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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Организация ЭВМ и систем»

Тема: Изучение режимов адресации и формирования исполнительного адреса

Студент гр. 9383 _____ Арутюнян С.Н.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Содержание

1. Цель работы

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 занести 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.

2. Текст программы lab2.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 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 (ds - смещение по умолчанию или ss если регистры esp ebp bp)
    mov al, [bx]
    ; mov mem3, [bx] ошибка! в x86 нельзя напрямую переносить данные из памяти в память

    ; Базированная адресация
    mov al, [bx] + 3
    mov cx, 3[bx]
```

```
    ; Индексная адресация (допускается использование только si и di в кач-ве индексов)
    mov di, ind
    mov al, vec2[di]
    mov cx, vec2[di] ; WARNING: OPERAND TYPES MUST MATCH

    ; Адресация с базированием и индексированием
    mov bx, 3
    mov al, matr[bx][di]
    mov cx, matr[bx][di] ; WARNING: OPERAND TYPES MUST MATCH
    ; mov ax, matr[bx*4][di]  ; ошибка! адресация только в правой паре кв. скобок

    ; 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
```

3. Обнаруженные ошибки

Ошибка 1. mov mem3, [bx]

Проблема этой инструкции в том, что она пытается перемещать данные напрямую из памяти в память. В архитектуре x86 это запрещено. Необходимо для этого использовать промежуточные регистры.

Ошибка 2. mov ax, matr[bx*4][di]

Проблема этой инструкции в том, что масштабирование допускается только во второй паре квадратных скобок.

Ошибка 3. mov ax, matr[bp+bx]

В качестве смещения разрешено использовать только регистры DI и SI, поэтому данная инструкция ошибочна.

Ошибка 4. mov ax, matr[bp+di+si]

В качестве смещения разрешено использовать только один регистр, поэтому данная инструкция ошибочна.

4. Протокол выполнения программы

Начальное содержимое сегментных регистров: (SP) = 0018, (SS) = 1A05, (IP) = 0000, (CX) = 00B8, (CS) = 1A01, (DS) = 19F5, (ES) = 19F5.

Таблица 1. Протокол выполнения программы hello1.asm

Адрес команды	Символический код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			До выполнения	После выполнения
0000	push ds	1E	Начальное содержимое	(SP) = 0016 (STACK + 0) = 19F5 (IP) = 0001
0001	sub ax, ax	2BC0	(SP) = 0016 (STACK + 0) = 19F5 (IP) = 0001 (AX) = 0000	(SP) = 0016 (STACK + 0) = 19F5 (IP) = 0003 (AX) = 0000
0003	push ax	50	(SP) = 0016 (STACK + 0) = 19F5 (IP) = 0003	(SP) = 0014 (STACK + 0) = 0000 (STACK + 2) = 19F5 (IP) = 0004
0004	mov ax, DATA	B8071A	(AX) = 0000 (IP) = 0004	(AX) = 1A07 (IP) = 0007
0007	mov ds, ax	8ED8	(IP) = 0007 (DS) = 19F5	(IP) = 0009 (DS) = 1A07
0009	mov ax, n1	B8F401	(IP) = 0009 (AX) = 1A07	(IP) = 000C (AX) = 01F4
000C	mov cx, ax	8BC8	(IP) = 000C (CX) = 00B8	(IP) = 000E (CX) = 01F4

000E	mov bl, EOL	B324	(IP) = 000E (BX) = 0000	(IP) = 0010 (BX) = 0024
