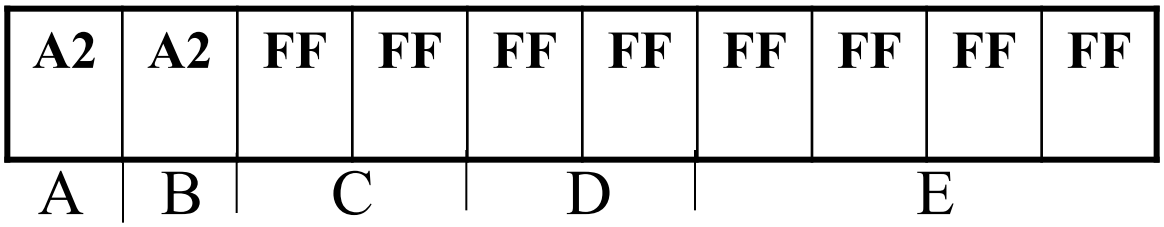
Директивы определения данных

[<имя>] **DB <операнд>** {,<операнд>}
[<имя>] **DW <операнд>** {,<операнд>}
[<имя>] **DD <операнд>** {,<операнд>}

Примеры:
A DB 162 ;описать константу-байт 162 и дать ей имя A
B DB 0A2h ;такая же константа, но с именем B
C DW -1 ;константа-слово -1 с именем C
D DW 0FFFFh ;такая же константа-слово, но с именем D
E DD -1 ;-1 как двойное слово



Расположение объявленных переменных в памяти

Команды общего назначения.

*Команда **MOV*** приемник, источник (reg1/mem1),(reg2/mem2).

Примеры:

MOV AX, TABLE ; из памяти в регистр

MOV TABLE, AX ; из регистра в память

MOV ES:[BX], AX ; с заменой используемого регистра сегмента

MOV CL, -30 ; константа в регистр

MOV TABLE, 5H ; константа в память

Исключения:

1) нельзя пересылать из памяти в память (надо через регистр);

2) нельзя непосредственно адресуемый операнд пересылать в регистр сегмента;

3) нельзя пересылать регистр сегмента в регистр сегмента (надо через регистр общего назначения);

4) нельзя использовать регистр CS в качестве приемника.

Команда ***PUSH*** источник ;слово пересылается на вершину стека.

Команда ***POP*** приемник ; вершина стека пересылается в слово.

Можно использовать любой 16-разрядный регистр (включая CS).

Примеры:

PUSH DX ; сохранить содержимое регистра DX в стеке

POP DS ; загрузить в регистр DS значение из вершины стека

Пересылка адресов.

*Команда **LEA*** регистр 16, память 16.

Пересылает смещение ячейки памяти в любой 16-битовый регистр общего назначения, регистр указателя или индексный регистр.

В отличие от команды MOV с операцией OFFSET, операнд «память 16» в команде LEA может быть индексирован.

Пример:

TAB dw 270, 304 , 500, 777, 907

. . .

MOV DI, 6 ; загружаем в регистр DI непосредственное число 6

LEA BX, TAB [DI] ; пересылает в регистр BX адрес четвертого элемента.

Это аналогично

MOV BX, OFFSET TAB ; пересылаем в регистр BX адрес первого элемента

ADD BX, 6 ; увеличиваем это смещение на шесть, получаем адрес четвертого элемента.

Если использовать команду MOV без OFFSET

MOV BX, TAB ; в регистр BX будет загружен первый элемент, то есть число 270.

Пересылка флагов.

*Команда **LAHF*** (load AH from Flags)

Команда LAHF без параметров. Загружает в регистр AH флаги CF, PF, AF, ZF, SF в соответствующие разряды (0, 2, 4, 6, 7).

*Команда **SAHF*** (store AH into Flags)

Загружает пять упомянутых выше разрядов регистра AH в регистр флагов.

