ОСНОВНЫЕ ПОНЯТИЯ

Программирование на языке низкого уровня Ассемблер

Что нужно для работы с ассемблером

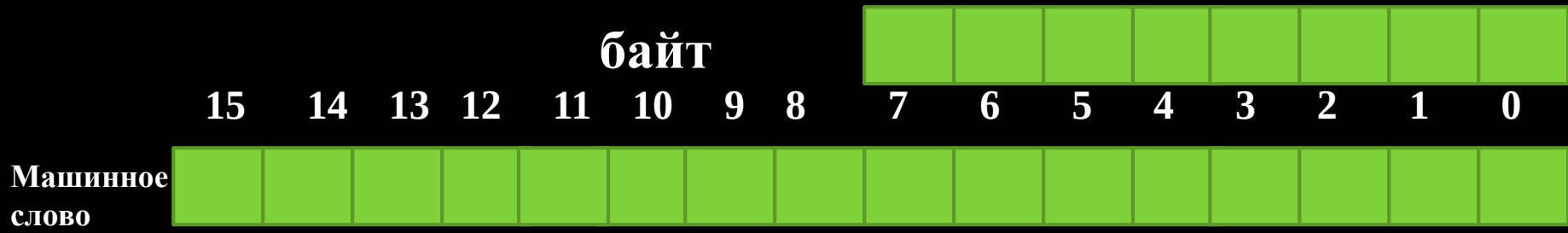
Ассемблер – это программа, которая переводит текст с языка, понятного человеку, в язык, понятный процессору, т.е. говорят, что она переводит язык ассемблера в машинный код.

