Программная модель микропроцессора

Микропроцессор содержит 32 регистра.

- 16 пользовательских регистров;
- 16 системных регистров.

<u>Пользовательскими</u> называются регистры, которые используются при написании программ. Эти регистры делятся на:

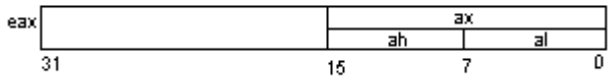
- регистры данных;
- регистры указатели;
- сегментные регистры;
- регистр флагов;
- регистр указателя команды.

Системные регистры используются в защищенном режиме работы.

Регистры данных

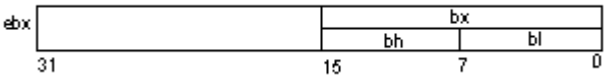
Так как эти регистры физически находятся в микропроцессоре внутри арифметико-логического устройства (АЛУ), то их еще называют *регистрами АЛУ*:

- **eax/ax/ah/al** (**<u>Accumulator register</u>**) — *аккумулятор*.



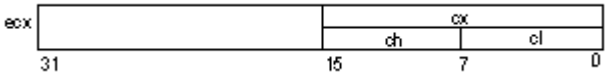
Применяется для хранения промежуточных данных. В некоторых командах использование этого регистра обязательно;

- **ebx/bx/bh/bl** (**<u>Base register</u>**) — *базовый* регистр.



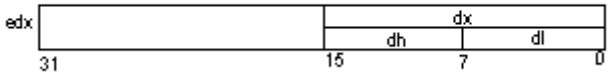
Применяется для хранения базового адреса некоторого объекта в памяти;

- **ecx/cx/ch/cl** (**<u>Count register</u>**) — *регистр-счетчик*.



Применяется в командах, производящих некоторые повторяющиеся действия.

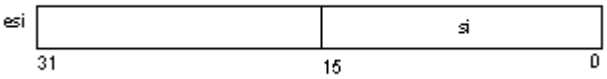
- **edx/dx/dh/dl** (**<u>Data register</u>**) — регистр *данных*.



Так же, как и регистр *аккумулятор*, он хранит промежуточные данные. В некоторых командах его использование обязательно.

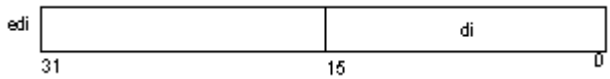
Регистры указатели.

- **esi/si** (**<u>Source Index register</u>**) — *индекс источника*.



Этот регистр в цепочечных операциях содержит текущий адрес элемента в цепочке-источнике;

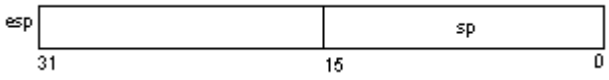
- **edi/di** (**<u>Destination Index register</u>**) — *индекс приемника* (получателя).



Этот регистр в

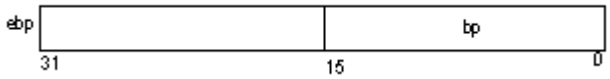
цепочечных операциях содержит текущий адрес в цепочке-приемнике.

- **esp/sp** (**Stack Pointer register**) — регистр *указателя стека*.



Содержит указатель вершины стека в текущем сегменте стека.

- **ebp/bp** (**Base Pointer register**) — регистр *указателя базы кадра стека*.



Предназначен для организации произвольного доступа к данным внутри стека.

Сегментные регистры

Микропроцессор содержит шесть сегментных регистров: *cs, ss, ds, es, gs, fs*. Любая программа состоит из частей называемых *сегментами*. Для того чтобы указать на сегменты, к которым программа имеет доступ, предназначены *сегментные регистры*. В этих регистрах содержатся адреса памяти, с которых начинаются соответствующие сегменты. Логика обработки машинной команды построена так, что при выборке команды, доступе к данным программы или к стеку неявно используются адреса во вполне определенных сегментных регистрах. Микропроцессор поддерживает следующие типы сегментов:

1. ***Сегмент кода***. Содержит команды программы.
   Для доступа к сегменту служит регистр **cs** (**code segment register**)
2. ***Сегмент данных***. Содержит обрабатываемые программой данные.
   Для доступа к этому сегменту служит регистр **ds** (**data segment register**)
3. ***Сегмент стека***. Работу со стеком микропроцессор организует по принципу: *последний записанный в эту область элемент выбирается первым*. Для доступа к этому сегменту служит регистр **ss** (**stack segment register**)
4. ***Дополнительный сегмент данных***. Алгоритмы выполнения большинства машинных команд предполагают, что обрабатываемые ими данные расположены в сегменте данных, адрес которого находится в сегментном регистре *ds*. Если программе недостаточно одного сегмента данных, то можно 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. Адреса дополнительных сегментов данных должны содержаться в регистрах **es, gs, fs** (**extension data segment registers**).

Регистр флагов.

eflags/flags (flag register) — регистр *флагов*. Разрядность — 32/16 бит. Отдельные биты данного регистра называются флагами. Младшая часть этого регистра полностью аналогична регистру *flags* для i8086.

