	0,48	**0,48**
Max по i	0,75	0,48		
Min max =	***0,48***			

Все три антагонистические игры имеют седловые точки

max min Dμij = min max Dμ, μ=(1, 2, 3), i=J=(1, 2)

i j j i

в чистых стратегиях, но цены игр у них разные. Интересным результатом анализа дисперсий ошибок анализируемой системы является то, что гарантированная суммарная ошибка системы (равна 0,48) меньше гарантированной ошибки, получаемой только при учете нормальной составляющей ошибки (равной 0,5). Понятно, что полученное уменьшение дисперсии суммарной ошибки в результате отказа от стратегии α1, которая ведет к увеличению дисперсии нормальной составляющей ошибки. Это также связано с тем, что дисперсии ошибок от изменения сигнала помехи существенно меньше при α2 влияют на дисперсию D1. По этой причине, если игрок А будет выбирать свою стратегию из условия увеличения дисперсии помехи, то он может быть наказан игроком В, который в этом случае может реализовать свою стратегию β201 = 1 (провести адаптивное

управление) благодаря этого получить дополнительный выигрыш, который будет равен 0,344, т.е. меньше гарантированного выигрыша на 0,135 (уменьшает дисперсию на 28.5%). Таким образом интеллектуальная подсистема на основе получения оценок интенсивности шума сожжет определить, какой стратегии придерживается игрок А при противодействии игроку В. Логика рассуждений интеллектуальной подсистемы может быть продолжена в том направлении, если игрок В откажется от использования гарантированных оценок, а будет применять критерии: минимаксного сожаления Сэвиджа, пессимизма-оптимизма Гурвица, «недостаточного основания» Бернулли [25, 27].

Рассмотрим теперь случай, когда для достижения цели управления необходимо выполнения нескольких операций по слежению, согласно определенному сценарию (рисунок 5.19), в котором последовательно реализуются три ситуации со временем управления Т1=1, Т2=2, Т3 =3 а также показано начало работы системы и достижение цели в случае успешного выполнения все планируемых операций.

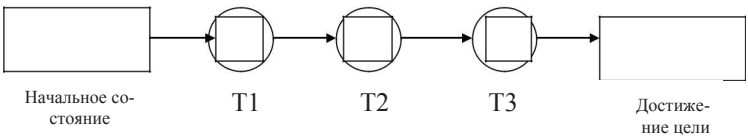


Рисунок 5.19 – Сценарий на основе сетевой модели, вершины которой соответствуют фактам, а дуги связям

В качестве критерия эффективности достижения цели примем условие получения оптимальных оценок в каждой ситуации, как это было сделано в первой ситуации при Т1.

Для произвольного Tν дисперсии и определяемые весовые функции будут равны

$$D_1^{IJ} = T_\nu \alpha i \, (\alpha i \, \beta^{-2}_{0J} + T_\nu)^{-2},$$

$$D_2^{IJ} = \alpha i^{\,2} \, \beta^2_{0J} \, (\alpha i + T_\nu \beta^2_{0J})^{-2},$$

$$\omega_{IJ}(\tau) = \beta^2_{0J} \, (\alpha i + T\nu \, \beta^2_{0J})^{-1},$$

где ν = (1, 2, 3), i =(1, 2), J=(1, 2), если использовать данные , принятые в начальных расчетах дисперсий и весовой функции при Т1.

Одним из интересных предложений по развитию интеллектуальной подсистемы для оценки степени антагонизма между игроками и

соответственно между интеллектуальными системами в условиях конфликта стало использование теории биматричных игр.

5.10. ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ В СОВРЕМЕННОЙ РОБОТОТЕХНИКЕ

5.10.1. Интеллектуальные роботы

Эволюция представлений о путях развития робототехники, ее целях и задачах весьма схожа с тем, что наблюдается с такой областью, как искусственный интеллект. Декларируемые общие принципы и, как казалось, понимание путей достижения некой глобальной цели исследования сменилось узкой специализацией, множеством частных, зачастую не связанных между собою подцелей и направлений.

Объясняется это тем, что поставленные изначально задачи оказались значительно более сложными, требующими создания совершенно иных моделей, методов и технологий, и прежде всего – технологий искусственного интеллекта.

Технологии ИИ всегда были тесно связаны с робототехникой. Не случайно одним из направлений ИИ до сих пор считается целенаправленное поведение роботов (создание интеллектуальных роботов, способных автономно совершать операции по достижению целей, поставленных человеком).

Робот – это технический комплекс, предназначенный для выполнения различных движений и некоторых интеллектуальных функций человека и обладающий необходимыми для этого исполнительными устройствами, управляющими и информационными системами, а также средствами решения вычислительно-логических задач.

В настоящее время различают 3 поколения роботов:

1. ***Программные.*** Работают по жестко заданной программе (циклограмме).
2. ***Адаптивные.*** Имеют возможность автоматически перепрограммироваться (адаптироваться) в зависимости от обстановки. Изначально задаются лишь основы программы действий.
3. ***Интеллектуальные.*** Задание вводится в общей форме, а сам робот обладает возможностью принимать решения или планировать свои действия в распознаваемой им неопределенной или сложной обстановке.

5.10.2. Архитектура интеллектуальных роботов

Общепринято мнение, что интеллектуальный робот обладает т.н. моделью внешнего мира или внутренней средой, что позволяет роботу действовать в условиях неопределенности информации. В том случае, если эта модель реализована в виде базы знаний, то целесообразно, чтобы эта база знаний была динамической. При этом коррекция правил вывода в

условиях меняющейся внешней среды естественным образом реализует механизмы самообучения и адаптации.

Если отойти от подобного «перечислительно-функционального» определения ИР, то останется лишь два более или менее конструктивных определения. Первое заключается в том, что интеллектуальный робот – это робот, в состав которого входит *интеллектуальная* система управления. Тогда достаточно только выбрать определение интеллектуальной системы (ИС). Например, определить ИС как компьютерную систему для решения задач, которые или не могут быть решены человеком в реальное время, или же их решение требует автоматизированной поддержки, или же их решение дает результаты сопоставимые по информативности с решениями человека.

Кроме того, среди прочего подразумевается, что задачи, решаемые ИС не предполагают полноты знаний, а сама ИС должна обладать способностями: к упорядочению данных и знаний с выделением существенных параметров; к обучению на основе позитивных и негативных примеров, к адаптации в соответствии с изменением множества фактов и знаний и т.д. ([Финн, 2004b])

Другим, менее формальным, определением интеллектуальности робота может быть способность системы решать задачи, *сформулированные в общем виде*. Это определение является, не смотря на свою «слабость», достаточно конструктивным по крайней мере для того, чтобы определить «степень интеллектуальности» робота.

Итак, несмотря на множество предлагаемых критериев интеллектуальности, самым сильным остается по-прежнему требование, согласно которому роль человека при взаимодействии с ИР должна свестись лишь к постановке задачи.

На сегодняшний день считается, что в состав интеллектуального робота должны входить (рисунок 5.20):

Исполнительные органы – это манипуляторы, ходовая часть и др. устройства, с помощью которых робот может воздействовать на окружающие его предметы. Причем по своей структуре это сложные технические устройства, имеющие в своем составе сервоприводы, мехатронные части, датчики, системы управления. По аналогии с живыми организмами это руки и ноги робота.

Датчики – это системы технического зрения, слуха, осязания, датчики расстояний, локаторы и др. устройства, которые позволяют получить информацию из окружающего мира.

Система управления – это мозг робота, который должен принимать информацию от датчиков и управлять исполнительными органами. Эта часть робота обычно реализуется программными средствами. В состав системы управления интеллектуального робота должны входить следующие компоненты:

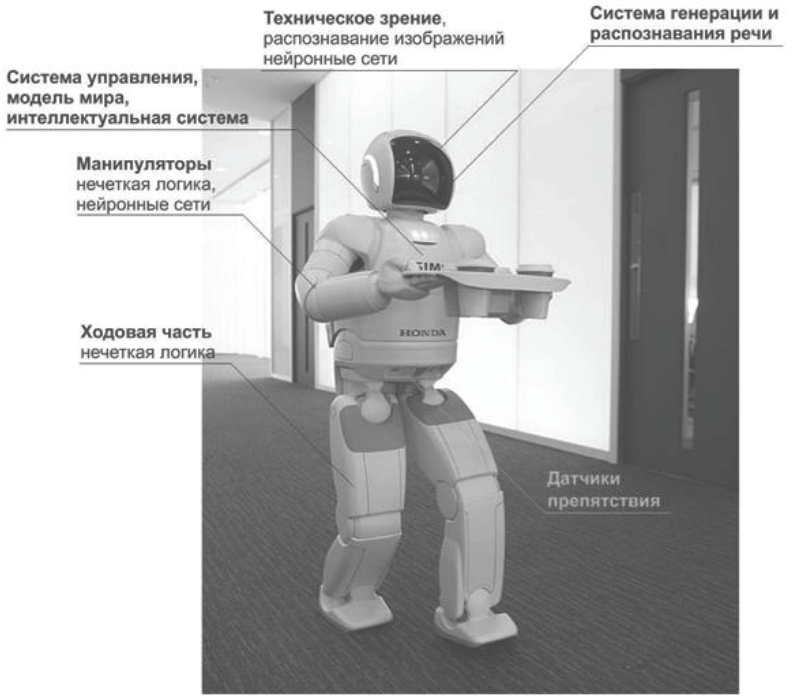


Рисунок 5.20 – Компоненты робота

Модель мира – отражает состояние окружающего робот мира в терминах, удобных для хранения и обработки. Модель мира выполняет функцию запоминания состояния объектов в мире и их свойств.

Система распознавания – сюда входят системы распознавания изображений, распознавания речи и т.п. Задачей системы распознавания является идентификация, т.е. «узнавание» окружающих робот предметов, их положения в пространстве. В результате работы компонентов системы распознавания строится модель мира.

Система планирования действий – осуществляет «виртуальное» преобразование модели мира с целью получения какого-нибудь действия. При этом обычно проверяется достижимость поставленной цели. Результатом работы планирования действий является построение планов, т.е. последовательностей элементарных действий.

Система выполнения действий – пытается выполнить запланированные действия, подавая команды на исполнительные устройства и контролируя при этом процесс выполнения. Если выполнение

элементарного действия оказывается невозможным, то весь процесс прерывается и должно быть выполнено новое (или частично новое) планирование.

Система управления целями – определяет иерархию, т.е. значимость и порядок достижения поставленных целей.

Важными свойствами системы управления является способность к обучению и адаптации, т.е. способность генерировать последовательности действий для поставленной цели, а также подстраивать свое поведение под изменяющиеся условия окружающей среды для достижения поставленных целей.

5.10.3. Технологии искусственного интеллекта для интеллектуальных роботов

Нечеткая логика находит применение, в основном, на нижнем уровне для управления конкретными устройствами. Методы нечеткой логики позволяют заменить решение дифференциальных уравнений для задач управления менее ресурсоемкими логическими методами нечеткого вывода.

Нейронные сети изначально были хорошо приспособлены для задач классификации. Первая модель перцептрона решала именно эту задачу. Именно поэтому наиболее широкое применение нейронные сети находят в системах распознавания образов. Возможно применение нейронных сетей для управления манипуляторами. Ведутся попытки создания на базе однородных нейроподобных структур систем выбора действий интеллектуальных роботов.

Интеллектуальные системы являются необходимым компонентом, решающим задачи создания модели мира, системы планирования действий и управления целями. База знаний в интеллектуальных системах является одной из главных частей модели мира и функций его преобразования.

Распознавание изображений давно стало необходимой частью сложных робототехнических систем. Системы объемного зрения позволяют получить информацию об ориентации объектов в пространстве. В этой области в настоящее время происходят значительные изменения.

Распознавание и генерация речи необходимы для эффективного общения с человеком. Без этих технологий полноценное общение с человеком невозможно. В области генерации речи по тексту достигнуты значительные успехи. С распознаванием речи дела обстоят хуже, поскольку это более сложная задача.

Многоагентные системы используются для коллективного управления большим количеством роботов, способных работать как по отдельности, так и единой командой.

5.10.4. Учебные роботы LEGO Mindstorms Education EV3

LEGO MINDSTORMS Education EV3 (рисунок 5.21) это третье поколение робототехнических конструкторов серии LEGO Mindstorms.

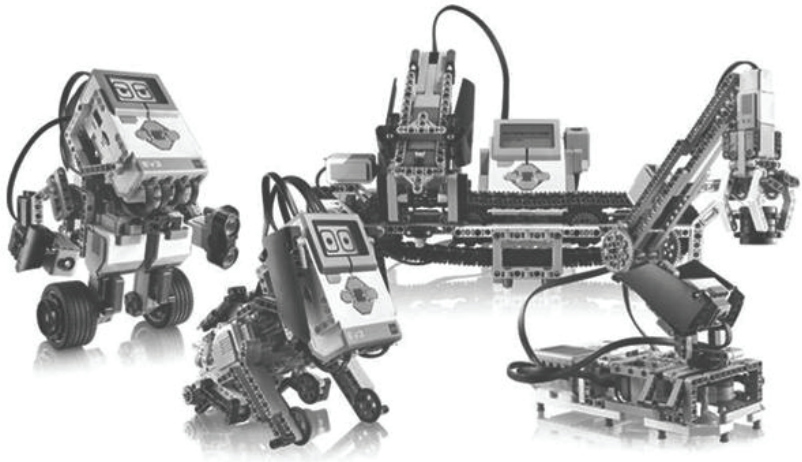


Рисунок 5.21 – Учебные роботы LEGO Mindstorms Education EV3

Данный набор разрабатывался в массачусетском технологическом институте (**MIT – Massachusetts Institute of Technology**) совместно с компанией LEGO. Вы можете конструировать роботов с множеством датчиков и моторов, или измерять расстоянии, освещенность, температуру, проводя научные эксперименты. Первый LEGO Mindstorms RCX, был выпущен в 1998, основой для которого послужил микроконтроллер H8. Следующим поколением стал LEGO MINDSTORMS Education NXT, выпущенный в 2006, он заметно расширил функционал первого набора, также в нем использовался новый микроконтроллер ARM7 32bit CPU. ROBOLAB, разработанный в качестве основы для LabView компанией National Instruments.

LEGO MINDSTORMS Education EV3 был улучшен новым процессором, увеличено количество поддерживаемых портов, новым программным обеспечением, новым USB портом, слотом для SD карт и функцией auto-ID.

Auto-ID функция С фукцией Auto-ID микрокомпьютер EV3 и программное обеспечение MINDSTORMS Education EV3 могут автоматически определять какое устройство подключено в каждый порт. Таким образом вы можете быстро обнаружить ошибку при соединении моторов и сенсоров.

Цепное последовательное соединение по USB Вы можете соединить до 4 микрокомпьютеров EV3 через USB порты. При помощи данной функции можно программировать все 4 микрокомпьютера как одно целое, получая большое количество портов. Это позволит создавать действительно огромные конструкции.

Поддержка Bluetooth и Wi-Fi Вы можете соединяться с 7-ю EV3 Микроконтроллерами через Bluetooth. Также возможно соединение через Wi-Fi, присоединив Wi-Fi ключ к USB порту. Стало доступным соединение с iPhone и Android.

LEGO Mindstorms EV3 выпускается в двух версиях Education и Home (Retail). And please note the differences below. Education version is including battery for the use of long time experiment or repeating usage. And data logging is also only for Education version. And more, LEGO MINDSTORMS Education EV3 Software includes many movies and materials for the use of teachers in classes. Более подробно о различиях можно прочитать в статье Различия между LEGO MINDSTORMS EV3 Education и Home версиями.

Запрограммировать робота можно как при помощи специального программного обеспечения на компьютере (LEGO Education software), так и при помощи микроконтроллера EV3. Программирование на компьютере более удобно и понятно, т.к. используется наглядный графический интерфейс для облегчения восприятия программ.

Снять измерения с датчиков можно двумя путями: при помощи программного обеспечения EV3 Software и непосредственно с микроконтроллера EV3. Данные могут быть представлены в виде графиков

5.10.5. Нечеткие множества и нечеткая логика

В основе нечеткой логики лежит теория нечетких множеств, изложенная в серии работ Л. Заде в 1965-1973 годах. Математическая теория нечетких множеств (fuzzy sets) и нечеткая логика (fuzzy logic) являются обобщениями классической теории множеств и классической формальной логики. Основной причиной появления новой теории стало наличие нечетких и приближенных рассуждений при описании человеком процессов, систем, объектов.

Л. Заде, формулируя это главное свойство нечетких множеств, базировался на трудах предшественников. В начале 1920-х годов польский математик Лукашевич трудился над принципами многозначной математической логики, в которой значениями предикатов могли быть не только «истина» или «ложь». В 1937 году еще один американский ученый М. Блэк впервые применил многозначную логику Лукашевича к спискам как множествам объектов и назвал такие множества неопределенными.

Прежде чем нечеткий подход к моделированию сложных систем получил признание во всем мире, с момента зарождения теории нечетких множеств прошло не одно десятилетие.

Нечеткая логика как научное направление развивалась непросто, не избежала она и обвинений в лженаучности. Даже в 1989 году, когда примеры успешного применения нечеткой логики в обороне, промышленности и бизнесе исчислялись десятками, Национальное научное общество США обсуждало вопрос об исключении материалов по нечетким множествам из институтских учебников.

Первый период развития нечетких систем (конец 60-х – начало 70-х гг.) характеризуется развитием теоретического аппарата нечетких множеств. В 1970 году Беллман совместно с Заде разработали теорию принятия решений в нечетких условиях.

В 70-80 годы (второй период) появляются первые практические результаты в области нечеткого управления сложными техническими системами (парогенератор с нечетким управлением). И. Мамдани в 1975 году спроектировал первый функционирующий на основе алгебры Заде контроллер, управляющий паровой турбиной. Одновременно стало уделяться внимание вопросам создания экспертных систем, построенных на нечеткой логике, разработке нечетких контроллеров. Нечеткие экспертные системы для поддержки принятия решений нашли широкое применение в медицине и экономике.

Наконец, в третьем периоде, который длится с конца 80-х годов и продолжается в настоящее время, появляются пакеты программ для построения нечетких экспертных систем, а области применения нечеткой логики заметно расширяются. Она применяется в автомобильной, аэрокосмической и транспортной промышленности, в области изделий бытовой техники, в сфере финансов, анализа и принятия управленческих решений и многих других. Кроме того, немалую роль в развитии нечеткой логики сыграло доказательство знаменитой теоремы FAT (Fuzzy Approximation Theorem) Б. Коско, в которой утверждалось, что любую математическую систему можно аппроксимировать системой на основе нечеткой логики.

Одним из самых впечатляющих результатов стало создание управляющего микропроцессора на основе нечеткой логики, способного автоматически решать известную задачу «о собаке, догоняющей кота». В 1990 году Комитет по контролю экспорта США внес нечеткую логику в список критически важных оборонных технологий, не подлежащих экспорту потенциальному противнику.

В бизнесе и финансах нечеткая логика получила признание после того, как в 1988 году экспертная система на основе нечетких правил для прогнозирования финансовых индикаторов единственная предсказала биржевой крах. И количество успешных фаззи-применений в настоящее время исчисляется тысячами.

В Японии это направление переживает настоящий бум. Здесь функционирует специально созданная организация Laboratory for

International Fuzzy Engineering Research. Программой этой организации является создание более близких человеку вычислительных устройств.

Информационные системы, базирующиеся на нечетких множествах и нечеткой логике, называют нечеткими системами.

Достоинства нечетких систем:

- функционирование в условиях неопределенности;
- оперирование качественными и количественными данными;
- использование экспертных знаний в управлении;
- построение моделей приближенных рассуждений человека;
- устойчивость при действии на систему всевозможных возмущений.

Недостатками нечетких систем являются:

- отсутствие стандартной методики конструирования нечетких систем;
- невозможность математического анализа нечетких систем существующими методами;
- применение нечеткого подхода по сравнению с вероятностным не приводит к повышению точности вычислений.

5.10.6. Нечеткие множества

Главное отличие теории нечетких множеств от классической теории четких множеств состоит в том, что если для четких множеств результатом вычисления характеристической функции могут быть только два значения – 0 или 1, то для нечетких множеств это количество бесконечно, но ограничено диапазоном от нуля до единицы.

