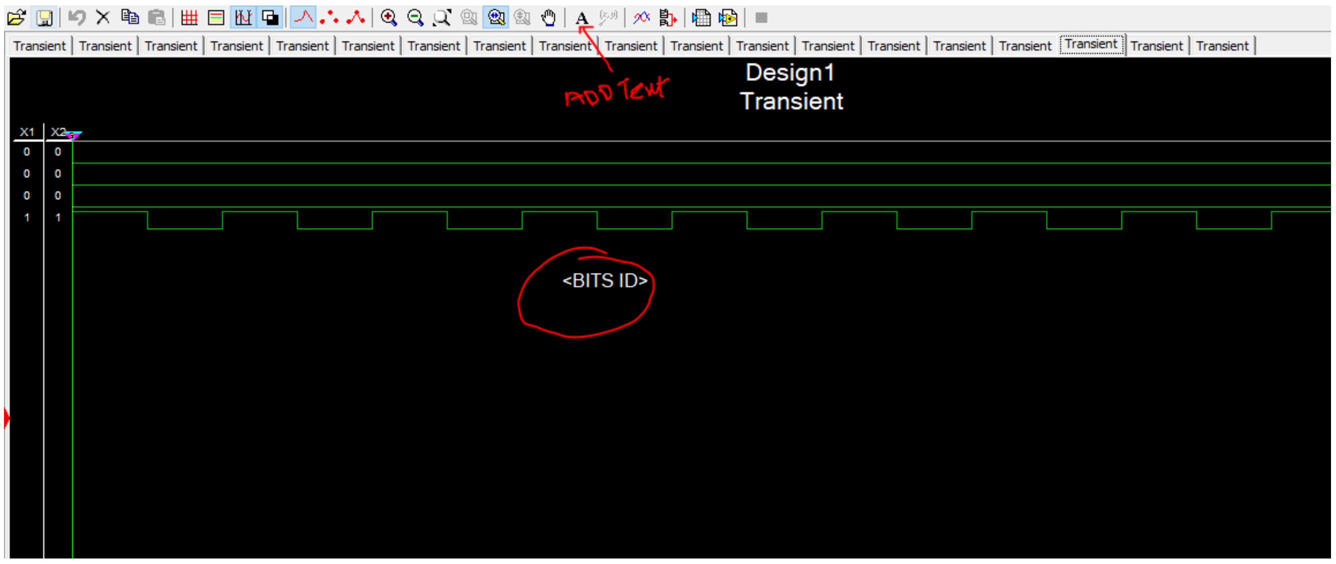
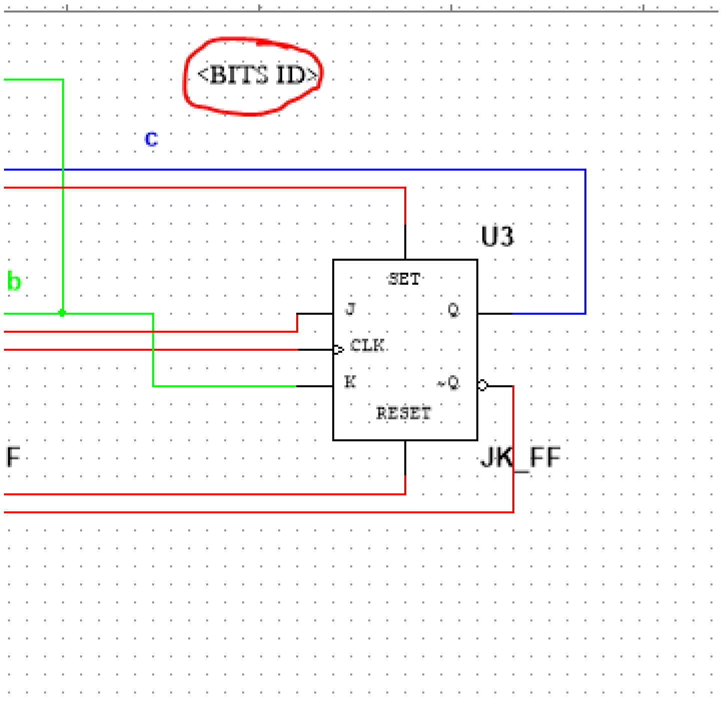
Lab Assignment DEMP

S2-19 CSIWZC263/S2-19 SSWTZC263

Show the implementation of these circuits using Multisim and the resultant wave forms. You are required to take the screenshot of your circuit and waveform and paste it as a word file with your BITS ID mentioned on the waveform plots. The screen shot of the circuit should have the bits ID as shown. This can be done as explained below.



For the circuit file the command is ctrl+Alt+A



Instruction:

There are two sections of the assignment.

The students are required to select two experiments from both the digital and assembly section each, and submit it on the e-learning portal. Only the assignment submitted to the e-learning portal will be evaluated. In making selection of two assignments, from the digital part, one has to select one experiment, from each of the combination and sequential part

The assignment component carries 10 marks and each question is of 2.5 marks each.

