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 и формирования исполнительного адреса

Студент гр. 9383 _____ Гордон Д.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Знакомство с видами адресации в Assembly.

Текст задания.

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. 6

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

ПРОТОКОЛ

2.
mov mem3, [bx]
LR2.asm(42): error A2052: Improper operand type
Чтение из и запись в память одной командой

7
LR2.asm(44): warning A4001: Extra characters on line

mov cx,vec2[di]
LR2.asm(50): warning A4031: Operand types must match
Регистр – 2 байта, vec2[di] – 1 байт

mov cx,matr[bx][di]
LR2.asm(54): warning A4031: Operand types must match
Регистр – 2 байта, matr[bx][di] – 1 байт

mov ax,matr[bx*4][di]
LR2.asm(55): error A2055: Illegal register value
Умножение 16 битных регистров

mov ax,matr[bp+bx]
LR2.asm(74): error A2046: Multiple base registers
2 базовых регистра

mov ax,matr[bp+di+si]
LR2.asm(75): error A2047: Multiple index registers
2 индексовых регистра

Main ENDP
LR2.asm(82): error A2006: Phase error between passes
Чтобы выйти из программы, вершина стека должна содержать адрес PSP и 0000. Когда мы доходим до ret 2, то после выполнения мы перейдем по адресу mem1:mem2.

lr2.exe

Адрес команды	Символический код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			До выполнения	После выполнения
0000	PUSH DS	1E	CS = 1A0A DS = 19F5	

			ES = 19F5 SS = 1A05 SP = 0018 Stack +0 0000	SP = 0016 Stack +0 19F5
0001	SUB AX, AX	2BC0	AX = 0000	AX = 0000
0003	PUSH AX	50	SP = 0016 Stack +0 19F5 Stack +2 0000	SP = 0014 Stack +0 0000 Stack +2 19F5
0004	MOV AX, 1A07	B8071A	AX = 0000	AX = 1A07
0007	MOV DS, AX	8ED8	DS = 19F5	DS = 1A07
0009	MOV AX, 01F4	B8F401	AX = 1A07	AX = 01F4
000C	MOV CX, AX	8BC8	CX = 00A8	CX = 01F4
000E	MOV BL, 24	B324	BX = 0000	BX = 0024
0010	MOV BH, CE	B7CE	BX = 0024	BX = CE24
0012	MOV [0002], FFCE	C7060200CEFF	2 3 DS:0000 00 00	2 3 DS:0000 CE FF
0018	MOV BX, 0006	BB0600	BX = CE24	BX = 0006
001B	MOV [0000], AX	A30000	0 1 DS:0000 00 00	0 1 DS:0000 F4 01
001E	MOV AL, [BX]	8A07	AX = 01F4	AX = 0101
0020	MOV AL, [BX+03]	8A4703	AX = 0101	AX = 0104
0023	MOV CX, [BX+03]	8B4F03	CX = 01F4	CX = 0804
0026	MOV DI, 0002	BF0200	DI = 0000	DI = 0002
0029	MOV AL. [000E+DI]	8A850E00	AX = 0104	AX = 010A
002D	MOV BX, 0003	BB0300	BX = 0006	BX = 0003
0030	MOV AL, [0016 + BX + DI]	8A811600	AX = 010A	AX = 01FD
0034	MOV AX, 1A07	B8071A	AX = 01FD	AX = 1A07
0037	MOV ES, AX	8EC0	ES = 19F5	ES = 1A07
0039	MOV AX, ES:[BX]	268B07	AX = 1A07	AX = 00FF
003C	MOV AX, 0000	B80000	AX = 00FF	AX = 0000
003F	MOV ES, AX	8EC0	ES = 1A07	ES = 0000

