Лекция 14

Раздел 4. Структуры и записи

4.1. Записи

4.1.1. Описание типа записи

Записи с битовыми полями - это тип записей, в которых каждое поле имеет указанную в битах длину и инициализировано некоторым значением. Размер записи с битовыми полями определяется суммой размеров ее полей.

Преимуществом данных такого типа является возможность хранения информации в наиболее компактной форме. Например, при необходимости сохранения значений 16-ти флагов, каждый из которых может иметь два состояния, традиционно используется область памяти из 16-ти байтов или слов, а при использовании записей с битовыми полями потребуется всего 16 битов.

Объявление записи в режиме Ideal имеет следующий синтаксис:

имя RECORD [поле[,поле...]]

Каждое "***поле***" имеет следующий синтаксис:

имя_поля : выражение_для_размера[=значение]

Параметр "***имя_поля***" - это символьное имя поля записи.

TASM для размещения этого поля выделяет область памяти размером, полученным в результате вычисления "***выражения_для_размера***". "***Значение***" описывает данные, которыми инициализируется поле (Это значение будет присваиваться полю при каждом создании экземпляра записи такого типа). "***Значение***" и "***выражение_для_размера***" не могут быть относительными адресами.

Например:

```
REC RECORD Am:3, Bm:3 = 7
DATE RECORD Yea:7, Mon:4, Day:5
```

Имена полей записи являются глобальными идентификаторами и не могут переопределяться. Значение имени поля при использовании в выражении является счетчиком сдвига для пересылки поля в крайнее правое положение.

"***Имя***" является символьным именем данных типа запись, по которому к ним можно обращаться в тексте программы. Кроме того, эти имена можно использовать для создания переменных и размещения их в памяти.

Данные типа запись (не отдельные поля !!!) являются переопределяемыми, т.е. в тексте программы можно несколько раз объявлять данные такого типа с одним и тем же именем.

Для определения данных типа запись TASM поддерживает многострочный синтаксис.

4.1.2. Описание переменных записей

При создании экземпляра данных типа записи в качестве директивы объявления и распределения данных используется имя данных типа записи.

Синтаксис описания переменных имеет вид:

имя_переменной имя_типа <нач_знач[; нач_знач]>

где **нач_знач** – это:

- константное выражение;
- или «?»;
- или пусто.

```
R1 Rec <5,6> ; Am=5, Bm = 6
R2 Rec <,?>
VIC DATE <49,7,8> Yea=49, Mon=7, Day=8
```

Если **пусто**, то берется начальное значение из описания типа. Предусмотрены и массивы записей:

```
X DATE 100 DUP (<>) ; массив
```

Например, если в программе определен тип:

```
MyRec RECORD Val:3=4,Mode:2,Asize:4=15
MTest MyRec ?
```

то оператором MTest MyRec ? будет создан экземпляр записи типа MyRec, определяемый переменной MTest.

Экземпляр записи всегда имеет размер байта, слова или двойного слова, в зависимости от числа битов в определении записи.

4.1.3. Работа с полями записей

Если сумма значений ширины полей <= 8, то размер экземпляра записи будет 1 байт, >8 и <=16 - слово, >16 и <=32 - двойное слово. Первое описанное поле помещается в старшие значащие биты записи, следующие поля помещаются в следующие направо биты. Если описанные поля не занимают полные 8, 16 или 32 бита, полная запись сдвигается направо так, что последний бит последнего поля становится младшим битом записи. Неиспользованные биты в старшей части инициализируются нулями.

MTest

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1
							Val			Mode		Asize			

Если при инициализации экземпляра данных типа запись какие-то значения инициализации полей записи опущены, то TASM заменяет их значением 0. Наиболее простой метод инициализации экземпляра записи заключается в присвоении полям экземпляра значений, указанных в определении типа записи, например:

```
MyRec{}
```

эквивалентно

```
DW (4 SHL 6) + (0 SHL 4) + (15 SHL 0)
```

4.1.4. Значение имени поля

Пустые фигурные скобки представляют нулевое значение инициализации записи. Значения, указанные в инициализаторе, определяют, какие из значений инициализации полей будут перекрываться и с какими значениями. При этом используется синтаксис:

{[имя_поля=выражение[, имя_поля=выражение..]]}

Здесь "***имя_поля***" - имя поля в записи, "***выражение***" - значение, которое должно быть назначено указанному полю в экземпляре записи. Значение «?» эквивалентно 0. Все значение, не указанные в инициализаторе, устанавливаются TASM равным значениям, указанным в определении записи.

