(Microprocessor - 22415)

Unit V : Procedure and Macros

(Weightage - 12 marks)

Pranjal Sare
(SY-CO)

5.1 : Defining and calling procedure - PROC, ENP, FAR and NEAR Directives; CALL and RET instructions; Parameter passing methods.

Questions.....

1. Define procedure and write its syntax. (2M)
2. Explain Re-Entrant and Recursive procedure with diagram. (4M)
3. Describe CALL and RET instruction with example. (4M)
4. Compare procedure and macros (4M/2M)
5. Differentiate between NEAR and FAR calls. (2M/4M)
   or
   Compare intersegment & interasegment.
6. List directive used for procedure. (2M)
7. Describe with suitable example how parameter is passed on the stack in 8086 Assembly language procedure. (4M)

Procedure

A procedure is a group of instructions that usually perform one task.

It is usuable section of software program which is stored in memory once but can be used as often as necessary.

A procedure can be of two types

1) Near Procedure
2) Far Procedure.

Large problems can be divided into smaller task to make them manageable.

Syntax

Procedure-name PROC {Near/Far}

- - -

Procedure-name ENDP

Directives used for procedure

i) PROC: The PROC directive is used to identify the start of Procedure.
The PROC directive follows a name given to Procedure. After that term FAR and Near is used to specify the type of procedure.

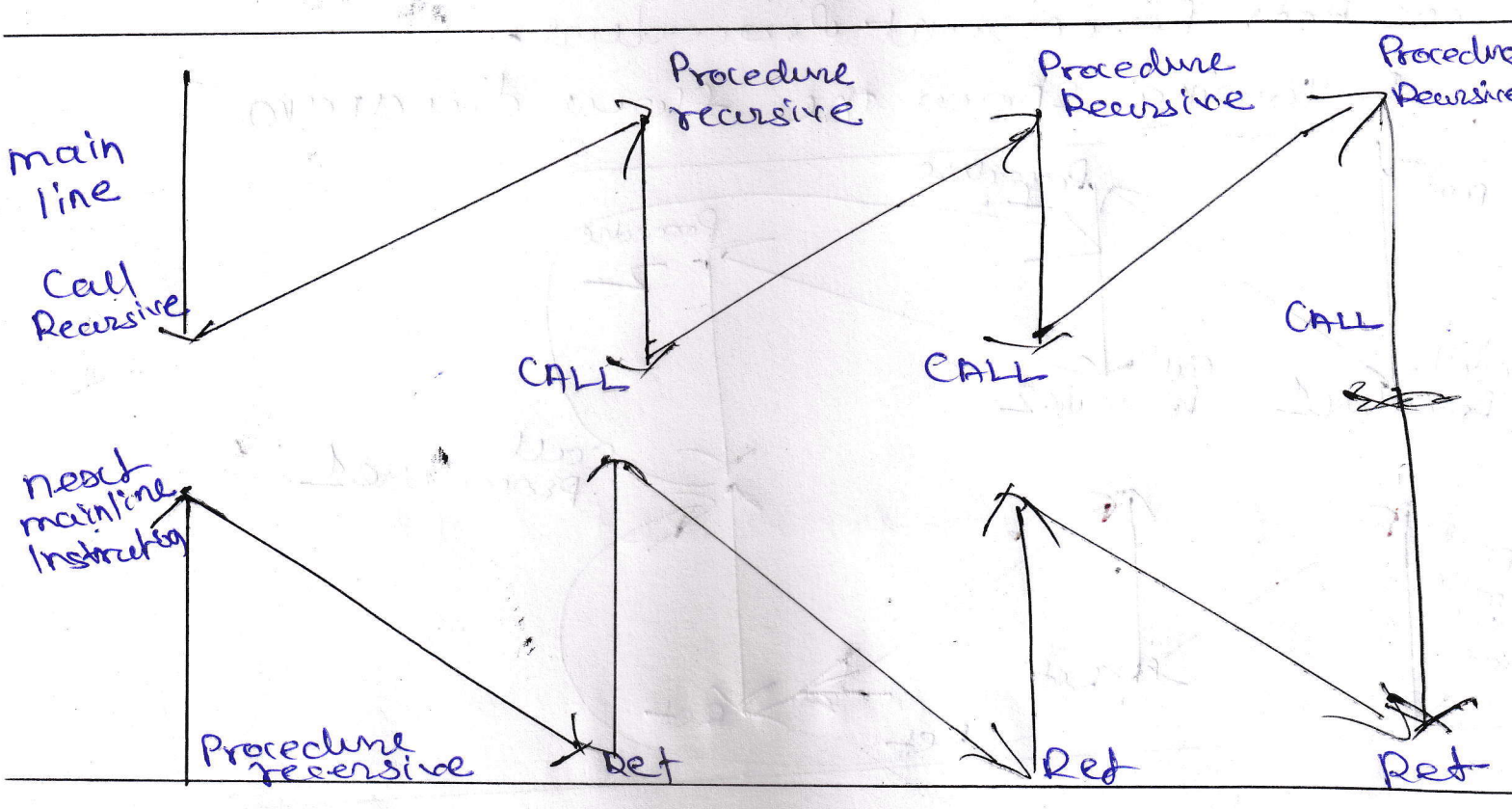
ii) ENDP: This directive is used along with the name of procedure to indicate the end of procedure to assembler.
The PROC and ENDP procedure directive are used.

