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

Учебная дисциплина «Встраиваемые микропроцессорные системы»

Инструкция

по выполнению лабораторной работы
«Программирование на языке Ассемблер. Составление и отладка программ с командами арифметических операций»

Минск

2017

Лабораторная работа № 7

Тема работы: «Программирование на языке Ассемблер. Составление и отладка программ с командами арифметических операций»

1. **Цель работы**:

Формирование практических навыков по составлению и отладке программ с командами арифметических операций

2. Задание

Изучить команды арифметических операций. Написать на языке ассемблера и выполнить линейную программу для вычисления алгебраического выражения с однобайтовыми и двухбайтовыми числами.

3. Оснащение работы

Техническое задание, ПК, эмулятор DOSBox.

4. Основные теоретические сведения

Особенности составления линейных программ.

Программирование линейных алгоритмов на языке ассемблера относительно просто. Хотя практически полезные программы чисто линейного типа едва ли встречаются, линейные участки найдутся в каждой программе.

4.1. Типовое обрамление программ

При работе с программами используются специальные файлы обрамления, содержащие все необходимое для сопряжения создаваемой программы с системой DOS. В разных случаях полезны несколько отличные версии обрамления. Для этой работы и для многих других используется обрамление, соответствующее файлу EXE_FORM.ASM. Вот содержимое этого файла с некоторыми пояснениями.

```
TITLE   PROGNAME (EXE)
;----------------------------------------
STACKSG SEGMENT PARA STACK 'Stack'
      DW    32 DUP(?)
STACKSG     ENDS
;----------------------------------------
DATASG  SEGMENT PARA 'Data'
;         ... |
;         ... | здесь должны быть ваши определения данных
;         ... |
```

```
        UVAR    DW (?)                  ; выходная переменная
        ASCVAL  DB '      ','$'         ; шаблон вывода
        EXIT    DW (?)                  ; выход по любой клавише
DATASG  ENDS
;---------------------------------------
CODESG  SEGMENT PARA 'Code'
BEGIN   PROC    FAR
        ASSUME  CS:CODESG, DS:DATASG, SS:STACKSG
        PUSH    DS
        XOR     AX,AX                   ; запись в стек
        PUSH    AX                      ;    нулевого адреса
        MOV     AX,DATASG               ; засылка адреса
        MOV     DS,AX                   ;    DATASG в регистр DS
        LEA     SI,ASCVAL
;---------------------------------------
;         ... |
;         ... | здесь должны быть команды ваших вычислений
;         ... |
        MOV     UVAR,AX                 ; запоминание результата
;---------------------------------------
; вызов подпрограммы вывода результата
;---------------------------------------
        CALL    SCRLIN                  ; вызов подпрограммы
        RET                             ; завершение программы
BEGIN   ENDP
;
;***************************************
; |
; | пропущенная часть обрамления
; |
;***************************************
CODESG  ENDS
        END     BEGIN
```

Обрамление содержит:

- определение сегмента стека, оформленное директивами STACKSG SEGMENT и STACKSG ENDS; сам сегмент состоит из 32-х слов памяти, что задается директивой DW 32 DUP(?);
- определение сегмента данных, оформленное директивами DATASG SEGMENT и DATASG ENDS;
- определение сегмента кода, оформленное директивами CODESG SEGMENT и CODESG ENDS.

В обрамлении сегмента кода участвуют директива ASSUME CS:CODESG,DS:DATASG,SS:STACKSG и две небольшие группы команд. Начальная группа обеспечивает сопряжение с DOS, требуемое для правильного входа в программу и для возврата из нее в DOS. Назначение отдельных команд объяснено в комментариях, хотя для полного понимания следует обратиться к справочной информации по командам.

За этой группой команд отточиями показано место, куда нужно записать команды, выполняющие действия, требуемые от разрабатываемой программы.

Затем идет очень небольшая группа команд, обеспечивающая запоминание результата, за которой перед самым концом программного кода стоит команда RET, необходимая для корректного завершения программы и для возврата в DOS. Но после нее здесь выпущен довольно большой кусок кода – текст подпрограммы, обеспечивающей вывод на экран числового результата выполнения программы и некоторые вспомогательные действия. Конечно, в самом обрамлении все это есть, а здесь пропущено потому, что сейчас разбираться с этим преждевременно.

Дальше – перед самым концом сегмента стоит команда RET, обеспечивающая возврат из подпрограммы в вызвавшую ее главную программу.

