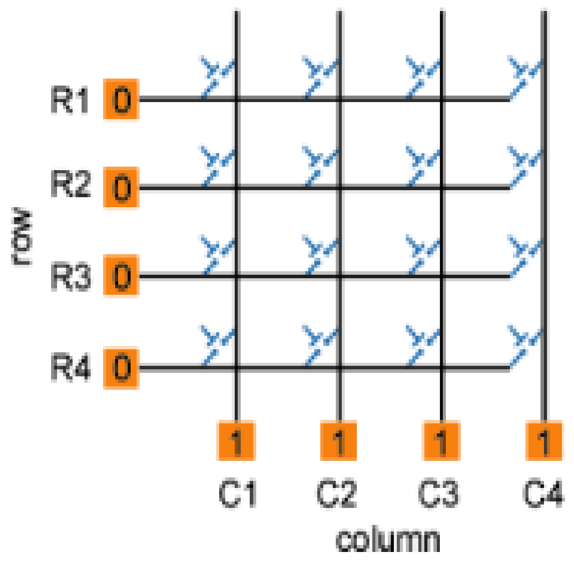
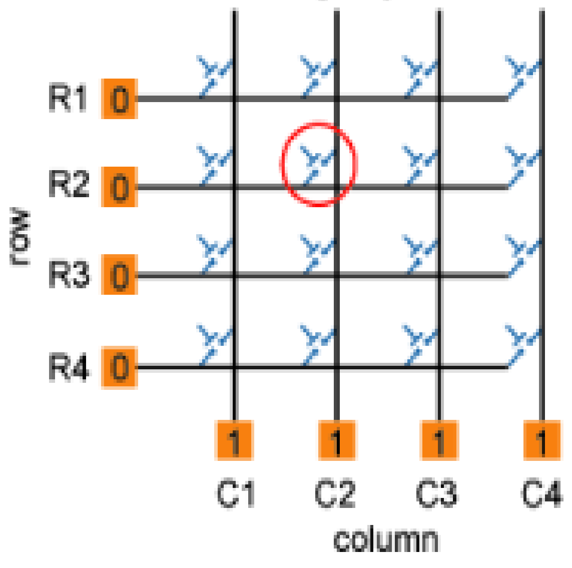
Keypad interfacing (port multiplexing)

- Keyboards are organized in a matrix of rows and columns
- When a key is pressed a row and a column make contact, otherwise there is no connection between them
- With 8-bit PORT 4x4 matrix of keys can be connected to a microcontroller.