Near Call/Procedure	Far Call/Procedure
It is also called Intra segment Call	It is also called Inter segment call
A near procedure refer to a procedure which is in same code segment from that of Call instruction.	A Far procedure refers to which is in different code segment from that of Call instruction.
It uses keyword ~~Near~~ Near for calling procedure.	It uses keyword Far for calling procedure.
Less stack locations are required.	More Stack locations are required.
A Near Call replaces the old IP with new IP	A Far Call replaces the CS & IP with new CS & IP
Example, Call Delay	Example Call Far PTR Delay

Procedures

1) Recursive Procedure

- A recursive procedure is procedure which calls itself.
- Recursive procedures are used to work with complex data structures such as trees.
  - If the procedures is called with N (recursion depth) = 3.
  - Then the n is decremented by one after each procedure CALL and the procedure is called until n=0.
  - Below Figure shows the flow diagram and pseudocode for recursive procedure.




```
If n=0
    Decrement N
      CALL Recursive
Else
    Ret.
```

2) Re-entrant Procedure

In some situation, it may happen that procedure 1 is called by main program and procedure 2 is called from procedure 1.

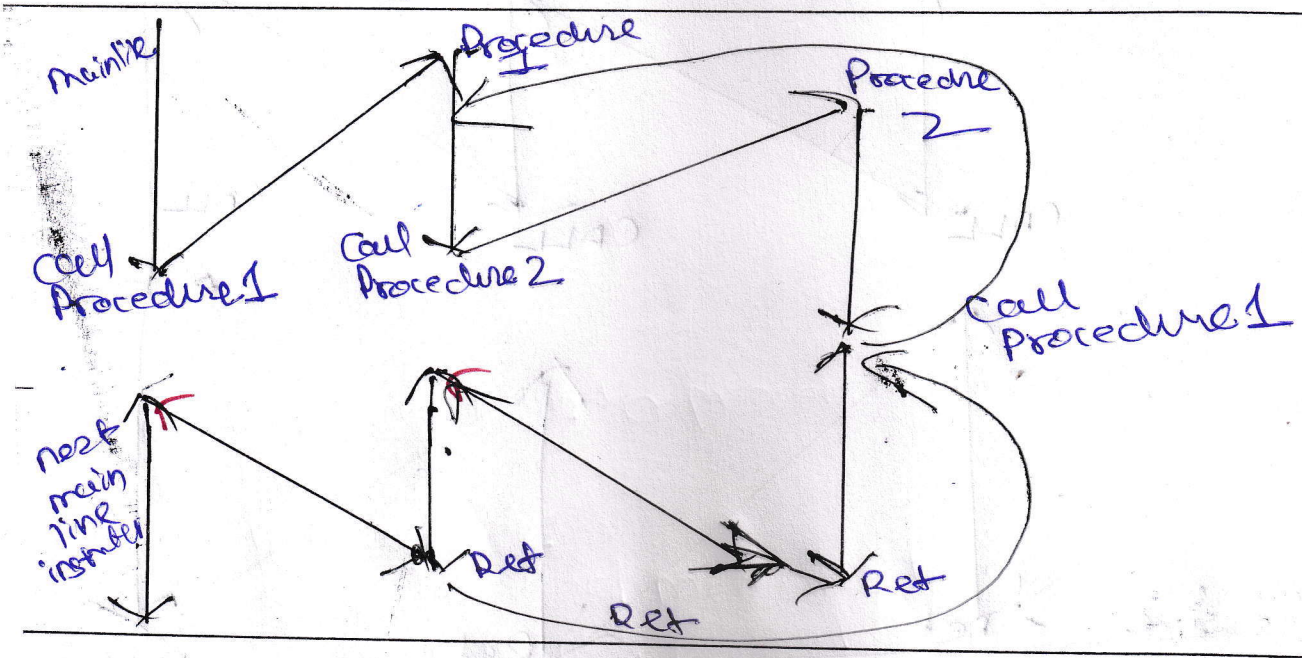
And again procedure 1 is called in procedure

