

Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебная дисциплина «Встраиваемые микропроцессорные системы»

по выполнению лабораторной работы
«Программирование на языке Ассемблер. Изучение команд работы с регистра-
ми и памятью микропроцессора»

Минск
2017

»

1. :

Получение практических навыков по использованию команд работы с регистрами и памятью микропроцессора.

2.

Используя команды передачи данных составить программу на языке Ассемблера.

3.

Техническое задание, ПК, эмулятор DOSBox.

4.

Микропроцессор (МП) может использовать различные методы (или режимы) для получения тех данных, которыми должна оперировать программа.

Эти оп / оп р D ☐ Э

операнда относительно начала сегмента, в котором находится операнд. При прямой адресации исполнительный адрес является составной частью команды. МП добавляет этот адрес к сдвинутому на 4 бита (т.е. умноженному на 16) содержимому регистра сегмента данных DS и получает 20-битовый физический адрес операнда. Обычно прямая адресация применяется, если операндом служит метка. Например,

MOV AX, TABLE ; загрузка в регистр AX содержимого ячейки памяти с меткой TABLE

При этом надо иметь в виду, что байты в памяти располагаются в обратной последовательности: старшие байты - по старшим адресам, младшие байты - по младшим адресам, т.е. если по адресу TABLE хранится значение AA, а по адресу TABLE+1 значение BB, то в результате приведенной операции регистр AX будет содержать значение BBAA.

При косвенной регистровой адресации исполнительный адрес операнда содержится в базовом регистре BX, регистре указателя базы BP или в индексном регистре SI или DI.

Косвенные регистровые операнды необходимо заключать в квадратные скобки. Например,

MOV AX, [BX] ; загрузить в регистр AX содержимое ячейки, исполнительный адрес, которой находится в регистре BX

Чтобы поместить смещение ячейки TABLE в регистр BX можно использовать операцию OFFSET. Например,

MOV BX, OFFSET TABLE ; поместить в регистр BX смещение ячейки TABLE

Данный способ адресации эффективен в тех случаях, когда необходимо получать доступ к последовательности ячеек, начиная с данного адреса.

При адресации по базе исполнительный адрес вычисляется путем сложения содержимого регистра BX или BP и сдвига, измеряемого в байтах. Например, команды выполняют одно и тоже действие:

MOV AX, [BX]+4 ; загрузить в регистр AX содержимое по адресу,
MOV AX, [BX+4] ; отстоящему на 4 байта от ячейки,
MOV AX, 4[BX] ; исполнительный адрес которой находится в регистре BX

При прямой адресации с индексированием исполнительный адрес вычисляется как сумма значений сдвига и индексного регистра SI или DI. Этот тип адресации удобен для доступа к элементам таблицы, когда сдвиг указывает на начало таблицы, а индексный регистр – на ее элемент. Например,

MOV DI, 2 ; загрузить в AL третий элемент таблицы TABLE
MOV AL, TABLE[DI] ;

При адресации по базе с индексированием исполнительный адрес вычисляется как сумма значений базового и индексного регистров и, возможно, сдвига. Этот метод удобен при адресации двумерных массивов, когда базовый регистр содержит начальный адрес массива, а значение сдвига и индексного

регистра определяют смещение по строке и столбцу. Например, допустимые форматы команды

MOV AX,[BX+2+DI]

MOV AX,[BX+2][DI]

MOV AX,[BX][DI+2]

При всех способах адресации (кроме тех, где используется регистр BP) исполнительный адрес вычисляется относительно регистра сегмента DS. Для регистра BP регистром сегмента служит регистр сегмента стека SS. Для изменения сегментных регистров, принятых по умолчанию, можно использовать операцию изменения префикса сегмента. Например, команда

MOV ES:[BX], DX

пересылает слово из регистра DX в ячейку памяти в сегменте, адресуемом текущим содержимым сегментного регистра ES со смещением, находящемся в регистре BX.

осуществляют обмен информацией между регистрами, ячейками памяти.

Команды пересылки данных делятся на 4 группы:

- команды общего назначения;
- команды ввода/вывода;
- команды пересылки адреса;
- команды пересылки флагов.

Рассмотрим наиболее распространенные из них.