Нечеткое множество

Пусть U – так называемое универсальное множество, из элементов которого образованы все остальные множества, рассматриваемые в данном классе задач, например множество всех целых чисел, множество всех гладких функций и т.д. Характеристическая функция множества $A \subseteq U$ – это функция μA , значения которой указывают, является ли $x \in U$ элементом множества A:

$$\mu A(x) = \begin{cases} 1, & x \in A; \\ 0, & x \notin A. \end{cases}$$

В теории *нечетких множеств* характеристическая функция называется **функцией принадлежности**, а ее значение $\mu A(x)$ – **степенью принадлежности** элемента x нечеткому множеству A.

Более строго: **нечетким множеством** A называется совокупность пар

$$A = \left\{ \langle x, \mu_A(x) \rangle \middle| x \in U \right\},$$

где μ_A – функция принадлежности, то есть $\mu_A : U \rightarrow [0,1]$.
Пусть, например,:

U ={a, b, c, d, e},
A = { a, 0, b, 0.1 , c, 0.5 , d, 0.9, e, 1} .

Тогда элемент a не принадлежит множеству A, элемент b принадлежит ему в малой степени, элемент c более или менее принадлежит, элемент d принадлежит в значительной степени, e является элементом множества A.

Пример. Пусть универсум U есть множество действительных чисел. Нечеткое множество A, обозначающее множество чисел, близких к 10, можно задать следующей функцией принадлежности (рисунок 3.3):

$\alpha \in [0,1]$.
$\mu_A(x) = (1 + x - 10^m)^{-1}$,
где $m \in N$.

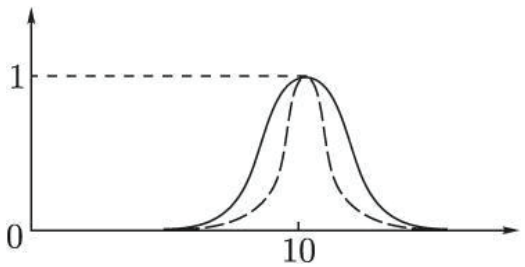


Рисунок 5.22 – Функция принадлежности μA

Показатель степени m выбирается в зависимости от степени близости к 10. Например, для описания множества чисел, очень близких к 10, можно положить m = 4, для множества чисел, не очень далеких от 10, m = 1.

Ядром нечеткого множества A называется четкое множество A таких точек в U, для которых величина $\mu A(x) = 1$.

Множеством уровня α (α *-срезом*) нечеткого множества A называется четкое подмножество универсального множества U, определяемое по формуле $A\alpha = \{x \mu_A(x) \geq \alpha\}$, где

Функцию принадлежности называют нормальной, если ядро нечеткого множества содержит хотя бы один элемент.

Операции над нечеткими множествами

Для нечетких множеств, как и для обычных, определены основные операции: объединение, пересечение и инверсия/дополнение.

Для определения пересечения и объединения нечетких множеств наибольшей популярностью пользуются следующие три группы операций:

Максиминные $\mu_{A \grave{E} B}(x) = \max\{\mu_A(x), \mu_B(x)\}$,
$\mu_{A \c{C} B}(x) = \min\{\mu_A(x), \mu_B(x)\}$

Алгебраические $\mu_{A \grave{E} B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \times \mu_B(x)$,
$\mu_{A \c{C} B}(x) = \mu_A(x) \times \mu_B(x)$

Ограниченные $\mu_{A \grave{E} B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\}$,
$\mu_{A \c{C} B}(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\}$

Дополнение нечеткого множества во всех трех случаях определяется одинаково:

$\mu_A(x) = 1 - \mu_A(x)$.

Пример. Пусть А – нечеткое множество «от 5 до 8» и В – нечеткое множество «около 4», заданные своими функциями принадлежности (рисунок 5.23).

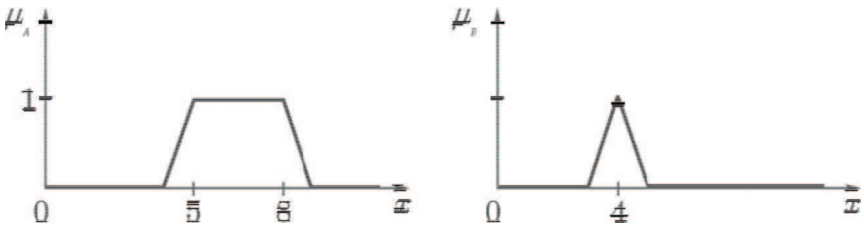


Рисунок 5.23 – Функции принадлежности нечетких множеств А и В

Тогда, используя максиминные операции, мы получим следующие множества, изображенные на рисунке 5.24.

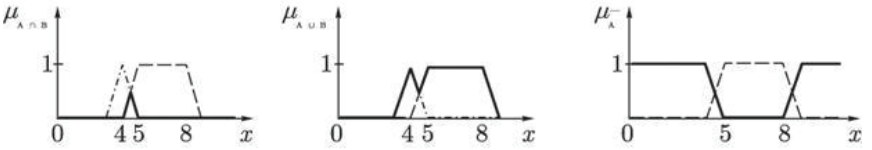


Рисунок 5.24 – Функции принадлежности нечетких множеств, полученных из A и B

При максиминном и алгебраическом определении операций не будут выполняться законы противоречия и исключения третьего:

A Ç A ¹ 0, A È A ¹ U, в случае ограниченных операций не будут выполняться свойства идемпотентности и дистрибутивности:

A È A ¹ A, A Ç A ¹ A ,

A È (B Ç C) ¹ (A Ç B) È (A Ç C), A Ç (B È C) ¹ (A È B) Ç (A È C).

Можно показать, что при любом построении операций объединения и пересечения в теории нечетких множеств приходится отбрасывать либо законы противоречия и исключения третьего, либо законы идемпотентности и дистрибутивности.

5.10.7. Нечеткая логика

Понятие нечеткой и лингвистической переменных используется при описании объектов и явлений с помощью нечетких множеств.

Нечеткая переменная характеризуется тройкой <α, X, A>, где α

– наименование переменной, X – универсальное множество (область определения α), A – нечеткое множество на X, описывающее ограничения (то есть μ A(x)) на значения нечеткой переменной α.

Лингвистической переменной называется набор <β, T, X, G, M>, где β – наименование лингвистической переменной, T – множество ее значений (терм-множество), представляющих собой наименования нечетких переменных, областью определения каждой из которых является множество X (множество T называется базовым терм-множеством лингвистической переменной), G – синтаксическая процедура, позволяющая оперировать элементами терм-множества T, в частности генерировать новые термы (значения), M – семантическая процедура, позволяющая превратить каждое новое значение лингвистической

переменной, образуемое процедурой G, в нечеткую переменную, то есть сформировать соответствующее нечеткое множество.

Лингвистическую переменную можно определить как переменную, значениями которой являются не числа, а слова или предложения естественного (или формального) языка. Например, лингвистическая переменная «возраст» может принимать следующие значения: «очень молодой», «молодой», «среднего возраста», «старый», «очень старый» и др. Ясно, что переменная «возраст» будет обычной переменной, если ее значения – точные числа; лингвистической она становится, будучи использованной в нечетких рассуждениях человека.

Каждому значению лингвистической переменной соответствует определенное нечеткое множество со своей функцией принадлежности. Так, лингвистическому значению «молодой» может соответствовать функция принадлежности, изображенная на рисунке 5.25.

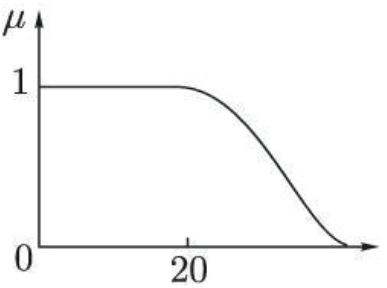


Рисунок 5.25 – Функция принадлежности значения «молодой» лингвистической переменной «возраст»

Пример. Пусть эксперт определяет толщину выпускаемого изделия с помощью понятий «малая толщина», «средняя толщина» и «большая толщина», при этом минимальная толщина равна 10 мм, а максимальная – 80 мм (рисунок 5.26).

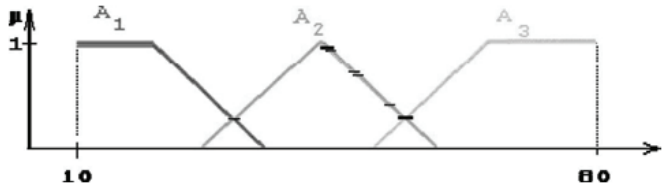


Рисунок 5.26 – Функции принадлежности нечетких множеств: «малая толщина» = A1, «средняя толщина» = A2, «большая толщина» = A3

Формализация такого описания может быть проведена с помощью следующей лингвистической переменной <β, T, X, G, M>, где

β – толщина изделия;

T – {«малая толщина», «средняя толщина», «большая толщина»};

X = [10, 80];

G – процедура образования новых термов с помощью связок и, или и модификаторов типа очень, не, слегка и др. Например: «малая или средняя толщина» (рисунок 3.8), «очень малая толщина» и др.;

M – процедура задания на X = [10, 80] нечетких подмножеств A1 = «малая толщина», A2 = «средняя толщина», A3 = «большая толщина», а также нечетких множеств для термов из G(T) в соответствии с правилами трансляции нечетких связок и модификаторов и, или, не, очень, слегка и др.

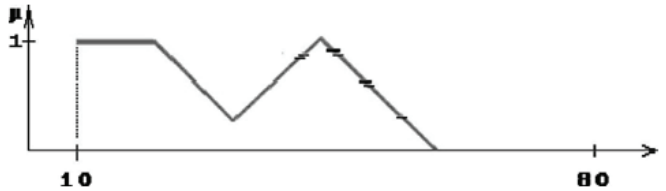


Рисунок 5.27 – Функция принадлежности нечеткого множества «малая или средняя толщина» = A1∪A1

Наряду с рассмотренными выше базовыми значениями лингвистической переменной «толщина» (T = {«малая толщина», «средняя толщина», «большая толщина»}) возможны значения, зависящие от области определения X. В данном случае значения лингвистической переменной «толщина изделия» могут быть определены как «около 20 мм», «около 50 мм», «около 70 мм», то есть в виде нечетких чисел.

Нечеткими высказываниями будем называть высказывания следующего вида:

Высказывание <β есть β'>, где β – наименование лингвистической переменной, β' – ее значение, которому соответствует нечеткое множество на универсальном множестве X.

Например, высказывание <давление большое> предполагает, что лингвистической переменной «давление» придается значение «большое», для которого на универсальном множестве X переменной «давление»

определено соответствующее данному значению «большое» нечеткое множество.

Высказывание <β есть mβ'>, где m – модификатор, которому соответствуют слова «очень», «более или менее», «много больше» и др.

Например: <давление очень большое>, <скорость много больше средней> и др.

Составные высказывания, образованные из высказываний видов 1 и 2 и союзов и; или; если… то...; если… то… иначе...

Основным правилом вывода в традиционной логике является правило modus ponens, согласно которому мы судим об истинности высказывания B по истинности высказываний A и A → B.

Например, если A – высказывание <Иван в больнице>, B – высказывание <Иван болен>, то если истинны высказывания <Иван в больнице> и <Если Иван в больнице, то он болен>, то истинно и высказывание <Иван болен>.

Во многих привычных рассуждениях, однако, правило modus ponens используется не в точной, а в приближенной форме. Так, обычно мы знаем, что A истинно и что $A^* \rightarrow B$, где A^* есть в некотором смысле приближение A. Тогда из $A^* \rightarrow B$ мы можем сделать вывод о том, что B приближенно истинно.

Нечеткая импликация выражается в следующем виде:

$$A \rightarrow B = A \cup B \text{ и } \mu_{A \rightarrow B}(x) = \max\{1 - \mu_A(x), \mu_B(x)\}.$$

Основой для проведения операции нечеткого логического вывода является база правил, содержащая нечеткие высказывания в форме если… то… и функции принадлежности для соответствующих лингвистических термов. При этом должны соблюдаться следующие условия:

- существует хотя бы одно правило для каждого лингвистического терма выходной переменной;
- для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки (левая часть правила).

В противном случае имеет место неполная база нечетких правил.

Для реализации логического вывода необходимо выполнить следующее:

Сопоставить факты с каждым из правил и определить степень соответствия, назначив текущую силой правил.

Для каждого правила, сила которого больше заданного порога, вычислить достоверность левой части.

Для каждого правила с помощью оператора импликации вычислить достоверность правой части.

Для многих результатов, полученных по различным правилам, выбрать одно (усредненное).

Пример. Пусть есть некоторая система, например, реактор, описываемая тремя параметрами: температура, давление и расход рабочего вещества. Все показатели измеримы, и множество возможных значений известно. Также из опыта работы с системой известны некоторые правила, связывающие значения этих параметров. Предположим, что сломался датчик, измеряющий значение одного из параметров системы, но знать его показания необходимо хотя бы приблизительно. Тогда встает задача об отыскании этого неизвестного значения (пусть это будет давление) при известных показателях двух других параметров (температуры и расхода) и связи этих величин в виде следующих правил:

- если Температура низкая и Расход малый, то Давление низкое;
- если Температура средняя, то Давление среднее;
- если Температура высокая или Расход большой, то Давление высокое.

Температура, Давление и Расход – лингвистические переменные.

Температура. Универсум (множество возможных значений) – отрезок [0, 150]. Начальное множество термов {Высокая, Средняя, Низкая}. Функции принадлежности термов имеют следующий вид (рисунок 5.28).

Рисунок 5.28 – Функции принадлежности термов лингвистической переменной Температура

Давление. Универсум – отрезок [0, 100]. Начальное множество термов {*Высокое, Среднее, Низкое*}. Функции принадлежности термов имеют следующий вид (рисунок 5.29).

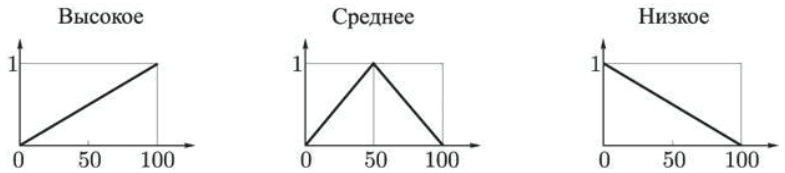


Рисунок 5.29 – Функции принадлежности термов лингвистической переменной Давление

Расход. Универсум – отрезок [0, 8]. Начальное множество термов {*Большой, Средний, Малый*}. Функции принадлежности термов имеют следующий вид (рисунок 5.30).

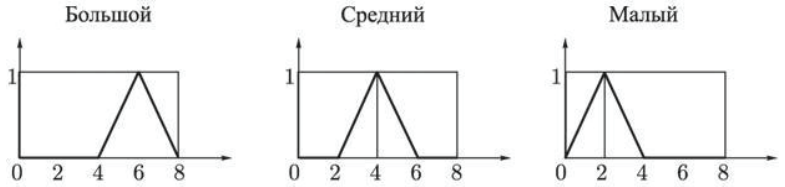


Рисунок 5.30 – Функции принадлежности термов лингвистической переменной Расход

Пусть известны значения: *Температура* – 85 и *Расход* – 3.5.

Произведем расчет значения давления.

Этап фаззификации (переход от заданных четких значений к степеням уверенности). По заданным значениям входных параметров найдем степени уверенности простейших утверждений:

Температура Высокая – 0.7; Расход Большой – 0;

Температура Средняя – 1; Расход Средний – 0.75;

Температура Низкая – 0.3; Расход Малый – 0.25.

Затем вычислим степени уверенности посылок правил:

Температура Низкая и Расход Малый: min (Темп. Низкая, Расход Малый)= min (0.3; 0.25)=0.25;

Температура Средняя: 1;

Температура Высокая или Расход Большой: max (Темп. Высокая, Расход Большой)= max (0.7; 0)=0.7.

Каждое из правил представляет собой нечеткую импликацию. Степень уверенности посылки мы вычислили, а степень уверенности заключения задается функцией принадлежности соответствующего терма. Поэтому, используя один из способов построения нечеткой импликации, мы получим новую нечеткую переменную, соответствующую степени уверенности в значении выходных данных при применении к заданным входным

соответствующего правила. Используя определение нечеткой импликации как минимума левой и правой частей (определение Мамдани), имеем (рисунок 5.31).

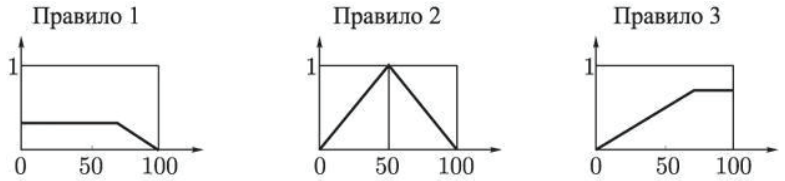


Рисунок 5.31 – Результат этапа фаззификации

Этап аккумуляции (объединение результатов применения всех правил). Один из основных способов аккумуляции – построение максимума полученных функций принадлежности (объединение функций принадлежности, полученных на этапе фаззификации).

Полученную функцию принадлежности уже можно считать результатом. Это новый терм выходной переменной Давление. Его функция принадлежности говорит о степени уверенности в значении давления при заданных значениях входных параметров и использовании правил, определяющих соотношение входных и выходных переменных. Но обычно все-таки необходимо какое-то конкретное числовое значение (рисунок 5.32).

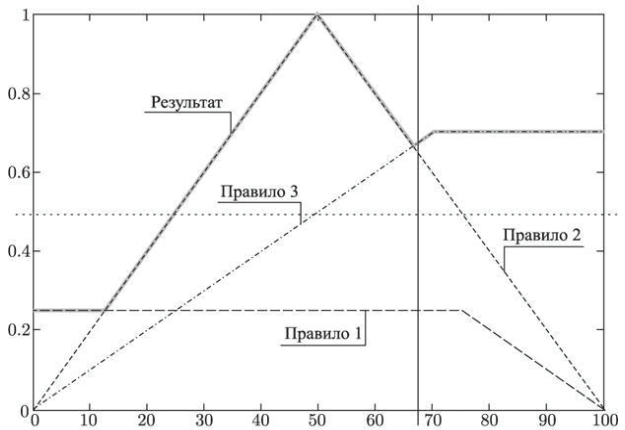


Рисунок 5.32 – Результат этапа аккумуляции

Этап дефаззификации (получение конкретного значения из универса по заданной на нем функции принадлежности). Существует множество методов дефаззификации, но в этом случае достаточно метода первого максимума. Применяя его к полученной функции принадлежности, получаем, что значение давления – 50. Таким образом, если мы знаем, что температура равна 85, а расход рабочего вещества – 3.5, то можем сделать вывод, что давление в реакторе равно примерно 50.

5.10.8. Нанороботы

Нанороботы – роботы, размером сопоставимые с молекулой (менее 10 нм), обладающие функциями движения, обработки и передачи информации, исполнения программ.

Нанороботы, способные к созданию своих копий, то есть самовоспроизводству, называются репликаторами. Возможность создания нанороботов рассмотрел в своей книге «Машины создания» американский учёный Эрик Дрекслер.

5.10.9. Роль нанотехнологий при разработке интеллектуальных роботов

Наиболее интересными исследованиями за последних два года, стали открытия в сфере искусственного интеллекта. Еще в 2010 году ученые, достигли серьезного прогресса в создании «искусственного мозга».

Любители научной фантастики не раз встречали в романах описание искусственного мозга, но не все знают, что ученые, работающие в области нейроморфной инженерии уже сейчас добились определенного прогресса в своих изысканиях. Главным в исследованиях, посвященных созданию искусственной нервной ткани, имитирующей ткань мозга, ученые считают разработку искусственных синапсов.

По мнению ученых, выполнять их функцию могут мемристоры (рисунок 5.33) – пассивные элементы в микроэлектронике, способные изменять свое сопротивление в зависимости от текущего по нему тока. Мозг живого существа представляет сеть клеток-нейронов, соединенных отростками-аксонами через контакты-синапсы. Говоря упрощенно, в процессе обучения некоторые синапсы формируют определенную сеть, в то время как другие остаются неактивными и не проводят сигналы.

