Тема 1.1. Знакомство с архитектурой компьютера

Архитектура компьютера, системы счисления, биты и байты, Фон-неймановская архитектура

Знакомство с архитектурой компьютера

программы на ассемблере напрямую манипулируют устройствами компьютера, в первую очередь процессором и памятью. Отсюда неслучайно ассемблер называют языком низкого уровня. Языки высокого уровня скрывают от программиста все манипуляции с компьютерным "железом".

Таким образом, чтобы научиться программировать на ассемблере, необходимо знать архитектуру компьютера.

1.1. Что такое архитектура компьютера

Четкого определения нет, однако для определенности приведу следующее:
Архитектура компьютера – это логическая организация, структура и ресурсы компьютера, которые может использовать программист.

Архитектура компьютера включает в себя архитектуры отдельных устройств, входящих в компьютер.

реально программисту на ассемблере приходится работать только с тремя устройствами компьютерной системы:

- **процессором**;
- **памятью**;
- **портами ввода-вывода**.

В сущности, эти три устройства определяют работу всего компьютера и работу всех внешних устройств подключенных к нему.

Все эти три устройства соединены между собой при помощи трех основных шин: шиной данных (ШД), шиной адреса (ША) и шиной управления (ШУ)

1.1. Что такое архитектура компьютера

Процессор, память, порты ввода-вывода соединены между собой при помощи трех основных шин: шиной данных (ШД), шиной адреса (ША) и шиной управления (ШУ)

Шины это просто набор проводников по которым передаются цифровые сигналы (машинные коды) от процессора к памяти, от процессора к портам ввода-вывода, и обратно от этих устройств к процессору.

Все три шины вместе образуют системную шину или ее еще называют магистраль.

1.1. Что такое архитектура компьютера

Существует еще такое понятие как **микроархитектура**. Если архитектура это программно-видимые свойства устройства, то микроархитектура – внутренняя реализация архитектуры. Для одной и той же архитектуры разными производителями могут применяться существенно различные микроархитектурные реализации с различной производительностью и стоимостью.

Например, к **архитектуре процессора** относятся регистры и набор инструкций (команд), которые может использовать программист в своей программе. А к **микроархитектуре процессора** – способы конвейеризации, распараллеливания, буферизации вычислений и другие технологии, предназначенные для увеличения скорости работы процессора. В микроархитектуре процессора тоже может использоваться свой набор команд, но он обычно недоступен программисту. Микроархитектура отличается у различных производителей и часто существенно меняется при переходе от одного поколения процессоров к другому (поэтому ее рассматривать не будем).

необходимо остановиться на системах счисления, а также битах и байтах, потому что они тесно связаны с архитектурой компьютера.

1.2. Системы счисления

Слово "**компьютер**" (computer) с английского языка переводится как "вычислитель", т. е. машина для проведения вычислений. И это полностью соответствует действительности, т. к. на уровне "железа" компьютер выполняет только простейшие арифметические операции с числами, такие как сложение и умножение. В СССР, да и в современной России часто компьютер прямо так и называют: **ЭВМ** ("электронная вычислительная машина").

Сердцем компьютера является **процессор**, называемый часто центральным процессором (ЦП) или микропроцессором. Именно центральный процессор выполняет все вычисления.

