**PMAS Arid Agriculture University Rawalpindi**

**University Institute of Information Technology**

**LAB MANUAL - XI**

**Class/Program: BS (CS)** **Course: COAL (CS530)**



**Objectives:**

**1. Introduction to procedures**

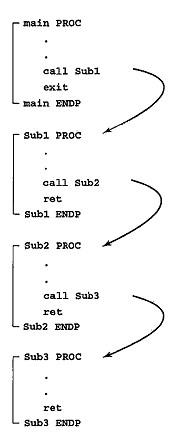
**2. Introduction to PROC statement**

**3. Procedure Definition**

**4. Procedure Calling**

**5. Introduction to Global Code label**

**Introduction to Procedure:**

Procedures are the logical breakdown block of a single bigger program. These help to implement complex problems more efficiently by solving smaller instances of that big problem. Assembly language also provide the facility to create user defined functions. That can be used and called in the main procedure for execution.

The general method for creating a procedure in assembly language is as follows.

**Procedure name (space) PROC**

**; General statements in the procedure**

**RET**

**Procedure name (space) ENDP**

Procedure name must be any meaningful words. The mnemonic PROC is used to create a procedure as PROC stands for procedure. RET instructions is a return instruction that simply returns the program control from where it is branched out for the procedure execution. The ENDP instruction ends the procedure.

To call the procedure simply write the procedure name in front of CALL command in the MAIN PROC.

**MAIN PROC**

**; General instructions before procedure**

**CALL (space) procedure name**

**; General instructions after procedure**

**EXIT**

**MAIN ENDP**

**END MAIN**

Assembly language also provides the facility to skip the MAIN PROC if not required and start the program with any other user defined procedure.  
  
The syntax for procedure declaration:

**name PROC  
  
      ; here goes the code  
      ; of the procedure ...  
  
RET  
name ENDP**

name - is the procedure name, the same name should be in the top and the bottom, this is used to check correct closing of procedures.  
  
Probably, you already know that **RET** instruction is used to return to operating system. The same instruction is used to return from procedure (actually operating system sees your program as a special procedure).  
  
**PROC** and **ENDP** are compiler directives, so they are not assembled into any real machine code. Compiler just remembers the address of procedure.  
  
**CALL** instruction is used to call a procedure.  
  
Here is an example:

|  |
| --- |
| ORG 100h  CALL m1  MOV AX, 2  RET ; return to operating system.  m1 PROC  MOV BX, 5  RET ; return to caller.  m1 ENDP  END |

The above example calls procedure **m1**, does **MOV BX, 5**, and returns to the next instruction after **CALL**: **MOV AX, 2**.  
  
There are several ways to pass parameters to procedure, the easiest way to pass parameters is by using registers, here is another example of a procedure that receives two parameters in **AL** and **BL** registers, multiplies these parameters and returns the result in **AX** register:

|  |
| --- |
| ORG 100h  MOV AL, 1  MOV BL, 2  CALL m2  CALL m2  CALL m2  CALL m2  RET ; return to operating system.  m2 PROC  MUL BL ; AX = AL \* BL.  RET ; return to caller.  m2 ENDP  END |

In the above example value of **AL** register is update every time the procedure is called, **BL** register stays unchanged, so this algorithm calculates **2** in power of **4**,  
so final result in **AX** register is **16** (or 10h).

Here goes another example,  
that uses a procedure to print a *Hello World!* message:

|  |
| --- |
| ORG 100h  LEA SI, msg ; load address of msg to SI.  CALL print\_me  RET ; return to operating system.  ; ==========================================================  ; this procedure prints a string, the string should be null  ; terminated (have zero in the end),  ; the string address should be in SI register:  print\_me PROC  next\_char:  CMP b.[SI], 0 ; check for zero to stop  JE stop ;  MOV AL, [SI] ; next get ASCII char.  MOV AH, 0Eh ; teletype function number.  INT 10h ; using interrupt to print a char in AL.  ADD SI, 1 ; advance index of string array.  JMP next\_char ; go back, and type another char.  stop:  RET ; return to caller.  print\_me ENDP  ; ==========================================================  msg DB 'Hello World!', 0 ; null terminated string.  END |

"**b.**" - prefix before [SI] means that we need to compare bytes, not words. When you need to compare words add "**w.**" prefix instead. When one of the compared operands is a register it's not required because compiler knows the size of each register.