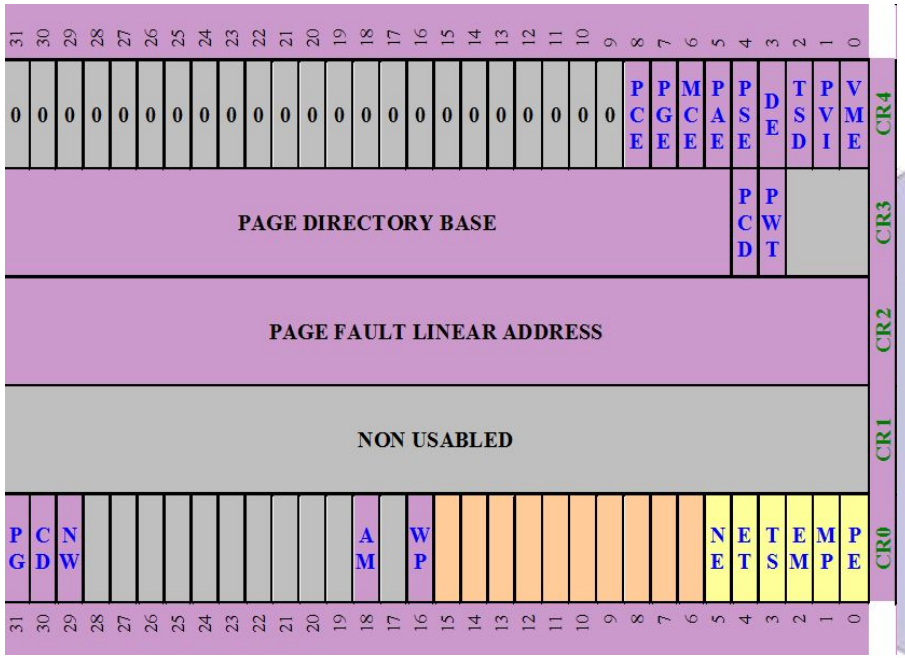
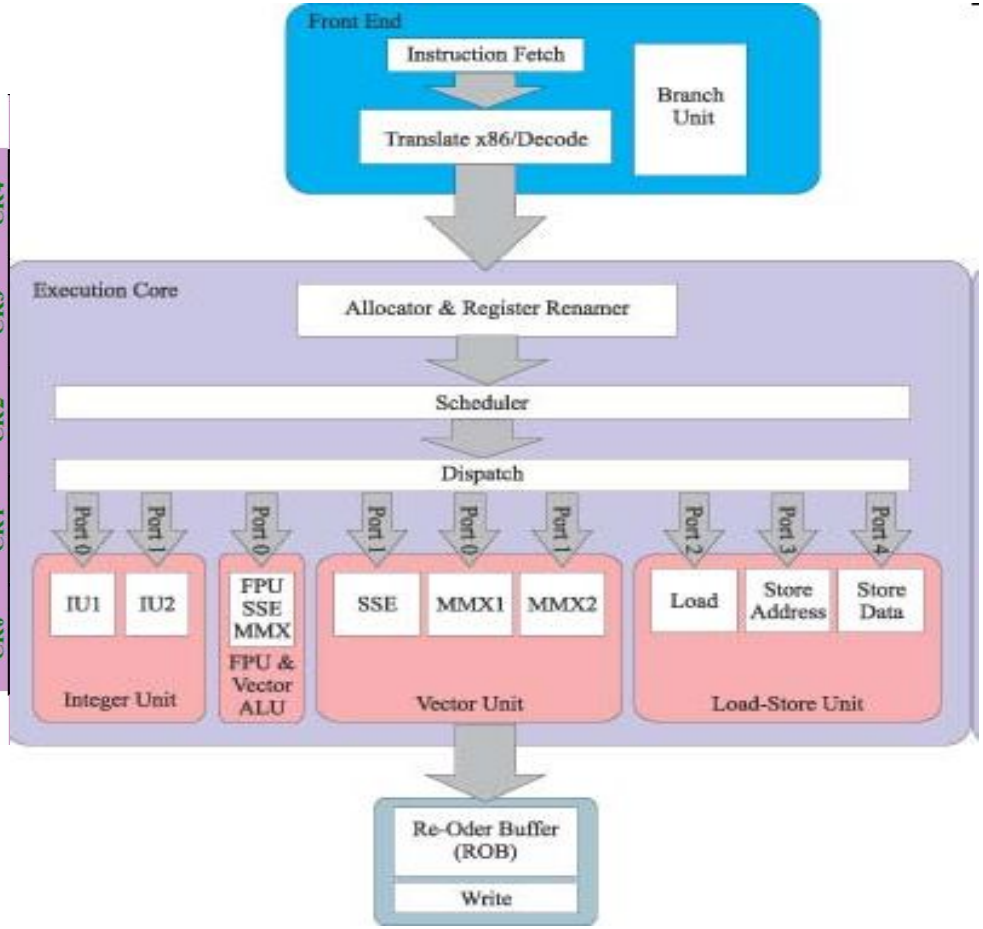
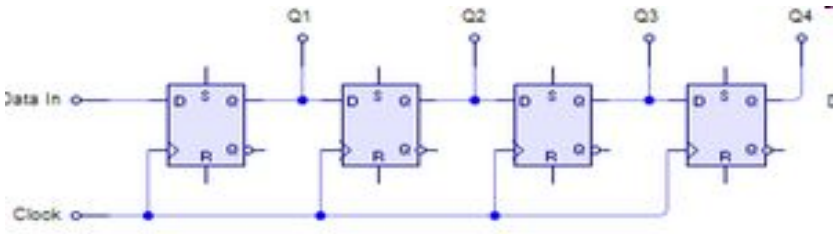
Lection
№9

