**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 |  | Насонов Я. К. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2022

## **Цель работы.**

Изучить работу с режимами адресации на языке программирования 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.

* 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; обнаруженные ошибки объяснены и закомментированы соответствующие операторы в тексте программы.

source \_lab2.asm(41): error A2502: Improper operand type

mov mem3,[bx]

Машинные команды не могут работать одновременно с 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.

source \_ lab2.asm(43): warning A4001: Extra characters on line 7

Лишний символ.

source \_ lab2.asm(49): warning A4031: Operand types must match

mov cx,vec2[di]

Разные типы операндов, cx – слово (2 байта), а vec2[di] – размерность 1 байт

source \_ lab2.asm(53): warning A4031: Operand types must match

mov cx,matr[bx][di]

Разные типы операндов, cx – слово (2 байта), а matr[bx][di] – размерность 1 байт

source \_ lab2.asm(54): error A2055: Illegal register value

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

В непосредственной адресации с базированием и индексированием для вычисления исполнительного адреса берется сумма базового и индексного регистра, к которым добавляется непосредственно фигурирующее в команде смещение. Там не фигурирует умножение.

source \_ lab2.asm(73): error A2046: Multiple base register

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.

source \_ lab2.asm(74): error A2047: Multiple index register

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.

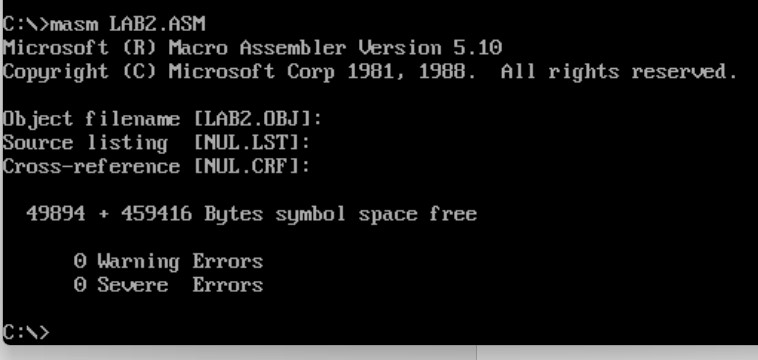
source \_ lab2.asm(81): error A2006: Phase error between passes

Main ENDP

Ошибка говорит о том, что в функции Main допущены ошибки.

3. Снова протранслирована программа и скомпонован загрузочный модуль.