0041	PUSH DS	1E	SP = 0014 Stack +0 0000 Stack +2 19F5 Stack +4 0000	SP = 0012 Stack +0 1A07 Stack +2 0000 Stack +4 19F5
0042	POP ES	07	SP = 0012 Stack +0 1A07 Stack +2 0000 Stack +4 19F5 ES = 0	SP = 0014 Stack +0 0000 Stack +2 19F5 Stack +4 0000 ES = 1A07
0043	MOV CX, ES:[BX-01]	268B4FFF	CX = 0804	CX = FFCE
0047	XCHG AX, CX	91	AX = 0000 CX = FFCE	AX = FFCE CX = 0000
0048	MOV DI, 0002	BF0200	DI = 0002	DI = 0002
004B	MOV ES:[BX+DI], AX	268901	5 6 DS:0000 00 01	5 6 DS:0000 CE FF
004E	MOV BP, SP	8BEC	BP = 0000	BP = 0014
0050	MOV BP, SP	8BEC	BP = 0014	BP = 0014
0052	MOV DX, [BP+02]	8B5602	DX = 0000	DX = 19F5
0055	RET Far 0002	CA0200	AX = FFCE BX = 0003 CX = 0000 DX = 19F5 CS = 1A0A DS = 1A07 ES = 1A07 SP = 0014 Stack +0 0000 Stack +2 19F5	AX = 0000 BX = 0000 CX = 0000 DX = 0000 CS = 1A0A DS = 19F5 ES = 19F5 SP = 0018 Stack +0 7244 Stack +2 0000

ВЫВОДЫ

SEG возвращает адрес сегмента, где находится операнд.

1. *Регистровая адресация:*

Операнды могут располагаться в любых регистрах общего назначения и сегментных регистрах. В этом случае в операторе программы (на языке ассемблера) указывается название соответствующего регистра.

2. *Прямая адресация:*

Если известен адрес операнда, располагающегося в памяти, можно использовать этот адрес. В реальных программах обычно для задания статических переменных используют директивы определения данных, которые позволяют ссылаться на статические переменные не по адресу, а по имени. Если селектор сегмента данных находится в DS, имя сегментного регистра при прямой адресации можно не указывать, так как DS используется по умолчанию. Прямая адресация иногда называется адресацией по смещению.

3. *Косвенная адресация:*

Адрес операнда в памяти можно не указывать непосредственно, а хранить в любом регистре. До процессоров i80386 для этого можно было использовать только регистры BX, SI, DI и BP, но потом эти ограничения были сняты и адрес операнда разрешили считывать также и из EAX, EBX, ECX, EDX, ESI, EDI, EBP и ESP (но не из AX, CX, DX или SP напрямую – надо использовать EAX, ECX, EDX, ESP соответственно или предварительно скопировать смещение в BX, S1, DI или BP). Как и в случае прямой адресации, DS используется по умолчанию, но не во всех случаях: если смещение берут из регистров ESP, EBP или BP, то в качестве сегментного регистра используется SS. В реальном режиме можно свободно пользоваться всеми 32-битными регистрами, надо только следить, чтобы их содержимое не превышало границ 16-битного слова.

4. *Индексная адресация:*

Такая форма адресации используется в тех случаях, когда в регистре находится адрес начала структуры данных, а доступ надо осуществить к какому-нибудь элементу этой структуры. Другое важное применение адресации по базе со сдвигом – доступ из подпрограммы к параметрам, переданным в кадре стека, используя регистр BP (EBP) в качестве базы и номер параметра в качестве смещения.

До процессора i80386 в качестве базового регистра можно было использовать только регистры BX, BP, SI или DI и сдвиг мог быть только байтом или словом (со знаком). Начиная с процессоров i80386 и старше, можно дополнительно использовать EAX, EBX, ECX, EDX, EBP, ESP, ESI и EDI, так же как и для обычной косвенной адресации. С помощью этого метода можно организовывать доступ к одномерным массивам байт: смещение соответствует адресу начала массива, а число в регистре – индексу элемента массива, который надо использовать. Очевидно, что если массив состоит не из байт, а из слов,

придется умножать базовый регистр на два, а если из двойных слов – на четыре. Для этого предусмотрен следующий специальный метод адресации.