Программная модель процессора: Регистры процессора

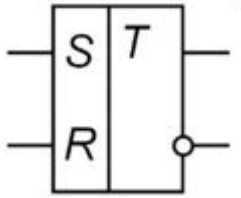


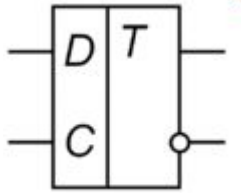


Регистр — последовательное логическое устройство, используемое для хранения n-разрядных двоичных слов (чисел) и выполнения преобразований над ними.



Основой построения регистров являются D-триггеры, RS-триггеры.



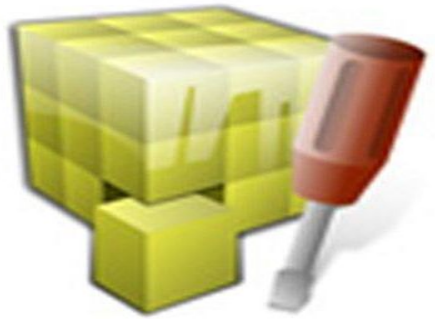


> Регистр представляет собой упорядоченную последовательность триггеров, число которых соответствует числу разрядов в слове.

> С каждым регистром обычно связано комбинационное цифровое устройство, с помощью которого обеспечивается выполнение некоторых операций над словами.

Фактически любое цифровое устройство можно представить в виде совокупности регистров, соединённых друг с другом при помощи комбинационных цифровых устройств.

Регистр процессора — сверхбыстрая память внутри процессора, предназначенная прежде всего для хранения промежуточных результатов вычисления (регистр общего назначения/регистр данных) или содержащая данные, необходимые для работы процессора — смещения базовых таблиц, уровни доступа и т. д. (специальные регистры).



