Problem 8

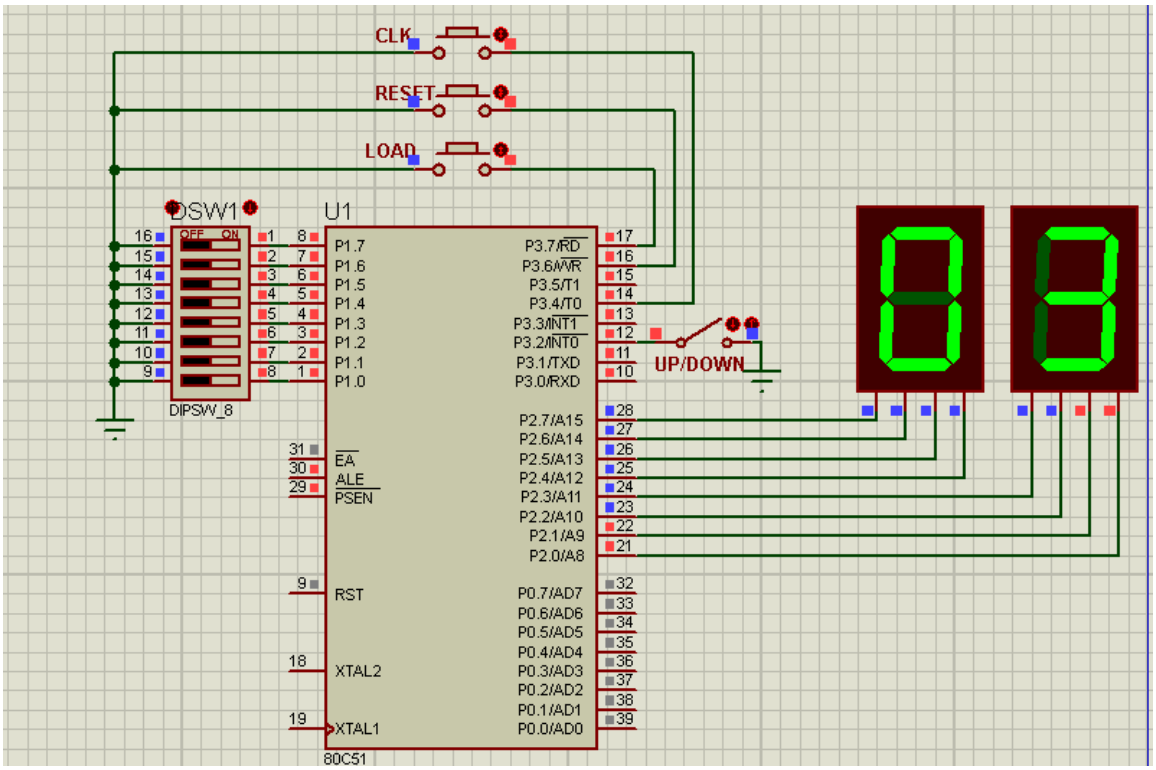
In this problem we implement an up/down 8 bits counter with reset and load

Pressing the reset button clears the counter to 0

Pressing the load button will put the value given by the dip switches into counter

Pressing the clk button will increase the count by if the up/down switch is open else it will decrease it by 1

The hex 7 segments shows the counter content



Section 1 initialization

```
2  DISPLAY EQU P2
3  COUNTER EQU 31H
4  UP EQU P3.2
5  CLK EQU P3.4
6  RST EQU P3.6
7  LOAD EQU P3.7
```

Pins used for up, clk, RST, LOAD, and an internal variable "COUNTER" to hold the counter value.

