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 и формирования исполнительного адреса

Студент гр. 1303 _____ Жилин И.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Целью лабораторной работы №2 является изучение различных режимов адресации в Ассемблере.

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.

Выполнение работы.

1. Выбран 8-й вариант из набора значений исходных данных (массивов) vec1, vec2 и matr из файла lr2.dat, полученного преподавателем, соответствующие данные были внесены вместо значений, указанных в приведенной ниже программе.
2. Протранслирована программа с созданием файла диагностических сообщений, объяснены обнаруженные ошибки и закомментированы соответствующие операторы в тексте программы.

   mov mem3, [bx] – машинные команды могут манипулировать одновременно двумя операндами, находящимися в оперативной памяти, то есть в команде только 1 операнд может указывать на ячейку памяти, другой операнд должен быть либо регистром, либо непосредственным значением.

*mov ax, matr[bx*4][di]* – попытка применить два разных вида адресации сразу: косвенную индексную адресацию и косвенную базовую индексную адресацию.

mov ax, matr[bp+bx] – в косвенной адресации с индексированием исполнительный адрес берется в виде суммы адресов, находящихся в базовом и индексном регистрах, а в данной строке оба регистра базовые.

mov ax, matr[bp+di+si] – в непосредственной адресации с базированием и индексированием берется сумма базового и индексного регистра, к которым добавляется непосредственно фигурирующее в команде смещение, а в данной строке фигурируют 2 индексных регистра и 1 базовый.

3. Снова протранслирована программа и скомпонован загрузочный модуль.
4. Программа выполнена в пошаговом режиме под управлением отладчика с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения команды.

Таблица 1 – Ход выполнения L2.EXE

Адрес команды	Символический код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			До выполнения	После выполне-ния
0000	PUSH DS	1E	IP = 0000 SP = 0018 STACK(+0)=0000	IP = 0001 SP = 0016 STACK(+0)=19F5
0001	SUB AX, AX	2BC0	IP = 0001 AX = 0000	IP = 0003 AX = 0000
0003	PUSH AX	50	IP = 0003 SP = 0016 STACK(+0)=19F5 STACK(+2)=0000	IP = 0004 SP = 0014 STACK(+0)=0000 STACK(+2)=19F5
0004	MOV AX, 1A07	B8071A	IP = 0004 AX = 0000	IP = 0007 AX = 1A07
0007	MOV DS, AX	8ED8	IP = 0007 DS = 19F5	IP = 0009 DS = 1A07

0009	MOV AX, 01F4	B8F401	IP = 0009 AX = 1A07	IP = 000C AX = 01F4
000C	MOV CX, AX	8BC8	IP = 000C CX = 00B0	IP = 000E CX = 01F4
000E	MOV BL, 24	B324	IP = 000E BX = 0000	IP = 0010 BX = 0024
0010	MOV BH, CE	B7CE	IP = 0010 BX = 0024	IP = 0012 BX = CE24
0012	MOV [0002], FFCE	C7060200-CEFF	IP = 0012	IP = 0018
0018	MOV BX, 0006	BB0600	IP = 0018 BX = CE24	IP = 001B BX = 0006
001B	MOV [0000], AX	A30000	IP = 001B	IP = 001E
001E	MOV AL, [BX]	8A07	IP = 001E AX = 01F4	IP = 0020 AX = 011C
0020	MOV AL, [BX+3]	8A4703	IP = 0020 AX = 011C	IP = 0023 AX = 0119
0023	MOV CX, [BX+3]	8B4F03	IP = 0023 CX = 01F4	IP = 0026 CX = 1519
0026	MOV DI, 0002	BF0200	IP = 0026 DI = 0000	IP = 0029 DI = 0002
0029	MOV AL, [000E+DI]	8A850E00	IP = 0029 AX = 0119	IP = 002D AX = 01EC
002D	MOV CX, [000E+DI]	8B8D0E00	IP = 002D CX = 1519	IP = 0031 CX = E2EC
0031	MOV BX, 0003	BB0300	IP = 0031 BX = 0006	IP = 0034 BX = 0003
0034	MOV AL, [0016+BX+DI]	8A811600	IP = 0034 AX = 01EC	IP = 0038 AX = 01FB
0038	MOV CX, [0016+BX+DI]	8B891600	IP = 0038 CX = E2EC	IP = 003C CX = 01FB
003C	MOV AX, 1A07	B8071A	IP = 003C AX = 01FB	IP = 003F AX = 1A07
003F	MOV ES, AX	8EC0	IP = 003F ES = 19F5	IP = 0041 ES = 1A07
0041	MOV AX, ES:[BX]	268B07	IP = 0041 AX = 1A07	IP = 0044 AX = 00FF
0044	MOV AX, 0000	B80000	IP = 0044 AX = 00FF	IP = 0047 AX = 0000
0047	MOV ES, AX	8EC0	IP = 0047 ES = 1A07	IP = 0049 ES = 0000
0049	PUSH DS	1E	IP = 0049 SP = 0014 STACK(+0)=0000 STACK(+2)=19F5 STACK(+4)=0000	IP = 004A SP = 0012 STACK(+0)=1A07 STACK(+2)=0000 STACK(+4)=19F5

004A	POP ES	07	IP = 004A SP = 0012 ES = 0000 STACK(+0)=1A07 STACK(+2)=0000 STACK(+4)=19F5	IP = 004B SP = 0014 ES = 1A07 STACK(+0)=0000 STACK(+2)=19F5 STACK(+4)=0000
004B	MOV CX, ES:[BX-01]	268B4FFF	IP = 004B CX = 01FB	IP = 004F CX = FFCE
004F	XCHG AX, CX	91	IP = 004F AX = 0000 CX = FFCE	IP = 0050 AX = FFCE CX = 0000
0050	MOV DI, 0002	BF0200	IP = 0050 DI = 0002	IP = 0053 DI = 0002
0053	MOV ES:[BX+DI], AX	268901	IP = 0053 ES = 1A07	IP = 0056 ES = 1A07
0056	MOV BP, SP	8BEC	IP = 0056 BP = 0000	IP = 0058 BP = 0014
0058	PUSH [0000]	FF360000	IP = 0058 SP = 0014 STACK(+0)=0000 STACK(+2)=19F5 STACK(+4)=0000	IP = 005C SP = 0012 STACK(+0)=01F4 STACK(+2)=0000 STACK(+4)=19F5
005C	PUSH [0002]	FF360200	IP = 005C SP = 0012 STACK(+0)=01F4 STACK(+2)=0000 STACK(+4)=19F5 STACK(+6)=0000	IP = 0060 SP = 0010 STACK(+0)=FFCE STACK(+2)=01F4 STACK(+4)=0000 STACK(+6)=19F5
0060	MOV BP, SP	8BEC	IP = 0060 BP = 0014	IP = 0062 BP = 0010
0062	MOV DX, [BP+02]	8B5602	IP = 0062 DX = 0000	IP = 0065 DX = 01F4
0065	RET Far 0002	CA0200	IP = 0065 SP = 0010 CS = 1A0A STACK(+0)=FFCE STACK(+2)=01F4 STACK(+4)=0000 STACK(+6)=19F5	IP = FFCE SP = 0016 CS = 01F4 STACK(+0)=19F5 STACK(+2)=0000 STACK(+4)=0000 STACK(+6)=0000

Выводы.

В ходе выполнения лабораторной работы №2 были изучены различные режимы адресации в Ассемблере.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: *L2.asm*

```
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 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

; 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
```