Procedure and Macro (16 marks)

Define procedure : A procedure is group of instructions that usually performs one task. It is a reusable section of a 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

Near Procedure: A procedure is known as NEAR procedure if is written(defined) in the same code segment which is calling that procedure. Only Instruction Pointer(IP register) contents will be changed in NEAR procedure.

FAR procedure : A procedure is known as FAR procedure if it is written (defined) in the different code segment than the calling segment. In this case both Instruction Pointer(IP) and the Code Segment(CS) register content will be changed.

Directives used for procedure :

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

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

CALL instruction and RET instruction :

CALL instruction : The CALL instruction is used to transfer execution to a procedure.It performs two operation.When it executes,first it stores the address of instruction after the CALL instruction on the stack.Second it changes the content of IP register in case of Near call and changes the content of IP register and cs register in case of FAR call.

There are two types of calls.
1)Near Call or Intra segment call.
2) Far call or Inter Segment call

Operation for Near Call : When 8086 executes a near CALL instruction, it decrements the stack pointer by 2 and copies the IP register contents on to the stack.Then it copies address of first instruction of called procedure.

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

Operation of FAR CALL: When 8086 executes a far call, it decrements the stack pointer by 2 and copies the contents of CS register to the stack. It the decrements the stack pointer by 2 again and copies the content of IP register to the stack.Finally it loads cs register with base address of segment having procedure and IP with address of first instruction in procedure.

SP ← sp-2
cs contents →stored on stack
SP ← sp-2
IP contents → stored on stack
CS ← Base address of segment having procedure
IP ← address of first instruction in procedure.

RET instruction : The RET instruction will return execution from a procedure to the next instruction after call in the main program. At the end of every procedure RET instruction must be executed.

Operation for Near Procedure : For NEAR procedure ,the return is done by replacing the IP register with a address popped off from stack and then SP will be incremented by 2.

IP ← Address from top of stack
SP ← SP+2

Operation for FAR procedure : IP register is replaced by address popped off from top of stack,then SP will be incremented by 2.The CS register is replaced with a address popped off from top of stack.Again SP will be incremented by 2.

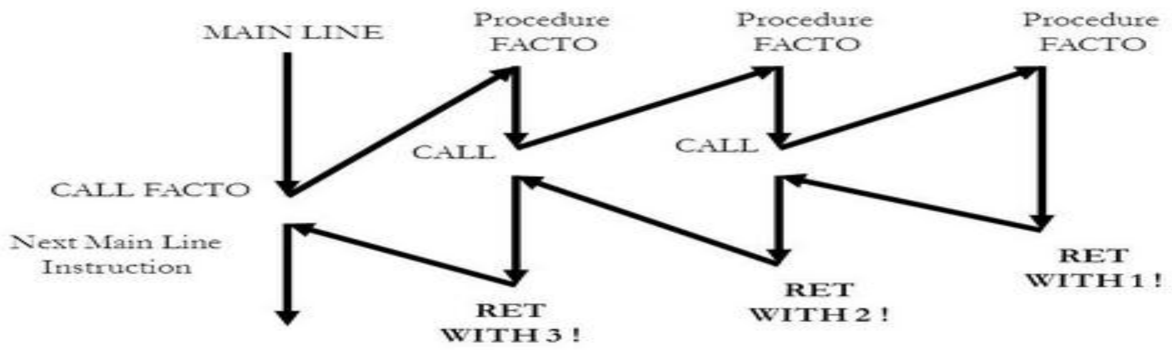
IP ← Address from top of stack
SP ← SP+2
CS ← Address from top of stack
SP ← SP+2

Difference between FAR CALL and NEAR CALL.

