ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ЭКОНОМИКИ И СЕРВИСА

Микропроцессорные системы

Микропроцессоры – ядро МПС.

Вопросы

- Программная модель микропроцессора.
- Формат команды (Intel 8086).
- Система команд (Intel 8086).

Программная модель микропроцессора

- **Программная модель процессора** - это функциональная модель, используемая программистом при разработке программ в кодах ЭВМ или на языке ассемблера. В такой модели игнорируются многие аппаратные особенности в работе процессора.
- В процессоре 8086 имеется несколько быстрых элементов памяти, которые называются **регистрами**. Каждый из регистров имеет уникальную природу и предоставляет определенные возможности, которые другими регистрами или ячейками памяти не поддерживаются. Регистры разбиваются на четыре категории:
  - регистры общего назначения;
  - регистр флагов;
  - указатель команд;
  - сегментные регистры.
- Все регистры 16-разрядные.

Регистры общего назначения

	15	7 0
ax	ah	al
bx	bh	bl
cx	ch	cl
dx	dh	dl
si		
di		
sp		
bp		

Восемь регистров общего назначения процессора 8086 (каждый разрядностью 16 бит) используются в операциях большинства инструкций в качестве источника или приемника при перемещении данных и вычислениях, указателей на ячейки памяти и счетчиков. Каждый регистр общего назначения может использоваться для хранения 6-битового значения, в арифметических и логических операциях, может выполняться обмен между регистром и памятью (запись из регистра в память и наоборот).

Регистры общего назначения

- **Регистр AX** называют также накопителем (аккумулятором). Этот регистр всегда используется в операциях умножения или деления и является также одним из тех регистров, который можно использовать для наиболее эффективных операций (арифметических, логических или операций перемещения данных). Младшие 8 битов регистра AX называются также **регистром AL**, а старшие 8 битов - **регистром AH**. Это может оказаться удобным при работе с данными размером в байт. Таким образом, регистр AX можно использовать, как два отдельных регистра.

- **Регистр BX** может использоваться для ссылки на ячейку памяти (указатель), т.е. 16-битовое значение, записанное в BX, может использоваться в качестве части адреса ячейки памяти, к которой производится обращение. По умолчанию, когда BX используется в качестве указателя на ячейку памяти, он ссылается на нее относительно сегментного регистра DS. Как и регистры AX, CX и DX, регистр BX может интерпретироваться, как два восьмибитовых регистра - BH и BL.

- **Специализация регистра CX** - использование в качестве счетчика при выполнении циклов. Уменьшение значения счетчика и цикл - это часто используемый
элемент программы, поэтому в процессоре 8086 используется специальная команда для того, чтобы циклы выполнялись быстрее и были более компактными. Эта команда называется LOOP. Инструкция LOOP вычитает 1 из CX и выполняет переход, если содержимое регистра CX не равно 0. Регистр CX можно интерпретировать, как два 8-разрядных регистра - CH и CL.

- **Регистр DX** - это единственный регистр, которые может использоваться в качестве указателя адреса ввода-вывода в командах IN и OUT. Фактически, кроме использования регистра DX нет другого способа адресоваться к портам ввода-вывода с 256 по 65535. Другие уникальные качества регистра DX относятся к операциям
деления и умножения. Когда вы делите 32-битовое делимое на 16-битовый делитель, старшие 16 битов делимого должны быть помещены в регистр DX (младшие 16 битов делимого должны быть помещены в регистр AX). После выполнения деления остаток также сохраняется в регистре DX (частное от деления будет записано в AX). Аналогич-
но, когда вы перемножаете два 16-битовых сомножителя, старшие 16 битов произведения сохраняются в DX (младшие 16 битов записываются в регистр AX).
Регистр DX можно интерпретировать, как два 8-разрядных регистра - DH и DL.

Регистры общего назначения

- Как и регистр BX, **регистр SI** может использоваться, как указатель на ячейку памяти. Особенно полезно использовать регистр SI для ссылки на память в строковых командах процессора 8086, которые не только изменяют содержимое по адресу памяти, на который указывает SI, но к SI также добавляется или вычитается 1. Это может оказаться очень эффективным при организации доступа к последовательным ячейкам памяти (например, к строке текста). Кроме того, можно сделать так, что строковые команды будут автоматически определенное число раз повторять свои действия, так что отдельная команда может выполнить сотни, а иногда и тысячи действий.