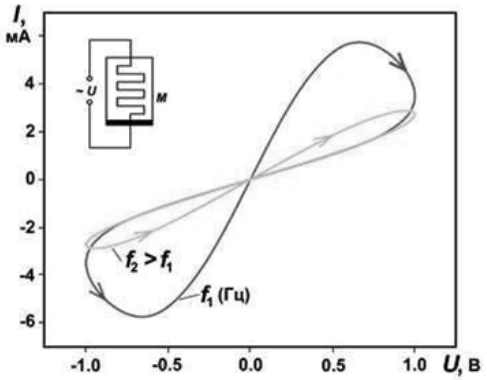


Рисунок 5.33 – Зависимость тока на мемристоре от переменного напряжения

Мемристор может выполнять ту же функцию – он может уже в процессе работы стать каналом или сопротивлением под воздействием внешнего стимула. Ученые пошли дальше рассуждений и создали небольшую сеть из наноразмерных кремниевых мемристоров, которые представляют две расположенные одна над другой сетки проводов в аморфном серебряном слое, причем содержание серебра при одном слое выше, чем при другом. При подаче напряжения ионы серебра начинают течь к нижнему слою и увеличивают электропроводность. Таким образом, ученые получили возможность полностью контролировать основной параметр мемристора и, соответственно, сроить на их основе сеть, которая сможет имитировать работу живого мозга.

В то же время следует отметить два важных аспекта, отличающих работу мозга от работы микросхемы. Во-первых, компьютеры могут выполнять вычисления с фантастической скоростью - в настоящее время самые мощные машины могут выполнять до 1000 триллионов операций в секунду, однако за единицу времени может выполняться только одна операция. Нейроны мозга выполняют лишь 1000 операций в секунду, однако мозг способен просчитывать миллионы операций одновременно и выполнять многие задачи быстрее и эффективней компьютера. Во-вторых, нейронная сеть постоянно модифицируется, а компьютер не способен к этому.

Над созданием искусственной самообучающейся, то есть в нашем случае – меняющей схему своих цепей структуры бьются ученые всего мира. Весной этого года исследователи из Японии и США выступили с докладом, в котором рассказали о своих успехах в области создания органической молекулярной схемы, способной к саморазвитию. По словам ученых,

созданный на основе их разработки компьютер может обрабатывать не одну, а свыше трехсот операций одновременно. Такой компьютер действует по совершенно новым алгоритмам и может решать сложные задачи, связанные с прогнозированием различных процессов.

Производители микросхем активно осваивают технологию 22 нм, а ученые в лабораториях заявляют, что при таких размерах элементы схем можно интегрировать в живую клетку. Средние размеры клетки человека – около 10 кв. микронов, так что в нее теоретически можно встроить до 2000 транзисторов, а это сопоставимо с параметрами процессоров первых персональных компьютеров. Ученые из Барселоны впервые собрали кремниевый процессор внутри отдельной клетки, используя различные технологии – липофекцию, фагоцитоз, микроинъекции. После операции клетки, в которые поместили процессоры, оставались живыми и, что наиболее важно, ученые смогли запустить процессор и использовать его, как датчик процессов, происходящих внутри клетки. Таким образом, в наших руках оказывается ключ к созданию электронной системы диагностики всех происходящих в живом организме процессов.

Университеты всего мира пишут, что их ученым удалось сделать очередной шаг на пути к разработке технологии массового производства наноустройств. Однако очевидно, что между современными лабораторными технологиями создания сложных трехмерных наноструктур и коммерциализацией лежит пропасть. В настоящее время в основном применяется метод оптической литографии, однако этот процесс сложен и имеет предельный барьер масштаба – 10 нм.

В июне 2010 года сингапурские ученые продемонстрировали совершенно новый метод создания полевых транзисторов и отдельных нанопроводов: электронный луч сканирует основу и наносит на нее металл или изолятор, создавая электронный элемент. Этот метод позволяет создавать наноустройства более точно, чем метод литографии и при этом он значительно быстрей и дешевле, что позволяет предположить, что именно он может быть взят на вооружение будущими производителями наноэлектроники.

Нанотехнологии нашли применение в такой интересной области исследований как **искусственный фотосинтез**. Целью ученых является создание материала, который бы использовал энергию света для того, чтобы разлагать воду на кислород и водород, так как последний можно использовать в качестве топлива. Ученые из университета Калифорнии и университета Сага в Японии разработали такой материал, имитирующий структуру листа. Листки материала, в основном состоящего из оксида титана копируют все поры, выступы и канальцы, которые можно видеть на живом листе под микроскопом, а вместо хлорофилла применяется искусственный катализатор. Опыты показали, что такой «искусственный лист» достаточно эффективно разлагает воду на кислород и водород. Перспективы такого направления трудно переоценить, ведь в наших руках

может оказаться ключ к почти безграничным запасам экологически чистого топлива.

Группе ученых из США и Японии впервые удалось создать **искусственные антитела**, успешно функционирующие в живом организме. Созданные из синтезированного полимера с помощью молекулярного импринтинга, наночастицы имеют такой же, как и у настоящих антител, участок на поверхности – рецептор антигена, который настроен на соединение с определенным белком. Таким образом антитела могут отфильтровывать токсические вещества из крови. Чтобы испытать их, ученые вводили раствор этих антител в кровеносную систему мышей, которые ранее получили смертельную дозу вещества меллитина. В результате спасительной инъекции многие мыши выжили, в то время как тестовая группа мышей, не получивших укола с искусственными антителами погибла целиком. В ходе этого опыта также выяснилось, что сами искусственные антитела не распознаются организмом, как инородные и не вызывают иммунного ответа. Это открытие, возможно, станет толчком к разработке препаратов, многократно усиливающих иммунную реакцию организма, а также уникальных систем очистки крови от токсинов.

В качестве последней разработки, приковавшей внимание хочется отметить создание **устройства, преобразующего механическую энергию тела, и акустические колебания в электричество** для подпитки мобильных устройств. В основе нового элемента лежат пьезоэлектрические волокна состоящие из органических нановолокон цирконат-титаната свинца, помещенных на пластинах из силиконового каучука. При сгибании или скручивании пластин вырабатывается электричество, превращая механическую энергию в электрическую. Применение данной технологии безгранично –обувь, вырабатывающая энергию при ходьбе для питания гаджетов, микрохирургические устройства, которые заряжаются от движений, и даже электрокардиостимуляторы, в которых используется новый материал вместо традиционных батарей – вот лишь несколько примеров использования нового наноустройства.

Наибольшим вниманием у ученых пользовались изыскания в области разработки искусственного мозга, компьютерной системы нового типа, имитирующей живую нейронную сеть. В то же время очевидными являются усилия многих лабораторий приблизить теоретические разработки к массовому производству, коммерциализации новых проектов.

5.10.10. Обзор научно-исследовательских разработок по нанороботам

В настоящее время нанороботы находятся в научно-исследовательской стадии создания. Некоторыми учёными утверждается, что уже созданы некоторые компоненты нанороботов. Разработке компонентов наноустройств и непосредственно нанороботам посвящен ряд международных научных конференций.

Уже созданы некоторые примитивные прототипы молекулярных машин. Например, датчик, имеющий переключатель около 1,5 нм, способный вести подсчет отдельных молекул в химических образцах. Недавно университет Райса продемонстрировал наноустройства для использования их в регулировании химических процессов в современных автомобилях.

Одним из самых сложных прототипов нанобота является «DNA box», созданный в конце 2008 года международной группой под руководством Йоргена Кьемса. Устройство имеет подвижную часть, управляемую с помощью добавления в среду специфических фрагментов ДНК. По мнению Кьемса, устройство может работать как «ДНК-компьютер», т.к на его базе возможна реализация логических вентилей. Важной особенностью устройства является метод его сборки, так называемый ДНК оригами (англ.), благодаря которому устройство собирается в автоматическом режиме.

В 2010 году были впервые продемонстрированы нанороботы на основе ДНК, способные перемещаться в пространстве.

Так как нанороботы имеют микроскопические размеры, то их, вероятно, потребуется очень много для совместной работы в решении микроскопических и макроскопических задач. Рассматривают стаи нанороботов, которые не способны к репликации (т. н. «сервисный туман») и которые способны к самостоятельной репликации в окружающей среде («серая слизь» и др. варианты).

Некоторые сторонники нанороботов в ответ на сценарий «серой слизи» высказывают мнение о том, что нанороботы способны к репликации только в ограниченном количестве и в определенном пространстве нанозавода. Кроме того, ещё только предстоит разработать процесс саморепликации, который сделает данную нанотехнологию безопасной. Кроме того, свободная саморепликация роботов является гипотетическим процессом и даже не рассматривается в текущих планах научных исследований.

Однако, имеются планы по созданию медицинских нанороботов, которые будут впрыскиваться в пациента и выполнять роль беспроводной связи на наноуровне. Такие нанороботы не могут быть получены в ходе самостоятельного копирования, так как это вероятно приведет к появлению ошибок при копировании, которые могут снизить надежность наноустройства и изменить выполнение медицинских задач. Вместо этого нанороботов планируется изготавливать на специализированных медицинских нанофабриках.

Первое полезное применение наномашин, если они появятся, планируется в медицинских технологиях, где они могут быть использованы для выявления и уничтожения раковых клеток. Также они могут обнаруживать токсичные химические вещества в окружающей среде и измерять уровень их концентрации.

Возможные области применения нанороботов:

- ранняя диагностика рака и целенаправленная доставка лекарств в раковые клетки;
- биомедицинский инструментарий;
- хирургия;
- фармакокинетика;
- мониторинг больных диабетом;
- производство посредством молекулярной сборки нанороботами устройства из отдельных молекул по его чертежам;
- военное применение в качестве средств наблюдения и шпионажа, а также в качестве оружия. Потенциальные возможности использования нанороботов в качестве оружия демонстрируются в некоторых фантастических произведениях (Терминатор 2: Судный день, День, когда остановилась Земля (фильм, 2008), Бросок кобры);
- космические исследования и разработки (например, зонды фон Неймана).

Попытаемся описать устройство медицинского нанороботa общего применения. Ниже остановимся на описании основных систем нанороботa и его предполагаемом устройстве.

Так как основная функция нанороботa – передвижение по кровеносной системе человека, то он должен иметь мощную навигационную систему.

Устройству необходимо иметь несколько типов различных сенсоров для мониторинга окружающей среды, навигации, коммуникации и работы с отдельными молекулами.

Также нанороботу необходима мощная транспортная система, доставляющая отдельные атомы и молекулы от хранилищ к наноманипуляторам, и обратно.

Для работы с пораженными структурами устройство будет оборудовано набором телескопических наноманипуляторов разного применения.

Материал, из которого будет изготовлен наноробот - алмазоид или сапфироид. Это обеспечит биосовместимость человека и большого количества наномашин.

Также необходимо наличие приемо - передаточных устройств, позволяющих нанороботам связываться друг с другом.

И, наконец, для удержания крупных объектов необходимы телескопические захваты.

На основании выдвинутых требований я постарался построить модель медицинского нанороботa общего применения. В идеальном случае, это устройство будет способно "ремонтировать" поврежденные клетки, ткани; производить диагностику и лечение раковых заболеваний и картографировать кровеносные сосуды; производить анализ ДНК с последующей ее корректировкой; уничтожать бактерии, вирусы, и т.п. Максимальный размер устройства не должен превышать 1x1x3 микрона

(без двигательных жгутиков). Ниже на картинке представлен вид нанороботa, выполненного из алмазоида (рисунок 3.15).

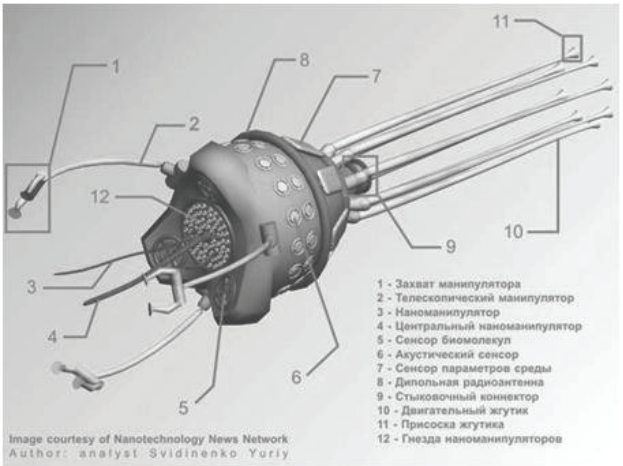


Рисунок 5.34 – Медицинский нанороботобщего применения из алмаза

Электромагнитные волны, которые смогут распространяться в теле человека не затухая, будут по длине волны сравнимы с нанороботом. Поэтому приемно-передающие антенны будут иметь вид диполей, выступающих за пределы корпуса. Наноманипуляторы, механические захваты и жгутики должны быть телескопическими, и при необходимости должны складываться в корпус робота для того, чтобы робот смог лучше передвигаться в кровеносном русле. Иммунная система, в основном, реагирует на «чужеродные» поверхности. Размер нанороботатакже играет важную роль при этом, так же как и мобильность устройства, шероховатость поверхности и ее подвижность. Ряд проделанных экспериментов подтвердил, что гладкие алмазоидные структуры вызывают меньшую активность лейкоцитов и меньше адсорбируют фибриноген. Поэтому кажется разумным надеяться, что такое алмазоидное покрытие («организованное», т.е. нанесенное атом-за-атомом, с нанометровой гладкостью), будет иметь очень низкую биологическую активность. Благодаря очень высокой поверхностной энергии алмазоидной поверхности и сильной ее гидрофобности, внешняя оболочка роботов будет полностью химически инертна. Для такого нанороботa, можно будет использовать нанокомпьютер, производящий ~106-109 операций в секунду для исполнения своей работы. Это на 4-7 порядков меньше вычислительной мощности человеческого мозга, составляющей ~1013

операций в секунду. Так что этот нанороботне будет обладать искусственным интеллектом.

Это всего лишь описательная работа. Она не основана на результатах каких-либо расчетов. Ниже мы рассмотрим отдельные подсистемы нанороботa (рисунки 5.35 – 5.37).

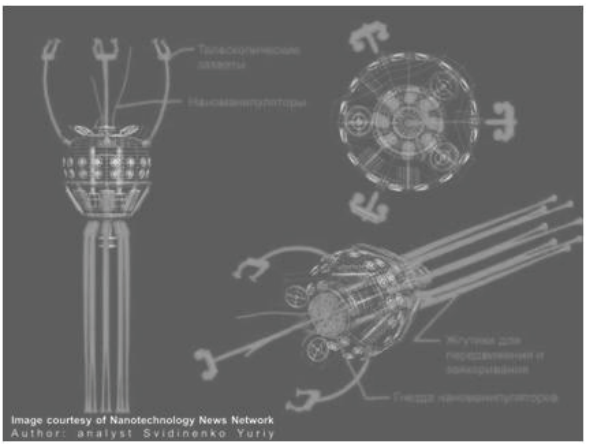


Рисунок 5.35 – Двигательная подсистема и подсистема заякоривания

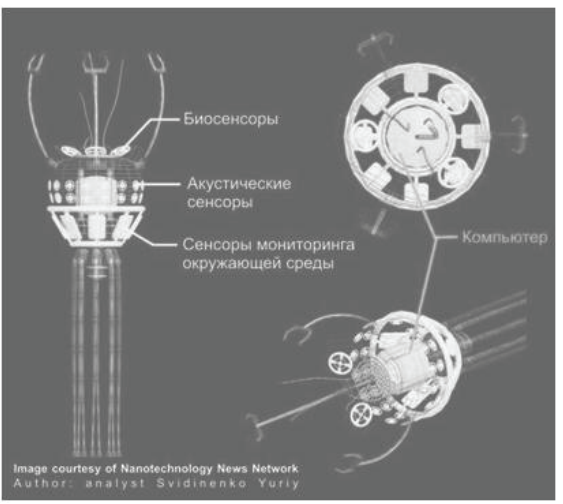


Рисунок 5.36 – Сенсорная и обрабатывающая подсистема

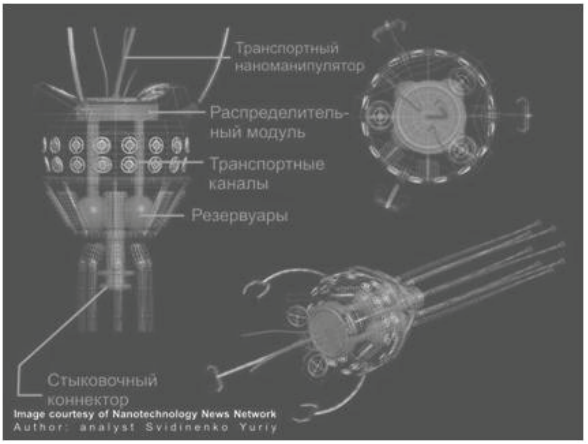


Рисунок 5.37 – Транспортная подсистема

Для работы с внутриклеточными структурами нанороботу вовсе не обязательно целиком проникать внутрь клетки (можно повредить внутриклеточный цитоскелет). Телескопические наноманипуляторы предотвратят повреждение органелл и цитоскелета. Ниже приведены рисунки, изображающие нанороботa в кровеносной системе, и нанороботa, ремонтирующего клетку in vivo (рисунки 3.19, 3.20).

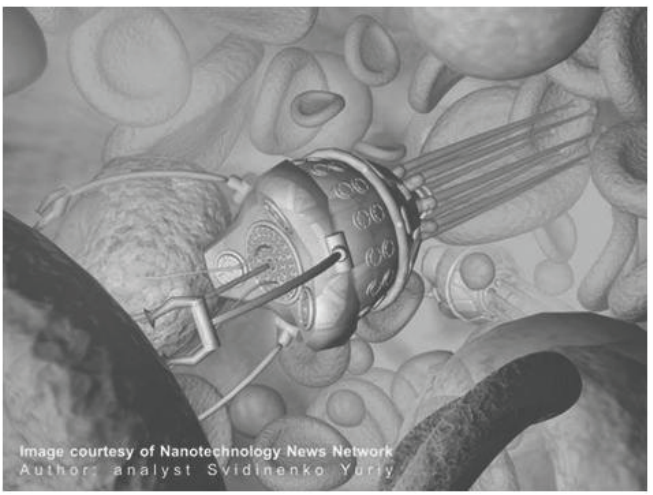


Рисунок 5.38 – Нанороботы в кровеносной системе

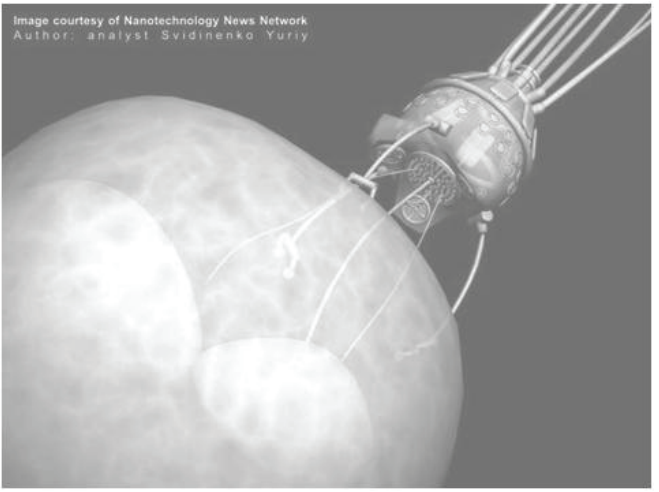


Рисунок 5.39 – Нанороботремонтирует клетку

Для связи нанороботов друг с другом, а также для формирования навигационной системы, полезно будет использовать еще один тип нанороботов - коммунноцитов, которые будут работать в виде усилительных станций.

ЗАКЛЮЧЕНИЕ

Создание систем искусственного интеллекта, сопряжено с решением множества проблем. На пути создания ИИ это и ограниченность ресурсов, и недостаточные знания в этой области, а также проблема осуществимости это сделать, и многие другие технические проблемы.

Искусственный интеллект в настоящее время остается научным направлением, связанным с компьютерным моделированием человеческих интеллектуальных функций.

Системы искусственного интеллекта обычно используются для обозначения способности вычислительной системы выполнять задачи, свойственные интеллекту человека, например, задачи логического вывода и обучения.

Любая задача, алгоритм решения которой заранее не известен или же данные неполные может быть отнесена к задачам области ИИ.

Системы, программы, выполняющие действия по решению задачи можно отнести к ИИ, если результат их деятельности аналогичен результату человека при решении той же задачи. Поэтому к ИИ можно отнести целый ряд программных средств: системы распознавания текста, автоматизированного проектирования, самообучающиеся программы и др. Но не только поэтому, а еще и потому, что они работают по сходным принципам с человеком.