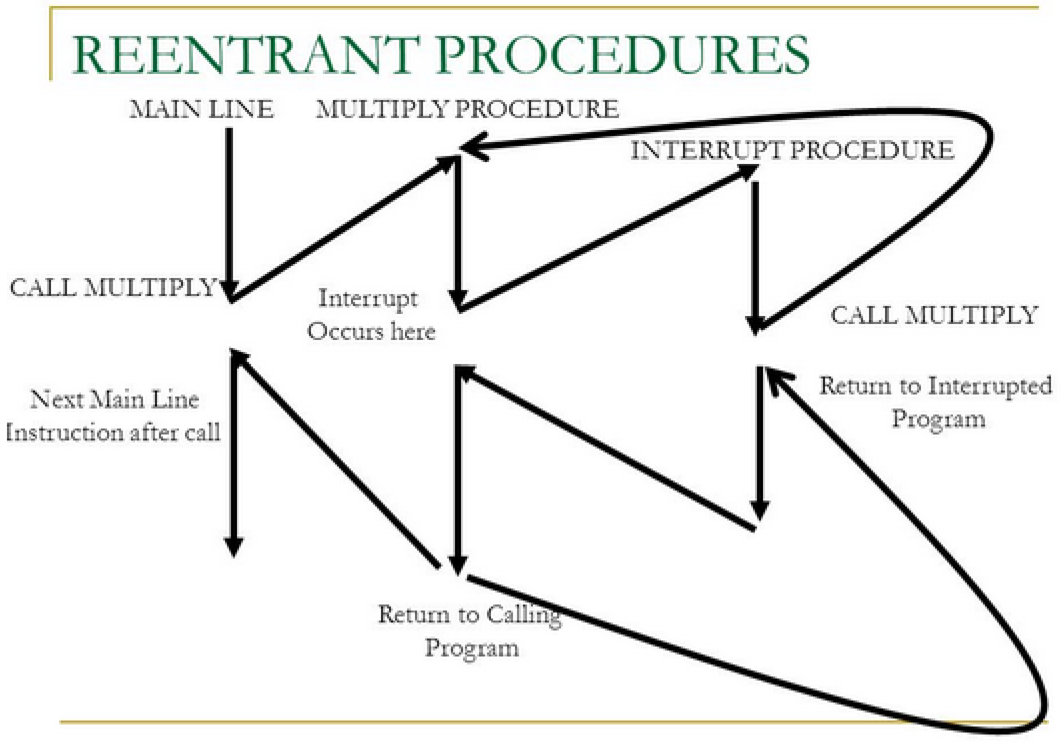
Near Call	Far Call
A near call refers a procedure which is in the same code segment.	A Far call refers a procedure which is in different code segment
It is also called Intra-segment call.	It is also called Inter-segment call
A Near Call replaces the old IP with new IP	A FAR replaces CS & IP with new CS & IP.
It uses keyword near for calling procedure.	It uses keyword far for calling procedure.
Less stack locations are required	More stack locations are required.

Explain recursive and Re-entrant procedure with schematic diagram

Recursive Procedure : It is a procedure which call itself. Recursive procedures are used to work with complex data structure like trees. If procedure is called with N (recursive depth) then N is decremented by one after each procedure CALL and the procedure is called until n=0.Recursive procedure takes less time to implement a particular task.But it needs certain condition for it's termination.



