**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 в** **Intel8086**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8383 |  | Ларин Антон |
| Преподаватель |  |  |

Санкт-Петербург

2019

**Цель работы.**

Изучить различные виды адресации. Изучить возможные ошибки при адресации и научиться их избегать

**Основные теоретические положения.**

Большинство команд процессора Intel Х86 выполняются с аргументами, которые принято называть *операндами.* Операнды в программе могут задаваться следующим образом:

1. в регистрах общего назначения;
2. непосредственно в коде команды;
3. в ячейках памяти, задаваемых в команде прямо или косвенно;
4. в портах ввода-вывода.

Для указания места расположения операнда используются 8 режимов адресации, использование которых иллюстрируется в таблице 3.2.

**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. Непосредственная адресация**

Некоторые команды (пересылки, все арифметические команды, кроме деления) по­зволяют указывать один из операндов непосредственно в операторе про­граммы.

**3. 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.

**4. 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 ВХ, SI, DI и ВР, но потом эти ограничения были сняты и адрес операнда разрешили считывать также и из ЕАХ, ЕВХ, ЕСХ, EDX, ESI, EDI**,** ЕВР и ESP (но не из AX, CX, DX или SP напрямую – надо исполь­зовать ЕАХ, ЕСХ, EDX, ESP соответственно или предварительно скопи­ровать смещение в ВХ, S1, DI или ВР). Как и в случае прямой адресации, DS используется по умолчанию, но не во всех случаях: если смещение берут из регистров ESP, ЕВР или ВР, 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.

**5-6. Базовая или 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 ВР (ЕВР) в качестве базы и но­мер параметра в качестве смещения.

До процессора i80386 в качестве базового регистра можно было использовать только регистры ВХ, ВР, SI или DI и сдвиг мог быть только байтом или словом (со знаком). Начиная с процессоров i80386 и старше, можно допол­нительно использовать ЕАХ, ЕВХ, ЕСХ, EDX, ЕВР, 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.

**7. Индексная адресация с масштабированием**

Этот метод адресации полностью идентичен предыдущему, за исключе­нием того, что с его помощью можно прочитать элемент массива слов, двойных слов или учетверенных слов, просто поместив номер элемента в регистр

mov ax, [esi\*2]+2

Множитель, который может быть равен 1, 2, 4 или 8, соответствует размеру элемента массива – байту, слову, двойному слову, учетверенно­му слову соответственно. Из регистров в этом варианте адресации мож­но использовать только ЕАХ, ЕВХ, ЕСХ, EDX, ESI, EDI, ЕВР, ESP, но не **SI,** DI, ВР или SP, которые можно было использовать в предыдущих ва­риантах.

**8. Адресация по базе с индексированием и масштабированием**

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 ЕВР используются в роли базового регистра, селектор сегмента операн­да берется по умолчанию из регистра SS, во всех остальных случаях – из DS.

Таблица 3.2. Использование режимов адресации в архитектуре Intel Х86

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Режим адресации | | | Описание в Ассемблере | | Регистр сегмента (по умолчанию) | | Пример использования |
| Код | Название |  | |  | |  | |
| 0 | Регистровая | EAX/AX/AL | |  | | MOV EAX, EDX ; DEC CX  MOV AL, CH ; PUSH DS  IN AL, DX | |
| 1 | Непосредственная | Данное в виде константы (номер порта ввода-вывода) | |  | | MOV AX, 1000h; ADD AL, -30h AND EAX, 0000FFFFh  OUT 21h, AL | |
| 2 | Прямая | Имя (метка) сме- щения в памяти | | DS | | MOV BL, Mem\_B1  MOV AX, OFFSET Table | |
| 3 | Косвенно-регистровая | [BX], [SI],  [BP],  [DI],  [EAX] и др. | | DS  SS  DS(ES)  DS | | MOV AL, [SI]  MOV CX, [BP]  MOV AX, ES:[BX]  ADD EAX, [EDX] | |
| 4 | Базовая | [BX] + смещение,  [BP] + смещение | | DS  SS | | MOV AX, CS:[BX] +4  MOV CX, PAR\_TAB[BP] | |
| 5 | Индексная | [SI] + смещение,  [DI] + смещение | | DS  DS(ES) | | CMP ‘A’, STR1[SI]  MOV AL, BYT\_TAB[DI] | |
| 6 | Базово-индексная | BX] [SI] + смещ.,  BX] [DI] + смещ.,  [BP] [SI] + смещ.,  [BP] [DI] + смещ. | | DS  DS  SS  SS | | MOV AX, ES:VALUE[BX][DI] | |
| 7 | Индексная с масштабированием | [ERg\* m] + смещ. | | DS  SS – для EBP,ESP | | MOV CX, WORD\_TAB[ESI\*2] | |
| 8 | Базово-индексная с масштабированием | [ERg1] [ERg2\* m] + смещение | | DS  SS – для EBP,ESP | | MOV BX, [EDX][ECX\*2] + 8 | |