*Команды **PUSHF**, **POPF*** - пересылка регистра флагов в стек и обратно.

Пример: обратиться к процедуре с сохранением флагов и регистра AX.

```
PUSH AX  ; сохранить содержимое регистра AX в стеке
PUSHF    ; сохранить содержимое регистра флагов в стеке
CALL SORT ; вызвать процедуру с именем  SORT
POPF   ; восстановить регистр флагов
POP AX ; восстановить содержимое регистра AX
```

Команды ввода-вывода

*Команда **IN*** аккумулятор, порт.

*Команда **OUT*** порт, аккумулятор.

Аккумулятор - регистр AL при обмене байтами и регистр AX при обмене словами.

Номер порта - десятичное значение от 0 до 255. В качестве операнда «порт» необходимо использовать регистр DX, если требуется указать порт больше 255.

Примеры:

IN AX, 30h ; ввести слово из порта 30h

MOV DX, 340 ; загрузить в регистр DX адрес порта

IN AL, DX ; ввести байт из порта 340

OUT 61H, AL ; вывести байт в порт 61h

OUT DX,AX ; вывести слово в порт, указанный DX.

(например: порт 96 - от клавиатуры, 97 - динамик, 64-67 - таймер).

Логические команды.

*Команда **AND*** (reg1/mem1),(reg2/mem2) ; логическое «И»

AND AX,BX

*Команда **OR*** (reg1/mem1), (reg2/mem2) ; логическое «ИЛИ»

OR TEMP, CX

*Команда **XOR*** (reg1/mem1), (reg2/mem2) ; сложение по mod2

XOR AL, 7

*Команда **NOT*** (reg/mem) ; логическое «НЕ»

NOT CH

*Команда **TEST***(reg1/mem1), (reg2/mem2) ; логическое «И» без записи в результат.

TEST BH,BL

В качестве источника могут использоваться непосредственные данные

Команды сдвига.

*Команда **SAL приемник, счетчик***

(reg),(1 или количество сдвигов в CL) ; сдвиг арифметический влево

SAL AL,1

MOV CL,5

SAL AX,CL

*Команда **SAR приемник, счетчик*** ;сдвиг арифметический вправо

*Команда **SHL приемник, счетчик*** ;сдвиг логический влево

*Команда **SHR приемник, счетчик*** ;сдвиг логический вправо

1.6 Команды сдвига и циклического сдвига

Логический сдвиг операнда влево/вправо

SHL операнд, количество_сдвигов
SHR операнд, количество_сдвигов

SHL и SHR сдвигают биты операнда (регистр/память) влево или вправо соответственно на один разряд и изменяют флаг переноса cf. При логическом сдвиге все биты равноправны, а освободившиеся биты заполняются нулями. Указанное действие повторяется количество раз, равное значению второго операнда.

Пример:
; al = 01011011 (двоичное)
shr al, 3

Это означает: сдвиг всех битов регистра al на 3 разряда вправо. Так что al станет 00001011. Биты слева заполняются нулями, а биты справа выдвигаются. Последний выдвинутый бит, становится значением флага переноса cf.

Команда shl такая же, как и shr, но сдвигает влево.
; bl = 11100101 (двоичное)
shl bl, 2

После выполнения команды регистр bl будет равен 10010100 (двоичное). Два последних бита заполнились нулями, флаг переноса установлен, потому, что последний выдвинутый слева бит был равен 1

Арифметический сдвиг влево/вправо

```
SAL операнд, количество_сдвигов
SAR операнд, количество_сдвигов
```

Команда **SAR** — сдвигает биты операнда (регистр/память) вправо на один разряд, значение последнего вытолкнутого бита попадает в флаг переноса, а освободившиеся биты заполняются знаковым битом.