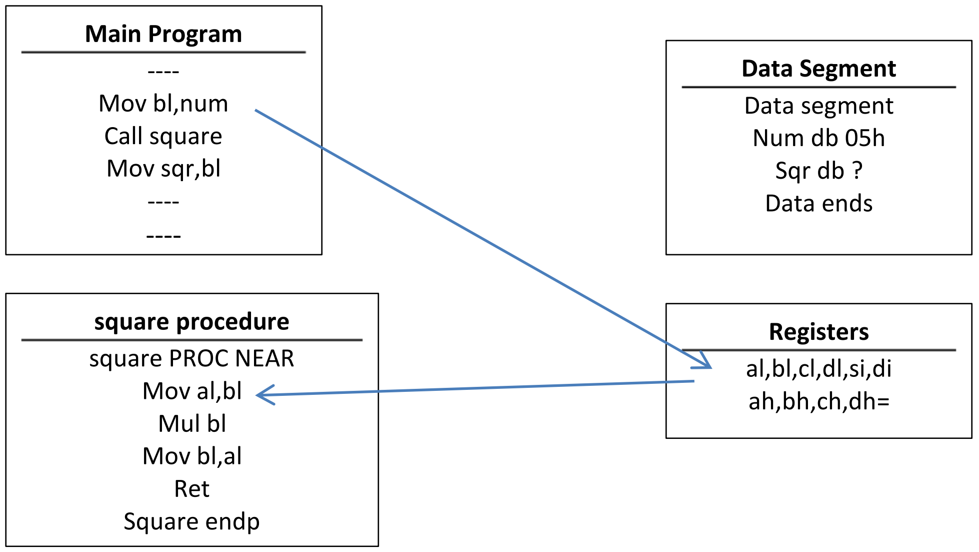
Re-entrant procedure : In some situation, it may happen that procedure 1 is called from main program and procedure 2 is called from procedure 1.And again procedure 1 is called from procedure 2. In this situation , program execution flow re-enters in the procedure 1 (first time when procedure 1 was called from main program and second time when procedure 1 was called from procedure 2) .Hence this type of procedure is called as Reentrant procedure.



Passing Parameter to procedure and from procedure : There are four major ways of passing parameters to and from a procedure.

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

1) Passing parameters in registers : The main program can pass upto 6 parameters to the procedure through the registers AX,BX,CX,DX,SI & DI before executing the call instruction.
e.g. consider the program to calculate a square of given number.



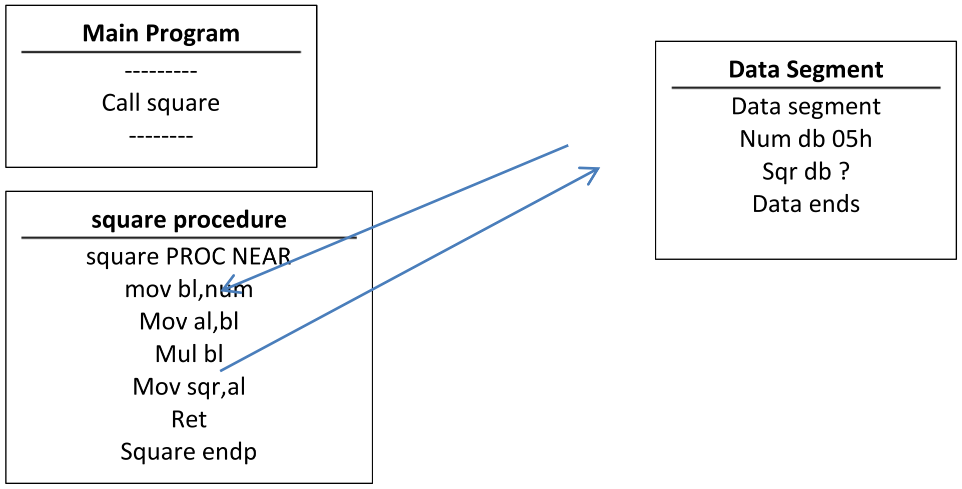
Main Program
```
----
Mov bl,num
Call square
Mov sqr,bl
----
----
```

Data Segment
```
Data segment
Num db 05h
Sqr db ?
Data ends
```

square procedure
```
square PROC NEAR
Mov al,bl
Mul bl
Mov bl,al
Ret
Square endp
```

Registers
```
al,bl,cl,dl,si,di
ah,bh,ch,dh=
```

2) **Passing parameters in dedicated memory locations accessed by name :** when large number of parameters is to be passed to the procedure, then these parameters can be placed in an argument list as dedicated memory locations in one of the data segment from memory.
e.g. consider the program to calculate a square of given number.