РОН

	15–8	7–0
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

Сегментные регистры

15–0
CS
DS
SS
ES

Индексные регистры

15–0
SI
DI

Регистр флагов

15–0
FLAGS

Регистры-указатели

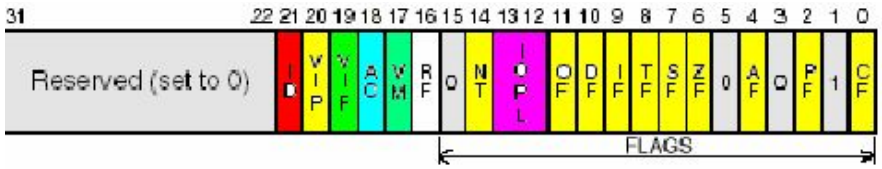
15–0
BP
SP

Указатель команд

15–0
IP

Доступ к значениям, хранящимся в регистрах, как правило, в несколько раз быстрее, чем доступ к ячейкам оперативной памяти (даже если кэш-память содержит нужные данные), но объем оперативной памяти намного превосходит суммарный объем регистров

Операции в регистрах:



- приём слова в регистр;
- передача слова из регистра;
- поразрядные логические операции;
- сдвиг слова влево или вправо на заданное число разрядов;
- преобразование последовательного кода слова в параллельный и обратно;
- установка регистра в начальное состояние (сброс).

```
;В программном сегменте:
mov DX,AX ;Из регистра в регистр
mov AL,memb ;Из памяти в регистр
mov AX,0B800h ;Непосредственное значение в регистр
mov ES,AX ;Из регистра в сегментный регистр
mov word ptr memd+2,ES ;Из сегментного регистра в память
mov word ptr memd, 2000;Непосредственное значение в память
mov BX,word ptr memb ;Слово из памяти в регистр (число 0605)
mov DI,word ptr memd ;Слово из памяти в регистр
mov ES,word ptr memd+2;Слово из памяти в сегментный регистр
```

Классификация регистров

накопительные (регистры памяти, хранения)

сдвигающие

Сдвигающие регистры

по способу ввода-вывода информации

по направлению передачи информации:

по основанию системы счисления

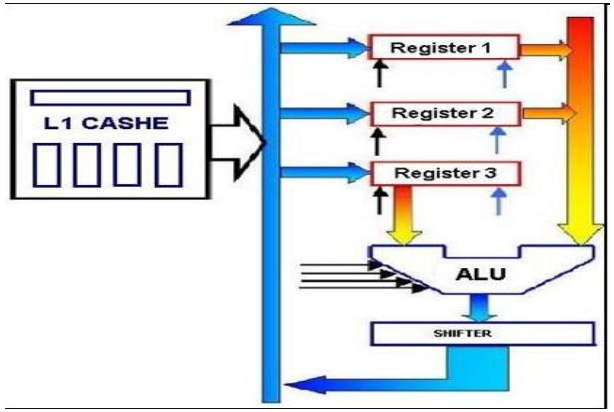
> параллельные - запись и считывание информации происходит одновременно на все входы и со всех выходов;

последовательные - запись и считывание информации происходит в первый триггер, а та информация, которая была в этом триггере, перезаписывается в следующий то же самое происходит и с остальными триггерами;

>комбинированные

> однонаправленные;

> реверсивные



двоичные;

Троичные;

Десятичные

Сегментные регистры — Регистры указывающие на сегменты: CS,DS,SS,ES,FS,GS

В реальном режиме работы процессора сегментные регистры содержат адрес начала 64Kb сегмента, смещенный вправо на 4 бита.

В защищенном режиме работы процессора сегментные регистры содержат селектор сегмента памяти, выделенного ОС.

CS — указатель на кодовый сегмент. Связка CS:IP (CS:EIP/CS:RIP - в защищенном/64-битном режиме) указывает на адрес в памяти следующей команды.

Процессоры x86 имеют регистры, подразделяющиеся на следующие категории:

- регистры общего назначения;
- указатель инструкций;
- регистр флагов;
- регистры сегментов;
- системные адресные регистры;
- управляющие регистры;
- регистры отладки;
- регистры тестирования;
- модельно-специфические (зависящие от конкретной модели процессора) регистры.