0010	mov bh, n2	B7CE	(IP) = 0010 (BX) = 0024	(IP) = 0012 (BX) = CE24
0012	mov mem2, n2	C7060200 CEFF	(IP) = 0012	(IP) = 0018
0018	mov bx, OFFSET vec1	BB0600	(IP) = 0018 (BX) = CE24	(IP) = 001B (BX) = 0006
001B	mov mem1, ax	A30000	(IP) = 001B	(IP) = 001E
001E	mov al, [bx]	8A07	(IP) = 001E (AX) = 01F4	(IP) = 0020 (AX) = 0101
0020	mov al, [bx] + 3	8A4703	(IP) = 0020 (AX) = 0101	(IP) = 0023 (AX) = 0104
0023	mov cx, 3[bx]	8B4F03	(IP) = 0023 (CX) = 01F4	(IP) = 0026 (CX) = 0804
0026	mov di, ind	BF0200	(IP) = 0026 (DI) = 0000	(IP) = 0029 (DI) = 0002
0029	mov al, vec2[di]	8A850E00	(IP) = 0029 (AX) = 0104	(IP) = 002D (AX) = 010A
002D	mov cx, vec[di]	8D8D0E0 0	(IP) = 002D (CX) = 0804	(IP) = 0031 (CX) = 140A
0031	mov bx, 3	BB0300	(IP) = 0031 (BX) = 0006	(IP) = 0034 (BX) = 0003
0034	mov al, matr[bx] [di]	8A811600	(IP) = 0034 (AX) = 010A	(IP) = 0038 (AX) = 01FD

0038	mov cx, matr[bx][di]	8B891600	(IP) = 0038 (CX) = 140A	(IP) = 0038 (CX) = FEFD
003C	mov ax, SEG vec2	B8071A	(IP) = 003C (AX) = 01FD	(IP) = 003F (AX) = 1A07
003F	mov es, ax	83C0	(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, 0	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	(IP) = 004A (SP) = 0012 (STACK + 0) = 1A07 (STACK + 2) = 0000 (STACK + 4) = 19F5
004A	pop es	07	(IP) = 004A (SP) = 0012 (STACK + 0) = 1A07 (STACK + 2) = 0000 (STACK + 4) = 19F5 (ES) = 0000	(IP) = 004B (SP) = 0014 (STACK + 0) = 0000 (STACK + 2) = 19F5 (ES) = 1A07
004B	mov cx, es:[bx-1]	268B4FFF	(IP) = 004B (CX) = FEFD	(IP) = 004F (CX) = FFCE
004F	xchg cx, ax	91	(IP) = 004F (AX) = 0000 (CX) = FFCE	(IP) = 0050 (CX) = 0000 (AX) = FFCE

0050	mov di, ind	BF0200	(IP) = 0050 (DI) = 0002	(IP) = 0053 (DI) = 0002
0053	mov es:[bx+di], ax	268901	(IP) = 0053	(IP) = 0056
0056	mov bp, sp	8BEC	(IP) = 0056 (BP) = 0000	(IP) = 0058 (BP) = 0014
0058	push mem1	FF360000	(IP) = 0058 (SP) = 0014 (STACK + 0) = 0000 (STACK + 2) = 10F5	(IP) = 005C (SP) = 0012 (STACK + 0) = 01F4 (STACK + 2) = 0000 (STACK + 4) = 19F5
005C	push mem2	FF360200	(IP) = 005C (SP) = 0012 (STACK + 0) = 01F4 (STACK + 2) = 0000 (STACK + 4) = 19F5	(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] + 2	8B5602	(IP) = 0062 (DS) = 0000	(IP) = 0065 (DX) = 01F4
0065	mov ret 2	CA0200	(IP) = 0065 (SP) = 0010 (STACK + 0) = FFCE (STACK + 2) = 01F4 (STACK + 4) =	(IP) = FFCE (SP) = 0016 (STACK + 0) = 19F5 (CS) = 01F4

			0000 (STACK + 6) = 19F5 (CS) = 1A0A	
FFCE	push es	06	(IP) = FFCE (SP) = 0016 (STACK + 0) = 19F5	(IP) = FFCF (SP) = 0014 (STACK + 0) = 1A07 (STACK + 2) = 19F5
FFCF	popf	9D	(IP) = FFCF (SP) = 0014 (STACK + 0) = 1A07 (STACK + 2) = 19F5	(IP) = FFD0 (SP) = 0016 (STACK + 0) = 19F5
FFD0	add ax, C033	0533C0	(IP) = FFD0 (AX) = FFCE	(IP) = FFD3 (AX) = C001
FFD3	jmp FFD8	EB03	(IP) = FFD3	(IP) = FFD8
FFD8	ret	C3	(IP) = FFD8 (SP) = 0016 (STACK + 0) = 19F5	(IP) = 19F5 (SP) = 0018 Стек пуст

Выводы

В процессе выполнения лабораторной работы были получены навыки работы с различными режимами адресации и формированием исполнительного адреса на языке ассемблера.