Section 2 code

```
22  Start:
23    MOV COUNTER,#0
24    MOV DISPLAY,COUNTER
25  Loop:
26    JNB LOAD,LOAD_COUNTER
27    JNB RST, RESET_COUNTER
28    JNB CLK,UPDATE_COUNTER
29    JMP LOOP
30
31  LOAD_COUNTER:
32     MOV COUNTER,P1
33     MOV DISPLAY,COUNTER
34     JMP LOOP
35  RESET_COUNTER:
36     MOV COUNTER,#0
37     MOV DISPLAY,COUNTER
38     JMP LOOP
```

We start with 0 as initial value and send it to the display (lines 23,24)

In the loop, we test for three buttons and jump to the corresponding code for each button. (note that the buttons are active low)

Loading the counter is done by simply moving the value of port1 to the counter (line 32)

Resetting the counter is done simply by putting 0 in it (line 36)

```
39  UPDATE_COUNTER:
40     JB UP,UP_COUNT
41     DEC COUNTER
42     JMP DISP_COUNT
43  UP_COUNT:
44     INC COUNTER
45  DISP_COUNT:
46     MOV DISPLAY,COUNTER
47  WAIT_CLK:
48     JNB CLK,WAIT_CLK
49     JMP LOOP
```

Pressing the clk button will update the counter content, but first we test for the up/down switch to increment/decrement the counter (line 40)

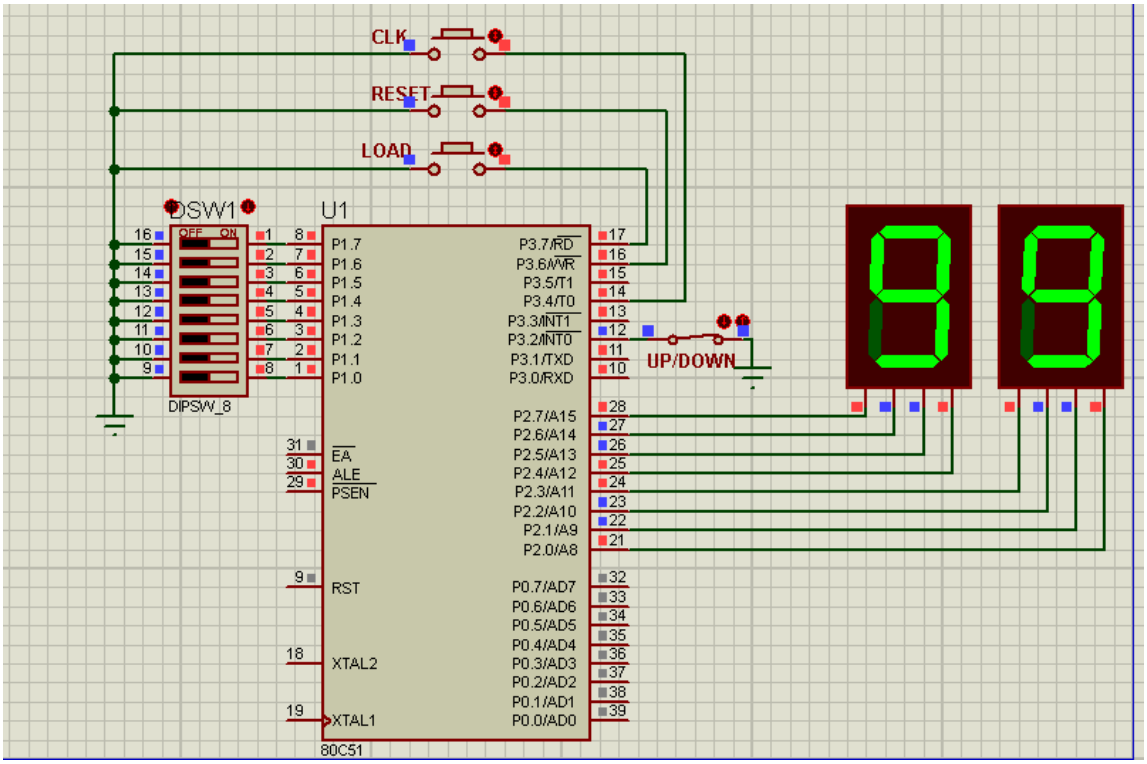
Line 41 decrement the counter content

Line 44 increment the counter content

Finally we must wait for the clock line to return to its idle state "1" at line 48