IP — 16-битный (младшая часть EIP)

EIP — 32-битный аналог (младшая часть RIP)

RIP — 64-битный аналог

Счётчик команд

IP (англ. Instruction Pointer) — регистр, содержащий адрес-смещение следующей команды, подлежащей исполнению, относительно кодового сегмента CS в процессорах семейства x86.

Регистр IP связан с CS в виде CS:IP, где CS является текущим кодовым сегментом, а IP — текущим смещением относительно этого сегмента.

Регистр IP является 16-разрядным регистром-указателем. Кроме него, в состав регистров этого типа входят SP (англ. Stack Pointer — указатель стека) и BP (англ. Base Pointer — базовый указатель).

Принцип работы

Например, CS содержит значение $2CB50_H$, в регистре IP хранится смещение 123_H.

Адрес следующей инструкции, подлежащей исполнению, вычисляется путем суммирования адреса в CS (сегменте кода) со смещением в регистре IP:
$2CB50_H + 123_H = 2CC73_H$

Таким образом, адрес следующей инструкции для исполнения равен $2CC73_H$.

При выполнении текущей инструкции процессор автоматически изменяет значение в регистре IP, в результате чего регистровая пара CS:IP всегда указывает на следующую подлежащую исполнению инструкцию.

Регистры данных — служат для хранения промежуточных вычислений.

RAX, RCX, RDX, RBX, RSP, RBP, RSI, RDI, R8 — R15 — 64-битные

EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI, R8D — R15D — 32-битные (extended AX)

AX, CX, DX, BX, SP, BP, SI, DI, R8W — R15W — 16-битные

AH, AL, CH, CL, DH, DL, BH, BL, SPL, BPL, SIL, DIL, R8B — R15B — 8-битные (половинки 16-ти битных регистров)

например, AH — high AX — старшая половинка 8 бит

AL — low AX — младшая половинка 8 бит

регистр AX — умножение, деление, обмен с устройствами ввода/вывода (команды ввода и вывода);
регистр BX — базовый регистр в вычислениях адреса;
регистр CX — счетчик циклов;
регистр DX — определение адреса ввода/вывода.

Регистры данных

AH	AL	Аккумулятор
BH	BL	Базовый регистр
CH	CL	Счетчик
DH	DL	Регистр данных

Регистры указатели

SI	Индекс источника
DI	Индекс приемника
BP	Указатель базы
SP	указаель стека

Сегментные регистры

CS	Регистр сегмента команд
DS	Регистр сегмента данных
ES	Регистр дополнительного сегмента данных
SS	Регистр сегмента стека

Прочие регистры

IP	Указатель команд
FLAGS	Регстр форматов

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

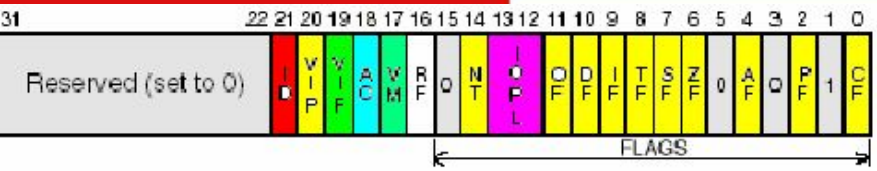
используются для работы со стеком. BP (Base Pointer) позволяет работать с переменными в стеке. Его также можно использовать в других целях. SP (Stack Pointer) указывает на вершину стека. Он используется командами, которые работают со стеком.