Команда MOV. Это основная команда общего назначения. Она позволяет переслать:

- байт или слово между регистрами или между регистром и ячейкой памяти;
- непосредственно адресуемое значение в регистр или ячейку памяти.

Формат команды:

MOV приемник, источник

Примеры:

MOV AX, TABLE ; переслать из памяти в регистр

MOV TABLE, AX ; переслать из регистра в память

MOV CL, 25 ; переслать в регистр CL значение 25

Запрещается:

- непосредственная пересылка данных из одной ячейки памяти в другую. Для такой пересылки необходимо использовать промежуточный регистр общего назначения;
- загружать непосредственно адресуемый операнд в регистр сегмента. Такую пересылку необходимо делать через промежуточный регистр общего назначения;
- непосредственно пересылать значение одного регистра сегмента в другой;
- использовать регистр CS в качестве приемника.

Команда LEA. Это команда загрузки исполнительного адреса. Она

пересылает относительный адрес (смещение) ячейки памяти в 16-битовый регистр общего назначения, регистр указателя или индексный регистр. Формат команды:

LEA регистр,память

Примеры:

LEA BX, TABLE[DI] ; если регистр DI содержит 5, то в регистре BX будет смещение ячейки TABLE+5 в сегменте, адресуемом текущим значением регистра DS

Команды PUSH и POP. Команда PUSH помещает содержимое регистра или ячейки памяти размером в слово в вершину стека. Формат команды:

PUSH источник

Примеры:

PUSH SI

PUSH CS

PUSH TABLE[BX][DI]

Команда POP извлекает слово с вершины стека и помещает его в ячейку памяти или регистр. Формат команды:

POP приемник

Примеры:

POP AX

Под вершиной стека понимается ячейка памяти в сегменте стека, смещение которой содержится в указателе SP. SP всегда указывает на слово, помещенное в стек последним. Команда PUSH уменьшает значение SP на 2, а команда POP – увеличивает на 2.

Команды PUSHF и POPF. Команда PUSHF помещает содержимое регистра флагов в вершину стека. Формат команды:

PUSHF

Команда POPF извлекает слово с вершины стека и помещает его в регистр флагов. Формат команды:

POPF

Дана строка из четырех символов. Необходимо осуществить круговую перестановку символов строки.

```
STACKSG SEGMENT PARA STACK
```

```
    DB 64 DUP(?)
```

```
STACKSG ENDS
```

```
DATASG  SEGMENT PARA 'DATA'
```

```
STR1    DB '1234'
```

```
STR2    DB 4 DUP(' ')
```

```
DATASG  ENDS
```

```
CODESG  SEGMENT PARA 'CODE'
```

```
ASSUME  CS:CODESG,DS:DATASG,SS:STACKSG
```

```
ENTRY   PROC FAR
```

```
; Стандартная часть
```

```

    PUSH DS
    SUB AX,AX
    PUSH AX
    MOV AX,DATASG
    MOV DS,AX
;
    MOV DX,4 ; общее количество перестановок
; Переслать первый символ из STR1 в STR2
M1:
    LEA DI,STR1 ; загрузить в DI смещение первого байта из STR1
    LEA SI,STR2 ; загрузить в SI смещение первого байта из STR3
    MOV CX,3
    MOV AL,[DI] ; переслать в AL первый байт из STR1
    MOV [SI]+3,AL ; переслать в последний байт из STR2 содержимое
                    AL
    INC DI      ; DI=DI+1 - следующий символ из STR1
;
; переслать остаток строки STR1 в STR2
M2:
    MOV AL,[DI] ; в AL следующий символ из STR1
    MOV [SI],AL ; переслать AL в очередной (с первого) байт в STR2
    INC DI      ; DI=DI+1 - следующий символ из STR1
    INC SI      ; SI=SI+1 - следующий символ из STR2
    LOOP M2     ; идти на M2
;
; переслать STR2 в STR1
    LEA DI,STR1
    LEA SI,STR2
    MOV CX,4 M3:
    MOV AL,[SI]
    MOV [DI],AL
    INC DI
    INC SI
    LOOP M3
;
    DEC DX
    CMP DX,00 ; Все перестановки сделаны?
    JNE M1    ; Нет - идти на M1
    RET
ENTRY ENDP
CODESG ENDS
END ENTRY

```

1. Используя текстовый редактор, создать исходный модуль программы Prog_5 с помощью шаблона, приведённого ниже. Начальные значения переменных A, B, C, D взять из таблицы 5.1 в соответствии с вариантом. В исходный модуль добавить недостающие комментарии.

