Оглавление

[Реальный режим работы микропроцессора х86 2](#_Toc405538135)

[Микропроцессор с точки зрения программиста 2](#_Toc405538136)

[Модификация адреса 2](#_Toc405538137)

[Арифметические команды 3](#_Toc405538138)

[Оператор условного перехода 4](#_Toc405538139)

**Ассемблер**

# Реальный режим работы микропроцессора х86

## Микропроцессор с точки зрения программиста

Программная модель микропроцесса. Способ организации и способ доступа к оперативной памяти. Он зависит от разряда адресной шины.

210= 1 килобайт= 1024 байт

220= 1 мегабайт

230=1 гигабайт

240= 1 терабайт

640 под оперативную память, еще 640 адресное пространство ввода-вывода. Существуют следующие программно- доступные регистры (некоторые процессы памяти и микропроцессор имеет к ним прямой доступ).

Регистры общего назначения AX, CX, DX и BX двухбайтовые. К байтам можно обращаться отдельно. AH (BH, CH, DH) высокий (старший), AL (BL, CL, DL) низкий.

Сегмент данных, сегмент стека.

Первая буква расшифровка типа.

## Модификация адреса

Все сегменты размещаются в оперативной памяти начиная с границы параграфа, то есть адреса кратного 16.

F= 11112

10=100002

Первая ячейка в сегменте кратен 16 и она является адресом. Ячейки памяти внутри сегмента адресуются относительно начала сегмента. Флаги отмечают действие.

DB B-байт по адресу, которому присваивается символическое имя А

Адресация по базе с индексированием используется при организации обращения к двумерным массивам.

MOV AX,[BX]

Индексные регистры Di и Si.

Массивы A DB 1,3,4

0 1 2

Индексация по базе индексированием:

A DB 1,2,4

DB 7,5,9

DB 2,6,8

Смещение и дет 0,1,2,3…..

ВХ смещение начала строки относительно начала массива.

1ст. 2ст. 3ст.

1 3 4 7 5 9 2 6 7

0 1 2 3 4 5 6 7 8

ВХ 0 3 6

DI 0 1 2 0 1 2 0 1 2

A [BX] [DI]

3 1

## Арифметические команды

1. Сложение ADD on1, on2 on1: =on1+on2

MOV AL, 2

MOV BL, 3

ADD AL, BL (AD =5)

Флаги ZF, SF, CF

1. Сложение с учетом переноса ADC on1, on2 on1: =on1+on2+CF

300

Ст. б мл. б

+ 1 2C

1 E4

3 20

+ 500

800

MOV AL, 2CL

MOV BL 0F4L

MOV AM, 1L

MOV BM, 1L

ADD AL, BL

ADC AM, BM

1. Вычитание SUB on1, on2 on1: = on1-on2
2. Вычитание с флагами SBB on1, on2 on1: = on1-on2-CF
3. Инкремент INC on on: = on+1
4. Декремент DEC on on: =on-1
5. Умножение MUL Mm2

Mm1\*Mm2= произведение

|  |  |  |  |
| --- | --- | --- | --- |
| длина | Mm1 | Mm2 | Произведение |
| byte | AL | В команде | AX |
| слово | AX | В команде | DX:AX |

1. DIV Деление: делитель= частное (остаток)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Длина | Делимое | Делитель | Частное | Остаток |
| Byte | AX | В команде | AL | AM |
| Слово | DX:AX | В команде | AX | DX |

# Оператор условного перехода

Jxx <Метка>

Метка- символ, адрес операции на которую переходит.

M1: MOV AX, 2

При решении вопроса о переходе анализируются флаги. Прежде использования операции, необходимо флаги установить.

Например, нам надо сложить AL!=AM, если они не равны.

AL!=AM, то AL:=AL+AM

**AL:=AL+AM**

**AL! =AM**

**-**

**+**

CMP AL, AH AL-AM

AL<AM

FZ=0

FS=1

Команды различаются на знаковые и без знаковые.

Знаковые:

> JG

< JLZ M1

>= JGE

<= JLE

!= JNE ( не равно)

=JE

<0 JS

>=0 JNS

**Начало**

**A,N**

## Умножение при ненулевом делителе.

MOV AX, 150

MOV BL, 0

CMP BL, 0

JZ M1

**S=0**

DIV BL

M1:

**i=1**

AND 0n1, on 2 Конъюнкция ^

OR on1, 0n2 Дизъюнкция

+

**Ai>0**

NOT on1, Отрицание

**S=S+Ai**

XOR on1, on2, O

-

TEST on1, on2

**I=i+1**

Сумма положительных элементов

-

+

**Конец**

**S**

**i>N**

A DB 3, -4, 5, 2

N DW 4

S DB ?