Keypad Matrix Basics



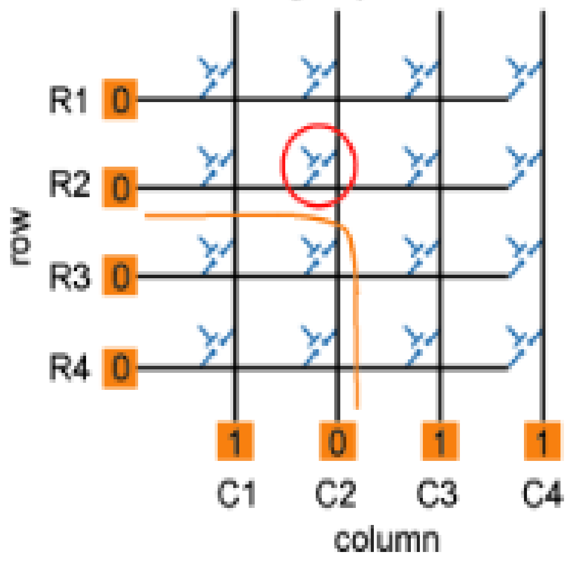
Before Pressing Key



R1 R2 R3 R4 = 0 0 0 0

C1 C2 C3 C4 = 1 1 1 1

After Pressing Key



R1 R2 R3 R4 = 0 0 0 0

C1 C2 C3 C4 = 1 0 1 1

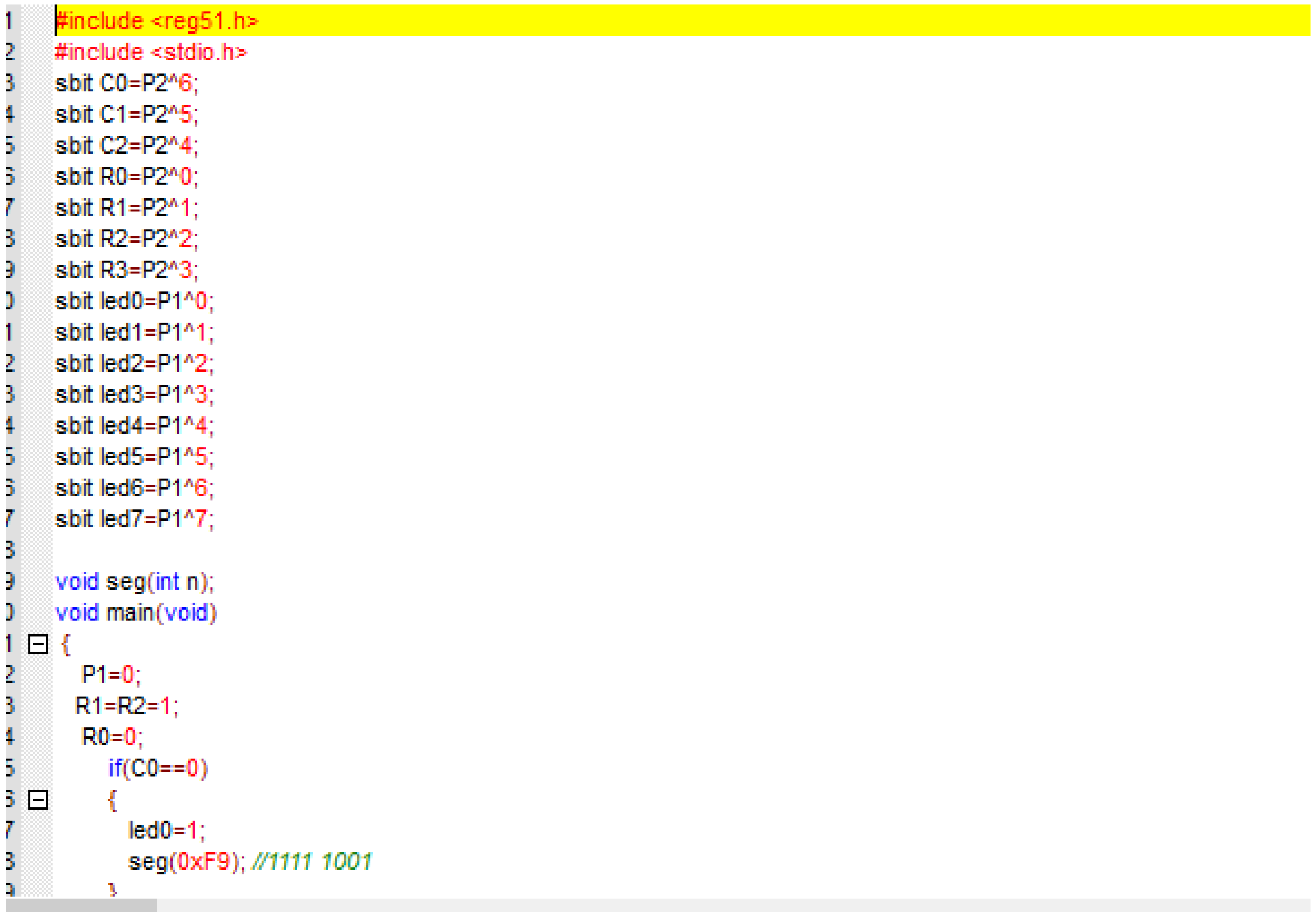
How it works ???

- The hex keypad has 8 communication lines namely
  - R1, R2, R3, and R4
  - C1, C2, C3, and C4
- R1 to R4 represents the four rows
- C1 to C4 represents the four columns
- When a particular key is pressed the corresponding row and column to which the terminals of the key are connected gets shorted.
- For example if key 1 is pressed row R1 and column C1 gets shorted and so on