Сегментные регистры

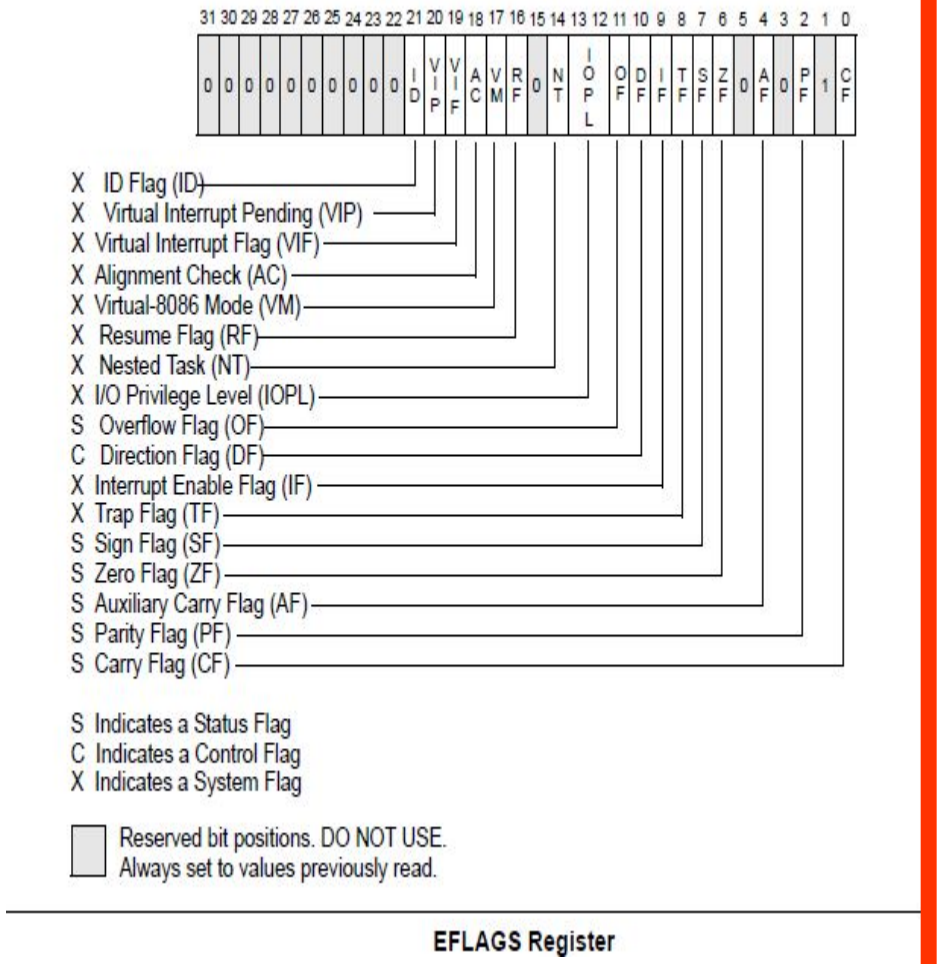
CS (Code Segment), DS (Data Segment), SS (Stack Segment) и ES (Enhanced Segment) предназначены для обеспечения сегментной адресации. Код находится в сегменте кода, данные — в сегменте данных, стек — в сегменте стека и есть еще дополнительный сегмент данных. Реальный физический адрес получается путём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

ПРИМЕНЯЮТСЯ

- в строковых операциях
- при работе со стеком
- для обращения сегменту памяти



Регистр флагов



EFLAGS Register

Значение некоторых флагов в регистре флагов можно изменять напрямую, с помощью специальных инструкций (например, CLD для сброса флага направления),

Но! нет инструкций, которые позволяют обратится (проверить или изменить) к регистру флагов как к обычному регистру.