Необходимость обрамления связана с тем, что для правильных запуска и работы программы в DOS эта программа должна иметь, с точки зрения языка ассемблера, небольшой набор команд и директив в самом ее начале и в самом конце. Каждый раз набирать одно и то же в создаваемой программе нецелесообразно и к тому же чревато появлением программных ошибок. Именно здесь помогут типовые обрамления. Файл обрамления следует скопировать с измененным именем соответственно создаваемой программе и как бы дополнить, вставив в определенные места обрамления определения данных (констант и переменных) и программный код.

Конечно, при необходимости можно внести изменения в само обрамление и даже написать собственное обрамление под свои нужды.

4.2. Пример программы для 16-битовых вычислений математического выражения.

Программные тексты вставлены в обрамление – с пропуском его части, о чем было сказано при описании обрамления.

```
TITLE   ALGLIN (EXE)                    ; вызов программы
;---------------------------------------
STACKSG SEGMENT PARA STACK 'Stack'
      DW    32 DUP(?)
STACKSG     ENDS
;---------------------------------------
DATASG  SEGMENT PARA 'Data'
; на месте трех вопросительных знаков нужно указать
;   конкретные числовые значения
        UVAR    DW (?)                   ; переменная U - неопределенная
        XVAR    DW  2                   ; переменная X
        YVAR    DW  2                   ; переменная Y
        ZVAR    DW  2                    ; переменная Z
        ASCVAL  DB '      ','$'          ; шаблон вывода строки на экран
        DIVCON  DW 2                     ; константа-знаменатель
        AUX     DW (?)                   ; переменная вспомогательная
        AUX1    DW (?)                   ; - для 1-го слагаемого числителя
        AUX2    DW (?)                   ; - для 2-го слагаемого числителя
        EXIT    DW (?)                   ; выход по любой клавише
DATASG  ENDS
```

```
;---------------------------------------
CODESG  SEGMENT PARA 'Code'
BEGIN   PROC    FAR
        ASSUME  CS:CODESG, DS:DATASG, SS:STACKSG
        PUSH    DS
        XOR     AX,AX                   ; записать в стек
        PUSH    AX                      ;   нулевой адрес
        MOV     AX,DATASG               ; поместить адрес
        MOV     DS,AX                   ;   DATASG в регистр DS
        LEA     SI,ASCVAL
;---------------------------------------
; Программа вычисления выражения
;
; U = ((X + Y - Z)^3  - (X - Y + Z)^2  + (X + Y + Z))/2
;
; символ '^' означает возведение в степень
;---------------------------------------
; вычисление первого слагаемого числителя
;---------------------------------------
        MOV     AX,XVAR                 ; вычисление (X+Y-Z)
        ADD     AX,YVAR
        SUB     AX,ZVAR
        MOV     AUX,AX                  ; запоминание
                                        ;   вспомогательной переменной
        IMUL    AUX                     ; возведение в квадрат
        IMUL    AUX                     ; возведение в куб
        MOV     AUX1,AX                 ;запоминание первого слагаемого
;---------------------------------------
; вычисление второго слагаемого числителя
;---------------------------------------
        MOV     AX,XVAR                 ; вычисление (X-Y+Z)
        SUB     AX,YVAR
        ADD     AX,ZVAR
        MOV     AUX,AX                  ; запоминание
                                        ;   вспомогательной переменной
        IMUL    AUX                     ; возведение в квадрат
        MOV     AUX2,AX                 ; запоминание второго слагаемого
;---------------------------------------
; вычисление третьго слагаемого числителя
;---------------------------------------
        MOV     AX,XVAR                 ; вычисление (X+Y+Z)
        ADD     AX,YVAR
        ADD     AX,ZVAR
;---------------------------------------
; вычисление числителя и результата
;---------------------------------------
        SUB     AX,AUX2                 ; вычитание второго слагаемого из первого
        ADD     AX,AUX1                 ; добавление третьего слагаемого
        CWD                             ; преобразование слова
                                        ;   в двойное слово - для делимого
        IDIV    DIVCON                  ; вычисление результата делением на 2
        MOV     UVAR,AX                 ; запоминание результата для вывода
;---------------------------------------
; вызов подпрограммы вывода результата
;---------------------------------------
```

```
     ; вызов подпрограммы вывода результата
;---------------------------------------
        CALL    SCRLIN                  ; вызов подпрограммы
        RET                             ; завершение программы
BEGIN   ENDP
;
;***************************************
SCRLIN  PROC    NEAR
;---------------------------------------
; очистка экрана
;---------------------------------------
        MOV     AX,0600H
        MOV     BH,07
        MOV     CX,0000
        MOV     DX,184FH
        INT     10H
;---------------------------------------
; установка курсора
;---------------------------------------
        MOV     AH,02
        MOV     BH,00
        MOV     DX,0000
        INT     10H
;---------------------------------------
; преобразование BIN в ASCII
;---------------------------------------
        MOV     CX,10
        MOV     AX,UVAR
        CMP      AX,0000
        JNS     L10
        MOV     BH,2DH
        MOV     [SI],BH
        NEG     AX
L10:    ADD     SI,5
L15:    CMP     AX,10
        JB      L20
        XOR     DX,DX
        DIV     CX
        OR      DL,30H
        MOV     [SI],DL
        DEC     SI
        JMP     L15
L20:    OR      AL,30H
        MOV     [SI],AL
;---------------------------------------
; вывод значения ASCVAL на экран
;---------------------------------------
        MOV     AH,09
        LEA     DX,ASCVAL
        INT     21H
;---------------------------------------
; задержка до нажатия клавиши
;---------------------------------------
        MOV     AH,00
        INT     16H
```

```
        RET
SCRLIN  ENDP
;***************************************
CODESG  ENDS
        END     BEGIN
```