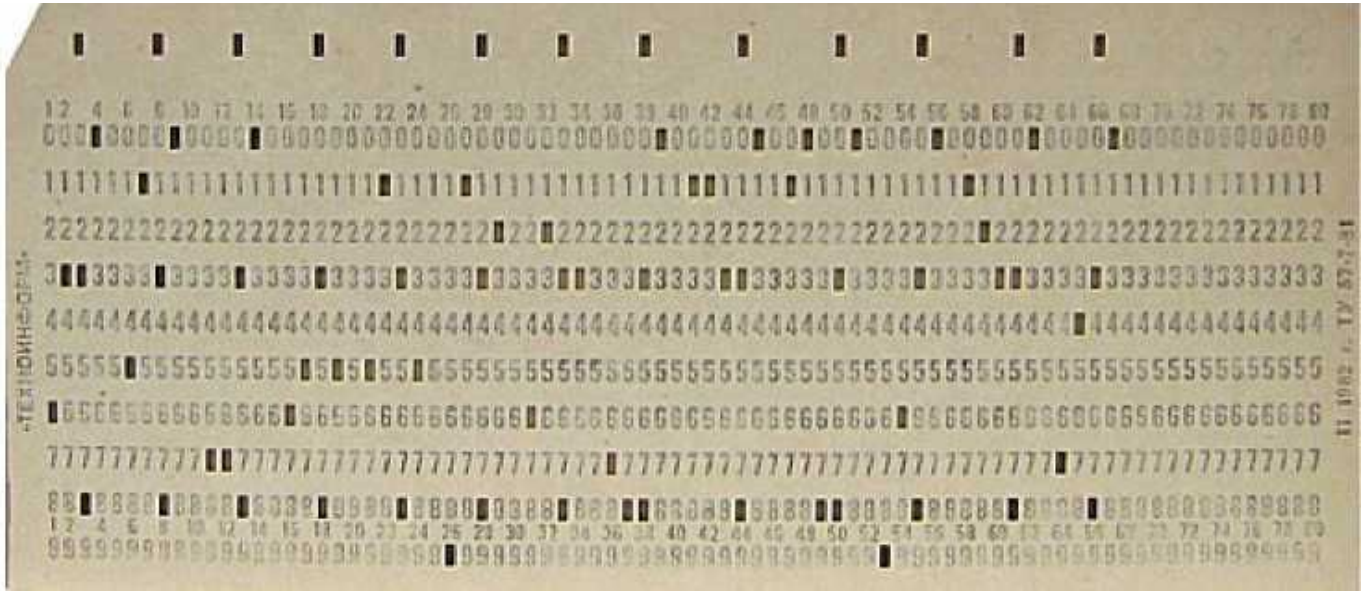
Так исторически сложилось, что практически все цифровые микросхемы, в том числе компьютерные процессоры, **работают только с двумя разрешенными уровнями напряжения**. Один из этих уровней называется уровнем логической единицы (или единичным уровнем), а другой — уровнем логического нуля (или нулевым уровнем). Чаще всего логическому нулю соответствует низкий уровень напряжения (от 0 до 0,4 В), а логической единице — высокий уровень (от 2,4 до 5 В). Здесь следует заметить, что диапазон приведен для процессоров с напряжением питания 5 В (в современных процессорах оно как правило ниже). Два уровня напряжения было выбрано исключительно из-за простоты реализации.

Таким образом, можно образно представлять, что в электронной цепи компьютера "бегают" только цепочки ноликов и единичек. За этими цепочками нулей и единичек закрепилось название **машинные коды**. Точно также можно представлять, что в память компьютера, а также на магнитные, оптические и прочие носители записываются нолики и единички, которые в совокупности составляют хранимую информацию.

1.2. Системы счисления

Следует отметить, что были попытки сделать ЭВМ на основе **троичной логики** (например, отечественная ЭВМ "Сетунь", 1959 г.) и даже на основе **десятичной логики** (американская ЭВМ "Марк-1", 1943 г.). Но распространения они не получили, т. к. их устройство сложнее машин на основе двоичной логики, а потому они дороже и менее надежны. К тому же все, что осуществимо на машинах с троичной и десятичной логикой (и вообще любой недвоичной логикой), то осуществимо и на машинах с двоичной логикой

Так как компьютер способен воспринимать только два управляющих сигнала: 0 и 1, то и **любая программа должна быть ему представлена только в двоичных кодах**, т. е. в машинных кодах. В старые добрые времена операторы первых ЭВМ программировали напрямую в машинных кодах, переключая специально предусмотренные для этого тумблеры, или пробивали двоичные коды на перфолентах и перфокартах, которые затем считывала ЭВМ и выполняла операции согласно этим кодам.



