Варианты **классификации ЭВМ**

Таблица 1.1 – Поколения ЭВМ

Поколение	Элементная база	Годы существования	Области применения
Первое	Электронные лампы	50 – 60	Научно-технические расчеты
Второе	Транзисторы, ферритовые сердечники	60 – 70	Научно-технические расчеты, планово-экономические расчеты
Третье	Интегральные схемы	70 – 80	Научно-технические расчеты, планово-экономические расчеты, системы управления
Четвертое	СИС, БИС, СБИС и т.д.	80 и по сей день	Все сферы деятельности

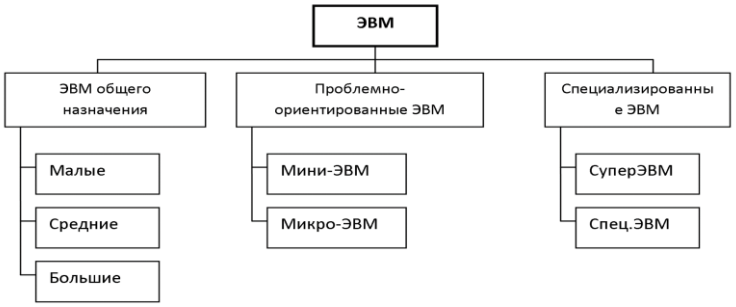


Рисунок 1.2 – Вариант классификации ЭВМ

1.3. Понятие вычислительного процесса

Вычислительная система - это взаимосвязанная совокупность аппаратных средств вычислительной техники и программного обеспечения, предназначенная для обработки информации.

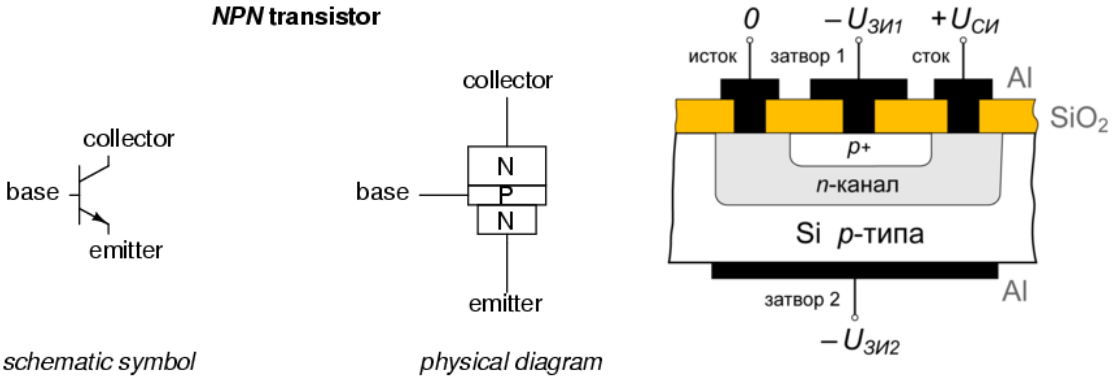
Различают следующие группы вычислительных систем:

- однозадачные и многозадачные;
- индивидуального или коллективного использования;
- с пакетной обработкой задач, с разделением времени между задачами и системы реального времени;
- однопроцессорные, многопроцессорные и многомашинные;
- универсальные, специализированные и проблемно-ориентированные.

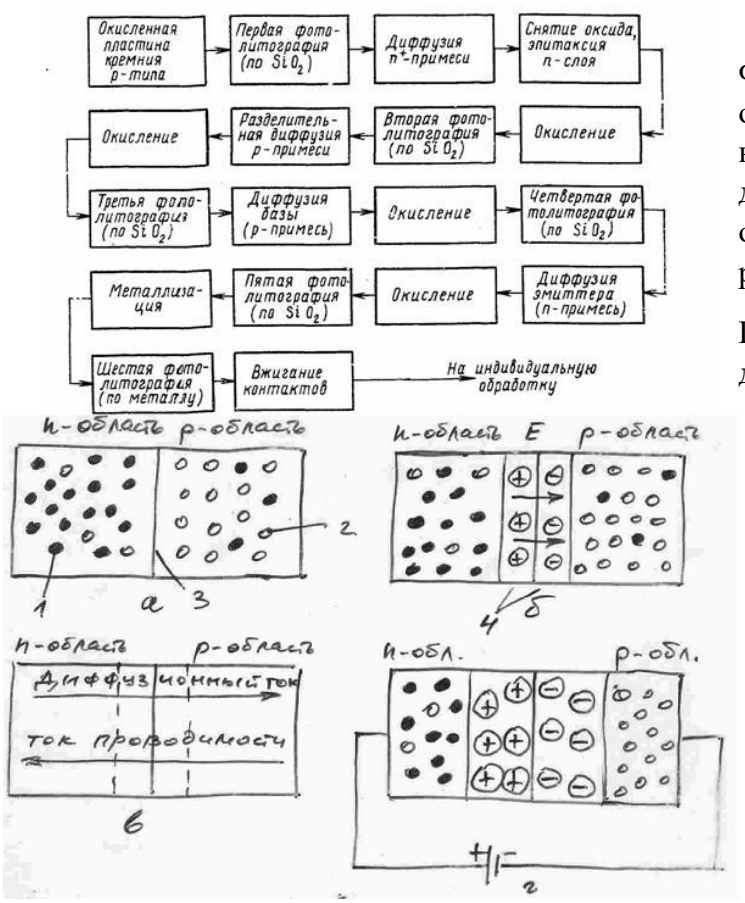
Вычислительный процесс как понятие связан с вычислительной системой и ее свойствами. Некоторые авторы определяют вычислительный процесс как *смену состояний вычислительной системы во времени, задаваемую программой.*

Любая программа, предназначенная для обработки данных, поступает через интерфейс (пользовательский или обмена данных). Принцип разработки программ может базироваться либо на потоке этих данных, либо на потоке команд, под требования которого следует реорганизовать поток этих данных.

Производство полупроводниковых приборов



Эмитор образует Фосфор, а базу – Бор, коллектор – кремниевая пластина



Фиксированная процедура обработки данных - сродни конвейеру. Вне зависимости от типа данных на входе, процедура обработки одинакова и правильный результат ничем не гарантируется.

При «управлении от потока данных»

необходимо изначально загрузить их в память, определить тип, выбрать команду для их обработки и полученный результат разместить по требуемому адресу.

При «управлении от потока команд»

изначально загружается команда, которая уже предписывает структуру данных, которые она может обработать. Такая модель более эффективна, чем предыдущая, поскольку не нужно анализировать тип данных. Если заранее сгруппировать блоки данных по типу, то можно поточно обработать большой массив данных. Недостаток системы управления от потока команд - сложность организации, обработки случайной последовательности событий.

Производители современных вычислительных систем, как правило, реализуют одну из двух вычислительных архитектур с управлением от потока команд:

CISC-архитектура (Complete Instruction Set Computer) ориентирована на выполнение большого набора команд и развитыми возможностями адресации, давая процессу возможность выбрать наиболее подходящую команду для выполнения необходимой операции. Выборка команды на исполнение осуществляется побайтно в течение нескольких циклов работы МК. Время выполнения команды может составлять от 1 до 12 циклов.

В *RISC-архитектуре* (Reduced Instruction Set Computer) набор исполняемых команд сокращен до минимума. Для реализации более сложных операций приходится комбинировать команды. При этом все команды имеют формат фиксированной длины (например, 12, 14 или 16 бит), выборка команды из памяти и ее исполнение осуществляется за один цикл (такт) синхронизации. Система команд RISC-процессора предполагает возможность равноправного использования всех регистров процессора. Это обеспечивает дополнительную гибкость при выполнении ряда операций.

Оптимизация вычислительного процесса может производиться несколькими способами:

-увеличение мощности вычислительной системы под требования рабочей нагрузки;

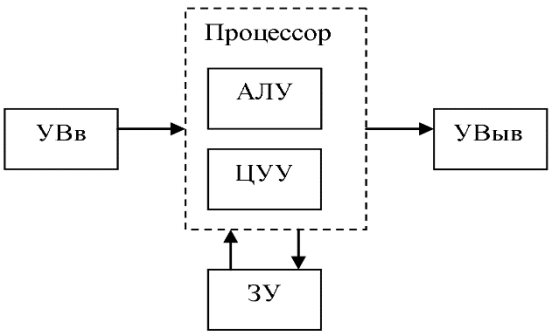
- перераспределение рабочей нагрузки в течение сеанса работы;

-изменение состава рабочей нагрузки с учетом возможностей оборудования.

1.4. Классическая архитектура ЭВМ

Считается, что основные идеи построения современных ЭВМ в 1945 г. Сформулировал американский математик Дж. фон Нейман, определив их как *принципы программного управления*:

1) информация кодируется в двоичной форме и разделяется на единицы – слова.
2) разнотипные по смыслу слова различаются по способу использования, но не по способу кодирования.
3) Слова информации размещаются в ячейках памяти и идентифицируются номерами ячеек – адресами слов.
4) Алгоритм представляется в форме последовательности управляющих слов, называемых *командами*. Команда определяет наименование операции и слова информации, участвующие в ней. Алгоритм, записанный в виде последовательности команд, называется *программой*.
5) Выполнение вычислений, предписанных алгоритмом, сводится к последовательному выполнению команд в порядке, однозначно определенном программой.



Программа вычислений (обработки информации) составляется в виде последовательности команд и загружается в память машины – *запоминающее устройство* (ЗУ). Там же хранятся исходные данные и промежуточные результаты обработки.

Центральное устройство управления (ЦУУ) последовательно извлекает из памяти команды программы и организует их выполнение.

Арифметико-логическое устройство (АЛУ) предназначено для реализации операций преобразования информации.

Программа и исходные данные вводятся в память машины через *устройства ввода* (УВв), а результаты обработки предъявляются на *устройства вывода* (УВыв).

Характерной особенностью архитектуры фон Неймона является то, что память представляет собой единое адресное пространство, предназначенное для хранения, как программ, так и данных.

Такой подход, с одной стороны, обеспечивает большую гибкость организации вычислений – возможность перераспределения памяти между программой и данными, возможность самомодификации программы в процессе её выполнения.

С другой стороны, без принятия специальных мер защиты снижается надежность выполнения программы, что особенно не допустимо в управляющих системах.

Действительно, поскольку и команды программы, и данные кодируются в ЭВМ двоичными числами, теоретически возможно как разрушение программы (при обращении

в область программы как к данным), так и попытка «выполнения» области данных как программы (при ошибочных переходах программы в область данных).

Альтернативной фон-неймановской является т.н. *гарвардская архитектура*.