In this situation, program execution re-enters in procedure.

First time when proc-1 was called by main program and second time when procedure 1 was called from procedure 2.

Hence this type of procedure is called as Re-entrant Procedure.

Following show the flow diagram



CALL & RET instruction

CALL instruction

The call instruction is used to transfer execution to a procedure.
It performs two operations - when it executes, first it stores the address of instruction after call instruction on the stack.
It changes the contains of IP register in case of Near call and changes contein of IP & CS register in case of FAR call.

There are two types of operation on calls

- Near Call
- Far Call

Near call

SP ← SP - 2
IP ← stores onto stack
IP ← starting address of procedure.

when 8086 execute a near call procedure, it decrement stack pointer by 2 and copies the IP register contents on the stack.
Then it copies address of first instruction of called procedure.

FAR call

SP ← SP-2
CS contents → stored on stack
SP ← SP-2
IP content → stored on stack
~~IP contents ← stored~~
CS ← Base address of segment have procedure
IP ← address of first instruction in procedure.

Syntax: CALL procedure_name

RET instruction

The RET instruction will return the execution from a procedure to next instruction after a call in main program.

At the end of every procedure RET instruction must be executed.

Operation for Near

IP ← Address from top stack

SP ← SP + 2

Operation for far

IP ← Address from top of Stack

SP ← SP + 2

CS IP ← Address from top of stack

SP ← SP + 2

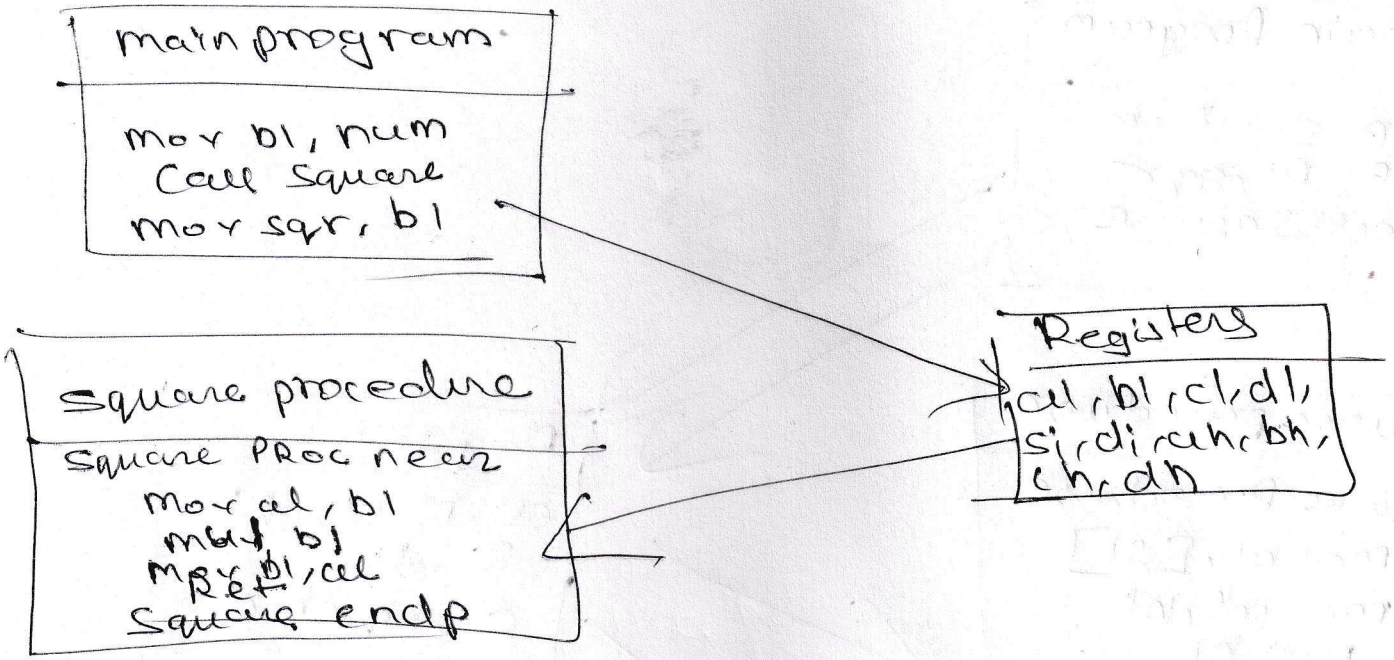
Passing parameter to procedure to and from procedure