Перфокарта представляла собой картонный прямоугольный лист, на который была нанесена цифровая сетка. Информация на перфокарты наносилась оператором путем пробивки отверстий в нужных местах цифровой сетки на перфокарте с помощью специального электромеханического устройства – перфоратора. Наличие отверстия означало код 1, а его отсутствие – код 0. Информация считывалась с перфокарты в процессе перемещения ее в специальном устройстве считывателе. В считыватель обычно подавалась сразу стопка перфокарт с нанесенной информацией.

1.2. Системы счисления

недвоичные системы счисления первые программисты стали использовать исключительно для личного удобства. Компьютер не способен воспринимать десятичные, шестнадцатеричные или восьмеричные числа, а только и только двоичные коды!

Сравните: двоичное число 11001000 будет представлено
в десятичном виде как **200**,
в восьмеричной **310**
в шестнадцатеричной как и **C8**

Как в литературе, так и в программах на ассемблере, для обозначения системы счисления, в которой записано число, принято ставить в конце числа букву
b (Bin) – для двоичного,
o (Oct) – для восьмеричного,
h (Hex) – для шестнадцатеричного числа.
Для обозначения десятичной системы счисления обычно не используется никакой буквенной приставки или в редких случаях ставится **d** (Dec).

Одно и то же число, записанное в различных системах счисления, будет выглядеть так:

200d = 11001000b = 310o = C8h

Таким образом, операторы первых ЭВМ стали составлять свои программы в более удобной системе счисления (восьмеричной, шестнадцатеричной или другой), а потом переводить их в двоичный машинный код.

1.2. Системы счисления

Наибольшее распространение у первых программистов из всех систем счисления получила шестнадцатеричная система счисления, которая до сих пор является основной в компьютерном мире.

В отличие от других систем счисления перевод из шестнадцатеричной системы счисления в двоичную систему и обратно осуществляется очень легко — вместо каждой шестнадцатеричной цифры, подставляется соответствующее четырехзначное двоичное число, например:

486h =10010000110b, где
4h = 0100b,
8h = 1000b,
6h = 0110b

1.2. Системы счисления

Для сравнения посмотрим, как перевести десятичное число в двоичное (возьмем для примера число 200). Для этого надо его делить на 2 и записывать остаток справа налево:

200/2 = 100 (остаток 0)
100/2 = 50 (остаток 0)
50/2 = 25 (остаток 0)
25/2 = 12 (остаток 1)
12/2 = 6 (остаток 0)
6/2 = 3 (остаток 0)
3/2 = 1 (остаток 1)
1/2 =0 (остаток 1)

Результат: 11001000b

А для обратного перевода двоичного числа в десятичное, необходимо сложить двойки в степенях, соответствующих позициям, где в двоичном стоят единицы. Пример:

$$11001000b=1x2^7+1x2^6+0x2^5+0x2^4+1x2^3+0x2^2+0x2^1+0x2^\circ=12\ 8+64+8=200$$

Во времена древних ЭВМ еще не существовало калькуляторов и программистам приходилось все численные преобразования делать в уме, либо с помощью карандаша и листка бумаги, поэтому использование шестнадцатеричной системы значительно облегчило этот труд.

1.2. Системы счисления

хотя шестнадцатеричная система облегчила работу с машинными кодами, но создавать программу в шестнадцатеричном виде все равно очень не просто. В итоге **родился язык ассемблера**, который давал возможность писать программы на более понятном человеку языке и в то же время позволял легко переводить их в машинный код.

Язык ассемблера прозвали **низкоуровневым языком**, потому что он максимально приближен к машинному языку, а значит к "железу" компьютера.

После языка ассемблера стали появляться высокоуровневые языки, такие как Бейсик, Паскаль, Фортран, Си, C++ и пр. Они еще более понятны человеку, но преобразование в машинный код высокоуровневых программ значительно сложнее, из-за чего размер кода, как правило, получается большим и менее быстрым по сравнению с ассемблерными программами.

