МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Организация ЭВМ и систем»

Тема: Изучение режимов адресации в Intel8086

Студент гр. 9382 _____ Субботин М.О.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить режимы адресации, указать на ошибки в программе и объяснить их.

Основные теоретические положения.

Задание:

Лабораторная работа 2 предназначена для изучения режимов адресации, использует готовую программу lr2_comp.asm на Ассемблере, которая в автоматическом режиме выполняться не должна, так как не имеет самостоятельного функционального назначения, а только тестирует режимы адресации. Поэтому ее выполнение должно производиться под управлением отладчика в пошаговом режиме.

В программу введен ряд ошибок, которые необходимо объяснить в отчете по работе, а соответствующие команды закомментировать для прохождения трансляции. Необходимо составить протокол выполнения программы в пошаговом режиме отладчика по типу таблицы 1 предыдущей лабораторной работы и подписать его у преподавателя.

На защите студенты должны уметь объяснить результат выполнения каждой команды с учетом используемого вида адресации. Результаты, полученные с помощью отладчика, не являются объяснением, а только должны подтверждать ваши объяснения.

Исходный код программы:

```
; Программа изучения режимов адресации процессора IntelX86
EOL EQU '$'
ind EQU 2
n1 EQU 500
n2 EQU -50

; Стек программы
AStack SEGMENT STACK
    DW 12 DUP(?)
AStack ENDS

; Данные программы
DATA SEGMENT
; Директивы описания данных
mem1 DW 0
mem2 DW 0
mem3 DW 0
vec1 DB 1,2,3,4,8,7,6,5
vec2 DB -10,-20,10,20,-30,-40,30,40
matr DB 1,2,3,4,-4,-3,-2,-1,5,6,7,8,-8,-7,-6,-5
DATA ENDS

; Код программы
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:AStack

; Головная процедура
Main PROC FAR
    push DS
    sub AX,AX
    push AX
    mov AX,DATA
    mov DS,AX

; ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ НА УРОВНЕ СМЕЩЕНИЙ
; Регистровая адресация
    mov ax,n1
    mov cx,ax
    mov bl,EOL
    mov bh,n2
; Прямая адресация
    mov mem2,n2
    mov bx,OFFSET vec1 ;
    mov mem1,ax
; Косвенная адресация
    mov al,[bx]
    ;mov mem3,[bx]
; Базированная адресация
    mov al,[bx]+3
    mov cx,3[bx]

; Индексная адресация
    mov di,ind
    mov al,vec2[di]
    ;mov cx,vec2[di]

; Адресация с базированием и индексированием
    mov bx,3
    mov al,matr[bx][di]
    ;mov cx,matr[bx][di]
    ;mov ax,matr[bx*4][di]

; ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ С УЧЕТОМ СЕГМЕНТОВ ; Переопределение сегмента
; ------ вариант 1
    mov ax, SEG vec2
    mov es, ax
    mov ax, es:[bx]
    mov ax, 0
; ------ вариант 2
    mov es, ax
    push ds
    pop es
    mov cx, es:[bx-1]
    xchg cx,ax
; ------ вариант 3
    mov di,ind
    mov es:[bx+di],ax
; ------ вариант 4
    mov bp,sp
    ;mov ax,matr[bp+bx]
    ;mov ax,matr[bp+di+si]
; Использование сегмента стека push mem1
    push mem2
    mov bp,sp
    mov dx,[bp]+2
    ret 2
Main ENDP
CODE ENDS
    END Main
```

Экспериментальные результаты.