4.3. Ассемблирование и компоновка программы

По окончании работы с текстом программы нужно проделать два шага для получения исполняемого файла программы: ассемблировать программу и затем выполнить ее компоновку.

Первый шаг включает в себя ассемблирование исходного текста программы в объектный код и генерацию OBJ-модуля. Для ассемблирования исходной программы следует запустить программу-ассемблер с необходимыми ключами:

TASM.EXE /la /z /zi PROG.ASM

где PROG.ASM – условное имя исходной программы.

Эти ключи означают:

/la – расширенная форма распечатки программы (в файле PROG.LST);

/z – распечатывание исходных строк вместе с сообщениями об ошибках;

/zi – распечатывание полной отладочной информации.

Если ассемблер не обнаруживает ошибок, то он создает объектный модуль. Формат OBJ-модуля уже приближен к исполнимой форме, но еще не готов для выполнения. Этот модуль содержит машинный код в шестнадцатеричной форме, не приспособленный для выполнения - поскольку ассемблер обычно не может определить все машинные адреса, так как исполнимая программа может загружаться для выполнения почти в любое место оперативной памяти. Кроме того, для объединения с основной программой могут использоваться другие программы или подпрограммы. Компоновщик завершает определение адресных ссылок и, если требуется, объединяет нескольких программ в один исполняемый файл. Шаг компоновки завершает преобразование OBJ-модуля в исполнимый EXE-модуль. Для компоновки программы следует запустить компоновщик следующим образом:

TLINK.EXE PROG.OBJ, PROG

где PROG.OBJ – имя OBJ-модуля ассемблированной программы, а PROG – имя исполняемой EXE-программы (без расширения).

Подробнее с ключами обеих программ можно ознакомиться, запустив TASM.EXE или TLINK.EXE без параметров.

После компоновки OBJ-модуля (или модулей) полученный исполняемый EXE-модуль можно выполнять любое число раз.

Итак:

- для ассемблирования программы нужна команда:

TASM.EXE /la /z /zi PROG.ASM

где PROG.ASM - условное имя исходной программы.

- для компоновки программы нужна команда:

TLINK.EXE PROG.OBJ, PROG-EXE

где PROG.OBJ – имя OBJ-модуля ассемблированной программы, а PROG-EXE – имя исполнимой EXE-программы (без расширения).

Подробнее с опциями обеих программ можно ознакомиться, запустив TASM.EXE или TLINK.EXE без параметров.

5. Порядок выполнения работы

Задание 1.

1. Скопировать из указанного каталога текст обрамления программы EXE_FORM.ASM, изменив его имя по усмотрению, в свой рабочий каталог.
2. Ввести в этот текст нужные дополнения согласно номеру своего варианта. Вставляемая часть программы должна выполнять вычисление заданного алгебраического выражения. При этом преобразование выражения делать не нужно. Вариант выбирается соответственно своему номеру по списку в учебном журнале.
3. Вставить в текст программы значения нужных переменных по собственному выбору. Если программа содержит команды деления, следует подобрать эти значения так, чтобы деление давало целочисленный результат. Нужно следить, чтобы все исходные и промежуточные значения не превышали пределов однобайтовых чисел.
4. Ассемблировать программу и убедиться в отсутствии ошибок.
5. Скомпоновать программу.
6. Выполнить полученный EXE-модуль и проверить результат вычислений. Записать результат вычислений в отчет.

