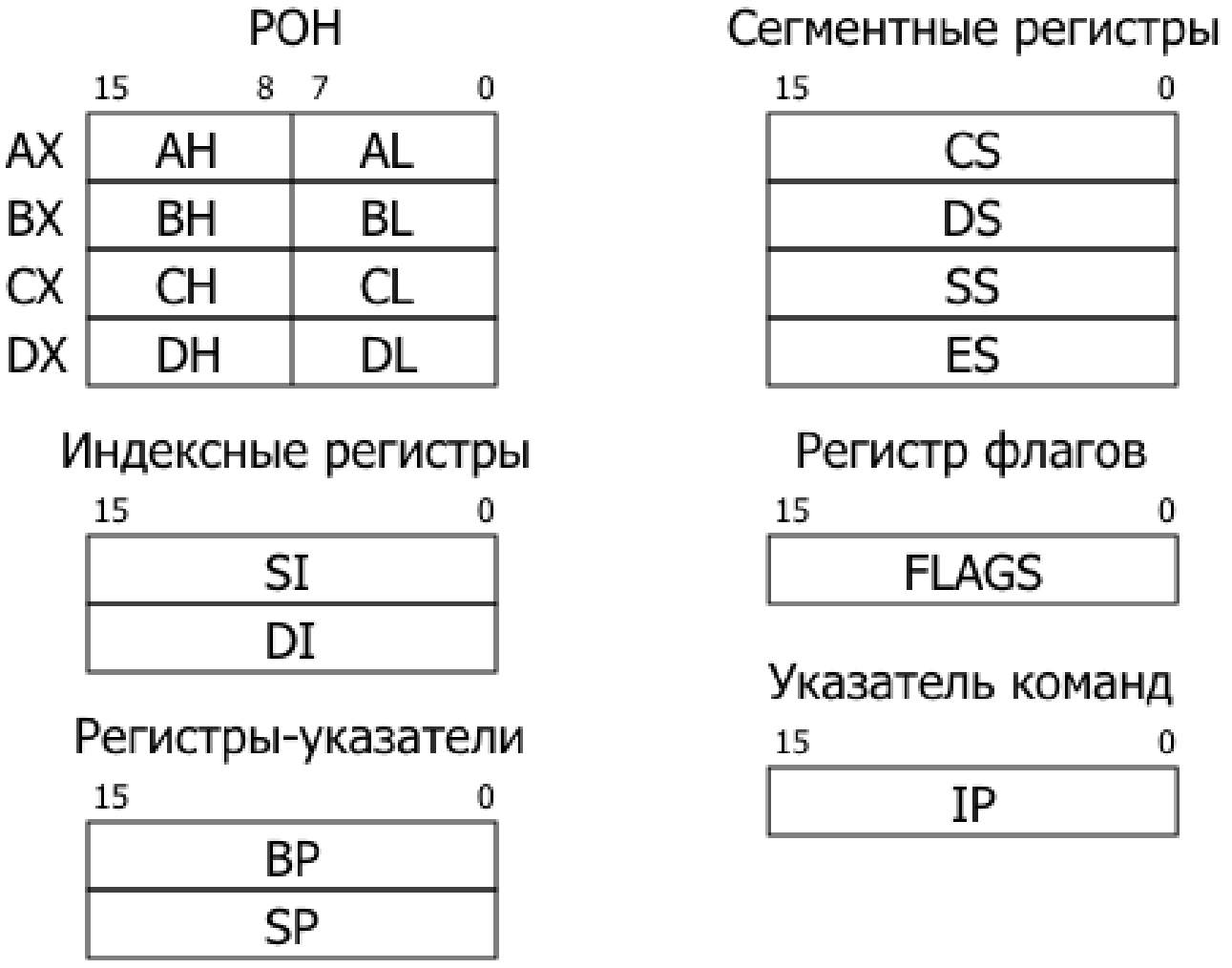
Учебный курс. Часть 4. Регистры процессора 8086

Автор: xrnd | Рубрика: Учебный курс | 13-03-2010 |
Распечатать запись

Для того, чтобы писать программы на ассемблере, нам необходимо знать, какие регистры процессора существуют и как их можно использовать. Все процессоры архитектуры x86 (даже многоядерные, большие и сложные) являются дальними потомками древнего Intel 8086 и совместимы с его архитектурой. Это значит, что программы на ассемблере 8086 будут работать и на всех современных процессорах x86.

Все внутренние регистры процессора Intel 8086 являются 16-битными:



Всего процессор содержит 12 программно-доступных регистров, а также регистр флагов (FLAGS) и указатель команд (IP).

Регистры общего назначения (РОН) AX, BX, CX и DX используются для хранения данных и выполнения различных арифметических и логических операций. Кроме того, каждый из этих регистров поделён на 2 части по 8-бит, с которыми можно работать как с 8-битными регистрами (AH, AL, BH, BL, CH, CL, DH, DL). Младшие части регистров имеют в названии букву L (от слова *Low*), а старшие H (от слова *High*). Некоторые команды неявно используют определённый регистр, например, CX может выполнять роль счетчика цикла.