Пример:

```
; al = 10100110
sar al, 3
; al = 11110100
sar al, 2
; al = 11111101

; bl = 00100110
sar bl, 3
; bl = 00000010
```

Команда **SAL** — сдвигает биты операнда (регистр/память) влево на один разряд, значение последнего вытолкнутого бита попадает в флаг переноса, а освободившиеся биты заполняются нулями, при этом знаковый бит не двигается.

Пример:

```
; al = 10100110
sal al, 3
; al = 10110000
sal al, 4
; al = 10000000
```

Команды циклического сдвига

rol операнд, количество_сдвигов
ror операнд, количество_сдвигов
rcl операнд, количество_сдвигов
rcr операнд, количество_сдвигов

Циклический сдвиг напоминает смещение, выдвигаемые биты, снова вдвигаются с другой стороны:
Пример: команды ror (циклический сдвиг вправо)

	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	выдвигается
Операнд	1	0	0	1	1	0	1	1	
ror операнд,3				1	0	**0**	**1**	**1**	011
Результат	**0**	**1**	**1**	1	0	0	1	1	

Из таблицы видно, что биты вращаются, то есть каждый бит, который выталкивается снова вставляется с другой стороны. Флаг переноса cf содержит значение последнего выдвинутого бита.

ROL и ROR сдвигают все биты операнда влево(для ROL) или вправо (для ROR) на один разряд, при этом старший (для ROL) или младший (для ROR) бит операнда вдвигается в операнд справа(для ROL) или слева (для ROR) и становится значением младшего(для ROL) или старшего (для ROR) бита операнда; одновременно выдвигаемый бит становится значением флага переноса cf. Указанные действия повторяются количество раз, равное значению второго операнда.

RCL и RCR сдвигают все биты операнда влево (для RCL) или вправо (для RCR) на один разряд, при этом старший (для RCL) или младший(для RCR) бит становится значением флага переноса cf; одновременно старое значение флага переноса cf вдвигается в операнд справа (для RCL) или слева (для RCR) и становится значением младшего (для RCL) или старшего(для RCR) бита операнда. Указанные действия повторяются количество раз, равное значению второго операнда.

Сложение.

*Команда **ADD*** приемник, источник (reg1/mem1),(reg2/mem2)

Выполняется два действия:

а) приемник = приемник + источник

б) формируется флаг переноса при выходе результата за разрядную сетку.

*Команда **ADC*** приемник, источник (reg1/mem1),(reg2/mem2)

Выполняется сложение с учетом флага переноса:

Приемник = приемник + источник + флаг переноса

Пример: сложить 32-разрядные числа

Первое число в регистрах (DX,CX)

Второе число в регистрах (BX,AX)

ADD AX, CX ; сложить младшие шестнадцать битов

ADC BX, DX ; сложить старшие 16 битов с учетом переноса

Вычитание.

*Команда **SUB*** приемник, источник (reg1/mem1),(reg2/mem2)

Вычитание с формированием знака переноса:

приемник = приемник – источник.

*Команда **SBB*** приемник, источник (reg1/mem1),(reg2/mem2)

Вычитание с использованием знака переноса (вычитание с заемом):

приемник = приемник – источник – перенос.

Например:

```
DAN1 dw 75, 40, 35    ; данные объявлены в сегменте данных
. . .
MOV SI, 505           ; загрузить в регистр SI первый операнд
SUB SI, DAN1          ; вычесть из регистра SI число 75, результат поместить в регистр SI и установить флажок CF
```

Приращение приемника на единицу.

*Команда **INC*** приемник (reg/mem)

INC AL ; увеличить содержимое регистра AX на единицу.

Уменьшение приемника на единицу.

*Команда **DEC*** приемник (reg/mem)

DEC [BX] ; уменьшить содержимое ячейки памяти, адрес которой находится в регистре BX.

Обращение знака.

*Команда **NEG*** приемник (reg/mem)

Выполнение: приемник = 0 - приемник.

NEG PRIM[SI]; поменять знак у содержимого ячейки памяти, адрес которой определяется именем PRIM плюс значение в регистре SI.