ЭВМ, реализованные по этому принципу, имеют два непересекающихся адресных пространства – для программы и для данных, причем программу нельзя разместить в свободной области памяти данных и наоборот.

Гарвардская архитектура применяется главным образом в управляющих ЭВМ.

2 ОБОРУДОВАНИЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Компьютерная система состоит из компонентов оборудования и программного обеспечения.

Оборудование – это набор физических компонентов. В его состав входят корпус, приводы систем хранения, клавиатуры, мониторы, кабели, динамики и принтеры. Программное обеспечение – это операционная система и программы.

Операционная система управляет работой компьютера. В процессе работы может выполняться определение информации, доступ к ней и ее обработка.

Программы, или приложения, выполняют различные функции. Программы могут различаться в зависимости от информации, которую они получают или создают. Например, команды для сведения баланса по чековой книжке отличаются от команд для создания виртуального мира в Интернете.

Компьютер способен работать при незначительных колебаниях напряжения сети, однако значительное отклонение напряжения может вызвать сбой блока питания.

Источник бесперебойного питания (ИБП) может защитить компьютер от проблем, вызванных изменением напряжения электропитания. В ИБП используется **инвертор**. Инвертор *подает переменный ток* на ПК со встроенного аккумулятора, преобразуя постоянный ток с аккумулятора ИБП в переменный. Встроенный аккумулятор непрерывно заряжается постоянным током, в который преобразуется переменный ток от сети.

Разъемы

Большинство современных разъемов имеют ключ. Разъем с ключом спроектирован так, чтобы его невозможно было вставить неправильно. В каждом разъеме блока питания используется разное напряжение. Каждый разъем предназначен для подключения определенных компонентов к различным портам материнской платы.

Разъем с ключом **Molex** предназначен для оптических приводов, жестких дисков или других устройств, использующих более старые технологии.

Разъем с ключом **Berg** предназначен для привода гибких дисков. Разъем Berg меньше, чем разъем Molex.

Разъем с ключом **SATA** предназначен для оптических приводов или жестких дисков. Разъем SATA шире и тоньше, чем разъем Molex.

20-штырьковый или 24-штырьковый разъем подключается к материнской плате. 24-штырьковый разъем имеет 2 ряда по 12 штырьков, а 20-штырьковый разъем имеет 2 ряда по 10 штырьков.

Дополнительный 4- или 8-штырьковый разъем питания имеет 2 ряда по 2 или 4 штырька и подает питание на все области материнской платы. Дополнительный разъем питания имеет ту же форму, что и основной, но его размер меньше. Он может также подавать питание и на другие устройства компьютера.

6/8-штырьковый разъем питания **PCIe** (англ. **Peripheral component interconnect** Express) имеет два ряда по три или четыре штырька и подает питание на другие внутренние компоненты.

В блоках питания более ранних стандартов для подключения к материнской плате использовались два разъема, называемые P8 и P9. Разъемы P8 и P9 не имели ключей. Их можно было менять местами, что могло нанести вред материнской плате или блоку питания. По правилам установки требовалось, чтобы черные провода посередине располагались рядом.

Напряжение	Цвет провода	Использование	Стандарт блока питания		
			AT	ATX	ATX12V
+12V	Желтый	Двигатели дисковых приводов, вентиляторы, охлаждающие устройства и гнезда системной шины	*	*	*
-12V	Синий	Некоторые виды схем последовательных портов и ранние типы программируемой памяти «только для чтения» (ЭППТЧ)	*	*	*
+3.3V	Оранжевый	Большинство современных ЦП, некоторые виды системной памяти и видеокарты AGP		*	*
+5V	Красный	Материнская плата, Baby AT и ранние типы ЦП, а также многие компоненты материнской платы	*	*	*
-5V	Белый	Карты шины ISA и ранние типы программируемой памяти «только для чтения» (ЭППТЧ)	*	*	*
0V	Черный	Земля — контакт для создания полной цепи.	*	*	*

В электрической цепи существуют 4 основных понятия:

- Напряжение (V)
- Сила тока (I)
- Мощность (P)
- Сопротивление (R)

Напряжение, сила тока, мощность и сопротивление являются понятиями электротехники, которые должен знать любой специалист по компьютерам.

- Напряжение – величина работы, необходимой для перемещения электронов по цепи. Напряжение измеряется в вольтах (В). Блок питания подает напряжение нескольких значений.
- Сила тока – это величина, соответствующая количеству электронов, перемещающихся по цепи. Сила тока измеряется в амперах (А). Блок питания компьютера подает различный ток на линии с разным выходным напряжением.
- Мощность – это величина работы, необходимой для перемещения электронов по электрической цепи (напряжение), умноженная на количество электронов, перемещающихся по этой цепи в секунду (ток). Эта величина изменяется в ваттах (Вт). Компьютерные блоки питания характеризуются по мощности.
- Сопротивление – физическая величина, характеризующая свойство проводника препятствовать прохождению электрического тока. Измеряется в Омах. Более низкое сопротивление позволяет проходить по цепи большему току и, соответственно, большему количеству электроэнергии. Хороший предохранитель имеет низкое сопротивление, почти равное 0 Ом.

В основном уравнении, известном также как **закон Ома**, показаны взаимоотношения этих трех величин. Закон Ома гласит, что напряжение равно произведению силы тока и сопротивления: **V = IR**.

В электросистеме мощность равна произведению напряжения и силы тока: **P = VI**.

Повышение силы тока или напряжения в электрической цепи приводит к увеличению мощности.

Пример: простая цепь, состоящая из лампы накаливания, рассчитанной на напряжение 9 В, подключенной к батарее с напряжением 9 В. Мощность, рассеиваемая лампой

накаливания, составляет 100 Вт. Используя уравнение P = VI, можно подсчитать, какая сила тока в амперах необходима для того, чтобы лампа, рассчитанная на напряжение 9 В, работала с мощностью 100 Вт.

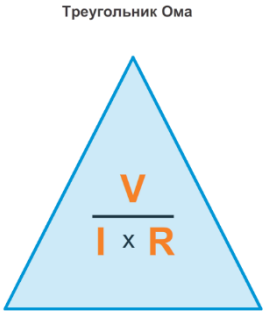
Решим это уравнение. Нам известно, что P = 100 Вт, а V = 9 В.

I = P/V = 100 Вт / 9 В = 11,11 А

Что произойдет, если для получения мощности 100 Вт используются лампа накаливания, рассчитанная на напряжение 12 В, и источник питания с напряжением 12 В?

I = P/V = 100 Вт / 12 В = 8,33 А

Система производит ту же мощность, но сила тока меньше



Для подсчета напряжения, силы тока или сопротивления, можно использовать треугольник Ома.

Если две оставшиеся величины известны.

Чтобы увидеть необходимую формулу, закройте неизвестную переменную и выполните подсчет по полученной формуле.

Например, если известны напряжение и сила тока, закройте R, чтобы получить формулу V / I.

Подсчитайте V / I, чтобы найти R.

Можно использовать схему «Закон Ома», показанную на следующем рисунке, для подсчета любой из четырех основных величин электрической цепи при наличии двух известных величин.

Материнская плата – это основная печатная плата компьютера, на которой находятся шины, или электрические проводники. Эти шины обеспечивают передачу данных между различными компонентами компьютера. Материнская плата также называется системной, или основной платой.

На материнскую плату с токопроводящими линиями, соединяющими компоненты платы, напаяны разъемы расширения, микросхема базовой системы ввода-вывода (BIOS), чипсет (набор микросхем сопровождения) и с помощью специальных приспособлений устанавливаются центральный процессор (ЦП), оперативное запоминающее устройство (ОЗУ), радиатор и вентилятор.

Гнезда, внешние и внутренние разъемы и различные порты также расположены на материнской плате

Форм-фактор материнских плат – их размер и форма

Он также характеризует физическое расположение компонентов и устройств на плате.

Форм-факторы		
AT	Advanced Technology	30,5 см X 35,1 см
ATX	Advanced Technology Extended	30,5 см X 24,4 см
Мини-ATX	Вариант ATX с меньшей занимаемой площадью	15 см X 15 см
Микро-ATX	Вариант ATX с меньшей занимаемой площадью	24,4 см X 24,4 см
LPX	Low-Profile Extended	33 см X 22,9 см
NLX	New Low-Profile Extended	от 20,3 см X 25,4 см до 22,9 см X 34,5 см
BTX	Balanced Technology Extended	32,5 см X 26,6 см
Мини-ITX	Формат компактнее, чем микро-ATX	17 см X 17 см
Нано-ITX	Вариант мини-ITX с меньшей занимаемой площадью	12 см X 12 см

Форм-фактор определяет форму корпуса компьютера и способ присоединения отдельных компонентов к ней. На рисунке показаны различные существующие форм-факторы для материнских плат.

Наиболее распространенным форм-фактором настольных ПК был AT, основанный на материнской плате IBM AT. Ширина материнской платы AT может достигать приблизительно 26 см. Этот неудобный размер привел к разработке меньших форм-факторов. Местоположение радиаторов и вентиляторов часто мешает использованию гнезд расширения в форм-факторах меньшего размера.

Более новый форм-фактор для материнских плат, ATX, стал улучшением конструкции AT. В корпусе стандарта ATX размещаются интегрированные порты ввода-вывода материнской платы ATX. Блок питания ATX подключается к материнской плате через один 20-штырьковый разъем, а не через одинаковые разъемы P8 и P9, которые использовались в более ранних форм-факторах. Блок питания ATX можно включать и выключать не тумблером, а с помощью сигналов с материнской платы.

Micro-ATX – это форм-фактор меньшего размера, обратно совместимый с ATX. Поскольку точки монтажа материнской платы Micro-ATX являются модификацией тех, что использовались на плате ATX, а панель ввода-вывода идентична на обеих платах, материнскую плату Micro-ATX можно устанавливать в полноразмерные корпуса стандарта ATX.

Форм-фактор ITX получил широкое распространение благодаря своему очень компактному размеру. Существует множество типов материнских плат форм-фактора ITX. Наиболее популярным типом является Mini-ITX. Форм-фактор Mini-ITX использует очень малую мощность, поэтому для его охлаждения не нужны вентиляторы.

Материнская плата типа Mini-ITX имеет только одно гнездо PCI для плат расширения. Компьютер на основе форм-фактора Mini-ITX можно использовать в тех случаях, когда громоздкий или шумный компьютер вызывает неудобства.

Важным набором компонентов материнской платы является чипсет (набор микросхем сопровождения). Чипсет состоит из разнообразных интегральных микросхем, напаянных на материнскую плату. Они управляют взаимодействием системного оборудования с ЦП и материнской платой. ЦП устанавливается в разъем или гнездо на материнской плате. Тип устанавливаемого ЦП определяется разъемом на материнской плате.