Есть два основных перспективных направления в исследовании ИИ. Первое заключается в приближении систем ИИ к принципам человеческого мышления. Второе заключается в создании ИИ, представляющего интеграцию уже созданных систем ИИ в единую систему, способную решать проблемы человечества.

ВОПРОСЫ К ГЛАВЕ 5

1. Определение интеллектуальной информационной системы.
2. Классификация интеллектуальных систем.
3. Определение информационной технологии.
4. Архитектура интеллектуалных систем.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Суркова Н.Е.* Методы проектирования информационных систем [Текст] / Н.Е. Суркова, А.В. Остроух. – М.: РосНОУ, 2004. – 144 с. – ISBN 5-89789-021-8.

2. *Остроух А.В.* Основы построения систем искусственного интеллекта для промышленных и строительных предприятий [Текст]: монография / А.В. Остроух. – М.: ООО «Техполиграфцентр», 2008. – 280 с. – ISBN 978-5-94385-033-2.

3. *Остроух А.В.* Автоматизация транспортировки продукции [Текст] / А.В. Остроух, Н.Г. Куфтинова. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2011. – 146 p. – ISBN 978-3-8454-1089-0.

4. *Остроух А.В.* Рефакторинг баз данных. Автоматизация технологических процессов рефакторинга баз данных промышленных предприятий [Текст] / А.В. Остроух, Д.А. Пшеничный, О.Б. Рогова. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2013. – 133 p. – ISBN 978-3-659-38753-1.

5. *Остроух А.В.* Автоматизация управления производством. Повышение эффективности автоматизированных аналитических систем предприятий автомобильной промышленности [Текст] / А.В. Остроух, Э.А. Чернов, Д.Т. Нгуен. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2013. – 285 p. – ISBN 978-3-659-34762-7.

6. *Остроух А.В.* Системы планирования перевозок. Программно-технологические решения по разработке системы планирования заданий для заказных пассажирских перевозок [Текст] / А.В. Остроух, А.Б. Львова, А.Р. Исмаилов. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2013. – 121 p. – ISBN 978-3-659-43619-2.

7. *Остроух А.В.* Интеллектуальные системы в науке и производстве [Текст] / А.В. Остроух, А.Б. Николаев. – Saarbrucken, Germany: Palmarium Academic Publishing, 2012. – 312 p. – ISBN 978-3-659-98006-0.

8. *Остроух А.В.* Мультимедиа-технологии [Текст] / А.В. Остроух, А.М. Васьковский, А.Б. Николаев. – Saarbrucken, Germany: Palmarium Academic Publishing, 2012. – 228 p. – ISBN 978-3-659-98030-5.

9. *Остроух А.В.* Ввод и обработка цифровой информации [Текст]: учебник для нач. проф. образования / А.В. Остроух. – М.: Издат. центр «Академия», 2012. – 288 с. – ISBN 978-5-7695-9457-1.

10. *Остроух А.В.* Оперативный контроль транспортно-экспедиционной деятельности. Процессный подход к агрегированию системы показателей деятельности транспортно-экспедиционного предприятия [Текст] / А.В. Остроух, А.М. Ивахненко, Н.А. Крупенский. – Saarbrucken, Germany: Palmarium Academic Publishing, 2013. – 221 p. – ISBN 978-3-659-98329-0.

11. *Николаев А.Б.* Информационные технологии в менеджменте и транспортной логистике [Текст]: учебное пособие / А.Б. Николаев, А.В. Остроух. – Saint-Louis, MO, USA: Publishing House Science and Innovation Center, 2013. – 254 с. – ISBN 978-0-615-67110-9.

12. *Остроух А.В.* Системы искусственного интеллекта в промышленности, робототехнике и транспортном комплексе [Текст]: монография / А.В. Остроух. – Красноярск: Научно-инновационный центр, 2013. – 326 с. – ISBN 978-5-906314-10-9.

13. *Остроух А.В.* Основы информационных технологий [Текст]: учебник для СПО / А.В. Остроух. – М.: Издат. центр «Академия», 2014. – 208 с. – ISBN 978-5-4468-0588-4.

14. *Суркова Н.Е.* Методология структурного проектирования информационных систем [Текст]: монография / Н.Е. Суркова, А.В. Остроух. – Красноярск: Научно-инновационный центр, 2014. – 190 с. – ISBN 978-5-906314-16-1.

15. *Остроух А.В.* Проектирование системы распределенных баз данных [Текст] / А.В. Остроух, А.В. Помазанов. – Saarbrucken, Germany: Palmarium Academic Publishing, 2015. – 117 p. – ISBN 978-3-659-60041-8.

16. *Остроух А.В.* Автоматизация управления автотранспортными предприятиями. Новый подход на основе интеллектуальных мультиагентных систем [Текст] / А.В. Остроух, А.В. Воробьева, Н.Е. Суркова. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2015. – 117 p. – ISBN 978-3-659-47576-4.

17. *Суркова Н.Е.* Профессиональные информационные системы и базы данных: методические указания к лабораторным работам [Текст] / Н.Е. Суркова, А.В. Остроух, Т.И. Еремина. – Красноярск: Научно-

инновационный центр, 2015. – 49 с. – ISBN 978-5-906314-23-9. – DOI: 10.12731/asu.madi.ru/PISDB.2015.49.

18. *Чувиков Д.А.* Исследование 3D форматов хранения данных в интеллектуальных системах виртуальной реальности [Текст] / Д.А. Чувиков, В.П. Феоктистов, А.В. Остроух // Международный журнал экспериментального образования. – 2015. – № 3–3. – С. 416–420.

19. *Терентьев Д.И.* Исследование дисковых массивов RAID по параметрам надежности и быстродействия [Текст] / Д.И. Терентье, А.Б. Николаев, А.В. Остроух // Международный журнал экспериментального образования. – 2015. – № 3–3. – С. 423–427.

20. *Круглов А.М.* Актуализация сведений о данных информационной системы средствами активного словаря-справочника данных [Текст] / А.М. Круглов, А.В. Будихин, Д.А. Буров, А.В. Остроух // Научный вестник МГТУ ГА. Серия: Аэромеханика и прочность. – 2007. – № 119 (9). – С. 166–171.

21. *Николаев А.В.* Принципы организации динамических интерфейсов доступа к данным с использованием словарей-справочников данных [Текст] / А.В. Николаев, А.В. Будихин, Д.А. Буров, А.В. Остроух // Научный вестник МГТУ ГА. Серия: Аэромеханика и прочность. – 2007. – № 119 (9). – С. 172–178.

22. *Остроух А.В.* Использование словаря-справочника данных для реализации пользовательских средств обработки информации [Текст] / А.В. Остроух, А.П. Баринов, А.В. Николаев, С.А. Будихин // Приборы и системы. Управление, контроль, диагностика. – 2008. – № 3. – С. 13–17.

23. *Пшеничный Д.А.* Анализ параметров и сравнение СУБД для реализации информационного обеспечения промышленного предприятия [Текст] / Д.А. Пшеничный, А.В. Будихин, А.В. Остроух // Промышленные АСУ и контроллеры. – 2010. – № 12. – С. 7–11.

24. *Помазанов А.В.* Методика оптимизации баз данных [Текст] / А.В. Помазанов, А.И. Белоусова, А.О. Васильева, А.В. Остроух // В мире научных открытий. Серия: Проблемы науки и образования. – 2012. – № 12. – С. 49–54.

25. *Ostroukh A.V.* Development of Automated Control System for University Research Projects [Text] / A.V. Ostroukh, M.N. Krasnyanskiy, S.V. Karpushkin, A.D. Obukhov // Middle East Journal of Scientific Research. – 2014. – Vol. 20 (12). – P. 1780–1784. – DOI 10.5829/idosi.mejsr.2014.20.12.21091.

26. *Ostroukh A.* Realtime Development and Testing of Distributed Data Processing System for Industrial Company [Text] / A. Ostroukh, A. Pomazanov // Middle East Journal of Scientific Research. – 2014. – Vol. 20 (12). – P. 2184–2193. – DOI: 10.5829/idosi.mejsr.2014.20.12.21106.

27. *Ostroukh A.V.* Problems of organization and search the knowledge base in the CRM-systems [Text] / A.V. Ostroukh, A.I. Belousova, D.A. Pavlov, P.F. Yurchik // IOSR Journal of Engineering (IOSRJEN). – 2014. – Vol. 04. – Is. 02. – V 3. – P. 18–23. – DOI: 10.9790/3021-04231823. ANED: 0.4/3021-04231823.

28. *Krasnyanskiy M.N.* Automated control system for university research projects [Text] / M.N. Krasnyanskiy, S.V. Karpushkin, A.D. Obukhov, A.V. Ostroukh // International Journal of Advanced Studies (iJAS). – 2014. – Vol. 4. – Is. 1. – P. 22–26. – DOI: 10.12731/2227-930X-2014-1-4.

29. *Krasnyanskiy M.N.* Electronic Document Management Systems Structure for University Research and Education [Text] / M.N. Kraskyanskiy, A.V. Ostroukh, S.V. Karpushkin [et al.] // Journal of Engineering and Applied Sciences. – 2014. – Vol. 9. – Is. 5. – P. 182–189. – DOI: 10.3923/jeasci.2014.182.189.

30. *Помазанов А.В.* Создание и тестирование распределённой системы работы с удалёнными узлами [Текст] / А.В. Помазанов, А.В. Остроух // Автоматизация и современные технологии. – 2014. – № 7. – С. 17–23.

31. *Помазанов А.В.* Новый подход к разработке прототипа распределенной системы баз данных промышленного предприятия [Текст] / А.В. Помазанов, А.В. Остроух // Промышленные АСУ и контроллеры. – 2014. – № 9. – С. 11–20.

32. *Суркова Н.Е.* Методика обучения технологии баз данных для студентов непрофильных направлений подготовки в технических вузах [Текст] / Н.Е. Суркова, А.В. Остроух // Инженерная педагогика. – М.: МАДИ, 2015. – Вып. 17. – Т.3. – С. 146–156.

33. *Суркова Н.Е.* Электронное учебное издание «Профессиональные информационные системы и базы данных: методические указания к лабораторным работам» [Текст] / Н.Е. Суркова, А.В. Остроух, Т.И. Еремина // Хроники объединенного фонда электронных ресурсов. Наука и образование. – 2015. – № 3 (70). – С. 10.

34. *Krasnyanskiy M.N.* Model of Documents Management for Academic and Research Universities on Basis Set Theory [Text] / M.N. Krasnyanskiy, S.V. Karpushkin, A.D. Obukhov, A.V. Ostroukh // American-Eurasian Journal of Agricultural & Environmental Sciences. – 2015. – Vol. 15. – № 5. – P. 824–831. – DOI: 10.5829/idosi.aejaes.2015.15.5.12639.

35. *Ostroukh A.V.* Simulated operation of the computer network [Text] / A.V. Ostroukh, K.N. Mezencev, M.N. Krasnyanskiy, J. Punam, N.E. Surkova // SGEM2015 Conference Proceedings. – ISBN 978-619-7105-34-6. – ISSN 1314-2704. – June 18–24. – 2015. – Book 2. – Vol. 1. – P. 299–306. – DOI: 10.5593/SGEM2015/B21/S7.037.

36. *Ostroukh A.V.* Development of automated information systems for monitoring of intellectual activity results [Text] / A.V. Ostroukh, M.N. Krasyanskiy, A.D. Obukhov, S.V. Karpov, D.L. Dedov // SGEM2015 Conference Proceedings. – ISBN 978-619-7105-34-6. –ISSN 1314-2704. – June 18–24. – 2015. – Book 2. – Vol. 1. – P. 101–108. – DOI: 10.5593/SGEM2015/B21/S7.014.

37. *Баринов К.А.* Опыт разработки и использования ролевых игр для подготовки и переподготовки специалистов предприятий промышленности и транспортного комплекса [Текст] / К.А. Баринов, Д.А. Буров, А.В. Бугаев, А.В. Остроух // Научный вестник МГТУ ГА. – 2009. – № 141. – С. 189–197.

38. *Лукащук П.И.* Адаптивная методика прогнозирования пассажиропотоков в АСУ пассажирского автотранспортного предприятия [Текст] / П.И. Лукащук, С. Бенгедаш, А.Г. Николаев, А.В. Остроух // Приборы и системы. Управление, контроль, диагностика. – 2006. – № 11. – С. 7–11.

39. *Кузнецов И.А.* Особенности реализации автоматизированной информационно-аналитической системы центра планирования перевозок строительных грузов [Текст] / И.А. Кузнецов, А.В. Остроух // Вестник МАДИ (ГТУ). – М.: МАДИ (ГТУ), 2008. – Вып. 1 (12). – С. 92–96.

40. *Куфтинова Н.Г.* Процессно-ориентированный подход к автоматизации планирования и управления транспортировкой продукции предприятий промышленности [Текст] / Н.Г. Куфтинова, А.В. Остроух // Вестник МАДИ. – 2010. – Вып. 4 (23). – С. 62–66.

41. *Остроух А.В.* Имитационное моделирование управления транспортными потоками в мегаполисе [Текст] / А.В. Остроух, Н.Г. Куфтинова // Автотранспортное предприятие. – 2010. – № 12. – С. 41–42.

42. *Польгун М.Б.* Анализ моделей оперативного диспетчерского управления городским пассажирским транспортом [Текст] / М.Б. Польгун, А.В. Воробьева, А.В. Остроух // Молодой ученый. – 2011. – № 4. – Т.3. – С. 9–13.

43. *Ostroukh A.V.* Automation of Planning and Management of the Transportation of Production for Food Processing Industry Enterprises [Text] / A.V. Ostroukh, N.G. Kuftinova // Automatic Control and Computer Sciences. – 2012. – Vol. 46. – № 1. – P. 41–48. – DOI: 10.3103/S0146411612010063.

44. *Данчук К.А.* Автоматизированные информационные системы на автотранспортном предприятии [Текст] / К.А. Данчук, А.Б. Львова, С.А. Порфирьева, А.В. Остроух, П.С. Якунин // В мире научных открытий. – 2012. – № 2.6 (27). – С. 34–38.

45. *Ефименко Д.Б.* Использование программного обеспечения радионавигационных диспетчерских систем для транспортного

обслуживания специальных объектов нефтедобывающих компаний [Текст] / Д.Б. Ефименко, А.В. Остроух, А.Б. Николаев, А.Р. Исмаилов // Автотранспортное предприятие. – 2012. – № 2. – С. 42–44.

46. *Куфтинова Н.Г.* Разработка информационно-логической модели транспортной сети мегаполиса [Текст] / Н.Г. Куфтинова, А.В. Остроух // Бюллетень транспортной информации. – 2013. – № 1 (211). – С. 23–26.

47. *Польгун М.Б.* Автоматизация процессов диспетчерского управления городским пассажирским транспортом [Текст] / М.Б. Польгун, А.В. Остроух, А.Б. Николаев, Д.Б. Ефименко // Промышленные АСУ и контроллеры. – 2013. – № 5. – С. 10–16.

48. *Башмаков И.А.* Процессная модель технологии транспортировки бетонных смесей автомобильным транспортом [Текст] / И.А. Башмаков, А.В. Остроух // Автоматизация и управление в технических системах. – 2013. – № 4.1. – С. 75–81. – DOI: 10.12731/2306-1561-2013-4-14.

49. *Башмаков И.А.* Оптимизация параметров процессов автотранспортного обслуживания потребителей бетонных смесей [Текст] / И.А. Башмаков, М.Б. Польгун, А.В. Остроух // Автоматизация и управление в технических системах. – 2013. – № 4.2. – С. 189–198. – DOI: 10.12731/2306-1561-2013-4-39.

50. *Башмаков И.А.* Минимизация производственных рисков при автотранспортном обслуживании потребителей бетонных смесей [Текст] / И.А. Башмаков, А.В. Остроух // Автоматизация и управление в технических системах. – 2013. – № 4.2. – С. 83–90. – DOI: 10.12731/2306-1561-2013-4-40.

51. *Ostroukh A.V.* Automation of processes supervisory control urban passenger transport [Text] / A.V. Ostroukh, M.B. Polgun // International Journal of Advanced Studies (iJAS). – 2013. – Vol. 3. – № 3. – P. 3–9. – DOI: 10.12731/2227-930X-2013-3-1.

52. *Ostroukh A.V.* New approaches to development of automated supervisory systems of industrial enterprises transport [Text] / A.V. Ostroukh, M.B. Polgun // International Journal of Advanced Studies (iJAS). – 2013. – Vol. 3. – № 4. – P. 3–9. – DOI: 10.12731/2227-930X-2013-4-1.

53. *Остроух А.В.* Применение графов дорожной сети в автоматизированной системе управления транспортировкой дорожно-строительных материалов [Текст] / А.В. Остроух, М.Б. Польгун, В.В. Тихоцкий // Приборы и системы. Управление, контроль, диагностика. – 2014. – № 1. – С. 12–17.

54. *Ostroukh A.V.* Process Model of the Technology of Concrete Mixtures Transportation by Road [Text] / A.V. Ostroukh, I.A. Bashmakov, N.E. Surkova

// World Applied Sciences Journal (WASJ). – 2014. – Vol. 31. – № 4. – P. 500–507. – DOI: 10.5829/idosi.wasj.2014.31.04.333.

55. *Krupensky N.A.* Process-oriented aggregation scorecards operational control forwarding activities [Text] / N.A. Krupensky, A.M. Ivakhnenko, A.V. Ostroukh // Автоматизация и управление в технических системах. – 2014. – № 1.2 (9). – P. 123-142. – DOI: 10.12731/2306-1561-2014-1-26.

56. *Ostroukh A.V.* Process-Functional Model of Transportation Mix Concrete [Text] / A.V. Ostroukh, I.A. Bashmakov, M.B. Polgun // Journal of Transportation Technologies (JTT). – 2014. – Vol. 4. – Is. 2. – P.157–163. –DOI: 10.4236/jtts.2014.42016.

57. *Ostroukh A.* Development of Process-Oriented System For Operational Control of Freight Forwarding Activity [Text] / A. Ostroukh, A. Ivakhnenko, N. Krupensky // Journal of Applied Sciences (JAS). – 2014. – Vol. 14. – № 20. – P. 2601–2607. – DOI: 10.3923/jas.2014.2601.2607.

58. *Ostroukh A.* Architecture of Automated Navigation System of Passenger Transportation at Winter Olympic Games [Text] / A. Ostroukh, A. Ismailov, A. Lvova, A. Nikolaev // Trends in Applied Sciences Research (TASR). – 2014. – Vol. 9. – № 8. – P. 425–437. – DOI: 10.3923/tasr.2014.425.437.

59. *Куфтинова Н.Г.* Разработка автоматизированной системы обследования пассажиропотоков [Текст] / Н.Г. Куфтинова, А.В. Остроух // Автоматизация и управление в технических системах. – 2014. – № 3 (11). – С. 23–33. – DOI: 10.12731/2306-1561-2014-3-3.

60. *Ostroukh A.V.* Comparative study of routing protocols in vehicular ad-hoc networks (vanets) [Text] / A.V. Ostroukh, H. Elhadi // International Journal of Advanced Studies (iJAS). – 2014. – Vol. 4. – № 2. – P. 9–14. – DOI: 10.12731/2227-930X-2014-2-2.

61. *Kuftinova N.Ya.G.E.* Development of an automated system of survey passenger traffics [Text] / N.Ya.G.E. Kuftinova, A.V. Ostroukh // International Journal of Advanced Studies (iJAS). – 2014. – Vol. 4. – № 4. – P. 3–9. – DOI: 10.12731/2227-930X-2014-4-2.

62. *Kuftinova N.G.* Automated Control System For Survey Passenger Traffics [Text] / N.G. Kuftinova, A.V. Ostroukh, A.V. Vorobieva // International Journal of Applied Engineering Research. – 2015. – Vol. 10. – № 7. – P. 16419–16427.

63. *Гусеница Д.О.* Применение облачного хранения данных в автоматизированной системе диспетчерского управления транспортом [Текст] / Д.О. Гусеница, П.Ф. Юрчик, А.В. Остроух // Промышленные АСУ и контроллеры. – 2015. – № 5. – С. 59–66.

