Problem 9_1

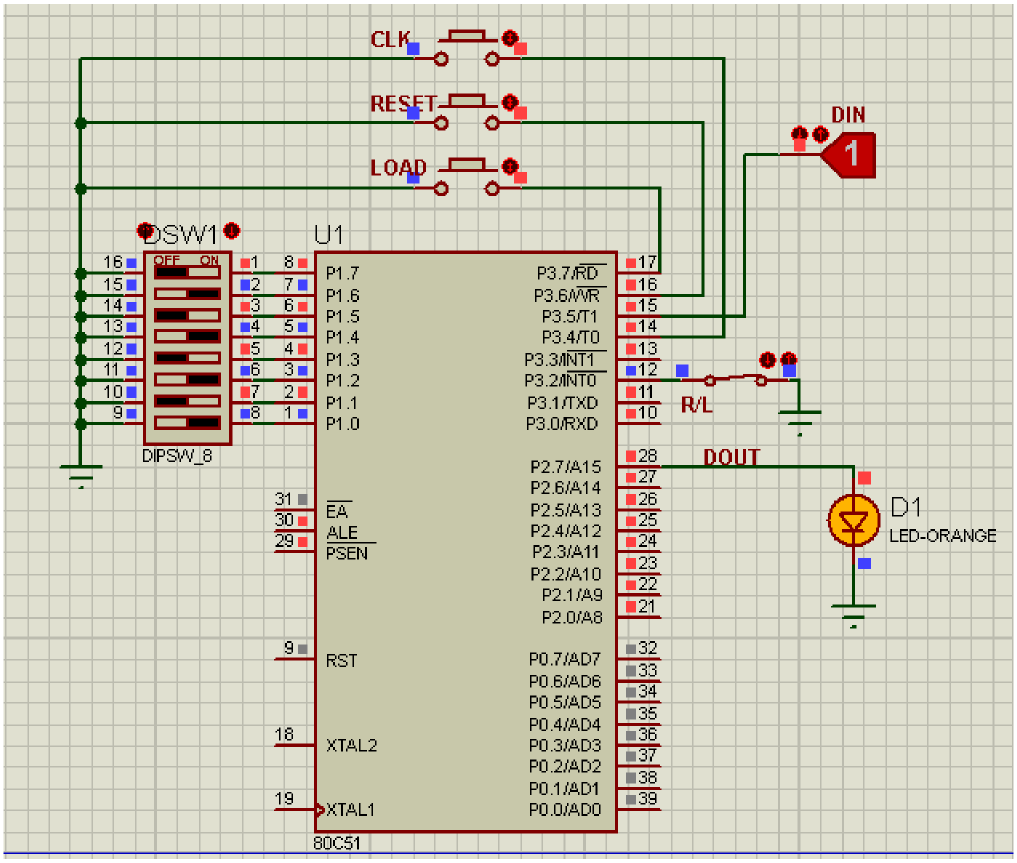
In this problem we implement parallel in serial out shift register

Pressing the reset button clears the register to 0

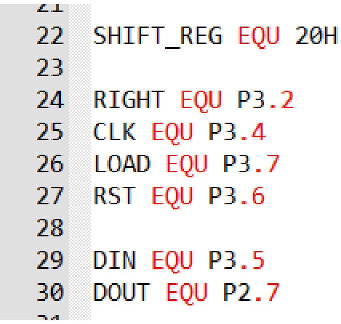
Pressing the load button will put the value given by the dip switches into register

Pressing the clk button will shift right if the R/L switch is open else it will shift left

The register will have DOUT (serial output)



Section 1 initialization



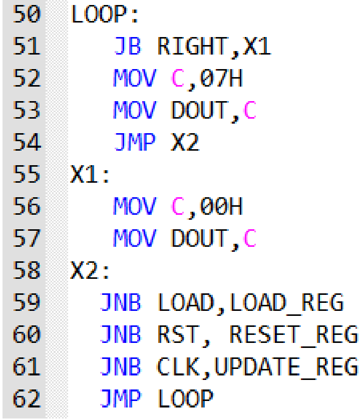
```
SHIFT_REG EQU 20H

RIGHT EQU P3.2
CLK EQU P3.4
LOAD EQU P3.7
RST EQU P3.6

DIN EQU P3.5
DOUT EQU P2.7
```

Pins used for RIGHT, CLK, RST, LOAD, and an internal variable "SHIFT_REG" to hold the REGISTER value.