- **Регистр DI** очень похож на регистр SI в том плане, что его можно использовать в качестве указателя ячейки памяти. При использовании его в строковых командах регистр DI несколько отличается от регистра SI. В то время как SI всегда используется в строковых командах, как указатель на исходную ячейку памяти (источник), DI всегда служит указателем на целевую ячейку памяти (приемник). Кроме того, в строковых командах регистр SI обычно адресуется к памяти относительно сегментного регистра DS, в то время как DI всегда адресуется к памяти относительно сегментного регистра ES. Когда SI и DI используются в качестве указателей на ячейки памяти в других командах (не строковых), то они всегда адресуются к памяти относительно регистра DS.
- Как и регистры BX, SI и DI, **регистр BP** также может использоваться в качестве указателя на ячейку памяти, но здесь есть некоторые отличия. Регистры BX, SI и DI обычно ссылаются на память относительно сегментного регистра DS (или, в случае использования в строковых командах регистра DI, относительно сегментного регистра ES), а регистр BP адресуется к памяти относительно регистра SS (сегментный регистр стека). Регистр BP создан для обеспечения работы с параметрами процедур, локальными переменными и другихслучаев, когда требуется адресация к памяти с использованием стека.

- **Регистр SP** называется также указателем стека. Это "наименее общий" из регистров общего назначения, поскольку он практически всегда используется для специальной цели - обеспечения стека. Стек - это область памяти, в которой можно сохранять значения и из которой они могут затем извлекаться по дисциплине "послед- ний-пришел-первый-ушел" (LIFO).

Регистр флагов

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
flags	**	**	**	**	OF	DF	IF	TF	SF	ZF	**	ZF	**	PF	**	CF

- Этот 16-разрядный регистр содержит всю необходимую информацию о состоянии процессора 8086 и результатах выполнения последней команды.

- Битовые флаги:
  OF - флаг переполнения;
  DF - флаг направления;
  IF - флаг прерывания;
  TF - флаг трассировки;
  SF - флаг знака;
  ZF - флаг нуля;
  AF - флаг дополнительного переноса;
  PF - флаг четности;
  CF - флаг переноса;
  ** - бит не используется, состояние не определено.

- Регистр флагов не считывается и не модифицируется непосредственно. Вместо этого в системе команд микропроцессора предусмотрены специальные команды, с помощью которых программист может задать необходимое ему состояние любого из флагов (кроме TF). Содержимое регистра флагов используется микропроцессором при выполнении команд условного перехода, циклических сдвигов, операций с цепочками байт или слов.

Регистр флагов

- **Флаг переполнения OF** сигнализирует о потере старшего бита результата сложения или вычитания. Имеется специальная команда прерывания при переполнении, которая генерирует программное прерывание.

- **Флаг направления DF** определяет порядок сканирования цепочек байт или слов в соответствующих командах: от меньших адресов к большим (DF = 0) или наоборот (DF = 1).

- **Флаг прерывания IF** определяет реакцию процессора на запросы внешних прерываний по входу INT. Если IF = 0, запросы прерываний игнорируются (говорят также, что прерывания запрещены или замаскированы), а если IF = 1, процессор распознает запросы на прерывания и реагирует на них соответствующим образом.

- Установка в состояние 1 **флага трассировки TF** переводит процессор в одношаговый (покомандный) режим работы, который применяется при отладке программ. В этом режиме процессор автоматически генерирует внутреннее прерывание после выполнения каждой команды с переходом к соответствующей подпрограмме обработки, которая может, например, демонстрировать содержимое регистров процессора на экране дисплея.

- **Флаг знака SF** повторяет значение старшего бита результата, который при спользовании дополнительного кода соответствует знаку числа. Флаг нуля ZF сигнализирует о получении нулевого результата операции. Флаг вспомогательного переноса AF фиксирует перенос (заем) из младшей тетрады в старшую 8- или 16-битного результата. Он необходим только для команд десятичной арифметики.

- **Флаг четности (паритета) PF** фиксирует наличие четного числа единиц в младших 8 разрядах результата операции. Этот флаг предназначен для контроля правильности передачи данных.

- **Флаг CF** фиксирует значение переноса (заема), возникающего при сложении или вычитании байт или слов, а также значение выдвигаемого бита при сдвиге операнда.

