**Процедуры**

* **Процедура** – участок кода, который может вызываться в различных местах программы
* После выполнения процедуры управление возвращается в точку вызова

**Инструкция CALL**

* Выполняет вызов процедур
* Помещает в стек значение **IP** и выполняет переход
* Пример:

**call** MyProc *; Вызывающая*

…

MyProc: *; Вызываемая*

…

**ret**

**Принцип работы (ПСЕВДОКОД):**

push ip

jmp MyProc

**Инструкция ret**

* Выполняет два действия:
  + Извлекает из стека значение
  + Помещает его в IP
* Есть необязательный операнд
  + Извлекает из стека значение
  + Извлекает из стека N байт (N – операнд)
    - По факту – увеличивает SP
  + Помещает в IP адрес возврата

Важно, чтобы обе стороны – вызывающая и вызываемая – соблюдали одно и то же соглашение вызова

* Способ передачи параметров
* Способ возврата значений
* Правила использования регистров
* И тд.

**Передача параметров:**

* Через регистры
* Через память (Глобальные переменные)
* Стек

**Возврат:**

* Через память
* Через стек
  + Такое себе, неудобно
* Через регистры
  + Топ

**Пролог и эпилог**

* **Пролог** – часть процедуры, которая подготавливает процедуру к работе с параметрами
* **Эпилог** – часть процедуры, которая восстанавливает состояние, необходимое для продолжения выполнения вызывающего кода

MyProc:

**push** **bp**

**mov** **bp**, **sp**

**sub** **sp**, sizeof**(**localvars**)**

…

**mov** **sp**, **bp**

**pop** **bp**

**ret** sizeof**(**params**)**

Для приведенного кода состояние стека будет таким:

…

[bp+6] – параметр 1

[bp+4] – параметр 2

[bp+2] – адрес возврата

[bp] – старое значение bp

[bp-2] – локальная переменная 1

[bp-4] – локальная переменная 1

…

**Соглашение вызова:**

* Pascal
* ccall
* std call
* comcall

**Соглашение вызова Pascal:**

* Параметры в стек помещаются слева направо
* Очистку стека выполняет вызываемая процедура

MyProc:

**push** **bp**

**mov** **bp**, **sp**

**mov** **ax**, **[bp**+4**]**

**sub** **ax**, **[bp**+6**]**

**pop** **bp**

**ret** 4

**push** Param1

**push** Param2

**call** MyProc

**Соглашение вызова C:**

* Параметры в стек помещаются справа налево
* Очистку стека выполняет вызывающая процедура

MyProc:

**push** **bp**

**mov** **bp**, **sp**

**mov** **ax**, **[bp**+6**]**

**sub** **ax**, **[bp**+4**]**

**pop** **bp**

**ret**

**push** Param2

**push** Param1

**call** MyProc

**add** **sp**, 4

**Соглашение вызова Std Call:**

* Параметры в стек помещаются справа налево
* Очистку стека выполняет вызываемая процедура

MyProc:

**push** **bp**

**mov** **bp**, **sp**

**mov** **ax**, **[bp**+6**]**

**sub** **ax**, **[bp**+4**]**

**pop** **bp**

**ret** 4

**push** Param2

**push** Param1

**call** MyProc

**Pascal, ccal, std call:**

* Процедура должна нетронутыми значения регистров:
  + (E)BX, (E)SI, (E)DI, (E)BP
* Может изменять значения:
  + (E)AX, (E)CX, (E)DX
  + (E)SP – изменяется в результате манипуляций со стеком

**Примеры:**

**Через регистры:**

Org 100h

EnteryPoint:

**mov** **ax**, **$**A3

**call** PrintHex

**call** pressAnyKey

**ret**

*; Input:*

*; None*

*; OutPut:*

*; None;*

*; Registers:*

*; Uses: None*

pressAnyKey:

**push** **ax**

**Mov** **ax**, **$**0c08

**Int** 21h

**Test** **al**, **al**

**jnz** @F

**Mov** **ah**, **$**08

**Int** 21h

@@:

**pop** **ax**

**ret**

*; Input:*

*; ax value*

*; Result:*

*; none*

*; Registers:*

*; Uses BX,CX,DX,AX*

PrintHex:

**mov** **bx**, **ax**

**mov** **cx**, 4 *; Ñ÷åò÷èê*

.PrintLoop:

**rol** **bx**, 4

**mov** **dx**, **bx**

**and** **dx**, **$**000F

**cmp** **dx**, 9

**ja** .Letter

**add** **dx**, '0'

**jmp** .EndIF

.Letter:

**add** **dx**, 'A' - 10

.EndIF:

**Mov** **ah**, **$**02

**Int** 21h

**loop** .PrintLoop

**ret**

Стек:

Org 100h

EnteryPoint:

push $A2

call PrintHex

call pressAnyKey

ret

; Input:

; ax value

; Result:

; none

; Registers:

; **Uses** BX,CX,DX,AX

PrintHex:

; Â ñòåêå õðàíèòñÿ åùå è àäðåñ âîçâðàòà (ââåðõó ñòåêà)

mov bp, sp

; [bp] - àäðåñ âîçâðàòà

; [bp + 2] - âîçâðàùàåìîå çíà÷åíèå

mov bx, [bp + 2]

mov cx, 4

.PrintLoop:

rol bx, 4

mov ax, bx

**and** ax, $000F

cmp al, $0A

sbb al, $69

das

mov dl, al

mov ah, $02

int 21h

loop .PrintLoop

ret 2

; Input:

; None

; OutPut:

; None;

; Registers:

; **Uses**: None

pressAnyKey:

push ax

mov ax, $0c08

Int 21h

Test al, al

jnz @F

Mov ah, $08

Int 21h

@@:

pop ax

ret

**Процедуры с дальним вызовом**

**Дальний вызов** (far call) – вызов, при котором вызываемая и вызывающая процедуры могут находится в разных сегментах

* Другое название – intersegment call
* Для выполнения дальнего использования используется инструкция **CALL**

*call far SegName:Proc*

* Выполняет 2 действия:
  + Помещает в стек значение регистров **CS:IP;**
  + Выполняет безусловный переход по адресу, заданному операндом.

Дальний возврат

* RetIP
* Выполняет 2 действия:

Стек:

…

RetIP

RetCS

Param2

Param1

* + Извлекает из стека 2 значения
  + Помещает их регистры **CS:IP**
* Как и у RET, есть необязательный операнд.

Смысл – тот же.