Чипсет позволяет ЦП связываться и взаимодействовать с другими компонентами компьютера, а также обмениваться данными с системной памятью (ОЗУ), жесткими дисками, видеокартами и другими устройствами вывода. Объем памяти, которую можно добавить к материнской плате, зависит от чипсета.

Кроме того, от чипсета также зависит тип разъемов на материнской плате.

Большинство чипсетов разделяются на два различных компонента: контроллер-концентратор памяти, или северный мост, и контроллер-концентратор ввода-вывода, или южный мост. Функции каждого из этих компонентов могут различаться в зависимости от производителя.

Северный мост управляет доступом к **ОЗУ**, **видеокарте**, а также скоростью их связи с ЦП. В ряде случаев *видеокарта* бывает встроена в северный мост. AMD и Intel производят микросхемы с контроллером памяти, интегрированным в кристалл ЦП. Это улучшает производительность системы и оптимизирует энергопотребление.

Южный мост в большинстве случаев отвечает за взаимодействие ЦП с жестким диском, звуковой картой, портами USB и другими портами ввода-вывода.

3 ФУНКЦИОНАЛЬНАЯ ОРГАНИЗАЦИЯ ЭВМ

3.1 Командный цикл процессора

Термин *«функциональная организация ЭВМ»* часто используют в качестве синонима (в некотором смысле) более широкого термина *«архитектура ЭВМ»*, который в свою очередь, трактуется разными авторами несколько в различных смыслах.

Наиболее близким может служить определение термина «Архитектура ЭВМ», данное в книжке Савельева Г.Н. Прикладная теория цифровых автоматов.

Архитектура ЭВМ – это абстрактное представление ЭВМ, которое

отражает ее структурную, схемотехническую и логическую организацию.

Понятие архитектуры ЭВМ является комплексным и включает в себя:

- Структурную схему ЭВМ;
- Средства и способы доступа к элементам структурной схемы;
- Организацию и разрядность интерфейсов ЭВМ;
- Набор и доступность регистров;
- Организацию и способы адресации памяти;
- Способы представления и форматы данных ЭВМ;
- Набор машинных команд ЭВМ;
- Форматы машинных команд;
- Обработку нештатных ситуаций (прерываний).

К числу общих архитектурных свойств и принципов можно отнести:

- 1) *Принцип хранимой программы*. Согласно ему, код программы и её данные находятся в одном адресном пространстве в оперативной памяти.
- *2) Принцип микропрограммирования*. Суть этого принципа заключается в том, что машинный язык всё-таки ещё не является той конечной субстанцией, которая физически приводит в действие процессы в машине. В состав процессора входит блок *микропрограммного управления*. Этот блок для каждой машинной команды имеет набор действий-сигналов, которые нужно сгенерировать для физического выполнения требуемой машинной команды. Здесь уместно вспомнить характеристику ЭВМ 1-го поколения. В них для генерации нужных сигналов необходимо было осуществить ручное всех логических схем.
- *3) Линейное пространство памяти*. – совокупность ячеек памяти, которым последовательно присваиваются номера (адреса) 0,1,2,….
- *4) Последовательное выполнение программ.* Процессор выбирает из памяти команды строго последовательно. Для изменения прямолинейного хода выполнения программы или осуществления ветвления необходимо использовать специальные команды. Они называются командами условного и безусловного перехода.

- *5)* С точки зрения процессора, нет принципиальной разницы между данными и командами. Данные и машинные команды находятся в одном пространстве памяти в виде последовательности нулей и единиц. Это свойство связано с предыдущим. Процессор, исполняя содержимое некоторых последовательных ячеек памяти, всегда пытается трактовать его как коды машинной команды, а если не так, то происходит аварийное завершение программы, содержащей некорректный фрагмент. Поэтому важно в программе всегда четко разделять пространство данных и команд
- *6) Безразличие к целевому назначению данных.* Машине всё равно, какую логическую нагрузку несут обрабатываемые ею данные.

Командный цикл процессора

В состав процессора входят следующие функциональные узлы:

- *микропроцессорное ядро;*
- *подсистема основной оперативной памяти;*
- *ROM содержащая BIOS;*
- подсистема клавиатуры;
- подсистема прямого доступа к памяти;
- подсистема прерываний;
- таймер;
- часы/календарь реального времени.

Командой называется элементарное действие, которое может выполнить процессор без дальнейшей детализации. Последовательность команд, выполнение которых приводит к достижению определённой цели, называется *программой*. Команды программы кодируются двоичными словами и размещаются в памяти ЭВМ. Вся работа ЭВМ состоит в последовательном выполнении команд программы. Действия по выбору из памяти и выполнению одной команды называется *командным циклом.*

В составе любого процессора имеется специальная ячейка, которая хранит адрес выполняемой команды – *счетчик команд* или *программный счетчик*. После выполнения очередной команды его значение увеличивается на единицу (если код одной команды занимает несколько ячеек памяти, то содержимое счетчика команд увеличивается на длину команды). Таким образом осуществляется выполнение последовательности команд. Существуют специальные команды (передачи управления), которые в процессе своего выполнения модифицируют содержимое программного счетчика, обеспечивая переходы по программе. Сама выполняемая программа помещается в *регистр команд* – специальную ячейку процессора.
Этапы цикла выполнения:

1. Процессор выставляет число, хранящееся в регистре счётчика команд, на шину адреса, и отдаёт памяти команду чтения;

2. Выставленное число является для памяти адресом; память, получив адрес и команду чтения, выставляет содержимое, хранящееся по этому адресу, на шину данных, и сообщает о готовности;
3. Процессор получает число с шины данных, интерпретирует его как команду (машинную инструкцию) из своей системы команд и исполняет её;
4. Если последняя команда не является командой перехода, процессор увеличивает на единицу (в предположении, что длина каждой команды равна единице) число, хранящееся в счётчике команд; в результате там образуется адрес следующей команды;
5. Снова выполняется п. 1.

Данный цикл выполняется неизменно, и именно он называется *процессом* (откуда и произошло название устройства).

Во время процесса процессор считывает последовательность команд, содержащихся в памяти, и исполняет их.

Такая последовательность команд представляет *алгоритм* работы процессора. Очерёдность считывания команд изменяется в случае, если процессор считывает команду перехода — тогда адрес следующей команды может оказаться другим. Другим примером изменения процесса может служить случай получения команды останова или переключение в режим обработки прерывания

Команды центрального процессора являются самым нижним уровнем управления компьютером, поэтому выполнение каждой команды неизбежно и безусловно.

Не производится никакой проверки на допустимость выполняемых действий, в частности, не проверяется возможная потеря ценных данных. Чтобы компьютер выполнял только допустимые действия, команды должны быть соответствующим образом организованы в виде необходимой *программы*.

Скорость перехода от одного этапа цикла к другому определяется *тактовым генератором*. Тактовый генератор вырабатывает импульсы, служащие ритмом для центрального процессора. Частота тактовых импульсов называется *тактовой частотой*.

Фактически вся работа процессора заключается в циклическом выполнении пунктов 1 – 5 командного цикла. При запуске машины в счётчик команд аппаратно помещается фиксированное значение – начальный адрес программы (часто 0 или последний адрес памяти; встречаются и более экзотические способы загрузки начального адреса). В дальнейшем содержимое программного счетчика модифицируется в командном цикле. Прекращение выполнения командных циклов может произойти только при выполнении специальной команды «СТОП».

Конвейерная архитектура (*pipelining*) была введена в центральный процессор с целью повышения быстродействия. Обычно для выполнения каждой команды требуется осуществить некоторое количество однотипных операций, например: выборка команды из ОЗУ, дешифрация команды, адресация операнда в ОЗУ, выборка операнда из ОЗУ, выполнение команды, запись результата в ОЗУ. Каждую из этих операций сопоставляют одной ступени конвейера.

Конвейер микропроцессора с архитектурой MIPS-I содержит четыре стадии:

- получение и декодирование инструкции (Fetch)
- адресация и выборка операнда из ОЗУ (Memory access)
- выполнение арифметических операций (Arithmetic Operation)
- сохранение результата операции (Store)

При отсутствии конвейера выполнение команды займёт *n* единиц времени (так как для выполнения команды по прежнему необходимо выполнять выборку, дешифрацию и т. д.), и для исполнения *m* команд понадобится единиц времени; при использовании конвейера (в самом оптимистичном случае) для выполнения *m* команд понадобится всего лишь *n* + *m* единиц времени.

3.2 Система команд процессора

Разнообразие типов данных, форм их представления и действий, которые необходимы для обработки информации и управления ходом вычислений, порождает необходимость использования различных команд – набора команд. Каждый процессор имеет собственный вполне определенный набор команд, называемый *системой команд процессора*. Система команд должна обладать двумя свойствами – *функциональной полнотой* и *эффективностью*.

Функциональная полнота – это достаточность системы команд для описания любого алгоритма. Требование функциональной полноты не является слишком жестким. Доказано, что свойством функциональной полноты обладает система, включающая всего три команды (система Поста): присвоение 0, присвоение 1, проверка на 0. Однако составление программ в такой системе команд крайне не эффективно.

Эффективность системы команд – степень соответствия системы команд назначению ЭВМ, т.е. классу алгоритмов, для выполнения которых предназначается ЭВМ, а также требованиям к производительности ЭВМ. Очевидно, что реализация развитой системы команд связана с большими затратами оборудования и, следовательно, с высокой стоимостью процессора. В тоже время ограниченный набор команд приводит к снижению производительности и повышенным требованиям к памяти для размещения программы. Даже простые и дешевые современные микропроцессоры поддерживают систему команд, содержащую несколько сотен команд.

Под ***форматом команды*** следует

понимать длину команды, количество,

размер, положение, назначение и способ

кодировки ее полей.

Машинная команда представляет собой закодированное по определенным правилам указание микропроцессору на выполнение некоторой операции или действия. Каждая команда содержит элементы, определяющие:

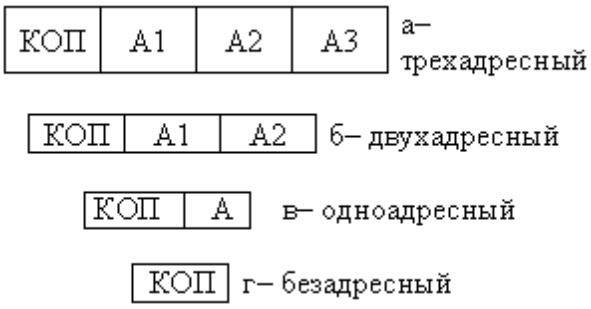
- *что делать?* (ответ на этот вопрос дает элемент команды, называемый кодом операции (КОП));
- *объекты, над которыми* нужно что-то делать (эти элементы называются **операндами**);

- *как делать?* (эти элементы называются типами операндов – обычно задаются неявно).

