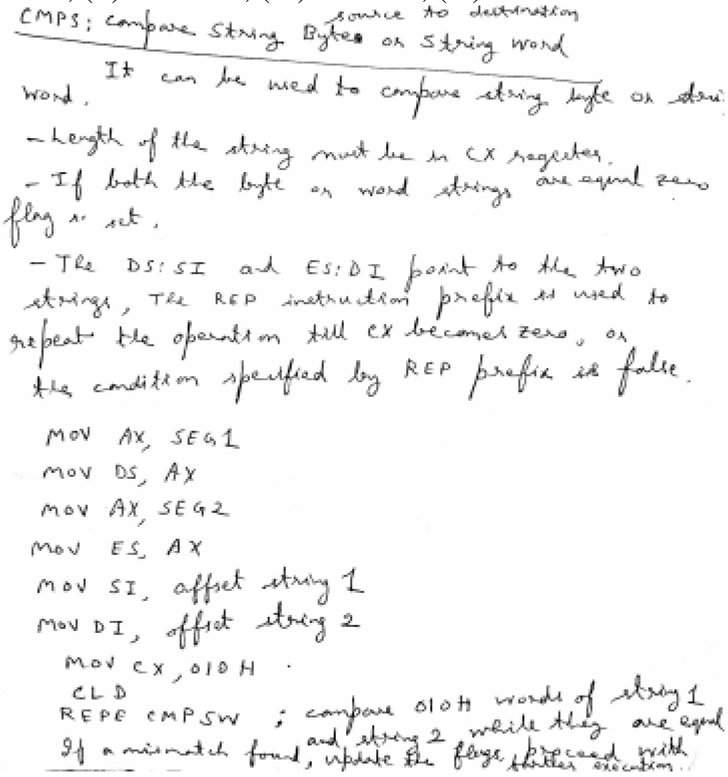
CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | |

Internal Assesment Test-II								
Sub:	MICROPROCESSORS						Code:	15EC42
Date:	16/04 / 2018	Duration:	90 mins	Max Marks:	50	Sem: 4th	Branch:	ECE(A,B, C,D)
Answer FIVE FULL Questions								

	Marks	OBE CO	OBE RBT
1. Briefly explain the operations of the following string instructions of 8086, indicating the initializations required to use them: (i) CMPSB, (ii) MOVSB, (iii) LODSB, (iv) STOSB.	[3+3+2+2]	CO2	L2