Задание 2.

1. Скопировать текст своей предыдущей программы в свой рабочий каталог, соответственно изменив имя файла.
2. Ввести в этот текст нужные дополнения и изменения.
3. Вставить в текст программы значения нужных переменных по собственному выбору. Если программа содержит команды деления, следует подобрать эти значения так, чтобы деление давало целочисленный результат. Нужно следить, чтобы все исходные и промежуточные значения не превышали пределов двухбайтовых чисел.
4. Ассемблировать программу и убедиться в отсутствии ошибок.
5. Скомпоновать программу.
6. Выполнить полученный EXE-модуль и проверить результат вычислений. Записать результат вычислений в отчет.

Текст измененной программы составляется на основе текста предыдущей программы с добавкой нужных изменений

- директивы определения байта, задающие значения переменных и констант, заменить директивами определения слова;
- перед делением вставить команду преобразования слова в двойное

слово CWD;

- поскольку результирующая переменная есть двухбайтовое число, никакие ее преобразования не нужны.

В остальном текст программы не отличается от прежнего.

Варианты заданий:

01. $Y = (A * X^2 - B * X + C) / D$
02. $U = (A * X + B * X^2) + C * X / D$
03. $U = (X^3 + Y^2 + 1) / (X - 1)$
04. $U = (A * X - B * Y) + (X / A + Y / B)$
05. $U = A * X^2 - (X * Y^2) / B$
06. $U = (X + Y)^2 / (X + 1)^2$
07. $U = (X - Y)^2 + (X + Y)^2 / (X - Y)$
08. $U = X * (X - 1) * (X + 2) / (X - 3)$
09. $U = ((A * X^2) + (A^2 * X) / 2$
10. $U = (X^3 - Y^3) + (X^2 - Y^2) + X / Y$
11. $U = (A * X^2 - B * Y^2) / (A * B)$
12. $U = (X - Y + Z) * (X + Y - Z) / (X + Y + Z)$
13. $U = (A * (X + 1)^2) / (X^2 + 1)$
14. $U = (X^2 - A)^3 + (X^2 + 1) / 2$
15. $U = (A * X^2 - B * X + C) * (A / (B + C))$

6. Форма отчета о работе

Лабораторная работа № ___
Номер учебной группы _____
Фамилия, инициалы учащегося _____
Дата выполнения работы _____
Тема работы: _____
Цель работы: _____
Оснащение работы: _____
Индивидуальное задание на работу.
Указание имен исходного и исполняемого файлов
Результат выполнения работы: _____
_____
_____

Отчет представляется в виде текстового файла. К отчету должны прилагаться файл исходного кода Программы и рабочий исполняемый файл.

7. Контрольные вопросы и задания

1. Формат команды «сложить», ее операнды.
2. Формат команды «вычесть», ее операнды.
3. Формат команды «умножить», ее операнды.
4. Формат команды «делить», ее операнды.

5. Каков диапазон беззнаковых чисел допустим в программах 16-тиразрядного микропроцессора?
6. Каков диапазон чисел со знаком допустим в программах 16-ти разрядного микропроцессора?
7. Какую информацию содержат арифметические флаги операций?
8. Какие флаги устанавливаются при выполнении команд «сложить» и «вычесть».
9. Какие флаги устанавливаются при выполнении команд «умножить» и «делить».
10. Как выполнить сложение (вычитание) двух операндов, находящихся в памяти?
11. Как выполнить умножение двух операндов, находящихся в памяти?
12. Как выполнить деление двух операндов, находящихся в памяти?
13. С числами какой системы счисления может работать Ассемблер?
14. Каким образом можно проконтролировать значения переменных при отладке программы?
15. Как можно контролировать область данных программы, загруженной в память?
16. Как можно контролировать область стека программы, загруженной в память?
17. Найдите ошибки в нижеприведенных командах:

MOV AL, E4h; ADD 64, BL; MUL 3Fh; MOV DS, 3F3Fh.

8. Рекомендуемая литература

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.

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 линия Телеком, 2004.

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