Указатель команд

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP																

- Указатель команд (**регистр IP**) всегда содержит смещение в памяти, по которому хранится следующая выполняемая команда. Когда выполняется одна команда, указатель команд перемещается таким образом, чтобы указывать на адрес памяти, по которому хранится следующая команда. Обычно следующей выполняемой командой является команда, хранимая по следующему адресу памяти, но некоторые команды, такие, как вызовы или переходы, могут привести к тому, что в указатель команд будет загружено новое значение. Таким образом, будет выполнен переход на другой участок программы. Значение счетчика команд нельзя прочитать или записать непосредственно. Загрузить в указатель команд новое значение может только специальная команда перехода. Указатель команд сам по себе не определяет адрес, по которому находится следующая выполняемая команда. Картину здесь опять усложняет сегментная организация памяти процессора 8086. Для извлечения команды предусмотрен регистр CS, где хранится базовый адрес, при этом указатель команд задает смещение относительно этого базового адреса.

Сегментные регистры

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cs																
ds																
es																
ss																

- Фактически, единственная операция, которую можно выполнять с сегментными регистрами, состоит в копировании значений между сегментными регистрами и другими общими регистрами или памятью.
  Особенность использования сегментов состоит в том, что каждая ячейка памяти адресуется через многие возможные сочетания "сегмент:смещение". Например, адрес памяти 100h адресуется с помощью следующих значений "сегмент:смещение": 0:100h, 1:F0h, 2:E0h и т.д., так как при вычислении всех этих пар "сегмент:смещение" получается значение адреса 100h.

- Аналогично регистрам общего назначения каждый сегментный регистр играет свою, конкретную роль. Регистр CS указывает на код программы, DS указывает на данные, SS - на стек. Сегментный регистр ES - это дополнительный сегмент, который может использоваться так, как это необходимо. Рассмотрим сегментные регистры более подробно. Регистр CS указывает на начало блока памяти объемом 64К, или сегмент кода, в котором находится следующая выполняемая команда. Следующая команда, которую нужно выполнить, находится по смещению, определяемому в сегменте кода регистром IP, то есть на нее указывает адрес (в форме "сегмент:смещение") CS:IP. Процессор 8086 никогда не может извлечь команду из сегмента, отличного от того, который определяется регистром CS.

Сегментные регистры

- **Регистр CS** можно изменять с помощью многих команд, включая отдельные команды перехода, вызовы и возвраты управления. Ни при каких обстоятельствах регистр CS нельзя загрузить непосредственно. Никакие другие режимы адресации или указатели памяти, отличные от IP, не могут нормально работать относительно регистра CS.

- **Регистр DS** указывает на начало сегмента данных, которые представляет собой блок памяти объемом 64К, в котором находится большинство размещенных в памяти операндов. Обычно для ссылки на адреса памяти используются смещения, предполагающие использование регистров BX, SI или DI. В основном сегмент данных представляет собой то, о чем говорит его название: как правило это сегмент, в котором находится текущий набор данных.

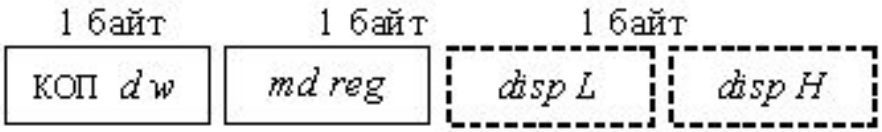
- **Регистр ES** указывает на начало блока памяти объемом 64К, который называется дополнительным сегментом. Как и подразумевает его название, дополнительный сегмент не служит для какой-то конкретной цели, но доступен тогда, когда в нем возникает необходимость. Иногда сегмент ES используется для выделения дополнительного блока памяти объемом 64К для данных. Однако доступ к памяти в дополнительном сегменте менее эффективен, чем доступ к памяти в сегменте данных. Особенно полезен дополнительный сегмент, когда используются строковые команды. Все строковые команды, которые выполняют запись в память, используют в качестве адреса, по которому нужно выполнить запись, пару регистров ES:DI. Это означает, что регистр