1) In register
2) In dedicated memory locations accessed by name
3) With pointer passed in register.
4) With stack.

1) Passing parameter in registers

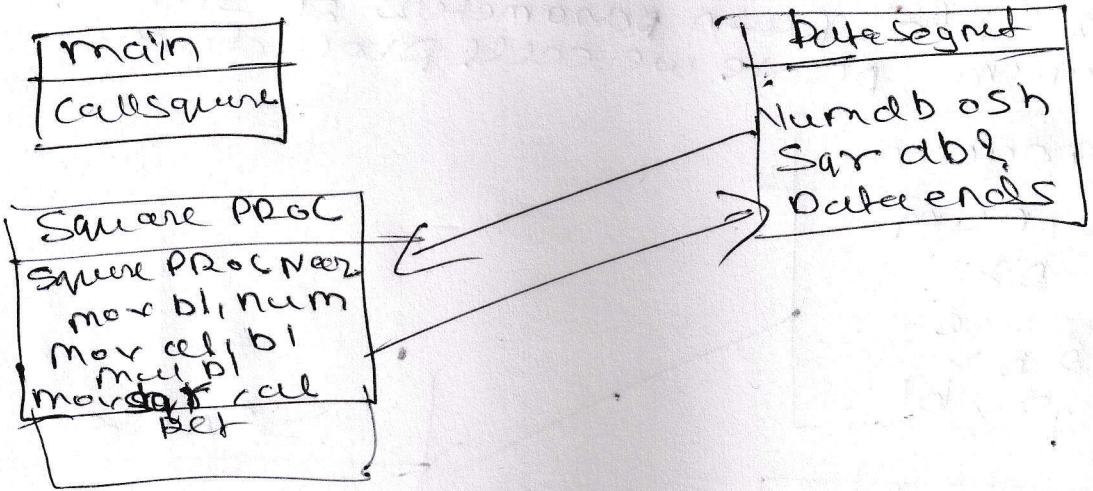
main program can pass upto 6 parameters to procedure through the register AX, BX, CX, DX, SI & DI before executing call instruction