Команды, как и любая информация в ЭВМ, кодируются двоичными словами, которые должны содержать в себе следующие виды информации:

- тип операции, которую следует реализовать в данной команде (КОП);
- место в памяти, откуда следует взять первый операнд (А1);
- место в памяти, откуда следует взять второй операнд (А2);
- место в памяти, куда следует поместить результат (А3).

Каждому из этих видов информации соответствует своя часть двоичного слова – поле, а совокупность полей (их длины, расположение в командном слове, способ кодирования информации) называется **форматом команды**. В свою очередь, некоторые поля команды могут делиться на подполя.



Команды трехадресного формата занимают много места в памяти, в то же время далеко не всегда поля адресов используются в командах эффективно. Действительно, наряду с двухместными операциями (сложение, деление, конъюнкция и др.)встречаются и одноместные (инверсия, сдвиг, инкремент и др.), для которых третий адрес не нужен.

При выполнении цепочки вычислений часто результат предыдущей операции используется в качестве операнда для следующей. Более того, нередко встречаются команды, для которых операнды не определены (СТОП) или подразумеваются самим кодом операции (DAA, десятичная коррекция аккумулятора). Поэтому чаще используются *двухадресные команды*, в этом случае в бинарных операциях результат помещается на место одного из операндов.

Для реализации одноадресных форматов в процессоре предусматривают специальную ячейку – *аккумулятор*. Первый операнд и результат всегда размещаются в аккумуляторе, а второй операнд адресуется полем А.

Существует пять основных способов адресации операндов в командах.

- *Прямая* – в этом случае в адресном поле располагается адрес операнда. Разновидность – *прямая регистровая* адресация, адресующая не ячейку памяти, а РОН (регистр общего назначения). Поле адреса регистра имеет в команде значительно меньшую длину, чем поле адреса памяти.
- *Непосредственная* – в поле адреса команды располагается не адрес операнда, а сам операнд. Такой способ удобно использовать в командах с константами.
- *Косвенная* – в поле адреса команды располагается адрес ячейки памяти, в которой хранится адрес операнда («адрес адреса»). Такой способ позволяет оперировать адресами как данными, что облегчает организацию циклов, обработку массивов данных и др. Его основной недостаток – потеря времени на двойное обращение к памяти – сначала за адресом, потом – за операндом. Разновидность – *косвенно-регистровая* адресация, при которой в поле команды размещается адрес РОН,

хранящего адрес операнда. Этот способ, помимо преимущества обычной косвенной адресации, позволяет обращаться к большой памяти с помощью коротких команд и не требует двойного обращения к памяти (обращение к регистру занимает гораздо меньше времени, чем к памяти).

- *Относительная* – адрес формируется как сумма двух слагаемых: базы, хранящейся в специальном регистре или в одном из РОН, и смещения, извлекаемого из поля адреса команды. Этот способ позволяет сократить длину команды (смещение может быть укороченным, правда в этом случае не вся память доступна в команде) и/или перемещать адресуемые массивы информации по памяти (изменяя базу). Разновидности – *индексная* и *базово-индексная* адресации. Индексная адресация предполагает наличие индексного регистра вместо базового. При каждом обращении содержимое индексного регистра автоматически модифицируется (обычно увеличивается или уменьшается на 1). Базово-индексная адресация формирует адрес операнда как сумму трех слагаемых: базы, индекса и смещения.
- *Безадресная* – поле адреса в команде отсутствует, а адрес операнда или не имеет смысла для данной команды, или подразумевается по умолчанию. Часто безадресные команды подразумевают действия над содержимым аккумулятора. Характерно, что безадресные команды нельзя применить к другим регистрам или ячейкам памяти. Одной из разновидностей безадресного обращения является использование т.н. магазинной памяти или *стека*.

Обращение к такой памяти напоминает обращение с магазином стрелкового оружия. Имеется фиксированная ячейка, называемая *верхушкой стека*. При чтении слово извлекается из верхушки, а все остальное содержимое «поднимается вверх» подобно патронам в магазине, так что в верхушке оказывается следующее по порядку слово. Одно слово нельзя прочитать из стека дважды. При записи новое слово помещается в верхушку стека, а все остальное содержимое «опускается вниз» на одну позицию. Таким образом, слово помещенное в стек первым, будет прочитано последним. Говорят, что стек поддерживает дисциплину LIFO – Last In First Out (последний пришел – первый ушел). Реже используется безадресная память типа *очередь* с дисциплиной FIFO – First In First Out (первый пришел – первый ушел).

Единицы измерения информации служат для измерения различных характеристик, связанных с информацией.

Чаще всего измерение информации касается измерения **ёмкости** запоминающих устройств и измерения **объёма** данных, передаваемых по цифровым каналам связи. Реже измеряется количество информации.

Для удобства представления информации все возможные виды информации переводятся в числовую форму, и эти числа хранятся в компьютере в двоичном виде, т.е. кодируются.

Кодирование информации — процесс преобразования информации из формы, удобной для непосредственного использования, в форму, удобную для передачи, хранения или автоматической переработки.

Для записи текстовой (знаковой) информации всегда используется какой-либо язык (естественный или формальный).

Всё множество используемых в языке символов называется **алфавитом**. Полное число символов алфавита **N** называют его ***мощностью***. При записи текста в каждой очередной

позиции может появиться любой из **N** символов алфавита, т.е. может произойти **N** событий. Следовательно, каждый символ алфавита содержит **i** бит информации, где **i** определяется из неравенства (формула Хартли): $\mathbf{2^i \geq N}$. Тогда общее количество информации в тексте определяется формулой:

V = k * i ,

где **V** – количество информации в тексте; **k** – число знаков в тексте (включая знаки препинания и даже пробелы), **i**- количество бит, выделенных на кодирование одного знака.

Так как каждый бит – это **0** или **1**, то **любой текст может быть представлен последовательностью нулей и единиц.**

Именно так текстовая информация хранится в памяти компьютера.

Присвоение символу алфавита конкретного двоичного кода – это вопрос соглашения, зафиксированного в кодовой таблице.

В настоящее время широкое распространение получили кодовые таблицы **ASCII** и **Unicode**.

ASCII (American Standart Code for Informational Interchange – Американский стандартный код информационного обмена) используется достаточно давно. Для хранения кода одного символа выделено **8** бит, следовательно, кодовая таблица поддерживает до $\mathbf{2^8 = 256}$ символов. Первая половина таблицы (**128** символов) – управляющие символы, цифры и буквы латинского алфавита. Вторая половина отводится под символы национальных алфавитов.

К сожалению, в настоящее время существует целых пять вариантов кодовых таблиц для русских букв (КОИ-8, Windows-1251, ISO, DOS,MAC), поэтому тексты созданные в одной кодировке неверно отображаются в другой.

Unicode - используется для хранения кода одного символа, выделено **16** бит, следовательно, кодовая таблица поддерживает до

$\mathbf{2^{16} = 65536}$ символов. Такого пространства достаточно, чтобы в одном стандарте объединить все "живые" официальные (государственные) письменности.

Стандарт ASCII вошел в состав Unicode.

Преобразование графической информации из аналоговой формы в дискретную производится путем дискретизации, т.е. разбиения непрерывного графического изображения на отдельные элементы. В процессе дискретизации производится кодирование, т.е. присвоение каждому элементу конкретного значения в форме кода.

В процессе кодирования изображения производится пространственная дискретизация. Пространственную дискретизацию изображения можно сравнить с построением изображения из мозаики. Изображение разбивается на отдельные мелкие фрагменты (точки), каждому из которых присваивается код цвета.

В результате пространственной дискретизации графическая информация представляется в виде растрового изображения. Растровое изображение состоит из определённого количества строк, каждая из которых содержит определённое количество точек (пиксел).

Качество изображения зависит от разрешающей способности.

Разрешающая способность растрового изображения определяется количеством точек по горизонтали (X) и количеством точек по вертикали (Y) на единицу длины изображения.

Чем меньше размер точки, тем больше разрешающая способность (больше строк растра и точек в строке) и, соответственно, выше качество изображения.
Величина разрешающей способности выражается в (dot per inch – точек на дюйм), т.е. в количестве точек в полоске изображения длиной в 1 дюйм (1дюйм = 2,54 см).

Оцифровка графических изображений с бумаги или плёнок производится с помощью сканера.

Сканирование производится путём перемещения светочувствительных элементов вдоль изображения. Характеристики сканера выражаются двумя числами, например **1200 x 2400** dpi.

Первое число определяет количество светочувствительных элементов на одном дюйме полоски и является оптическим разрешением.

Второе - является аппаратным разрешением и определяет количество микрошагов при перемещении на один дюйм вдоль изображения.

В процессе дискретизации могут использоваться различные палитры цветов. Каждый цвет можно рассматривать как возможное состояние точки. Количество цветов N в палитре и количество информации для кодирования цвета каждой точки связаны между собой известной формулой Хартли: $\mathbf{N=2^I}$**,** где I – глубина цвета, а N – количество цветов (палитра).

Количество информации, которое используется для кодирования цвета точки изображения, называется ***глубиной*** цвета. Пространственное разрешение экрана монитора определяется как произведение количества строк изображения на количество точек в строке. Разрешение может быть: 1024 x 768, 1920 x 1080 и выше. Количество отображаемых цветов может изменяться от 256 цветов до более чем 16 миллионов.

Цветное изображение на экране монитора формируется за счет смешивания базовых цветов: красного, зеленого и синего (палитра RGB). Для получения богатой палитры цветов базовым цветам могут быть заданы различные интенсивности. Например, при глубине цвета в 24 бита на каждый из цветов, выделяется по 8 бит, т.е. для каждого из цветов возможны $N=2^8=256$ уровней интенсивности, заданные двоичными кодами от минимального 00000000 до максимального 11111111.

Часто цвет записывается в виде - #RRGGBB, где RR – шестнадцатеричный код красной цветовой компоненты, GG - шестнадцатеричный код зеленой цветовой компоненты, BB - шестнадцатеричный код синей цветовой компоненты. Чем больше значение компоненты, тем больше интенсивность свечения соответствующего базового цвета. 00 – отсутствие свечения, FF – максимальное свечение ($FF_{16}=255_{10}$), 80_{16} – среднее значение яркости. Если компонента имеет интенсивность цвета $<80_{16}$, то это даст темный оттенок, а если $>=80_{16}$, то светлый.