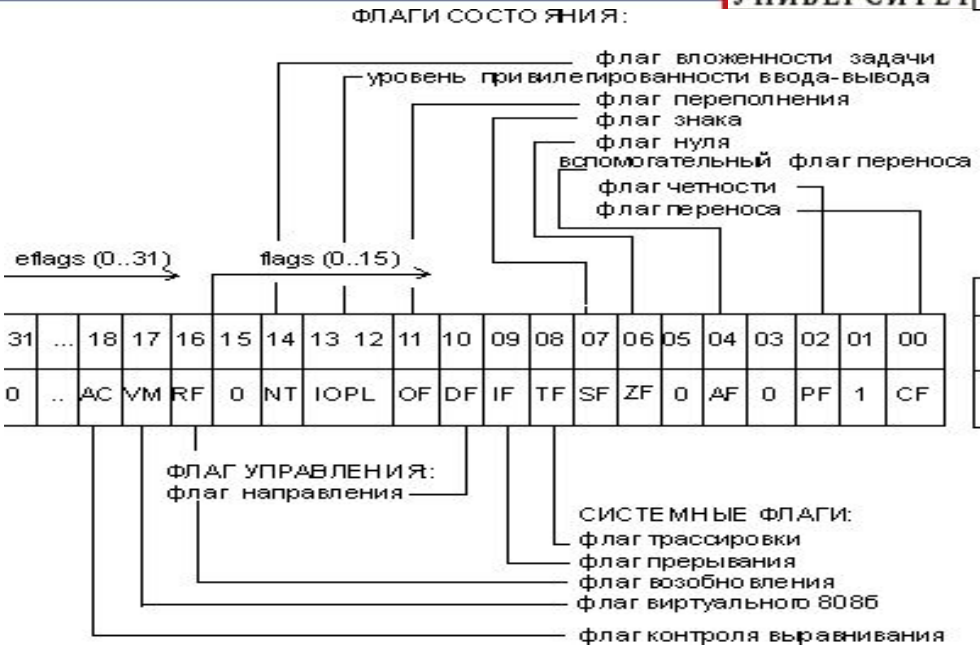
Однако можно сохранять регистр флагов в стек или регистр (R)(E)AX и восстанавливать регистр флагов из них с помощью инструкций LAHF, SAHF, PUSHF, PUSHFD, POPF и POPFD.

При приостановке задачи (используя многозадачные возможности процессора), процессор автоматически сохраняет значение флага регистров в TSS (task state segment), при активизации новой задачи процессор загружает регистр флагов из TSS новой задачи.

Флаги состояния

Флаги состояния (биты 0, 2, 4, 6, 7 и 11) отражают результат выполнения арифметических инструкций, таких как ADD, SUB, MUL, DIV.

- CF — устанавливается при переносе из/заёме в (при вычитании) старший значащий бит результата и показывает наличие переполнения в беззнаковой целочисленной арифметике. Также используется в длинной арифметике. только флаг CF можно изменять напрямую с помощью инструкций STC, CLC и CMC. Также, битовые инструкции (BT, BTS, BTR и BTC) копируют указанный бит во флаг CF.
- PF — устанавливается, если младший значащий байт результата содержит чётное число единичных (ненулевых) битов. Изначально этот флаг был ориентирован на использование в коммуникационных программах.
- AF — устанавливается при переносе из/заёме из бита 3 результата. Этот флаг ориентирован на использование в двоично-десятичной (binary coded decimal, BCD) арифметике.



- ZF — устанавливается, если результат равен нулю.
- SF — равен значению старшего значащего бита результата, который является знаковым битом в знаковой арифметике.
- OF — устанавливается, если целочисленный результат слишком длинный для размещения в целевом операнде (регистре или ячейке памяти). Этот флаг показывает наличие переполнения в знаковой целочисленной арифметике (в дополнительном коде).

Флаги состояния

позволяют одной и той же арифметической инструкции выдавать результат трёх различных типов:

Беззнаковое

Если результат считать беззнаковым числом, то флаг CF показывает условие переполнения (перенос или заём)

Двоично-десятично кодированное (BCD) целое число

для BCD-результата перенос/заём показывает флаг AF

Знаковое

для знакового результата (в дополнительном коде) перенос или заём показывает флаг OF

Флаг SF отражает знак знакового результата, флаг ZF отражает и беззнаковый, и знаковый нулевой результат.