Индексные регистры предназначены для хранения индексов при работе с массивами. SI (*Source Index*) содержит индекс источника, а DI (*Destination Index*) — индекс приёмника, хотя их можно использовать и как регистры общего назначения.

Регистры-указатели BP и SP используются для работы со стеком. BP (*Base Pointer*) позволяет работать с переменными в стеке. Его также можно использовать в других целях. SP (*Stack Pointer*) указывает на вершину стека. Он используется командами, которые работают со стеком. (Про стек я подробно расскажу в отдельной части учебного курса)

Сегментные регистры CS (*Code Segment*), DS (*Data Segment*), SS (*Stack Segment*) и ES (*Enhanced Segment*) предназначены для обеспечения сегментной адресации. Код находится в сегменте кода, данные — в сегменте данных, стек — в сегменте стека и есть еще дополнительный сегмент данных. Реальный физический адрес получется путём сдвига содержимого сегментного регистра на 4 бита влево и прибавления к нему смещения (относительного адреса внутри сегмента). Подробнее о сегментной адресации рассказывается в части 31.

COM-программа всегда находится в одном сегменте, который является одновременно сегментом кода, данных и стека. При запуске COM-программы сегментные регистры будут содержать одинаковые значения.

Указатель команд IP (*Instruction Pointer*) содержит адрес команды (в сегменте кода). Напрямую изменять его содержимое нельзя,

но процессор делает это сам. При выполнении обычных команд значение IP увеличивается на размер выполненной команды. Существуют также команды передачи управления, которые изменяют значение IP для осуществления переходов внутри программы.

Регистр флагов FLAGS содержит отдельные биты: флаги управления и признаки результата. Флаги управления меняют режим работы процессора:

- D (*Direction*) — флаг направления. Управляет направлением обработки строк данных: DF=0 — от младших адресов к старшим, DF=1 — от старших адресов к младшим (для специальных строковых команд).
- I (*Interrupt*) — флаг прерывания. Если значение этого бита равно 1, то прерывания разрешены, иначе — запрещены.
- T (*Trap*) — флаг трассировки. Используется отладчиком для выполнения программы по шагам.

Признаки результата устанавливаются после выполнения арифметических и логических команд:

- S (*Sign*) — знак результата, равен знаковому биту результата операции. Если равен 1, то результат — отрицательный.
- Z (*Zero*) — флаг нулевого результата. ZF=1, если результат равен нулю.
- P (*Parity*) — признак чётности результата.
- C (*Carry*) — флаг переноса. CF=1, если при сложении/вычитании возникает перенос/заём из старшего разряда. При сдвигах хранит значение выдвигаемого бита.
- A (*Auxiliary*) — флаг дополнительного переноса. Используется в операциях с упакованными двоично-десятичными числами.
- O (*Overflow*) — флаг переполнения. CF=1, если получен результат за пределами допустимого диапазона значений.

Не волнуйтесь, если что-то показалось непонятным. Из дальнейшего объяснения станет ясно, что к чему и как всем этим пользоваться 🙂

Следующая часть »

Комментарии:

Falcon
13-10-2010 14:29

Грубо говоря о сегментных регистрах :
При записании програмы на жёсткий диск , сегменты программы будут сохранятся на опрелённых местах в винчестере . И при запуска екзешника , программа должна выполнять эти сегменты , которые записаны в память . Но как найти эти сегменты в огромнейшим винчестере ? Конечно , сегментные регистры , они очень важны и всего их 4 — CS : сегмент комант (текст программы) , DS : сегмент даты (данные) , SS даёт нам возможность обращения к стекам , ES не помню . Сегментные регистры носят начальную адрес закрепленных за ними сегментов . В дальнейшем любое обращение к ячейкам выполняется через указания сегментов и номер байта ячейки .

[Ответить]

xrnd
13-10-2010 16:28

Нет, прямо на винчестер сегменты не записываются.
Сегментные регистры инициализируются операционной системой DOS при запуске программы. Потому что она занимается распределением памяти для программ.
А содержимое сегмента считывается из исполняемого файла. Если интересно, как оно там хранится, можешь найти структуру MZ EXE-файлов. Для COM-файла всё очень просто — содержимое файла загружается в единственный сегмент, начиная с адреса 100h

[Ответить]

Егор
18-02-2011 15:08

ES (экстракодовый сегмент) используется для передачи данных между ячейками ОП, не попадающими в непрерывную область размером 64 К байт. Например, если необходимо переслать данные из байта с номером 100000 в байт с номером 200000, то это невозможно сделать в пределах одного сегмента данных (DS). Тогда первый байт включается в сегмент DS, а второй — в сегмент ES и выполняются необходимые операции.

[Ответить]

magi
13-06-2012 11:03

Не путайте сектор винчестера и сегменты исполяемой программы — это разные понятия. Сегментом называется условно выделенная область адресного пространства определённого размера в оперативной памяти.