Например,

#FF0000 – красный цвет (красная составляющая максимальная, а остальные равны нулю)

#000000 – черный цвет (ни одна компонента не светится)

#FFFFFF – белый цвет (все составляющие максимальны и одинаковы, наиболее яркий цвет)

#404040 – темно-серый цвет (все составляющие одинаковы и значения меньше среднего значения яркости)

#8080FF – светло-синий (максимальная яркость у синий составляющей, а яркости других компонент одинаковые и равны 80_{16}).

Звук представляет собой звуковую волну с непрерывно меняющейся амплитудой и частотой. Чем больше амплитуда сигнала, тем он громче, чем больше частота, тем выше тон. Для того, чтобы компьютер мог обрабатывать звук, непрерывный звуковой сигнал должен быть превращен в последовательность электрических импульсов (двоичных нулей и единиц).

В процессе кодирования непрерывного звукового сигнала производится его временная дискретизация. При этом звуковая волна разбивается на мелкие временные участки, для каждого из которых устанавливается значение амплитуды.

Временная дискретизация – процесс, при котором, во время кодирования непрерывного звукового сигнала, звуковая волна разбивается на отдельные маленькие временные участки, причем для каждого такого участка устанавливается определенная величина амплитуды. Чем больше амплитуда сигнала, тем громче звук.

На графике (см. рис.) это выглядит как замена гладкой кривой на последовательность ”ступенек”, каждой из которых присваивается значение уровня громкости. Чем большее количество уровней громкости будет выделено в процессе кодирования, тем более качественным будет звучание.

Глубина звука (глубина кодирования) - количество бит на кодировку звука.

Уровни громкости (уровни сигнала) - звук может иметь различные уровни громкости. Количество различных уровней громкости рассчитываем по формуле Хартли: $\mathbf{N= 2^I}$ где I – глубина звука, а N – уровни громкости.

Современные звуковые карты обеспечивают 16-битную глубину кодировки звука. Количество различных уровней сигнала можно рассчитать по формуле: $N=2^{16}=65536$. Т.о., современные звуковые карты обеспечивают кодирование 65536 уровней сигнала. Каждому значению амплитуды присваивается 16-ти битный код.

При двоичном кодировании непрерывного звукового сигнала он заменяется последовательностью дискретных уровней сигнала. Качество кодирования зависит от количества измерений уровня сигнала в единицу времени, т.е. частотой дискретизации. Чем большее количество измерений проводится в 1 секунду (чем больше частота дискретизации), тем точнее процедура двоичного кодирования.

Частота дискретизации – количество измерений уровня входного сигнала в единицу времени (за 1 сек). Чем больше частота дискретизации, тем точнее процедура двоичного кодирования. Частота измеряется в герцах (Гц).

одно измерение за 1 секунду -**1** ГЦ, 1000 измерений за 1 секунду **1** кГц.

Обозначим частоту дискретизации буквой **F**. Для кодировки выбирают одну из трех частот: **44,1 КГц, 22,05КГц, 11,025 КГц.**

Считается, что диапазон частот, которые слышит человек, составляет от **20 Гц до 20 кГц**.

Качество двоичного кодирования звука определяется глубиной кодирования и частотой дискретизации.

Частота дискретизации аналогового звукового сигнала может принимать значения от 8 кГц до 48 кГц. При частоте 8 кГц качество дискретизованного звукового сигнала соответствует качеству радиотрансляции, а при частоте 48кГц – качеству звучания аудио-CD. Следует также учитывать, что возможны как моно-, так и стереорежимы.

Физическое кодирование данных

Основная функция физического уровня – организация физической передачи сигналов. На этом уровне осуществляется физическое кодирование передаваемой информации.

Физическое кодирование – способы представления данных в виде электрических, оптических или радиоимпульсов.

При передаче данных на физическом уровне используют два типа кодирования дискретных данных:

- аналоговое;
- цифровое.

Линии связи, по которым передаются сигналы, характеризуются *полосой пропускания*.

Полоса пропускания - это диапазон частот передаваемого по линии связи сигнала, при котором сигнал передается без значительных искажений.

Пропускная способность линии связи - это максимально возможная скорость передачи данных, которая может быть достигнута на этой линии.

Амплитудная модуляция (AM – Amplitude Modulation), при которой логической **единице** соответствует наличие сигнала (или сигнал большей амплитуды), а логическому **нулю** – отсутствие сигнала (или сигнал меньшей амплитуды).

Частота сигнала при этом остается постоянной. *Недостаток* амплитудной модуляции состоит в том, что AM-сигнал сильно подвержен действию помех и шумов, а также предъявляет повышенные требования к затуханию сигнала в канале связи.

Достоинства – простота аппаратурной реализации и узкий частотный спектр.

Частотная модуляция (ЧМ, FM – Frequency Modulation), при которой логической **единице** соответствует сигнал более высокой частоты, а логическому **нулю** – сигнал более низкой частоты (или наоборот). Амплитуда сигнала при частотной модуляции остается постоянной, что является большим преимуществом по сравнению с амплитудной модуляцией.

Фазовая модуляция (ФМ, PM – Phase Modulation), при которой смене логического **нуля** на логическую **единицу** и наоборот соответствует резкое изменение фазы синусоидального сигнала одной частоты и амплитуды.

Важно, что амплитуда модулированного сигнала остается постоянной, как и в случае частотной модуляции. Применяются и значительно более сложные методы модуляции, являющиеся комбинацией перечисленных простейших методов. Чаще всего аналоговое

кодирование используется при передаче информации по каналу с узкой полосой пропускания, например, по телефонным линиям в глобальных сетях. Кроме того, аналоговое кодирование применяется в радиоканалах, что позволяет обеспечивать связь между многими пользователями одновременно. В локальных кабельных сетях аналоговое кодирование практически не используется из-за высокой сложности и стоимости как кодирующего, так и декодирующего оборудования. (МОДЕМЫ)

При передаче информации в сетях никогда не применяется прямое двоичное кодирование бита 0 напряжением 0 вольт и бита 1 - напряжением +5 вольт, так как такой способ приводит к неоднозначности. Если одна станция посылает битовую строку 00010000, то другая станция может интерпретировать её либо как 10000, либо как 01000, так как она не может отличить «отсутствие сигнала» от бита 0. Поэтому приемнику необходим способ однозначного определения начала, конца и середины каждого бита(синхронизация), передаваемого передатчиком, без помощи внешних дополнительных сигналов. Для этих целей применяются различные методы кодирования, позволяющие приемнику синхронизироваться от принимаемого сигнала (самосинхронизация).

Цифровое кодирование делится на:

- *потенциальное;*
- *импульсное.*

Системы счисления

Представление чисел в компьютере имеет две особенности: числа записываются в двоичной системе счисления; для записи и обработки чисел отводится конечное количество разрядов (в «некомпьютерной» арифметике такое ограничение отсутствует).

Способ представления числа определяется *системой счисления*.

Система счисления – это правило записи чисел с помощью заданного набора специальных знаков – цифр.

Людьми использовались различные способы записи чисел, которые можно объединить в несколько групп:

Унарная – это система счисления, в которой для записи чисел используется только один знак – | (вертикальная черта, палочка).

Непозиционные. Наиболее распространенной можно считать римскую систему счисления. В ней некоторые базовые числа обозначены заглавными латинскими буквами: 1 – I, 5 – V, 10 – X, 50 – L, 100 – C, 500 – D, 1000 – M.

Все другие числа строятся комбинацией базовых в соответствии со следующими правилами:

- если цифра меньшего значения стоит справа от большей цифры, то их значения суммируются; если слева – то меньшее значение вычитается из большего;
- цифры I,X,C и M могут следовать подряд не более трех раз каждая;
- цифры V,L, и D могут использоваться в записи числа не более одного раза.

В числе 555 цифра 5 встречается трижды, однако значение этих цифр различно и определяется их положением (позицией) в числе.

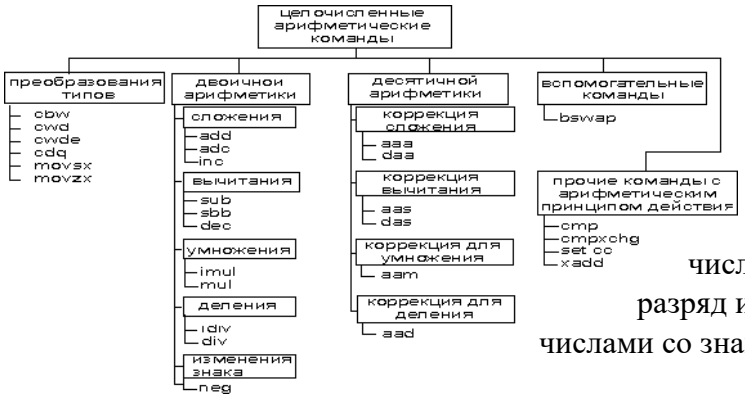
Очевидно, что значение целого числа, т.е. общее количество входящих в него единиц, не зависит от способа его представления и остается одинаковым во всех системах счисления; различают только *формы представления* одного и того же количественного содержания числа.

Например: $|||||_1 = 5_{10} = 101_2 = 5_{16}$. В ЭВМ используются, в основном, два способа представления двоичных чисел – с фиксированной и плавающей запятой, причем в формате с фиксированной запятой используются как *беззнаковое* представление чисел, так и представление чисел со знаком. В последнем случае знак также кодируется двоичной цифрой – обычно плюсу соответствует 0, а минусу – 1. Под код знака обычно отводится старший разряд a_0 двоичного вектора $a_0\, a_1\, a_2\, \ldots a_n$, называемый *знаковым*. Микропроцессор может выполнять целочисленные операции и операции с плавающей запятой. Для этого в его архитектуре есть два отдельных блока:

- устройство для выполнения целочисленных операций;
- устройство с плавающей запятой.

Каждое из этих устройств имеет свою систему команд. Целочисленное устройство может взять на себя многие функции устройства с плавающей запятой. Для большинства задач, использующих язык ассемблера, достаточно целочисленной арифметики.

Целочисленное вычислительное устройство поддерживает чуть больше десятка арифметических команд. На рисунке 5.1 приведена классификация команд этой группы.



Группа арифметических целочисленных команд работает с двумя типами чисел:

- целыми двоичными числами. Числа могут, иметь знаковый разряд или не иметь такового, то есть быть числами со знаком или без знака;

целыми десятичными числами. ***Целые двоичные числа***

Целое двоичное число с фиксированной запятой – это число, закодированное в двоичной системе счисления.