**Примечания:**

Базовая адресация применяется для работы со строками или записями, при этом в базовый регистр заносится начало структуры или записи, а смещение задает начало некоторого поля записи или структуры (регистр BP используется по умолчанию для доступа к параметрам процедур через кадр стека). Индексная адресация применяется для доступа к элементам однородных (обычно одномерных) массивов, смещение задает начало элемента этого массива. Базово-индексная адресация используется при работе с элементами полей записи и для работы с двумерными массивами.

**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.

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.

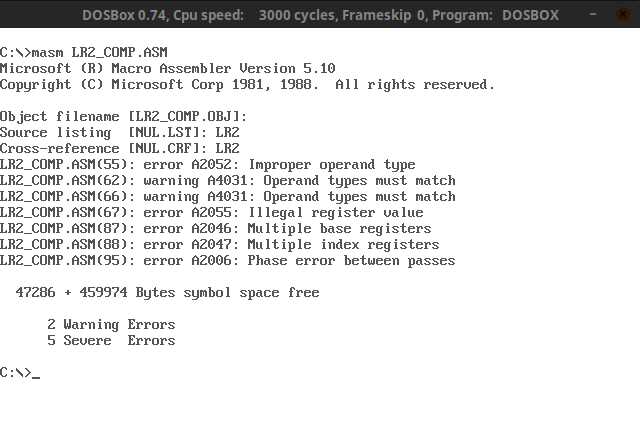
***Отчет по работе должен содержать:***

1. текст задания;
2. текст исходного файла программы с заданными значениями исходных данных;
3. описание обнаруженных при первоначальной трансляции ошибок и их объяснение;
4. листинг успешной трансляции программы с закомментированными ошибочными операторами;
5. протокол работы на компьютере, включающий описание выполнения каждой команды в пошаговом режиме под управлением отладчика, представленный в виде таблицы 1 (черновики протоколов должны быть подписаны преподавателем).
6. выводы по работе.

**Выполнение**

Первая компиляция прошла с ошибками(Рис. 1)

Рисунок 1 — Компиляция с ошибками

****

Пояснение ошибок:

55: Improper operand type

mov mem3,[bx]; невозможно перемещать значения из памяти в память

62: Operands types must match

mov cx,vec2[di]; несоответствие типов. попытка поместить байт в слово

66: Operands types must match

mov cx,matr[bx][di]; несоответствие типов. попытка поместить байт в слово

67: Illegal register value

mov ax,matr[bx\*4][di]; регистр dx нельзя умножать

87: Multiple base registers

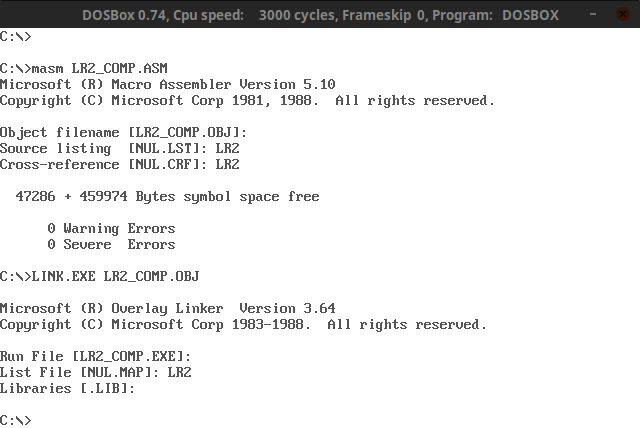
mov ax,matr[bp+bx]; в операнде нельзя использовать более одного базового регистра

88: Multiple index registers

mov ax,matr[bp+di+si]; в операнде нельзя использовать более одного индексного регистра

После комментирование строк с ошибками компиляция и сборка прошла успешно(Рис. 2)

Рисунок 2 — Удачная компиляция и сборка



Результаты отладки занесены в таблицу(Таб. 1)

