MPMC Assignment:

Name: J. Sadhana
ECE-C
15B81A04D7

① Passing parameters to procedures:

Procedures or subroutines may require input data or constants for their execution. Their data or constants may be passed to the subroutine by the main program or some subroutine may access readily available data of constants available in memory

Generally, the following techniques are used to pass input data to procedures in assembly language programs.

(i) Using global declared variable
(ii) Using registers of CPU architecture.
(iii) Using memory locations (reserved)
(iv) Using stack
(v) Using PUBLIC & EXTRN

Besides these methods if a procedure is interactive it may directly accepts inputs from input devices.

① NUMBER EQU 77H GLOBAL

Ex: MOV AX, DATA
MOV DS, AX
MOV BX, NUMBER (i.e. NUMBER = 77H Global)

②

→ The CPU general purpose registers may be used to pass parameters to the procedures. The main program may store

the parameters to be passed to the procedure in the available CPU registers and the procedure may use the same register contents for execution. The original contents of the used CPU register may change during execution of the procedure.

Ex:② – Assume CS: code

```
code segment
start: MOV AX, 5555H
       MOV BX, 7272H
        !
       CALL Procedure1

Procedure Procedure1 NEAR
        !
       ADD AX, BX
        !
       RET

Procedure1 ENDP
code ends
end start
```

→ Memory locations may also be used to pass parameters in the same way as registers. A main program may store the parameter to be passed to a procedure at a known memory address location and the procedure may use the same location for accessing the parameter.

Ex: ③ Assume CS: code, DS: data

```
Data segment
NUM dB (55H)
Count EQU 10H
Data ends
Code segment
start: MOV AX, DATA
       MOV DS, AX
        :
        !
       CALL ROUTINE
        :
        !
PROCEDURE ROUTINE NEAR
       MOV BX, NUM
       MOV CX, count
        :
ROUTINE ENDP
 code ends
 end start .
```

→ Stack memory can also be used to pass parameters to a procedure. A main program may store the parameters to be passed to a procedure in the CPU registers. The registers will further be pushed onto the stack. The procedure during its execution pops back the appropriate parameters as and when required. This procedure of poping back the

parameters to be passed to the procedure ~~and~~ the stack contains other important information.

(4) Assume CS: code, SS: stack.

```
code segment
start: MOV AX, stack
       MOV SS, AX
       MOV AX, 5577H
       MOV BX, 2929H
         :
       PUSH AX
       PUSH BX
       CALL Routine : Decrements SP by 2 (by 4 for routine)
         :
PROCEDURE ROUTINE NEAR
         :
       MOV DX, SP
       ADD SP, 02 ; Leave initial two stack bytes of
                    return offset and segment address
                    after executing subroutine ;
       POP BX    ; The data passes into
       POP AX    ;  BX, AX
       MOV SP, DX
         :
stack segment
stackdata dB 2004 DUP(?)
stack ends.
```

→ For passing parameters to procedures using the PUBLIC & EXTRN directives, must be declared PUBLIC (for all routines) in the main routine and the same should be declared EXTRN in the procedure. Thus the main program can pass the PUBLIC parameter to a procedure in which it is declared EXTRN (external)

Ex ① Assume CS:code, DS:data

```
Data segment
PUBLIC number EQU 200H
Data ends
code segment
start : MOV AX, DATA
        MOV DS, AX
          !
        CALL ROUTINE
          !
        PROCEDURE ROUTINE NEAR
EXTRN number
        MOV AX, number
          !
ROUTINE ENDP
```

② Passing parameters to a MACRO:

Using parameters in a definition, the programmer specifies the parameters of the macro those are likely to be

changed each time the macro is called. For example, the DISPLAY macro written can be made to display two different messages MSG1 and MSG2 as shown

```
DISPLAY MACRO MSG
        MOV AX, SEG MSG
        MOV DS, AX
        MOV DX, offset MSG
        MOV AH, 09h
        Int 21h
    ENDM
```

This parameter MSG can be replaced by MSG1 or MSG2 while calling the macro as shown.

```
        :
DISPLAY MSG1
        :
DISPLAY MSG2
        :
```

```
MSG1 dB 0AH, 0DH, "Program terminated normally", 0AH, 0DH, "$"
MSG2 dB 0AH, 0DH, "Retry, Abort, Fail", 0AH, 0DH, "$"
```

There may be more than one parameter appearing in the macro definition, meaning thereby that there may be more than one parameters to be passed to the macro, and each of them is liable to be changed. All the parameters are specified in the definition sequentially and also in the call with the same sequence.