Note: we use –ve edge triggered clock

Version2 of the problem implements a BCD counter



It is the same structure as the binary counter except the update section will increment/decrement in BCD format

Definitions:

```
21  DISPLAY EQU P2
22  COUNTER EQU 31H
23
24  DIG0 EQU 32H
25  DIG1 EQU 33H
26
27  UP EQU P3.2
28  CLK EQU P3.4
29  RST EQU P3.6
30  LOAD EQU P3.7
```

The same as before but with the addition of two variable DIG0 and DIG1 to implement the BCD increment/decrement algorithm

```
45  Start:
46    MOV DIG0,#0
47    MOV DIG1,#0
48    CALL DISP
49  Loop:
50    JNB LOAD,LOAD_COUNTER
51    JNB RST, RESET_COUNTER
52    JNB CLK,UPDATE_COUNTER
53    JMP LOOP
```

Initially we reset the values of the two digits (lines 46,47)

Lines 50 to 52 is the same as previous

```
55  LOAD_COUNTER:
56     MOV A,P1
57     ANL A,#00001111B
58     MOV DIG0,A
59     MOV B,#10
60     CLR C
61     SUBB A,B
62     JC U1
63     MOV DIG0,#9
64  U1:
65     MOV A,P1
66     SWAP A
67     ANL A,#00001111B
68     MOV DIG1,A
69     MOV B,#10
70     CLR C
71     SUBB A,B
72     JC U2
73     MOV DIG1,#9
74  U2:
75     CALL DISP
```

The loading of the counter is done in a two steps.

First we get the value of P1 (dip switches) into ACC, then we isolate the first digit by the logical AND with "00001111" → resetting the upper digit (line 57)

- ➔ Here is a check to see if the input value is greater than 9 (not BCD format),we force it to any valid BCD value (in our code, we choose 9)
- ➔ This is done by subtracting 10 from the input value (line 60,61), then if a carry =1 → this means that the value is less than 10 (valid BCD) else, we force it to 9 (line 63)

2nd same procedure applied to DIG1 in lines 65 to 73 (note that we swap nibbles of ACC to put the 2nd digit in low order nibbles.

```
83  UPDATE_COUNTER:
84     JB UP,UP_COUNT
85     MOV R0,#DIG0
86     DEC @R0
87     CJNE @R0,#255,DISP_COUNT
88     MOV @R0,#9
89     INC R0
90     DEC @R0
91     CJNE @R0,#255,DISP_COUNT
92     MOV @R0,#9
93     JMP DISP_COUNT
94
95  UP_COUNT:
96     MOV R0,#DIG0
97     INC @R0
98     CJNE @R0,#10,DISP_COUNT
99     MOV @R0,#0
100    INC R0
101    INC @R0
102    CJNE @R0,#10,DISP_COUNT
103    MOV @R0,#0
```

Updating the counter will also done in two steps

After first checking up/down switch we either increment or decrement the count.

To increment the count (lines 95 to 103)

- ➔ We use R0 as a pointer to DIG0
- ➔ Increment the digit using indirect mode with R0 (line 97)
- ➔ If the value is not valid BCD (=10) as in line 98, we reset the first digit to 0 (line 99) and increment the next digit and adjust it if it is not a valid BCD(lines 102,103)

The decrement procedure (85-92)

- ➔ We decrement DIG0 (line 86)
- ➔ (87,88)If DIG0 rolls back to 255 (note that if DIG0 = 0 and we decrement it, it will become =255), we reset it to 9 and
- ➔ Decrement DIG1 and adjust it if the becomes 255 (lines 91-92)

Display subroutine

```
111 DISP:
112   MOV R0,#DIG1
113   MOV A,@R0
114   SWAP A
115   DEC R0
116   ADD A,@R0
117   MOV DISPLAY,A
118   RET
```

Again we use R0 to points to DIG1 and put it into ACC (lines 112, 113)
We move it to the higher nibble (line 114)
Decrement R0 to point to DIG0, and we add it to ACC (line 116)
Now the content of ACC → DIG1:DIG0
Finally send it to the display port (line 117)