INDIAN INSTITUTE OF TECHNOLOGY

KHARAGPUR

DEPARTMENT OF ELECTRONICS AND ELECTRICAL COMMUNICATION

EC49001

MICROCONTROLLER SYSTEMS LABORATORY

ASSIGNMENT 2

SHIHKAR MOHAN

18EC10054

E & ECE

(B. Tech)

2021 – 2022

Questions

1. Store the binary truth table outputs of a four input combinational function at locations 30H to 3FH. Design a circuit that uses 4 LEDs to represent the inputs and a fifth LED for the output, such that the program will cyclically show all the inputs from 0000 to 1111 as well as the corresponding outputs.
2. Consider a switch input `sw0`. If the switch is pressed, all 5 LED's blink and when released, the operation comes back to its normal mode starting at the combination 0000.

Objectives

1. Implement a truth table with the help of LED's and perform iteration through its inputs in assembly code using a 4-bit counter.
2. Add a switch to run a special routine and reset the truth table.

Description

1. We create a counter for traversing through the truth table. This counter ranges from 0 to 15, which we add to our accumulator. `ACC` now has information regarding the 4-bit counter in its last 4 bits. Then we swap the first and last 4 bits of `ACC` so that bits 7,6,5,4 represent the inputs ABCD, and we add the output stored in the ram RAM. E.g. if at the 11th input (3BH) we have 1, the 8 bits of our `ACC` would be `1011001`. This perfectly describes what our LEDs should output, so we simply complement this using the `CPL` operation and move it to P1, where it is displayed accordingly.

2. At the beginning of every iteration we check if `sw0` is active/pressed (assigned to `P2.0`). If it is pressed, we move to the `EXCEPTION` subroutine. All 5 LEDs (at position 7, 6, 5, 4 and 1) high means the value of `P1` is `00001110` (high LEDs have low bits since P1 is active low), so we move the value 14 to P1 to make all the LEDs high. This is followed by a `DELAY_MODULE`, which just consists of a single memory operation performed multiple times by nested loops. To turn all LEDs off, we simply move the value 255 to `P1`, followed by another

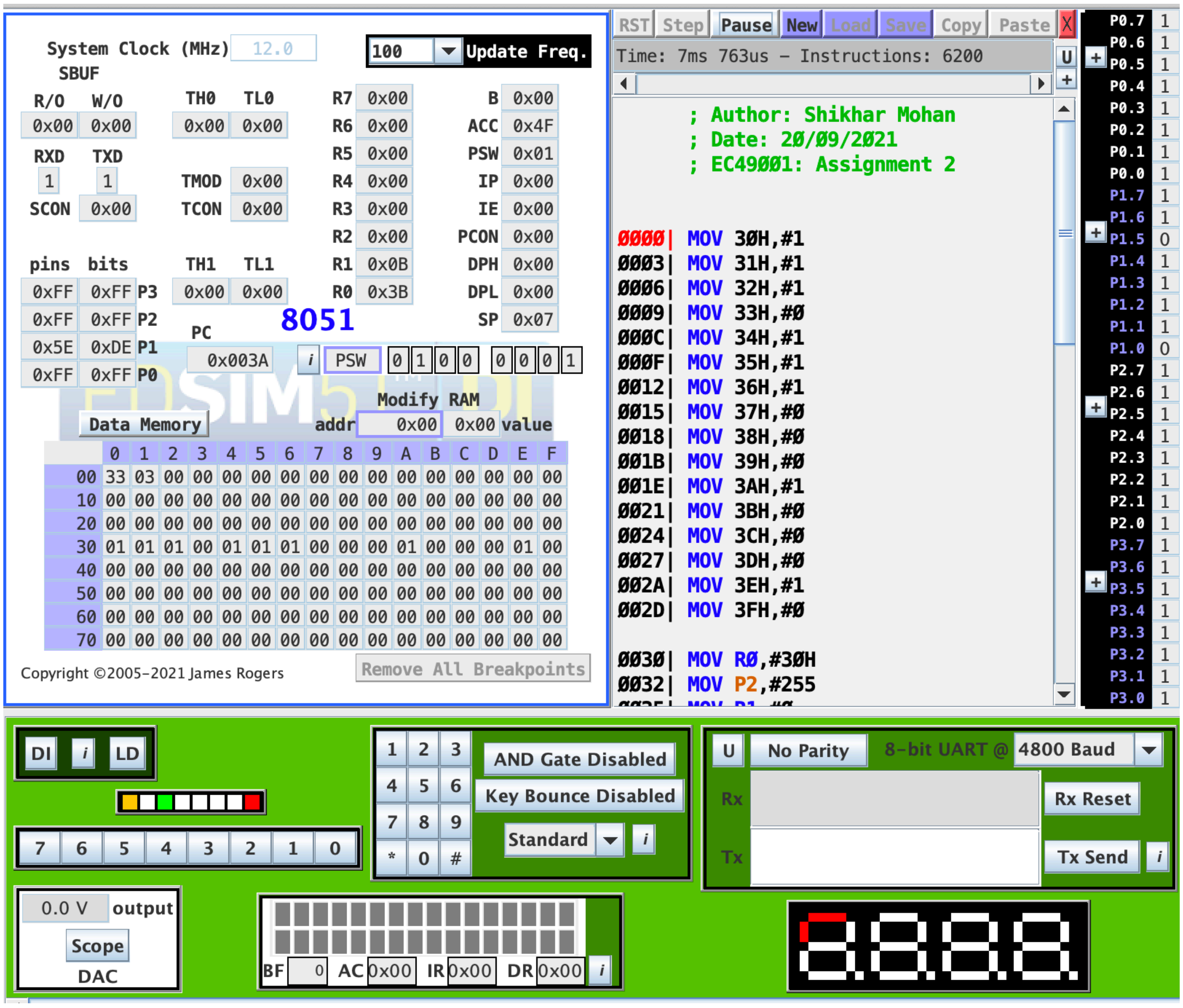
call to the DELAY_MODULE subroutine. This repeats indefinitely until the switch is released, in which case the iteration starts again from 0000.