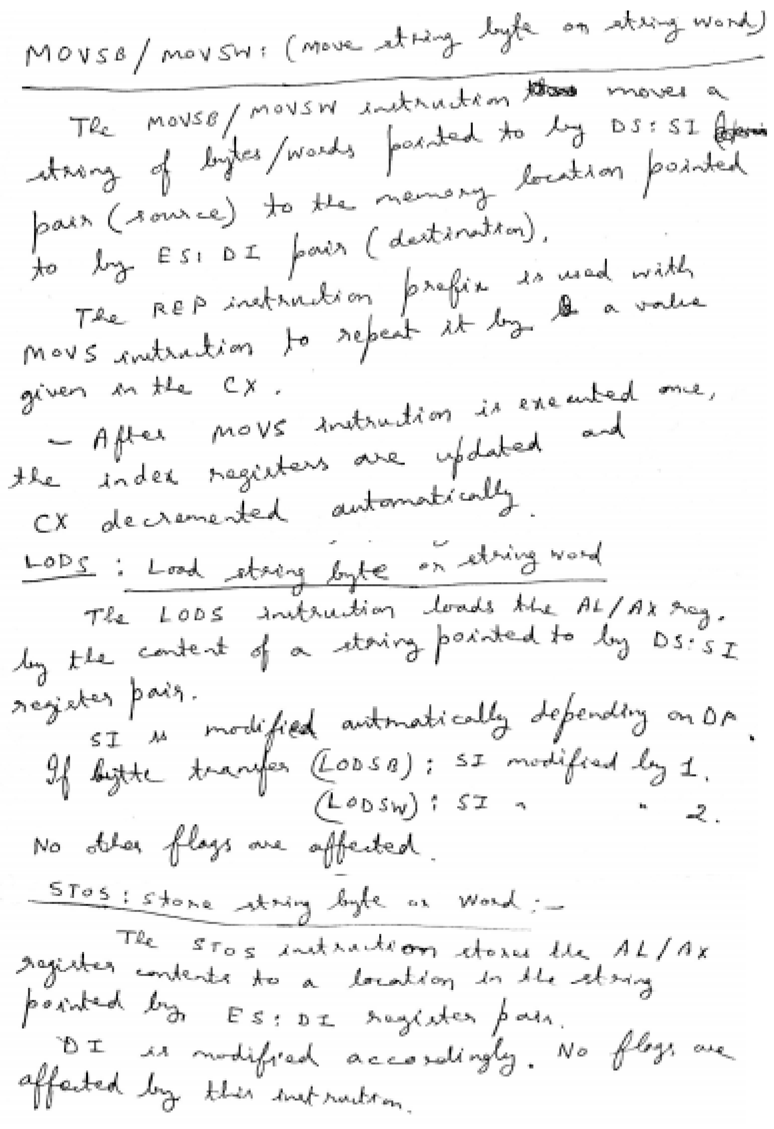


CMPS: compare String Bytes or String word source to destination word.

It can be used to compare string byte or stri:

- Length of the string must be in CX register.
- If both the byte or word strings are equal zero flag is set.
- The DS:SI and ES:DI point to the two strings, The REP instruction prefix is used to repeat the operation till CX becomes zero, or the condition specified by REP prefix is false.

```
MOV AX, SEG1
MOV DS, AX
MOV AX, SEG2
MOV ES, AX
MOV SI, offset string 1
MOV DI, offset string 2
MOV CX, 010H
CLD
REPE CMPSW   ; compare 010H words of string 1 and string 2 while they are equal
If a mismatch found, update the flags, proceed with further execution.
```



MOVSB / MOVSW : (move string byte or string word)

The MOVSB / MOVSW instruction ~~[illegible]~~ moves a string of bytes/words pointed to by DS : SI ~~[illegible]~~ pair (source) to the memory location pointed to by ES : DI pair (destination).

The REP instruction prefix is used with MOVS instruction to repeat it by ~~[illegible]~~ a value given in the CX.

– After MOVS instruction is executed once, the index registers are updated and CX decremented automatically.

LODS : Load string byte or string word

The LODS instruction loads the AL / AX reg. by the content of a string pointed to by DS : SI register pair.

SI is modified automatically depending on DF.

If ~~byte~~ transfer (LODSB) : SI modified by 1.

(LODSW) : SI " " 2.

No other flags are affected.

STOS : store string byte or Word :–

The STOS instruction stores the AL / AX register contents to a location in the string pointed by ES : DI register pair.

DI is modified accordingly. No flags are affected by this instruction.

2. What are assembler directives? Explain any five assembler directives with example. [10] CO2 L2

An assembler is a program used to convert an assembly language program into the equivalent machine code modules which may further be converted to executable codes.
- It decides the address of each label and substitutes the values for each of the constants and variables.

DB (Define Byte) :- used to reserve a byte or bytes of memory locations available in memory.
- while preparing .exe file this directive directs the assembler to allocate the specified no. of memory bytes to the said data type that may be a constant, variable, string etc.

n1 dB 01H, 02H, 03H, 04H
→ reserve four memory locations and initialize them with the specified values.

MSG dB 'Good Morning'
- Reserve the no. of bytes of memory equal to no. of characters.

DW (define word); Reserves Word or words of memory locations in the available memory.

n1 dW 1234H, 4567H, 78ABH, 045CH
- Reserves 4 words in memory, initialize the words with specified values.

