MP unit test 2 FS19C0042

Second Year, First Shift

Course name - Microprocessor
Course code - IT 19207

Name - Omkar Phansopkar

Q 1. Any 3

a. Define CALL & RET instruction

→ (i) CALL - The CALL instruction is used to transfer execution to a subprogram or a procedure. It is used whenever we need to start execution of some other reusable group of instructions again & again.
Syntax: CALL subprogram_name

(ii) RET - The RET instruction stands for 'return'. This instruction is used at end of procedure or subprogram to transfer execution to the caller program
Syntax: RET

b. State any 2 differences between NEAR & FAR procedure

→

	NEAR procedure	FAR procedure
1	A NEAR procedure is a procedure which is in the same code segment as that of the call instruction.	1. A FAR procedure is a procedure which is in the different code segment from that of call instruction.
2.	A NEAR procedure call replaces old IP with new IP.	2. A FAR procedure call replaces old CS & IP pair with new CS & IP pair.
3	Syntax: CALL near_procedure Example: CALL Delay	3. Syntax: CALL far_procedure Example: CALL FAR PTR Delay

FS19C0042

1.

d. List different types of interrupt supported by 8086?

→ There are two main types of Interrupts in 8086:

(i) Hardware interrupts

Sub-divided as:

- Maskable interrupt (INTR)
- Non-maskable interrupt (NMI)

(ii) Software interrupts

Example, INT 0, INT 1, ... INT 255

e. Enlist types of Memory

→ Types of memory:

- ROM (Read only memory)
- RAM ~~(Read~~ (Random access memory)
  i.e. Read write memory

FS19C0042

Q2. Attempt any 2 out of 3

c. Differentiate between memory mapped I/O & I/O mapped I/O

→

	I/O mapped I/O	Memory mapped I/O
1.	Interfacing in which 8-bit address values are assigned to input/output devices is called I/O mapped I/O interfacing.	1. Interfacing in which memory address can be assigned in same manner as used in normal memory location is called memory mapped I/O
2.	I/O device ~~has~~ can only be accessed by IN and OUT instructions	2. I/O devices can now be accessed using any memory instruction eg. LDA, STA, etc.
3.	Only accumulator i.e. AL/AH/AX register can be used to transfer data with I/O device.	3. Any register can be used to transfer data with I/O device.
4.	Decoding is cheaper and easier due to lesser address lines	4. Decoding is expensive and more complex due to more address lines
5.	I/O mapped I/O works faster due to less delays	5. ~~I/O~~ Memory mapped I/O works slower due to more gates i.e. more delays.
6.	Popular technique in Microprocessors	6. Popular technique in Microcontrollers

FS19C0042

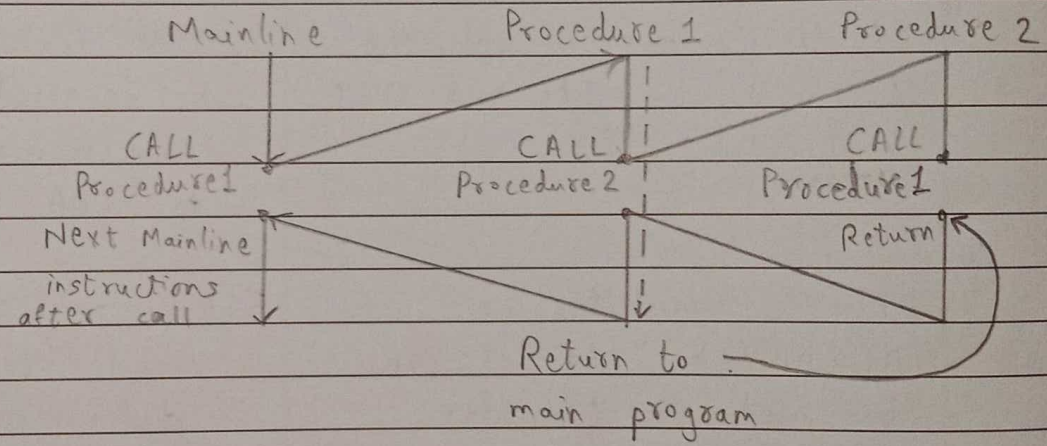
Q 2.