eg. consider program to calculate square



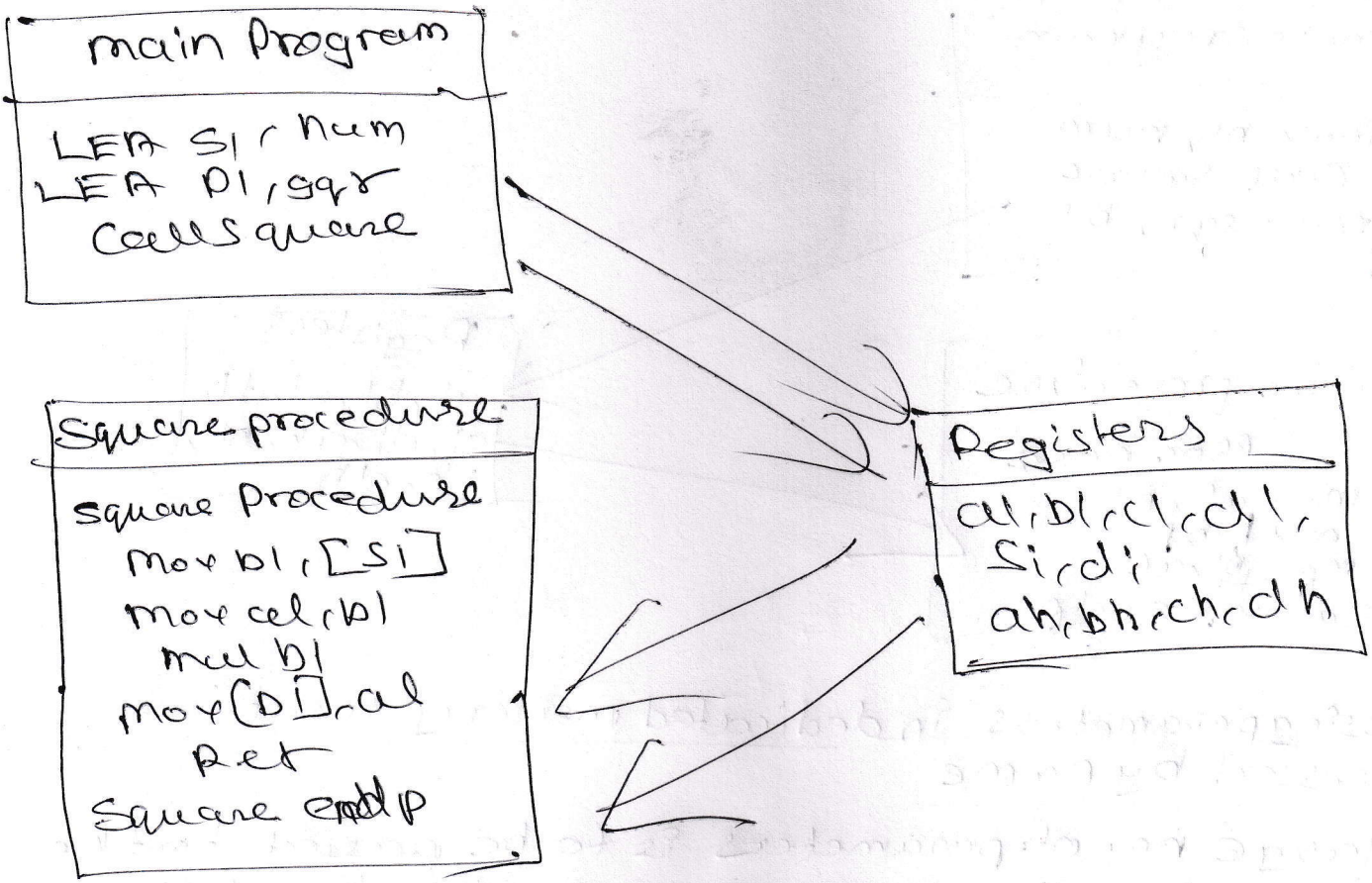
2) Passing parameters in dedicated memory location accessed by name

when large no. of parameters is to be passed to the procedure, then these parameters can be placed in an arrugement list as dedicated memory locations in one of data segment from memory.