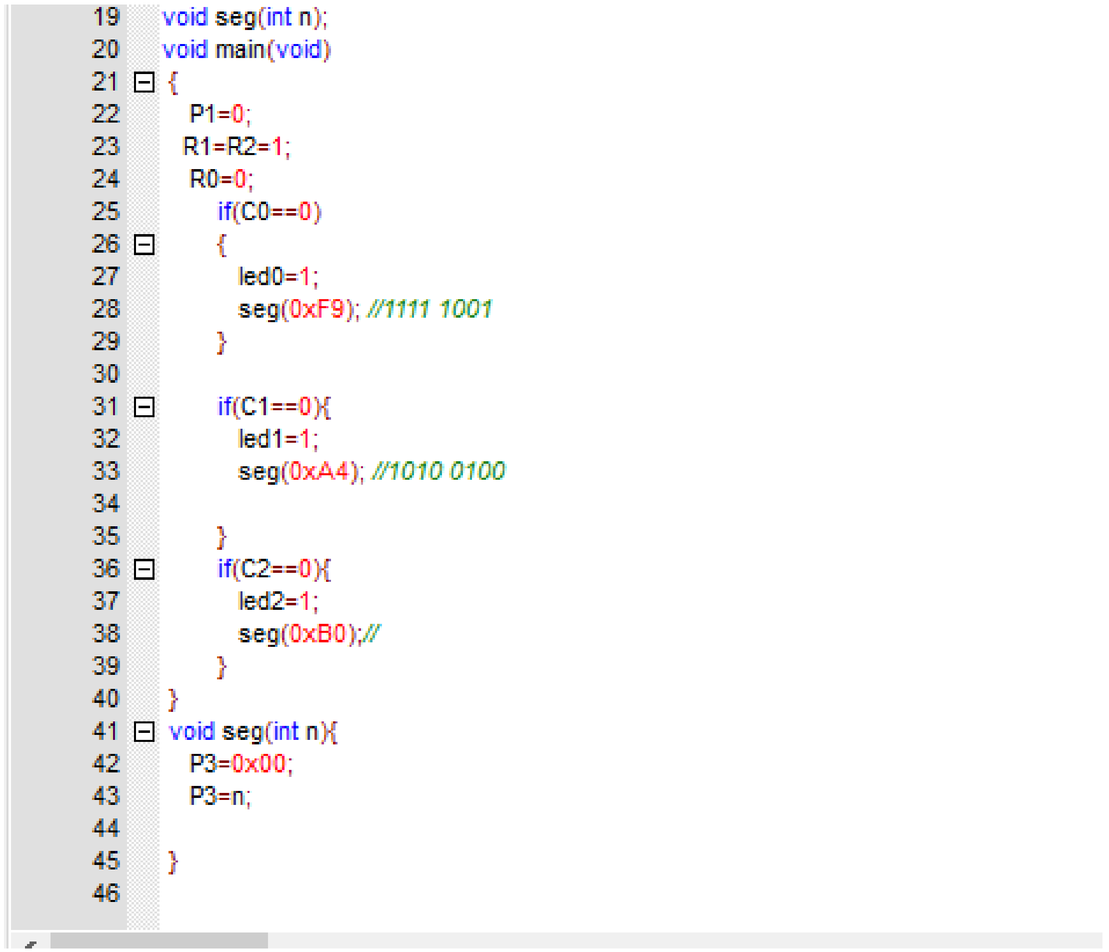
Key Scanning:

- To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, and then it reads the columns
- If the data read from columns is = 1111, no key has been pressed
- After pressing key, it makes contact of row with column
- If one of the column bits has a zero, this means that a key press has occurred.
  For example, if C1:C4 = 1011, this means that a key in the C2 column has been pressed.
- After detecting a key press, microcontroller will go through the process of identifying the key.