Формат команд

- Микропроцессор Intel-8086 (К1810ВМ80) имеет **двухадресную систему команд**. Ее особенностью является отсутствие команд, использующих оба операнда из оперативной памяти. Исключение составляют лишь команды пересылки и сравнения цепочек байт или слов. Таким образом, в командах допустимы следующие сочетания операндов: RR, RS, RI, SI. Здесь R обозначает операнд, находящийся в одном из регистров регистровой памяти микропроцессора, S - операнд, находящийся в оперативной памяти, адрес которого формируется по одному из допустимых способов адресации, I - непосредственный операнд, закодированный в адресном поле самой команды. Формат команды определяется способом адресации операнда, находящего в оперативной памяти, длиной используемого непосредственного операнда, а также наличием и длиной смещения, используемого при относительных режимах адресации.
- **Непосредственная адресация** предполагает, что операнд занимает одно из полей команды и, следовательно, выбирается из оперативной памяти одновременно с ней. В зависимости от форматов обрабатываемых процессором данных непосредственный операнд может иметь длину 8 или 16 бит, что в дальнейшем будем обозначать data8 и data16 соответственно.
- Механизмы адресации операндов, находящихся в регистровой памяти и в оперативной памяти, существенно различаются. К регистровой памяти допускается лишь прямая регистровая адресация. При этом в команде указывается номер регистра, содержащего операнд. 16-разрядный операнд может находиться в регистрах AX, BX, CX, DX, DI, SI, SP, BP, а 8-разрядный - в регистрах AL, AH, BL, BH, CL, CH, DL, DH.
- Адресация оперативной памяти имеет свои особенности, связанные с ее разбиением на сегменты и использованием сегментной группы регистров для указания начального адреса сегмента. 16-разрядный адрес, получаемый в блоке формирования адреса операнда на основе указанного режима адресации, называется **эффективным адресом** (ЭА). Иногда эффективный адрес обозначается как EA (effective address). 20-разрядный адрес, который получается сложением эффективного адреса и увеличенного в 16 раз значения соответствующего сегментного регистра, называется **физическим адресом** (ФА).
- Именно физический адрес передается из микропроцессора по 20-ти адресным линиям, входящим в состав системной шины, в оперативную память и используется при обращении к ее ячейке на физическом уровне. При получении *эффективного адреса* могут использоваться все основные режимы адресации, рассмотренные выше, а также некоторые их комбинации.
- **Прямая адресация** предполагает, что эффективный адрес является частью команды. Так как ЭА состоит из 16 разрядов, то и соответствующее поле команды должно иметь такую же длину.
- При регистровой **косвенной адресации** эффективный адрес операнда находится в базовом регистре BX или одном из индексных регистров DI либо SI

Формат команд

- Форматы двухоперандных команд представлены на рисунке. Пунктиром показаны поля, которые в зависимости от режима адресации могут отсутствовать в команде.



а) формат команд типа *RR* и *RS*

б) формат команды с непосредственным операндом

- Поле КОП содержит код выполняемой операции. Признак w указывает на длину операндов. При w = 1 операция проводится над словами, а при w = 0 - над байтами.
- Второй байт команды, называемый **постбайтом**, определяет операнды, участвующие в операции. Поле reg указывает регистр регистровой памяти.
- Поля md и r/m задают режим адресации второго операнда.

Система команд

- В общем случае система команд процессора включает в себя следующие 5 основных групп команд:
  - команды пересылки данных;
  - арифметические команды;
  - логические команды;
  - команды переходов.
  - команды управления
- **Команды пересылки** данных не требуют выполнения никаких операций над операндами. Операнды просто пересылаются (точнее, копируются) из источника (Source) в приемник (Destination). Источником и приемником могут быть внутренние регистры процессора, ячейки памяти или устройства ввода/вывода. АЛУ в данном случае не используется.
- **Арифметические команды** выполняют операции сложения, вычитания, умножения, деления, увеличения на единицу (инкрементирования), уменьшения на единицу (декрементирования) и т.д. Этим командам требуется один или два входных операнда. Формируют команды один выходной операнд.
- **Логические команды** производят над операндами логические операции, например, логическое И, логическое ИЛИ, исключающее ИЛИ, очистку, инверсию, разнообразные сдвиги (вправо, влево, арифметический сдвиг, циклический сдвиг). Этим командам, как и арифметическим, требуется один или два входных операнда, и формируют они один выходной операнд.
- Наконец, **команды переходов** предназначены для изменения обычного порядка последовательного выполнения команд. С их помощью организуются переходы на подпрограммы и возвраты из них, всевозможные циклы, ветвления программ, пропуски фрагментов программ и т.д. Команды переходов всегда меняют содержимое счетчика команд. Переходы могут быть условными и безусловными. Именно эти команды позволяют строить сложные алгоритмы обработки информации.