команда на Си	на языке ассемблера	в шестнадцатеричном виде	в машинном (двоичном) коде
x = x + 33; (прибавить к переменной x число 33 и результат сохранить в x)	add ax,33	83h C0h 21h	1000001111000000010 0001

1.2. Системы счисления

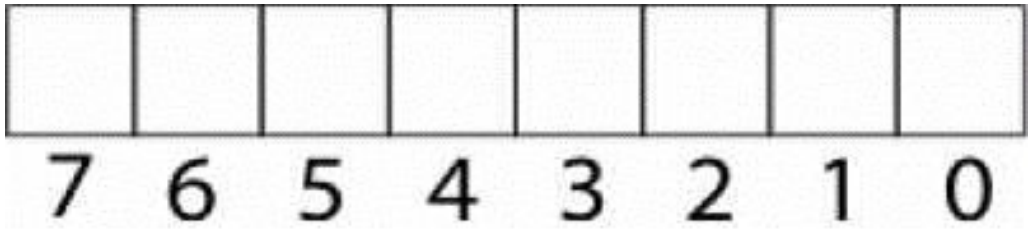
Если операторы первых ЭВМ переводили свои программы в машинный код вручную, то сейчас эту работу выполняют специальные программы — **трансляторы** (англ. translator — переводчик). Для языков высокого уровня транслятор принято называть **компилятором** (англ. compiler — составитель, собиратель). Для языка ассемблера обычно тоже не используется слово транслятор, а говорят просто: "ассемблер". Таким образом, ассемблером называют, как язык программирования, так и транслятор этого языка.

Соответственно процесс работы ассемблера называют **ассемблированием**.
Процесс работы компилятора называют **компилированием**.
Процесс обратный ассемблированию, т. е. преобразование машинного кода в программу на языке ассемблера называют **дизассемблированием**.

1.3. Биты и байты

Цифра в двоичной арифметике называется **разрядом** (или точнее "двоичным разрядом") и может принимать значение ноль или единица. В компьютерном мире вместо разряда часто употребляют название **бит**. Таким образом, бит, либо разряд, в данном контексте – это одно и тоже.

минимальной единицей информации в компьютерной системе является **бит**, который может принимать только значение 0 или 1. Однако минимальным объемом данных, которым позволено оперировать любой компьютерной программе является не бит, а **байт**.
Байт состоит из восьми бит. Если программе нужно изменить значение только одного бита, то она все равно должна считать целый байт, содержащий этот бит.



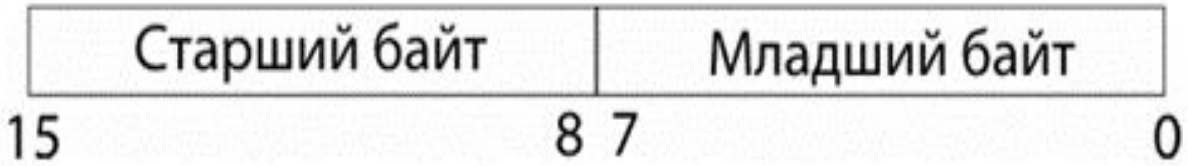
Биты в байте нумеруются справа налево от 0 до 7, при этом нулевой бит принято называть младшим, а седьмой — старшим

Так как в байте всего восемь бит, а бит может принимать только два значения, то простой арифметический подсчет показывает, что байт может принимать до $2^8=256$ различных значений.

Поэтому в байте могут быть представлены целые числа в диапазоне от 0 до 255, или числа со знаком от –128 до +127.

1.3. Биты и байты

Однако не только байтами может оперировать компьютерная программа, но и более крупными единицами данных — **словами**, **двойными словами** и **учетверенными словами**.



Слово состоит из двух байт, при этом биты с 0 по 7 составляют младший байт в слове, а биты с 8 по 15 — старший. Слово может принимать до 2^{16}=65536 различных значений.

Двойное слово, как следует из самого названия, состоит из двух слов или четырех байт, а значит из 32-х бит,
а два двойных слова составляют **учетверенное слово** (64 бита).
Существует еще более крупная единица, которая называется **параграф** и представляет собой 16 смежных байт.