Сравнение значений.

*Команда **CMP*** (compare).

CMP приемник, источник (reg1/mem1), (reg2/mem2)

CMP PRIM1[BX+SI], CX; сравнить значение в регистре CX и ячейке памяти, адрес которой определяется именем PRIM1 плюс сумма содержимого регистров BX и SI.

Команды расширения знака

CBW –воспроизводит седьмой бит регистра AL во всех битах регистра AH

CWD –воспроизводит 15-ый бит регистра AX во всех битах регистра DX

Команды умножения

*Команда **MUL*** множитель (reg/mem); беззнаковое умножение

*Команда I**MUL*** множитель (reg/mem); знаковое умножение

Умножение байта на байт. Множимое находится в регистре AL, а множитель в байте памяти или в однобайтовом регистре. После умножения результат находится в регистре AX.

Умножение слова на слово. Множимое находится в регистре AX, а множитель - в слове памяти или в регистре. После умножения произведение находится в двойном слове: старшая (левая) часть произведения находится в регистре DX, а младшая (правая) часть в регистре AX.

Примеры умножения.

```
data1 segment para public  'data‘  ;Описание сегмента  данных
byte1 db 80h
byte2 db 40h
word1 dw 8000h
word2 dw 4000h                    ; Данные
data1 ends
code1 segment para public 'code'          ; Сегмент кода
. . . .
MOV AL, byte1                             ;байт на байт
MUL byte2                        ;произведение в AX

MOV AX, word1                             ;слово на слово
MUL word2                        ;произведение в DX:AX

MOV AL, byte1                             ;байт на слово
SUB AH, AH                                ;обнуление AH
MUL word1                        ;произведение в DX:AX
```

```
                    ;Знаковое умножение
MOV AL, byte1                    ;байт на байт
IMUL byte2                ;произведение в AX

MOV AX, word1                    ;слово на слово
IMUL word2                ;произведение в DX:AX

MOV AL, byte1             ;байт на слово
CBW                        ;расширение множимого в  AH
IMUL word1           ;произведение в DX:AX
 . . . .
Code ends
End
```

При умножении на степень числа 2 (2,4,8 и т.д.) более эффективным является сдвиг влево на требуемое число битов. Сдвиг более чем на 1 требует загрузки величины сдвига в регистр CL. В следующих примерах предположим, что множимое находится в регистре AL или AX:

Умножение на 2: SHL AL,1

Умножение на 8: MOV CL,3

Деление.

*Команда **DIV*** делитель (reg/mem); беззнаковое деление

*Команда **IDIV*** делитель (reg/mem); знаковое деление

Деление «слова на байт». Делимое находится в регистре AX, а делитель - в байте памяти или а однобайтовом регистре. После деления остаток получается в регистре AH, а частное - в AL. Так как однобайтовое частное очень мало (максимально +255 (шест.FF) для беззнакового деления и +127 (шест.7F) для знакового), то данная операция имеет ограниченное использование.

Деление «двойного слова на слово». Делимое находится в регистровой паре DX:AX, а делитель - в слове памяти или а регистре. После деления остаток получается в регистре DX, а частное в регистре AX. Частное в одном слове допускает максимальное значение +32767 (шест.FFFF) для беззнакового деления и +16383 (шест.7FFF) для знакового.