Например:

```
MyRec {Val=2,ASize=?}
```

эквивалентно

```
DW (2 SHL 6) + (0 SHL 4) + (0 SHL 0)
```

Другим методом инициализации экземпляра записи является применение инициализатора с **угловыми скобками** (<>). Значения, указанные в инициализаторе не имеют названий, но они должны быть заданы в том же порядке, что и соответствующие поля в определении записи. Синтаксис инициализатора такого типа:

<[выражение[, выражение..]]>

где "***выражение***" - задаваемое значение для соответствующего поля в определении записи. Пропущенное значение указывает, что будет инициализироваться значение, принятое по умолчанию (из определения записи). Ключевое слово ? указывает, что значение инициализации записи будет 0.

Например:

```
MyRec <,2,?> ; Mode=2,Asize=0
```

эквивалентно

```
DW (4 SHL 6) + (2 SHL 4) + (0 SHL 0)
```

Если в инициализаторе указано меньше значений, чем необходимо для инициализации всех полей экземпляра записи, TASM для всех остальных полей использует значения, принятые по умолчанию. Так, оператор

```
MyRec <1> ; Val=1
```

эквивалентен оператору

Если имя поля структуры имеет значение смещения поля в байтах от начала структуры, то имени поля записи ассемблер присваивает также определенное число. Оно равняется количеству битов, на которое поле нужно сдвинуть справа, чтобы оно оказалось прижатым к правому краю слова.

Встречая имя поля записи в тексте программы, ассемблер заменяет его своим значением. Это нужно для подготовки выполнения команд сдвига после выделения поля командой **AND**. В предыдущем фрагменте поле ***Day*** является крайним дело, и, чтобы проверить его содержание, не нужно его сдвигать справа.

Ясно, если необходимо проверить поле Mon, то его предварительно нужно сдвинуть справа на столько бит, сколько занимает поле Day, или на значение Mon = 5.

Следовательно, проверить, поле Mon = 7?

```
MOV AX, VIC
AND AX, MASK Mon; AX: 0M0
MOV CL, Mon; константа 5 к CL
SHR AX, CL; AX: 00M
CMP AX,7
JE Yes
No:
```

Оператор **MASK** возвращает *битовую маску поля*. Оператор **WIDTH** возвращает *длину поля* в битах.

Оператор MASK

Синтаксис

MASK имя_поля_записи

или

MASK имя_записи

Оператор MASK **имя_поля_записи** возвращает битовую маску, равную 1, для битовых позиций, занятых заданным полем записи, и равную 0 для остальных битовых позиций.

Оператор **MASK имя_записи** возвращает битовую маску, в которой биты, зарезервированные для битовых полей, установлены в 1, остальные - в 0.

Пример:

```
.DATA
COLOR    RECORD BLINK:1,BACK:3,INTENSE:1,FORE:3
MESSAGE  COLOR <0,5,1,1>
.CODE
MOV AH,MESSAGE            ;загрузка первоначального      0101 1001
AND AH,NOT MASK BACK;закрыть маску BACK     AND 1000 1111
                          ;                                0000 1001
OR  AH,MASK BLINK         ;открыть маску BLINK     OR   1000 0000
                          ;                                1000 1001
XOR AH,MASK INTENSE       ;                        XOR 0000 1000
                          ;                                1000 0001
```

Оператор WIDTH

Синтаксис

WIDTH имя_записи - возвращает общее количество битов, зарезервированное в описании записи;

WIDTH имя_поля записи - возвращает общее количество битов, зарезервированное для поля в описании записи;

Кроме того, как уже упоминалось, имя поля при использовании в выражении является счетчиком сдвига для пересылки поля в крайнее правое положение.