a. Describe re-entrant & recursive procedure with schematic diagram

→ (i) Re-entrant procedure

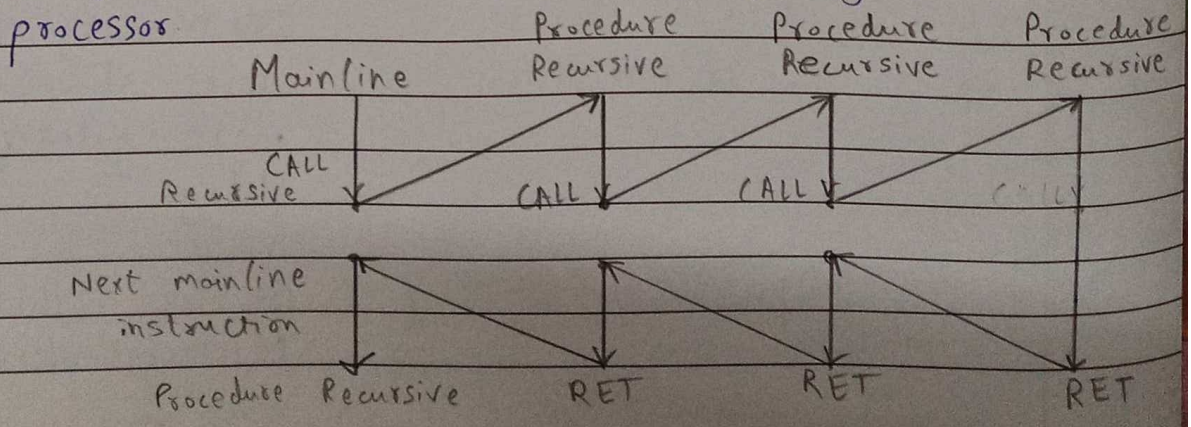
The ~~ren~~ re-entrant procedure is a procedure in which Procedure 1 is called from mainline program, then procedure 2 is called from procedure 1 and again procedure 1 is called from procedure 2. This can be seen in below diagram.

This is a called re-entrant procedure because procedure is re-entering into itself from another procedure which is also called inside its own body.



(ii) Recursive procedure

A recursive procedure is a procedure which calls itself. This results in procedure call to be generated from within procedures again & again & keep on executing until termination condition is reached. Recursive procedures are effective but take large ~~amount~~ amount of stack space & time putting extra load on processor



FS19C0042

Q3. Attempt any 1

a. Write an ALP to Sort 16 bit numbers in descending & ascending order.

→ ~~Prog~~

(i) Assembly Language program to Sort 16 bit numbers in Ascending Order:

```
data segment
    string1 dw 14h, 10h, 5h, 15h, 21h
    data ends
code segment
    assume cs:code, ds:data
    start:
        mov ax, data
        mov ds, ax
        mov bx, 5
        up1: lea si, string1
             mov cx, 4
             up: mov ax, [si]
                 ~~cmp~~
                 cmp ax, [si+2]
                 jc down
             xchg ax, [si+2]
             xchg ax, [si]
             down: add si, 2
                   loop up
                   dec bx
                   jnz up1
                   int 3

code ends
end start
```

FS19CO042

Q3. a.

(ii) Assembly language program to sort 16 bit numbers in descending order

```
data segment
        string1 dw 14h, 10h, 5h, 15h, 21h
        data ends
code segment
        assume cs:code, ds:data
        start:
            mov ax, data
            mov ds, ax
            mov bx, 5
            up1: lea si, string1
                 mov cx, 4
                 up: mov ax, [si]
                     cmp ax, [si+2]
                     jnc down
                 xchg ax, [si+2]
                 xchg ax, [si]
                 down: add si, 2
                       loop up
                       dec bx
                       jnz up1
                 int 3
    code ends
end start
```