Course Code	:	**BCSL-022**
Course Title	:	**Assembly Language Programming Lab**
Assignment Number	:	**BCA(2)/L-022/Assignment/16-17**
Maximum Marks	:	**50**
Weightage	:	**25%**
Last Dates for Submission	:	**15th October, 2016 (For July 2016 Session)** **15th April, 2017 (For January 2017 Session)**

This assignment has two questions of total of 40 marks. Rest 10 marks are for viva voce. Please go through the guidelines regarding assignments given in the programme guide for the format of presentation.

1. , Design a two bit counter circuit that counts from 1, that is, it will have state 01, 10, 11 only. The initial state of the counter may be assumed to be 01. The counter will be in following successive states: 01, 10, 11, 01, 10, 11, 01, 10, 11 ... Use any flip flop to design the circuit. You must design them using state transition diagram and Karnaugh's map. , ***(10 Marks)***

Ans:

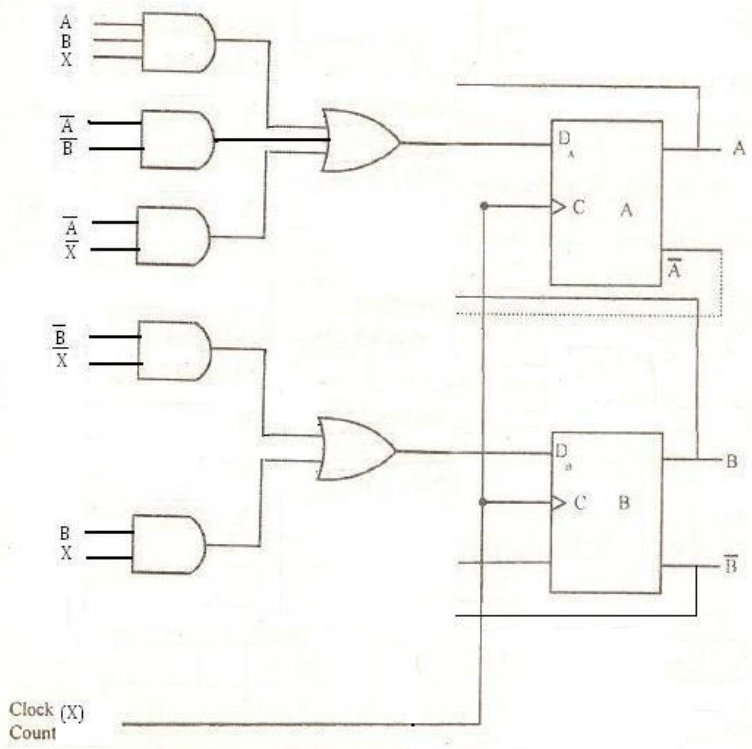
A sequential circuit is specified by a time sequence of external inputs, external outputs and internal flip-flop binary states. Thus firstly, a *state table and state diagram* is used to describe behaviour of the circuit.

Present State		Input	Next State		Flip-Flops Inputs	
A	B	X	A	B	D_A	D_B
0	0	0	0	0	1	1
0	0	1	0	1	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	1	0	0	1
1	0	1	1	1	0	0
1	1	0	1	1	0	0
1	1	1	0	0	1	1

There are 2 flip-flop inputs for counter i.e. A, B. The next state of flip-flop is given in the table. DA indicates the flip flop input corresponding to flip-flop-A. This counter requires 2-flip-flops.

From this the flip flop input equations are simplified using K-Maps as shown below.

	00	01	11	10
0	1 0	1	1 3	2
1	1 4	5	1 7	6



Logic Diagram for down counter.

The logic circuit can be made with 2 D flip flops, 2 OR gates & 4 AND gates.

2. , Write and run the following programs using 8086 assembly language.

(a) Write and run an Assembly language program that converts lowercase alphabets in a given input string to uppercase. The input may consist of uppercase alphabets, special characters and lowercase alphabets. For example, for the input string A@abAYaf, the output will be A@ABAYAF. You may assume that the string is available in the memory and output is stored in a memory array. , *(10 Marks)*