Пример

```
.DATA
COLOR    RECORD BLINK:1,BACK:3,INTENSE:1,FORE:3
WBLINK   EQU WIDTH BLINK; "WBLINK"=1 "BLINK"=7
;(показать от куда)
WBACK    EQU WIDTH BACK ; "WBACK"=3 "BACK"=4
WINTENSE EQU WIDTH INTENSE; "WINTENSE"=1 "INTENST"=3
WFORE    EQU WIDTH FORE         ; "WFORE"=3  "FORE"=0
WCOLOR   EQU WIDTH COLOR     ; "WCOLOR"=8
PROMPT   COLOR <1,5,1,1>
.CODE
.....
IF (WIDTH COLOR) GE 8      ;если COLOR=16 бит, загрузить
MOV AX,PROMPT              ;в 16-битовый регистр
ELSE                       ;иначе
MOV AL,PROMPT               ;загрузить в 8-битовый регистр
XOR AH,AH                  ;очистить старший 8-битовый
END IF                     ;регистр
```

Еще один пример использования

Пример:

```
.DATA
COLOR    RECORD BLINK:1,BACK:3,INTENSE:1,FORE:3
CURSOR   COLOR <1,5,1,1>
.CODE
;УВЕЛИЧИТЬ НА 1 "BACK" В "CURSOR" БЕЗ ИЗМЕНИЯ ДРУГИХ
;ПОЛЕЙ
MOV AL,CURSOR       ;загрузить значение из памяти
MOV AH,AL           ;создать рабочую копию   1101 1001   AH/AL
AND AL,NOT MASK BACK ;                  AND  1000 1111   MASK
                    ;                       1000 1001
;замаскировать старшие биты для запоминания старого CURSOR
MOV CL,BACK       ;загрузить позицию бита
SHR AH,CL         ;сдвинуть вправо          0000 0101    AH
INC AH            ;увеличить на 1           0000 0110    AH
SHL AH,CL         ;сдвинуть назад влево     0110 0000    AH
AND AH,MASK BACK  ;                  AND    0111 0000 MASK
                  ;                         0110 0000    AH
OR AH,AL          ;                   OR    1000 1001    AL
                  ;                         1110 1001    AH
MOV CURSOR,AH
```

Дополнительные возможности обработки

Понимая важность типа данных «запись» для эффективного программирования, разработчики транслятора TASM, начиная с версии 3.0, включили в систему его команд две дополнительные команды на правах директив. Последнее означает, что эти команды внешне имеют формат обычных команд ассемблера, но после трансляции они приводятся к одной или нескольким машинным командам. Введение этих команд в язык TASM повышает наглядность работы с записями, оптимизирует код и уменьшает размер программы. Эти команды позволяют скрыть от программиста действия по выделению и установке отдельных полей записи.

Для установки значения и выборки значения некоторого поля записи используются команда **SETFIELD** и **GETFIELD**

SETFIELD имя_элемента_записи приемник, регистр_источник – устанавливает по смещению **имя_элемента_записи** в **приемник** значение, которое равно **регистр_источник.**

Работа команды **SETFIELD** заключается в следующем. Местоположение записи определяется операндом приемник, который может представлять собой имя регистра или адрес памяти. Операнд **имя_элемента_записи** определяет элемент записи, с которым ведется работа (по сути, если вы были внимательны, он определяет смещение элемента в записи относительно младшего разряда). Новое значение, в которое необходимо установить указанный элемент записи, должно содержаться в операнде **регистр_источник**. Обрабатывая данную команду, транслятор генерирует последовательность команд, которые выполняют следующие действия.

1. Сдвиг содержимого операнда **регистр_источник** влево на количество разрядов, соответствующее расположению **элемента_ записи**;
2. Выполнение логической операции **OR** над операндами **приемник** и **регистр_источник**. Результат операции помещается в операнд **приемник**.

Важно отметить, что **SETFIELD** не производит предварительной очистки элемента, в результате после логического сложения командой **OR** возможно наложение старого содержимого элемента и нового устанавливаемого значения. Поэтому требуется предварительно подготовить поле в записи путем его обнуления.

GETFIELD имя_элемента_записи регистр_приемник, источник - извлекает значение по смещению **имя_элемента_записи**, обнаруженное в **источнике** или по **адресу памяти**, и устанавливает в это значение в соответствующую по смещению **имя_элемента_записи** часть **регистра_приемника**.