Условные переходы, обусловленные Флагами состояния

JA	Jump if above	CF=0 and ZF=0
JAE	Jump if above or equal	CF=0
JB	Jump if below	CF=1
JBE	Jump if below or equal	CF=1 or ZF=1
JC	Jump if carry	CF=1
JCXZ	Jump if CX register is 0	
JECXZ	Jump if ECX register is 0	
JE	Jump if equal	ZF=1
JG	Jump if greater	ZF=0 and SF=OF
JGE	Jump if greater or equal	SF=OF
JL	Jump if less	SF<>OF
JLE	Jump if less or equal	ZF=1 or SF<>OF
JNA	Jump if not above	CF=1 or ZF=1
JNAE	Jump if not above or equal	CF=1
JNB	Jump if not below	CF=0
JNBE	Jump if not below or equal	CF=0 and ZF=0
JNC	Jump if not carry	CF=0
JNE	Jump if not equal	ZF=0
JNG	Jump if not greater	ZF=1 or SF<>OF
JNGE	Jump if not greater or equal	SF<>OF
JNL	Jump if not less	SF=OF
JNLE	Jump if not less or equal	ZF=0 and SF=OF
JNO	Jump if not overflow	OF=0
JNP	Jump if not parity	PF=0
JNS	Jump if not sign	SF=0
JNZ	Jump if not zero	ZF=0
JO	Jump if overflow	OF=1
JP	Jump if parity	PF=1
JPE	Jump if parity even	PF=1
JPO	Jump if parity odd	PF=0
JS	Jump if sign	SF=1
JZ	Jump if zero	ZF=1

Абстрактный пример использования регистров флагов состояния

```
cmp AX,BX        ;Сравнение двух регистров
je equal         ;Переход, если AX=BX
cmp SI,mem       ;Сравнение регистра и ячейки памяти
jne notequ       ;Переход, если SI<>mem
int 21h          ;Вызов DOS
jc syserr        ;Переход, если была ошибка и флаг CF=1
or BX,BX         ;Анализ BX
jz zero          ;Переход, если BX=0 по флагу ZF
inpt: in  AL,DX  ;Ввод данного из устройства
test AL,80h      ;Анализ бита 7 в данном
je inpt          ;Ввод до тех пор , пока
                 ;бит 7=0 (ожидание установки бита 7)
test AX,7        ;Анализ битов 0,1,2 в AX
jne found        ;Переход, если хотя бы 1 бит
                 ;из них установлен
test DI,OFh      ;Анализ битов 0...3 в DI
jz reset         ;Переход, если все они сброшены
```

В длинной целочисленной арифметике флаг CF используется совместно с инструкциями сложения с переносом (ADC) и вычитания с заёмом (SBB) для распространения переноса или заёма из одного вычисляемого разряда длинного числа в другой.

Инструкции условного перехода Jcc (переход по условию cc — например, JNZ для перехода, если результат не ноль), SETcc (установить значение байта-результата в зависимости от условия cc), LOOPcc (организация цикла) и CMOVcc (условное копирование) используют один или несколько флагов состояния для проверки условия. Например, инструкция перехода JLE (jump if less or equal — переход, если «меньше или равен», ≤) проверяет условие «ZF=1 или SF ≠ OF».

Флаг PF был введён для совместимости с другими микропроцессорными архитектурами и по прямому назначению используется редко.

- Более распространено его использование совместно с остальными флагами состояния в арифметике FPU.

Управляющий флаг

Флаг направления (DF, бит 10 в регистре флагов) управляет строковыми инструкциями (MOVS, CMPS, SCAS, LODS и STOS): установка флага заставляет уменьшать адреса (обрабатывать строки от старших адресов к младшим), обнуление заставляет адреса увеличивать. Инструкции STD и CLD соответственно устанавливают и обнуляют флаг DF.