1.4. Фон-неймановская архитектура

Подавляющее большинство современных вычислительных машин, в том числе IBM PC-совместимые компьютеры, представляют собой реализацию так называемой **фоннеймановской архитектуры**. Эту архитектуру предложил американский ученый венгерского происхождения Джордж фон Нейман в 1945 году.

Существует версия, что авторство идеи принадлежит не ему, а разработчикам сверхсекретного в то время компьютера ENIAC Джону Маучли (J. Mauchly) и Джону Эккерту (J. Eckert), у которых Нейман проходил стажировку. Учитывая это, в настоящее время данную архитектуру все чаще называют принстонской, по названию университета, в котором работали Маучли и Эккерт.

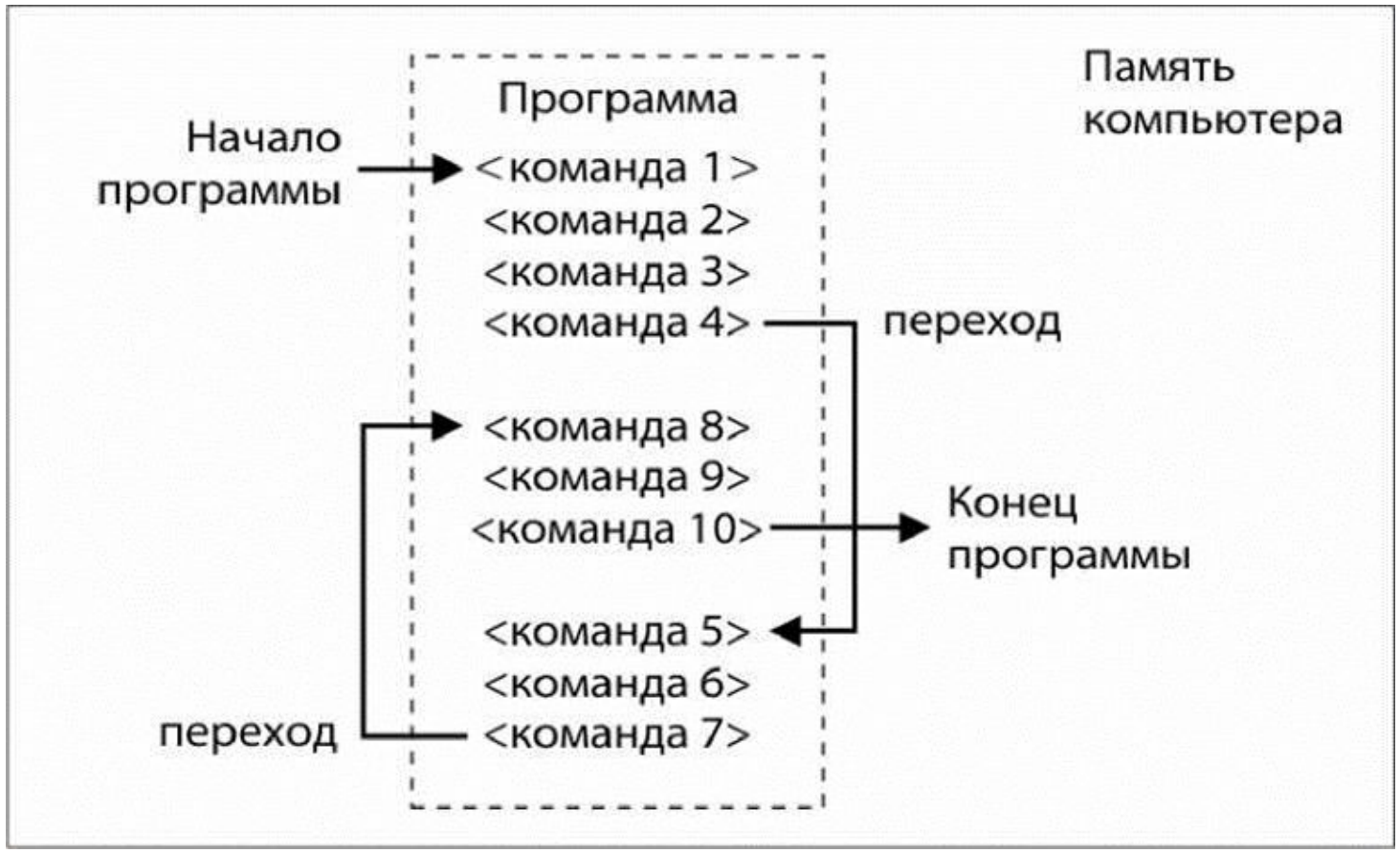
Машина фон Неймана состояла из следующих устройств:

- АЛУ – арифметико-логическое устройство для выполнения арифметических и логических операций;
- ОП – оперативная память для хранения выполняющейся в данный момент программы;
- УВВ – устройства ввода-вывода для ввода и вывода информации;
- УУ – управляющее устройство, которое организует работу компьютера следующим образом:
  - помещает в оперативную память коды программы из устройств ввода;
  - считывает из ячейки оперативной памяти и организует выполнение первой команды программы;
  - определяет очередную команду и организует ее выполнение;
  - постоянно синхронизирует работу устройств, имеющих различную скорость выполнения операций, путем приостановки выполнения программы.

В современных компьютерах роль АЛУ и УУ выполняет центральный процессор.

1.4. Фон-неймановская архитектура

Архитектура фон-неймановской машины основана на следующих **фундаментальных принципах** (на этих же принципах работают все современные компьютеры): принцип программного управления, принцип хранимой программы.



Принцип программного управления. Машиной управляет заранее подготовленная программа, представляющая собой последовательность инструкций (команд), расположенных в основной памяти машины линейно друг за другом. Команды исполняются последовательно, в порядке их записи. Такая последовательность выполнения команд называется **естественной**. Естественный порядок выполнения команд может нарушаться **командами перехода**

1.4. Фон-неймановская архитектура

Принцип хранимой программы. Для выполнения программы она должна быть предварительно помещена в оперативную память машины, а затем инициировано выполнение первой команды. Команды выбираются центральным процессором из оперативной памяти автоматически и интерпретируются в соответствии с принципом программного управления. Команды и данные программы хранятся одинаково в единой области памяти. Что считать командами, а что данными процессор определяет неявно в зависимости от использования информации.

Существуют и другие архитектуры отличные от фон-неймановской, например гарвардская архитектура. В гарвардской архитектуре команды и данные размещаются в разных видах памяти: в памяти команд и в памяти данных, соответственно. Это обеспечивает более высокую надежность работы машины, т. к. исключается возможность обращения с командами как с данными и наоборот.

В фоннеймановской архитектуре из-за того, что данные и команды располагаются совместно в единой памяти, возможны ситуации, когда процессор начинает интерпретировать данные как команды или команды как данные, что обычно приводит к сбою. С другой стороны совместное размещение команд и данных позволяет писать более гибкие программы.

Так или иначе, как уже было сказано, работа подавляющего большинства современных компьютеров основана именно на фон-неймановских принципах, включая и сложные многопроцессорные комплексы.

1.4. Фон-неймановская архитектура

Таким образом, согласно фон-неймановской архитектуре программа транслированная (откомпилированная или ассемблированная) в машинный код перед непосредственным выполнением всегда **помещается в память**.

Программа в памяти — это просто последовательность инструкций (команд) расположенных линейно друг за другом (в двоичном виде, разумеется).

Процессор считывает команды из памяти в порядке их записи и выполняет. В зависимости от считанной команды процессор выполняет определенные действия, это могут быть какие-либо вычисления или сигнал на какой-либо порт ввода-вывода, например указание жесткому диску переместить головку в определенное положение.

Работа компьютера определяется тремя основными устройствами: процессором, памятью и портами ввода-вывода.