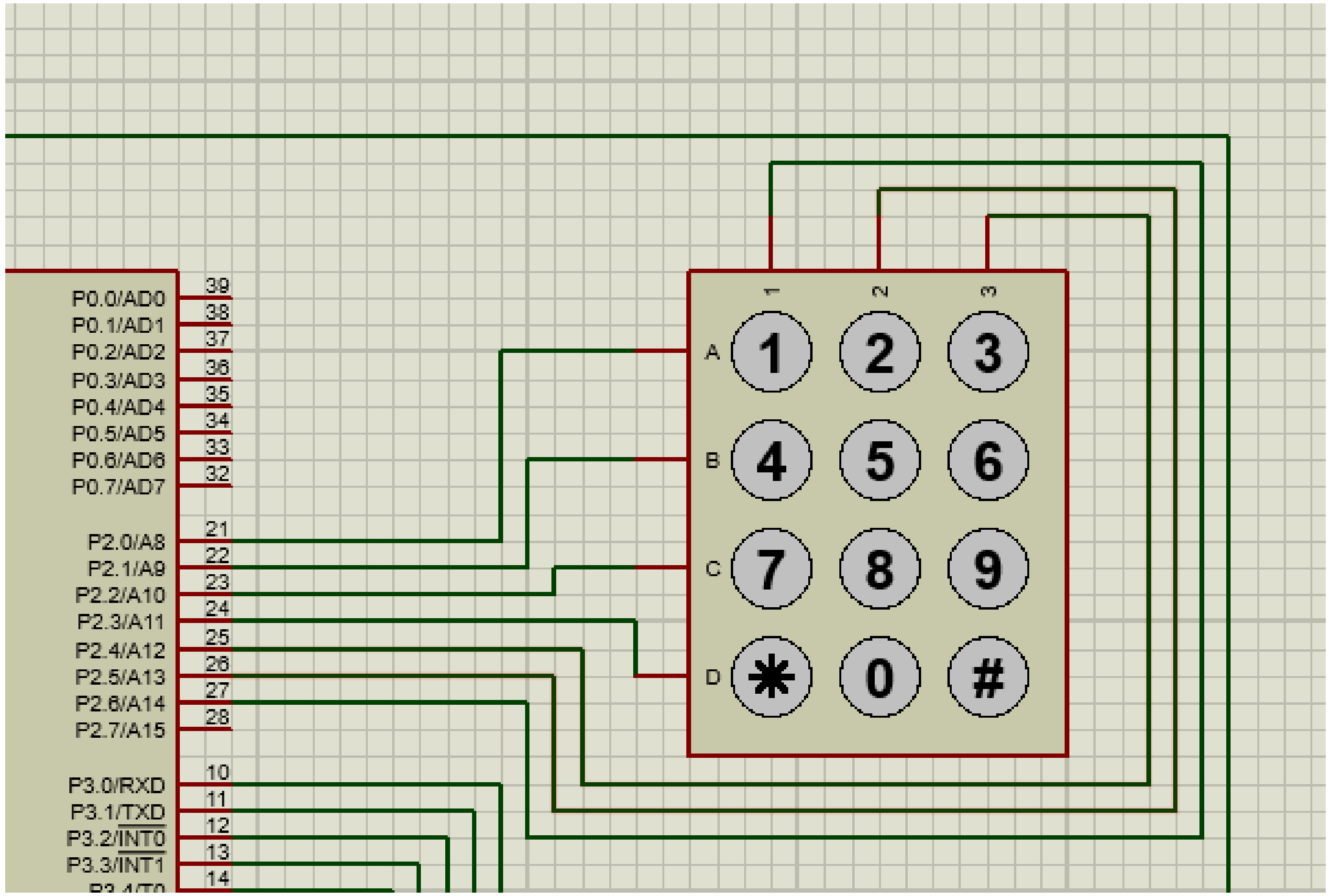
- Starting from the top row, the microcontroller will ground it by providing a low to row R1 only.
- Now read the columns, if the data read is all 1s, no key in that row is pressed and the process continues for the next row.
- So, now ground the next row, R2. Read the columns, check for any zero and this process continues until the row is identified.
  E.g. In above case we will get row 2 in which column is not equal to 1111. So, after identification of the row in which the key has been pressed we can easily find out the key by row and column value.