Code

```
; Author: Shikhar Mohan
; Date: 20/09/2021
; EC49001: Assignment 2


MOV 30H,#1
MOV 31H,#1
MOV 32H,#1
MOV 33H,#0
MOV 34H,#1
MOV 35H,#1
MOV 36H,#1
MOV 37H,#0
MOV 38H,#0
MOV 39H,#0
MOV 3AH,#1
MOV 3BH,#0
MOV 3CH,#0
MOV 3DH,#0
MOV 3EH,#1
MOV 3FH,#0

MOV R0,#30H
MOV P2,#255
MOV R1,#0

OUTER_LOOP:
JNB P2.0,EXCEPTION
MOV A,R1
SWAP A
ADD A,@R0
CPL A
MOV P1,A

MOV A,R1
SUBB A,#15
JZ RESET
INC R0
INC R1
JMP OUTER_LOOP

RESET:
MOV R0,#30H
MOV R1,#00H
JMP OUTER_LOOP

EXCEPTION:
MOV P1,#14
CALL DELAY_MODULE
MOV P1,#255
CALL DELAY_MODULE
```

```
JMP RESET

DELAY_MODULE:  MOV R5, #1
DELAY1:   MOV R6, #1
DELAY2:   MOV R4, #32
STAY:     DJNZ R4, STAY
          DJNZ R6, DELAY2
          DJNZ R5, DELAY1
          RET
```

Screenshots



```
System Clock (MHz) 12.0        100  Update Freq.
SBUF
R/O   W/O      TH0   TL0     R7  0x00        B  0x4F?
0x00  0x00     0x00  0x00    R6  0x00      ACC  0x4F
RXD   TXD                    R5  0x00      PSW  0x01
 1     1       TMOD  0x00    R4  0x00       IP  0x00
SCON  0x00     TCON  0x00    R3  0x00       IE  0x00
                             R2  0x00     PCON  0x00
pins  bits     TH1   TL1     R1  0x0B      DPH  0x00
0xFF  0xFF P3  0x00  0x00    R0  0x3B      DPL  0x00
0xFF  0xFF P2                8051           SP  0x07
0x5E  0xDE P1  PC
0xFF  0xFF P0  0x003A    PSW 0 1 0 0 0 0 0 1
                         Modify RAM
Data Memory         addr 0x00  0x00 value
```