[Ответить]

Мдфв
17-09-2015 11:17

Да нет. Все это храниться в программе, которую вы запустите. Прочитав оттуда уже происходит расталкивания данных по сегментам. А так на винте это общий файлик

[Ответить]

diger
29-04-2011 21:19

а как насчёт 386 ? какие у него есть регистры ?

[Ответить]

xrnd
06-05-2011 00:21

У 386-го регистры 32-битные: EAX, EBX, ECX, EDX, EDI, ESI и т.д. Ещё есть некоторые дополнительные регистры, например FS и GS. Различий много, в комментарии всё не рассказать.

[Ответить]

Владислав
25-10-2011 18:43

Хотелось бы узнать, что подразумевается под размером выполненной команды.

[Ответить]

xrnd
27-10-2011 00:55

Количество байтов, которое занимает команда в памяти.
Команды могут иметь разную длину. От 1 до 14 байтов.
Процессор декодирует команду и выполняет её, затем переходит к следующей.

[Ответить]

Andrew
06-11-2011 19:48

O (Overflow) — флаг переполнения. CF=1, если получен результат за пределами допустимого диапазона значений.
CF = 1? Или O = 1?

[Ответить]

Денис
08-11-2011 19:09

P (Parity) — признак чётности результата. Как узнать когда числа четные а когда не четные?

C (Carry) — флаг переноса. CF=1, если при сложении/вычитании возникает перенос/заём из старшего разряда. При сдвигах хранит значение выдвигаемого бита.
Что за перенос/заем из старшего разряда? Что за выдвигаемый бит?

A (Auxiliary) — флаг дополнительного переноса. Используется в операциях с упакованными двоично-десятичными числами.
Опять непонятно что за перенос? И что за упакованные двоично-десятичные числа?

[Ответить]

Сергей Александрович
19-04-2012 20:51

Денис! Найди в Инете книжку В.Юрова «АССЕМБЛЕР» и сразу ВСЕ вопросы отпадут.
Если по тексту вопроса, то флаг P (PF) = 1, когда в результате операции в двоичном представлении имееися четное число единичных битов. Флаг CF = 1 если при сложениях или сдвигах крайний бит вытесняется за битовую сетку операнда. AF — то же самое, но за пределы полубайта. Упакованные Дв-Дес Числа (BCD формат) — это когда в каждый байт пишутся 2 цифры числа. Цепочка байт м.б. длинной.

[Ответить]

artyfact
26-06-2012 02:44

1) не сами числа, а количество единиц в двоичном коде числа. Например: 2(10) = 10(2) — PF = 0, 3(10) = 11(2) — PF = 1;
2) допустим значение регистра ax = FFFF, тогда ax + 1 = 1 0000 т. е. возникает перенос единицы из старшего разряда регистра, здесь — переполнение, в таких случаях CF = 1. Тоже самое произойдет если вместо ax взять al;
3) флаг AF = 1 если выполняется следующие условие (может не совсем корректное, но на примере должно быть понятно): сумма 2-х чисел в 16-ти разрядной сис. превышает либо равна 16. Пример: 1A + 7 = 1A + 07 (A + 7 = 10 + 7 = 17 = 1*16 + 1), таким образом AF = 1.

[Ответить]

Андрей
11-01-2012 18:00

А какие адреса у регистров общего назначения? ну или как узнать.

[Ответить]

vv20
12-01-2012 02:32

А где находится T (Trap) флаг? Не смог его найти.

[Ответить]

Denis
28-04-2013 09:15

Здесь, кажется, ошибка?
«O (Overflow) — флаг переполнения. CF=1, если получен » — название флага «OF»?

[Ответить]

Александр
28-06-2013 18:04

непонимаю зачем в учить людей ездить на москвиче 16-ти битном, когда под рукой есть отличный 64-х битный порш с автоматической коробкой передач, гидроусилитель руля, и еще целая куча наворотов, к которым придется потом привыкать или еще страшнее придется ПЕРЕУЧИВАТСЯ после отсталого москвича.

[Ответить]

Айдар
03-03-2014 17:15

А можно какой-то способ как пробовать примеры на x64 платформе. У меня не получается запустить не один пример. Пишет что не совместимы с 16 битными приложениями.

[Ответить]

Кто-то
27-07-2015 13:18

через dosbox

[Ответить]

Илья
02-01-2015 22:15

Все эти регистры вместе представляют собой кэш?

[Ответить]

Илья
02-01-2015 23:27

«P (Instruction Pointer) содержит адрес команды (в сегменте кода)» — это как??

[Ответить]

Максим
21-12-2015 16:45

Рискну предположить, что в описании флага переполнения, возможно, допущена опечатка. В частности, там написано:

O (Overflow) — флаг переполнения. CF=1, если получен результат за пределами допустимого диапазона значений.

… хотя, вероятно, должно быть OF (вместо CF).

[Ответить]

aliaksandr
22-12-2016 10:07

Auxiliary Flag is used as CF but when working with BCD

[Ответить]

Ваш комментарий

Имя *

Почта (скрыта) *

Сайт

Добавить

Уведомить меня о новых комментариях по email.

Уведомлять меня о новых записях почтой.