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 циклических программ»

Минск

2017

Лабораторная работа № 11

Тема работы: «Программирование на языке Ассемблер. Составление и отладка циклических программ»

1. **Цель работы**:

Получение практических навыков составления и выполнения циклических программ

2. Задание

Написать на языке ассемблера и выполнить циклическую программу для вычисления заданного алгебраического выражения. Программа должна работать с двухбайтовыми числами.

3. Оснащение работы

Техническое задание, ПК, эмулятор DOSBox.

4. Основные теоретические сведения

Особенности программирования циклических алгоритмов на языке ассемблера.

К таковым относится применение команд цикла. Хотя в случае более общих форм цикла взамен последней иногда приходится оформлять цикл с помощью команд условного перехода.

Циклы в программе можно построить с помощью команд условных переходов. Цикл организуется как блок кода, завершаемый условным переходом, благодаря чему блок может выполняться повторно до выполнения условия завершения.

Циклы служат многим целям и применяются для работы с массивами, для проверки состояния портов ввода-вывода, для очистки блоков памяти, для чтения символьных цепочек с клавиатуры и для вывода их на экран, и пр. Циклы дают основное средство, используемое для выполнения повторяемых действий. Поэтому они часто встречаются, и в наборе команд процессора предусмотрено несколько специальных команд цикла: LOOP, LOOPNE, LOOPE и JCXZ.

Рассмотрим сначала команду LOOP. Пусть нужно вывести 17 символов текстовой цепочки TestString. Это можно сделать так:

```
            . . .
            .DATA
TestString        DB   'Это проверка! ...'
            . . .
            .CODE
```

```
            . . .
            MOV   CX,17
            MOV   BX,OFFSET TestString
 PrintStringLoop:
            MOV   DL,[BX]          ; получить следующий символ
            INC   BX               ; ссылка на следующий символ
            MOV   AH,2             ; назначить функцию вывода на экран
            INT   21H              ; вызвать функцию DOS для вывода символа
            DEC   CX               ; уменьшить счетчик длины цепочки
            JNZ   PrintStringLoop  ; обработать следующий символ,
                                   ;   если он есть
            . . .
```

Примечание. Здесь использованы директивы TASM для режима Ideal.

Но есть лучший способ. Регистр CX как счетчик полезен для организации циклов.

Команда

LOOP PrintStringLoop

делает то же, что команды:

DEC CX

JNZ PrintStringLoop

но выполняется быстрее и занимает на один байт меньше. Всякий раз, как нужно организовать цикл, пока значение счетчика не станет равным 0, начальное значение счетчика записывается в регистр CX и используется команда LOOP.

Для циклов с более сложным условием завершения предусмотрены команды LOOPE и LOOPNE.

Команда LOOPE работает так же, как LOOP, но цикл завершится, если регистр CX примет значение 0 или если будет установлен флаг нуля (он устанавливается, если результат последней арифметической операции был нулевым или если два операнда в последней операции сравнения не совпали). Аналогично, команда LOOPNE завершает выполнение цикла, если регистр CX принял значение 0 или флаг нуля сброшен.

Пусть нужно повторять цикл, сохраняя коды нажатых клавиш, пока не будет нажата клавиша ввода или не будет накоплено 128 символов. Для этого можно написать такую программу:

```
            . . .
            .DATA
 KeyBuffer      DB   128 DUP (?)
            . . .
            .CODE
            . . .
            MOV   CX,128
            MOV   BX,OFFSET KeyBuffer
KeyLoop:
            MOV   AH,1             ; функция DOS ввода с клавиатуры
            INT   21H              ; считать следующую клавишу
            MOV   [BX],AL          ; сохранить ее
```

```
        INC   BX                ; установить указатель для
                                ;   следующей клавиши
        CMP   AL,0dH            ; это клавиша ENTER?
        LOOPNE KeyLoop          ; если нет, то получить следующую
                                ;   клавишу, пока не достигнуто
                                ;   максимальное число клавиш
        . . .
```

Команде LOOPE эквивалентна команда LOOPZ, команда LOOPNE – LOOPNZ (так же как команда JE эквивалентна команда JZ): это – команды-синонимы.

Есть еще одна команда цикла – JCXZ. Команда JCXZ выполняет переход, только если значение регистра CX равно 0. Это дает удобный способ проверить регистр CX перед началом цикла. Например, в следующем фрагменте программы, при обращении к которому регистр BX указывает на блок байтов, которые нужно обнулить, команда JCXZ используется для пропуска тела цикла в том случае, если регистр CX имеет значение 0:

```
        . . .
        JCXZ   SkipLoop         ; если CX имеет значение 0, то
                                ;   ничего не делать
ClearLoop:
        MOV   BYTE PTR [SI],0   ; очистить следующий байт
        INC   SI                ; ссылка на следующий очищаемый
                                ;   байт
SkipLoop:
        . . .
```

Если значение регистра CX равно 0, то выполнение цикла желательно пропустить, потому что иначе значение CX будет уменьшено до величины 0FFFFH и команда LOOP выполнит переход на указанную метку. После этого цикл будет выполняться 65535 раз. Команда JCXZ позволяет быстро и эффективно выполнить нужную проверку.