64. *Gusenitsa D.O.* Cloud Computing Application on Transport Dispatching Informational Support Systems [Text] / D.O. Gusenitsa, P.F. Yurchik, A.V. Os-

troukh // International Journal of Advanced Studies (iJAS). – 2015. – Vol. 5. – № 1. – P. 22–27. – DOI: 10.12731/2227-930X-2015-1-6.

65. *Ostroukh A.V.* Automated supervisory control system of urban passenger transport [Text] / A.V. Ostroukh, N.E. Surkova, M.B. Polgun, A.V. Vorobieva // ARPN Journal of Engineering and Applied Sciences. 2015. – Vol. 10. – № 10. – P. 4334–4340.

66. *Остроух А.В.* Математическая модель системы дистанционной диагностики неисправностей автомобилей [Текст] / А.В, Остроух, Н.Е. Суркова, А.В. Воробьева, Х.С. Салих // В мире научных открытий. – 2015. – № 6. – С. 63–70. – DOI: 10.12731/WSD-2015-6-6.

67. *Васюгова С.А.* Исследование перспектив и проблем интеграции человека с компьютером: искусственный интеллект, робототехника, технологическая сингулярность и виртуальная реальность [Текст] / С.А. Васюгова, А.В. Остроух, М.Н. Краснянский, А. Самаратунга // Перспективы науки. – Тамбов: ТМБПринт, 2011. – № 4 (19). – С. 109–112.

68. *Белоусова А.И.* Подход к формированию многоуровневой модели мультиагентной системы с использованием миваров [Текст] / А.И. Белоусова, О.О. Варламов, М.Н. Краснянский, А.В. Остроух // Перспективы науки. – Тамбов: ТМБПрин», 2011. – № 5 (20). – С. 57–61.

69. *Варламов О.О.* Анализ возможностей миварного подхода для систем искусственного интеллекта и современной робототехники [Текст] / О.О. Варламов, А.В. Остроух, М.Н. Краснянский, Т.Л. Давыдова // Вестник ТГТУ. – 2011. – Т.17. – № 3. – С.687–694.

70. *Ostroukh A.* Distributed System of Real Time Head Gesture Recognition in Development of Contactless Interfaces [Text] / A. Ostroukh, V. Nikonov, I. Ivanova, T. Morozova, V. Strakhov // Middle East Journal of Scientific Research. – 2014. – Vol. 20 (12). – P. 2177–2183. – DOI: 10.5829/idosi.mejsr.2014.20.12.21105.

71. *Ostroukh A.* Development of Contactless Integrated Interface of Complex Production Lines [Text] / A. Ostroukh, V. Nikonov, I. Ivanova [et al.] // Journal of Artificial Intelligence (JAI). – 2014. – Vol. 7. – № 1. – P. 1–12. – DOI: 10.3923/jai.2014.1.12.

72. *Morozova T.* Contactless integrated interface of production lines [Text] / T. Morozova, K. Sumkin, D. Akimov, A. Ostroukh // International Journal of Advanced Studies (iJAS). – 2014. – Vol. 4. – Is. 1. – P. 32–38. – DOI: 10.12731/2227-930X-2014-1-6.

73. *Омар М.* Применение систем распознавания образов в различных предметных областях [Текст] / М. Омар, Ф. Омар, М.И. Исмоилов, А.В. Остроух // Автоматизация и управление в технических системах. – 2014. – № 4 (12). – С. 32–47. – DOI: 10.12731/2306-1561-2014-4-4.

74. *Омар М.* Анализ современного состояния развития интеллектуальных роботов [Текст] / М. Омар, Ф. Омар, М.И. Исмоилов, А.В. Остроух // Автоматизация и управление в технических системах. – 2014. – № 4 (12). – С. 48–54. – DOI: 10.12731/2306-1561-2014-4-5.

75. *Ле К.Х.* Генетические алгоритмы в задачах рациональной организации информационно-вычислительных процессов сетей [Текст] / К.Х. Ле, Н.Е. Суркова, А.В. Остроух // Автоматизация и управление в технических системах. – 2014. – № 4 (12). – С. 82–99. – DOI: 10.12731/2306-1561-2014-4-9.

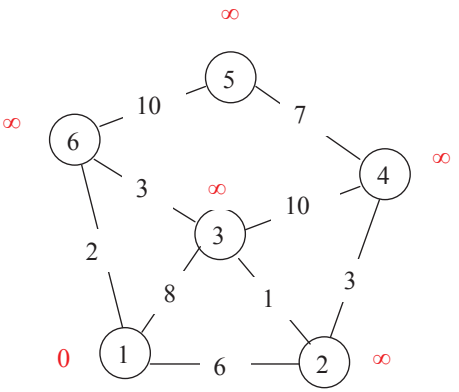
76. *Ivchenko V.* The Remotely Reconfigurable Intelligence of the Space-Based Mobile Robot [Text] / V. Ivchenko, P. Krug, T. Morozova, A. Ostroukh, S. Pavelyev // Journal of Engineering and Applied Sciences. – 2014. – Vol. 9. – Is. 10. – P. 389–395. – DOI: 10.3923/jeasci.2014.389.395.

77. *Ostroukh A.V.* Integration of PDM and ERP systems within a unified information space of an enterprise [Text] / A.V. Ostroukh, D.O. Gusenitsa, V.B. Golubkova, P.F. Yurchik // IOSR Journal of Computer Engineering (IOSR-JCE). – 2014. – Vol. 16. – Is. 02. – V. 6. – P. 31–33. – DOI: 10.9790/0661-16263133. – ANED: 11.0661/iosr-jce-E016263133.

Приложение 1

Задание по теме «Алгоритм Дейкстры»

Алгоритм Дейкстры (англ. Dijkstra's algorithm) – алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса.



Требуется найти кратчайшие расстояния от 1-й вершины до всех остальных.

Решение:

Шаг 1. Минимальную метку имеет вершина 1. Ее соседями являются вершины 2, 3 и 6.

Первый по очереди сосед вершины 1 – вершина 2, потому что длина пути до нее минимальна. Длина пути в нее через вершину 1 равна сумме значения метки вершины 1 и длины ребра, идущего из 1-й в 2-ю, то есть 0 + 6 = 6. Это меньше текущей метки вершины 2, бесконечности, поэтому новая метка 2-й вершины равна 6.

Аналогичную операцию проделываем с двумя другими соседями 1-й вершины – 3-й и 6-й.

К вершине 3: 0+8=8. Новая метка вершины три равна 8.

К вершине 6: 0+2=2. Новая метка вершины шесть равна 2.

Шаг 2. Шаг алгоритма повторяется.

Снова находим «ближайшую» из непосещённых вершин. Это вершина два с меткой 6.

Снова пытаемся уменьшить метки соседей выбранной вершины, пытаясь пройти в них через 2-ю вершину. Соседями вершины 2 являются вершины 1, 3 и 4. Первый (по порядку) сосед вершины 2 – вершина 1. Но она уже посещена, поэтому с 1-й вершиной ничего не делаем.

Следующий сосед вершины 2 – вершина 3, так как имеет минимальную метку из вершин, отмеченных как не посещённые. Если идти в неё через 2,

то длина такого пути будет равна 22 (6+16 = 22). Но текущая метка третьей вершины равна 8, а это меньше 22, поэтому метка не меняется.

Ещё один сосед вершины 2 – вершина 4. Если идти в неё через 2-ю, то длина такого пути будет равна сумме кратчайшего расстояния до 2-й вершины и расстояния между вершинами 2 и 4, то есть 9 (6+3=9). Поскольку $9<\infty$, устанавливаем метку вершины 4 равной 9.

Все соседи вершины 2 просмотрены, замораживаем расстояние до неё и помечаем её как посещённую.

Шаг 3. Повторяем шаг алгоритма, выбрав вершину 3.

Ранее вершине три мы присвоили значение 8.

Следующий сосед вершины 3 – вершина 6. Вершина 6 имеет метку 2. Если идти в вершину 3 через вершину 6, то длина такого пути будет равна 5 (2+3=5). Эта метка меньше предыдущей (5<8), значит вершине 3 присваиваем метку 5.

Следующий сосед вершины 3 – вершина 4. Вершина 4 имеет метку 9. Если идти в нее через вершину 3, то длина такого пути будет равна 15 (5+10=15). Но текущая метка четвертой вершины равна 9, а это меньше 15, поэтому метка не меняется.

Шаг 4. Повторяем шаг алгоритма, выбрав вершину 6.

Ранее вершине 6 мы присвоили метку 2.

Следующий сосед вершины 6 – вершина 5. Если идти в вершину 5 через вершину 6, то длина такого пути будет 12 (2+10). Присваиваем пятой вершине метку 12.

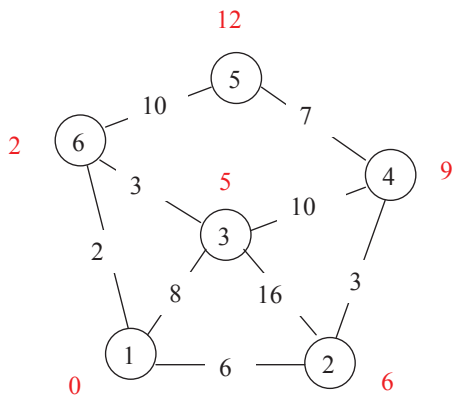
Шаг 5. Повторяем шаг алгоритма, выбрав вершину 5.

Ранее вершине 5 я присвоила метку 12.

Сосед вершины 5 – вершина 4. Если идти в вершину 5 через вершину 4, то длина такого пути будет 16 (9+7). Но текущая метка пятой вершины 12, что меньше 16. Значит метку 5 вершины не меняем.

Завершение выполнения алгоритма. Алгоритм заканчивает работу, когда нельзя больше обработать ни одной вершины. Результат работы алгоритма: кратчайший путь от вершины

до 2-й составляет 6,
до 3-й составляет 5,
до 4-й составляет 9,
до 5-й составляет 12,
до 6-й составляет 2.


12
5
10
7
2
6
3
5
4
9
10
3
2
8
16
3
1
6
2
0
6

Графы. Нахождение кратчайшего расстояния между двумя вершинами с помощью алгоритма Дейкстры

Задача о кратчайшем пути – задача поиска самого короткого пути (цепи) между двумя точками (вершинами) на графе, в которой минимизируется сумма весов рёбер, составляющих путь.

Задача о кратчайшем пути является одной из важнейших классических задач теории графов. Сегодня известно множество алгоритмов для её решения[⇨].

У данной задачи существуют и другие названия: задача о минимальном пути или, в устаревшем варианте, задача о дилижансе.

Значимость данной задачи определяется её различными практическими применениями[⇨]. Например, в GPS-навигаторах осуществляется поиск кратчайшего пути между двумя перекрёстками. В качестве вершин выступают перекрёстки, а дороги являются рёбрами, которые лежат между ними. Если сумма длин дорог между перекрёстками минимальна, тогда найденный путь самый короткий.

Задача поиска кратчайшего пути на графе может быть определена для неориентированного, ориентированного или смешанного графа. Далее будет рассмотрена постановка задачи в самом простом виде для неориентированного графа. Для смешанного и ориентированного графа дополнительно должны учитываться направления ребер.

В связи с тем, что существует множество различных постановок данной задачи, есть наиболее популярные алгоритмы для решения задачи поиска кратчайшего пути на графе:

Алгоритм Дейкстры находит кратчайший путь от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса.

Алгоритм Беллмана – Форда находит кратчайшие пути от одной вершины графа до всех остальных во взвешенном графе. Вес рёбер может быть отрицательным.

Алгоритм поиска A* находит маршрут с наименьшей стоимостью от одной вершины (начальной) к другой (целевой, конечной), используя алгоритм поиска по первому наилучшему совпадению на графе.

Алгоритм Флойда – Уоршелла находит кратчайшие пути между всеми вершинами взвешенного ориентированного графа.

Алгоритм Джонсона находит кратчайшие пути между всеми парами вершин взвешенного ориентированного графа.

Алгоритм Ли (волновой алгоритм) основан на методе поиска в ширину. Находит путь между вершинами s и t графа (s не совпадает с t), содержащий минимальное количество промежуточных вершин (рёбер). Основное применение – трассировки электрических соединений на кристаллах микросхем и на печатных платах. Так же используется для поиска кратчайшего расстояния на карте в стратегических играх.

Поиск кратчайшего пути на основе алгоритма Килдала.

Алгоритм Дейкстры – алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании и технологиях, например, его используют протоколы маршрутизации OSPF и IS-IS.

Примеры

Вариант 1. Дана сеть автомобильных дорог, соединяющих города Московской области. Некоторые дороги односторонние. Найти кратчайшие пути от города Москвы до каждого города области (если двигаться можно только по дорогам).

Вариант 2. Имеется некоторое количество авиарейсов между городами мира, для каждого известна стоимость. Стоимость перелёта из A в B может быть не равна стоимости перелёта из B в A. Найти маршрут минимальной стоимости (возможно, с пересадками) от Копенгагена до Барнаула.

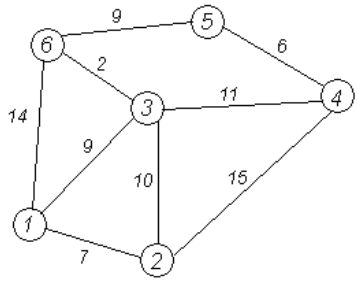
Каждой вершине из V сопоставим метку – минимальное известное расстояние от этой вершины до a. Алгоритм работает пошагово – на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Инициализация. Метка самой вершины a полагается равной 0, метки остальных вершин – бесконечности. Это отражает то, что расстояния от a до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

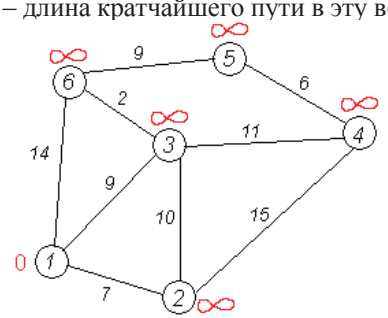
Шаг алгоритма. Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина u, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u, назовём соседями этой вершины. Для каждого соседа вершины u, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину u как посещённую и повторим шаг алгоритма.

Рассмотрим выполнение алгоритма на примере графа, показанного на рисунке.

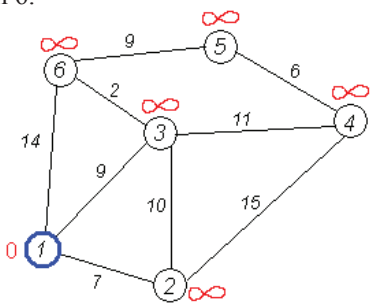
Пусть требуется найти кратчайшие расстояния от 1-й вершины до всех остальных.



Кружками обозначены вершины, линиями – пути между ними (рёбра графа). В кружках обозначены номера вершин, над рёбрами обозначена их «цена» – длина пути. Рядом с каждой вершиной красным обозначена метка – длина кратчайшего пути в эту вершину из вершины 1.

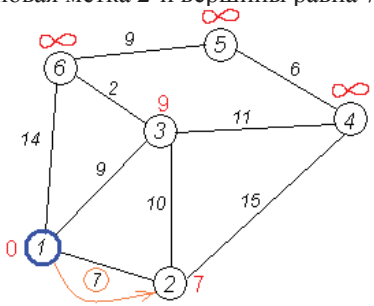


Первый шаг. Рассмотрим шаг алгоритма Дейкстры для нашего примера. Минимальную метку имеет вершина 1. Её соседями являются вершины 2, 3 и 6.

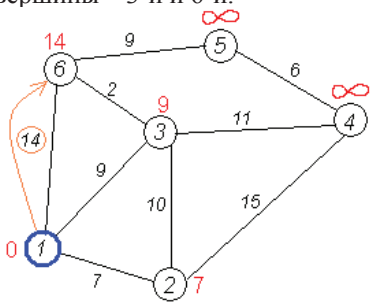


Первый по очереди сосед вершины 1 – вершина 2, потому что длина пути до неё минимальна. Длина пути в неё через вершину 1 равна сумме значения метки вершины 1 и длины ребра, идущего из 1-й в 2-ю, то есть 0 +

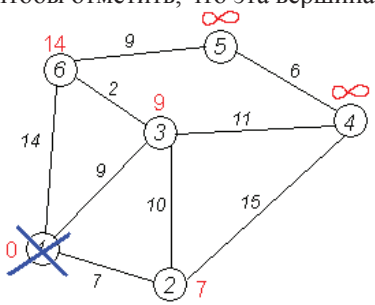
7 = 7. Это меньше текущей метки вершины 2, бесконечности, поэтому новая метка 2-й вершины равна 7.



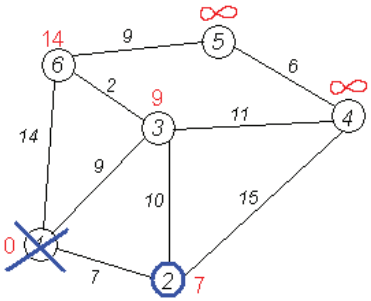
Аналогичную операцию проделываем с двумя другими соседями 1-й вершины – 3-й и 6-й.



Все соседи вершины 1 проверены. Текущее минимальное расстояние до вершины 1 считается окончательным и пересмотру не подлежит (то, что это действительно так, впервые доказал Э. Дейкстра). Вычеркнем её из графа, чтобы отметить, что эта вершина посещена.



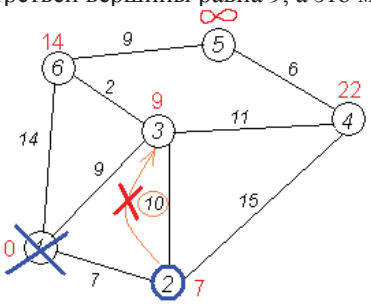
Второй шаг. Шаг алгоритма повторяется. Снова находим «ближайшую» из непосещённых вершин. Это вершина 2 с меткой 7.



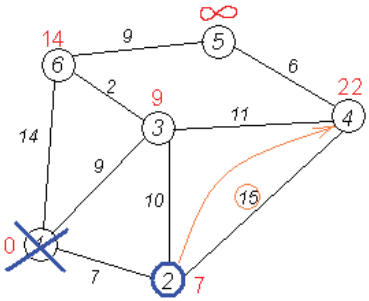
Снова пытаемся уменьшить метки соседей выбранной вершины, пытаясь пройти в них через 2-ю вершину. Соседями вершины 2 являются вершины 1, 3 и 4.

Первый (по порядку) сосед вершины 2 – вершина 1. Но она уже посещена, поэтому с 1-й вершиной ничего не делаем.

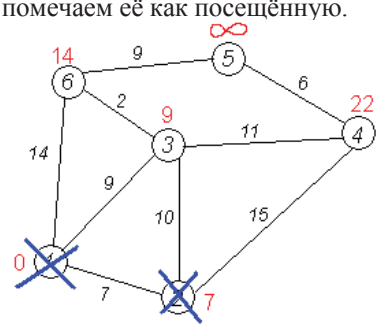
Следующий сосед вершины 2 – вершина 3, так как имеет минимальную метку из вершин, отмеченных как не посещённые. Если идти в неё через 2, то длина такого пути будет равна 17 (7 + 10 = 17). Но текущая метка третьей вершины равна 9, а это меньше 17, поэтому метка не меняется.



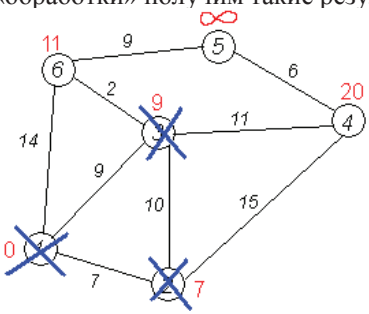
Ещё один сосед вершины 2 – вершина 4. Если идти в неё через 2-ю, то длина такого пути будет равна сумме кратчайшего расстояния до 2-й вершины и расстояния между вершинами 2 и 4, то есть 22 (7 + 15 = 22). Поскольку $22<\infty$, устанавливаем метку вершины 4 равной 22.