Таблица 1—Отладка LR2\_COMP.ASM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес  Команды | Символический  код команды | 16-ричный  код команды | Содержимое регистров и ячеек памяти | |
| до выполнения . | После выполнения |
| 0000 | PUSH DS | 2BC0 | (CS) = 11BC  (DS) = 119C  (ES) = 119C  (SS) = 11A0  (SP) = 0018  (IP) = 0000  Stack +0 = 0000 | (CS) = 11BC  (DS) = 119C  (ES) = 119C  (SS) = 11A0  (SP) = 0016  (IP) = 0001  Stack +0 = 119C |
| 0001 | SUB AX,AX | 2BC0 | (IP) = 0001  (ZF) = 0  (PF) = 0 | (IP) = 0003  (ZF) = 1  (PF) = 1 |
| 0003 | PUSH AX | 50 | (SP) = 0016  (IP) = 0003  Stack +0 = 119C  Stack +2 = 0000 | (SP) = 0014  (IP) = 0004  Stack +0 = 0000  Stack +2 = 119C |
| 0004 | MOV AX,11AE | B8AE11 | (AX) = 0000  (IP) = 0004 | (AX) = 11AE  (IP) = 0007 |
| 0007 | MOV DS,AX | 8ED8 | (IP) = 0007  (DS) = 119C | (IP) = 0009  (DS) = 11AE |
| 0009 | MOV AX,01F4 | B8F401 | (AX) = 11AE  (IP) = 0009 | (AX) = 01F4  (IP) = 000C |
| 000C | MOV CX,AX | 8BC8 | (CX) = 00AE  (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  (IP) = 001E | (AX) = 0126  (IP) = 0020 |
| 0020 | MOV AL,[BX+03] | 8A4703 | (AX) = 0126  (IP) = 0020 | (AX) = 0123  (IP) = 0023 |
| 0023 | MOV CX,[BX+03] | 8B4F03 | (CX) = 01F4  (IP) = 0023 | (CX) = 1F23  (IP) = 0026 |
| 0026 | MOV DI,0002 | BF0200 | (DI) = 0000  (IP) = 0026 | (DI) = 0002  (IP) = 0029 |
| 0029 | MOV AL,[DI+000E] | 8A850E00 | (AX) = 0123  (IP) = 0029 | (AX) = 01BA  (IP) = 002D |
| 002D | MOV BX,0003 | BB0300 | (BX) = 0006  (IP) = 002D | (BX) = 0003  (IP) = 0030 |
| 0030 | MOV AL,[BX+DI+0016] | 8A811600 | (AX) = 01BA  (IP) = 0030 | (AX) = 01F9  (IP) = 0034 |
| 0034 | MOV AX,11AE | B8AE11 | (AX) = 01F9  (IP) = 0034 | (AX) = 11AE  (IP) = 0037 |
| 0037 | MOV ES,AX | 8EC0 | (ES) = 119C  (IP) = 0037 | (ES) = 11AE  (IP) = 0039 |
| 0039 | MOV AX.ES:[BX] | 268B07 | (AX) = 11AE  (IP) = 0037 | (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 | (SP) = 0014  (IP) = 0041  Stack +0 = 0000  Stack +2 = 119C  Stack +4 = 0000 | (SP) = 0012  (IP) = 0042  Stack +0 = 11AE  Stack +2 = 0000  Stack +4 = 119C |
| 0042 | POP ES | 07 | (SP) = 0012  (IP) = 0042  (ES) = 0000  Stack +0 = 11AE  Stack +2 = 0000  Stack +4 = 119C | (SP) = 0014  (IP) = 0043  (ES) = 11AE  Stack +0 = 0000  Stack +2 = 119C  Stack +4 = 0000 |
| 0043 | MOV CX,ES:[BX-01] | 268B4FFF | (CX) = 1F23  (IP) = 0043 | (CX) = FFCE  (IP) = 0047 |
| 0047 | XCHG AX,CX | 91 | (CX) = FFCE  (AX) = 0000  (IP) = 0047 | (CX) = 0000  (AX) = FFCE  (IP) = 0048 |
| 0048 | MOV DI,0002 | BF0200 | (DI) = 0002  (IP) = 0048 | (DI) = 0002  (IP) = 004B |
| 004B | MOV ES:[BX+DI],AX | 268901 | (ES) = 11AE  (IP) = 004B | (ES) = 11AE  (IP) = 004E |
| 004E | MOV BP,SP | 8BEC | (BP) = 0000  (IP) = 004E | (BP) = 0014  (IP) = 0050 |
| 0050 | PUSH [0000] | FF360000 | (SP) = 0014  (IP) = 0050  Stack +0 = 0000  Stack +2 = 119C  Stack +4 = 0000 | (SP) = 0012  (IP) = 0054  Stack +0 = 01F4  Stack +2 = 0000  Stack +4 = 119C |
| 0054 | PUSH [0002] | FF360200 | (SP) = 0012  (IP) = 0054  Stack +0 = 01F4  Stack +2 = 0000  Stack +4 = 119C  Stack +6 = 0000 | (SP) = 0010  (IP) = 0058  Stack +0 = FFCE  Stack +2 = 01F4  Stack +4 = 0000  Stack +6 = 119C |
| 0058 | MOV BP,SP | 8BEC | (BP) = 0014  (IP) = 0058 | (BP) = 0010  (IP) = 005A |
| 005A | MOV DX,[BP+02] | 8B5602 | (DX) = 0000  (IP) = 005A | (DX) = 01F4  (IP) = 005D |
| 005D | RET Far | CB | (SP) = 0010  (ES) = 11B1  (IP) = 005D  Stack +0 = FFCE  Stack +2 = 01F4  Stack +4 = 0000  Stack +6 = 119C | (SP) = 0014  (ES) = 01F4  (IP) = FFCE  Stack +0 = 0000  Stack +2 = 119C  Stack +4 = 0000  Stack +6 = 0000 |