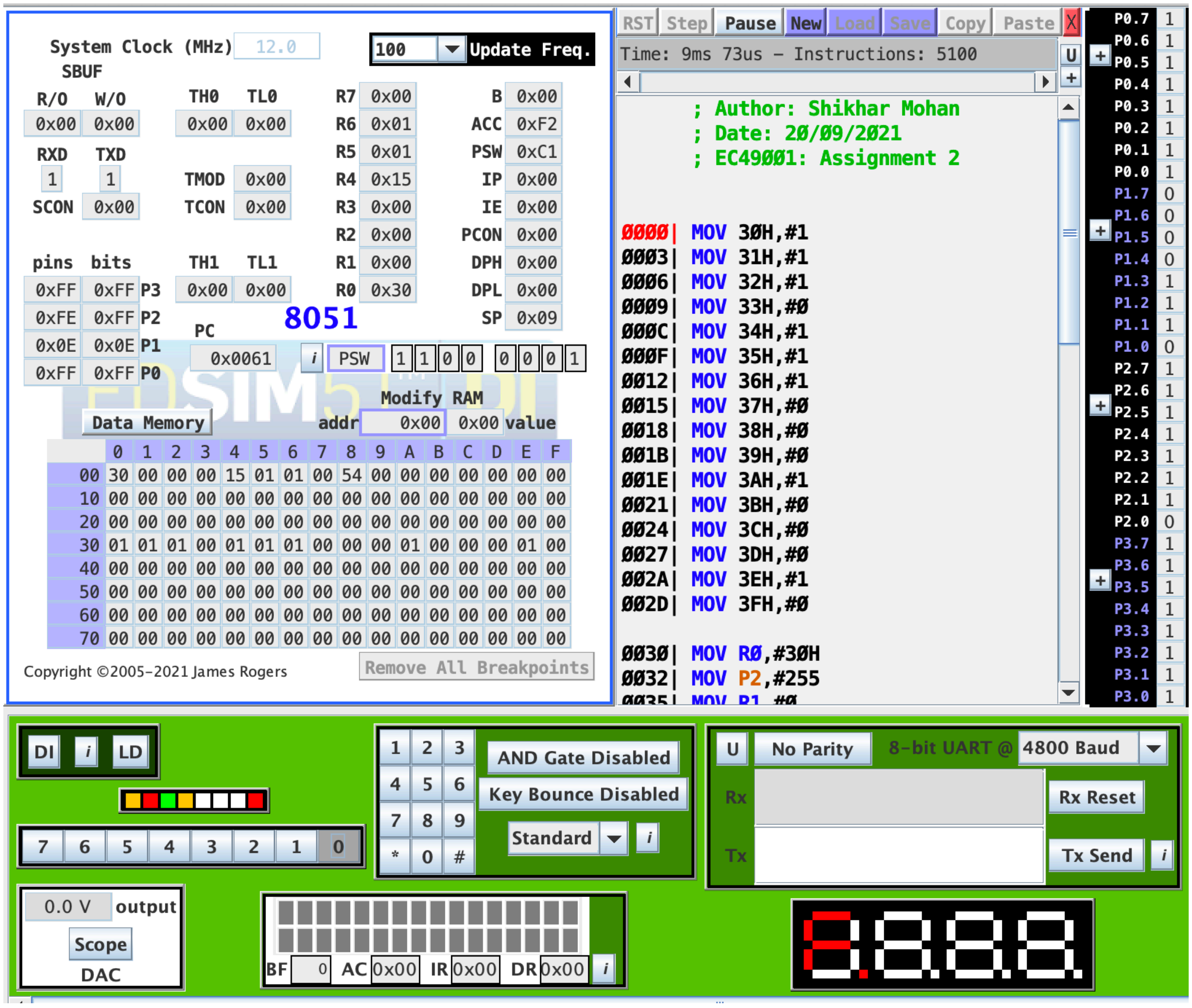
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	33	03	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	01	01	01	00	01	01	01	00	00	00	01	00	00	00	01	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2021 James Rogers

```
Time: 7ms 763us - Instructions: 6200

      ; Author: Shikhar Mohan
      ; Date: 20/09/2021
      ; EC49001: Assignment 2

0000| MOV 30H,#1
0003| MOV 31H,#1
0006| MOV 32H,#1
0009| MOV 33H,#0
000C| MOV 34H,#1
000F| MOV 35H,#1
0012| MOV 36H,#1
0015| MOV 37H,#0
0018| MOV 38H,#0
001B| MOV 39H,#0
001E| MOV 3AH,#1
0021| MOV 3BH,#0
0024| MOV 3CH,#0
0027| MOV 3DH,#0
002A| MOV 3EH,#1
002D| MOV 3FH,#0

0030| MOV R0,#30H
0032| MOV P2,#255
```

```
P0.7 1   P0.6 1   P0.5 1   P0.4 1   P0.3 1   P0.2 1   P0.1 1   P0.0 1
P1.7 1   P1.6 1   P1.5 0   P1.4 1   P1.3 1   P1.2 1   P1.1 1   P1.0 0
P2.7 1   P2.6 1   P2.5 1   P2.4 1   P2.3 1   P2.2 1   P2.1 1   P2.0 1
P3.7 1   P3.6 1   P3.5 1   P3.4 1   P3.3 1   P3.2 1   P3.1 1   P3.0 1
```

