19. Подпрограммы. Объявление, вызов.

Процедурой в ассемблере является все то, что в других языках называют подпрограммами, функциями, процедурами и т. д. Ассемблер не накладывает на процедуры никаких ограничений - на любой адрес программы можно передать управление командой CALL, и оно вернется к вызвавшей процедуре, как только встретится команда RET. Такая свобода выражения легко может приводить к труд- . нечитаемым программам, и в язык ассемблера были включены директивы логического оформления процедур.

Подпрограмма - именованная часть программы.

Пример объявления подпрограммы:

```
MyProc proc near ; указываем тип перехода
    ...
Myproc endp
```

CALL <операнд >

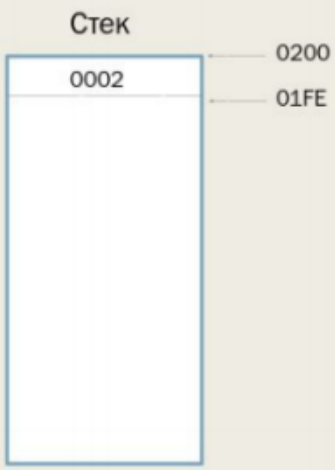
Сохраняет текущий адрес в стеке и передает управление по адресу, указанному в операнде. Операндом может быть непосредственное значение адреса (метка в ассемблерных программах), регистр или переменная, содержащие адрес перехода. Если в качестве адреса перехода указано только смещение, считается, что адрес расположен в том же сегменте, что и команда CALL. При этом, так же как и в случае с JMP, выполняется ближний вызов процедуры. Процессор помещает значение регистра EIP (IP при 16-битной адресации), соответствующее следующей за CALL команде, в стек и загружает в EIP новое значение, осуществляя тем самым передачу управления. Если операнд CALL - регистр или переменная, то его значение рассматривается как абсолютное смещение, если операнд - ближняя метка в программе, то ассемблер указывает ее относительное смещение. Чтобы выполнить дальний CALL в реальном режиме, режиме V86 или в защищенном режиме при переходе в сегмент с теми же привилегиями, процессор помещает в стек значения регистров CS и EIP (IP при 16-битной адресации) и осуществляет дальний переход аналогично команде JMP.

Перед вызовом подпрограммы вызывающая программа кладёт в стек параметры, затем выполняет команду CALL, подпрограмма сама в своём начале сохраняет текущее значение регистра SP в регистре BP и дополнительно уменьшает SP на размер области памяти, необходимый для хранения локальных переменных.

Если подпрограмма использует стек для хранения локальных переменных, ESP изменится, но EBP можно будет использовать для того, чтобы считывать Значения параметров напрямую из стека.

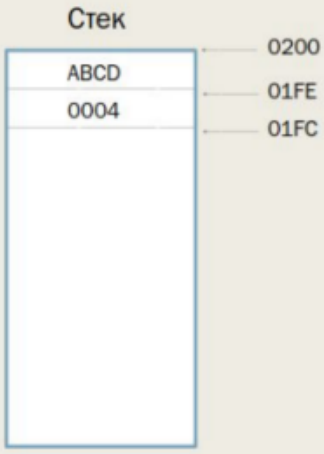
Пример вызова подпрограммы №1

```
                    0. SP = 0200
0000: CALL P1       1. SP = 01FE
0002: MOV BX, AX
…
P1:
0123: MOV AX, 5
0125: RET           2. SP = 0200
```



Пример вызова подпрограммы №2

```
                                    0. SP = 0200
0000: PUSH ABCDh ;передача параметра 1. SP = 01FE
0002: CALL P1                       2. SP = 01FC
0004: POP DX                        4. SP = 0200
0006: MOV BX, AX
…
P1:
0123: MOV BP, SP ;ss:[bp] - адрес возврата
;ss:[bp+2] - параметр
...
0223: MOV AX, 5
0225: RET                           3. SP = 01FE
```



Пример вызова подпрограммы №3

```
                                    0. SP = 0200
0000: PUSH ABCDh ;передача параметра 1. SP = 01FE
0002: CALL P1                       2. SP = 01FC
0004: MOV BX, AX
…
P1:
0123: MOV BP, SP ;ss:[bp] - адрес возврата
;ss:[bp+2] - параметр
0125: SUB SP, 10 ; ss:[bp-1 .. bp-10] - локальные переменные
                                    3. SP = 01F2
…
0221: ADD SP, 10
0223: MOV AX, 5                     4. SP = 01FC
0225: RET 2                         5. SP = 0200
```