Лекция 1

Вступление

Раньше мы рассматривали языки программирования высокого уровня. Преимущества этих языков - приближение к широкому пользователю, удобство программирования сложных логических задач.

Недостатки - сравнительно невысокая эффективность машинных программ. Все возможности ЭВМ полностью не используются. Поэтому для программирования системных задач, управления работой оборудования такие языки не достаточно эффективны.

Здесь преимущество имеют машинный ориентированные языки, одними из которых есть ассемблеры. Что такое ассемблер?

После трансляции программы из языка высокого уровня она превращается в последовательность машинных команд. Одна первичная команда реализуется несколькими машинными. Все имена пользователя (переменные, метки) заменяются адресами. Например, какая-то двухадресная команда имеет вид:

```
256 4725 0648
```

где первые три цифры - это код команды, дальше адрес первого операнда и адрес второго операнда.

В языке ассемблер коды команд и адреса заменяются их символическими именами. Например:

```
MOV X,Y
```

Это облегчает работу программиста - не нужно помнить машинные коды команд, самому заниматься делением памяти. Кроме того, ассемблер имеет средства модульного программирования - процедуры, макрокоманды.

Одна ассемблерная команда транслируется в одну машинную. Потому программы на ассемблере имеют намного больший объем, чем на языках высокого уровня. Но с помощью ассемблера можно использовать все возможности ЭВМ и эффективность выполнения таких программ будет очень высокой. Ясно, что программировать на ассемблере сложные логические программы будет очень трудоемким и потому нецелесообразным занятием. Назначение ассемблера для современных машин - системное программирование, для согласования отдельных программных систем, управления работой оборудования, систем передачи данных, и т.п.

ЭВМ	Розрядность	Максим. память, Мб
XT 8088/86	16	1
AT 80286	16	16
A 80386 DX	32	1024
80386 SX	16	16
80386 SL		

Не случайно, что программная оболочка Norton Commander состоит из 20 тыс. строк языком С и 10 тыс. - на ассемблере..

Если универсальные языки программирования незначительной мерой зависят от типа ЭВМ, то ассемблер полностью связан с конкретным типом. Необходимо знать особенности архитектуры определенного класса ЭВМ, то есть, высшей квалификации программиста.

С другой стороны, изучение ассемблера позволяет познакомиться со всеми возможностями ЭВМ, глубже понять, как в действительности реализуется программа на машинном уровне.

Раздел 1

Особенности архитектуры ПЭВМ на базе микропроцессоров 8088/86

1.1. Структура ПЭВМ

ПЭВМ состоит из нескольких функциональных устройств, которые реализуют арифметические и логические операции, управления, запоминания, ввода/вывода данных.