Примеры деления.

```
data1 segment para public 'data‘ ;Описание сегмента данных
byte1 db 80h
byte2 db 16h
word1 dw 2000h
word2 dw 0010h
word3 dw 1000h ; Данные
data1 ends
code1 segment para public 'code' ; Сегмент кода
. . . .
MOV AX, word1 ;слово / на байт
DIV byte1 ;ост. : частное в AH:AL
MOV AL, byte1 ;байт / на байт
SUB AH, AH ;обнуление AH
DIV byte2 ; ост. : частное в AH:AL
MOV DX, word2 ;двойное слово / на слово
MOV AX, word3 ;делимое в DX:AX
DIV word1 ;ост. : частное в AH:AL
MOV AX, word1 ;слово / на слово
SUB DX, DX ;обнуление DX
DIV word3 ; ост. : частное в DX:AX
```

```
; Знаковое деление
MOV AX, word1                ;слово / на байт
IDIV byte1                   ;ост. : частное в AH:AL
MOV AL, byte1                ;байт / на байт
CBW                          ;расширяем делимое в  AH
IDIV byte2                   ; ост. : частное в AH:AL
MOV DX, word2                ;двойное слово / на слово
MOV AX, word3                ;делимое в DX:AX
IDIV word1                   ;ост. : частное в AH:AL
MOV AX, word1                ;слово / на слово
CWD                          ;расширяем делимое в  DX
DIV word3                    ; ост. : частное в DX:AX
 . . . .
code ends
End
```

При делении на степень числа 2 (2, 4, и т.д.) более эффективным является сдвиг вправо на требуемое число битов. В следующих примерах предположим, что делимое находится в регистре AX:

```
Деление на 2:    SHR  AX,1
Деление на 8:    MOV  CL,3
                 SHR  AX,CL
```

Команды безусловных переходов

Двухбайтная команда ***JMP dispL*** содержит во втором байте смещение, которое интерпретируется как знаковое целое. Пример:

(IP)=1240 JMP E8 (IP)=1228

Трехбайтная команда ***JMP disp*** производит такое же действие, как предыдущая команда, но содержит 16-битное смещение. При этом увеличивается область перехода до -32768 (+32767 относительно адреса команды, находящейся после команды JMP disp.

Пример:

(PC)=3E60 JMP 0002 (PC)=4060

Команда ***JMP mem/reg*** реализует косвенный безусловный переход в программе.

Пример:

(BX)=3000 JMP BX (PC)=3000

Пример:

(BX)=68A0 JMP [BX] (PC)=3560

(DS)=AA00

([B08A0])=3560

Команда ***JMP addr*** (прямого межсегментного перехода) : значение **offset** загружается в IP, а значение **segment -** в регистр CS.

Пример:

(PC)=18A6 JMP 0020 A040 (PC)=2000

(CS)=0200 (CS)=40A0

Команда ***JMP mem*** (косвенного межсегментного перехода) допускает адресацию только памяти. Слово из адресуемой ячейки памяти загружается в PC, а следующее слово - в регистр CS.

Пример:

(DS)=6000 JMP[DI+100H]

(PC)=3750

(DI)=2680 (CS)=F000

([62780])=3750

([62782])=F000

Команды условного перехода

Общий формат команд условного перехода:

JCC <метка перехода>,

где J - первая буква от Jump – прыжок;

CC описывают в сокращенном виде условия перехода.

По этому признаку все команды делятся на 3 подгруппы.

1-я группа команд условного перехода (значение CC):

E - Equal = равно

N - Not <> не, отрицание

G - Greater > больше, для чисел со знаком

L - Less < меньше, для чисел со знаком

A - Above > больше (выше), для чисел без знака

B -Below < меньше (ниже), для чисел без знака

Вторая группа:

Мнемокод/ синонимы	Состояние флагов	Мнемокод/ синонимы	Состояние флагов
JZ \ JE	ZF=1	JNZ / JNE	ZF=0
JS	SF=1	JNS	SF=0
JC /JB /JNAE	CF=1	JNC/JAE/ JNB	CF=0
JO	OF=1	JNO	OF=0
JP	PF=1	JNP	PF=0

Третья группа:

JCXZ <метка> ; (Jump if CX is Zero).

Эта команда проверяет не флаги (как другие команды условного перехода), а содержимое регистра CX.