Размерность целого двоичного числа может составлять 8, 16 или 32 бит. Знак двоичного числа определяется тем, как интерпретируется старший бит в представлении числа. Это 7-й, 15-й или 31-й биты для чисел соответствующей размерности. Среди арифметических команд есть всего две команды, которые учитывают этот старший разряд как знаковый, – это команды целочисленного умножения и деления imul и idiv. *В остальных случаях*

Размерность поля	Целое без знака	Целое со знаком
байт	0...255	–128...+127
слово	0...65 535	–32 768...+32 767
двойное слово	0...4 294 967 295	–2 147 483 648...+2 147 483 647

ответственность за действия со знаковыми числами и, соответственно, со знаковым разрядом ложится на программиста. Диапазон значений двоичного числа зависит от его размера и трактовки старшего бита либо как старшего значащего бита числа, либо как бита знака числа (таблица 5.1).

Десятичные числа – специальный вид представления числовой информации, в основу которого положен принцип кодирования каждой десятичной цифры числа группой из четырех бит. При этом каждый байт числа содержит одну или две десятичные цифры в так называемом двоично-десятичном коде (BCD – Binary-Coded Decimal). Микропроцессор хранит BCD-числа в двух форматах :

упакованном формате – в этом формате каждый байт содержит две десятичные цифры. Десятичная цифра представляет собой двоичное значение в диапазоне от 0 до 9 размером 4 бита. При этом код старшей цифры числа занимает старшие 4 бита. Следовательно, диапазон представления десятичного упакованного числа в одном байте составляет от 00 до 99;

неупакованном формате – в этом формате каждый байт содержит одну десятичную цифру в четырех младших битах. Старшие четыре бита имеют нулевое значение. Это так называемая зона. Следовательно, диапазон представления десятичного неупакованного числа в одном байте составляет от 0 до 9.

СТРУКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПК

Программное обеспечение - совокупность программ, предназначенная для решения задач на ПК, называется программным обеспечением. Состав программного обеспечения ПК называют программной конфигурацией.

Программное обеспечение, можно условно разделить на три категории:

1. системное ПО (программы общего пользования), выполняющие различные вспомогательные функции, например создание копий используемой информации, выдачу справочной информации о компьютере, проверку работоспособности устройств компьютера и т.д.
2. прикладное ПО, обеспечивающее выполнение необходимых работ на ПК: редактирование текстовых документов, создание рисунков или картинок, обработка информационных массивов и т.д.
3. инструментальное ПО (системы программирования), обеспечивающее разработку новых программ для компьютера на языке программирования.

Системное ПО - это программы общего пользования не связаны с конкретным применением ПК и выполняют традиционные функции:

- планирование
- управление задачами
- управления вводом-выводом

и т.д.

Другими словами, системные программы выполняют различные вспомогательные функции, например, создание копий используемой информации, выдачу справочной информации о компьютере, проверку работоспособности устройств компьютера и т.п.

К системному ПО относятся:

- операционные системы (эта программа загружается в ОЗУ при включении компьютера)
- программы – оболочки (обеспечивают более удобный и наглядный способ общения с компьютером, чем с помощью командной строки DOS, например, Norton Commander, Total Commander)
- операционные оболочки – интерфейсные системы, которые используются для создания графических интерфейсов, мультипрограммирования и.т.
- драйверы (программы, предназначенные для управления портами периферийных устройств, обычно загружаются в оперативную память при запуске компьютера)
- утилиты (вспомогательные или служебные программы, которые представляют пользователю ряд дополнительных услуг).

К утилитам относятся:

- ✓ диспетчеры файлов или файловые менеджеры;
- ✓ средства динамического сжатия данных (позволяют увеличить количество информации на диске за счет ее динамического сжатия);
- ✓ средства просмотра и воспроизведения;
- ✓ средства диагностики; средства контроля позволяют проверить конфигурацию компьютера и проверить работоспособность устройств компьютера, прежде всего жестких дисков
- ✓ средства коммуникаций (коммуникационные программы) предназначены для организации обмена информацией между компьютерами
- ✓ средства обеспечения компьютерной безопасности (резервное копирование, антивирусное ПО).

Примечание: часть **утилит** входит в состав операционной системы, а другая часть функционирует автономно.

Большая часть общего (системного) ПО входит в состав ОС. Часть общего ПО входит в состав самого компьютера (часть программ ОС и контролирующих тестов записана в ПЗУ или ППЗУ, установленных на системной плате). Часть общего ПО относится к автономными программам и поставляется отдельно.

Операционная система – **это комплекс взаимосвязанных системных программ, назначение которого — организовать взаимодействие пользователя с компьютером и выполнение всех других программ.**

Операционная система выполняет роль связующего звена между аппаратурой компьютера, с одной стороны, и выполняемыми программами, а также пользователем, с другой стороны.

Операционная система обычно хранится во внешней памяти компьютера — *на диске*. При включении компьютера она считывается с дисковой памяти и размещается в *ОЗУ*.

Этот процесс называется ***загрузкой операционной системы.***

В функции операционной системы входит:

- осуществление диалога с пользователем;
- ввод-вывод и управление данными;
- планирование и организация процесса обработки программ;
- распределение ресурсов (оперативной памяти и кэша, процессора, внешних устройств);
- запуск программ на выполнение;
- всевозможные вспомогательные операции обслуживания;
- передача информации между различными внутренними устройствами;
- программная поддержка работы периферийных устройств (дисплея, клавиатуры, дисковых накопителей, принтера и др.).

В зависимости от количества одновременно обрабатываемых задач и числа пользователей, которых могут обслуживать ОС, различают четыре основных класса операционных систем:

1. **однопользовательские однозадачные**, которые поддерживают одну клавиатуру и могут работать только с одной (в данный момент) задачей;
2. **однопользовательские однозадачные с фоновой печатью**, которые позволяют помимо основной задачи запускать одну дополнительную задачу, ориентированную, как правило, на вывод информации на печать. Это ускоряет работу при выдаче больших объёмов информации на печать;
3. **однопользовательские многозадачные**, которые обеспечивают одному пользователю параллельную обработку нескольких задач. Например, к одному компьютеру можно подключить несколько принтеров, каждый из которых будет работать на "свою" задачу;
4. **многопользовательские многозадачные**, позволяющие на одном компьютере запускать несколько задач нескольким пользователям. Эти ОС очень сложны и требуют значительных машинных ресурсов.

В различных моделях компьютеров используют операционные системы с разной архитектурой и возможностями. Для их работы требуются разные ресурсы. Они предоставляют разную степень сервиса для программирования и работы с готовыми программами.

Операционная система для персонального компьютера, ориентированного на профессиональное применение, должна содержать следующие основные компоненты:

- программы управления вводом/выводом;

- программы, управляющие файловой системой и планирующие задания для компьютера;
- процессор командного языка, который принимает, анализирует и выполняет команды, адресованные операционной системе.

Каждая операционная система имеет свой **командный язык**, который позволяет пользователю выполнять те или иные действия:

- обращаться к каталогу;
- выполнять разметку внешних носителей;
- запускать программы;
- другие действия.

Анализ и исполнение команд пользователя, включая загрузку готовых программ из файлов в оперативную память и их запуск, осуществляет **командный процессор** операционной системы.

Для управления внешними устройствами компьютера используются специальные системные программы — **драйверы**. Драйверы стандартных устройств образуют в совокупности **базовую систему ввода-вывода** (BIOS), которая обычно заносится в постоянное ЗУ компьютера.

Прикладные программы могут использоваться автономно или в составе программных комплексов или пакетов. **Прикладное ПО** – программы, непосредственно обеспечивающие выполнение необходимых работ на ПК: редактирование текстовых документов, создание рисунков или картинок, создание электронных таблиц и т.д.

Пакеты прикладных программ – это система программ, которые по сфере применения делятся на *проблемно – ориентированные, пакеты общего назначения и интегрированные пакеты*.

Современные интегрированные пакеты содержат до пяти функциональных компонентов: тестовый и табличный процессор, СУБД, графический редактор, телекоммуникационные средства.

К прикладному ПО, например, относятся:

- Комплект офисных приложений MS OFFICE
- Бухгалтерские системы
- Финансовые аналитические системы
- Интегрированные пакеты делопроизводства
- CAD – системы (системы автоматизированного проектирования)
- Редакторы HTML или Web – редакторы
- Браузеры – средства просмотра Web - страниц
- Графические редакторы
- Экспертные системы

- и так далее.

Классификация **прикладного ПО**
По типу

- программные средства общего назначения
  - Текстовые редакторы
  - Текстовые процессоры
  - Системы компьютерной вёрстки
  - Графические редакторы
  - СУБД
  - Электронные таблицы
  - Веб-браузеры
- программные средства специального назначения
  - Экспертные системы
  - Мультимедиа приложения (медиаплееры, программы для создания и редактирования видео, звука пр.)
  - Гипертекстовые системы (электронные словари, энциклопедии, справочные системы)
  - и др.
- профессиональные программные средства
  - САПР
  - АРМ
  - АСУ
  - АСУ ТП – автоматизированная система управления технологическим процессом
  - АСНИ – автоматизированная система научных исследований (реже используются термины САНИ – система автоматизации научных исследований и САЭ – система автоматизации эксперимента)
  - Геоинформационные системы
  - Биллинговые системы
  - CRM – (Customer Relationship Management) – управление взаимоотношениями с клиентами
  - BI (Business Intelligence) — Аналитические Системы
  - DMS (Document Management System) – СЭД (Системы Электронного Документооборота)

- CMS (Content Management System) – Системы Управления Содержанием (контентом)
- ERP-системы – системы планирования ресурсов предприятия
- и др.

По сфере применения

- Прикладное программное обеспечение предприятий и организаций. Например, финансовое управление, система отношений с потребителями, сеть поставок. К этому типу относится также ведомственное ПО предприятий малого бизнеса, а также ПО отдельных подразделений внутри большого предприятия. (Примеры: Управление транспортными расходами, Служба IT поддержки)
- Программное обеспечение обеспечивает доступ пользователя к устройствам компьютера.
- Программное обеспечение инфраструктуры предприятия. Обеспечивает общие возможности для поддержки ПО предприятий. Это системы управления базами данных, серверы электронной почты, управление сетью и безопасностью.
- Программное обеспечение информационного работника. Обслуживает потребности индивидуальных пользователей в создании и управлении информацией. Это, как правило, управление временем, ресурсами, документацией, например, текстовые редакторы, электронные таблицы, программы-клиенты для электронной почты и блогов, персональные информационные системы и медиа редакторы.
- Программное обеспечение для доступа к контенту. Используется для доступа к тем или иным программам или ресурсам без их редактирования (однако может и включать функцию редактирования). Предназначено для групп или индивидуальных пользователей цифрового контента. Это, например, медиа-плееры, веб-браузеры, вспомогательные браузеры и др.
- Образовательное программное обеспечение по содержанию близко к ПО для медиа и развлечений, однако в отличие от него имеет четкие требования по тестированию знаний пользователя и отслеживанию прогресса в изучении того или иного материала. Многие образовательные программы включают функции совместного пользования и многостороннего сотрудничества.
- Имитационное программное обеспечение. Используется для симуляции физических или абстрактных систем в целях научных исследований, обучения или развлечения.
- Инструментальные программные средства в области медиа. Обеспечивают потребности пользователей, которые производят печатные или электронные медиа ресурсы для других потребителей, на коммерческой или образовательной основе. Это программы полиграфической обработки, верстки, обработки мультимедиа, редакторы HTML, редакторы цифровой анимации, цифрового звука и т. п.
- Прикладные программы для проектирования и конструирования. Используются при разработке аппаратного и программного обеспечения. Охватывают автоматизированное проектирование (computer aided design – CAD), автоматизированный инжиниринг (computer aided engineering – CAE),

редактирование и компилирование языков программирования, программы интегрированной среды разработки (Integrated Development Environments).