Fig.1: The LEDs tell us that the functional output at 10 (1010_2) is 1, which we can see by directly inspecting the memory byte 3A.



```
System Clock (MHz) 12.0          100  Update Freq.
SBUF
R/O   W/O     TH0   TL0     R7 0x00      B 0x00
0x00  0x00    0x00  0x00    R6 0x01    ACC 0xF2
                            R5 0x01    PSW 0xC1
RXD   TXD     TMOD  0x00    R4 0x15     IP 0x00
 1     1      TCON  0x00    R3 0x00     IE 0x00
SCON  0x00                  R2 0x00   PCON 0x00
pins  bits    TH1   TL1     R1 0x00    DPH 0x00
0xFF  0xFF P3 0x00  0x00    R0 0x30    DPL 0x00
0xFE  0xFF P2          8051             SP 0x09
0x0E  0x0E P1  PC
0xFF  0xFF P0  0x0061   PSW 1 1 0 0  0 0 0 1
                                Modify RAM
Data Memory               addr 0x00  0x00 value

    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 30 00 00 00 15 01 01 00 54 00 00 00 00 00 00 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 01 01 01 00 01 01 01 00 00 00 01 00 00 00 01 00
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Copyright ©2005-2021 James Rogers
```

```
Time: 9ms 73us - Instructions: 5100

        ; Author: Shikhar Mohan
        ; Date: 20/09/2021
        ; EC49001: Assignment 2


0000| MOV 30H,#1
0003| MOV 31H,#1
0006| MOV 32H,#1
0009| MOV 33H,#0
000C| MOV 34H,#1
000F| MOV 35H,#1
0012| MOV 36H,#1
0015| MOV 37H,#0
0018| MOV 38H,#0
001B| MOV 39H,#0
001E| MOV 3AH,#1
0021| MOV 3BH,#0
0024| MOV 3CH,#0
0027| MOV 3DH,#0
002A| MOV 3EH,#1
002D| MOV 3FH,#0

0030| MOV R0,#30H
0032| MOV P2,#255
```

```
P0.7 1   P1.7 0   P2.7 1   P3.7 1
P0.6 1   P1.6 0   P2.6 1   P3.6 1
P0.5 1   P1.5 0   P2.5 1   P3.5 1
P0.4 1   P1.4 0   P2.4 1   P3.4 1
P0.3 1   P1.3 1   P2.3 1   P3.3 1
P0.2 1   P1.2 1   P2.2 1   P3.2 1
P0.1 1   P1.1 1   P2.1 1   P3.1 1
P0.0 1   P1.0 0   P2.0 0   P3.0 1
```

Fig. 2: Here we can see that the P2.0 switch is pressed and all LEDs are high.

Results

1. We observe that the 4-bit counter implementation is working fine and the required output is observed on the LEDs while traversing through the truth table.

2. When the switch is pressed all LEDs blink until it is released, upon which the LEDs start at zero.

Discussion

1. Creating the LED array inside our accumulator makes the counter very simple to implement because it uses the fact that `ACC` and `P1` are both 8-bit values and saves us the trouble of obtaining bits manually. Directly adding the RAM value means the LED stays low when it's zero and is high when the value is one. One needs to keep in mind that P1 is active low, so the complement needs to be taken in operations like these prior to moving. It can also be observed that random segments of the 7-segment panel are glowing as well. This is because the LED panel is connected to the 7-segment display panel, which is active low as well (i.e. common cathode).
2. Instead of using a time based delay, we implement a simple delay module subroutine where a memory operation is carried out 10 times with a loop, and this loop is nested twice so that the operation is carried out a 1000 times. This provides sufficient reliable delay.

Summary

We utilise an efficient algorithm to create the LED output in our `ACC` itself, and add an `EXCEPTION` subroutine to blink our LEDs based on the input of a switch `sw0`.