**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 |  | Беззубов Д.В. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2022

## **Цель работы.**

Изучить режимы адресации и формирование исполнительного адреса на языке ассемблер.

* 1. 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. 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.

* 1. Выполнение работы

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

2. Программа протранслирована с созданием файла диагностических сообщений; Первая трансляция прошла с ошибками:

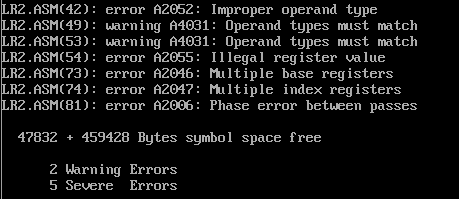


Рис.1 – трансляция программы с ошибками

Пояснение ошибок:

1. LR2.ASM(42): error A2052: Improper operand type

mov mem3,[bx];

Данная ошибка возникает, так как команды не могут оперировать сразу с двумя ячейками памяти. Один из операндов должен быть либо регистром, либо значением, другой может быть ячейкой в памяти.

1. LR2.ASM(49): warning A4031: Operand types must match и LR2.ASM(53): warning A4031: Operand types must match

mov cx,vec2[di] и mov cx,matr[bx][di];

Данная ошибка возникает из-за несоответствия типов. Попытка поместить байт в слово в обоих случаях.

1. LR2.ASM(54): warning A2055: Illegal register value

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

Данная ошибка возникает из-за попытки умножения базового регистра bx на 4, данный регистр нельзя использовать при индексной адресации с масштабированием.

1. LR2.ASM(73): warning A2046: Multiple base registers

mov ax,matr[bp+bx];

В данном случае ошибка возникает из-за того, что берутся два базовых регистра при том, что исполняемый адрес при адресации с базированием и индексированием берется как сумма адресов, расположенных в базовом и индексном регистрах.

1. LR2.ASM(74): warning A2047: Multiple index registers

mov ax,matr[bp+di+si];

В данном случае ошибка возникает из-за того, что берутся два индексных регистра при том, что исполняемый адрес при адресации с базированием и индексированием берется как сумма адресов, расположенных в базовом и индексном регистрах.

3. Исправленная программа снова протранслированна 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. Результат представлен в таблице 1.

Исходный код программы и листинг программы с закомментированными ошибочными операциями приведены в приложении А.