- Инструментальное ПО или системы программирования - это системы для автоматизации разработки новых программ на языке программирования.

В самом общем случае для создания программы на выбранном языке программирования (языке системного программирования) нужно иметь следующие компоненты:

- 1. Текстовый редактор для создания файла с исходным текстом программы.
- 2. Компилятор или интерпретатор. Исходный текст с помощью программы-компилятора переводится в промежуточный объектный код. Исходный текст большой программы состоит из нескольких *модулей* (файлов с исходными текстами). Каждый модуль компилируется в отдельный файл с объектным кодом, которые затем надо объединить в одно целое.
- 3. Редактор связей или сборщик, который выполняет связывание объектных модулей и формирует на выходе работоспособное приложение – исполнимый код.

Исполнимый код – это законченная программа, которую можно запустить на любом компьютере, где установлена операционная система, для которой эта программа создавалась. Как правило, итоговый файл имеет расширение .EXE или .COM.

В последнее время получили распространение визуальный методы программирования (с помощью языков описания сценариев), ориентированные на создание Windows-приложений. Этот процесс автоматизирован в средах быстрого проектирования.

При этом используются готовые визуальные компоненты, которые настраиваются с помощью специальных редакторов.

Наиболее популярные редакторы (системы программирования программ с использованием визуальных средств) визуального проектирования:

- Borland Delphi – предназначен для решения практически любых задачи прикладного программирования
- Borland C++ Builder – это отличное средство для разработки DOS и Windows приложений
- Microsoft Visual Basic – это популярный инструмент для создания Windows-программ
- Microsoft Visual C++ – это средство позволяет разрабатывать любые приложения, выполняющиеся в среде ОС типа Microsoft Windows
- **Для создания сайтов:**
- Платные WYSIWYG-редакторы:
- — Microsoft FrontPage
  — Adobe Dreamweaver
  — Adobe GoLive
  — IBM WebSphere Studio Homepage Builder
  — Macromedia HomeSite (до версии 5.0)

— NetObjects Fusion
— Namo Web Editor
— WYSIWYG Web Builder

- Бесплатные WYSIWYG-редакторы:
- — Microsoft FrontPage Express
— OpenOffice. org
— HTMLArea
— TinyMCE
— FCKeditor
— Quanta Plus
— Nvu

К этой категории относятся программы, предназначенные для разработки программного обеспечения на конкретном языке программирования:

- ассемблеры – компьютерные программы, осуществляющие преобразование программы в форме исходного текста на языке ассемблера в машинные команды в виде объектного кода.
- трансляторы – программы или технические средства, выполняющие трансляцию программы.
    - компиляторы – Программы, переводящие текст программы на языке высокого уровня, в эквивалентную программу на машинном языке.
- интерпретаторы – Программы (иногда аппаратные средства), анализирующие команды или операторы программы и тут же выполняющие их
- **компоновщики** (редакторы связей) – программы, которые производят компоновку — принимают на вход один или несколько объектных модулей и собирают по ним исполнимый модуль;
- **препроцессоры исходных текстов** – это компьютерные программы, принимающие данные на входе и выдающие данные, предназначенные для входа другой программы, например, такой, как компилятор;
- **отладчик (debugger)** – является модулем среды разработки или отдельным приложением, предназначенным для поиска ошибок в программе;
- **текстовые редакторы** – компьютерные программы, предназначенные для создания и изменения текстовых файлов, а также их просмотра на экране, вывода на печать, поиска фрагментов текста и т. п.
    - специализированные редакторы исходных текстов – текстовые редакторы для создания и редактирования исходного кода программ. Специализированный редактор исходных текстов может быть отдельным приложением, или быть встроен в интегрированную среду разработки (IDE);
- **библиотеки подпрограмм** – сборники подпрограмм или объектов, используемых для разработки программного обеспечения.
- редакторы графического интерфейса.

- **Целочисленный тип данных** — это тип, переменные которого могут содержать только целые числа (без дробной части, например: -2, -1, 0, 1, 2). В языке C++ есть 5 основных целочисленных типов, доступных для использования:

Категория	Тип	Минимальный размер
Символьный тип данных	char	1 байт
Целочисленный тип данных	short	2 байта
	int	2 байта (но чаще всего 4 байта)
	long	4 байта
	long long	8 байт

Основным различием между целочисленными типами, перечисленными выше, является их **размер**, чем он больше, тем больше значений сможет хранить переменная этого типа.

Диапазон - это значения от и до, которые может хранить определенный тип данных. Диапазон целочисленной переменной определяется двумя факторами: её размером (измеряется в битах) и её **знаком** (который может быть *signed* или *unsigned*).

Целочисленный тип signed (со знаком) означает, что переменная может содержать как положительные, так и отрицательные числа.

Чтобы объявить переменную как signed, используется ключевое слово signed

По умолчанию, ключевое слово signed пишется перед типом данных.

1-байтовая целочисленная переменная со знаком (signed) имеет диапазон значений от -128 до 127, т.е. любое значение от -128 до 127 (включительно) может храниться в ней безопасно. В некоторых случаях можно заранее знать, что отрицательные числа в программе использоваться не будут.

Это очень часто встречается при использовании переменных для хранения количества или размера чего-либо (например, ваш рост или вес не может быть отрицательным).

Целочисленный тип unsigned (без знака) может содержать только положительные числа. Чтобы объявить переменную как **unsigned**, используется ключевое слово unsigned. 1-байтовая целочисленная переменная без знака (unsigned) имеет диапазон значений от 0 до 255. !!!объявление переменной как unsigned означает, что она не сможет содержать отрицательные числа (только положительные).

Размер/Тип	Диапазон значений
1 байт signed	от -128 до 127
1 байт unsigned	от 0 до 255
2 байта signed	от -32 768 до 32 767
2 байта unsigned	от 0 до 65 535
4 байта signed	от -2 147 483 648 до 2 147 483 647
4 байта unsigned	от 0 до 4 294 967 295
8 байтов signed	от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
8 байтов unsigned	от 0 до 18 446 744 073 709 551 615

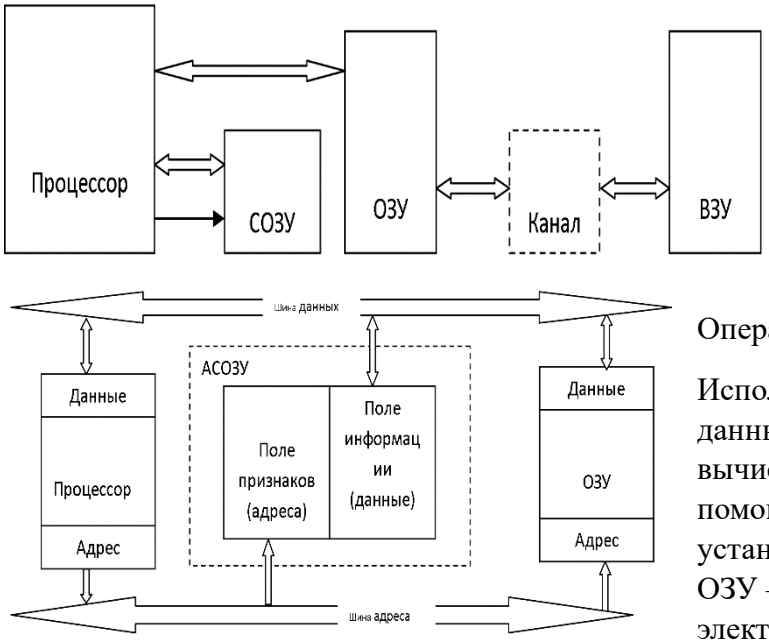
6 Организация памяти в ЭВМ

ЭВМ, реализованная по классической фон-неймановской архитектуре, включает в себя:

- процессор, содержащий арифметико-логическое устройство (АЛУ) и центральное устройство управления (ЦУУ);
- память, которая в современных ЭВМ подразделяется на оперативную (ОП или ОЗУ) и сверхоперативную (СОЗУ);
- внешние устройства, к которым относят внешнюю память (ВЗУ) и устройства ввода/вывода (УВВ)

Взаимодействие ЗУ различных уровней в составе ЭВМ

СОЗУ с ассоциативным доступом



Оперативная память (ОЗУ)

Используется для хранения текущих данных и программ во время вычислений. ОЗУ реализуется с помощью набора микросхем, установленных на материнской плате. ОЗУ – совокупность специальных электронных ячеек, каждая из которых может хранить конкретную комбинацию из 0 и 1 – один байт. Эти ячейки нумеруются порядковыми номерами: 0, 1, 2, ..., 32000, 32001,... Номер ячейки называется АДРЕСОМ того байта, который записан в ней в данный момент. Когда ПК отправляет данные на хранение в оперативную память, он запоминает адреса, в которые эти данные помещены.

Содержимое ячейки – значение байта данных.

Адресная ячейка ОЗУ хранит 1 байт, а поскольку байт состоит из 8 битов, то в ней есть 8 БИТОВЫХ ЯЧЕЕК, в каждой из которых в данный момент может находиться только одно из 2-х значений:

0 или 1 Двумя байтами (от 0 до 255) можно записать адрес для 65536 ячеек памяти (от 0 до 65535). Для большего количества ячеек адрес должен иметь больше байтов. ОЗУ обеспечивает режимы записи, считывания и хранения информации, причем в любой момент времени возможен доступ к любой произвольно выбранной ячейки памяти.

Принципиальной особенностью ОЗУ является его *способность хранить информацию только во время работы ПК*. После загрузки новой программы, прежнее содержимое ОЗУ замещается новым, после выключения ПК пропадает вовсе.

Постоянная память При включении ПК, в ОЗУ заносятся (ЗАГРУЖАЮТСЯ) цепочки байтов, в которых хранится операционная система.