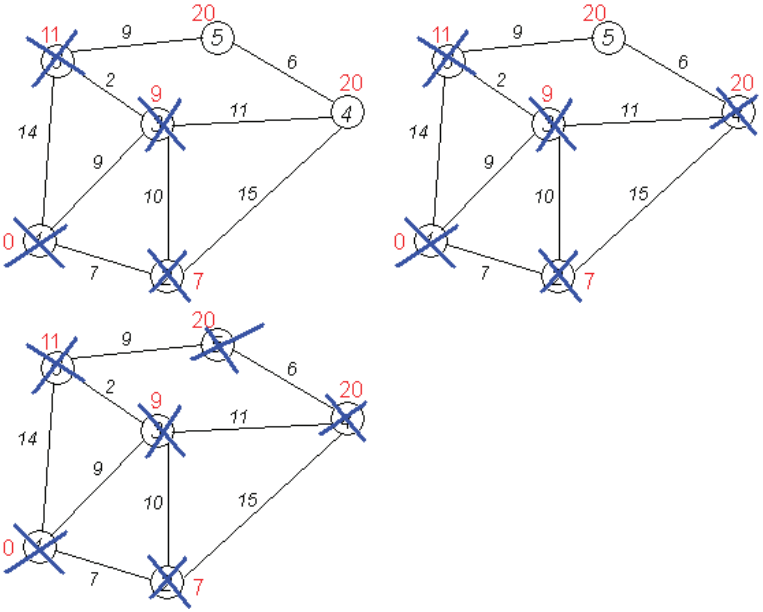
Все соседи вершины 2 просмотрены, замораживаем расстояние до неё и помечаем её как посещённую.



Третий шаг. Повторяем шаг алгоритма, выбрав вершину 3. После её «обработки» получим такие результаты:



Дальнейшие шаги. Повторяем шаг алгоритма для оставшихся вершин. Это будут вершины 6, 4 и 5, соответственно порядку.



Завершение выполнения алгоритма. Алгоритм заканчивает работу, когда нельзя больше обработать ни одной вершины. В данном примере все вершины зачёркнуты, однако ошибочно полагать, что так будет в любом примере – некоторые вершины могут остаться незачёркнутыми, если до них нельзя добраться, т. е. если граф несвязный. Результат работы алгоритма виден на последнем рисунке: кратчайший путь от вершины 1 до 2-й составляет 7, до 3-й – 9, до 4-й – 20, до 5-й – 20, до 6-й – 11.

Задание по теме «Алгоритм Флойда – Уоршелла»

Найти для сети, показанной на рисунке 1, кратчайшие пути между любыми двумя узлами. Расстояние между узлами этой сети проставлены на рисунке возле соответствующих ребер. Ребро (3,5) ориентировано, поэтому не допускается движение от узла 5 к узлу 3. Все остальные ребра допускают движение в обе стороны:

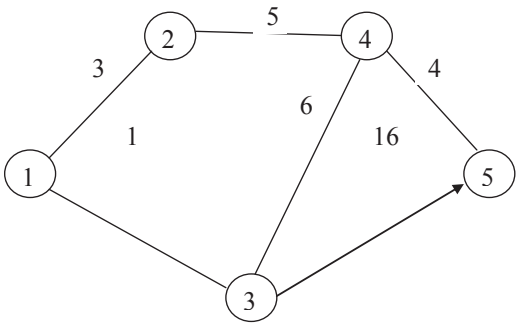


Рисунок 1

Необходимо найти для сети кратчайшие пути между любыми двумя узлами.

Решение:

Шаг 0. Начальные матрицы $\mathbf{D_0}$ и $\mathbf{S_0}$ строятся непосредственно по заданной схеме сети.

$\mathbf{D_0}$

	1	2	3	4	5
1	-	3	11	∞	∞
2	3	-	∞	5	∞
3	11	∞	-	6	16
4	∞	5	6	-	4
5	∞	∞	∞	4	-

$\mathbf{S_0}$

	1	2	3	4	5
1	-	2	3	4	5
2	1	-	3	4	5
3	1	2	-	4	5
4	1	2	3	-	5
5	1	2	3	4	-

Шаг 1. В матрице выделены ведущие строка и столбец **(k=1).**

Элементы d_{23} и d_{21} единственные среди элементов матрицы $\mathbf{D_0}$, значения которых можно улучшить с помощью треугольного оператора. Таким образом, чтобы на основе матриц $\mathbf{D_0}$ и $\mathbf{S_0}$ получить матрицы $\mathbf{D_1}$ и $\mathbf{S_1}$, выполняем следующие действия.

1. Заменяем $\mathbf{d_{23}}$ на $\mathbf{d_{21}+d_{13}=3+11=14}$ и устанавливаем $\mathbf{s_{23}=1}$.

2. Заменяем $\mathbf{d_{32}}$ на $\mathbf{d_{31}+d_{12}=11+3=14}$ и устанавливаем $\mathbf{s_{32}=1}$.

Матрицы **D1** и **S1** имеют следующий вид:

D_1

	1	2	3	4	5
1	-	3	11	∞	∞
2	3	-	**14**	5	∞
3	11	**14**	-	6	16
4	∞	5	6	-	4
5	∞	∞	∞	4	-

S_1

	1	2	3	4	5
1	-	2	3	4	5
2	1	-	**1**	4	5
3	1	**1**	-	4	5
4	1	2	3	-	5
5	1	2	3	4	-

Шаг 2. Полагаем **k = 2**; в матрице $\mathbf{D_1}$ выделены ведущие строка и столбец. Треугольный оператор применяется к элементам матрицы $\mathbf{D_1}$ (d_{14}, d_{41}) и $\mathbf{S_1}$ (s_{14} , s_{41}). В результате получаем матрицы $\mathbf{D_2}$ и $\mathbf{S_2}$:

D_2

	1	2	3	4	5
1	-	3	11	**8**	∞
2	3	-	14	5	∞
3	11	14	-	6	16
4	**8**	5	6	-	4
5	∞	∞	∞	4	-

S_2

	1	2	3	4	5
1	-	2	3	**2**	5
2	1	-	1	4	5
3	1	1	-	4	5
4	**2**	2	3	-	5
5	1	2	3	4	-

Шаг 3. Полагаем **k = 3**; в матрице $\mathbf{D_2}$ выделены ведущие строка и столбец. Треугольный оператор применяется к элементам матрицы $\mathbf{D_2}$ (d_{15} и d_{25}) и $\mathbf{S_2}$ (s_{15} и s_{25}). В результате получаем матрицы $\mathbf{D_3}$ и $\mathbf{S_3}$:

D_3

	1	2	3	4	5
1	-	3	11	8	**27**
2	3	-	14	5	**30**
3	11	14	-	6	16
4	8	5	6	-	4
5	∞	∞	∞	4	-

S_3

	1	2	3	4	5
1	-	2	3	2	**3**
2	1	-	1	4	**3**
3	1	1	-	4	5
4	2	2	3	-	5
5	1	2	3	4	-

Шаг 4. Полагаем **k = 4**, ведущие строка и столбец в матрице $\mathbf{D_3}$ выделены.

Треугольный оператор применяется к элементам матрицы $\mathbf{D_3}$ (d_{15}, d_{23}, d_{25}, d_{32}, d_{35}, d_{51}, d_{52}, d_{53}) и $\mathbf{S_3}$ (s_{15}, s_{23}, s_{25}, s_{32}, s_{35}, s_{51}, s_{52}, s_{53}). Получаем новые матрицы $\mathbf{D_4}$ и $\mathbf{S_4}$:

D4

	1	2	3	4	5
1	-	3	11	8	**12**

S4

	1	2	3	4	5
1	-	2	3	2	**4**

•	2	3	-	**11**	5	**9**	•
•	3	11	**11**	-	6	**10**	•
•	4	8	5	6	-	4	•
•	5	**12**	**9**	**10**	4	-	•

•	2	1	-	**4**	4	**4**
•	3	1	**4**	-	4	**4**
•	4	2	2	3	-	5
•	5	**4**	**4**	**4**	4	-

Шаг 5. Полагаем **k = 5**, ведущие строка и столбец в матрице $\mathbf{D_4}$ выделены. Никаких действий на этом шаге не выполняем; вычисления закончены.

Конечные матрицы $\mathbf{D_4}$ и $\mathbf{S_4}$ содержат всю информацию, необходимую для определения кратчайших путей между любыми двумя узлами сети. Например, кратчайшее расстояние между узлами 1 и 5 равно $\mathbf{d_{15} = 12}$.

Приложение 3

Задание по теме
«Данные дистанционного зондирования Земли (ДДЗ) и спектральное представление цвета»

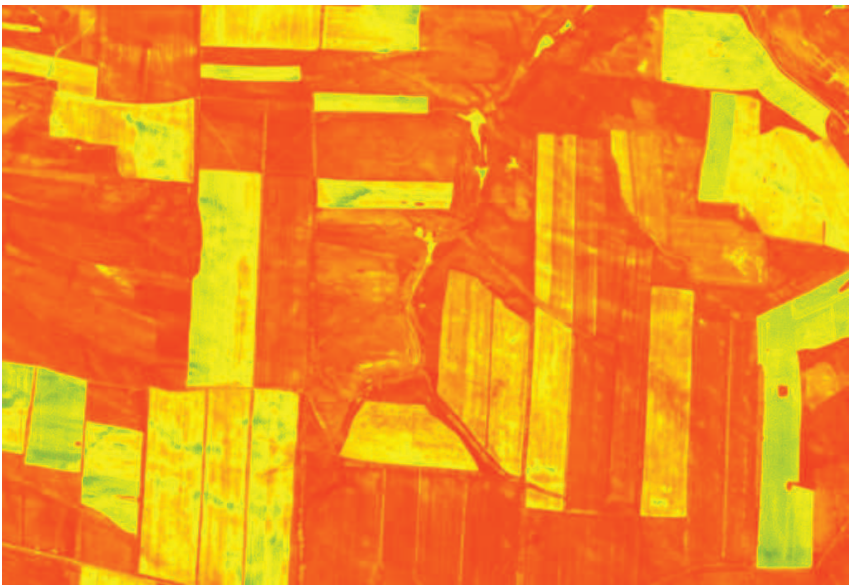
1. Определение ДЗЗ

Дистанционное зондирование Земли (ДЗЗ) – наблюдение поверхности Земли наземными, авиационными и космическими средствами, оснащёнными различными видами съемочной аппаратуры. Рабочий диапазон длин волн, принимаемых съёмочной аппаратурой, составляет от долей микрометра (видимое оптическое излучение) до метров (радиоволны). Методы зондирования могут быть пассивные, то есть использующие естественное отраженное или вторичное тепловое излучение объектов на поверхности Земли, обусловленное солнечной активностью, и активные – использующие вынужденное излучение объектов, инициированное искусственным источником направленного действия. Данные ДЗЗ, полученные с космического аппарата (КА), характеризуются большой степенью зависимости от прозрачности атмосферы. Поэтому на КА используется многоканальное оборудование пассивного и активного типов, регистрирующее электромагнитное излучение в различных диапазонах.

Аппаратура ДЗЗ первых КА, запущенных в 1960–70-х гг. была трассового типа – проекция области измерений на поверхность Земли представляла собой линию. Позднее появилась и широко распространилась аппаратура ДЗЗ панорамного типа – сканеры, проекция области измерений на поверхность Земли которых представляет собой полосу.

Космические аппараты дистанционного зондирования Земли используются для изучения природных ресурсов Земли и решения задач метеорологии. КА для исследования природных ресурсов оснащаются в основном оптической или радиолокационной аппаратурой. Преимущества последней заключаются в том, что она позволяет наблюдать поверхность Земли в любое время суток, независимо от состояния атмосферы.

2. Мультивременной композит.



3. RGB канал.

RGB (аббревиатура английских слов **r**ed, **g**reen, **b**lue – красный, зелёный, синий) или **КЗС** – аддитивная цветовая модель, описывающая способ кодирования цвета для цветовоспроизведения с помощью трёх цветов, которые принято называть основными. Выбор основных цветов обусловлен особенностями физиологии восприятия цвета сетчаткой человеческого глаза.

RGB-модель является аддитивной, где цвета получаются путём добавления к чёрному цвету. При отсутствии краски нет никакого цвета – чёрный, максимальное смешение даёт белый. Если цвет экрана, освещённого цветным прожектором, обозначается в RGB как (r1, g1, b1), а цвет того же экрана, освещённого другим прожектором, – (r2, g2, b2), то при освещении двумя прожекторами цвет экрана будет обозначаться как (r1+r2, g1+g2, b1+b2).

Изображение в данной цветовой модели состоит из трёх каналов. При смешении основных цветов, например, синего (B) и красного (R), получается пурпурный (M, magenta), зелёного (G) и красного (R) – жёлтый (Y, yellow), зелёного (G) и синего (B) – циановый (C, cyan). При смешении всех трёх основных цветов получается белый цвет (W, white).

В телевизорах и мониторах ЭЛТ применяются три электронных пушки для красного, зелёного и синего каналов. В ЖК- и других матричных

мониторах и телевизорах носителями трёх цветов являются светоточки (светодиоды, светофильтры).

Наиболее распространённое цветовое пространство с использованием модели RGB – sRGB – имеет по многим тонам цвета более широкий цветовой охват (может представить более насыщенные цвета), чем в цветовых пространствах CMYK, поэтому иногда изображения, замечательно выглядящие в RGB, значительно тускнеют и гаснут в CMYK.

При этом, однако, в отличие от CMYK (где основные цвета сине-зелёный циан, пурпурный-маджента и жёлтый) и модели ИоганнесаИттена (где основные цвета красный, жёлтый и синий), по RGB-модели синтезирование всех цветов возможно только в компьютерных и телевизионных технологиях, но не на практике – в красках, в светотехнике и т.п. Например, по RGB-модели на практике невозможно синтезирование белого и жёлтого цвета (смешением зелёного и красного) и всех цветов, где жёлтый участвует как составляющий или оттенок

Рисунок 2 – Аддитивная цифровая модель

4. Выбрать цвет

Выбрать цвет.

RAL 3016-

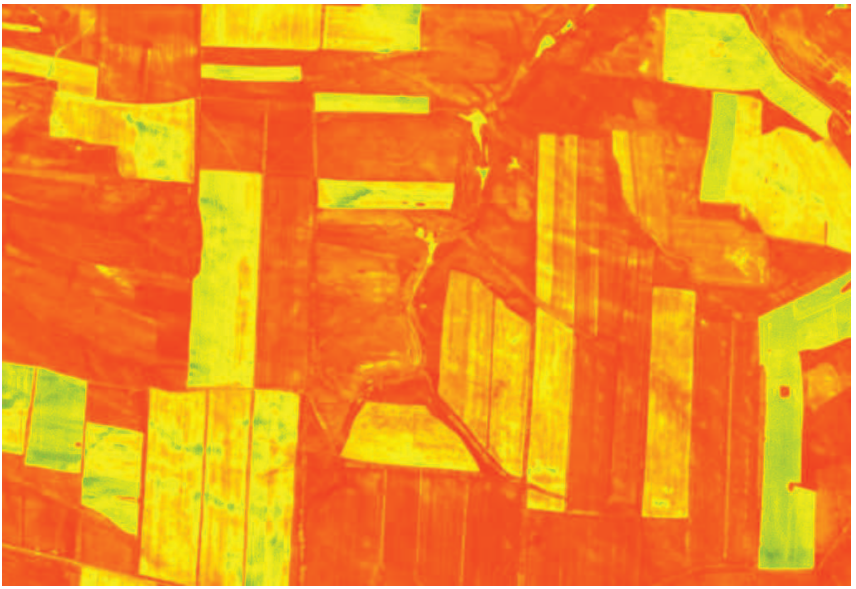
Задача описать  *выбранный нами цвет в виде RGB - цифровое представление цвета.*

В виде RGB (*,*,*), где «*» – числа от 0 до 255, обозначающих количество соответствующего цвета (красный, зелёный, синий) в получаемом.

RGB (166, 61, 47)

Задача выбрать цветовую палитру своего мультивременного композита. Составить таблицу применяемых цветов - https://ru.inettools.net/image/opredelit-tsvet-piksela-na-kartinke-onlayn

СОСТАВИТЬ ТАБЛИЦУ ПРИМЕНЯЕМЫХ ЦВЕТОВ



Цвет №1	Цвет №2	Цвет №3	Цвет №4	Цвет №5
#fe5900	#fe7d00	#fe3800	#fdf700	#feb200
RGB(254,89,0)	RGB(254,125,0)	RGB(254,56,0)	RGB(253,247,0)	RGB(254,178,0)

Выводы:

Алгоритмы ИИС обрабатывают обратно отраженный сигнал и предоставляют отраженный сигнал в цветовой гамме на мультивременном композите.

Приложение 4

Задания по теме
«Технология распознавания психофизиологического состояния человека»

Задания :

Дать определение технологии распознавания лиц

Дать определение технологии распознавания образов

Прикрепить рисунок человека (лицо), расставить 72 основные точки распознавания мимики человека

Задачи классификации:

распознавание символов;

распознавание речи;

установление медицинского диагноза;

прогноз погоды;

распознавание лиц

классификация документов и др.

IT-технологии нацелены на выявление психофизиологического состояния человека. Технология распознавания лиц является основной технологией определения психофизиологического состояния человека.

Задание №1

1. Определение

Технология распознавания лиц – это метод идентификации или проверки личности человека по его лицу, а точнее по его особым антропометрическим точкам. Системы распознавания лиц используются для идентификации людей на фотографиях, видео или в режиме реального времени.

2. Этапы

Алгоритм работы технологии распознавания лиц состоит из двух этапов: идентификация (кто этот человек?) и верификация (а тот ли это человек, за которого он себя выдает?).

Задание №2

1. Определение

Технология распознавания образов – это метод выделения и сегментирования данных в соответствии с установленными критериями или общими элементами, который выполняется специальными алгоритмами. Распознавание образов имеет дело с автоматическим обнаружением закономерностей в данных с помощью компьютерных алгоритмов и с использованием этих закономерностей для принятия действий, таких как классификация данных по различным категориям.

2. Сферы применения:

Контроль доступа к объектам или системам

Выявление нарушителей

Определение портрета покупателя

Идентификация в банковском секторе

Управление рабочим временем
Оплата услуг
Проход на стадионы, вокзалы
Умный город
Доступ к экзаменам

Задание №3

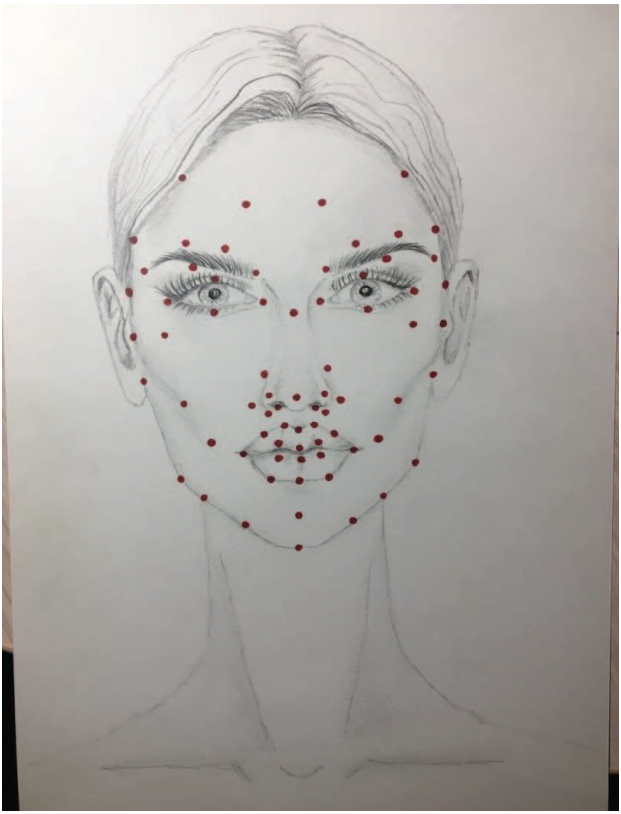
1) Определение антропометрических точек

Основные точки лица- антропометрические точки.

Минимальное кол-во антропометрических точек- 68 точек (расположены по контуру лица, определяют положение и форму подбородка, глаз, носа и рта, расстояние между ними). Проводятся дополнительные преобразования изображения (устранение наклона головы, коррекция цвета лица и так далее) с целью получения четкого фронтального снимка.

Вычисляется дескриптор – набор характеристик, описывающих лицо независимо от посторонних факторов (возраст, прическа, макияж). Анализируются специальные локальные признаки, характеризующие, например, текстуру определенных областей на лице. Сопоставление разных дескрипторов позволяет оценить, относятся ли два полученных изображения лица к одному и тому же человеку.

2) Рисунок мимических точек на лице.