Для работы с ассемблером можно использовать пакет MASM для Windows, в который входит:

- **TASM – транслятор;**
- **LINK – компоновщик;**
- **TD – отладчик.**

Представление данных в компьютере

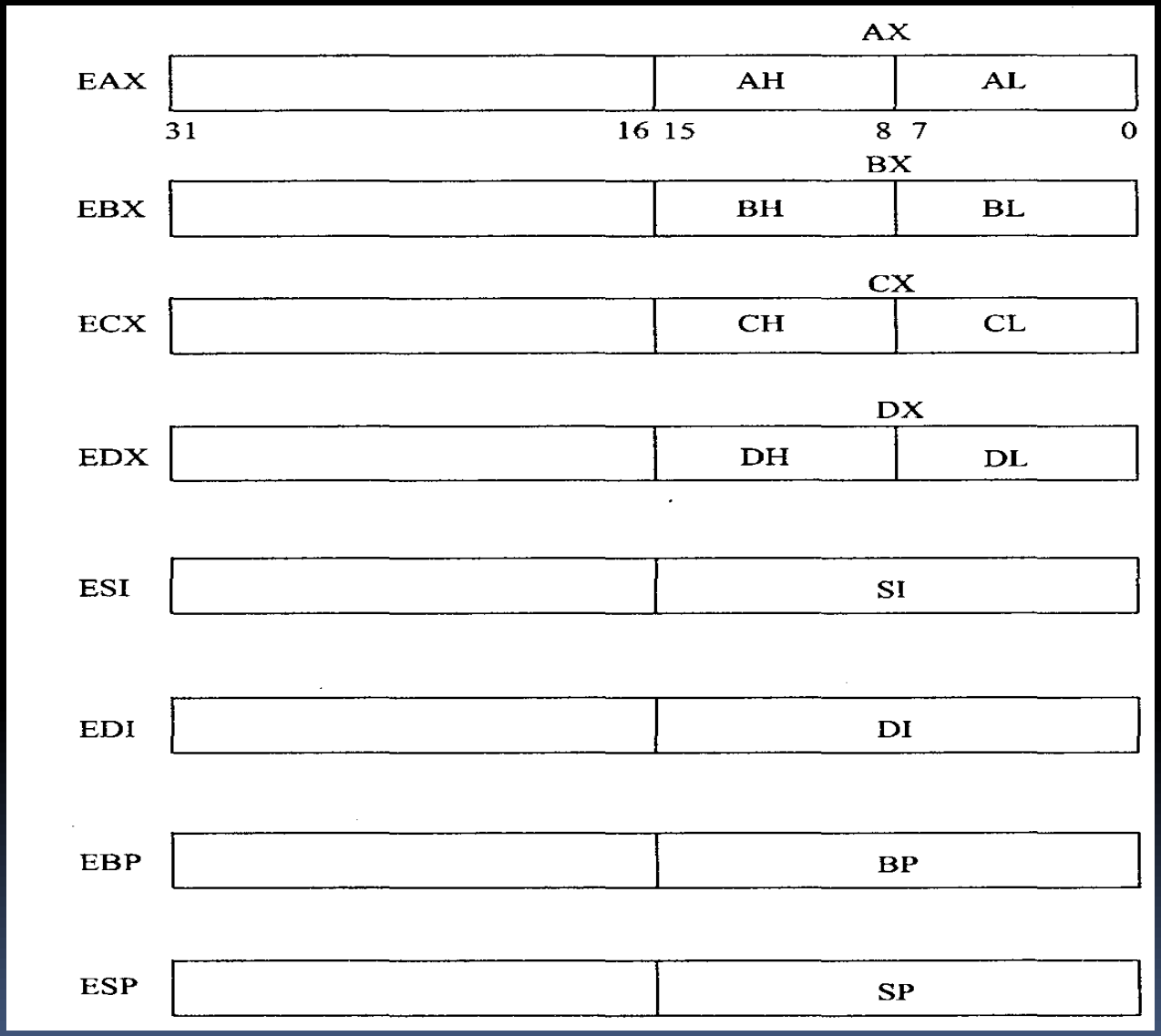
- **двоичная система счисления и шестнадцатеричная; перевод;**
- **биты, байты, слова;**



