Лекция 11

Раздел 3. Процедуры и макрокоманды

3.1 Процедуры и особенности их использования

Процедурой (или подпрограммой) называется фрагмент программы, к которому можно перейти, и из которого можно вернуться туда, откуда осуществляется переход.

Переход к процедуре называется вызовом, а переход назад - возвращением. Возвращение осуществляется к команде, следующей после вызова.

Эти функции выполняют две команды:

CALL (call a procedure) и

RET (return from procedure - вернуться из процедуры).

Команда CALL имеет формат:

CALL имя

где имя - это имя процедуры, которая вызывается.

Сама же процедура, как отмечалось, имеет вид:



Имя процедуры считается меткой, которая принадлежит к первой команде процедуры. Поэтому имя процедуры можно отмечать в командах перехода, который означает переход на первую команду процедуры.

В отличие от языков высокого уровня, имена в процедурах (переменные, метки) не локализуются внутри их, следовательно должны быть уникальными для соответствующего сегмента.

Где размещать процедуру? Где угодно. Но так, чтобы процедура (подпрограмма) не выполнялась, если к ней не обращаются. Поэтому можно рекомендовать 3 варианта:

- 1. В одном сегменте после основной программы;
- 2. В одном сегменте перед точкой входа;
- 3. В отдельном сегменте.

Тогда можно в конце основной программы поставить команду FINISH.

A SEGMENT BEG: <och. прогр.=""> FINISH</och.>	B SEGMENT	C SEGMENT
	Процедура	Процедура
Процедура	BEG:	C ENDS
	<осн. прогр.>	D SEGMENT
		BEG:<осн. прогр.>
A ENDS	B ENDS	D ENDS
END BEG	END BEG	END BEG

Если процедура предназначена для использования и в других сегментах, то она имеет атрибут дистанции **FAR**. Главная процедура также должна иметь атрибут **FAR**. По умолчанию процедура имеет атрибут **NEAR**.

При выполнении команды **CALL** *NAME* сначала нужно занести адрес возвращения стек, а затем сделать переход на имя *NAME*.

Для **NEAR**-процедуры в стек достаточно занести лишь адрес смещения.

Следовательно, в этом случае команда CALL NAME эквивалентна

PUSH IP JMP NAME

Для **FAR**-процедуры адрес возвращения является абсолютным адресом и состоит из двух слов **CS:IP**. Поэтому команда **CALL** *NAME* будет эквивалентна *трем командам*:

PUSH CS PUSH IP JMP NAME

Другими словами:

- 1. Сначала в стек заносится адрес начала сегмента кода, который содержится в регистре СЅ
- 2. Затем заносится содержание регистра указателя команд ІР смещение.

Команда **RET** возвращает управление к точке вызова процедуры. Эквивалентна двум командам:

POP IP POP CS

Если процедура имеет атрибут **NEAR** - выполняется только первая команда. То есть, особенности выполнения команды **RET** определяются заглавием процедуры - наличием аттрибута **NEAR** или **FAR**.

Пример: Пусть, первый фрагмент программы транслирован по таким адресам:

0012 E8 0007	CALL MY_PROC; вызов процедуры	
0015 8B C3	MOV AX, ВХ; вернуться сюда из процедуры	
0017 B8 4C00	MOV AX,4c00h	
001A CD 21	INT 21h	
001C	MAIN ENDP	
	; описание процедуры	
001C	MY PROC PROC; начало процедуры	
001C B1 0A	MOV CL, 10; первая команда процедуры	
001E C3	RET; вернуться к точке вызова	
001F	MY PROC ENDP	
	_	

По умолчанию MY_PROC имеет атрибут NEAR.

Влияние процедуры на стек:

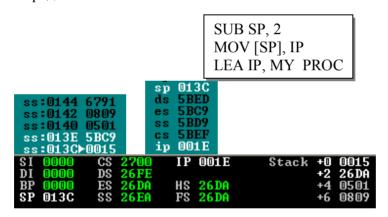
1. Перед выполнением CALL:

```
5BED
5BC9
ss:0146
ss:0144
ss:0142
                                 es
                                      5BD9
5BEF
                                 SS
              0809
                                  ip 0012
ss:013E > 5BC9
                                 ΙP
                                     0012
                                                     Stack
               DS
ES
                    26FE
26DA
                                HS 26DA
FS 26DA
   013E
               รีรี
```

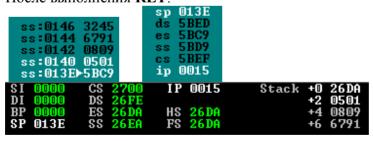
После выполнения CALL:



Перед выполнением **RET**:



После выполнения **RET**:



MOV IP [SP] ADD SP, 2

Продлится выполнение, начиная с команды NEXT:

Поскольку значение IP изменилось, то начнется выполнение процедуры, начиная с команды $\underline{MOV\ CL}$, $\underline{10}$ и до команды \underline{RET} .

Процедура может размещаться в том же сегменте, откуда осуществляются к ней обращения, или в другом сегменте. В первом случае атрибутом процедуры может быть **NEAR** и обращение также типа **NEAR**. Во втором случае атрибутом процедуры будет FAR и обращение также будет **FAR**. Но если процедура все-таки имеет атрибут FAR, то обращение к ней также должно иметь атрибут **FAR**.

Например:

SEGX SEGMENT
.......
SUBT PROC FAR
......
RET
SUBT ENDP
......
CALL FAR PTR SUBT
......
SEGX ENDS
SEGY SEGMENT
.....
CALL FAR PTR SUBT
......
SEGY ENDS

То есть, в обращении к процедуре могут быть атрибуты **FAR PTR** или **NEAR PTR**. В данном случае процедура определена перед ее вызовом, то есть, осуществляется вызов "назад". Поэтому если не отметить **FAR PTR**, то ассемблер все равно правильно реализует и вызовы и возвращения.

В данном случае их можно оставить для ясности. Но когда процедура реализуется после вызова, то есть, осуществляется вызов "вперед", то эти атрибуты обязательны.

Однако таких действий маловато. Для вызова процедуры из сегмента SEGY в нем обязательно нужно объяснить, что имя SUBT является внешним. То есть, включить директиву EXTRN SUBT: FAR.

В сегменте SEGX нужно объяснить, что к имени SUBT может быть доступ извне, то есть, нужно включить директиву Public SUBT.

Отметим, что имя SUBT находится в том же сегменте, что и директива Public. Поэтому в ней отмечать тип имени не нужно (транслятор разберется). В директиве EXTRN отметить тип нужно. По аналогии с командой **JMP** в процедуре возможны обращения:

- 1. прямые;
- 2. непрямые;
- 3. близкие;
- 4. далекие.

Не позволяются короткие **SHORT**, потому что считается, что процедура не располагается рядом с командой вызова. Поэтому, возможны такие варианты вызова:

CALL Р; близкий переход

CALL FAR PTR Q; далекий переход

CALL Q; близкий переход

CALL NEA; близкий непрямой вызов

CALL FA; далекий непрямой вызов

LEA BX,Q

CALL [BX]; близкий вызов процедуры Q

CALL DWORD PTR [BX]; далекий непрямой вызов

Пример:

NEAR DW P

FA DD Q

P PROC

P1:

P ENDP

Q PROC FAR

Q1: Q ENDP