Трансляция программы после исправления ошибок



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

Процесс выполнения 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 | IP = 0001  SP=0016  STACK (+0) = 19F5 |
| 0001 | SUB AX, AX | 2BCO | 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 | C7060200CEFF | 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 = 011F |
| 0020 | MOV AL, [BX+03] | 8A4703 | 1. IP = 0020 2. AX = 011F | 1. IP = 0023 2. AX = 0122 |
| 0023 | MOV CX, [BX+03] | 8B4F03 | 1. IP = 0023 2. CX = 01F4 | 1. IP = 0026 2. CX = 2622 |
| 0026 | MOV DI, 0002 | BF0200 | 1. IP = 0026 2. DI = 0002 | 1. IP = 0029 2. DI = 0002 |
| 0029 | MOV AL, [000E+DI] | 8A850E00 | 1. IP = 0029 2. AX = 012 | 1. IP = 002D 2. AX = 01CE |
| 002D | MOV BX, 0003 | BB0300 | 1. IP = 002D 2. BX = 0006 | 1. IP = 0030 2. BX = 0003 |
| 0030 | MOV AL, [0016+BX+DI] | 8A811600 | 1. IP = 0030 2. AX = 01CE | 1. IP = 0034 2. AX = 01FF |
| 0034 | MOV AX, 1A07 | B8071A | 1. IP = 0034 2. AX = 01FF | 1. IP = 0037 2. AX = 1A07 |
| 0037 | MOV ES, AX | 8EC0 | 1. IP = 0037 2. ES = 19F5 | 1. IP = 0039 2. ES = 1A07 |
| 0039 | MOV AX, ES: [BX] | 268B07 | 1. IP = 0039 2. AX = 1A07 | 1. IP = 003C 2. AX = 00FF |
| 003C | MOV AX, 0000 | B80000 | 1. IP = 003C 2. AX = 00FF | 1. IP = 003F 2. AX = 0000 |
| 003F | MOV ES, AX | 8EC0 | 1. IP= 003F 2. ES = 1A07 | 1. IP= 0041 2. ES= 0000 |
| 0041 | PUSH DS | 1E | 1. IP = 0041 2. SP = 0014 3. STACK (+0) = 0000 4. STACK (+2) =19F5 5. STACK (+4) = 0000 | 1. IP = 0042 2. SP = 0012 3. STACK (+0) = 1A07 4. STACK (+2) = 0000 5. STACK (+4) = 19F5 |
| 0042 | POP ES | 07 | 1. IP = 0042 2. ES = 0000 3. SP = 0012 4. STACK (+0) = 1A07 5. STACK (+2) = 0000 6. STACK (+4) =19F5 | 1. IP = 0043 2. ES = 1A07 3. SP = 0014 4. STACK (+0) = 0000 5. STACK (+2) = 19F5 6. STACK (+4) = 0000 |
| 0043 | MOV CX, ES: [BX—01] | 268B4FFF | 1. IP = 0043 2. CX = 2622 | 1. IP= 0047 2. CX= FFCE |
| 0047 | XCHG AX, CX | 91 | 1. IP=0047 2. AX = 0000 3. CX = FFCE | 1. IP=0048 2. AX = FFCE 3. CX = 0000 |
| 0048 | MOV DI, 0002 | BF0200 | 1. IP = 0048 2. DI=0002 | 1. IP = 004B 2. DI=0002 |
| 004B | MOV ES: [BX + DI], AX | 268901 | 1. IP = 004B | 1. IP = 004E |
| 004E | MOV BP, SP | 8BEC | 1. IP = 004E 2. BP = 0010 | 1. IP = 0050 2. BP = 0014 |
| 0050 | PUSH [0000] | FF360000 | IP = 0050   1. SP = 0014 2. STACK (+0) = 0000 3. STACK (+2) = 19F5 4. STACK (+4) = 0000 | 1. IP = 0054 2. SP = 0012 3. STACK (+0) = 01F4 4. STACK (+2) = 0000 5. STACK (+4) =19F5 |
| 0054 | PUSH [0002] | FF360200 | 1. IP = 0054 2. SP = 0012 3. STACK (+0) = 01F4 4. STACK (+2) = 0000 5. STACK (+4) = 19F5 6. STACK (+6) = 0000 | 1. IP = 0058 2. SP = 0010 3. STACK (+0) = FFCE 4. STACK (+2) = 01F4 5. STACK (+4) = 0000 6. STACK (+6) = 19F5 |
| 0058 | MOV BP, SP | 8BEC | 1. IP = 0058 2. BP = 0014 | 1. IP = 005A 2. BP = 0010 |
| 005A | MOV DX, [BP+02] | 8B560 | 1. IP = 005A 2. DX = 01F4 | 1. IP = 005D 2. DX = 01F4 |
| 005D | RET Far 0002 | CA0200 | 1. IP = 005D 2. SP = 0010 3. CS =1A0A 4. STACK (+0) = FFCE 5. STACK (+2) = 01F4 6. STACK (+4) = 0000 7. STACK (+6) = 19F5 | 1. IP = FFCE 2. SP = 0016 3. CS = 01F4 4. STACK (+0) = 19F5 5. STACK (+2) = 0000 6. STACK (+4) = 0000 7. STACK (+6) = 0000 |

* 1. Выводы

В ходе выполнения лабораторной работы были получены основные навыки работы с режимами адресации на языке программирования Ассемблер.

1. Приложение А  
   Исходный код программы

Название файла: 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(?) ; word - 2 байта

AStack ENDS

; Данные программы

DATA SEGMENT

; Директивы описания данных

mem1 DW 0 ; word - 2 байта

mem2 DW 0

mem3 DW 0

vec1 DB 31,32,33,34,38,37,36,35 ; byte - 1 байт

vec2 DB 50,60,-50,-60,70,80,-70,-80

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

DATA ENDS

; Код программы

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

; Головная процедура

Main PROC FAR

push DS ; push data segment

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]

; нельзя работать с операндами,

; оба из которых находятся в оперативной памяти

;mov mem3,[bx]

; Базированная адресация

mov al,[bx]+3

mov cx,3[bx]

; Индексная адресация

mov di,ind ; destination index

mov al,vec2[di]

; cx - слово (2 bytes), vec2[di] - 1 byte

;mov cx,vec2[di]

; Адресация с базированием и индексированием

mov bx,3

mov al,matr[bx][di]

; cx - word (2 bytes), matr[bx][di] - byte

;mov cx,matr[bx][di]

; в непосредственной адресации с базированием

; и индексированием берётся сумма базового и

; индексного регистра, к ним добавляется смещение, но

; умножение там не фигурирует

;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

; в косвенной адресации с с индексированием адрес

; берётся в виде суммы адресов базового и индексного

; регистров, но тут оба регистра базовые

;mov ax,matr[bp+bx]

; 2 индексных и 1 базовый регистр, должны быть

; базовый и индексный регистры

;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