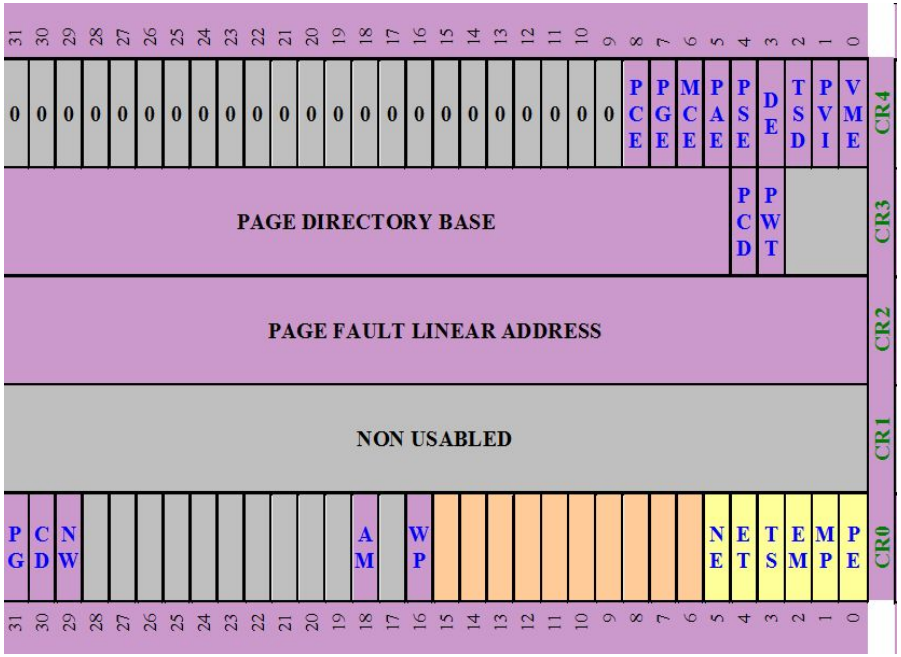
Системные флаги и поле IOPL

Системные флаги и поле IOPL управляют операционной средой и не предназначены для использования в прикладных программах.

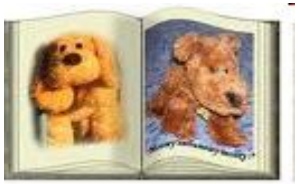
- IF — обнуление этого флага запрещает отвечать на маскируемые запросы на прерывание.
- TF — установка этого флага разрешает пошаговый режим отладки, когда после каждой выполненной инструкции происходит прерывание программы и вызов специального обработчика прерывания (см. также: Int3).
- IOPL — показывает уровень приоритета ввода-вывода исполняемой программы или задачи: чтобы программа или задача могла выполнять инструкции ввода-вывода или менять флаг IF, её текущий уровень приоритета (CPL) должен быть ≤ IOPL.

- NT — этот флаг устанавливается, когда текущая задача «вложена» в другую, прерванную задачу, и сегмент состояния TSS текущей задачи обеспечивает обратную связь с TSS предыдущей задачи. Флаг NT проверяется инструкцией IRET для определения типа возврата — межзадачного или внутризадачного.
- RF — флаг маскирования ошибок отладки.
- VM — установка этого флага в защищённом режиме вызывает переключение в режим виртуального 8086.
- AC — установка этого флага вместо с битом AM в регистре CR0 включает контроль выравнивания операндов при обращениях к памяти — обращение к невыравненному операнду вызывает исключительную ситуацию.
- VIF — виртуальная копия флага IF; используется совместно с флагом VIP.
- VIP — устанавливается для указания наличия отложенного прерывания; используется совместно с флагом VIF.
- ID — возможность программно изменить этот флаг в регистре флагов указывает на поддержку инструкции CPUID

Регистры управления



Используемая литература:



- *http://asmworld.ru/uchebnyj-kurs/004-registry-processora-8086/*
- http://www.intuit.ru/department/hardware/mpbasics/4/2.html
- http://ru.wikipedia.org/wiki/Регистр_процессора
- Книга «Ассемблер. Учебник для ВУЗов», авторы Михаил Гук, Виктор Юров
- Книга «Архитектура ЭВМ»,автор Мюллер