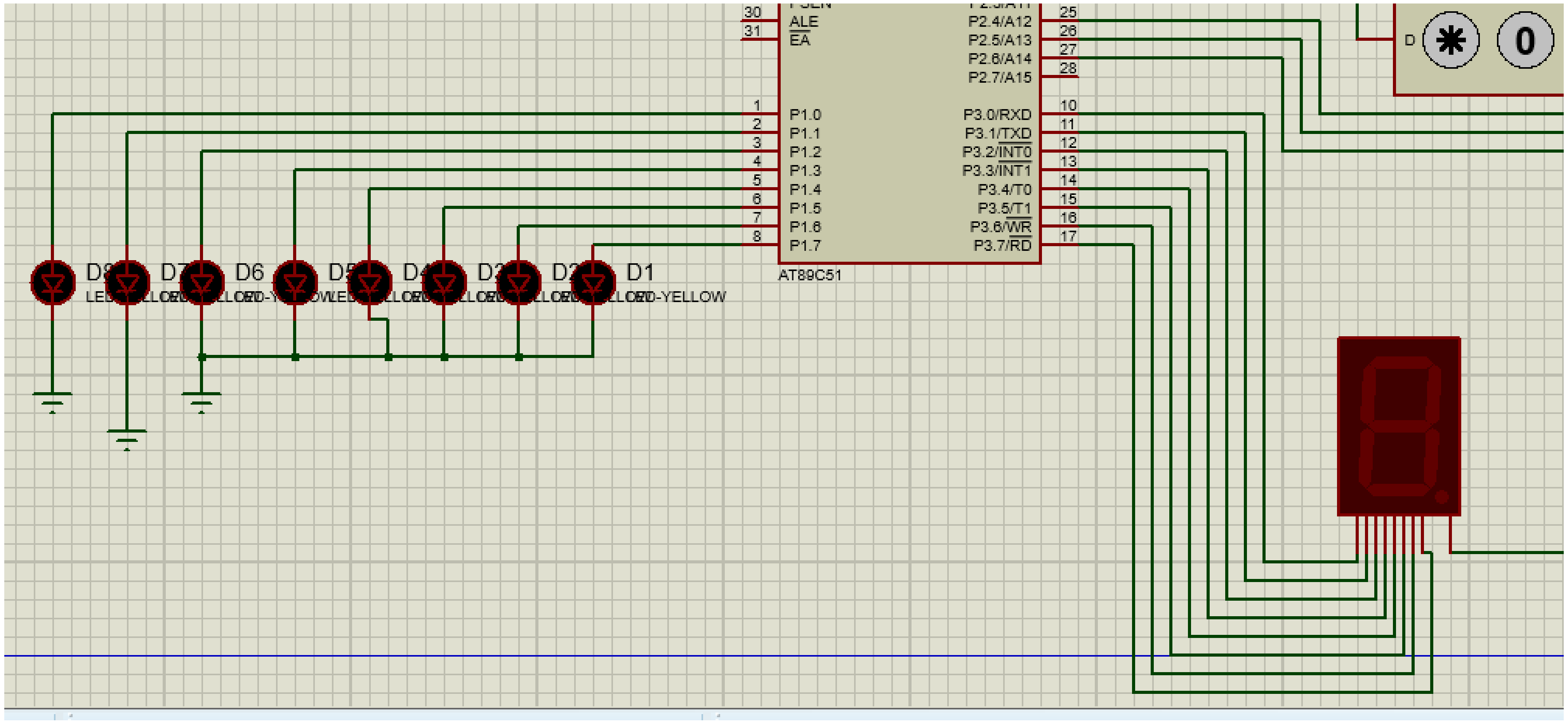
```c
#include <reg51.h>
#include <stdio.h>
sbit C0=P2^6;
sbit C1=P2^5;
sbit C2=P2^4;
sbit R0=P2^0;
sbit R1=P2^1;
sbit R2=P2^2;
sbit R3=P2^3;
sbit led0=P1^0;
sbit led1=P1^1;
sbit led2=P1^2;
sbit led3=P1^3;
sbit led4=P1^4;
sbit led5=P1^5;
sbit led6=P1^6;
sbit led7=P1^7;

void seg(int n);
void main(void)
{
   P1=0;
  R1=R2=1;
   R0=0;
       if(C0==0)
       {
          led0=1;
          seg(0xF9); //1111 1001
       }
```

