**Форматы команд**

|  |  |  |
| --- | --- | --- |
| КОП | ОП1 | ОП2 |

• регистр – регистр

• регистр – память

• память – регистр

• регистр - непосредственный операнд

• память - непосредственный операнд

**КОНСТАНТЫ**

* Числовые целые константы

десятичные 456 -876 78

шестнадцатеричные 0a43h 0xa43

восьмеричные 7734q -76577q

двоичные 11101101b

* Числовые вещественные константы

форма с фиксированной точкой 11.6785 -78.6743

экспоненциальная форма записи 1.45Е-3 -0.67E7

* Символьные константы

укладывается в длину двойного слова (не более 4-х символов)

заключается в кавычки или в апострофы.

c db ‘\*’

c1 dd ‘abcd’

несмотря на то, что в двойном слове, байты располагаются в обратном порядке, в данном случае — прямой порядок.

* Строковые константы

заключается в кавычки или в апострофы.

Содержат больше 4-х символов.

Ассемблер может воспринимать в качестве строковых такие

константы, которые короткие и могли бы считаться символьными.

Например, видит символьную константу длиной более одного

символа в директивах db или более двух в директивах dw.

Регистры общего назначения

**EAX**

AX

|  |  |  |  |
| --- | --- | --- | --- |
|  | AH | AL | аккумулятор |

**EBX**

BX

|  |  |  |  |
| --- | --- | --- | --- |
|  | BH | BL | база |

**ECX**

CX

|  |  |  |  |
| --- | --- | --- | --- |
|  | CH | CL | счетчик |

**EDX**

DX

|  |  |  |  |
| --- | --- | --- | --- |
|  | DH | DL | данные |

ESI / SI индекс источника

EDI / DI индекс приемника

EBP/ BP указатель базы

ESP / SP указатель стека

Сегментные регистры

CS сегмент команд

DS сегмент данных

SS сегмент стека

ES дополнительный сегмент

Указатель команд EIP /IP

Регистр флагов

eflags / flags

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | OF | DF | IF | TF | SF | ZF |  | AF |  | PF |  | CF |

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

CF флаг переноса AF флаг дополнительного переноса

OF флаг переполнения DF флаг направления

ZF флаг нуля IF флаг прерываний

SF флаг знака TF флаг трассировки

PF флаг четности

1. Идентификаторы
2. Целые числа

десятичные 5, 8, -35, -340d

двоичные 100b, 101010b, -11b

восьмеричные 231q, -784q

шестнадцатиричные 98h, -1BBFh,

AFh => 0AFh 0xAF

1. Символьные данные

‘R’ “R” одиночные символы

‘rtyuFHG’ “rtyuFHG” символьные строки

1. Предложения

Комментарии ;

comment <маркер> <текст> <маркер>

Команды [<метка>:] <мнемокод> [<операнды>] [;<комментарии>]

Директивы [<имя>] <директива> [<операнды>] [;<комментарии>]

**ВЫРАЖЕНИЯ**

* **Вычисляемые выражения**

выражение должны быть целочисленным, т.е. состоять из целочисленных констант и меток и использовать следующие операции

- + и - сложение и вычитание

\* умножение

/ и % целочисленное деление и остаток от деления (для беззнаковых целых чисел)

// и %% целочисленное деление и остаток от деления (для знаковых целых чисел)

& , |, ^ операция побитового «и», «или», «исключающего или»

<< и >> операции побитового сдвига влево и вправо

-и + унарные. – меняет знак числа на противоположный, + ничего не делает

~ унарная операция, означает побитовое отрицание

При применении операций % и %% необходимо оставлять

Пробельный символ после знака операции.

Унарные операции имеют самый высокий приоритет, следом за

ними идут операции умножени, деления и остатка от деления,

далее – сложение и вычитание.

Затем операции сдвигов, &, ^, и |.

Порядок выполнения операций можно изменить, применив

круглые скобки.

**Директивы резервирования памяти (задание исходных данных)**

section .data

**DB**

**DW**

**DD**

**DQ**

**DT**

a db 3,9,15 ; 3 → a

; 9 → a + 1

; 15 → a+2

x times 5 dd 0 ; x dd 0, 0, 0, 0, 0

; 0 → x

; 0 → x + 4

…………..

; 0 → x + 16

**Директивы резервирования неинициализированной памяти**

section .bss

**RESB**

**RESW**

**RESD**

**RESQ**

**REST**

t resb 3 ; выделено 3 байта неинициализированной памяти с адресами

; t, t + 1, t + 2

s resw 5 ; выделено 5 слов неинициализированной памяти с адресами

; s, s + 2, s + 4, s + 6, s + 8

**Директивы эквивалентности**

**<имя> EQU <операнд>**

k**=**1 k=1

N **EQU**  k N **EQU** k+10

A **DW** k A=? C **DW** N C=?

k=2 k=2

B **DW** N B=? D **DW** N D=?

**Виды операндов**

* если в роли операнда выступает название регистра, то этот операнд – ***регистровый операнд*** (r8, r16, r32)
* если в роли операнда указано значение, то этот операнд – ***непосредственный операнд*** (i8, i16, i32)
* если в роли операнда указан операнд типа “память”, то этот операнд – ***адресный операнд*** (m8, m16, m32). В NASM операнд типа “память” всегда обозначается [ ], в которых и пишется собственно адрес

**Команды пересылки MOV и перестановки XCHG**

**mov op1, op2 op2 → op1**

|  |  |  |
| --- | --- | --- |
| op1 | op2 |  |
| r8 | i8, r8, m8 | пересылка байтов |
| m8 | i8, r8 |  |
| r16 | i16, r16, m16 | пересылка слов |
| m16 | i16, r16 |  |
| r32 | i32, r32, m32 | пересылка двойных слов |
| m32 | i32, r32 |  |

xchg **op1, op2 op2 → op1, op1 → op2**

|  |  |  |
| --- | --- | --- |
| op1 | op2 |  |
| r8 | r8, m8 | перестановка байтов |
| m8 | r8 |  |
| r16 | r16, m16 | перестановка слов |
| m16 | r16 |  |
| r32 | r32, m32 | перестановка двойных слов |
| m32 | r32 |  |

section .data

a db 56

b dd 1234

c db 62

……………

mov al, [a]

mov ebx, [b]

xchg [c], al

**Спецификатор размера ( оператор явного указания типа)**

BYTE 1 байт

WORD 2 байта

DWORD 4 байта

Используется в случае необходимости явно указать размер операнда, или использовать только часть содержимого ячейки, адрес которой указан в поле адреса команды.

mov byte [a], 25 ; необходимо дать информацию компилятору о том, константу

; какого размера необходимо сформировать ( a - адрес, а размер

; неизвестен)

mov al, 25 ; нет необходимости использовать byte, поскольку на размер

; константы указывает размер регистра al

mov ax, word [b+2] ; в регистр ax будет записан b+2 и b+3 байты двойного слова b

; (старшее слово двойного слова b)