Двойное машинное слово - 32 бита (4 байта)

десятичная	двоичная	шестнадцатеричная
0	0000	00
1	0001	01
2	0010	02
3	0011	03
4	0100	04
5	0101	05
6	0110	06
7	0111	07
8	1000	08
9	1001	09
10	1010	0A
11	1011	0B
12	1100	0C
13	1101	0D
14	1110	0E
15	1111	0F
16	10000	10

Регистры общего назначения



Аккумулятор

Базовый регистр

Регистр-счетчик

Регистр данных

Индекс источника

Индекс приемника

Указатель стека

Указатель базы

Организация доступа к памяти

- Используется «Сегментированная модель» доступа к памяти.
  - Адрес начала сегмента хранится в соответствующем регистре.
  - Внутри сегмента программа обращается к адресам относительно начала сегмента линейно, т.е. начиная с 0 и заканчивая адресом, равным размеру сегмента. Этот относительный адрес, или ***смещение***, который микропроцессор использует для доступа к данным внутри сегмента, называется *эффективным*.

Организация доступа к памяти

Диапазон изменения физического адреса в реальном режиме от 0 до 1 Мбайт.

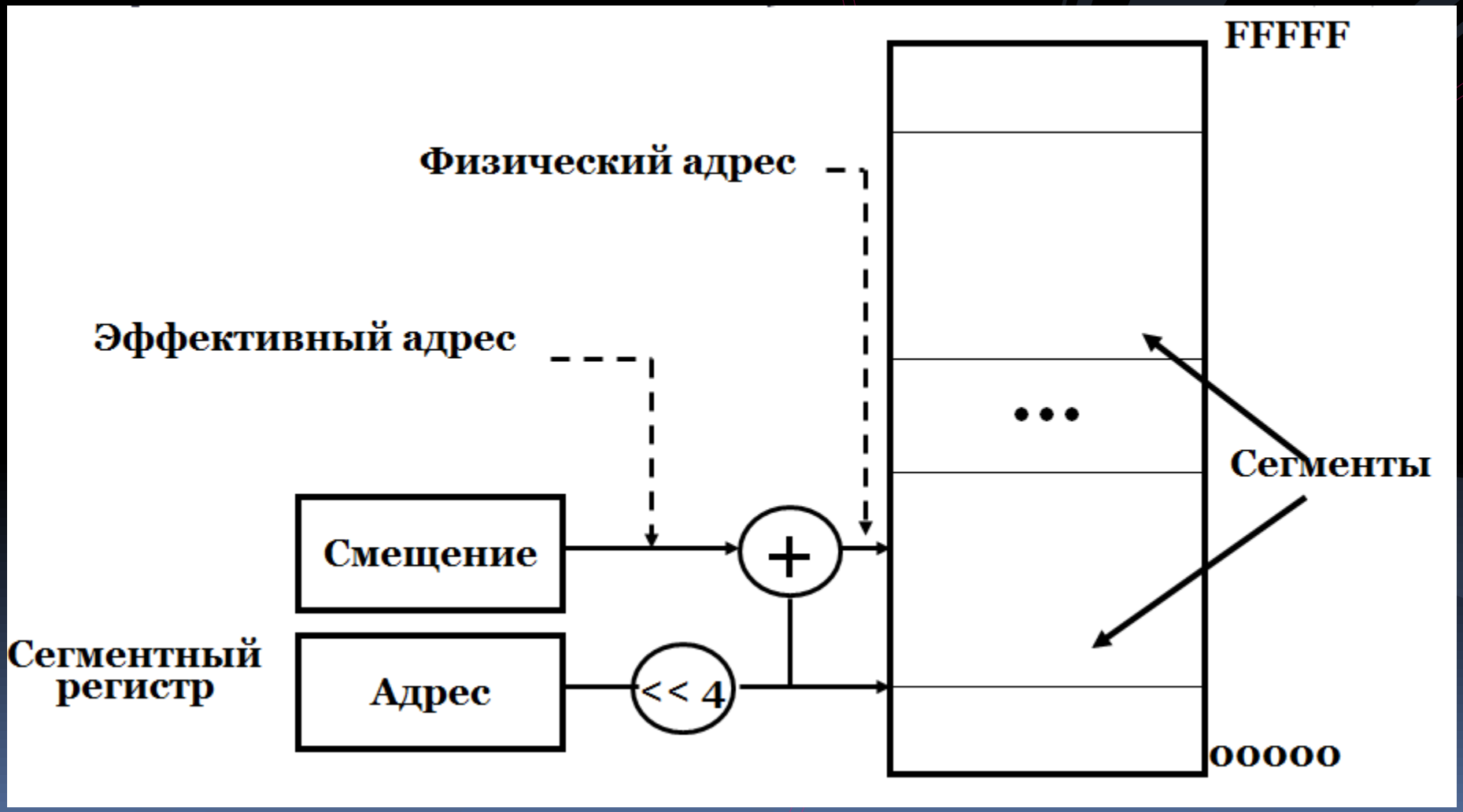
Максимальный размер сегмента
64 Кбайт. Это объясняется 16-разрядной архитектурой регистров. Максимальное значение, которое может содержать 16-ти разрядный регистр равно:
$2^{(16-1)} = 65535 = 64$ Кбайт

Организация доступа к памяти

В сегментном регистре содержаться только старшие 16 бит физического адреса начала сегмента.

Недостающие младшие 4 бита 20-битного адреса получаются сдвигом в сегментном регистре влево на 4 разряда.

Организация доступа к памяти



Сегментные регистры

В процессорах Intel предусмотрено шесть 16-битных регистров:

CS – сегмент кода;

DS – сегмент данных;

SS – сегмент стека;