5. *Адресация с базированием и индексированием:*

Это самая полная возможная схема адресации, в которую входят все случаи, рассмотренные ранее, как частные. Смещение может быть байтом, словом или двойным словом. Если ESP или EBP используются в роли базового регистра, селектор

ПРИЛОЖЕНИЕ

LR2.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 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 7
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
```

LR2.lst:

```
Microsoft (R) Macro Assembler Version 5.10              10/22/20 11:17:2
                                                        Page     1-1


                        ; Программа изучения режиЙ
                        ♀ъЖадресации процессора I
                        ntelX86
= 0024                          EOL EQU '$'
= 0002                          ind EQU 2
= 01F4                          n1 EQU 500
=-0032                          n2 EQU -50
                        ; Стек программы
0000                    AStack SEGMENT STACK
0000  000C[                     DW 12 DUP(?) ;12*2=24b
        ????
                    ]

0018                    AStack ENDS
                        ; Данные программы
0000                    DATA SEGMENT
                        ; Директивы описания даннэ
                        ~o
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 ;8b
    06 05
000E  F6 EC 0A 14 E2 D8      vec2 DB -10,-20,10,20,-30,-40,30,40 ;8b
    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
                              ;16b
    FE FF 05 06 07 08
    F8 F9 FA FB
0026                    DATA ENDS
                        ; Код программы
0000                    CODE SEGMENT
                        ASSUME CS:CODE, DS:DATA, SS:AStack
                        ; Головная процедура
0000                    Main PROC FAR
0000  1E                push DS
0001  2B C0                    sub AX,AX ;AX = 0
0003  50                push AX
```

```
0004  B8 ---- R          mov AX,DATA
0007  8E D8                   mov DS,AX
                         ; ПРОВЕРКА РЕЖИМОВ АДРЕСАЙ
                         ЇيНА УРОВНЕ СМЕЩЕНИЙ
                         ; Регистровая адресация
0009  B8 01F4                 mov ax,n1
000C  8B C8                   mov cx,ax
000E  B3 24                   mov bl,EOL
0010  B7 CE                   mov bh,n2
                         ; Прямая адресация
0012  C7 06 0002 R FFCE       mov mem2,n2
0018  BB 0006 R          mov bx,OFFSET vec1
001B  A3 0000 R           mov mem1,ax
                         ; Косвенная адресация
001E  8A 07                   mov al,[bx]
                         ;mov mem3,[bx] ;improrer operand type
```

```
                        ; Базированная адресация
                        ;7 ;extra char-s in line (warning)
0020  8A 47 03                mov al,[bx]+3
0023  8B 4F 03                mov cx,3[bx]
                        ; Индексная адресация
0026  BF 0002                 mov di,ind
0029  8A 85 000E R            mov al,vec2[di]
                        ;mov cx,vec2[di] ;operand types must match (war
                        ning)
                        ; Адресация с базированиеЙ
                        Ли индексированием
002D  BB 0003                 mov bx,3
0030  8A 81 0016 R            mov al,matr[bx][di]
                        ;mov cx,matr[bx][di] ;operand types must match
                        (warning)
                        ;mov ax,matr[bx*4][di] ;illegal register value
                        ; ПРОВЕРКА РЕЖИМОВ АДРЕСАЙ
                        ЇцС УЧЕТОМ СЕГМЕНТОВ
                        ; Переопределение сегмент
                        a
                        ; ------ вариант 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
                        ; ------ вариант 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
                        ; ------ вариант 3
0048  BF 0002                 mov di,ind
004B  26: 89 01         mov es:[bx+di],ax
                        ; ------ вариант 4
004E  8B EC                   mov bp,sp
                        ;mov ax,matr[bp+bx] ;multiple base registers
                        ;mov ax,matr[bp+di+si] ;multiple index register
                        s
                        ; Использование сегмента э
```

```
                        ‘ ĐيЙ
                        ;push mem1
                        ;push mem2
0050  8B EC                   mov bp,sp
0052  8B 56 02                mov dx,[bp]+2
0055  CA 0002                 ret 2
0058                    Main ENDP ;phase error between passes
0058                    CODE ENDS
                        END Main
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA		STACK
CODE	0058	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 = 0058
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	LR2		
@VERSION	TEXT	510		

84 Source Lines
84 Total Lines
19 Symbols

47842 + 459418 Bytes symbol space free

0 Warning Errors

0 Severe Errors