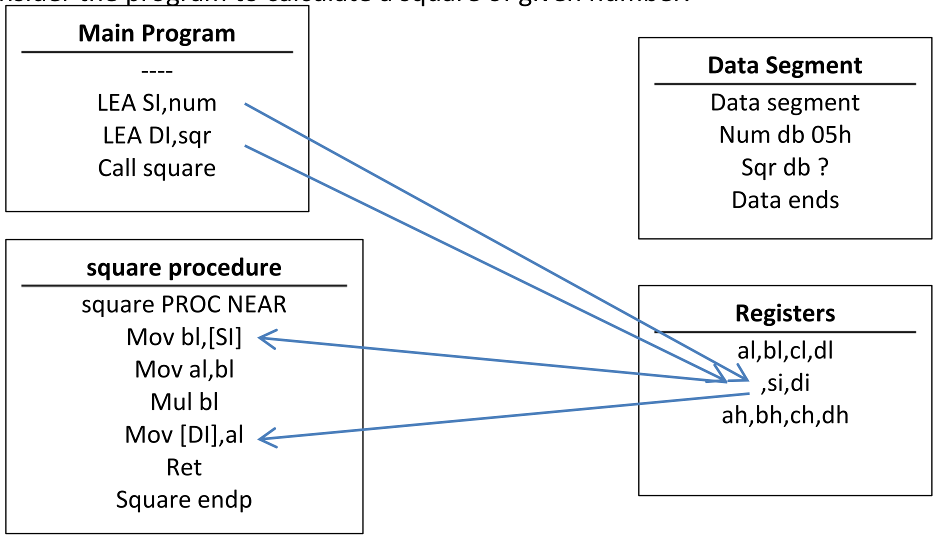


Main Program
```
---------
Call square
--------
```

Data Segment
```
Data segment
Num db 05h
Sqr db ?
Data ends
```

square procedure
```
square PROC NEAR
mov bl,num
Mov al,bl
Mul bl
Mov sqr,al
Ret
Square endp
```

3) **Passing parameters with Pointer :** In the main program, before we call the procedure we can set up SI as a pointer to pass values to procedures and set DI as pointer to receive values from procedures.
e.g. consider the program to calculate a square of given number.



Main Program

```
----
LEA SI,num
LEA DI,sqr
Call square
```

Data Segment

```
Data segment
Num db 05h
Sqr db ?
Data ends
```

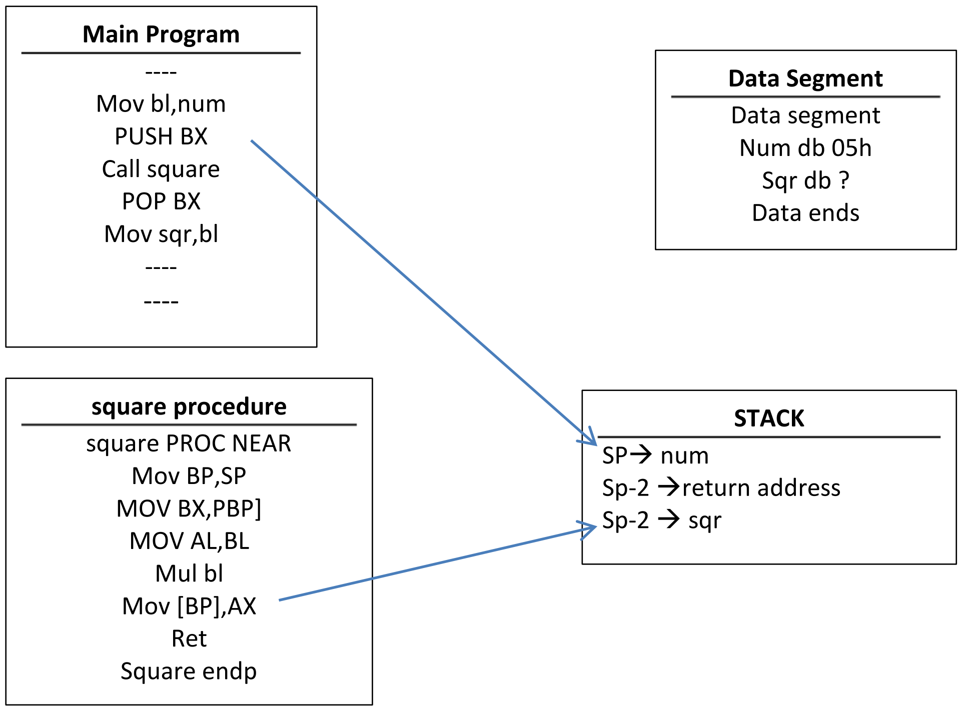
square procedure

```
square PROC NEAR
Mov bl,[SI]
Mov al,bl
Mul bl
Mov [DI],al
Ret
Square endp
```