Таблица 5.1 - Начальные значения переменных для программы Prog_5

№ варианта	Значения переменных				№ варианта	Значения переменных			
	A	B	C	D		A	B	C	D
1	3	9	2Eh	AAh	9	32	6	9h	eh
2	5Ah	2	42	9	10	22h	32	25	10h
3	B5h	55h	15	8	11	32	C1h	6	21
4	22h	7	8	12	12	3Bh	10	12h	9
5	15	1Ah	1Fh	6	13	3Bh	1Fh	11	12
6	3	1Eh	12	22h	14	5	8	10h	0Fh
7	7h	12	1Dh	9	15	12h	12	05h	9
8	5	2Eh	18h	11	16	9	1Ch	8	10h

;Program_5 – Команды передачи данных, вариант 16

```

Data SEGMENT                                ;Открыть сегмент данных
    A DB ?                                  ;Зарезервировать место
    B DB ?                                  ;в памяти для
    C DB ?                                  ;переменных
    D DB ?                                  ;A, B, C, D
Data ENDS                                    ;Закрыть сегмент данных
Ourstack SEGMENT Stack                      ;Открыть сегмент стека
    DB 100h DUP (?)                         ;Отвести под стек 256 байт
Ourstack ENDS                               ;Закрыть сегмент стека
ASSUME CS:Code, DS:Data, SS:Ourstack        ;Назначить сегментные регистры
Code SEGMENT                                ;Открыть сегмент кодов
Start: mov AX, Data                          ;Инициализировать          1
      mov DS, AX                             ;сегментный регистр DS        2
      mov A, 9                               ;Инициализировать          3
      mov B, 1Ch                             ;переменные A, B, C, D      4
      mov C, 8                               ;значениями Вашего         5
      mov D, 10h                             ;варианта                   6
      mov AL, A                              7
      mov AH, B                              8
      xchg AL, AH                            9
      mov BX, 3E10h                          10
      mov CX, BX                             11
      push BX                                12

```

push CX		13
push AX		14
lea SI, C		15
mov AX, SI		16
lea DI, D		17
mov BX, DI		18
pop AX		19
pop CX		20
pop BX		21
mov BX, AX		22
mov A, AL		23
mov B, AH		24
mov C, 0		25
mov AX, 4C00h	;Завершить программу	26
int 21h	;с помощью DOS	27
Code ENDS	;Закрыть сегмент кодов	28
END Start	;Конец исходного модуля	29

2. Создать исполняемый модуль программы Prog_5.exe выполнив этапы асемблирования и компоновки.

3. Запустить программу на выполнение.

4. В окне CPU произвести трассировку программы (пошаговое выполнение). На каждом шаге контролируйте содержимое регистров, флагов и состояние стека.

Лабораторная работа № ____

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

1. Как записываются общие команды передачи данных на Ассемблере?
2. Что может использоваться в качестве операндов команды?
3. Для чего предназначена команда LEA и что является ее операндами?
4. Поясните выполнение команд работы со стеком.

5. Поясните выполнение команды обмена данными.
6. Как загрузить отлаживаемую программу?
7. Как осуществляется изменение содержимого оперативной памяти и регистров средствами отладчика?
8. Каким образом можно редактировать ассемблерную программу?
9. В каком окне можно наблюдать результат выполнения программы?
10. Что такое трассировка программы и как она осуществляется в отладчике?

Костров, Б. В. Микропроцессорные системы и микроконтроллеры / Б. В. Костров, В. Н. Ручкин. – М.: ТехБук (Десс), 2007.-320с.

Максимов, Н.В. Архитектура ЭВМ и вычислительных систем: учебник. – М.: ФОРУМ-ИНФРА-М, 2005.-512с.

Юров, В.И. Assembler: учебник / В.И. Юров – СПб.: Питер, 2008.-637с.

Финогенов, К. Г. Основы языка Ассемблера [Текст] / К. Г. Финогенов. – М.: Радио и связь, 2000.

Финогенов, К. Г. Использование языка Ассемблера [Текст]: учеб. пособие для вузов / К. Г. Финогенов. – М.: Горячая линияТелеком, 2004.