Действие команды **GETFIELD** обратно действию **SETFIELD**. В качестве операнда **источник** может быть указан либо регистр, либо адрес памяти. В регистр, указанный операндом **регистр_приемник**, помещается результат работы команды — значение элемента записи. Интересная особенность связана с операндом **регистр_приемник**. Команда **GETFIELD** всегда использует 16-разрядный регистр, даже если вы укажете в этой команде имя 8-разрядного регистра.

Обрабатывая данную команду, транслятор генерирует последовательность команд, которые выполняют следующие действия.

1. Пересылка значания **источник** в **имя_элемента_записи регистр_приемник**

2. Выполнение логической операции **AND** над операндами **регистр_приемник** и **маской** (еденичные биты по смещению **имя_элемента_записи**). Результат операции помещается в операнд **регистр_приемник**.

В качестве примера применения команд **SETFIELD** и **GETFIELD** рассмотрим пример.

```
masm
model small
stack 256
iotest record i1:1,i2:2=11,i3:1,i4:2=11,i5:2=00
.data
flag   iotest <>
.code
main:
        mov    ax,@data
        mov    ds,ax
        mov    al,flag  ;al=108=06ch=01101100
        mov    bl,3
        setfield i5 al,bl ;al=111=06fh=01101111
        mov    al,110
        xor    bl,bl
        getfield        i5 bl,al ;i5=00  bl=10=2h=2  al=06eh=01101110
        mov    bl,1
        setfield i4 al,bl ;al=110=06eh=01101110 bl=100=4h=4
        setfield i5 al,bl ;al=108=06eh=01101110 bl=100=4h=4
exit:
        mov    ax,4c00h        ; стандартный выход
        int    21h
end main                       ;конец программы
```

4.3. Команды прерываний

Их выполнения имеет много общего с командой вызова процедуры. Здесь так же происходит переход к группе команд, которая начинается с определенного адреса, и называется программой обработки прерывания.

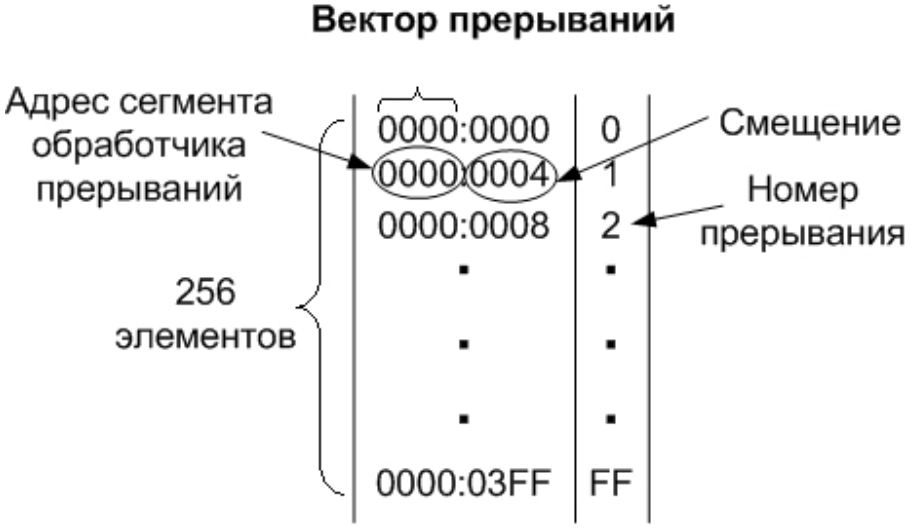
Если при вызове процедуры можно использовать разные режимы адресации, чтобы определить адрес процедуры, то переход к обработке прерывания есть непрямым и осуществляется через вектор прерывания. Вектор прерывания занимает два слова и хранит значение регистра кода **CS** и указателя команд **IP**, с которых начинается соответствующая программа обработки прерывания. Всего может быть **256** прерываний, каждое характеризуется своим номером, например, 21H. Адрес вектора прерывания определяется как номер, умноженный на 4: 4*21H.

Где находятся векторы прерывания в памяти?

