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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Организация ЭВМ и систем»
Тема: Изучение режимов адресации

Студент гр. 9382 _____ Савельев И.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить режимы адресации в ассемблере и формирование исполняемого кода на практике.

Задание.

Вариант 8

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.Порядок выполнения работы.

1. Получить у преподавателя вариант набора значений исходных данных (массивов) vec1, vec2 и matr из файла lr2.dat, приведенного в каталоге Задания и внести свои данные вместо значений, указанных в приведенной ниже программе.

2. Протранслировать программу с созданием файла диагностических сообщений; объяснить обнаруженные ошибки и закомментировать соответствующие операторы в тексте программы.

3. Снова протранслировать программу и скомпоновать загрузочный модуль.

4. Выполнить программу в пошаговом режиме под управлением отладчика с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения команды.

5. Результаты прогона программы под управлением отладчика должны быть подписаны преподавателем и представлены в отчете.

Пример используемой программы приведен ниже.; Программа изучения режимов адресации процессора Intel X86

Исходный код программы.

```
EOL EQU '$'
ind EQU 2
n1 EQU 500
n2 EQU -50

AStack SEGMENT STACK
DW 12 DUP(?)
AStack ENDS

DATA SEGMENT

mem1 DW 0
mem2 DW 0
mem3 DW 0
vec1 DB 28,27,26,25,21,22,23,24
vec2 DB 20,30,-20,-30,40,50,-40,-50
matr DB -8,-7,3,4,-6,-5,1,2,-4,-3,7,8,-2,-1,5,6
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA, SS:AStack

Main PROC FAR
    push DS
    sub AX,AX
    push AX
    mov AX,DATA
    mov DS,AX
; Р е г и с т р о в а я
    mov ax,n1
    mov cx,ax
    mov bl,EOL
    mov bh,n2
; П р я м а я
```

```
        mov mem2,n2
        mov bx,OFFSET vec1
        mov mem1,ax
    ; К о с в е н н а я
        mov al,[bx]
        ;mov mem3,[bx]
    ; Б а з и р о в а н н а я
        mov al,[bx]+3
        mov cx,3[bx]
    ; И н д е к с н а я
        mov di,ind
        mov al,vec2[di]
        ;mov cx,vec2[di]
    ; С  б а з и р о в а н и е м  и  и н д е к с и р о в а н и е м
        mov bx,3
        mov al,matr[bx][di]
        ;mov cx,matr[bx][di]
        ;mov ax,matr[bx*4][di]
    ; В а р  1
        mov ax, SEG vec2
        mov es, ax
        mov ax, es:[bx]
        mov ax, 0
    ; В а р  2
        mov es, ax
        push ds
        pop es
        mov cx, es:[bx-1]
        xchg cx,ax
    ; В а р  3
        mov di,ind
        mov es:[bx+di],ax
    ; В а р  4
        mov bp,sp
        mov ax,matr[bp+bx]
        mov ax,matr[bp+di+si]

        push mem1
        push mem2
        mov bp,sp
        mov dx,[bp]+2
        ret 2
    Main ENDP
    CODE ENDS
        END Main
```

Листинг исправленной программы.

```
    Microsoft (R) Macro Assembler Version 5.10
10/12/20 22:10:2
```

```
= 0024                  EOL EQU '$'
= 0002                  ind EQU 2
= 01F4                  n1 EQU 500
=-0032                  n2 EQU -50

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

0018                AStack ENDS

0000                DATA SEGMENT

0000  0000          mem1 DW 0
0002  0000          mem2 DW 0
0004  0000          mem3 DW 0
0006  1C 1B 1A 19 15 16    vec1 DB 28,27,26,25,21,22,23,24
   17 18
000E  14 1E EC E2 28 32    vec2 DB 20,30,-20,-30,40,50,-40,-50
   D8 CE
  0016       F8    F9    03    04    FA    FB             matr    DB
-8,-7,3,4,-6,-5,1,2,-4,-3,7,8,-2,-1,5,6
   01 02 FC FD 07 08
   FE FF 05 06
0026                DATA ENDS

0000                CODE SEGMENT
              ASSUME CS:CODE, DS:DATA, SS:AStack

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
               ; Р е г и с т р о в а я
0009  B8 01F4                 mov ax,n1
000C  8B C8              mov cx,ax
000E  B3 24              mov bl,EOL
0010  B7 CE              mov bh,n2
               ; П р я м а я
0012  C7 06 0002 R FFCE       mov mem2,n2
0018  BB 0006 R          mov bx,OFFSET vec1
001B  A3 0000 R          mov mem1,ax
               ; К о с в е н н а я
001E  8A 07              mov al,[bx]
```

```
                        ;mov mem3,[bx]
                    ; Базированная
0020  8A 47 03                mov al,[bx]+3
0023  8B 4F 03                mov cx,3[bx]
                    ; Индексная
0026  BF 0002                 mov di,ind
0029  8A 85 000E R            mov al,vec2[di]
                        ;mov cx,vec2[di]
      Microsoft (R) Macro Assembler Version 5.10
10/12/20 22:10:2
                                                                Page
1-2


                    ; С базированием и индекси
                    рованием
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]
                    ; Вар 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
                    ; Вар 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
                    ; Вар 3
0048  BF 0002                  mov di,ind
004B  26: 89 01           mov es:[bx+di],ax
                    ; Вар 4
004E  8B EC           mov bp,sp
                    ; mov ax,matr[bp+bx]
                    ; mov ax,matr[bp+di+si]

                    ; push mem1
                   ; push mem2
0058  8B EC               mov bp,sp
005A  8B 56 02                 mov dx,[bp]+2
005D  CA 0002                  ret 2
0060                  Main ENDP
0060                  CODE ENDS
                     END Main
      Microsoft (R) Macro Assembler Version 5.10
10/12/20 22:10:2

Symbols-1
```

```
       Segments and Groups:

                N a m e                 Length     Align    Combine
Class

       ASTACK . . . . . .               0018    PARA    STACK
       CODE . . . . . .                 0060    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 = 0060
       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  lab2
       @VERSION . . . . . .             TEXT  510


           79 Source  Lines
           79 Total   Lines
           19 Symbols

         47820 + 459440 Bytes symbol space free

           0 Warning Errors
           0 Severe  Errors
```

Обработка результатов эксперимента.