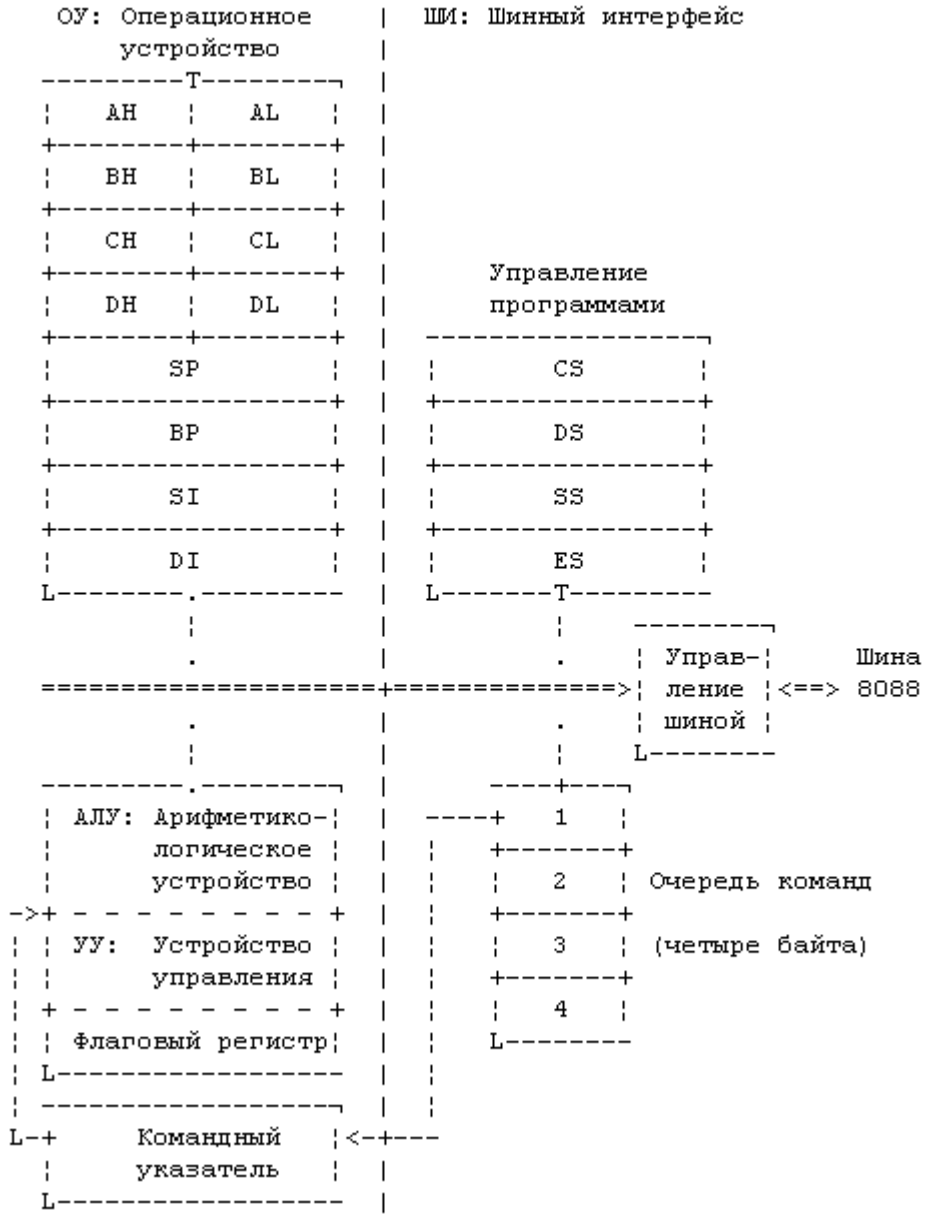


Рис.1.1. Операционное устройство и шинный интерфейс

Как показано на рис.1.1 процессор разделен на две части: операционное устройство (ОУ) и шинный интерфейс (ШИ). Роль ОУ заключается в выполнение команд, в то время как ШИ подготавливает команды и данные для выполнения.

Операционное устройство содержит арифметико-логическое устройство (АЛУ), устройство управления (УУ) и 14 регистров. Эти устройства обеспечивают выполнение команд, арифметические вычисления и логические операции (сравнение на больше, меньше или равно). Программы операционной системы и программы, которые выполняются, находятся в оперативной памяти (ОЗП).

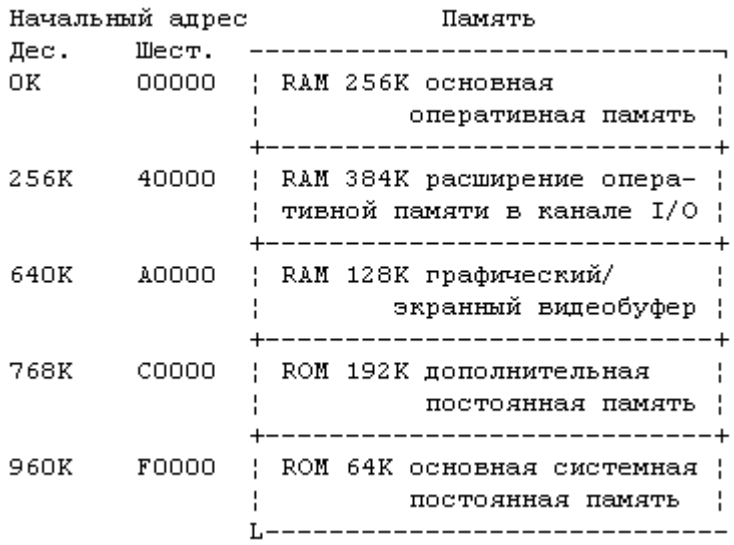


Рис.1.2. Карта физической памяти

Память RAM включает *первые три четверти* памяти, а ROM -- последнюю четверть. Первые 256К памяти находятся на системной плате. Для программиста доступные первые 640 К памяти.

1.1.1. Центральный процессор

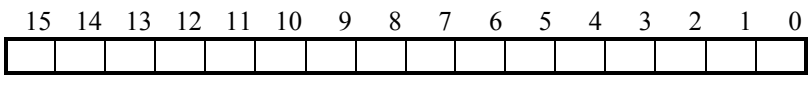
Как отмечалось, АЛУ и УУ объединяются в одном кремниевом кристалле и составляют центральный процессор (ЦП).

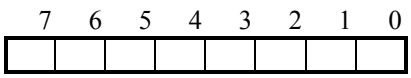
АЛУ выполняет основные арифметические операции (добавление, вычитание) и другие, логические операции. Эти действия реализуются специальными устройствами, например, сумматорами. Если бы между сумматором и ОЗУ не было никаких вспомогательных устройств, то нужно было бы очень много пересылок. Для уменьшения их количества В ЭВМ применяются специальные промежуточные ЗУ - *регистры*.