Они находятся на основной системной плате - памяти RAM 64Кбайт. Самый первый 1Кбайт занимают векторы прерываний.

Главной исполнительной программой ЭВМ является базовая система введения-выведения (Basic Input/Output System) BIOS.

Она запоминает символы, которые набирают на клавиатуре, отображает символы на экране, осуществляет обмен данными между устройствами ЭВМ: дисководами, принтером и другими.



Таблица

Назначение некоторых наиболее важных векторов

Номер	Описание
0	**Ошибка деления.** Вызывается автоматически после выполнения команд DIV или IDIV, если в результате деления происходит переполнение (например, при делении на 0). DOS обычно при обработке этого прерывания выводит сообщение об ошибке и останавливает выполнение программы. Для процессора 8086 при этом адрес возврата указывает на следующую после команды деления команду, а в процессоре 80286 - на первый байт команды, вызвавшей прерывание
1	**Прерывание пошагового режима.** Вырабатывается после выполнения каждой машинной команды, если в слове флагов установлен бит пошаговой трассировки TF. Используется для отладки программ. Это прерывание не

	вырабатывается после выполнения команды MOV в сегментные регистры или после загрузки сегментных регистров командой POP
2	**Аппаратное немаскируемое прерывание.** Это прерывание может использоваться по-разному в разных машинах. Обычно вырабатывается при ошибке четности в оперативной памяти и при запросе прерывания от сопроцессора
3	**Прерывание для трассировки.** Это прерывание генерируется при выполнении однобайтовой машинной команды с кодом CCh и обычно используется отладчиками для установки точки прерывания
4	**Переполнение.** Генерируется машинной командой INTO, если установлен флаг OF. Если флаг не установлен, то команда INTO выполняется как NOP. Это прерывание используется для обработки ошибок при выполнении арифметических операций
5	**Печать копии экрана.** Генерируется при нажатии на клавиатуре клавиши PrtScr. Обычно используется для печати образа экрана. Для процессора 80286 генерируется при выполнении машинной команды BOUND, если проверяемое значение вышло за пределы заданного диапазона
6	Неопределенный код операции или длина команды больше 10 байт (для процессора 80286)
7	Особый случай отсутствия математического сопроцессора (процессор 80286)
8	***IRQ0*** - прерывание интервального **таймера**, возникает 18,2 раза в секунду
9	***IRQ1*** - прерывание от **клавиатуры**
A	***IRQ2*** - используется для каскадирования аппаратных прерываний
B	***IRQ3*** - прерывание асинхронного порта **COM2**
C	***IRQ4*** - прерывание асинхронного порта **COM1**
D	***IRQ5*** - прерывание от контроллера жесткого диска для XT
E	***IRQ6*** - прерывание генерируется контроллером флоппи-диска
F	***IRQ7*** - прерывание **принтера**
10	Обслуживание **видеоадаптера**
11	**Определение конфигурации** устройств в системе
12	**Определение размера оперативной памяти** в системе
13	Обслуживание **дисковой** системы
14	Последовательный **ввод/вывод**
15	Расширенный сервис для AT-компьютеров
16	Обслуживание клавиатуры
17	Обслуживание принтера
18	Запуск BASIC в ПЗУ, если он есть
19	Загрузка операционной системы
1A	Обслуживание часов
1B	Обработчик прерывания Ctrl-Break
1C	Прерывание возникает 18.2 раза в секунду
1D	**Адрес видео таблицы** для контроллера видеоадаптера 6845
1E	Указатель на таблицу параметров дискеты
1F	Указатель на графическую таблицу для символов с кодами ASCII 128-255
20-5F	Используется DOS или зарезервировано для DOS
60-67	Прерывания, зарезервированные для пользователя
68-6F	Не используются

70	***IRQ8*** - прерывание от часов реального времени
71	***IRQ9*** - прерывание от контроллера EGA
72	***IRQ10*** - зарезервировано
73	***IRQ11*** - зарезервировано
74	***IRQ12*** - зарезервировано
75	***IRQ13*** - прерывание от математического сопроцессора
76	***IRQ14*** - прерывание от контроллера жесткого диска
77	***IRQ15*** – зарезервировано
78 - 7F	Не используются
80-85	Зарезервированы для BASIC
86-F0	Используются интерпретатором BASI
F1-FF	Не используются