Листинг успешной трансляции программы:

```
Microsoft (R) Macro Assembler Version 5.10                  9/17/20 22:25:19
                                                             Page     1-1

                        ; [illegible]
                        ntelX86
 = 0024                     EOL EQU '$'
 = 0002                     ind EQU 2
 = 01F4                     n1 EQU 500
 =-0032                     n2 EQU -50

                        ; [illegible]
 0000                       AStack SEGMENT STACK
 0000  000C[                        DW 12 DUP(?)
        ????
              ]

 0018                       AStack ENDS

                        ; [illegible]
 0000                       DATA SEGMENT
                        ; [illegible]
 0000  0000                 mem1 DW 0
 0002  0000                 mem2 DW 0
 0004  0000                 mem3 DW 0
 0006  01 02 03 04 08 07        vec1 DB 1,2,3,4,8,7,6,5
       06 05
 000E  F6 EC 0A 14 E2 D8        vec2 DB -10,-20,10,20,-30,-40,30,40
       1E 28
 0016  01 02 03 04 FC FD        matr DB 1,2,3,4,-4,-3,-2,-1,5,6,7,8,-8,-7,-6,-5

       FE FF 05 06 07 08
       F8 F9 FA FB
 0026                       DATA ENDS

                        ; [illegible]
 0000                       CODE SEGMENT
                            ASSUME CS:CODE, DS:DATA, SS:AStack

                        ; [illegible]
 0000                       Main PROC FAR
 0000  1E                         push DS
 0001  2B C0                          sub AX,AX
 0003  50                         push AX
 0004  B8 ---- R                      mov AX,DATA
 0007  8E D8                          mov DS,AX

                        ; [illegible]
 0009  B8 01F4                        mov ax,n1
 000C  8B C8                          mov cx,ax
 000E  B3 24                          mov bl,EOL
 0010  B7 CE                          mov bh,n2
                        ; [illegible]
 0012  C7 06 0002 R FFCE                  mov mem2,n2
Microsoft (R) Macro Assembler Version 5.10                  9/17/20 22:25:19
                                                             Page     1-2

 0018  BB 0006 R                      mov bx,OFFSET vec1 ;
 001B  A3 0000 R                      mov mem1,ax
                        ; [illegible]
 001E  8A 07                          mov al,[bx]
                            ;mov mem3,[bx]
                        ; [illegible]
 0020  8A 47 03                       mov al,[bx]+3
 0023  8B 4F 03                       mov cx,3[bx]

                        ; [illegible]
 0026  BF 0002                        mov di,ind
 0029  8A 85 000E R                   mov al,vec2[di]
                            ;mov cx,vec2[di]

                        ; [illegible]
 002D  BB 0003                        mov bx,3
 0030  8A 81 0016 R                   mov al,matr[bx][di]
                            ;mov cx,matr[bx][di]
                            ;mov ax,matr[bx*4][di]

                        ; [illegible]
                        ; ------ [illegible] 1
 0034  B8 ---- R                      mov ax, SEG vec2
 0037  8E C0                          mov es, ax
 0039  26: 8B 07                      mov ax, es:[bx]
 003C  B8 0000                        mov ax, 0
                        ; ------ [illegible] 2
 003F  8E C0                          mov es, ax
 0041  1E                         push ds
 0042  07                         pop es
 0043  26: 8B 4F FF                   mov cx, es:[bx-1]
 0047  91                         xchg cx,ax
                        ; ------ [illegible] 3
 0048  BF 0002                        mov di,ind
 004B  26: 89 01                      mov es:[bx+di],ax
                        ; ------ [illegible] 4
 004E  8B EC                          mov bp,sp
                            ;mov ax,matr[bp+bx]
                            ;mov ax,matr[bp+di+si]
                        ; [illegible] push mem1
 0050  FF 36 0002 R                   push mem2
 0054  8B EC                          mov bp,sp
 0056  8B 56 02                       mov dx,[bp]+2
 0059  CA 0002                        ret 2
 005C                       Main ENDP
 005C                       CODE ENDS
                            END Main
Microsoft (R) Macro Assembler Version 5.10                  9/17/20 22:25:19
                                                             Symbols-1

Segments and Groups:

                N a m e             Length   Align   Combine Class

ASTACK . . . . . . . . . . . . . .  0018     PARA    STACK
CODE . . . . . . . . . . . . . . .  005C     PARA    NONE
DATA . . . . . . . . . . . . . . .  0026     PARA    NONE

Symbols:

                N a m e             Type     Value    Attr

EOL  . . . . . . . . . . . . . . .  NUMBER   0024

IND  . . . . . . . . . . . . . . .  NUMBER   0002

MAIN . . . . . . . . . . . . . . .  F PROC   0000     CODE     Length = 005C
MATR . . . . . . . . . . . . . . .  L BYTE   0016     DATA
MEM1 . . . . . . . . . . . . . . .  L WORD   0000     DATA
MEM2 . . . . . . . . . . . . . . .  L WORD   0002     DATA
MEM3 . . . . . . . . . . . . . . .  L WORD   0004     DATA

N1 . . . . . . . . . . . . . . . .  NUMBER   01F4
N2 . . . . . . . . . . . . . . . .  NUMBER   -0032

VEC1 . . . . . . . . . . . . . . .  L BYTE   0006     DATA
VEC2 . . . . . . . . . . . . . . .  L BYTE   000E     DATA

@CPU . . . . . . . . . . . . . . .  TEXT  0101h
@FILENAME  . . . . . . . . . . . .  TEXT  myhello
@VERSION . . . . . . . . . . . . .  TEXT  510


     89 Source  Lines
     89 Total   Lines
     19 Symbols

  47798 + 459462 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```

Были обнаружены и закомментированы 6 ошибок:

```
;mov mem3,[bx]
;mov cx,vec2[di]
;mov cx,matr[bx][di]
;mov ax,matr[bx*4][di]
;mov ax,matr[bp+bx]
;mov ax,matr[bp+di+si]
```

Обработка результатов эксперимента.

Mov mem3,[bx]

Ошибка: “Improper operand type”

Мы не можем прямо передавать объекты с памяти в память. Если мы все же хотим перетащить данные из ячейки [bx] в ячейку, на которую ссылается переменная mem3 то это следует делать через регистр.

Mov cx,vec2[di]

Ошибка: “Operand types must match”

Переменная vec2 – массив, и каждая его ячейка имеет тип DB т.е. занимает ровно 1байт.

В то же время регистр cx занимает 2 байта. Место, которое занимают операнды должно быть одинаковым. Мы могли бы передать vec2[di] в ch или cl, но не в cx.

Mov cx,matr[bx][di]

Ошибка: “Operand types must match”

То же самое, что и в прошлой ошибке. Ячейки двумерного массива имеют размерность 1 байт(DB), а регистр cx – 2 байта.

Mov ax,matr[bx*4][di]

Ошибка: “Illegal register value”

Операцию – умножение на число можно применять только к регистрам с префиксом e.

Выводы.

Изучил режимы адресации, указал на ошибки в программе и объяснил их.

ПРОТОКОЛ

Начальные значения регистров:

CS = 1A0A, DS=19F5, ES=19F5, SS=1A05

Адрес команды	Символический код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			До выполнения	После выполнения
0000	PUSH DS	1E	DS=19F5 SP=0018 STACK=+0|0000 IP=0000	DS=19F5 SP=0016 STACK=+0|19F5 IP=0001
0001	SUB AX,AX	2BC0	AX=0000 IP=0001	AX=0000 IP=0003
0003	PUSH AX	50	AX=0000 SP=0016 STACK=+0|19F5 IP=0003	AX=0000 SP=0014 STACK=+0|0000 +2|19F5 IP=0004
0004	MOV AX,1A07	B8071A	AX=0000 IP=0004	AX=1A07 IP=0007
0007	MOV DS,AX	8ED8	AX=1A07 DS=19F5 IP=0007	AX=1A07 DS=1A07 IP=0009
0009	MOV AX,01F4	B8F401	AX=1A07 IP=0009	AX=01F4 IP=000C
000C	MOV CX,AX	8BC8	AX=01F4 CX=00AC IP=000C	AX=01F4 CX=01F4 IP=000E

000E	MOV BL,24	B324	BX=0000 IP=000E	BX=0024 IP=0010
0010	MOV BH,CE	B7CE	BX=0024 IP=0010	BX=CE24 IP=0012
0012	MOV [0002],FFCE	C7060200C EFF	IP=0012 DS[0002]=00 DS[0003]=00	IP=0018 DS[0002]=CE DS[0003]=FF
0018	MOV BX, 0006	BB0600	BX=CE24 IP=0018	BX=0006 IP=001B
001B	MOV [0000],AX	A30000	AX=01F4 IP=001B DS[0000]=00 DS[0001]=00	AX=01F4 IP=001E DS[0000]=F4 DS[0001]=01
001E	MOV AL,[BX]	8A07	AX=01F4 DS[BX]= DS[0006]=01 IP=001E	AX=0101 DS[BX]= DS[0006]=01 IP=0020
