

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

Хулап О.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Изучить режимы адресации и формирования исполнительного адреса

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.

Ход работы.

1. Изменение набора значений исходных данных (массивов) `vec1`, `vec2` и `matr` из файла `lr2.dat`, согласно своему варианту.
2. Трансляция программы с созданием файла диагностических сообщений. Объяснение обнаруженных ошибок и предупреждений и закомментирование операторов с ошибками в тексте программы.

❶ Ошибка l2.asm(41): error A2052: Improper operand type (Неверный тип операнда)

Строка: `mov mem3,[bx]`

Нельзя одновременно читать из памяти и писать в память. Нужно сначала перенести данные из памяти в регистр, а уже потом из регистра в необходимый сегмент.

❶ Предупреждение l2.asm(48): warning A4031: Operand types must match (Несоответствие типов операндов)

Строка: `mov cx, vec2[di]`

Типы операндов должны совпадать, а в данном случае, `cx` – 1 слово, элемент `vec2` – 1 байт.

- Предупреждение l2.asm(52): warning A4031: Operand types must match (Несоответствие типов операндов)

Строка: `mov cx, matr[bx][di]`

Типы операндов должны совпадать, а в данном случае, `cx` – 1 слово, элемент `matr` – 1 байт.

- Ошибка l2.asm(53): error A2055: Illegal register value (Незаконное использование регистра)

Строка: `mov ax, matr[bx*4][di]`

В данном случае используется базово-индексная адресация. В таких случаях в регистре хранится адрес начала структуры данных, а доступ осуществляется к какому-нибудь элементу этой структуры. При данном типе адресации надо сначала изменить значение регистра, а уже потом переводить информацию.

- Ошибка l2.asm(72): error A2046: Multiple base registers (несколько базовых регистров)

Строка: `mov ax,matr[bp+bx]`

Регистры `bp` и `bx` базовые, поэтому сначала складываются значения регистров, а уже затем данные передается указателю одного из регистров. Таким образом, сначала нужно в регистр `bp` занести общую сумму, а потом производить смещение.

- Ошибка l2.asm(73): error A2047: Multiple index registers (несколько индексных регистров)

Строка: `mov ax,matr[bp+di+si]`

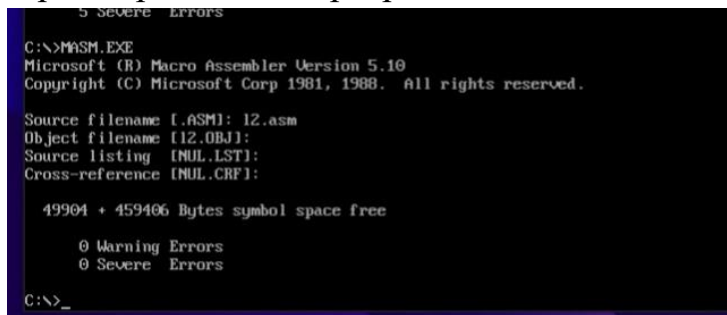
Регистры `di` и `si` индексные, поэтому сначала складываются их значения, а потом данные передаются указателю из одного регистра. Сначала в регистр `di` заносится общая сумма, а потом производится смещение.

- Ошибка l2.asm(80): error A2006: Phase error between passes

Строка: `Main ENDP`

Данная ошибка свидетельствует о том, что в функции `main` содержатся ошибки.

3. Повторная трансляция программы и компоновка загрузочного модуля.



```
5 Severe Errors
C:\>MASM.EXE
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

Source filename [LASM1]: l2.asm
Object filename [L2.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

49904 + 459406 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>_
```

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

Изначальные значения:

SP = 0018

IP = 0000

DS = 19F5

CX = 00B0

Адрес команды	Символьный код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			До выполнения	После выполнения
0000	PUSH DS	1E	(SP)=0018 (IP)=0000 STACK +0 = 0000	(SP)=0016 (IP)=0001 STACK +0 = 19F5
0001	SUB AX, AX	2BC0	(IP)=0001	(IP)=0003
0003	PUSH AX	50	(SP)= 0016 (IP)= 0003 STACK +0 = 19F5 STACK +2 = 0000	(SP)= 0014 (IP)= 0004 STACK +0 = 0000 STACK +2 = 19F5
0004	MOV AX, 1A07	B8071A	(AX)=0000 (IP)=0004	(AX)= 1A07 (IP)=0007
0007	MOV DS,AX	8ED8	(DS)=19F5 (IP)= 0007	(DS)=1A07 (IP)= 0009
0009	MOV AX, 01F4	B8F401	(AX)=1A07 (IP)=0009	(AX)= 01F4 (IP)= 000C
000C	MOV CX,AX	8BC8	(IP)=000C (CX)=00B0	(IP)=000E (CX)=01F4
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	C7060200CEF F	(IP)=0012	(IP)=0018