Таблица

Резервирование групп прерываний

00000	Векторы прерываний BIOS (0H - 1FH)
00080h	Векторов прерываний DOS (20H - 3FH)
0180h	Векторов прерываний пользователя (60H - 7FH)
0100h	Зарезервировано векторы прерываний (40H - 5FH)
0400h	Области данных BIOS
0200h	Векторов прерываний Бейсика (80H - FFH)
0600h	62,5K Области данных, доступной для чтения и записи.
0500h	Области данных Бейсик и DOS

Как и вызов подпрограммы, прерывания может иметь дистанцию **NEAR** или **FAR**.

Во время выполнения программы обработки прерывания запрещаются маскируемые прерывания и трассировки программы. Поэтому флаги **IF** и **TF** обнуляются. Следовательно, программы обработки прерываний изменяют значение регистра флагов. Поэтому перед началом обработки прерывания в стек засылается регистр флагов **F**. Поэтому, в отличие от вызова подпрограмм, здесь в стеке будет три слова:

a	IP
a+2	CS
a+4	F

Существует три команды прерываний:

MOV AH, номер_функции

<параметры>

INT 21H; прерывание DOS

1. INT;
2. INTO;
3. IRET.

INT (interrupt) - команда условного прерывания

Прерывает обработку программы, передает управление в **DOS** или **BIOS** для определенного действия и затем возвращает управление в прерванную программу для продолжения обработки. Наиболее часто прерывание используется для выполнения

операций ввода или вывода. Для выхода из программы на обработку прерывания и для последующего возврата команда **INT** выполняет следующие действия:

1. уменьшает указатель стека на 2 и заносит в вершину стека содержимое флагового регистра;
2. очищает флаги **TF** и **IF**;
3. уменьшает указатель стека на 2 и заносит содержимое регистра **CS** в стек;
4. уменьшает указатель стека на 2 и заносит в стек значение командного указателя;
5. обеспечивает выполнение необходимых действий;
6. восстанавливает из стека значение регистра и возвращает управление в прерванную программу на команду, следующую после **INT**.

Этот процесс выполняется полностью автоматически. Необходимо лишь определить сегмент стека достаточно большим для записи в него значений регистров.

Рассмотрим два типа прерываний: команду **BIOS INT 10H** и команду **DOS INT 21H** для вывода на экран и ввода с клавиатуры. В последующих примерах в зависимости от требований используются как **INT 10H** так и **INT 21H**.

INTO (interrupt if overflow) - прерывание при переполнении. Приводит к прерыванию при возникновении переполнения (флаг **OF** установлен в 1) и выполняет команду **IRET 4**. Адрес подпрограммы обработки прерывания (вектор прерывания) находится по адресу 10H.

IRET (interrupt return) - команда возвращения после прерывания. Обеспечивает возврат из подпрограммы обработки прерывания. Команда **IRET** выполняет следующее:

1) помещает слово из вершины стека в регистр **IP** и увеличивает значение **SP** на 2;
2) помещает слово из вершины стека в регистр **CS** и увеличивает значение **SP** на 2;
3) помещает слово из вершины стека во флаговый регистр и увеличивает значение **SP** на 2.

ПРЕРЫВАНИЯ BIOS

INT 05H Печать экрана **-** выполняет вывод содержимого экрана на печатающее устройство. Команда INT 05H выполняет данную операцию из программы, а нажатие клавишей Ctrl/PrtSc - с клавиатуры. Операция запрещает прерывания и сохраняет позицию курсора.