Для этого предназначена специальная микросхема – **ПЗУ** (постоянное запоминающее устройство). В отличие от оперативной она не стирается при выключении. Программы микросхемы записывают на заводе. Этот комплекс программ называется **BIOS** – базовая система ввода/вывода. При включении компьютера через некоторое время раздаётся один короткий звуковой сигнал – это означает, что с компьютером все в порядке и в дальнейшем будет произведена загрузка операционной системы. Если при загрузке обнаружены какие-либо неисправности, то BIOS сигнализирует об этом, издавая соответствующие звуки, по которым можно определить характер неисправности и определиться с дальнейшими действиями.

Сигналы BIOS (сигналы спикера) различаются в зависимости от версий системы и в зависимости от характера неполадки.

Данную микросхему устанавливают так, чтобы ее память заняла нужные адреса. Поэтому, когда процессор начинает работу, он попадает в **ПОСТОЯННУЮ** память, заготовленную для него заранее. После загрузки BIOS далее, по нашим указаниям, в ОЗУ с магнитного диска помещаются прикладные программы и данные, обрабатываемые этими программами. **ПЗУ** (ROM – Read Only Memory) и **оперативная память** – не противоположные понятия. ROM представляет собой часть оперативной памяти системы, т.е часть адресного пространства ОЗУ отводиться для ROM. Это необходимо для хранения ПО, которое позволяет загрузить ОС.

Основной код BIOS содержится в микросхеме ROM на системной плате, но на платах адаптеров также имеются аналогичные микросхемы. Они содержат вспомогательные подпрограммы базовой системы ввода-вывода и драйверы, необходимые для конкретной платы, особенно для тех плат, которые должны быть активизированы на раннем этапе начальной загрузки, например видеоадаптер. Платы, не нуждающиеся в драйверах на раннем этапе начальной загрузки, обычно не имеют ROM, потому что их драйверы могут быть загружены с жесткого диска позже — в процессе начальной загрузки.

В настоящее время в большинстве систем используется одна из форм **Flash-памяти**, которая называется электронно-перепрограммируемой постоянной памятью (*Electrically Erasable Programmable Read-only Memory — EEPROM*).

Flash-память является по-настоящему энергонезависимой и перезаписываемой, она позволяет пользователям легко модифицировать ROM, программно-аппаратные средства системных плат и других компонентов (таких, как видеоадаптеры, платы SCSI-интерфейса, периферийные устройства и т. п.).

Динамическая оперативная память (Dynamic RAM — DRAM) используется в большинстве систем оперативной памяти современных персональных компьютеров. Основное преимущество памяти этого типа состоит в том, что ее ячейки упакованы очень плотно, т. е. в небольшую микросхему можно упаковать много битов, а значит, на их основе можно построить память большой емкости. Ячейки памяти в микросхеме DRAM — это крошечные конденсаторы, которые удерживают заряды. Именно так (наличием или отсутствием зарядов) и кодируются биты. Проблемы, связанные с памятью этого типа, вызваны тем, что она динамическая, т. е. должна постоянно регенерироваться, так как в противном случае электрические заряды в конденсаторах памяти будут *"стекать"* и данные будут потеряны. Регенерация происходит, когда контроллер памяти системы берет крошечный перерыв и обращается ко всем строкам данных в микросхемах памяти. Большинство систем имеют контроллер памяти (обычно встраиваемый в набор микросхем системной платы), который настроен на соответствующую промышленным стандартам

частоту регенерации, равную 15 мкс. Ко всем строкам данных обращение осуществляется по прохождении 128 специальных циклов регенерации. Это означает, что каждые 1,92мс прочитываются все строки в памяти для обеспечения регенерации данных.

Регенерация памяти отнимает время у процессора: каждый цикл регенерации по длительности занимает несколько циклов центрального процессора. В старых компьютерах циклы регенерации могли занимать до 10% (или больше) процессорного времени, но в современных системах, работающих на частотах, равных сотням мегагерц, расходы на регенерацию составляют 1% (или меньше) процессорного времени. Некоторые системы позволяют изменить параметры регенерации с помощью программы установки параметров CMOS, но увеличение времени между циклами регенерации может привести к тому, что в некоторых ячейках памяти заряд *"стечет"*, а это вызовет сбои памяти.

В большинстве случаев надежнее придерживаться рекомендуемой или заданной по умолчанию частоты регенерации. **Динамическая оперативная память** используется в персональных компьютерах; поскольку она недорогая, микросхемы могут быть плотно упакованы, а это означает, что запоминающее устройство большой емкости может занимать небольшое пространство. Память этого типа не отличается высоким быстродействием, обычно она намного *"медленнее"* процессора. Поэтому существует множество различных типов организации DRAM, позволяющих улучшить эту характеристику.

КЭШ- память (СОЗУ) Для увеличения производительности ПК, согласования устройств с различным быстродействием используют КЭШ-ПАМЯТЬ – промежуточное ЗУ или буфер. Существует 2 типа кэш-памяти: внутренняя (от 64 Кбайт) – размещается внутри процессора и внешняя (от 256 Кбайт) устанавливается на системной плате.

Это тип памяти, совершенно отличный от других, — **статическая оперативная память (Static RAM — SRAM)**. Она названа так потому, что, в отличие от динамической оперативной памяти (DRAM), для сохранения ее содержимого не требуется периодической регенерации.

Но это не единственное ее преимуществоSRAM имеет более высокое быстродействие, чем динамическая оперативная память, и может работать на той же частоте, что и современные процессоры.

Однако для хранения каждого бита в конструкции SRAM используется кластер из шести транзисторов. Использование транзисторов без каких-либо конденсаторов означает, что нет необходимости в регенерации. (Ведь если нет никаких конденсаторов, то и заряды не теряются.) Пока подается питание, SRAM будет помнить то, что сохранено.

Почему же тогда микросхемы SRAM не используются для всей системной памяти? По сравнению с динамической оперативной памятью быстродействие SRAM намного выше, но плотность ее гораздо ниже, а цена довольно высока. Более низкая плотность означает, что микросхемы SRAM имеют большие габариты, хотя их информационная емкость намного меньше. Большое число транзисторов и кластеризованное их размещение не только увеличивает габариты микросхем SRAM, но и значительно повышает стоимость технологического процесса по сравнению с аналогичными параметрами для микросхем DRAM. Таким образом, габариты SRAM в среднем в 30 раз превышают размер динамической оперативной памяти, то же самое можно сказать и о стоимости.

Все это не позволяет использовать память типа SRAM в качестве оперативной памяти в персональных компьютерах.

Несмотря на это, разработчики все-таки применяют память типа SRAM для повышения эффективности PC. Но во избежание значительного увеличения стоимости устанавливается только небольшой объем высокоскоростной памяти SRAM, которая используется в качестве кэш-памяти. Кэш-память работает на тактовых частотах, близких или даже равных тактовым частотам процессора, причем обычно именно эта память непосредственно используется процессором при чтении и записи.

- Быстродействие процессора выражается в мегагерцах (МГц), а быстродействие запоминающего устройства и его эффективность — в наносекундах (нс). Наносекунда — это одна миллиардная доля секунды, т.е. очень короткий промежуток времени.
- Быстродействие процессоров и микросхем выражается в мегагерцах (МГц), т.е. в миллионах циклов, выполняемых в течение одной секунды. Рабочая частота современных процессоров достигает 3000 и более МГц (3 ГГц, или 3 млрд циклов в секунду), а далее возрастет до 4 ГГц.
- Как правило, компьютер работает гораздо быстрее, если пропускная способность шины памяти соответствует пропускной способности шины процессора.
- Сравнивая скорость шины памяти с быстродействием шины процессора, можно заметить, что между этими параметрами существует определенное соответствие. Тип памяти, пропускная способность которой соответствует скорости передачи данных процессора, является наиболее приемлемым вариантом для систем, использующих соответствующий процессор.
- Если скорость шины памяти равняется частоте шины процессора, быстродействие памяти в такой системе будет оптимальным.

Внешняя память Основной функцией внешней памяти ПК является способность долговременно хранить большой объем информации. Устройство, которое обеспечивает запись/считывание информации, называется **НАКОПИТЕЛЕМ** или **ДИСКОВОДОМ**, а хранится информация на **НОСИТЕЛЯХ** (например, дисках). В накопителях на жестких магнитных дисках **(НЖМД)** или "винчестерах" в основу записи, хранения и считывания информации положен магнитный принцип, а в лазерных дисководах **CD-ROM** (компакт-диск) и **DVD-ROM** (цифровой видео-диск) – оптический принцип. **ЛАЗЕРНЫЕ ДИСКОВОДЫ CD-ROM** и **DVD-ROM** используют оптический принцип чтения информации. Информация на лазерном диске записана на одну спиралевидную дорожку, содержащую чередующиеся участки с различной отражающей способностью. Лазерный луч падает на поверхность вращающегося диска, а интенсивность отраженного луча зависит от отражающей способности участка дорожки и приобретает значения 0 или 1. Производятся **CD-ROM** и **DVD-ROM** либо путем штамповки (диски белого цвета), либо записываются (диски золотистого цвета) на специальных устройствах **CD-recorder**. Существуют также **CD-RW** и **DVD-RW** – перезаписываемые, платинового оттенка. Перед перезаписью записанную информацию "стирают" путем нагревания участков поверхности диска с помощью лазера. Запись на диски такого типа отличается от «*-**R**» тем, что при записи информации на данный диск, дорожки «прогибаются», а не прожигаются как на дисках **R-**типа.

Виртуальная память В современных ЭВМ реализовано *динамическое распределение памяти* между несколькими задачами, существующими в ЭВМ в процессе решения. Данное распределение памяти, называется *страничной организацией виртуальной памяти*. Суть концепции виртуальной памяти заключается в том, что виртуальные адреса к которым обращается выполняющийся процесс отделяются от адресов реально существующих в

первичной памяти (реальные, физические или абсолютные адреса). Диапазон виртуальных адресов к которым может обращаться выполняющийся процесс называется *виртуальным адресным пространством* этого процесса.

Диапазон реальных адресов, существующих в конкретной ЭВМ (2^n, где n– разрядность регистра физического адреса), называется *пространством реальных адресов R* этой машины. Предполагают, что объем *V* значительно больше *R*. Перевод виртуальных адресов в реальные во время выполнения процесса, называется *динамическим преобразованием адресов* (DAT). Это преобразование адресов выполняет сама система, причем это делается прозрачно (невидимо) для пользователя.

Существует два наиболее распространенных способа реализации виртуальной памяти – это страничная и сегментная организация.

Блоки фиксированного размера, называются *страницами*, блоки переменного размера называются *сегментами*. В современной ВС оба вида блоков комбинируются, причем как правило применяются сегменты длина которых выражается целым числом страниц (*странично-сегментная* организация памяти).