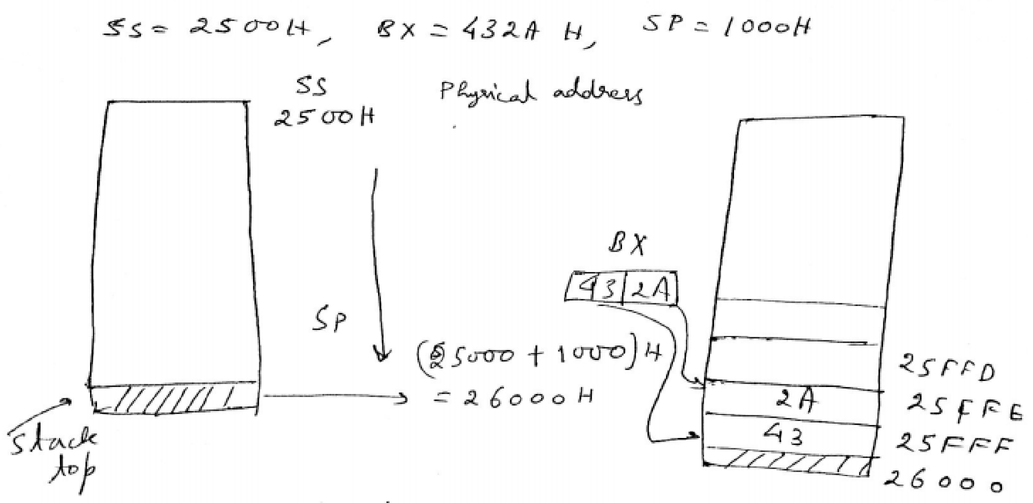
Assume: Assume Logical Segment Name
- It is used to inform the assembler the names of the logical segments to be assumed for different segments used in the program.

Assume CS: CODE ⇒ directs assembler that the machine codes are available in a segment named CODE.

END: END of Program: It marks the end of an assembly language program.
Assembler ignores the source lines available after END directive.

ENDP ~~[illegible]~~: End of procedure.
In ALP, subroutines are called procedures.
The procedures may be independent program modules which return particular results or values to the calling programs.

3.(a) Sketch the content of stack memory indicating the value of SP register before PUSH BX operation and after the PUSH BX operation. Assume SS = 2500 H, BX = 432AH and SP = 1000 H. [04] CO4 L3



(b) Write a program to generate 100 ms delay using 8086 microprocessor operating at 10 MHz frequency. Show calculation for delay. [06] CO3 L3

Solution: The time delay program is as follows:

	Instruction	T-states for e
	MOV BX, Count	**4**
L1:	**DEC BX**	**2**
	NOP	**3**
	JNZ L1	**16**
	RET	**8**

In this program, the instructions DEC BX, NOP, and JNZ L1 form the loop as they are executed repeatedly until BX becomes zero. Once BX becomes zero, the 8086 returns to the main program.

- Number of clock cycles for execution of the loop once (n) = 2 + 3 + 16 = 21
- Time required for the execution of loop once = n X T = 21 X 1/(10 X 10^6) = 2.1 μs
- Count = td/(n X T) = 100 X 10^-3 /(2.1 X 10^-6) =47619 (BA03).
- By loading BA03H in BX, the time taken to execute the delay program is approximately 100ms.
- The NOP included in the delay program is to increase the execution time

of the loop. To get more delay, the number of NOP instructions in the delay loop can be increased.

The exact delay obtained using this time delay subroutine can be calculated as shown below,

The MOV BX, Count & RET instructions in the delay program are executed only once. The JNZ instruction takes 16 T-states when the condition is satisfied (i.e. ZF= 0) and 4 T – states when the condition is not satisfied, which occurs only once.

Exact delay = [4 x 0.1 + (2+3) x 47619x 0.1 + 16 x 47618 x0.1 + 4 X 0.1 + 8 x 0.1] µs

=99.9999ms

=100ms.