XOR AL, 1; S=0

XOR BX,BX; i=1

MOV CX, N

M2. CMP A [BX], 0; ai<= 0

JLE M1

ADD AL,A [BX]; i=i+1

M1: WC BX

DEC CX

CMP CX, 0

JG M2

MOV S, AX

Цикл с предусловием

**Начало**

**I=i+1**

**Нечет?**

**I<= N**

**A,N**

**S+S+ai**

**I=1**

**S=0**

**S**

**Конец**

1. Задан одномерный массив байт длинной N. Найти сумму положительных элементов.

DATA SEGMENT PARA PUBLIC 'DATA'

A DB 3,-4,5,-2,1

N DW 5

S DW ?

DATA ENDS

;-----------------------------

STK SEGMENT STACK

DB 256 DUP(0)

STK ENDS

;-----------------------------

CODE SEGMENT PARA PUBLIC 'CODE'

MAIN PROC

ASSUME CS:CODE, DS:DATA, SS:STK

MOV AX,DATA

MOV DS,AX

XOR AL,AL

XOR BX,BX

MOV CX,N

M2: CMP A[BX],0

JLE M1

ADD AL,A[BX]

M1: INC BX

DEC CX

CMP CX,0

JG M2

MOV S,AX

MOV AX,4C00h

INT 21h

MAIN ENDP

CODE ENDS

END MAIN

2) Программа №1 с данными длиной слово.

DATA SEGMENT PARA PUBLIC 'DATA';??????? ??????

A DW 3,-4,5,-1,2

N DW 5

S DW ?

DATA ENDS

;-----------------------------

STK SEGMENT STACK;??????? ?????

DB 256 DUP(0)

STK ENDS

;-----------------------------

CODE SEGMENT PARA PUBLIC 'CODE'; ??????? ????

MAIN PROC

ASSUME CS:CODE, DS:DATA, SS:STK

MOV AX,DATA

MOV DS,AX

XOR AX,AX

XOR BX,BX

MOV CX,N

M2: CMP A[BX],0

JLE M1

ADD AX,A[BX]

M1: ADD BX,2

DEC CX

CMP CX,0

JG M2

MOV S,AX

MOV AX,4C00h

INT 21h

MAIN ENDP

CODE ENDS

END MAIN

1. Задан массив, найти сумму четных элементов для байта

DATA SEGMENT PARA PUBLIC 'DATA';??????? ??????

A DB 3,2,5,4

N DB 4

S DB ?

DATA ENDS

;-----------------------------

STK SEGMENT STACK;??????? ?????

DB 256 DUP(0)

STK ENDS

;-----------------------------

CODE SEGMENT PARA PUBLIC 'CODE'; ??????? ????

MAIN PROC

ASSUME CS:CODE, DS:DATA, SS:STK

MOV AX,DATA

MOV DS,AX

XOR AL,AL

XOR BX,BX

MOV CL,N

M2: TEST A[BX],0000000000000001b

JNZ M1

ADD AL,A[BX]

M1: INC BX

DEC CL

CMP CL,0

JG M2

MOV S,AL

MOV AX,4C00h

INT 21h

MAIN ENDP

CODE ENDS

END MAIN

4) и слово

DATA SEGMENT PARA PUBLIC 'DATA'

A DW 3,4,5,1,2

N DW 5

S DW ?

DATA ENDS

;-----------------------------

STK SEGMENT STACK

DB 256 DUP(0)

STK ENDS

;-----------------------------

CODE SEGMENT PARA PUBLIC 'CODE'

ASSUME CS:CODE, DS:DATA, SS:STK

MOV AX,DATA

MOV DS,AX

XOR AX,AX

XOR BX,BX

MOV CX,N

M2: TEST A[BX],0000000000000001b

JNZ M1

ADD AX,A[BX]

M1: ADD BX,2

DEC CX

CMP CX,0

JG M2

MOV S,AX

MOV AX,4C00h

INT 21h

MAIN ENDP

CODE ENDS

END MAIN

1. Найти максимальный элемент массива

assume CS:code, DS:data *; инициализируем сегментные регистры - CS* *будет указывать на код, DS - на данные*

code segment *; начало сегмента кода*

start:

mov AX, data *; помещаем в AX указатель на начало сегмента данных*

mov DS, AX *; копируем адрес начала сегмента данных в регистр DS* *из регистра AX*

*; т.к. прямая запись указателя в регистр DS запрещена*

mov CX,5 *;сколько всего элементов*

mov BX,0 *;индекс*

mov AL,mas[0]

lap:

mov BH,mas[BX] *; и следующий за ним*

inc BX *; увеличиваем на 1*

cmp BH,AL *; сравниваем их*

jb Goe *; если первый больше второго то идем на GoE*

jmp NGoE *;если нет то начинаем сначала*

GoE:

mov AL,BH ;*и запоминаем элемпент в AL*

NGoE:

loop lap

int 21h *; вызываем прерывание DOS*

code ends *; конец сегмента кода*

data segment ; *начало сегмента данных*

mas db 1,2,3,4,8

*; Все строковые массивы должны заканчиваться символом ;*

data ends *; конец сегмента данных*

end start ; *конец программы*