ES –

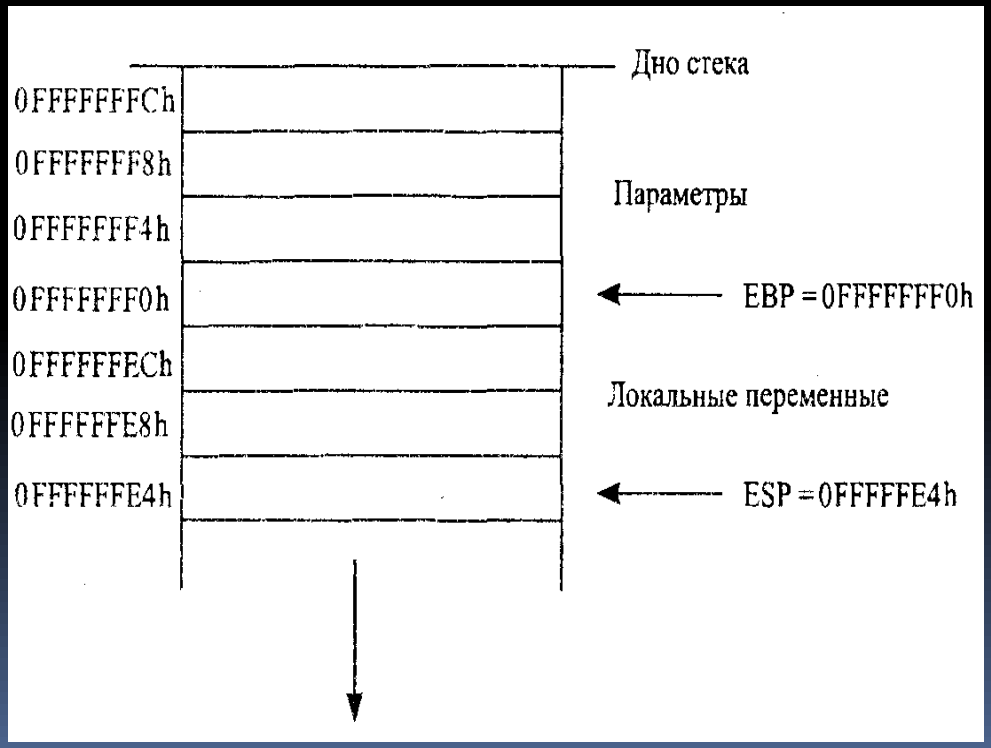
GS –

FS –

дополнительные регистры данных

Стек

Стек **- организованный специальным образом участок памяти, который используется для временного хранения переменных, передачи параметров вызываемым подпрограмм и сохранения адреса возврата при вызове процедур и прерываний. Стек располагается в сегменте памяти, описываемом регистром SS, и текущее смещение вершины стека отражено в регистре ESP, причем во время записи значение этого смещения уменьшается, т. е. он «растет вниз».**