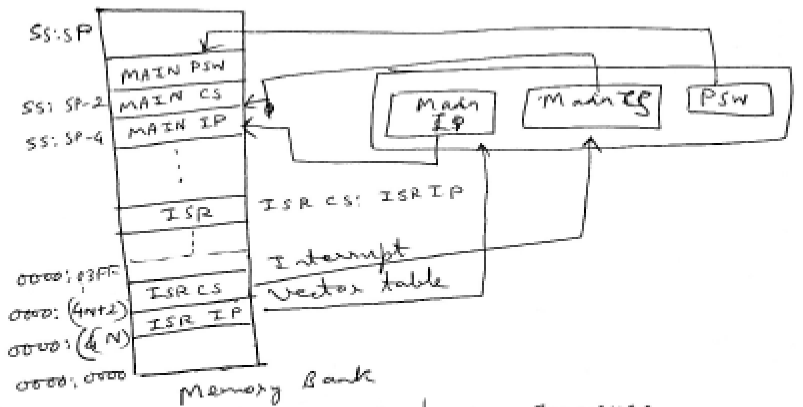
4.(a)	What are the methods that can be used to pass parameters to a procedure? Explain anyone of them with an example. Passing parameters to procedures (i) Using global declared variable (ii) Using registers of CPU architecture (iii) Using memory locations (reserved) (iv) Using stack (v) Using PUBLIC and EXTRN (i) Using global declared variables assume CS: code1, DS: Data Data segment Number equ 77h global Data ends code1 segment start: mov ax, data ~~start~~ mov DS, ax : mov AX, number : code ends assume cs: code 2 code 2 segment mov AX, DATA mov DS, AX mov BX, NUMBER code2 ends end start	[06]	CO3	L3
(b)	Differentiate between procedure and macro. Macro i) complete code of the instruction string is inserted at each place where the macro-name appears. ii) Hence lengthy EXE file iii) Does not use stack; Subroutine i) control of execution is transferred to the SR, every time it is called. ii) smaller EXE file. iii) Utilizes stack service.	[04]	CO3	L2
5.	What are the sequence of actions taken by 8086 and the device, when a device interrupts 8086 over INTR line? Explain about the software and reserved internal interrupts of 8086.	[10]	CO4	L3

	Interrupt – To break the sequence of operation. While the CPU is executing a program, an 'interrupt' breaks the normal sequence of execution of instructions, diverts its execution to some other program called Interrupt Service Routine (ISR). While one interrupt is being served, another interrupt may appear. – ISR are the programs to be executed by interrupting the main program execution of the CPU, after an interrupt request appears. After the execution of ISR, the main program continues the execution further from the point at which it was interrupted. — Say, an external device interrupts the CPU at the interrupt pin (NMI or INTR) while CPU is executing an instruction of a program. – CPU completes the execution of the current instruction – IP is incremented to point to the next instruction.			

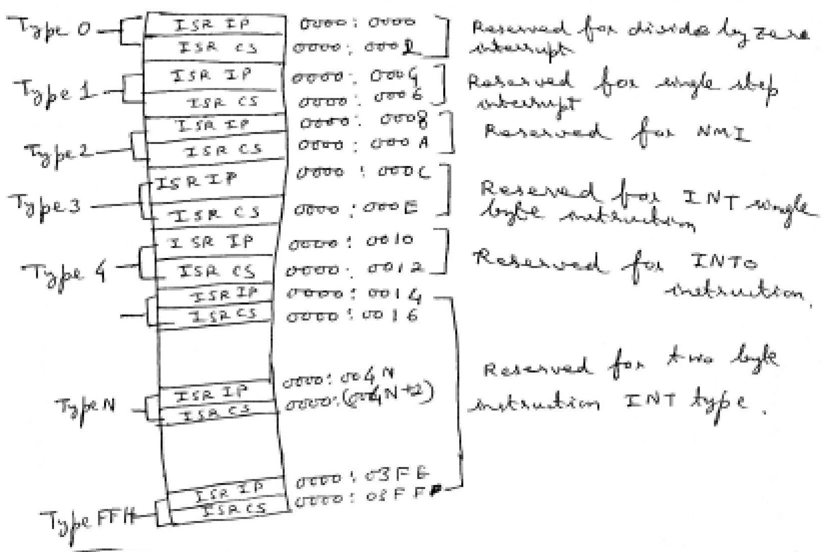
- If it is NMI, TRAP or divide by zero CPU acknowledges immediately on its $\overline{INTA}$ pin.
- If it is INT request, CPU checks IF.
- If IF ~~[illegible]~~ =1, $\overline{INTA}$ goes low (acknowledged)
- If, IF=0, interrupt requests are ignored.
- CPU computes the vector address from the type of the interrupt, that may be passed to the interrupt structure of the CPU internally (in case of S/W interrupts, NMI, TRAP and divide by zero) or externally
- IP and CS point to the next instruction to be executed after ISR, CS, IP and PSW pushed to stack.