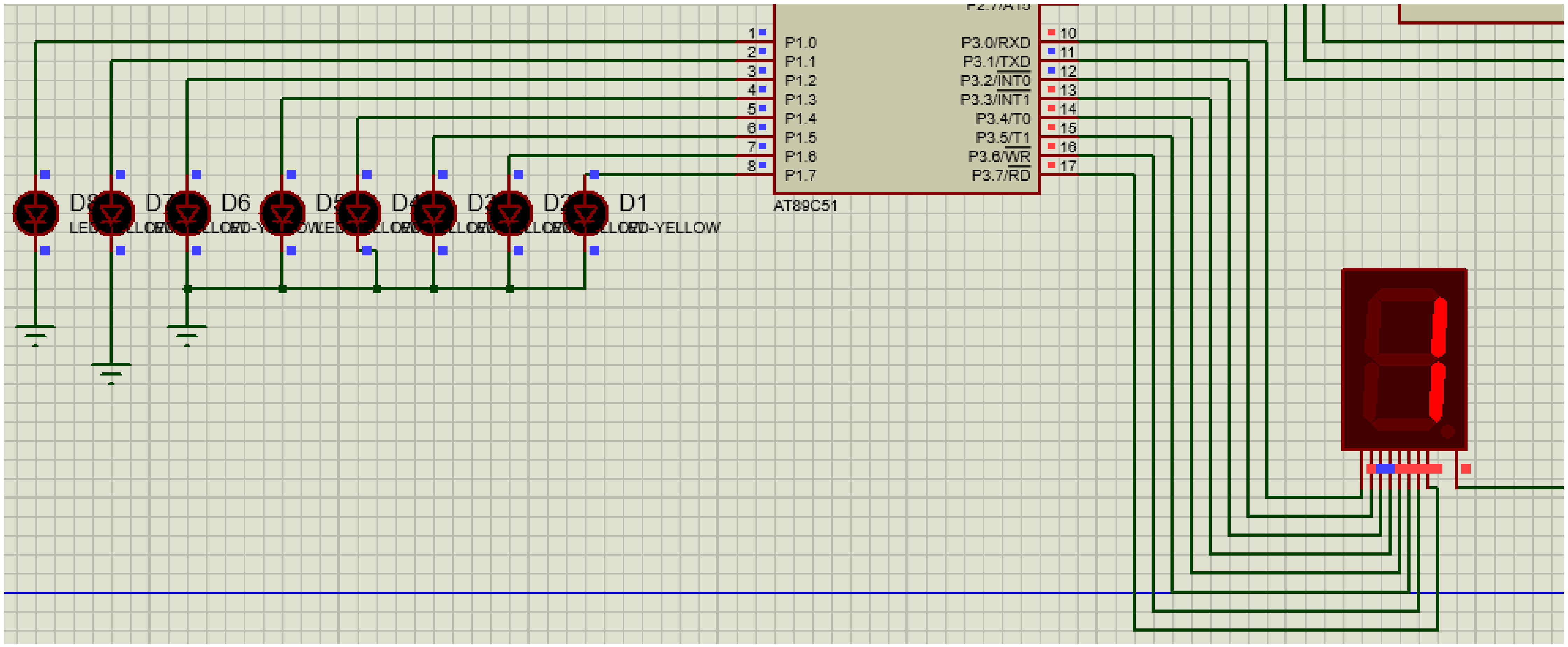


```
void seg(int n);
void main(void)
{
   P1=0;
  R1=R2=1;
   R0=0;
      if(C0==0)
      {
        led0=1;
        seg(0xF9); //1111 1001
      }

      if(C1==0){
        led1=1;
        seg(0xA4); //1010 0100

      }
      if(C2==0){
        led2=1;
        seg(0xB0);//
      }
 }
 void seg(int n){
   P3=0x00;
   P3=n;

 }

```



P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
39
38
37
36
35
34
33
32
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
21
22
23
24
25
26
27
28
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
10
11
12
13
14
A
B
C
D
1
2
3
1 2 3
4 5 6
7 8 9
* 0 #


30
ALE
31
EA
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
25
26
27
28
1 P1.0
2 P1.1
3 P1.2
4 P1.3
5 P1.4
6 P1.5
7 P1.6
8 P1.7
P3.0/RXD 10
P3.1/TXD 11
P3.2/INT0 12
P3.3/INT1 13
P3.4/T0 14
P3.5/T1 15
P3.6/WR 16
P3.7/RD 17
AT89C51
D8
D7
D6
D5
D4
D3
D2
D1
LED-YELLOW
D
0


P2.7/A15
1
2
3
4
5
6
7
8
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
10
11
12
13
14
15
16
17
AT89C51
D8
D7
D6
D5
D4
D3
D2
D1
LED-YELLOW

TASKS:

- 1) Run the program given in the lecture
- 2) Display Numbers from 1 to 9 on the seven segment display and ON the corresponding LED’s attached with P1

Note: Attach the seven segment with P3 and keypad with P1.