INT 10H Управление дисплеем **-** обеспечивает экранные и клавиатурные операции. Прерывание INT 10H обеспечивает управление всем экраном. В регистре AH устанавливается код, определяющий функцию прерывания. Команда сохраняет содержимое регистров **BX, CX, DX, SI** и **BP.**

Пример:

```
                MOV  AH,02          ;Установить положение курсора
                MOV  BH,00          ;Страница 0
                MOV  DH,строка      ;Строка
                MOV  DL,столбец     ;Столбец
                INT  10H            ;Вызвать BIOS
```

или

```
                MOV  AX,0600H  Полная прокрутка вверх всего экрана,  очищая  его
```

пробелами

```
                MOV  AX,0601H  ;Прокрутить на одну строку вверх
                MOV  BH,07     ;Атрибут: нормальный, черно-белый
                MOV  CX,0000   ;Координаты от 00,00
                MOV  DX,184FH  ; до 24,79 (полный экран)
                INT  10H       ;Вызвать BIOS
```

INT 11H Запрос списка присоединенного оборудования **-** определяет наличие различных устройств в системе, результирующее значение возвращает в регистре AX. При включении компьютера система выполняет эту операцию и сохраняет содержимое AX в памяти по адресу 410h. Значения битов в регистре AX:

Бит	Устройство
15,14	Число подключенных принтеров
13	Последовательный принтер
12	Игровой адаптер
11-9	Число последовательных адаптеров стыка RS232
7,6	Число дискетных дисководов, при бите 0=1: 00=1, 01=2, 10=3 и 11=4
5,4	Начальный видео режим: 00 = не используется, 01 = 40x25 плюс цвет, 10 = 80x25 плюс цвет, 11 = 80x25 черно-белый режим
1	Значение 1 говорит о наличии сопроцессора
0	Значение 1 говорит о наличии одного или более дисковых устройств и загрузка операционной системы должна осуществляться с диска

INT 12H Запрос размера физической памяти **-**возвращает в регистре AX размер памяти в килобайтах, например, шест.200 соответствует памяти в 512 К. Данная операция полезна для выравнивания размера программы в соответствии с доступной памятью.

INT 13H Дисковые операции ввода/вывода **-** обеспечивает операции ввода-вывода для дискет и винчестера.

INT 14H Управление коммуникационным адаптером **-** обеспечивает последовательный ввод-вывод через коммуникационный порт RS232. Регистр DX должен содержать номер (0 или 1) адаптера стыка RS232. Четыре типа операции, определяемые регистром AH, выполняют прием и передачу символов и возвращают в регистре AX байт состояния коммуникационного порта.

INT 15H Кассетныеоперациивводa-вывода и специальные функциидлякомпьютеров AT **-** обеспечивает операции ввода-вывода для кассетного магнитофона, а также расширенные операции для компьютеров AT.

INT 16H Ввод склавиатуры **-** обеспечивает три типа команд ввода с клавиатуры.

INT 17H Вывод напринтер **-** обеспечивает вывод данных на печатающее устройство.

INT 18H Обращение к BASIC, встроенному в ROM **-** вызывает BASIC-интерпретатор, находящийся в постоянной памяти ROM.

INT 19H Перезапуск системы **-** данная операция при доступном диске считывает сектор 1 с дорожки 0 в область начальной загрузки в памяти (сегмент 0, смещение 7C00) и передает управление по этому адресу. Если дисковод не доступен, то операция передает управление через INT 18H в ROM BASIC. Данная операция не очищает экран и не инициализирует данные в ROM BASIC, поэтому ее можно использовать из программы.

INT 1AH Запрос и установка текущего времени и даты **-** считывает и записывает показание часов в соответствии со значением в регистре AH. Для определения продолжительности выполнения программы можно перед началом выполнения установить часы в 0, а после считать текущее время. Отсчет времени идет примерно 18,2 раза в секунду. Значение в регистре AH соответствует следующим операциям:

AH=00 Запрос времени. В регистре CX устанавливается старшая часть значения, а в регистре DX - младшая. Если после последнего запроса прошло 24 часа, то в регистре AL будет не нулевое значение.

AH=01 Установка времени. Время устанавливается по регистрам CX (старшая часть значения) и DX (младшая часть значения).

Коды 02 и 06 управляют временем и датой для AT.

INT 1FH Адрес таблицы графических символов **-** в графическом режиме имеется доступ к символам с кодами 128-255 в 1К таблице, содержащей по восемь байт на каждый символ. Прямой доступ в графическом режиме обеспечивается только к первым 128 ASCII-символам (от 0 до 127).