Пример: При далекой метке M, оператор **IF AX=BX THEN GOTO M** следует реализовать так:

IF AX<>BX THEN GOTO L ; (короткий переход)

GOTO M ; (длинный переход)

L: . . . .

На ЯА это записывается следующим образом:

CMP AX, BX

JNE L

JMP M

L: . . . .

Команды управления циклами

Команда	Действие	Используемые данные
LOOP	Повторяет цикл до конца счетчика, т.е. уменьшает CX на 1 и передает управление на метку, если CX не равен 0	CX-счетчик
LOOPE LOOPZ	Уменьшает CX на 1 и осуществляет переход, если CX не равно 0 и флаг нуля ZF=1	CX-счетчик ZF-флаг нуля (пока ZF=1)
LOOPNE LOOPNZ	Уменьшает CX на 1 и осуществляет переход, если CX не равно 0, и флаг нуля ZF=0	CX-счетчик ZF-флаг нуля (пока ZF=0)

Команда прерывания

Команда ***INT*** **<тип прерывания>** (тип прерывания - число от 0 до 255)

INT **21h**

Выполнение:

1. Регистр флагов загружается в стек.
2. Обнуляет флаг трассировки TF и флаг включения/выключения прерываний IF для исключения пошагового режима выполнения команд и блокировки других маскируемых прерываний.
3. Помещается в стек значение регистра CS.
4. Вычисляется адрес вектора прерываний (умножением <типа прерывания> на 4).
5. Загружается второе слово вектора прерываний в регистр CS.

- Помещается в стек значение указателя команд IP.
- Загружается в указатель команд IP первое слово вектора прерываний.

Команды обработки строк

Команда	Операнды	Байт	Слово
MOVS	DI,SI	MOVSB	MOVSW
LODS	AL,SI или AX,SI	LODSB	LODSW
STOS	DI,AL или DI,AX	STOSB	STOSW
CMPS	SI,DI	CMPSB	CMPSW
SCAS	DI,AL или DI,AX	SCASB	SCASW

```
STRING1   DB   20 DUP('*')
STRING2   DB   20 DUP(' ')
                ...
CLD                        ;Сброс флага DF
MOV  CX,20                 ;Счетчик на 20 байт
LEA  DI,STRING2    ;Адрес области "куда"
LEA  SI,STRING1     ;Адрес области "откуда"
REP MOVSB                ;Переслать данные
```

REP - повторять операцию, пока CX не равно 0;

REPZ или **REPE** - повторять операцию, пока флаг ZF показывает "равно или ноль". Прекратить операцию при флаге ZF, указывающему на не равно или не ноль или при CX равном 0;

REPNE или **REPNZ** - повторять операцию, пока флаг ZF показывает "не равно или не ноль". Прекратить операцию при флаге ZF, указывающему на "равно или нуль" или при CX равным 0.

```
 DATASG  SEGMENT  'Data'
        NAME1   DB      'Assemblers'    ;Элементы данных. Строка из 10
символов
        NAME2   DB      10 DUP(' ')     ;Две строки чистых для работы
        NAME3   DB      10 DUP(' ')
 DATASG  ENDS
        ; -----------------------------------------------
 CODESG  SEGMENT  'Code'
       ASSUME  CS:CODESG, DS:DATASG, SS:STACKSG, ES:DATASG
       MOV     AX,DATASG
       MOV     DS,AX
       MOV     ES,AX
      CLD
      MOV     CX,10
      LEA     SI,NAME1
      LEA     DI,NAME2
     REPE CMPSB       ;Сравнить NAME1 и NAME2
     JNE     G20              ;Не равны?
     MOV     BH,01
     …..
     G20:    MOV     CX,10
```

```
    G20:   MOV    CX,10
    LEA    DI,NAME1
    MOV    AL,'m'          ;Поиск символа 'm'
    REPNE SCASB            ;  в NAME1
    JNE    H20             ;Если не найден
    MOV    AH,03
H20:   ....
```