~~IP~~ - IF is cleared.

- TF is cleared after every response to the single step interrupt.
- The new address of ISR found from the interrupt vector table.
- ISR executes.
- During ISR execution if some other interrupt to be ~~service~~ served, IF = 1 once more by ISR of the 1st interrupt.
- If IF is not ~~[illegible]~~set the subsequent interrupts won't be acknowledged, till the current one is completed.
- At the end of ISR, the last instruction should by IRET.
- When IRET is executed, the contents of flags, IP and CS are retrieved to the respective registers.
- Execution continues.



Interrupt Response Sequence



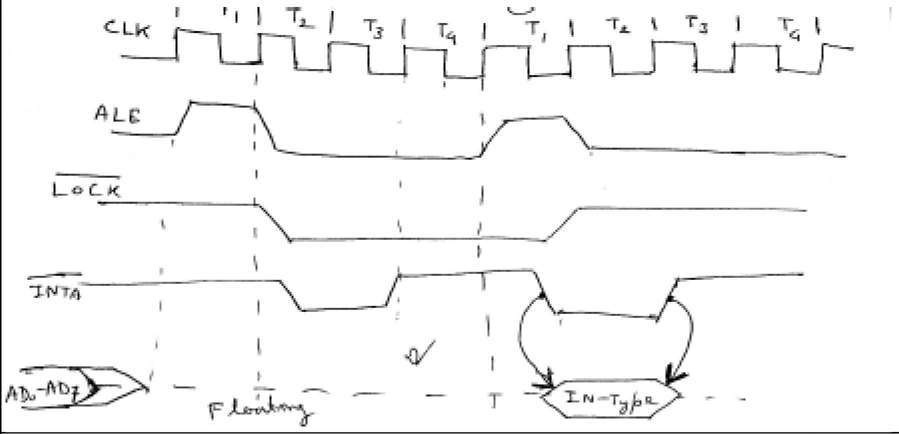
— For the INTR signal, to be responded to in the next instruction cycle, it must go high in the last clock cycle of the current instruction or before that

— If INTR request appears after the last clock cycle of of the current instruction, it will be responded to after the execution of the next instruction,

IF = 1 → processor responds to INTR interrupt.
IF = 0, → " does not respond to INTR.

Once processor responds to INTR, IF = 0

– External signal interrupts the processor.
– ALE pulse appears after interrupt signal, preventing use of bus for any other purpose.
– $\overline{LOCK}$ goes low at the trailing edge of the first ALE pulse.
– $\overline{LOCK}$ remains low till the next machine cycle
– At trailing edge of $\overline{LOCK}$, $\overline{INTA}$ goes low and remains ~~for~~ low for two clock states before returning back to the high state.
– It remains high till the start of the next machine cycle. (till next trailing edge of ALE)
– Then $\overline{INTA}$ again goes low, remains low for two states, before returning to the high state.
– The first trailing edge of ALE floats the ~~bus~~ bus $AD_0 - AD_7$, while the second trailing edge prepares the bus to accept the type of ~~the~~ interrupt.
— The type of interrupt remains on the bus for a period of two cycles.