Система команд

Ниже приведены только некоторые команды процессора:

- **Непосредственная загрузка значения в регистр**

1011 W DST младший байт [старший байт]

Бит W указывет, что должен быть загружен байт (W=0) или слово. Три бита DST указывают, в какой регистр производится загрузка.

- **Загрузка значения из регистра в регистр**

1000101 W 11 DST SRC

Биты DST и SRC указывают регистр-получатель и регистр-источник (SRC интерпретируется аналогично DST).

- **Увеличение значения в регистре на единицу**

01000 REG

- **Сложение значений в двух регистрах**

0000001 W 11 DST SRC

Результат сложения заносится в регистр, определяемый DST.

- **Вычитание**

0010101 W 11 DST SRC

- **Умножение значения в AL (AX) на значение в регистре**

1111011 W 11100 SRC

Результат перемножения восьмиразрядных значений записывается в AX, шестнадцатиразрядных - в DX:AX.

- **Деление значения в AX (DX:AX) на значение в регистре**

1111011 W 11110 SRC

- **Логическое сложение (ИЛИ)**

0000101 W 11 DST SRC

- **Логическое умножение (И)**

0010001 W 11 DST SRC

- **Условный переход**

0111 COND смещение

Эта команда увеличивает значение указателя команды на указанное число байт, если выполнено условие, определяемое четырьмя битами COND (т.е. если установлены определенные биты регистра PSW).

В противном случае не выполняет никаких действий.

Система команд

- Все коды воспринимаются процессором, но много ли вам говорит последовательность B1 0A F6 F1 B1 1F B5 30 02 C5? Вместо кодов обычно используется символический язык (язык ассемблера), в котором каждая команда процессора представляется символическим именем, и именами регистров, которые в ней используются:

- **mov DST,SRC** - загрузка в DST значения из SRC
- **push SRC** - запись SRC в стек
- **pop DST** - загрузка слова из стека в DST
- **inc DST** - увеличение DST на единицу
- **add DST,SRC** - сложение DST и SRC
- **div SRC** - деление на значение в SRC
- **and DST,SRC** - логическое умножение DST и SRC
- **jz LBL** - условный переход, если ноль
- **jmp LBL** - безусловный переход (LBL - метка)
- **call LBL** - вызов подпрограммы
- **int NUM** - вызов подпрограммы обработки прерывания
- **ret** - возврат из подпрограммы
- **iret** - возврат из подпрограммы обработки прерывания

- Кроме того, в программе на языке ассемблера могут быть описаны переменные, например:

Buff db 128 dup - массив из 128 байт
P dw - слово

Описание каждой переменной состоит из имени, длины (db - байт, dw - слово) и, возможно, количества байт/слов (dup).

Имена переменных могут указываться в командах, например:

```
mov DI,P
mov AX,Buff[DI]
```

Запись программы с помощью этих обозначений точно соответствует машинному коду, но гораздо лучше читается.

Литература

1. Айден К. , Фибельман Х. , Крамер М. Аппаратные средства PC, Пер. с нем. - СПб.: BHV - Санкт-Петербург,1996. - 544 с.,ил.
2. Борзенко А.Е. IBM PC: Устройство, ремонт, модернизация. М.: ТОО фирма ∀КомпьютерПресс", 1995,298 с.
3. Токхайм Р. Микропроцессоры: Курс и упражнения / Пер. с англ. Под ред. Грасевича. М.: Энергоатомиздат, 1987. 338 с.
4. Вьюхин В.В. и др. Информатика и вычислительная техника. М.:Высшая школа, 1992,-286 с.
5. Опадчий Ю.Ф., Глудкин О.П. Аналоговая и цифровая электроника. Учебник для вузов.-М.: Горячая линия-Телеком, 1999.-768с.
6. Гук М. Процессоры Intel: от 8086 до Pentium II.-Спб.: Питер, 1998. -224 с.
7. Калабеков Б.А. Цифровые устройства и микропроцессорные системы. - М.: Радио и связь, 1997.