Были закомментированы 7 ошибок.

mov mem3,[bx]

error A2052: Improper operand type

В ассемблере нельзя с помощью команды mov считать объекты из одного участка памяти и записать в другой. Для этого нужно использовать 2 команды mov и использовать регистры общего назначения, например AX.

mov cx,vec2[di]

warning A4031: Operand types must match

Регистр cx занимает в памяти два байта, а переменная vec2 является массивом каждый элемент которого занимает в памяти один байт. Переменные между которыми происходит обмен данных должны занимать в памяти одинаковое количество места.

mov cx,matr[bx][di]

warning A4031: Operand types must match

Тоже предупреждение что и выше о не соответствие размерности операнд.

cx 2 байта, а matr[bx][di] один байт.

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

error A2055: Illegal register value

Нельзя умножать 16-битные регистры

mov ax, matr[bp+bx]

error A2046 Multiple base registers

Нельзя использовать более одного регистра

mov ax,matr[bp+di+si]

error A2046 Multiple base registers

Нельзя использовать более одного регистра

push mem1

push mem2

Семантическая ошибка, для завершения программы и возвращения в DOS необходимо, чтобы вершина стека содержала смещение и сегмент начала PSP.

А если эти команды выполняться, то вершина стека содержит mem2 и mem1 и при выполнении команды ret 2 управление перейдет на адрес meml:mem2 и программа не сможет корректно завершится.

Протокол работы на компьютере.

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	SP=0018 STACK=+0|0000 IP=0000	SP=0016 STACK=+0| 30B0 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	DS,AX	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=00A8 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	C7060200CE FF	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=011C DS[BX]= DS[0006]=01 IP=0020
0020	MOV AL,[BX+03]	8A4703	AX=011C DS[BX+03]= DS[0009]=19 IP=0020	AX=0119 DS[BX+03]= DS[0009]=19 IP=0023
0023	MOV CX,[BX+03]	8B4F03	CX=01F4 DS[BX+03]= DS[0009]=19 DS[000A]=15 IP=0023	CX=0804 DS[BX+03]= DS[0009]=19 DS[000A]=15 IP=0026
0026	MOV Dl,0002	BF0200	DI=0000 IP=0026	DI=0002 IP=0029
0029	MOV AL,[000E+DI]	8A850E00	AX=0119 DS[000E+DI]= DS[0010]=EC IP=0029	AX=01EC DS[000E+DI]= DS[0010]=EC IP=002D
002D	MOV BX,0003	BB0300	BX=0006 IP=002D	BX=0003 IP=0030
0030	MOV AL,[0016+BX+DI]	8A811600	ES=19F5 AX=01EC IP=0037	ES=19F5 AX=01FB IP=0039
0034	MOV AX, 1A07	B8C230	AX=01FB ES=19F5 ES[BX]=ES[000 3]=FF	AX=1A07 ES=19F5 ES[BX]=ES[0003]=FF

			ES[0004]=00 IP=0039	ES[0004]=00 IP=003C
0037	MOV ES,AX	8EC0	AX=1A07 IP=003C	AX=1A07 IP=003F
0039	MOV AX,ES:[BX]	268B07	ES=1A07 AX=1A07 IP=003F	ES=1A07 AX=00FF IP=0041
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 IP=0041	DS=1A07 SP=0012 STACK=+0|1A07 +2|0000 +4|19F55 IP=0042
0042	POP ES	07	SP=0012 ES=0000 STACK=+0|1A0 7 +2|0000 +4|19F55 IP=0042	SP=0014 ES=1A07 STACK=+0|0000 +2|19F55 IP=0043
0043	MOV CS,ES:[BX-01]	268B4FFF	CX=1519 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	8BEC	DI=0002 IP=0048	DI=0002 IP=004B
004B	MOV ES:[BX+DI], AX	FF360000	ES=1A07 ES[BX+DI] = [0005]= 00 ES[0006] = 01 AX=FFCE IP=004B	ES=1A07 ES[0005] = CE ES[0006] = FF AX=FFCE IP=004E

004E	MOV BP,SP	FF360200	DS[0002] = CE DS[0003] = FF SP = 0014 IP=0054	DS[0002] = CE DS[0003] = FF SP = 0014 IP=0058
0050	MOV BP,SP	8BEC	SP=0014 BP=0014 IP=0058	SP=0014 BP=0014 IP=005A
0052	MOV DX,[BP+02]	8B5602	DX=0000 SS[BP+02] = SS[0012]=F4 SS[0013]=01 IP=005A	DX=19F5 SS[BP+02] = SS[0012]=F4 SS[0013] = 01 IP=005D
0055	RET FAR 0002	CA0200	CS=1A0A SP=0014 IP=005D STACK=+0|0000 +2|19F5	CS=1A0A SP=0018 IP=0000 STACK=+0|7244 +2|0000

Вывод.

В результате выполнения лабораторной работы были изучены режимы адресации в ассемблере и получены навыки по отладке программ.