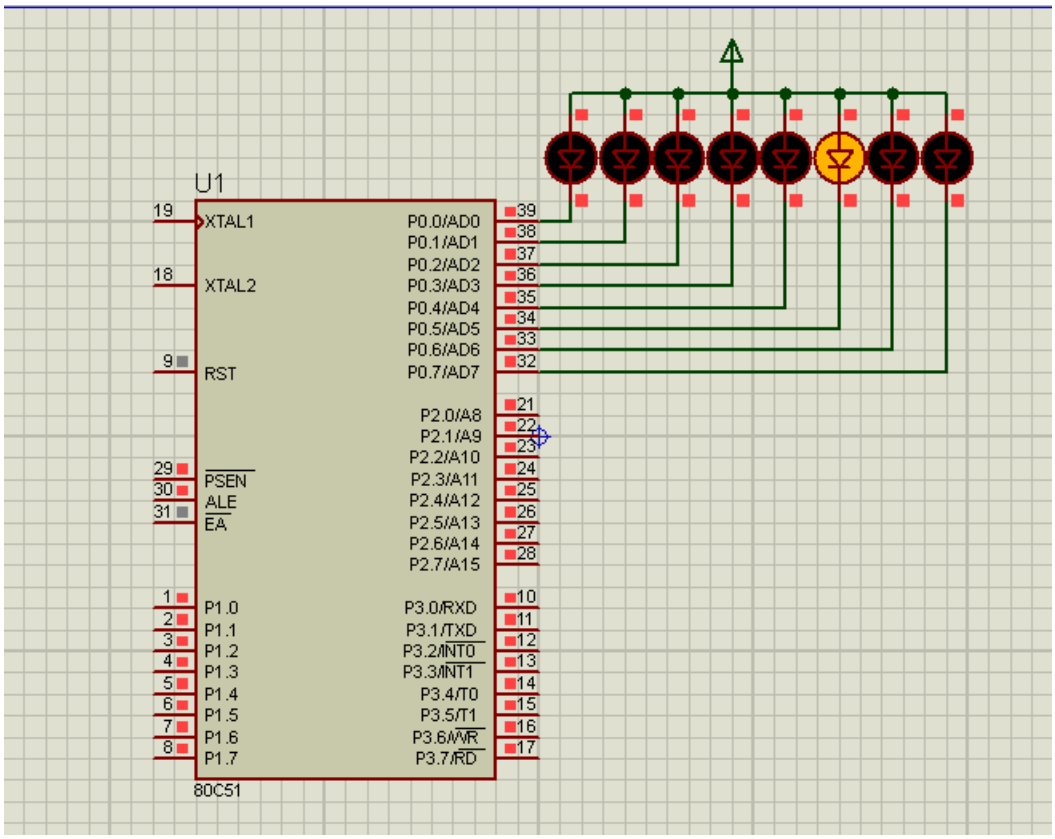
Problem 7

It is the same as problem 6 except that we will use 8051's internal timers to generate the time delay



Section 3 Subroutine

```
42  DELAY_50MS:
43    MOV TMOD,#01H  ; Timer0 mode1
44    MOV TH0, #3CH ;  High byte initial value for 10ms
45    MOV TL0, #0B0H  ; // Low byte initial value for 10ms
46    SETB TR0      ; // timer0 start
47  WAIT_OVERFLOW:
48    JNB TF0,WAIT_OVERFLOW
49    CLR TR0       ; // Stop Timer
50    CLR TF0       ; // Clear flag
51  RET
```

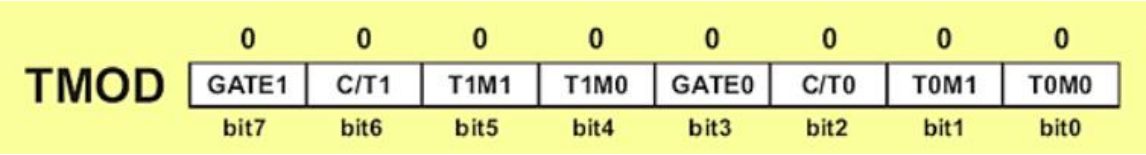
Before explaining the code, we must explain first the internal timers of 8051.

The 8051 has 2 internal counters. These counters can count from external microcontroller's pin and in this case it called a counter. Or it can count from a clock driven from the main clock (Clock/12) and in this case it is called a Timer.

For delay generation, we use these counters in "timer mode" where it will counts

the clock cycles until it overflows.
For example; if we have a 12Mhz main clock, then the input frequency to these timers will be 12Mhz/12 = 1Mhz. Or we can call that these counters will increment its value every 1us. If we start from 0 value and after 100us, the timer will have a value of 100 indicating that 100us had elapsed.

8051 timers have 3 modes that can be set by a register called "TMOD"



- **GATE0** enables and disables Timer 1 using a signal brought to the INT0 pin (P3.2):
  - **1** - Timer 0 operates only if the INT0 bit is set.
  - **0** - Timer 0 operates regardless of the logic state of the INT0 bit.
- **C/T0** selects pulses to be counted up by the timer/counter 0:
  - **1** - Timer counts pulses brought to the T0 pin (P3.4).
  - **0** - Timer counts pulses from internal oscillator.
- **T0M1,T0M0** These two bits select the oprtaional mode of the Timer 0.

T1M1	T1M0	MODE	DESCRIPTION
0	0	0	13-bit timer
0	1	1	16-bit timer
1	0	2	8-bit auto-reload
1	1	3	Split mode

The first 4 bits are for Timer 0, and the next 4 bits are for Timer 1.

In our example, we will use Timer 0

We will use Timer 0 as a 16-bit timer → Mode 1

So we must set TMOD register with TIM1:TIM0 = 01 for Timer 0 section

Also to work as a timer we must make $C/\overline{T}\,bit = 0$

The Gate bit is disabled "=0" to enable Timer 0

So the value of TMOD = xxxx0001

"MOV TMOD,#01

To make the timer start counting, we must set bit TR0 to 1 as in line 46

Now the timer will count until it reaches it maximum value "2^{16}", then it will return to "0" and count again.

Each time the counter rolls of from its maximum value to 0 an overflow flag is set. This overflow flag is called TF0.

Now how to make a delay for 50ms = 50000 count?

- We must make the initial value of this timer to be at (max value – 50000), so overflow will occur after 50000 count = 50ms.
- Then, we will test the overflow flag TF0 and wait for it to become '1' as in line 48

So the initial value of the timer must be = $2^{16} - 5000 = 15536 = 3CB0H$

Split this value into high and low bytes and initialize both TH0 and TL0 as in lines 44,45.

Finally we stop the timer and reset TF0 as in lines 49,50

To refine this delay function, we can take into account the time consumed by all instructions in the subroutine and subtract it from the required time of 50ms.

For example; all "MOV" and "RET" instructions takes 2us- others takes 1us

So we have extra times = 4*2us + 4*1us = 12us. So we can increase the initial count of the timer by 12.