Таблица 1 – протокол отладки программы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес  команды | Символический код команды | 16-ричный код команды | Изменяемые данные | |
| до | после |
| 0000 | PUSH DS | 1E | IP = 0000  SP=0018  Stack +0 = 0000  Stack +2 = 0000  Stack +4 = 0000  Stack +6 = 0000 | IP = 0001  SP=0016  Stack +0 = 19F5  Stack +2 = 0000  Stack +4 = 0000  Stack +6 = 0000 |
| 0001 | SUB AX, AX | 2BCO | AX=0000  IP = 0001  SP=0016 | AX=0000  IP = 0003  SP=0016 |
| 0003 | PUSH AX | 50 | IP = 0003  SP=0016  Stack +0 = 19F5  Stack +2 = 0000  Stack +4 = 0000  Stack +6 = 0000 | IP = 0004  SP=0014  Stack +0 = 0000  Stack +2 = 19F5  Stack +4 = 0000  Stack +6 = 0000 |
| 0004 | MOV AX,1A07 | B8071A | AX = 0000  IP = 0004  SP=0014 | AX =1A07  IP = 0007  SP=0014 |
| 0007 | MOV DS,AX | 8ED8 | DS=19F5  IP = 0007  SP=0014 | DS=1A07  IP = 0009  SP=0014 |
| 0009 | MOV AX,01F4 | B8F401 | AX = 1A07  IP = 0009 | AX = 01F4  IP = 000C |
| 000C | MOV CX,AX | 8BC8 | CX = 00B0  IP = 000C | 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 | C7060200CEFF | 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  [BX] = [0006] = 05  IP = 001E | AX = 0105  IP = 0020 |
| 0020 | MOV AL,[BX+03] | 8A4703 | 1. AX = 0105 2. [BX+03] = 08 3. IP = 0020 | 1. AX = 0108 2. IP = 0023 |
| 0023 | MOV CX, [BX+03] | 8B4F03 | 1. CX = 01F4 2. [BX+03] = 08 3. IP = 0023 | 1. CX = 0C08 2. IP = 0026 |
| 0026 | MOV DI, 0002 | BF0200 | 1. DI = 0000 2. IP = 0026 | 1. DI = 0002 2. IP = 0029 |
| 0029 | MOV AL, [000E+DI] | 8A850E00 | 1. AX = 0108 2. [000E+DI] = 14 3. IP = 0029 | 1. AX = 0114 2. IP = 002D |
| 002D | MOV BX, 0003 | BB0300 | 1. BX = 0006 2. IP = 002D | 1. BX = 0003 2. IP = 0030 |
| 0030 | MOV AL, [0016+BX+DI] | 8A811600 | 1. [0016+BX+DI] = 03 2. AX = 0114 3. IP = 0030 | 1. AX = 0103 2. IP = 0034 |
| 0034 | MOV AX, 1A07 | B8071A | 1. AX = 0103 2. IP = 0034 | 1. AX = 1A07 2. IP = 0037 |
| 0037 | MOV ES, AX | 8EC0 | 1. ES = 19F5 2. AX = 1A07 3. IP = 0037 | 1. ES = 1A07 2. IP = 0039 |
| 0039 | MOV AX, ES:[BX] | 268B07 | 1. AX = 1A07 2. IP = 0039 | 1. AX = 00FF 2. IP = 003C |
| 003C | MOV AX, 0000 | B80000 | 1. AX = 00FF 2. IP = 003C | 1. AX = 0000 2. IP = 003F |
| 003F | MOV ES, AX | 8EC0 | 1. ES = 1A07 2. AX = 0000 3. IP = 003F | 1. ES = 0000 2. IP = 0041 |
| 0041 | PUSH DS | 1E | 1. IP = 0041 2. SP = 0014   Stack+0 = 0000  Stack+2 = 19F5  Stack+4 = 0000 | 1. IP = 0042 2. SP = 0012   Stack+0 = 1A07  Stack+2 = 0000  Stack+4 = 19F5 |
| 0042 | POP ES | 07 | 1. ES = 0000 2. IP = 0042 3. SP = 0012   Stack+0 = 1A07  Stack+2 = 0000  Stack+4 = 19F5 | 1. ES = 1A07 2. IP = 0043 3. SP = 0014   Stack+0 = 0000  Stack+2 = 19F5  Stack+4 = 0000 |
| 0043 | MOV CX, ES:[BX—01] | 268B4FFF | 1. CX = 0C08 2. IP = 0043 | 1. CX = FFCE 2. IP = 0047 |
| 0047 | XCHG AX, CX | 91 | 1. AX = 0000 2. CX = FFCE 3. IP=0047 | 1. AX = FFCE 2. CX = 0000 3. IP=0048 |
| 0048 | MOV DI, 0002 | BF0200 | 1. DI = 0002 2. IP = 0048 | 1. DI = 0002 2. IP = 004B |
| 004B | MOV ES:[BX+DI], AX | 268901 | 1. IP = 004B | 1. IP = 004E |
| 004E | MOV BP, SP | 8BEC | 1. BP = 0000 2. SP = 0014 3. IP = 004E | 1. BP = 0014 2. IP = 0050 |
| 0050 | PUSH [0000] | FF360000 | IP = 0050  [0000] = 01F4   1. SP = 0014   Stack+0 = 0000  Stack+2 = 19F5  Stack+4 = 0000 | 1. IP = 0054   [0000] = 01F4   1. SP = 0012   Stack+0 = 01F4  Stack+2 = 0000  Stack+4 = 19F5 |
| 0054 | PUSH [0002] | FF360200 | 1. IP = 0054   [0002] = FFCE   1. SP = 0012   Stack+0 = 01F4  Stack+2 = 0000  Stack+4 = 19F5  Stack+6 = 0000 | 1. IP = 0058 2. [0002] = FFCE 3. SP = 0010 4. Stack+0 = FFCE   Stack+2 = 01F4  Stack+4 = 0000  Stack+6 = 19F5 |
| 0058 | MOV BP, SP | 8BEC | 1. BP = 0014 2. SP = 0010 3. IP = 0058 | 1. BP = 0010 2. SP = 0010 3. IP = 005A |
| 005A | MOV DX, [BP+02] | 8B5602 | 1. DX = 0000 2. [BP+02] = 01F4 3. IP = 005A | 1. DX = 01F4 2. IP = 005D |
| 005D | RET Far 0002 | CA0200 | 1. IP = 005D 2. SP = 0010 3. CS = 1A0A 4. Stack+0 = FFCE   Stack+2 = 01F4  Stack+4 = 0000  Stack+6 = 19F5 | 1. IP = FFCE 2. SP = 0016 3. CS = 01F4 4. Stack+0 = 19F5   Stack+2 = 0000  Stack+4 = 0000  Stack+6 = 0000 |

* 1. Выводы

Изучены режимы адресации и формирование исполнительного адреса на языке ассемблер.

1. Приложение А

Название файла: lr2.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 5,6,7,8,12,11,10,9

vec2 DB -20,-30,20,30,-40,-50,40,50

matr DB -5,-6,-7,-8,4,3,2,1,-1,-2,-3,-4,8,7,6,5

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

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

Название файла: lr2.lst

Microsoft (R) Macro Assembler Version 5.10 10/9/22 13:49:46

Page 1-1

; Программа изучения режи΍

ов адресации 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(?)

????

]

0018 AStack ENDS

; Данные программы

0000 DATA SEGMENT

; Директивы описания даннύ

х

0000 0000 mem1 DW 0

0002 0000 mem2 DW 0

0004 0000 mem3 DW 0

0006 05 06 07 08 0C 0B vec1 DB 5,6,7,8,12,11,10,9

0A 09

000E EC E2 14 1E D8 CE vec2 DB -20,-30,20,30,-40,-50,40,50

28 32

0016 FB FA F9 F8 04 03 matr DB -5,-6,-7,-8,4,3,2,1,-1,-2,-3,-4,8,7,6,5

02 01 FF FE FD FC

08 07 06 05

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

0003 50 push AX

0004 B8 ---- R mov AX,DATA

0007 8E D8 mov DS,AX

; ПРОВЕРКА РЕЖИМОВ АДРЕСА΍

ИИ НА 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]

; Базированная адресация

Microsoft (R) Macro Assembler Version 5.10 10/9/22 13:49:46

Page 1-2

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]

; Адресация с базирование΍

и индексированием

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]

; ПРОВЕРКА РЕЖИМОВ АДРЕСА΍

ИИ С УЧЕТОМ СЕГМЕНТОВ

; Переопределение сегмент

а

; ------ 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]

; mov ax,matr[bp+di+si]

; Использование сегмента ύ

тека

0050 FF 36 0000 R push mem1

0054 FF 36 0002 R 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 10/9/22 13:49:46

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 LR2

@VERSION . . . . . . . . . . . . TEXT 510

83 Source Lines

83 Total Lines

19 Symbols

47840 + 459420 Bytes symbol space free

0 Warning Errors

0 Severe Errors