**Выводы.**

В результате работы были разобраны некоторые базовые концепции языка асемблера. Были изучены различные способы адресации в языке ассемблера. Была исправлена транслирована и собрана тестовае программа, разобраны ошибки, встречающиеся в ней, произведена пошаговая отладка с занесением результатов в таблицу.

Приложение 1

HELLO1.ASM

; HELLO1.ASM - упрощенная версия учебной программы лаб.раб. N1

; по дисциплине "Архитектура компьютера"

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Назначение: Программа формирует и выводит на экран приветствие

; пользователя с помощью функции ДОС "Вывод строки"

; (номер 09 прерывание 21h), которая:

; - обеспечивает вывод на экран строки символов,

; заканчивающейся знаком "$";

; - требует задания в регистре ah номера функции=09h,

; а в регистре dx - смещения адреса выводимой

; строки;

; - использует регистр ax и не сохраняет его

; содержимое.

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DOSSEG ; Задание сегментов под ДОС

.MODEL SMALL ; Модель памяти-SMALL(Малая)

.STACK 100h ; Отвести под Стек 256 байт

.DATA ; Начало сегмента данных

Greeting LABEL BYTE ; Текст приветствия

DB 'Ларин Антон 8383',13,10,'$'

.CODE ; Начало сегмента кода

mov ax, @data ; Загрузка в DS адреса начала

mov ds, ax ; сегмента данных

mov dx, OFFSET Greeting ; Загрузка в dx смещения

; адреса текста приветствия

DisplayGreeting:

mov ah, 9 ; # функции ДОС печати строки

int 21h ; вывод на экран приветствия

mov ah, 4ch ; # функции ДОС завершения программы

int 21h ; завершение программы и выход в ДОС

END

Приложение 2

HELLO2.ASM

; HELLO2 - Учебная программа N2 лаб.раб.#1 по дисциплине "Архитектура компьютера"

; дштл Программа использует процедуру для печати строки

;

; ТЕКСТ ПРОГРАММЫ

EOFLine EQU '$' ; Определение символьной константы

; "Конец строки"

; Стек программы

AStack SEGMENT STACK

DW 12 DUP(?) ; Отводится 12 слов памяти

AStack ENDS

; Данные программы

DATA SEGMENT

; Директивы описания данных

HELLO DB 'Кто здесь?', 0AH, 0DH,EOFLine

GREETING DB 'Ларин Антон 8383$'

DATA ENDS

; Код программы

CODE SEGMENT

ASSUME CS:Code ;DS:DATA SS:STACKSS

; Процедура печати строки

WriteMsg PROC NEAR

mov AH,9

int 21h ; Вызов функции DOS по прерыванию

ret

WriteMsg ENDP

; Головная процедура

Main PROC FAR

push DS ;\ Сохранение адреса начала PSP в стеке

sub AX,AX ; > для последующего восстановления по

push AX ;/ команде ret, завершающей процедуру.

mov AX,DATA ; Загрузка сегментного

mov DS,AX ; регистра данных.

mov DX, OFFSET HELLO ; Вывод на экран первой

call WriteMsg ; строки приветствия.

mov DX, OFFSET GREETING ; Вывод на экран второй

call WriteMsg ; строки приветствия.

ret ; Выход в DOS по команде,

; находящейся в 1-ом слове PSP.

Main ENDP

CODE ENDS

END Main