Section 2 code



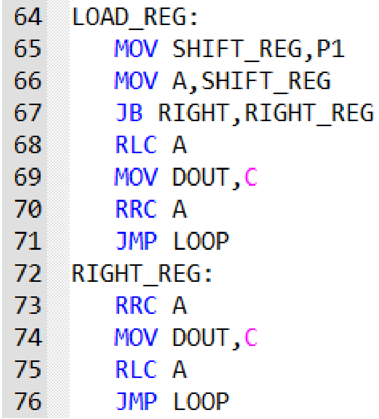
```
50 LOOP:
51      JB RIGHT,X1
52      MOV C,07H
53      MOV DOUT,C
54      JMP X2
55 X1:
56      MOV C,00H
57      MOV DOUT,C
58 X2:
59    JNB LOAD,LOAD_REG
60    JNB RST, RESET_REG
61    JNB CLK,UPDATE_REG
62    JMP LOOP
```

First, we check the shift direction (line 51), if it is left→ we output MSB to DOUT (52,53) else we output LSB to DOUT (lines 56,57)

Important:

- ➔ We put the SHIFT_REG in the first byte of bit-addressable memory area "20H" (line 22)
- ➔ The byte in location 20H can be accessed bit by bit
- ➔ Bit with address 0 → LSB of byte at location 20H (SHIFT_REG)
- ➔ Bit with address 8 → MSB

Lines 59 to 61 determines the required operation to be done.



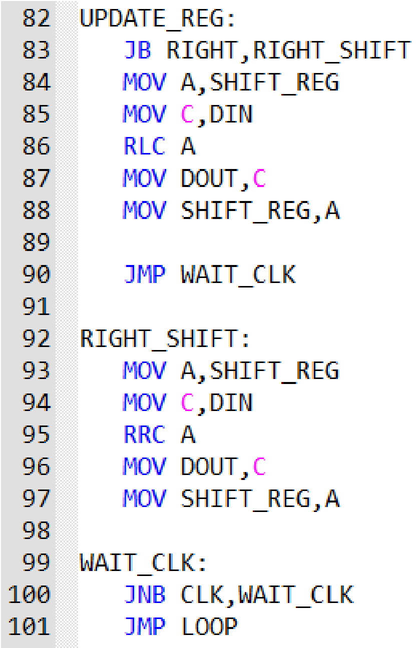
```
64 LOAD_REG:
65      MOV SHIFT_REG,P1
66      MOV A,SHIFT_REG
67      JB RIGHT,RIGHT_REG
68      RLC A
69      MOV DOUT,C
70      RRC A
71      JMP LOOP
72 RIGHT_REG:
73      RRC A
74      MOV DOUT,C
75      RLC A
76      JMP LOOP
```

Loading the register with dip-switches value

Simply we put the value of P1 into SHIFT_REG, but we need to determine which bit will be sent to the DOUT according to RIGHT/LEFT shift (67)

- ➔ For left shift, we rotate ACC left with carry (68), so carry will contain the MSB, then send it to DOUT (69)
- ➔ For right shift, we make the same procedure but with rotate right (73 to 75)

Update register



```
 82 UPDATE_REG:
 83     JB RIGHT,RIGHT_SHIFT
 84     MOV A,SHIFT_REG
 85     MOV C,DIN
 86     RLC A
 87     MOV DOUT,C
 88     MOV SHIFT_REG,A
 89 
 90     JMP WAIT_CLK
 91 
 92 RIGHT_SHIFT:
 93     MOV A,SHIFT_REG
 94     MOV C,DIN
 95     RRC A
 96     MOV DOUT,C
 97     MOV SHIFT_REG,A
 98 
 99 WAIT_CLK:
100     JNB CLK,WAIT_CLK
101     JMP LOOP
```

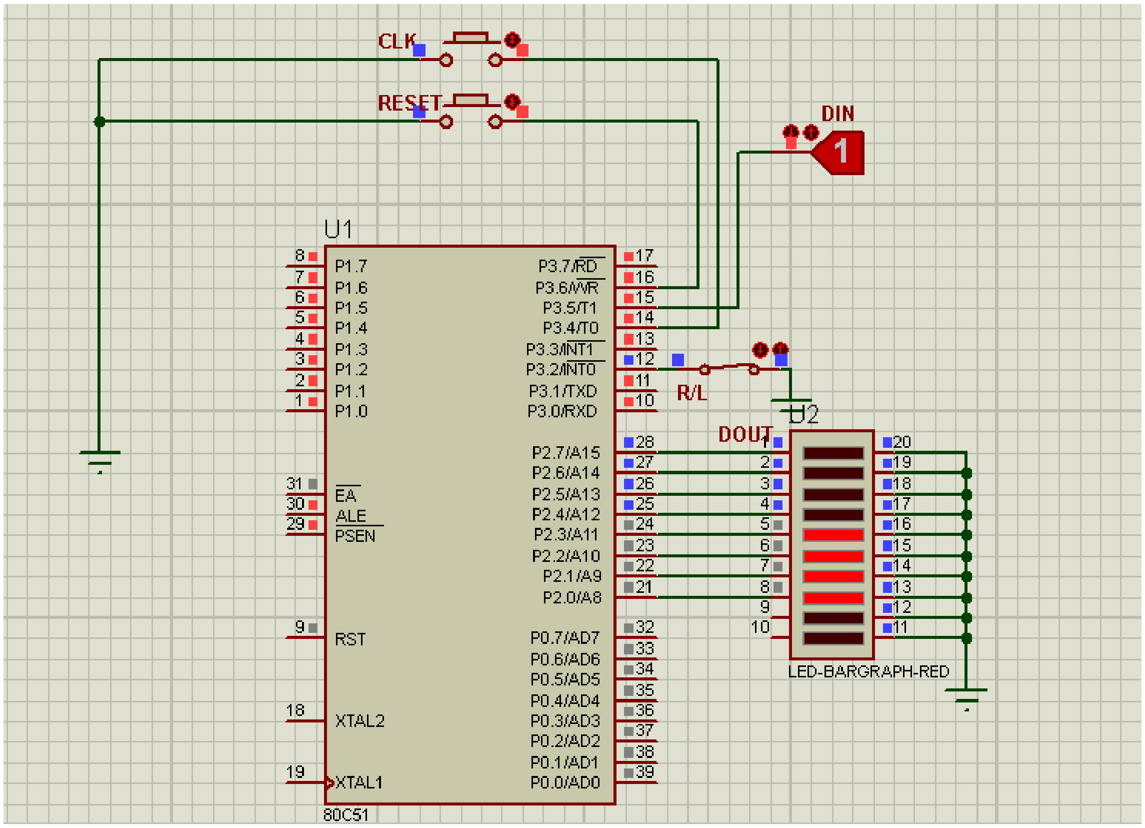
In update register → either right shift or left shift according to the state of the RIGHT/LEFT switch