Registers

```
al,bl,cl,dl
,si,di
ah,bh,ch,dh
```

4**) Passing parameters with stack :** Alternate method of passing large number of parameters is to push the parameters on the stack in the main program before we call the procedure.
e.g. consider the program to calculate a square of given number.



Main Program

```
----
Mov bl,num
PUSH BX
Call square
POP BX
Mov sqr,bl
----
----
```

Data Segment

```
Data segment
Num db 05h
Sqr db ?
Data ends
```

square procedure

```
square PROC NEAR
Mov BP,SP
MOV BX,PBP]
MOV AL,BL
Mul bl
Mov [BP],AX
Ret
Square endp
```

STACK

```
SP→ num
Sp-2 →return address
Sp-2 → sqr
```

Advantages and Disadvantages of using procedure : Advantages :

1) Allows to save memory space.
2) Program development becomes easier.
3) Debugging of errors in program become easy.
4) Reduced size of program
5) Reusability of procedure.

Disadvantages :

1) CALL and RET instructions are always required to integrate with procedures.
2) Requires the extra time to link procedure and return from it.
3) For small group of instructions, linking and returning back time more than the execution time, hence for small group of instructions procedures cannot be preffered.

Defining Macro : A MACRO is group of small instructions that usually performs one task. It is a reusable section of a software program.A macro can be defined anywhere in a program using directive MACRO &ENDM.

General Form :

```
MACRO-name  MACRO [ARGUMENT 1,……….ARGUMENT N]
                     -----
         MACRO CODIN GOES HERE
  ENDM
```

```
E.G     DISPLAY MACRO 12,13
         ---------------------
        MACRO STATEMENTS
        ----------------------
        ENDM
```

The Label prior to MACRO is the macro name which should be used in the actual program. The ENDM directive marks the end of the instructions.A macro can be called by quoting its name along with any values to be passed to the macro.12 & 13 are values to be passed with macro.

Advantages and disadvantages of MACRO :

Advantages:

1) Program written with macro is more readable.
2) Macro can be called just writing by its name along with parameters, hence no extra code is required like CALL & RET.

3) Execution time is less bcz of no linking and returning
4) Finding errors during debugging is easier.

Disadvantages :

1) object code is generated every time a macro is called hence object file becomes lengthy.
2) For large group of instructions macro cannot be preferred

Difference between PROCEDURE & MACRO.

Procedure	Macro
Procedures are used for large group of instructions to be repeated.	**Procedures are used for small group of instructions to be repeated.**
Object code is generated only once in memory.	**Object code is generated everytime the macro is called.**
CALL & RET instructions are used to call procedure and return from procedure.	**Macro can be called just by writing its name.**
Length of the object file is less	**Object file becomes lengthy.**
Directives PROC & ENDP are used for defining procedure.	**Directives MACRO and ENDM are used for defining MACRO**
More time is required for it's execution	**Less time is required for it's execution**
Procedure can be defined as **Procedure_name PROC** **----** **------** **Procedure_name ENDP**	**Macro can be defined as** MACRO-name MACRO [ARGUMENT ,………. ARGUMENT N] ------ **-------** **ENDM**
For Example **Addition PROC near** **------** **Addition ENDP**	**For Example** **Display MACRO msg** **---------** **ENDM**