О командах циклов стоит сделать несколько замечаний:

- Команды циклов, как и условных переходов, могут выполнять переход лишь на метку, отстоящую от команды не более чем на 128 байтов в ту или другую сторону. Циклы, превышающие 128 байтов, требуют выбора безусловных переходов с помощью условных и безусловных переходов.
- Команды циклов не влияют на состояния флагов. Это значит, что команда

LOOP LoopTop

не эквивалентна в точности командам

DEC CX

JNZ LoopTop

поскольку команда DEC изменяет флаги переполнения, знака, нуля, дополнительного переноса и четности, а команда LOOP на флаги не влияет. К тому же использование команды DEC не эквивалентно варианту

```
SUB   CX,1
JNZ   LoopTop
```

поскольку команда SUB влияет на флаг переноса, а команда DEC – нет. Различия невелики, но при программировании на языке ассемблера важно понимать, какие именно флаги устанавливаются либо нет конкретной командой.

5. Порядок выполнения работы

1. Скопировать из указанного каталога текст обрамления программы EXE_FORM.ASM, изменив ему имя по усмотрению, в свой рабочий каталог.
2. Ввести в этот текст нужные дополнения и изменения по заданию 1.
3. Вставить в текст программы значения нужных переменных по собственному выбору.
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.
7. Для выполнения задания 2 скопировать исходный текст программы задания 1 под новым именем (по выбору). Внести в этот текст нужные дополнения и изменения по заданию 2.
8. Вставить в текст программы значения нужных переменных по собственному выбору.
9. Ассемблировать программу и убедиться в отсутствии ошибок.
10. Скомпоновать программу.
11. Выполнить полученный EXE-модуль и проверить результат вычислений. Записать результат вычислений в отчет.

Варианты заданий:

Задание 1. Накопить в цикле сумму чисел заданного вида в переменной U. Деление целочисленное, остатком пренебречь. Если указано значение X и число повторений, на каждом цикле увеличивать значение X на единицу.

01. U = A*X + B/X - C, X = 3 ... 11, A = 3, B =12 , C = 7
02. U = A/X + B*X - C, X = 4 ... 10, A = 24, B = 2, C = 3
03. U = A*X - B/X + C, X = 5 ... 13, A = 2, B = 15, C = 5
04. U = A/X - B*X + C, X = 2 ... 9, A = 5, B = 3, C = 4
05. U = A*X + B/X - C, X = 2 при числе повторений 7, A = 6, B = 16, C = 5
06. U = A/X + B*X - C, X = 3 при числе повторений 9, A = 12, B = 3, C = 6
07. U = A*X - B/X + C, X = 5 при числе повторений 6, A = 7, B = 12, C = 4
08. U = A/X - B*X + C, X = 4 при числе повторений 8, A = 8, B = 3, C = 11
09. U = A*X + B/X - C, X = 7 ... 11, A = 3, B = 15, C = 5
10. U = A/X + B*X - C, X = 4 ... 12, A = 12, B = 5, C = 8
11. U = A*X - B/X + C, X = 3 ... 9, A = 2, B = 24, C = 6
12. U = A/X - B*X + C, X = 5 ... 10, A = 14, B = 4, C = 7

13. U = A*X + B/X - C, X = 4 при числе повторений 10, A = 2, B = 9, C = 3
14. U = A/X + B*X - C, X = 3 при числе повторений 7, A = 10, B = 4, C = 5
15. U = A*X - B/X + C, X = 5 при числе повторений 5, A = 3, B = 12, C = 7

Задание 2 - усложненное: требует цикла с ветвлениями. Взяв за основу свое задание 1, доработать его так, чтобы при суммировании учитывались лишь те подвыражения с делением, где деление выполнилось нацело (то есть остаток нулевой).

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. Какие команды могут использоваться для организации циклов?
2. Какова максимальная длина переходов при организации циклов?
3. Какие признаки, кроме CX=0, могут быть использованы при организации циклов?
4. Как микропроцессор выполняет команду LOOP?
5. Как осуществляется переход к процедурам разных типов?
6. Назовите варианты команды возврата из процедуры.
7. В чем состоит разница кодов строчных и заглавных символов английского алфавита?
8. Каким образом можно заменить код строчной буквы на код заглавной?
9. Каким образом можно заменить код заглавной буквы на код строчной?
10. Как можно автоматически вычислять длину символьной строки?
11. Какие правила следует соблюдать при организации циклов на Ассемблере?

8. Рекомендуемая литература

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 [Текст]: учеб. пособие для вузов / В. И. Юров. 2-е изд. – СПб.: Питер, 2007.