Методы распознавания лиц:

Методы извлечения признаков условно делятся на две группы: использующие локальные и глобальные признаки лица. При использовании локальных – алгоритм выделяет отдельные части (глаза, нос, рот и др.) и уже по ним распознает лицо. При использовании глобальных – оперирует со всем лицом в целом.

Один из самых распространенных методов – нейросетевой.

Российский рынок технологии распознавания на сегодняшний день значительно вырос и не перестаёт расти. Сейчас технология распознавания лиц получает в России существенную поддержку со стороны государства, в том числе на уровне законодательства. К концу 2018 года доля технологий распознавания лиц в общем объеме российского биометрического рынка составила почти 50%, а в течение четырех лет этот сегмент показывал рост на уровне 106,7% в год. Растущий спрос эксперты связывают с технологическим прорывом в области машинного обучения, IT-технологий.

Выводы:

Подводя итоги работы, можно сделать вывод, что ключевые точки, по которым распознают лица - это обычно:

расстояние между глазами;

форма надбровных дуг;

положение и ширина носа;

форма подбородка.

По таким признакам распознает лица фирменный сервис от Гугл- Google Фото, сервисы магазинов и бутиков, программы для определения владельца телефона и т.д.

Технологии распознавания лиц применяются согласно строго регламентированной нормативно-законодательной базе: обработка биометрических данных разрешается лишь с письменного разрешения субъекта, хотя есть ряд исключений – в основном связанных с охраной порядка, противостоянием терроризму и оборонными задачами. Сами данные должны защищаться от «неправомерного или случайного доступа к ним, их уничтожения, изменения, блокирования, копирования, предоставления, распространения».

К технологиям хранения данных, в свою очередь, прописано три требования, первое- доступ для уполномоченных лиц, а два других – это возможность использования цифровой подписи либо других способов обеспечить целостность и неизменность данных, а также проверка на предмет того, есть ли письменное согласие субъекта на обработку его биометрических данных.

Приложение 5

Нравственно-этические и правовые требования в отношении разработчиков и производителей юнитов искусственного интеллекта

№ 2102 (2017) от 28.04.2017 «Слияние с технологиями, искусственный интеллект и права человека»

Искусственный интеллект и робототехника Олег Фиговский (Израиль) и Валерий Гумаров (Россия) «Слияние человека с искусственным интеллектом (ИИ) принесёт людям пользу и улучшит качество их жизни», – отметил футуролог Рэй Курцвейл во время выступления на фестивале SXSW в Остине. Люди перенесут своё сознание в облако и смогут «разгрузить» мозг. По мнению эксперта, это приведёт к увеличению неокортекса – новых областей коры головного мозга, которые у человека отвечают за сенсорное восприятие, осознанное мышление, речь, способность к искусствам и чувство юмора.

За последнее время ИИ развивается так быстро, что теперь не проходит и месяца без сообщений о прорывах в сфере ИИ. В самых разных областях человеческой деятельности компьютер все чаще начинает превосходить человека. И все чаще говорится о том, как ИИ повлияет на занятость людей. Не только дремучие обыватели, но и многомудрые эксперты опасаются, что по мере развития искусственного интеллекта людям будет оставаться все меньше работы, а значит, будет расти количество безработных, которые экономически не смогут конкурировать с машинами.

Давайте посмотрим на обычный банкомат. Если бы пришлось выбирать технологию, которая выглядела бы так, словно собиралась заменить всех людей, банкомат бы вполне подошёл. Но, тем не менее, банковских служащих сегодня намного больше, чем банкоматов. Как так? Да все просто: банкоматы снизили стоимость открытия банковских отделений, банки открыли больше отделений, пригласив больше людей. Аналогичным образом ИИ создаст миллионы рабочих мест, которые намного превзойдут наши представления. Например, ИИ станет экспертом в области языкового перевода, и вместе с этим вырастет спрос на переводчиков. Почему? Если стоимость обычного перевода упадёт почти до нуля, упадёт и стоимость ведения бизнеса с теми, кто говорит на других языках. Таким образом, предприниматели будут расширять бизнес за границей, создавая больше работы для людей-переводчиков. ИИ может делать простую работу, но для тонкой работы нужны люди.

Темпы внедрения инноваций будут и далее нарастать, и развивающимся странам для обеспечения своей конкурентоспособности в экономике будущего необходимо будет действовать быстро. Чтобы использовать преимущества новых технологий и смягчать наиболее острые из порождаемых ими проблем, им придётся «с ощущением совершенной неотложности» осуществлять инвестиции в своих граждан – прежде всего, в образование и здравоохранение, которые являются краеугольным камнем человеческого капитала. Но сегодня огромное число стран таких критически важных капиталовложений не осуществляют.

По данным последнего обследования «Евробарометра», три четверти граждан Европейского Союза, который по образу жизни является мировой сверхдержавой, уверены, что для их рабочих мест новые технологии являются

благом. Две трети утверждают, что новые технологии благотворно влияют на общество и ещё больше улучшают качество жизни.

Медицинская помощь. В апреле 2018 года Управление по санитарному надзору за качеством пищевых продуктов и медикаментов США разрешило продажу первого ИИ, который диагностирует проблемы со здоровьем в клиниках первичной медицинской помощи без специального наблюдения. Программа, которая проверяет изображения глаз на наличие признаков потери зрения, связанной с диабетом, может быть крайне полезна для людей в отдалённых районах или районах с ограниченными ресурсами, где не хватает офтальмологов. Другие программы искусственного интеллекта учатся распознавать самые разные проблемы со здоровьем – от возрастной потери зрения до нарушений в работе сердца.

Инструмент для дезинформации. Конечно, умный ИИ не всегда хорошая новость. Один ИИ, появившийся в 2018 году, генерирует реалистичные фальшивые видеоматериалы, заставляя объект одного видео отражать движения и эмоции другого человека в другом видео. В чужих руках этот ИИ мог бы стать мощным инструментом распространения дезинформации.

ИИ уже сдал экзамен лучше человека. Нейросеть сдала Стэнфордский тест на чтение и понимание текста лучше человека. Тест считается одним из наиболее точных инструментов для измерения способностей интеллекта. В этом году ИИ прошёл опросник с результатом 82,6%, лучший результат человека – 82,3%. Чем лучше ИИ понимает человека, тем проще бизнесу применять его для различных задач, в том числе связанных с обслуживанием клиентов. ИИ сможет полноценно принимать и правильно адресовать специалистам вопросы от пользователей, регистрировать людей на рейс, автоматически открывать счёт в банке, делать заказы в интернет-магазине и выполнять другие поручения. Развитие технологий обработки естественного языка особенно значимо для юристов, анализирующих большой объем договоров и контрактов на предмет нарушений, риск-менеджеров, которые оценивают последствия решений для компании, а также для создания более интеллектуальных виртуальных ассистентов.

ИИ стал главной темой Всемирного экономического форума в Давосе 2018 года – глобальной трибуны для обсуждения экономических и общественных вопросов. Представитель Accenture озвучил оценку: при условии, что бизнес будет активно инвестировать во взаимодействие машин и людей, в 2022 году доходы компаний от ИИ вырастут на 38%. Глава IBM Джинни Рометти рассказала о концепции «объясняемого ИИ», когда технологии не только решают задачи компаний, но и аргументируют свои действия. Это уменьшает недоверие людей к новым технологиям и допускает их применение в более сложных процессах. Генеральный директор Salesforce Марк Бениофф сообщил: совет директоров корпорации использует ИИ, чтобы принимать стратегические решения. ВЭФ ещё раз продемонстрировал, что компаниираз работчики и производители вычислительных мощностей становятся влиятельными игроками на международной арене.

Изобретатель ставит роботу задачу с новыми начальными условиями. Ни секунды не задумываясь, робот даёт ответ: «Решение этой задачи сводится к решению предыдущей: надо вынуть дрова из печи, снять чайник с плиты, вылить

из него воду, далее действовать по предыдущему алгоритму». Это к тому, что, впрочем, многие разработчики про то знают, но не всегда в свои программы встроить могут, что логика машины отличается от логики человека. ИИ, конечно, быстрее считает, чем человек, но человек быстрее думает – он не тратит время на обработку заведомо ненужных вариантов решения задачи и не сводит множество решений в один алгоритм. Человек действует по ситуации, которая постоянно меняется, машина по логике, которую в неё человек заложил. «Алгоритм мудрости» для ИИ – принятие нелогичных, но правильных решений – пока не создан.

Сегодня уже сложно представить такую область деятельности, в которую бы ни проникли различные умные устройства, упрощающие нашу работу или берущие на себя часть наших обязанностей. Среди таких сфер – медицина, образование, бизнес, наука, развлечения, борьба с преступностью, решение многочисленных бытовых вопросов. Скорее всего, в будущем подобных разработок станет еще больше, и использоваться они, наверняка, будут повсеместно. Таким образом, уже в ближайшем будущем применение искусственного интеллекта качественно преобразит практически все сферы нашей жизни.

Столь широкое использование ИИ обусловлено двумя важнейшими факторами. С одной стороны, он способен автоматизировать даже те процессы, которые ранее требовали участия человека: например, управление роботизированными механизмами на производстве (то есть в данном случае ИИ берет на себя наши обязанности). С другой стороны, он может быстро обрабатывать и анализировать поистине гигантские объемы информации и просчитывать варианты, используя множество переменных. И по данному направлению ИИ дает качественно лучшие результаты по сравнению с человеком. Добавим к этому то, что машина не подвержена человеческому фактору, а ее работоспособность не зависит от эмоций и личных проблем. Как итог – области применения искусственного интеллекта очень широки и фактически ограничиваются только нашей фантазией и скоростью внедрения технологических новаций.

В каких сферах ИИ применяется уже сейчас?

Несмотря на сравнительную молодость данных технологий, ИИ уже нашел широкое применение в самых разных сферах, и многие проекты, будто пришедшие к нам из фантастических книг, становятся вполне реальными. Приведем интересные примеры применения искусственного интеллекта, которые внедрены на данный момент или планируются к внедрению в ближайшем будущем.

Медицина

В медицине особенно ценится отменная память искусственного интеллекта и его способность обрабатывать большое количество данных, сопоставлять и анализировать информацию. Так работает DeepMindHealth от компании Google. Эти и аналогичные им умные помощники не просто дают советы врачам, но и определяют предрасположенность к заболеваниям или выявляют их на очень ранних стадиях, когда они могут скрыться от человеческого глаза.

Рисунок 1 – Приложение DeepMindHealth

В конце 2017 года премьер-министр РФ Д. Медведев обозначил стратегию, которая в том числе подразумевает использование в российском здравоохранении возможностей искусственного интеллекта. Например, планируется развивать систему поддержки принятия решений врача «Третье мнение». Сейчас она умеет анализировать снимки клеток крови и глазного дна, УЗИ мочевого пузыря и рентгенограммы легких, а в будущем научится обрабатывать данные компьютерных томографов и МРТ. Еще одна аналогичная российская система – Botkin.AI. Среди ее задач – анализ диагностических данных, подсказки и советы врачам, мониторинг проводимого лечения. Пока Botkin.AI помогает онкологам, но планируется, что уже скоро он будет работать и в других областях.

Проект Face2Gene от компании FDNA обещает определить генетические заболевания по фото. По словам разработчиков, по чертам лица можно выявить около 3 500 генетических заболеваний, даже если по симптомам они себя еще не проявили. Приложение доступно для смартфонов на Andriod и iOS.

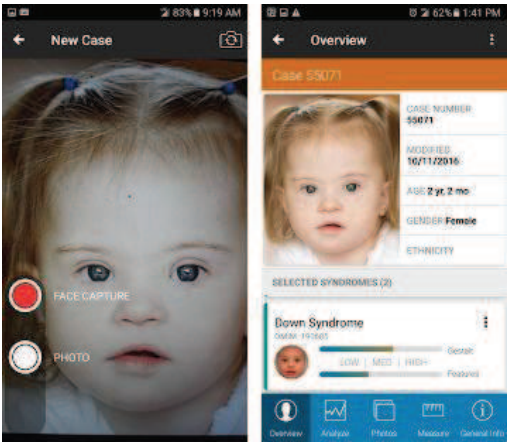


Рисунок 2 – Face2Gene

Искусственный интеллект помогает не только врачам, но и пациентам. В последние годы растет популярность телемедицины и соответствующих приложений. Они используют различные алгоритмы: некоторые собирают данные с носимых датчиков вроде фитнес-браслетов; другие, скорее, представляют собой опросники, цель которых – установить точные симптомы и проблемы пациентов. Некоторые ИИ распознают речь, и им можно отвечать устно, другие предпочитают письменную коммуникацию. Получив нужную информацию, приложения либо дают рекомендации, что делать дальше и как лечиться, либо отправляют соответствующие сведения лечащему врачу. Одни из самых известных интеллектуальных помощников такого рода – Ada и Your.MD (можно скачать в GooglePlay и AppStore).

Промышленность и сельское хозяйство

В промышленности искусственный интеллект позволяет делать работу все более и более автоматизированной, вплоть до того что участие человека практически перестает требоваться. В частности, LG планирует в 2023 году открыть завод, где все процессы – от закупки расходных материалов до контроля выпускаемой продукции и ее отгрузки – будут осуществляться с помощью искусственного интеллекта. Также ИИ будет контролировать износ оборудования, выполнение поставленных планов и другие факторы, которые обычно отслеживает человек.

Что касается сельского хозяйства, то тут искусственный интеллект используется для контроля за состоянием растений, уровнем влажности, наличием в почве необходимых питательных веществ и в принципе для надлежащего ухода за посадками. Например, роботы научились идентифицировать сорняки и аккуратно избавляться от них (выдергивая или обрабатывая химикатами). Умные помощники способны определять заболевания растений или напавших на них вредителей по фотографиям, а

также точечно доставлять необходимые препараты. Это помогает экономнее расходовать пестициды и гербициды.

Дорожное движение

Во многих странах умение искусственного интеллекта обрабатывать огромные объемы данных используется для того, чтобы облегчить проблему пробок. В частности, в России ИИ помогает движению в крупных городах и на федеральных трассах. Компьютер анализирует данные со светофоров, собирает информацию о плотности движения, авариях, погодных условиях и иных причинах, которые могут повлиять на трафик. В итоге интеллектуальная система в режиме реального времени следит за дорогами, строит прогнозы, как будет развиваться ситуация, и в соответствии с этим переключает светофоры.

ИИ, следящий за дорожным движением, не только наблюдает за авариями, но и помогает водителям. Например, может вызвать эвакуатор.

Подобные системы работают во многих городах Европы, Азии, Северной Америки, для которых актуальна проблема пробок. Конечно, полностью избавиться от заторов в большинстве случаев не удается, однако ИИ позволяет улучшить ситуацию с дорожным движением, порой – значительно ускорить движение. Возможно, прогресс будет заметнее, когда в широкий обиход войдут автономные автомобили – еще одна сфера применения искусственного интеллекта.

Искусственный интеллект в быту

Конечно, типичным примером использования ИИ в быту станут системы умных домов, которые получают все большее распространение. Задача большинства подобных разработок – максимально автоматизировать и облегчить наш быт. Например, с утра ИИ сможет раздвинуть занавески, чтобы в спальню проник солнечный свет, разбудить вас с помощью радио и включить кофеварку, чтобы на завтрак вас уже ждал ароматный кофе, а когда вы уйдете на работу, он активирует сигнализацию. В будущем функционал таких систем наверняка будет значительно расширен, вплоть до того, что холодильник сам закажет вашу любимую еду, а шкаф – отпарит одежду.

Умный дом оптимизирует энергопотребление, обогрев и вентиляцию, контролирует работу различных приборов, подстраиваясь под ваше расписание. В совокупности это не только делает быт удобнее, но и помогает экономнее расходовать электроэнергию.

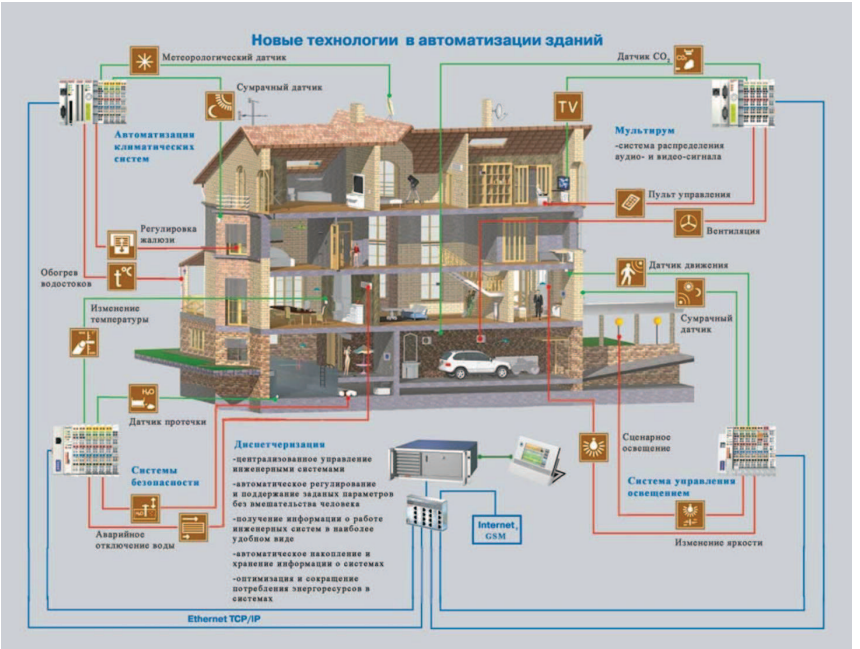


Рисунок 3 – Проект умного дома

Еще один пример бытового использования ИИ – автоматические переводчики. Если раньше качество пропущенного через них текста оставляло желать много лучшего, то сейчас ситуация меняется. Алгоритмы учатся подбирать правильный перевод в зависимости от контекста и согласовывать части предложения между собой. Как итог, вместо «машинного перевода» можно получить вполне читаемый текст. Внедрение ИИ в свой переводчик осенью 2017 года анонсировал «Яндекс». Алгоритм не разбивает текст на отдельные слова, а воспринимает предложение целиком, что позволяет получить текст более высокого качества.

Опасность искусственного интеллекта

Развитие искусственного интеллекта однажды приведёт к превосходству его над умственными способностями человека. Однако не станет ли это опасным для человечества? Изучить ситуацию можно в результате более точного определения понятия ИИ, взяв за основу для сравнения естественный интеллект. Может ли в одном человеке сочетаться ум и интеллект одновременно? Или же умный человек не может быть интеллектуалом и наоборот?

Такие вопросы возникают в связи с приближением эры ИИ, о возможной опасности которого человечество должно знать заранее и своевременно принять меры для обеспечения своей безопасности. В первую очередь

опасность ИИ будет связана с его самостоятельностью и неконтролируемым принятием решений. На данный момент уже выделены средства для изучения этой проблемы. В институте OpenAI изучаются перспективы развития ИИ. На теперешней стадии развития систем ИИ опасность его применения может быть связана со следующим факторами: ошибками ПО (программного обеспечения). Любому программному обеспечению может угрожать такая опасность; самостоятельной активностью ИИ, вредной для человека. Опасность от ИИ может исходить после изобретения умного компьютера. Поэтому необходимо определиться со степенью ума для компьютера, которая способна быть допустимой, а также чрезмерной, представляющей опасность для человека. Эти свойства должны быть точно измерены из-за сближения умственных способностей человека и компьютера, которые неизбежны. Информационные системы, существующие сегодня, представляют собой человеко-машины, которые могут работать благодаря интеллекту пользователя либо эксперта по компьютерам.

Например, какая опасность будет исходить от интеллектуальной бухгалтерской системы, которой может выдаваться неверная информация? Опасность может возникнуть, когда у такой системы появятся элементы личности, например, собственная заинтересованность, не имеющая ничего общего с человеческой. Решением такой проблемы может быть запрет на создание систем, отличающихся возможностью эволюции.

Кроме того, опасность может быть связана с содержанием в ИИ логических ошибок. Он может использоваться для решения достаточно сложных задач, список которых сразу неизвестен. Поэтому должны быть предусмотрены специальные меры для подтверждения правильности решение, которое будет получено. Скорее всего, возникнет необходимость в разработке всевозможных способов контроля таких систем, например, специальных программных средств, которыми будет автоматически проверяться правильность решения и при этом участие человека не потребуется.