0020	MOV AL,[BX+03]	8A4703	AX=0101 DS[BX+03]= DS[0009]=04 IP=0020	AX=0104 DS[BX+03]= DS[0009]=04 IP=0023
0023	MOV CX,[BX+03]	8B4F03	CX=01F4 DS[BX+03]= DS[0009]=04 DS[000A]=08 IP=0023	CX=0804 DS[BX+03]= DS[0009]=04 DS[000A]=08 IP=0026
0026	MOV DI, 0002	BF0200	DI=0000 IP=0026	DI=0002 IP=0029
0029	MOV AL,[000E+DI]	8A850E00	AX=0104	AX=010A

			DS[000E+DI]= DS[0010]=0A IP=0029	DS[000E+DI]= DS[0010]=0A IP=002D
002D	MOV BX,0003	BB0300	BX=0006 IP=002D	BX=0003 IP=0030
0030	MOV AL,[0016 + BX + DI]	8A811600	AX=010A DS[0016+BX+DI]= DS[001B]=FD IP=0030	AX=01FD DS[0016+BX+DI]= DS[001B]=FD IP=0034
0034	MOV AX,1A07	B8071A	AX=01FD IP=0034	AX=1A07 IP=0037
0037	MOV ES,AX	8EC0	ES=19F5 AX=1A07 IP=0037	ES=1A07 AX=1A07 IP=0039
0039	MOV AX,ES:[BX]	268B07	AX=1A07 ES=1A07 ES[BX]=ES[0003]=FF ES[0004]=00 IP=0039	AX=00FF ES=1A07 ES[BX]=ES[0003]=FF ES[0004]=00 IP=003C
003C	MOV AX,0000	B80000	AX=00FF IP=003C	AX=0000 IP=003F
003F	MOV ES,AX	8EC0	ES=1A07 AX=0000 IP=003F	ES=0000 AX=0000 IP=0041
0041	PUSH DS	1E	DS=1A07 SP=0014 STACK=+0|0000 +2|19F5	DS=1A07 SP=0012 STACK=+0|1A07 +2|0000

			IP=0041	+4|19F5 IP=0042
0042	POP ES	07	SP=0012 ES=0000 STACK=+0|1A07 +2|0000 +4|19F5 IP=0042	SP=0014 ES=1A07 STACK=+0|0000 +2|19F5 IP=0043
0043	MOV CX,ES:[BX-01]	268B4FFF	CX=0804 ES=1A07 ES[BX-01]= ES[0002]=CE ES[0003]=FF IP=0043	CX=FFCE ES=1A07 ES[BX-01] =ES[0002]=CE ES[0003]=FF IP=0047
0047	XCHG AX,CX	91	AX = 0000 CX = FFCE IP=0047	AX=FFCE CX=0000 IP=0048
0048	MOV DI,0002	BF0200	DI=0002 IP=0048	DI=0002 IP=004B
004B	MOV ES:[BX+DI],AX	268901	ES=1A07 ES[BX+DI] = [0005]= 00 ES[0006] = 01 AX=FFCE IP=004B	ES=1A07 ES[0005] = CE ES[0006] = FF IP=004E
004E	MOV BP,SP	8BEC	BP=0000 SP=0014 IP=004E	BP=0014 SP=0014 IP=0050
0050	PUSH [0002]	FF360200	DS[0002] = CE	DS[0002]=CE

			DS[0003] = FF SP = 0014 STACK = +0|0000 +2|19F5 IP=0050	DS[0003] = FF SP = 0012 STACK=+0|FFCE +2|0000 +4|19F5 IP=0054
0054	MOV BP,SP	8BEC	SP=0012 BP=0014 IP=0054	SP=0012 BP=0012 IP=0056
0056	MOV DX,[BP+02]	8B5602	DX=0000 SS[BP+02] SS[0014]=00 SS[0015]=00 IP=0056	DX=0000 SS[BP+02] SS[0014]=00 SS[0015] = 00 IP=0059
0059	RET FAR 0002	CA0200	CS=1A0A SP=0012 IP=0059 STACK=+0|FFCE +2|0000 +4|19F5	CS=0000 SP=0018 IP=FFCE STACK=+0|0000