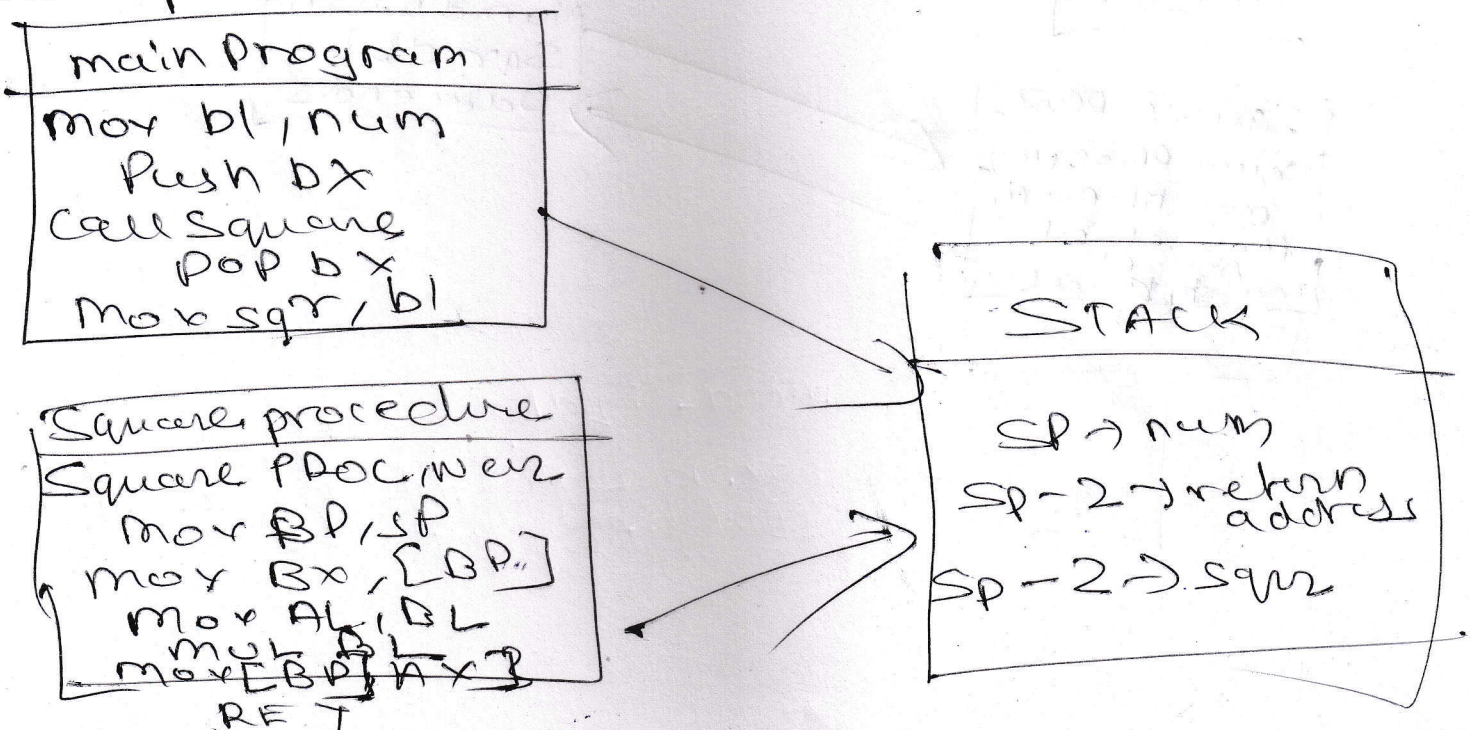
3) Passing with pointer

In main program, before we call procedure we can set up SI as a pointer to pass values and set DI as pointer to receive values from procedure.



4) Passing parameters with stack

Alternate method of passing large numbers of parameters is to push parameters on stack in main program before we call procedure.



Procedure	Macros
Procedure are used for large group of instructions to be repeated.	Macros are used for small group of instructions to be repeated.
object code is generated only once in memory	As much time you call macro object code generates.
CALL & RET instructions are used to call procedure and return from procedure.	Macro can be called just by writing its name
Length of object file is less	length of object file is more
Directives used: ENDP, PROC	Directive used: MACRO ENDM
more time required for execution	Less time required for execution.
Stack is required to call procedure.	Stack is not required to call MACRO
RET ~~not~~ required	RET is ~~mandatory~~ not required

Syntax	Example	Syntax	Example
Procedure_name PROC (Near/far)	Addition PROC near	MACRO-name MACRO [Arguements]	Display macro msg
— — —	— — —	— — —	— — —
Procedure_name ENDP	Addition ENDP	ENDM	ENDM

5.2 : Defining macro, MACRO and ENDM Directives, macro with parameters.

MACRO

Questions . . .

Q) Define macro with its syntax. (2m)

Q) Explain macro. Any four advantages of using macro. (4m)

MACRO

A macro is small instruction that usually performs one task.

It is reusable section of software program.

A macro can be defined anywhere in program using directive MACRO & Endm.

General Form

```
macro-name  MACRO(Arguement 1 - - -
                  Arguement n)
        _ _ _ _
  //macro code
Endm
```

Example

```
DISPLAY MACRO  12,13
   _ _ _ _
MACRO statements
   _ _ _ _
Endm.
```

(value passed to macro arugements)

Advantages

- Program written with macro is more readable
- Just can be called by its name along with parameters.
- Execution time is less
- Finding errors during debugging is easier

Disadvantages

- object code is lengthy.
- not prefered for large group of instructions.