For the left shift (84-88), Dout←SHIFT_REG←Din, so we put Din into C (85), then rotate left with carry (86). Now C has the MSB, so we send it to DOUT (87)

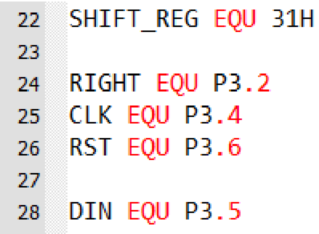
The same procedure for RIGHT SHIFT (93-97) but with rotate right.

Finally, we must wait for the clock to return to 1.

Version2 of the problem implements a Serial in Parallel out shift register



This is simpler than previous register, where it has serial line in "DIN" which will be shifted in to 8 bits register. The contents of the register is displayed with port2

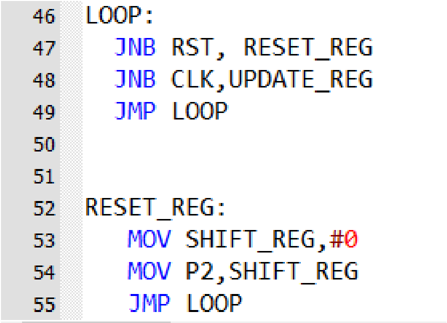


```
SHIFT_REG EQU 31H

RIGHT EQU P3.2
CLK EQU P3.4
RST EQU P3.6

DIN EQU P3.5
```

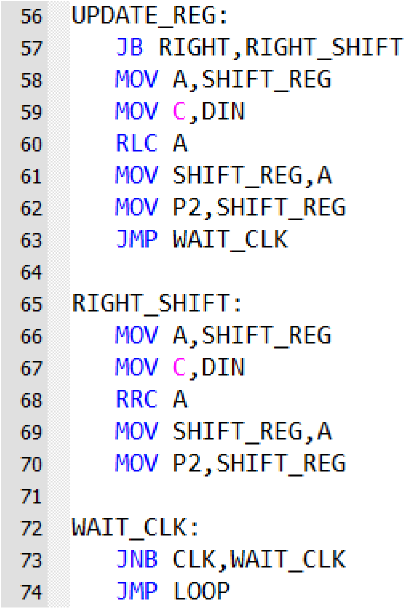
We need only clk to shift, reset to clear the register, and R/L to determine the shift direction.



```
46  LOOP:
47    JNB RST, RESET_REG
48    JNB CLK,UPDATE_REG
49    JMP LOOP
50
51
52  RESET_REG:
53     MOV SHIFT_REG,#0
54     MOV P2,SHIFT_REG
55     JMP LOOP
```

47-48 test for the required operation either reset or shift

In reset (53-54) we clear the SHIFT_REG



```
56  UPDATE_REG:
57     JB RIGHT,RIGHT_SHIFT
58     MOV A,SHIFT_REG
59     MOV C,DIN
60     RLC A
61     MOV SHIFT_REG,A
62     MOV P2,SHIFT_REG
63     JMP WAIT_CLK
64
65  RIGHT_SHIFT:
66     MOV A,SHIFT_REG
67     MOV C,DIN
68     RRC A
69     MOV SHIFT_REG,A
70     MOV P2,SHIFT_REG
71
72  WAIT_CLK:
73     JNB CLK,WAIT_CLK
74     JMP LOOP
```

For the update we first determine the required direction (57)

For SHIFT LEFT (58-63), SHIFT_REG←DIN "C", so we make an RLC A after initializing c by DIN (59)

The same procedure for SHIFT RIGHT(66-70) except that we use RRC A