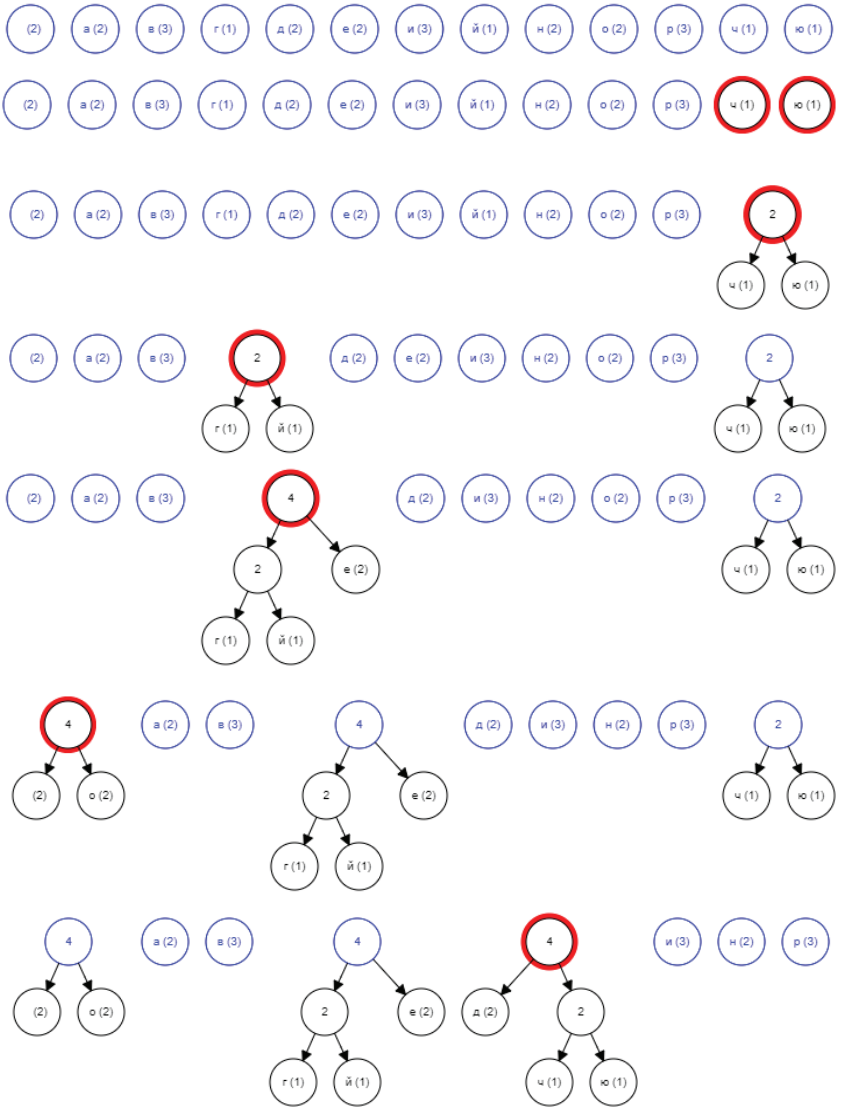
Приложение 6

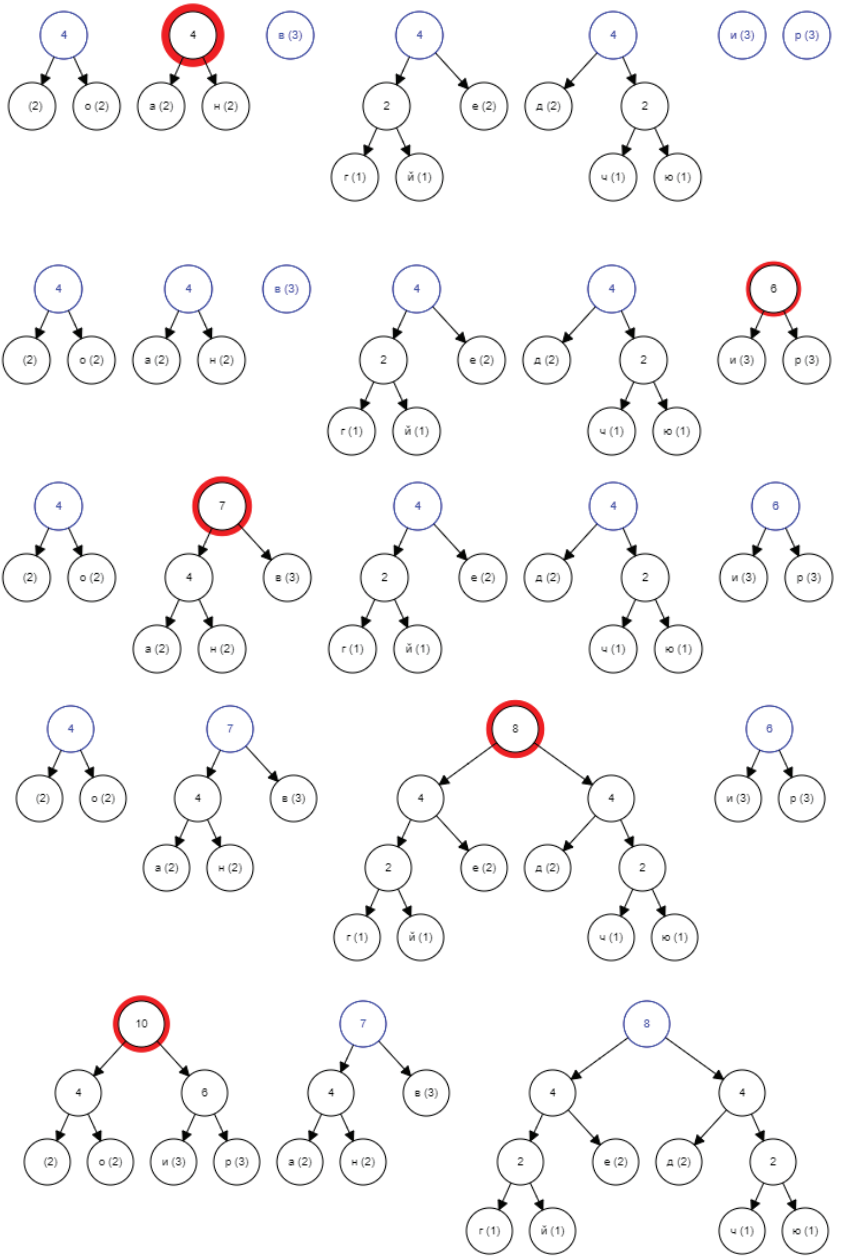
Задание по теме
«Алгоритм кодирования Хаффмана»

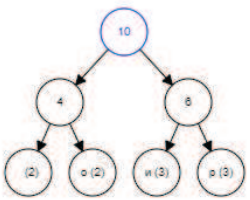
Задача - провести кодирование ФИО по алгоритму Хаффмана (алгоритм сжатия). Идея, положенная в основу кодировании Хаффмана, основана на частоте появления символа в последовательности. Символ, который встречается в последовательности чаще всего, получает новый очень маленький код, а символ, который встречается реже всего, получает, наоборот, очень длинный код. Это нужно, так как мы хотим, чтобы, когда мы обработали весь ввод, самые частотные символы заняли меньше всего места (и меньше, чем они занимали в оригинале), а самые редкие – побольше (но так как они редкие, это не имеет значения). Составляем таблицу повторяемости букв. Символ - это буквы и пробелы междц словами в указанном фамилии, имени и отчестве.

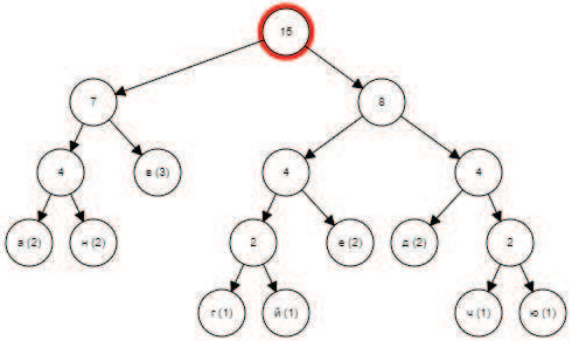
Виноградов Юрий Андреевич

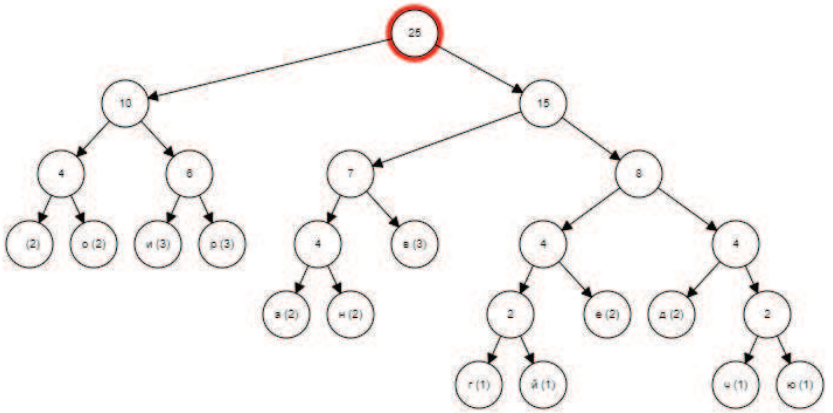
Символ	Частота
'р'	3
'и'	3
'в'	3
'о'	2
'н'	2
‘д’	2
‘а’	2
'е'	2
Space	2
‘ю’	1
‘г’	1
‘ч’	1
‘й’	1


(2) а (2) в (3) г (1) д (2) е (2) и (3) й (1) н (2) о (2) р (3) ч (1) ю (1)
(2) а (2) в (3) г (1) д (2) е (2) и (3) й (1) н (2) о (2) р (3) ч (1) ю (1)
(2) а (2) в (3) г (1) д (2) е (2) и (3) й (1) н (2) о (2) р (3) 2 ч (1) ю (1)
(2) а (2) в (3) 2 д (2) е (2) и (3) н (2) о (2) р (3) 2 г (1) й (1) ч (1) ю (1)
(2) а (2) в (3) 4 д (2) и (3) н (2) о (2) р (3) 2 2 е (2) ч (1) ю (1) г (1) й (1)
4 а (2) в (3) 4 д (2) и (3) н (2) р (3) 2 (2) о (2) 2 е (2) ч (1) ю (1) г (1) й (1)
4 а (2) в (3) 4 4 и (3) н (2) р (3) (2) о (2) 2 е (2) д (2) 2 г (1) й (1) ч (1) ю (1)


4
(2)
о (2)
4
а (2)
н (2)
в (3)
4
2
е (2)
г (1)
й (1)
4
д (2)
2
ч (1)
ю (1)
и (3)
р (3)
4
(2)
о (2)
4
а (2)
н (2)
в (3)
4
2
е (2)
г (1)
й (1)
4
д (2)
2
ч (1)
ю (1)
6
и (3)
р (3)
4
(2)
о (2)
7
4
в (3)
а (2)
н (2)
4
2
е (2)
г (1)
й (1)
4
д (2)
2
ч (1)
ю (1)
6
и (3)
р (3)
4
(2)
о (2)
7
4
в (3)
а (2)
н (2)
8
4
2
е (2)
г (1)
й (1)
4
д (2)
2
ч (1)
ю (1)
6
и (3)
р (3)
10
4
(2)
о (2)
6
и (3)
р (3)
7
4
а (2)
н (2)
в (3)
8
4
2
е (2)
г (1)
й (1)
4
д (2)
2
ч (1)
ю (1)


10
4
6
(2)
о (2)
и (3)
р (3)


15
7
8
4
в (3)
4
4
в (2)
н (2)
2
е (2)
д (2)
2
г (1)
й (1)
ч (1)
ю (1)


25
10
15
4
6
7
8
(2)
о (2)
и (3)
р (3)
4
в (3)
4
4
в (2)
н (2)
2
е (2)
д (2)
2
г (1)
й (1)
ч (1)
ю (1)

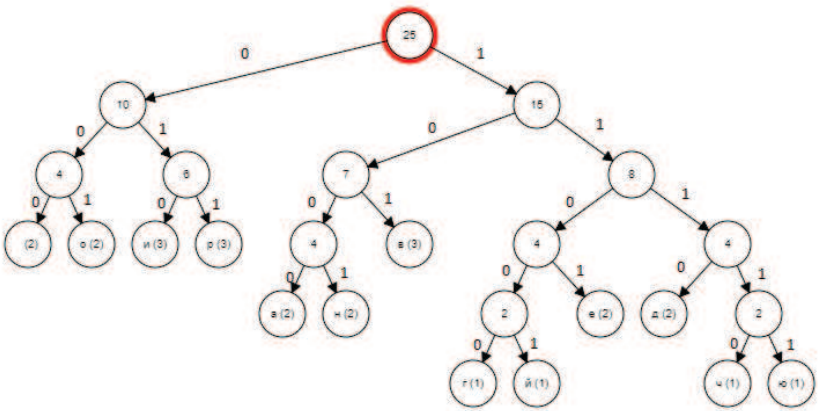

25
0
1
10
15
0
1
0
1
4
6
7
8
0
1
0
1
0
1
0
1
(2)
о (2)
и (3)
р (3)
4
в (3)
4
4
0
1
0
1
0
1
в (2)
н (2)
2
е (2)
д (2)
2
0
1
0
1
г (1)
й (1)
ч (1)
ю (1)

Таблица перевода ФИО в код

Символ	Код
'р'	011
'и'	010
'в'	101
'о'	001
'н'	1001
‘д’	1110
‘а’	1000
'е'	1101
Space	000
‘ю’	11111
‘г’	11000
‘ч’	11110
‘й’	11001

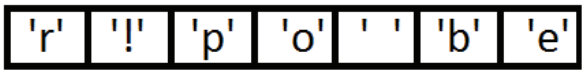
Детальное пояснение: Входная строка: «beep boop beer!», чтобы построить дерево, мы воспользуемся слегка модифицированной очередью с приоритетами – первыми из неё будут извлекаться элементы с наименьшим приоритетом, а не наибольшим. Это нужно, чтобы строить дерево от листьев к корню.

Для начала посчитаем частоты всех символов:

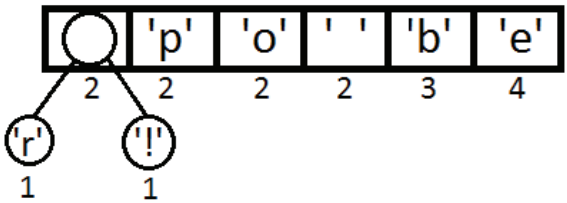
Символ	Частота
'b'	3
'e'	4
'p'	2
' '	2

'o'	2
'r'	1
'!'	1

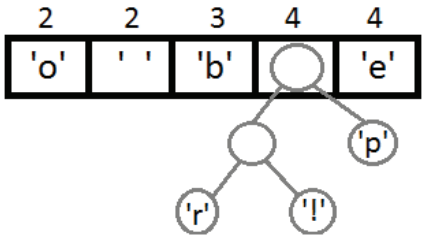
После вычисления частот мы создадим узлы бинарного дерева для каждого знака и добавим их в очередь, используя частоту в качестве приоритета:

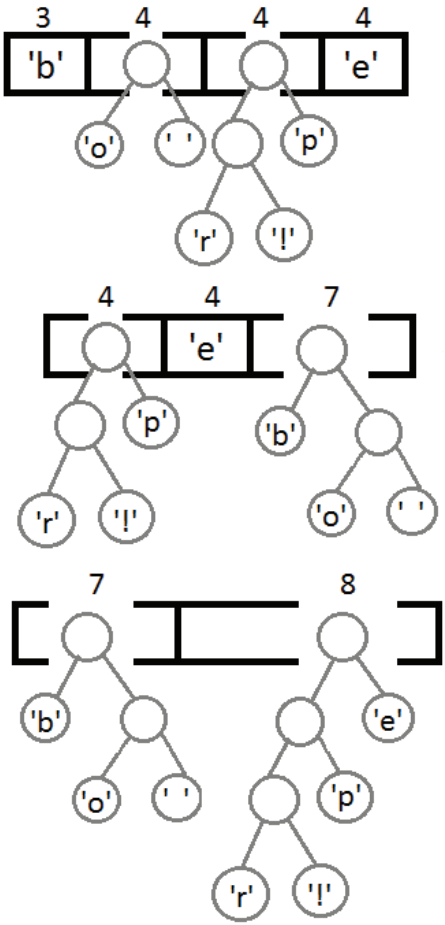


Теперь мы достаём два первых элемента из очереди и связываем их, создавая новый узел дерева, в котором они оба будут потомками, а приоритет нового узла будет равен сумме их приоритетов. После этого мы добавим получившийся новый узел обратно в очередь.

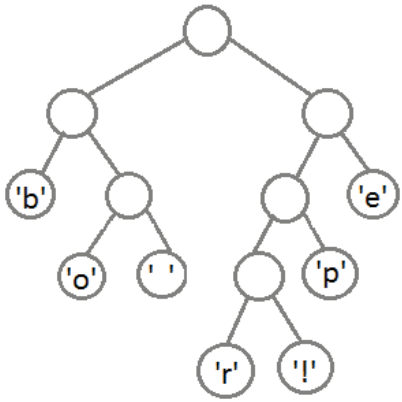


Повторим те же шаги и получим последовательно:

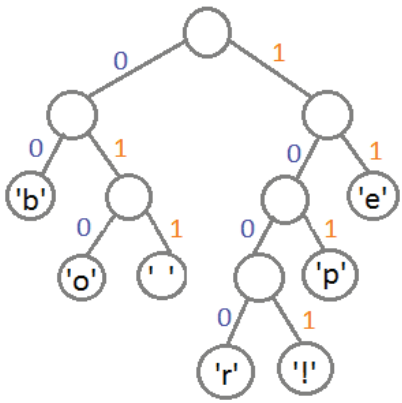




после того, как мы свяжем два последних элемента, получится итоговое дерево:



Теперь, чтобы получить код для каждого символа, надо просто пройтись по дереву, и для каждого перехода добавлять 0, если мы идём влево, и 1 – если направо:



Если мы так сделаем, то получим следующие коды для символов:

Символ	Код
'b'	00
'e'	11
'p'	101
' '	011

'o'	010
'r'	1000
'!'	1001

Чтобы расшифровать закодированную строку, нам надо, соответственно, просто идти по дереву, сворачивая в соответствующую каждому биту сторону до тех пор, пока мы не достигнем листа. Например, если есть строка «101 11 101 11» и наше дерево, то мы получим строку «pepe». Важно иметь в виду, что каждый код не является префиксом для кода другого символа. В нашем примере, если 00 – это код для 'b', то 000 не может оказаться чьим-либо кодом, потому что иначе мы получим конфликт. Мы никогда не достигли бы этого символа в дереве, так как останавливались бы ещё на 'b'. На практике, при реализации данного алгоритма сразу после построения дерева строится таблица Хаффмана. Данная таблица – это по сути связный список или массив, который содержит каждый символ и его код, потому что это делает кодирование более эффективным. Довольно затратно каждый раз искать символ и одновременно вычислять его код, так как мы не знаем, где он находится, и придётся обходить всё дерево целиком. Как правило, для кодирования используется таблица Хаффмана, а для декодирования – дерево Хаффмана.

Входная строка: «beep boop beer!»

Входная строка в бинарном виде: «0110 0010 0110 0101 0110 0101 0111 0000 0010 0000 0110 0010 0110 1111 0110 1111 0111 0000 0010 0000 0110 0010 0110 0101 0110 0101 0111 0010 0010 000»

Закодированная строка: «0011 1110 1011 0001 0010 1010 1100 1111 1000 1001»

Как вы можете заметить, между ASCII-версией строки и закодированной версией существует большая разница.

Учебное издание

Веретехина Светлана Валерьевна,
Симонов Владимир Львович,
Мнацаканян Ольга Леонидовна

Модели, методы, алгоритмы и программные решения вычислительных машин, комплексов и систем

Учебник

Издание второе

Текст приводится в авторской редакции.

Ответственный редактор *Ю. Барабанщикова*
Верстальщик *Е. Семенова*

Издательство «Директ-Медиа»
117342, Москва, ул. Обручева, 34/63, стр. 1
Тел/факс + 7 (495) 334-72-11
E-mail: manager@directmedia.ru
www.biblioclub.ru
www.directmedia.ru