Procedures & Programming

Program 1: Write an ALP for addition of series of 8-bit number using procedure (4m)

```
Data segment
 Num1 DB 10h, 20h, 30h, 40h, 50h
 Result DB 0h
  Carry DB 0h.
Data ends.

Code Segment
 Assume CS:Code, DS:Data
Start: mov DX, Data
       mov DS, DX
       mov CL, 05h
       mov SI, offset Num1

  UP:  CALL SUM
       INC SI
       LOOP UP
       MOV AH, 4CH
       INT 21H

SUM PROC          ;PROCEDURE
MOV AL, [SI]
Add Result, AL
JNC NEXT
INC Carry
NEXT: RET
 SUM ENDP
 Code Ends
End start
```

program2

write an ALP using procedure to solve equation such as Z=(A+B)*(C+D) .... (4M)

```
Sum PROC Near
 Add AL, BL
 RET
Sum ENDP

Data Segment
 Num1  DB 10H
 Num2  DB 20H
 Num3  DB 30h
 Num4  DB 40h
 Result DB ?
Data Ends

code Segment
Assume CS:Code, DS:Data
Start: Mov AX, Data
       Mov DS, AX
       Mov AL, Num1
       Mov BL, Num2
       CALL SUM
       Mov CL, AL
       Mov AL, Num3
       Mov BL, Num4
       CALL SUM
       MUL CL
       Mov RESULT, AX
 Mov AH, 4CH
  INT 21H
Code Ends
End start.
```

Write ALP to subtract two BCD number using procedure ----(4M)

```
Data Segment
    Num1  DB 25h
    Num2  DB 12h
    Result DB ?
Data Ends

Code Segment
Assume CS:Code, DS:Data
Start: mov AX, Data
       mov DS, AX

    mov SI, OFFSET Num1
    mov DI, OFFSET Num2
CALL Subtract_BCD
 mov AH, 02h
 MOV DL, ' '
 INT 21H
 mov DL, RESULT
 Add DL, 30H
 INT 21H

mov AH, 4CH
INT 21H
Subtract_BCD PROC
  mov AL, [SI]
  mov BL, [DI]
  SUB AL, BL
  DAA
  mov [RESULT], AL
  RET
Subtract_BCD endp
Code Ends
End Start.
```

Q3 Write the procedure to find factorial of given number.

```
Factorial PROC
 PUSH CX
 PUSH BX

 Mov CX, [SI]
 Mov BX, 1
 Mov AX, 1

Factorial_LOOP:
      MUL BX
      INC BX
    LOOP FACTORIAL_LOOP

 Mov FACT_RESULT, AX

  POP BX
  POP CX
  RET
FACTORIAL ENDP
```

MACRO - Programs

Write an ALP using macro to perform multiplication of two 8 bit unsigned numbers. (4M)

```
PRoduct MACRO First, second
  Mov AL, First
  Mov BL, Second
  MUL BL
  Product Endm

Data Segment
No1 DB 05h
No2 DB 04h
multiply DW ?
Data Ends

Code Segment
Assume CS: Code, DS: Data
Start: Mov AX, Data
       Mov DS, AX
       Product No1, NO2
       Mov MULTIPLE, AX

Mov AH, 4Ch
INT 21h

Code Ends
End Start
```

Write a assembly language program to solve $P = x^2 + y^2$ using macro (x & y are 8-bit nos) (4M)

```
PROG MACRO a,b
  mov al,a
  mov bl,b
  MUL Al
  mov bl,al
  mov al,b
  mul al
  Add al,bl
  EndM

Data Segment
  x  DB  03H
  y  DB  04H
  p  DB  DUP()
Data Ends
Assume: CS:Code, DS:Data
Code Segment
Start: mov AX,data
       mov DS,AX
       PROG x,y
  mov p,al
  mov AH,4CH
  Int 21H
End Start.
```

Write ALP using MACRO to perform following operation (4M).

$X = (A+B) * (C+D)$

```
PROG  MACRO  no1, no2
      mov    al, no1
      mov    bl, no2
      add    al, bl
      ENDM

Data Segment
      a   db   01h
      b   db   02h
      c   db   10h
      d   db   20h
      x   dw   ?
Data Ends.

Code Segment
ASSUME  CS:code, DS:Data
Start:  mov AX, Data
        mov DS, AX
        PROG MACRO a, b
        mov CL, AL
        MOV AL, C
        mov BL, d
        PROG MACRO c, d
        MUL CL
        mov result, ax
        INT 21h
Code Ends
End start.
```