11. Подпрограммы. Передача параметров через регистры.

В отличии от языков программирования высокого уровня, использование подпрограмм в ассемблерных программах требует большей ручной работы. Прежде всего, необходимо оформить подпрограмму как логически законченный именованный набор команд. Для этого используются две директивы PROC и ENDP, причем в каждой их них в поле метки указывается имя подпрограммы:

```
ИмяПП   PROC
        . . . . . .
        тело подпрограммы
        . . . . . .
ИмяПП   ENDP
```

Для вызова подпрограммы и возврата из нее используются две специальные команды:

```
CALL  ИмяПП   ; вызов (размещается в вызывающей программе)
RET           ; возврат в точку вызова (размещается в подпрограмме)
```

Очень важно понимать, какие действия выполняют обе эти команды. Команда CALL:

- помещает адрес следующей за ней команды в стек (PUSH EIP), сохраняя тем самым адрес возврата из подпрограммы
- выполняет безусловный переход по адресу точки входа в подпрограмму, т.е. по адресу первой выполняемой команды (JMP точка_входа)

Команда RET:

- считывает с вершины стека адрес возврата и помещает его в регистр EIP, что приводит к переключению процессора на выборку команд по этому адресу (POP EIP)

Использование стека для сохранения адресов возврата позволяет организовать вложенные вызовы подпрограмм, в том числе и рекурсивные.

Схематично использование подпрограмм можно представить следующим образом.

MyProc PROC . . . . . . тело подпрограммы . . . . . . RET MyProc ENDP	объявление подпрограммы
. . . . .	
CALL MyProc	вызов подпрограммы
MOV AX, BX	точка возврата
. . . . .	
CALL MyProc	вызов подпрограммы
ADD CX, 2	точка возврата
. . . . .	

Огромное значение при использовании подпрограмм имеет передача параметров между ними. При этом можно использовать два способа – через РОНы или через стек. Первый способ более быстрый, т.к. не требует обращения к памяти, но и более ограничен по своим возможностям. Второй чуть медленнее, но и более универсален.

Передача параметров через регистры

Данный способ используется, если передаваемых параметров немного. Для этого основная программа записывает перед вызовом подпрограммы необходимые значения в выбранные регистры, а подпрограмма после запуска берет их из этих регистров. Также - при возврате результата. При этом можно использовать 2 способа:

- записать в регистры сами значения
- записать в регистры адреса передаваемых данных

В любом случае для правильного взаимодействия подпрограмм должны быть известны используемые для передачи параметров регистры. Для этого в подпрограммах оформляются и документируются соглашения о регистрах.

Рассмотрим пример. Пусть в программе надо многократно находить максимум из двух чисел, что удобно оформить как подпрограмму. Начинаем с соглашения о регистрах. В подпрограмму надо передать два сравниваемых числа и вернуть максимальное из них. Используемые регистры:

- AX – первое число
- BX – второе число
- результат – в регистре AX.

Тогда объявление подпрограммы будет следующим:

```
Max2    PROC     ; заголовок, имя Max2 определяет адрес команды CMP
        CMP   AX,  BX       ; берем числа из регистров
        JGE     M1          ; если (AX) ≥ (BX), идем на выход
        MOV   AX,  BX       ; иначе в AX заносим число из BX
M1:     RET                 ; команда возврата
Max2    ENDP                ; директива завершения подпрограммы
```

Фрагмент основной программы:

```
        MOV  AX,  X     ; (AX) = первое число из области X
        MOV  BX,  Y     ; (BX) = второе число из области У
        CALL  Max2      ; вызов, после возврата (AX) = max (X, Y)
        MOV  BX, Z      ; (BX) = еще одно число, из области Z
        CALL  Max 2     ; (AX) = max (X, Y, Z)
```

Во втором способе в основной программе надо перед вызовом использовать команды загрузки адреса LEA, а в самой подпрограмме – косвенную адресацию. Основное применение этого способа – при передаче в подпрограмму больших структур данных, например массивов.

Пример. Даны 2 массива целых неотрицательных двухбайтовых чисел с разным числом элементов:

```
Mas1   DW  100   DUP  (?)
Mas2   DW   50    DUP  (?)
```

Требуется найти сумму максимальных элементов из этих массивов, оформив для поиска максимума соответствующую подпрограмму.

Соглашения по регистрам:

- BX – начальный адрес массива
- CX – количество элементов в массиве
- AX – результат работы подпрограммы

Объявление подпрограммы:

```
MaxMas   PROC
         MOV AX, 0  ; начальный максимум  (числа – неотрицательные)
M1:      CMP   [BX], AX   ;  используем косвенную адресацию
         JLE   M2
         MOV   AX, [BX]       ; замена текущего максимума
M2:      ADD   BX,  2         ; наращивание адреса элементов массива
         LOOP  M1
         RET
MaxMas   ENDP
```

Фрагмент основной программы:

```
         LEA    BX,   Mas1     ; (BX) = адрес первого массива
         MOV    CX,  100       ; установка счетчика цикла
         CALL   MaxMas         ; (AX) = максимум в первом массиве
         MOV    DX, AX         ; (AX) → DX  для сохранения
         LEA    BX,  Mas2      ; (BX) = адрес второго массива
         MOV    CX, 50         ; счетчик цикла для второго массива
         CALL   MaxMas         ; (AX) = максимум во втором массиве
         ADD    DX, AX         ; результат
```

При использовании данного способа возможны проблемы: если подпрограмма для своей работы должна использовать регистры, которые используются в основной программе, но не для передачи параметров, а для своих внутренних целей, то их содержимое в начале работы подпрограммы полагается сохранить в стеке, а в конце работы – восстановить из стека. Это дает возможность основным программам и подпрограммам использовать одни и те же регистры. Можно сохранять лишь нужные регистры, а можно – сразу все, с помощью команд PUSHA и POPA.

Практические задания к теме №11.

Задание 1. Оформить подпрограмму нахождения максимума из двух целых чисел, передаваемых через регистры. В главной программе вызвать подпрограмму два раза.

Задание 2. Оформить подпрограмму для поиска максимального элемента в массиве целых чисел. Входные параметры: адрес массива и число элементов в массиве. Использовать подпрограмму для нахождения суммы максимальных элементов в двух массивах с разным числом элементов.