1.1.2. Память и особенности ее использования

Память состоит из отдельных ячеек. Внутри ЭВМ данные подаются в двоичной системе счисления. Поэтому ячейка состоит из ряда двоичных разрядов (*битов*), в каждом из которых записано 1 или 0.

8 бит составляют *байт*, который является наименьшей адресованной единицей данных. Ячейка состоит из одного *слова*, или двух байтов. Все разряды слова или регистра нумеруются справа налево от 0 до 15. В байте разряды нумеруются от 0 до 7.





Операции можно выполнять как над словами, так и над байтами. Каждая такая ячейка памяти определяется местом в памяти - *адресом* и данными, которые содержатся в ней, то есть, *содержанием*.

Поскольку байт является наименьшей адресованной единицей данных, то, чтобы к нему добраться, нужно иметь его адрес или номер. Все байты нумеруются от 0. В слове правый байт имеет меньший адрес, а левый - больший. Поэтому и называются соответственно -- *младший и старший*. Адрес слова совпадает с адресом его младшего байта. Следовательно, байты нумеруются специально от 000, а слова - лишь парной нумерацией. То есть, слово не может иметь непарного адреса.

Наибольший номер, который можно записать до 16 битов, это будет 2^{16}-1, а учитывая нулевой номер - 2^{16}. Следовательно, с использованием 16-ти битов можно пронумеровать $2^{16} = 2^6 * 2^{10}$ = 64 Кбайт. В 16-ой системе это будет наибольший адрес 0FFFFH.

Вся память разделяется на *сегменты* не больше 64 Кбайт. Одновременно можно использовать не больше 4 сегментов. Каждая ячейка будет определяться своим адресом относительно начала сегмента - смещением или *исполнительным адресом*. Адреса начала сегмента сохраняются в *регистрах сегментов*. Адрес слова в памяти будет равняться сумме исполнительного адреса и адреса сегмента (IP + CS) и называется *абсолютным адресом*. Адреса сегмента и исполнительная занимают 16 бит, а абсолютный адрес - 20 бит. Это достигается специальным превращением.

1.1.3. Регистры ЭВМ

Рассмотрим особенности использования отдельных регистров.

Регистр - это промежуточная память, вместимостью одно машинное слово (16 бит). В ЦП существуют много регистров, большинство из которых недосягаемые для программиста. Но есть несколько регистров, которые используются в программах на языке ассемблер.

За ними закреплены названия и имена, через которые к ним можно обращаться:

1. 4 регистра общего назначения (**AX, BX, CX, DX**).
2. 4 регистра указатели -**SP, BP, SI, DI**.
3. 4 сегментных регистра - **CS, DS, SS, ES**.
4. 2 управляющих регистра - указателя команд **IP** и регистр флагов **F**.

Назначение отдельных регистров и особенности использования рассмотрим дальше.

Если программе недостаточно одного сегмента данных, то она имеет возможность использовать еще три дополнительных сегмента данных. Но в отличие от основного сегмента данных, адрес которого содержится в сегментном регистре ds, при использовании дополнительных сегментов данных их адреса требуется указывать явно с помощью специальных префиксов переопределения сегментов в команде.

Адреса дополнительных сегментов данных должны содержаться в регистрах **ES, GS, FS** (extension data segment registers).

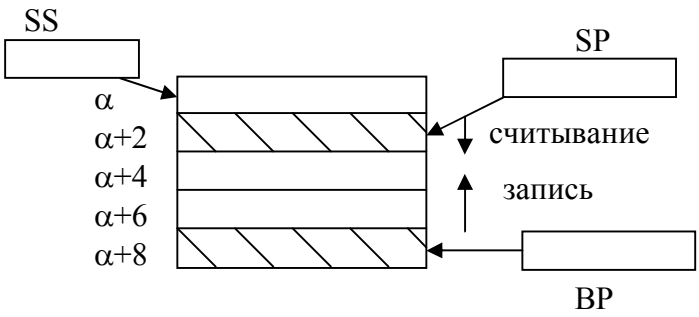


Рис. 1.3. Модель работы стека

В современных ЭВМ широко используются *стеки*, например, при работе с подпрограммами, при обработке прерываний, а также для запоминания промежуточных результатов. *Стек* - это участок памяти, которая заполняется в сторону меньших адресов, а освобождается в сторону увеличения адресов. Здесь реализуется дисциплина “последний пришел - первый обслуживайся”. Пример стека - пачка листов бумаги на столе.

Стек характеризуется своим базовым адресом, который записывается в базовый регистр SS, и адресом вершины, которая записывается в регистр указателя стека SP (stack pointer). В стек записываются слова (2 байта).