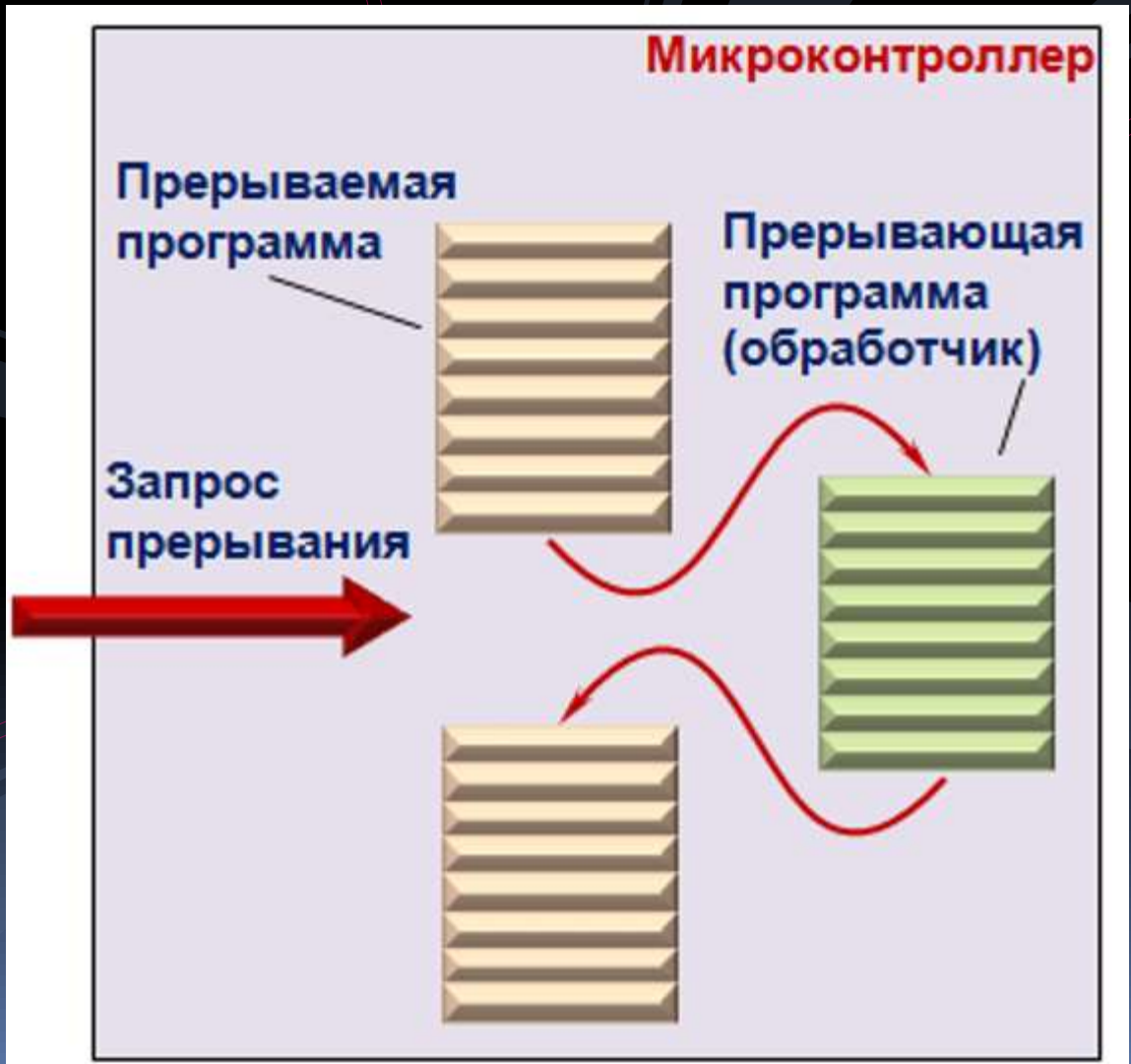
Регистр флагов

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

CF – флаг переноса;
PF – флаг четности;
AF – флаг полупереноса;
ZF – флаг нуля;
SF – флаг знака;
TF – флаг ловушки;
IF – флаг прерываний;
DF – флаг направления;
OF – флаг переполнения;

Прерывания в микроконтроллере

Прерывание (interrupt) – событие, требующие немедленной реакции со стороны процессора. … При возникновении **прерывания** **микроконтроллер** сохраняет в стеке содержимое счетчика команд и загружает в него адрес соответствующего вектора**прерывания**.