0018	MOV BX,0006	BB0600	(BX)=CE24 (IP)=0018	(BX)=0006 (IP)=001B
001B	MOV [0000],AX	A30000	(IP)=001B	(IP)=001E
001E	MOV AL,[BX]	8A07	(AX)=01F4 (IP)=001E	(AX)=010B (IP)=0020
0020	MOV AL,[BX+03]	8A4703	(IP) = 0020 (AX) = 010B	(IP)= 0023 (AX) = 010E
0023	MOV CX, [BX+03]	8B4F03	(CX) = 01F4 (IP) = 0023	(CX) = 120E (IP) = 0026
0026	MOV DI, 0002	BF0200	(DI) = 0000 (IP) = 0026	(DI) = 0002 (IP) = 0029
0029	MOV AL, [DI+ 000E]	8A850E00	(AX) = 010E (IP) = 0029	(AX)= 01F6 (IP)= 002D
002D	MOV BX, 0003	BB03000	(IP) = 002D (BX) = 0006	(IP) = 0030 (BX) = 0003
0030	MOV AL, [BX+DI+0016]	8A811600	(IP) = 0030 (AX) = 01F6	(IP) = 0034 (AX) = 0104
0034	MOV AX, 11 AE	B8AE11	(AX) = 0104 (IP)= 0034	(AX) = 1A07 (IP)= 0037
0037	MOV ES, AX	8EC0	(ES) = 19F5 (IP)= 0037	(ES) = 1A07 (IP)= 0039
0039	MOV AX, ES:[BX]	268B07	(AX) = 1A07 (IP) = 0039	(AX)= 00FF (IP) = 003C
003C	MOV AX, 0000	B80000	(AX)= 00FF (IP)= 003C	(AX)=0000 (IP) = 003F
003F	MOV ES, AX	8EC0	(ES) = 11AE (IP)= 003F	(ES)= 0000 (IP)= 0041

0041	PUSH DS	1E	(IP)= 0041 (SP)= 0014 STACK +0 = 0000 STACK +2 = 19F5 STACK +4 =0000	(IP)= 0042 (SP)= 0012 STACK +0 = 1A07 STACK +2 = 0000 STACK +4 =19F5
0042	POP ES	07	(SP)= 0012 (ES)=0000 (IP)= 0042 STACK +0 = 1A07 STACK +2 = 0000 STACK +4 =19F5	(SP) = 0014 (ES)=1A07 (IP)= 0043 STACK +0 = 0000 STACK +2 = 19F5 STACK +4 =0000
0043	MOV CX, ES:[BX—01]	268B4FFF	(CX) = 120E (IP) = 0043	(CX)= FFCE (IP)= 0047
0047	XCHG AX, CX	91	(AX) = 0000 (CX) = FFCE (IP)=0047	(AX) = FFCE (CX) = 0000 (IP)=0048
0048	MOV DI, 0002	BF0200	(IP) = 0048	(IP) = 004B
004B	MOV ES:[BX+DI], AX	268901	(IP) = 004B	(IP) = 004E
004E	MOV BP, SP	8BEC	(IP) = 004E (BP) = 0010	(IP) = 0050 (BP) = 0014
0050	PUSH [0000]	FF360000	(IP) = 0050 (SP)=0014 STACK +0 = 0000 STACK +2 = 19F5 STACK +4 =0000	(IP) = 0054 (SP)=0012 STACK +0 = 01F4 STACK +2 = 0000 STACK +4 =19F5

0054	PUSH [0002]	FF360200	(IP) = 0054 (SP) = 0012 STACK +0 = 01F4 STACK +2 = 0000 STACK +4 = 19F5 STACK + 6 = 0000	(IP) = 0058 (SP) = 0010 STACK +0 = FFCE STACK +2 = 01F4 STACK +4 = 0000 STACK + 6 = 19F5
0058	MOV BP, SP	8BEC	(IP) = 0058 (BP) = 0014	(IP) = 005A (BP) = 0010
005A	MOX DX, [BP+02]	8B5602	(IP) = 005A (DX) = 0000	(IP) = 005D (DX) = 01F4
005D	RET Far	CB	(IP) = 005D (SP) = 0010 (CS)=1A0A STACK +2 = 01F4 STACK +4 = 0000 STACK + 6 = 19F5	(IP) = FFCE (SP)= 0014 (CS)=01F4 STACK +2 = 19F5 STACK +4 = 0000 STACK + 6 = 0000

Выводы.

В ходе выполнения лабораторной работы были изучены основные режимы адресации памяти.

Приложение А. Код программы l2.asm

; 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 11,12,13,14,18,17,16,15
vec2 DB 10,20,-10,-20,30,40,-30,-40
matr DB 1,2,-4,-3,3,4,-2,-1,5,6,-8,-7,7,8,-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

```

**Приложение Б. Листинг успешной трансляции
программы с закомментированными ошибочными
операторами**