The last date of submission will be 24th September 2021, midnight

The students are advised not to wait till the last date for the submission, as there are limited number of licenses and last hour crowding may prevent from accessing the tool.

Digital Design

1. 2-bit Comparator Design a circuit that takes two unsigned 2-bit numbers (a and b), and displays one of greater(a < b), lesser (a < b) or equal (a == b) signals.

2. Implement the function using Y= Σ(1,2,3,4,6.7,9,14,15) the following list of components [3]

Slno	Component	quantity
1	4X1 Mux	1
2	2 input OR gates	2
3	NOT gate	2
4	2 input AND gate	1

3. Desgin a synchronous BCD counter using T flip flop and verify its performance using Multisim
4. Design a sequence detector circuit that produces an output pulse z=1 whenever the sequence 1111 appears. Overlapping sequences are accepted; for example, if the input is 010101111110 the output is 000000001110. Realise the logic using JK flip-flop , Verify the result using multisim and use four channel oscilloscope to show the waveforms as result. The waveform should include the clk, IP signal and the states of the flip-flop.

Assembly Language Programing

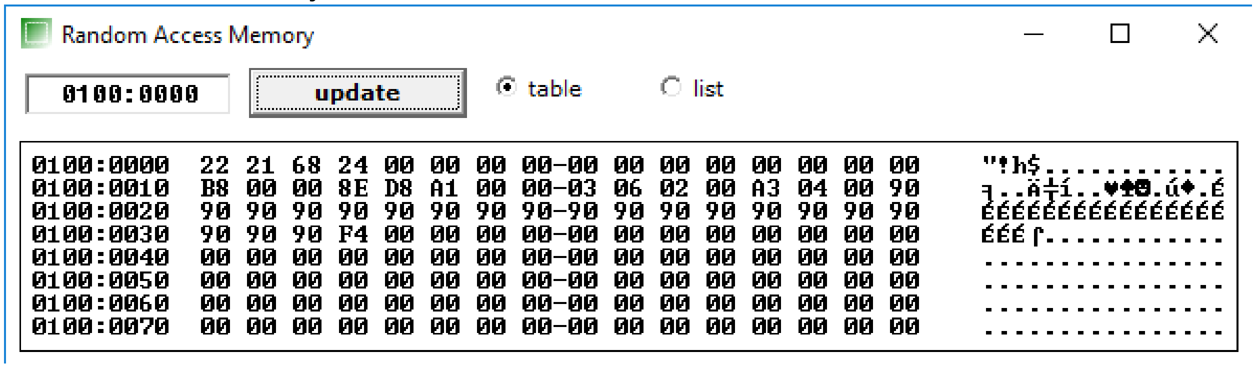
5. Write an assembly language program to multiply the two 16-bit words in the memory location called Multiplicand and Multiplier. The result is stored in the memory location product

6. Write an assembly language program to take average of two Number named ONE-NO and SECOND- NO and put the result in the memory location.

7. Write an assembly language program to find the LCM of Two Numbers

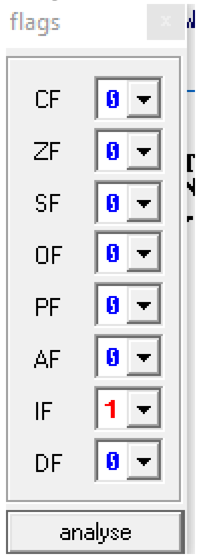
For all the assembly language assignment us 8086 emulator. You are required to show the following screen shots . Bits ID needs to be mentioned on all the screen shots

a. Random access memory,



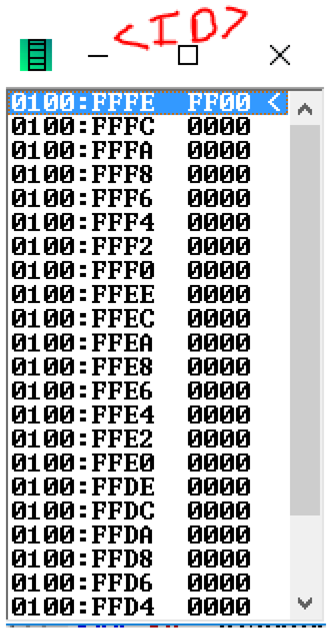
```
0100:0000  22 21 68 24 00 00 00 00-00 00 00 00 00 00 00 00    "!h$............
0100:0010  B8 00 00 8E D8 A1 00 00-03 06 02 00 A3 04 00 90    ╕..Ä╪í..♥♠☻.ú♦.É
0100:0020  90 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90    ÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉ
0100:0030  90 90 90 F4 00 00 00 00-00 00 00 00 00 00 00 00    ÉÉÉ⌠............
0100:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
0100:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
0100:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
0100:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
```