Вектор состояния программы

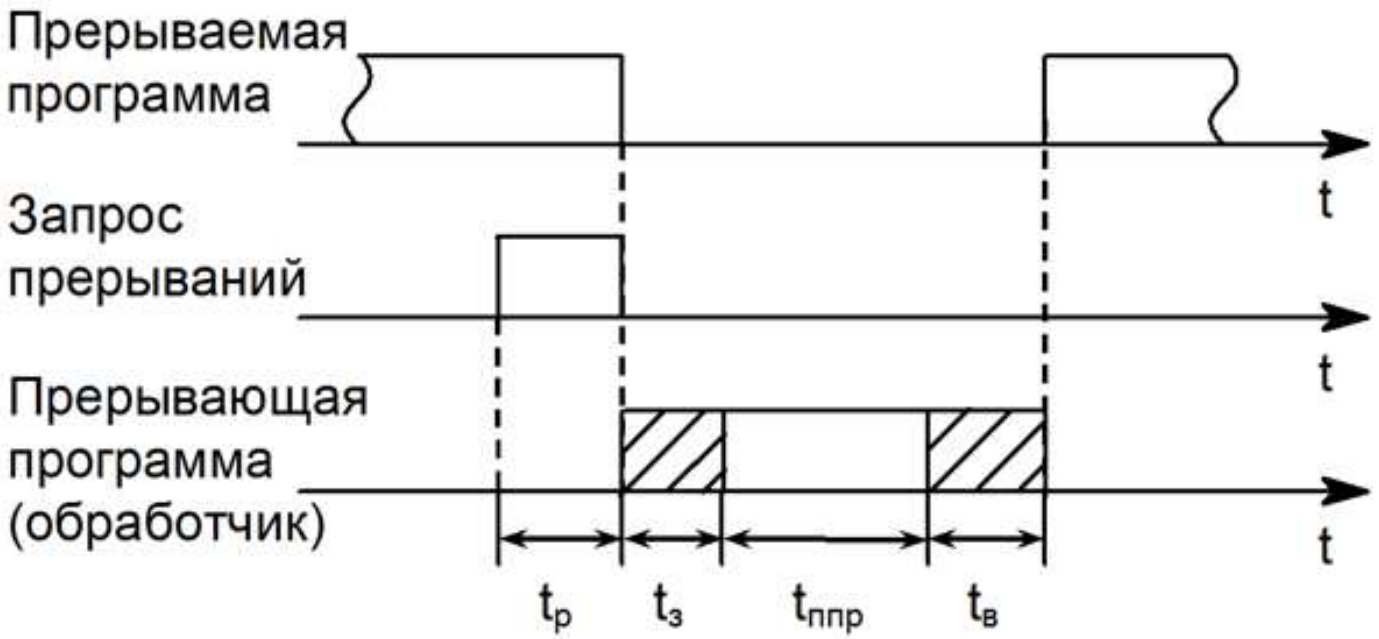
Состояние программы представляет собой совокупность состояний всех запоминающих элементов в соответствующий момент времени (например, после выполнения последней команды). При возникновении прерывания микроконтроллер сохраняет в стеке содержимое счетчика команд и загружает в него адрес соответствующего вектора прерывания. Последней командой подпрограммы обработки прерывания должна быть команда, которая осуществляет возврат в основную программу и восстановление предварительно сохраненного счетчика команд.

Вектор состояния программы это набор элементов, подвергающихся изменению при прерывании.

Вектор начального состояния содержит всю необходимую информацию для начального запуска программы. Во многих случаях вектор начального состояния содержит только один элемент – начальный адрес запускаемой программы.

Векторы прерываний

Вектор прерывания является вектором начального состояния прерывающей программы (обработчика) и содержит всю необходимую информацию для перехода к обработчику, в том числе его начальный адрес. Каждому типу прерываний соответствует свой вектор прерывания, который инициализирует выполнение соответствующего обработчика. Обычно векторы прерывания хранятся в специально выделенных фиксированных ячейках памяти с короткими адресами, представляющих собой ***таблицу векторов прерываний***. Для перехода к соответствующей прерывающей программе процессор должен располагать вектором прерывания и адресом этого вектора. По этому адресу, как правило, находится команда безусловного перехода к подпрограмме обработки прерывания.



$t_р$ – время реакции системы на прерывание;

$t_з$ – время запоминания состояния прерываемой программы;

$t_{ппр}$ – время собственно прерывающей программы;

$t_в$ – время восстановления состояния прерванной программы

Прерывания, которые генерируются самим процессором

- Деление на ноль. Генерируется при, собственно, делении на ноль.
- Отладочное исключение. Генерироваться само не может, используется для, собственно, отладки.
- Немаскируемое прерывание. Генерируется при ошибках ОЗУ и невосстановимых ошибках «железа»..
- Точка останова, используется для отладки.
- Переполнение.
- Выход за пределы.
- Недопустимый опкод. Генерируется при попытке выполнения недопустимого кода операции.
- Устройство недоступно.
- Сегмент отсутствует. Возникает при попытке загрузки сегмента с битом Present == 0.
- Ошибка сегмента стека.
- И др.