Ans:

```
DATA SEGMENT
   MSG1 DB 10,13,'ENTER ANY STRING :- $'
   MSG2 DB 10,13,'ENTERED STRING IS :- $'
   MSG3 DB 10,13,'CONVERTED STRING IS : $'
    P1 LABEL BYTE
```

```
    M1 DB 0FFH
    L1 DB ?
    P11 DB 0FFH DUP ('$')
DATA ENDS
DISPLAY MACRO MSG
    MOV AH, 9
    LEA DX, MSG
    INT 21H
ENDM
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START:
        MOV AX, DATA
        MOV DS, AX

        DISPLAY MSG1

        LEA DX, P1
        MOV AH, 0AH
        INT 21H

        DISPLAY MSG2

        DISPLAY P11

        DISPLAY MSG3

        LEA SI, P11

        MOV CL, L1
        MOV CH, 0
CHECK:
        CMP [SI], 41H
        JB DONE

        CMP [SI], 5BH
        JB LWR

        CMP [SI], 61H
        JB DONE

        CMP [SI], 7BH
        JG DONE

UPR:    SUB [SI], 20H
        JMP DONE
LWR:    ADD [SI], 20H
```

```
DONE:   INC SI
        LOOP CHECK

        DISPLAY P11

        MOV AH, 4CH
        INT 21H
CODE ENDS
END START
```

(b) Write and run (using appropriate calling program) a near procedure in 8086 assembly language that converts 2 ASCII digits stored in two registers (say BH and BL) into an equivalent binary number. For example, if the BH and BL registers contain digits 4 and 5 respectively, then the binary number obtained will be 0010 1101 which is 45 in decimal. The parameters should be passed using registers and the result should be returned in AL register. , ***(12 Marks)***

Ans:

```
DATA_SEG SEGMENT
        BCD DB 25h     ; storage for BCD value
        BIN DB ?       ; storage for binary value
DATA_SEG ENDS
STACK_SEG SEGMENT STACK
        DW 200 DUP(0)          ; stack of 200 words
        TOP_STACK LABEL WORD
STACK_SEG ENDS
CODE_SEG SEGMENT
        ASSUME CS:CODE_SEG, DS:DATA_SEG, SS:STACK_SEG
        START: MOV AX, DATA_SEG        ; Initialise data segment
        MOV DS, AX                     ; Using AX register
        MOV AX, STACK_SEG              ; Initialise stack
        MOV SS, AX                     ; Segment register. Why; stack?
        MOV SP, OFFSET TOP_STACK       ; Initialise stack pointer
        MOV AH, BCD
        CALL BCD_BINARY                ; Do the conversion
        MOV BIN, AH                    ; Store the result
BCD_BINARY PROC NEAR
        PUSHF
        PUSH   BX      ; and registers used in procedure
        PUSH   CX      ; before starting the conversion; Do the conversion
        MOV BH, AH  ; Save copy of BCD in BH
```

```
        AND BH, 0Fh    ; and mask the higher bits. The lower digit        is in BH
        AND AH, 0F0h ; mask the lower bits. The higher digit is in AH   ; but in upper 4 bits.
        MOV CH, 04     ; so move upper BCD digit to lower
        ROR AH, CH     ; four bits in AH
        MOV AL, AH   ; move the digit in AL for multiplication
        MOV BH, 0Ah   ; put 10 in BH
        MUL BH          ; Multiply upper BCD digit in AL; by 0Ah in BH, the result is in AL
        MOV AH, AL    ; the maximum/ minimum number would not ; exceed 8 bits so move AL to
        ADD AH, BH   ; Add lower BCD digit to MUL result ; End of conversion, binary result in
        POP CX          ; Restore registers
        POP BX
        POPF
        RET     ; and return to calling program
BCD_BINARY ENDP
CODE_SEG ENDS
END START
```

(c) Write and run an 8086 assembly language program displays a string stored in memory. You must use appropriate interrupt for the same.

Ans:

```
DATA SEGMENT
PASSWORD DB 'FAILSAFE'   ; source string
DESTSTR DB 'FEELSAFE'       ; destination string
MESSAGE DB 'String are equal $'
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS: DATA, ES: DATA
MOV AX, DATA
MOV DS, AX   ; Initialise data segment register
MOV ES, AX   ; Initialise extra segment register
LEA SI, PASSWORD   ; Load source pointer
LEA DI, DESTSTR       ; Load destination pointer
MOV CX, 08    ; Load counter with string length
CLD     ; Clear direction flag so that comparison is
REPE CMPSB ; Compare the two string byte by byte
JNE NOTEQUAL ; If not equal, jump to NOTEQUAL
```

```
MOV AH, 09 ; else display message
MOV DX, OFFSET MESSAGE ;
INT 21h ; display the message
NOTEQUAL: MOV AX, 4C00h ; interrupt function to halt
INT 21h
CODE ENDS
END
```