ПРЕРЫВАНИЯ DOS

Во время своей работы BIOS использует два модуля DOS: IBMBIO.COM и IBMDOS.COM. Так как модули DOS обеспечивают большое количество разных дополнительных проверок, то операция DOS проще в использовании и менее машиннозависимы, чем их BIOS аналоги.

Модуль IBMBIO.COM обеспечивает интерфейс с BIOS низкого уровня. Эта программа выполняет управление вводом-выводом при чтении данных из внешних устройств в память и записи из памяти на внешние устройства.

Модуль IBMDOS.COM содержит средства управления файлами и ряд сервисных функций, таких как блокирование и деблокирование записей. Когда пользовательская программа выдает запрос INT 21H, то в программу IBMDOS через регистры передается определенная информация. Затем программа IBMDOS транслирует эту информацию в один или несколько вызовов IBMBIO, которая в свою очередь вызывает BIOS. Указанные связи приведены на следующей схеме:

Пользовательский уровень	**Высший уровень**	**Низший уровень**	**ROM**	**Внешний уровень**
Программный запрос в/в	DOS IBMDOS.COM	DOS IBMBIO.COM	BIOS	Устройство

Прерывания от 020h до 062h зарезервированы для операций DOS. Ниже приведены наиболее основные из них:

INT 20H Завершение программы **-** запрос завершает выполнение программы и передает управление в DOS. Данный запрос обычно находится в основной процедуре.

INT 21H Запрос функций DOS **-** Основная операция DOS, вызывающая определенную функцию в соответствии с кодом в регистре AH.

INT 22H Адрес подпрограммы обработки завершения задачи **(см.INT 24H).**

INT 23H Адрес подпрограммы реакции на Ctrl/Break **(см.INT 24H).**

INT 24H Адрес подпрограммы реакции на фатальную ошибку **-** в этом элементе и в двух предыдущих содержатся адреса, которые инициализируются системой в префиксе программного сегмента и, которые можно изменить для своих целей.

INT 25H Абсолютное чтение с диска

INT 26H Абсолютная запись надиск

INT 27H Завершение программы,оставляющее ее резидентной **-** позволяет сохранить COM-программу в памяти.

ФУНКЦИИ ПРЕРЫВАНИЯ DOS INT 21H

Ниже приведены базовые функции для прерывания **DOS INT 21H**. Код функции устанавливается в регистре **AH:**

00 Завершение программы (аналогично INT 20H).
01 Ввод символа с клавиатуры с эхом на экран.
02 Вывод символа на экран.
03 Ввод символа из асинхронного коммуникационного канала.
04 Вывод символа на асинхронный коммуникационный канал.
05 Вывод символа на печать.
06 Прямой ввод с клавиатуры и вывод на экран.
07 Ввод с клавиатуры без эха и без проверки Ctrl/Break.
08 Ввод с клавиатуры без эха с проверкой Ctrl/Break.
09 Вывод строки символов на экран.
0A Ввод с клавиатуры с буферизацией.
0B Проверка наличия ввода с клавиатуры.
0C Очистка буфера ввода с клавиатуры и запрос на ввод.
0D Сброс диска (гл.16).
0E Установка текущего дисковода.
0F Открытие файла через FCB.
10 Закрытие файла через FCB.

11 Начальный поиск файла по шаблону.
12 Поиск следующего файла по шаблону.
13 Удаление файла с диска.
14 Последовательное чтение файла.
15 Последовательная запись файла.
16 Создание файла.
17 Переименование файла.
18 Внутренняя операция DOS.
19 Определение текущего дисковода.
1A Установка области передачи данных (DTA.
1B Получение таблицы FAT для текущего дисковода.
1C Получение FAT для любого дисковода.
21 Чтение с диска с прямым доступом.
22 Запись на диск с прямым доступом.
23 Определение размера файла.
24 Установка номера записи для прямого доступа.
25 Установка вектора прерывания.
26 Создание программного сегмента.
27 Чтение блока записей с прямым доступом.
28 Запись блока с прямым доступом.
29 Преобразование имени файла во внутренние параметры.
2A Получение даты (CX-год,DH-месяц,DL-день).
2B Установка даты.
2C Получение времени (CH-час,CL-мин,DH-с,DL-1/100с).
2D Установка времени.
2E Установка/отмена верификации записи на диск.