b. Flag


flags

CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

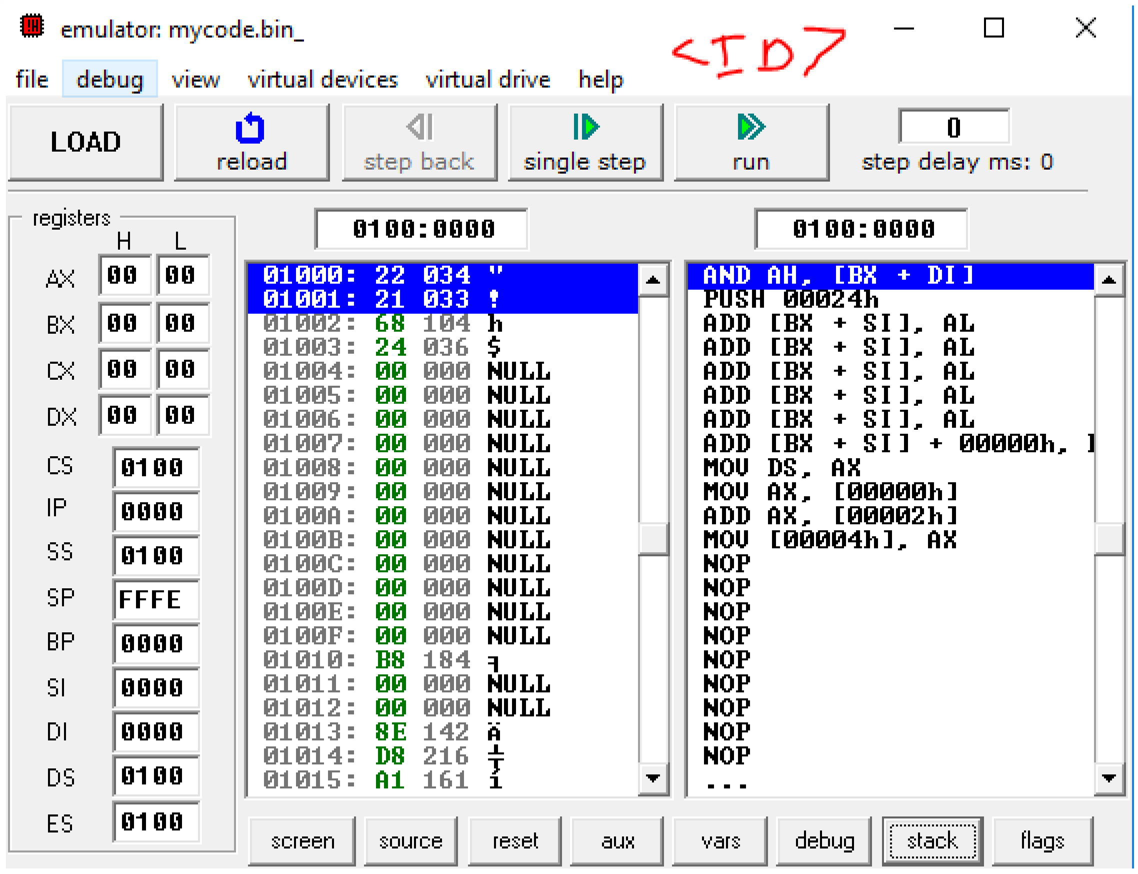
c. Stack



```
0100:FFFE  FF00 <
0100:FFFC  0000
0100:FFFA  0000
0100:FFF8  0000
0100:FFF6  0000
0100:FFF4  0000
0100:FFF2  0000
0100:FFF0  0000
0100:FFEE  0000
0100:FFEC  0000
0100:FFEA  0000
0100:FFE8  0000
0100:FFE6  0000
0100:FFE4  0000
0100:FFE2  0000
0100:FFE0  0000
0100:FFDE  0000
0100:FFDC  0000
0100:FFDA  0000
0100:FFD8  0000
0100:FFD6  0000
0100:FFD4  0000
```

d. Emulator

BITS Pilani
Pilani | Dubai | Goa | Hyderabad



<ID7

```
registers
      H    L
AX   00   00
BX   00   00
CX   00   00
DX   00   00
CS   0100
IP   0000
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

```
0100:0000

01000: 22 034 "
01001: 21 033 !
01002: 68 104 h
01003: 24 036 $
01004: 00 000 NULL
01005: 00 000 NULL
01006: 00 000 NULL
01007: 00 000 NULL
01008: 00 000 NULL
01009: 00 000 NULL
0100A: 00 000 NULL
0100B: 00 000 NULL
0100C: 00 000 NULL
0100D: 00 000 NULL
0100E: 00 000 NULL
0100F: 00 000 NULL
01010: B8 184 ╕
01011: 00 000 NULL
01012: 00 000 NULL
01013: 8E 142 Ä
01014: D8 216 ╪
01015: A1 161 í
```

```
0100:0000

AND AH, [BX + DI]
PUSH 00024h
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI] + 00000h, 
MOV DS, AX
MOV AX, [00000h]
ADD AX, [00002h]
MOV [00004h], AX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```