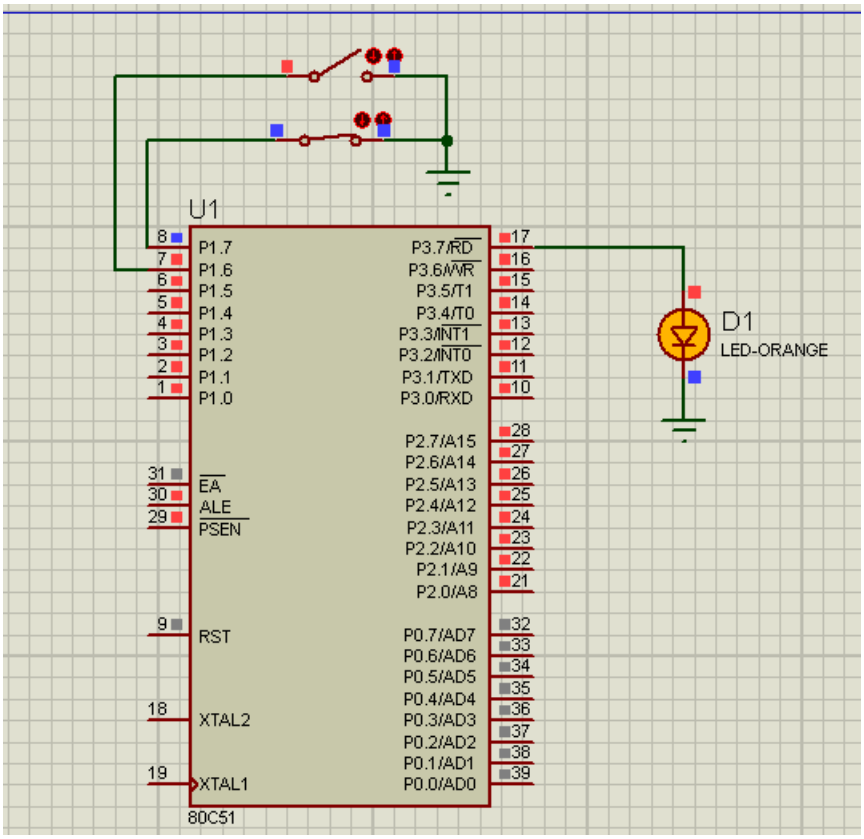
Problem 17



This project uses Timer1 as in problem 16, where it will interrupt the processor every 50ms, and uses this time interval to turn ON a LED for T_ON and OFF for T_OFF.
The difference here is that, the LED will blink only if input X1 = 1 and X2 = 0;

Definitions

```
19  LED1_ON_TIME EQU 40
20  LED1_OFF_TIME EQU 20
21
22  COUNT_ON1 EQU 30H
23  COUNT_OFF1 EQU 31H
24
25  LED1 EQU P3.7
26  X1 equ P1.6
27  X2 equ P1.7
28
29  LED1_STATE EQU 00H
```

19 to 20, defines the ON/OFF values for LED (multiple of 50ms)

LED ON Time of 40 → 40*50ms = 2 seconds
LED OFF Time of 20 → 20 *50 = 1 second

We need to define a counter for each TIME_ON and TIME_OFF (lines 22-23)

(25) Port pin connected to LED

We keep the status of the LED in a boolean variable as in line 29

Main loop

```
38        org   0000h
39        jmp   Start
40  ORG 001BH
41     LJMP ISR_T1
42  ;=====================================
43  ; CODE SEGMENT
44  ;=====================================
45
46        org   0100h
47  Start:
48
49     MOV TMOD,#10H
50     MOV TL1,#0B0H
51     MOV TH1,#03CH
52     SETB EA
53     ;SETB ET1
54     ;SETB TR1
55     CLR LED1
56
57  START2:
```

First we jump to start label (47), where the interrupt vector of Timer1 overflow will be at address 001Bh.

When Timer1 overflow, the 8051 will be interrupted and execute the code at address 001Bh. So we put a jump to the starting code for our timer handler (77)

At 49, we set Timer 1 in mode 1
At 50-51, we initialize the Timer by 3CB0h, which will cause an overflow after counting 50000 clock cycle, and since the clock of the counter = 12M/12 = 1us→ the time passed will be 50ms

In this problem we don't enable/run Timer1 now – but wait for the input condition to be valid

Testing input X1, X2

```
57  START2:
58     CLR LED1
59     MOV C,X1
60     ANL C,/X2
61     JNC START2
62     SETB LED1_STATE
63     SETB ET1
64     SETB TR1
65     MOV COUNT_ON1,#LED1_ON_TIME
66     MOV COUNT_OFF1,#LED1_OFF_TIME
67
68  WAIT1:
69     MOV C,X1
70     ANL C,/X2
71     JC WAIT1
72     CLR ET1
73     CLR TR1
74     JMP START2
```

First we start with LED1 off (58), read X1 into carry (59), logical AND the complement of X2 with carry (60) → so now the carry = X1 AND not X2; if no carry produced → wait till condition becomes valid (61);

If condition becomes valid → we initialize the ON and OFF times, enable Timer1 overflow interrupt, and run Timer1 (62-66)

Now the interrupt will handle the blinking of LED1 as before

Next in the main code we will recalculate the input condition again (69-70) ; and wait until the condition becomes invalid (71) → disable interrupt and timer1 then jump to start2 to wait for the condition to be valid again.

Timer1 overflow interrupt)

Timer1 Overflow Interrupt

```
77  ISR_T1:
78     MOV TL1,#0B0H
79     MOV TH1,#03CH
80  TEST_LED1:
81     JB LED1_STATE,LED1_ON
82     CLR LED1
83     DJNZ COUNT_OFF1,TEST_LED2
84     SETB LED1_STATE
85     MOV COUNT_OFF1,#LED1_OFF_TIME
86     JMP TEST_LED2
87  LED1_ON:
88     SETB LED1
89     DJNZ COUNT_ON1,TEST_LED2
90     CLR LED1_STATE
91     MOV COUNT_ON1,#LED1_ON_TIME
92  TEST_LED2:
93
94
95  RETI
```

It is the same as problem 16 but with only one LED