6.(a)	Write an ALP to convert Hexadecimal number to BCD number.	[06]	CO4	L3

```
                    .MODEL SMALL
                    .DATA
      HEXN DW       0FFFFH
      TEN           DW     000AH
```

```
BCD         DB 3 DUP (00H)

            .CODE
            MOV AX,@DATA
            MOV DS, AX

            MOV CL, 04H
            MOV AX, HEXN

AGAIN:      XOR DX, DX
            DIV TEN
            MOV BCD[DI], DL

            XOR DX, DX
            DIV TEN
            ROL DL, CL
            OR BCD[DI], DL

            INC DI
            CMP AX, 00H
            JNE AGAIN

            MOV AH, 4CH
            INT 21H
            END
```

(b)	Create a macro that would find the logical NAND value of two operands.	[04]	CO3	L3

```
    .MODEL SMALL
            .DATA
NUM1        DW     4556H
NUM2        DW     0FFFFH
RES         DW     0000H
            .CODE
LOGNAND     MACRO N1, N2
            MOV AX,N1
            MOV BX,N
            AND AX, BX
            NOT AX
            ENDM

            MOV AX, @DATA
            MOV DS, AX2


LOGNAND     NUM1, NUM2
            MOV RES, AX

            MOV AH, 4CH
            INT 21H
            END
```

7.	Write an alp which replaces all occurrences of character '-' in a given string by '*'.	[10]	CO2	L3

```
    .MODEL SMALL
                .DATA
    SOURCE  DB      'C-M-R', '$'
    COUNT   EQU     5H
    SEARCH  DB      '-'
                .CODE
                MOV AX,@DATA
                MOV DS, AX
                MOV ES, AX
                LEA DI, SOURCE
                MOV AL, SEARCH
                MOV CX, COUNT
                CLD
    BACK:       SCASB
                JNZ NEXT
                DEC DI
                MOV BYTE PTR [DI], '*'
                INC DI
    NEXT:       LOOP BACK
                LEA DX, SOURCE
                MOV AH, 09H
                INT 21H
                MOV AH, 4CH
                INT 21H
                END
```

8.(a)	Write an ALP to find whether the given number is 2 out of 5 code.	[06]	CO2	L3

```
.MODEL SMALL
            .DATA
NUM  DB     09H
MSG1 DB     'NUMBER IS 2 OUT OF 5 CODE', '$'
MSG2 DB     'NUMBER IS NOT A 2 OUT OF 5 CODE','$'
             .CODE
            MOV AX,@DATA
            MOV DS,AX
            MOV AL,NUM
            MOV CX,03H
FIRST:      ROL AL,01H
            JC LAST
            LOOP FIRST
            MOV CX,05H
SECOND:     ROL AL,01H
            ADC BL,00H
            LOOP SECOND
            CMP BL,02H
            JNE LAST
            LEA DX, MSG1
            JMP SKIP
LAST:       LEA DX, MSG2
SKIP:       MOV AH, 09H
            INT 21H
            MOV AH,4CH
            INT 21H
            END
```

(b)	Write an ALP which computes the factorial of an 8-bit number. The factorial value to be maximum 8-bit in size.	[04]	CO2	L3

```
            .MODEL SMALL
            .STACK 64H
            .DATA
NUM1        DB     05H
FACTRES     DB 00H
            .CODE
            MOV AX,@DATA
            MOV DS, AX
            MOV AL, NUM1
            CALL FACTN
            MOV FACTRES, AL

            MOV AH, 4CH
            INT 21H

FACTN       PROC NEAR
            MOV BL, AL
            MOV AL, 01H
            CMP BL, 00H
            JE   L1
AGAIN:      MUL BL
            DEC BL
            JNZ AGAIN
L1:         RET
FACTN       ENDP
            END
```