Флаги можно разделить на три группы:

- *флаги состояния*.
- *флаг управления*.
- *системные флаги.*

Флаги состояния.

Эти флаги могут изменяться после выполнения машинных команд и отражают особенности результата исполнения арифметических или логических операций. Это дает возможность анализировать состояние вычислительного процесса и реагировать на него с помощью команд условных переходов и вызовов подпрограмм.

Мнемоника флага	Флаг	Номер бита в *eflags*	Содержание и назначение
cf	Флаг переноса **(Carry Flag)**	0	1 — арифметическая операция произвела перенос из старшего бита результата. Старшим является 7, 15 или 31-й бит в зависимости от размерности операнда; 0 — переноса не было
pf	Флаг паритета **(Parity Flag)**	2	1 — 8 младших разрядов (этот флаг — только для 8 младших разрядов операнда любого размера) результата содержат четное число единиц; 0 — 8 младших разрядов результата содержат нечетное число единиц
af	Вспомогательный флаг переноса **(Auxiliary carry Flag)**	4	Только для команд работающих с BCD-числами. Фиксирует факт заема из младшей тетрады результата: 1 — в результате операции сложения был произведен перенос из разряда 3 в старший разряд или при вычитании был заем в разряд 3 младшей тетрады из значения в старшей тетраде; 0 — переносов и заемов в(из) 3 разряд(а) младшей тетрады результата не было
zf	Флаг нуля **(Zero Flag)**	6	1 — результат нулевой; 0 — результат ненулевой
sf	Флаг знака **(Sign Flag)**	7	Отражает состояние старшего бита результата (биты 7, 15 или 31 для 8, 16 или 32-разрядных операндов соответственно): 1 — старший бит результата равен 1; 0 — старший бит результата равен 0
of	Флаг переполнения **(Overflow Flag)**	11	Флаг of используется для фиксирования факта потери значащего бита при арифметических операциях: 1 — в результате операции происходит перенос (заем) в(из) старшего, знакового бита результата (биты 7, 15 или 31 для 8, 16 или 32-разрядных операндов соответственно); 0 — в результате операции не происходит переноса (заема) в(из) старшего, знакового бита результата

Флаг управления.

Обозначается **df** (Directory Flag). Используется цепочечными командами. Значение флага *df* определяет направление поэлементной обработки в этих операциях: от начала строки к концу (*df* = 0) либо наоборот, от конца строки к ее началу (*df* = 1).

Системные флаги

Управляют вводом/выводом, маскируемыми прерываниями, отладкой, переключением между задачами и виртуальным режимом 8086.

Мнемоника флага	Флаг	Номер бита в *eflags*	Содержание и назначение
tf	Флаг трассировки **(Trace Flag)**	8	Предназначен для организации пошаговой работы микропроцессора. 1 — микропроцессор генерирует прерывание с номером 1 после выполнения каждой машинной команды. Может использоваться при отладке программ, в частности отладчиками; 0 — обычная работа
if	Флаг прерывания **(Interrupt enable Flag)**	9	Предназначен для разрешения или запрещения (маскирования) аппаратных прерываний (прерываний по входу INTR). 1 — аппаратные прерывания разрешены; 0 — аппаратные прерывания запрещены

Регистр-*указатель команд*

eip/ip (Instraction Pointer register). Имеет разрядность 32/16 бит и содержит смещение следующей подлежащей выполнению команды относительно содержимого сегментного регистра cs в текущем сегменте команд. Этот регистр непосредственно недоступен программисту, но загрузка и изменение его значения производятся различными командами управления, к которым относятся команды условных и безусловных переходов, вызова процедур и возврата из процедур. Возникновение прерываний также приводит к модификации регистра *eip/ip*.

Команды, устанавливающие значения флагов

Эти команды занимают один байт и выполняются два такта.

Команда	Назначение	Псевдокод	Описание
CLC Clear Carry Flag	Сброс флага переноса	CF := 0-	Обнуляет бит флага переноса
CMC Comlement Carry Flag	Инвертирование флага переноса	CF := NOT CF	Изменяет значение бита флага переноса на обратное
STC Set Carry Flag	Установка флаг переноса	CF := 1	Устанавливает бит флага переноса в 1
CLD Clear Direction Flag	Сброс флага направления	DF := 0	Обнуляет бит флага направления. Обработка цепочек будет производится от начала к концу.
STD Set Direction Flag	Установка флага направления	DF := 1	Устанавливает бит флага направления в 1. Обработка цепочек будет производиться от конца к началу.
CLI Clear Interrupt Flag	Сброс флага разрешения прерываний	IF := 0	Запрещает внешние прерывания по входу INTR процессора
STI Set Interrupt Flag	Установка флага разрешения прерываний	IF := 1	Разрешает внешние прерывания по входу INTR процессора

Примечание. Процессор воспринимает сигнал прерывания не раньше, чем в конце выполнения следующей команды. Поэтому между командами
STI
CLI
Должна быть, по крайней мере, одна команда. Например, пустая операция NOP (No operation).