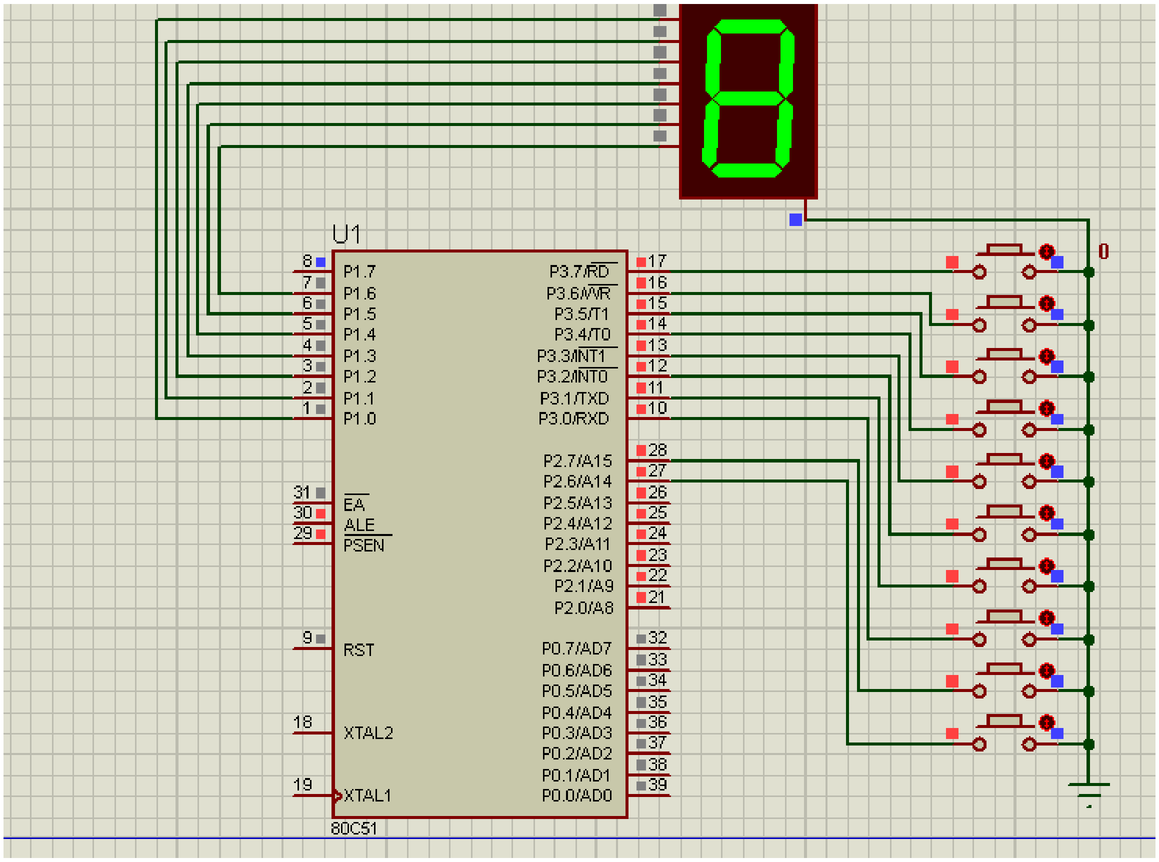
Problem 10

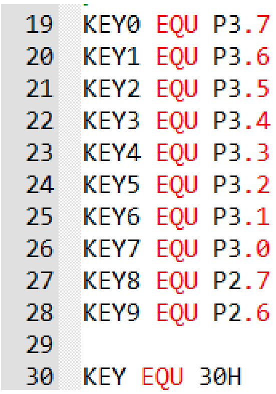
In this problem we use 0-9 keypad with one pin for each key

Keys are active low

7 segment is common cathode



Section 1 initialization

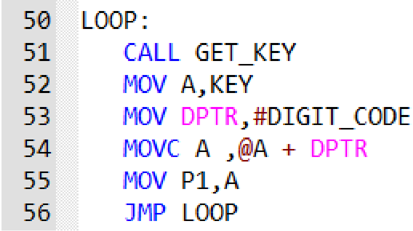

```
KEY0 EQU P3.7
KEY1 EQU P3.6
KEY2 EQU P3.5
KEY3 EQU P3.4
KEY4 EQU P3.3
KEY5 EQU P3.2
KEY6 EQU P3.1
KEY7 EQU P3.0
KEY8 EQU P2.7
KEY9 EQU P2.6

KEY EQU 30H
```

Definitions for the key buttons

One variable KEY, used to store the key to be displayed

Section 2 code

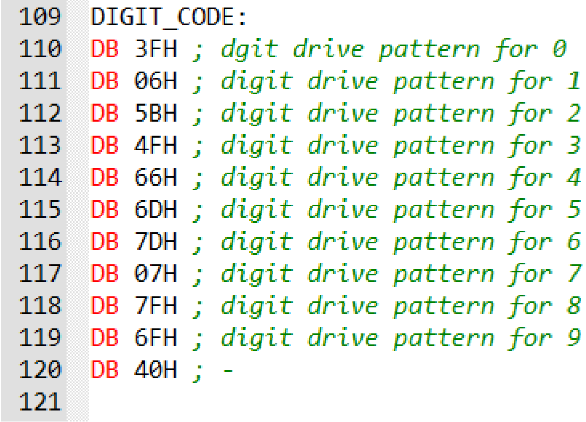


```
50 LOOP:
51     CALL GET_KEY
52     MOV A,KEY
53     MOV DPTR,#DIGIT_CODE
54     MOVC A ,@A + DPTR
55     MOV P1,A
56     JMP LOOP
```

Here is the main loop where we call the subroutine GET_KEY to read the key status and store it into variable key

Then we convert it into 7-segment code (53,54)

The conversion process depend on the following table in the code



```
DIGIT_CODE:
DB 3FH ; dgit drive pattern for 0
DB 06H ; digit drive pattern for 1
DB 5BH ; digit drive pattern for 2
DB 4FH ; digit drive pattern for 3
DB 66H ; digit drive pattern for 4
DB 6DH ; digit drive pattern for 5
DB 7DH ; digit drive pattern for 6
DB 07H ; digit drive pattern for 7
DB 7FH ; digit drive pattern for 8
DB 6FH ; digit drive pattern for 9
DB 40H ; -
```

First we make the DPTR (data pointer) points to the starting address of the table "DIGIT_CODE" (line 53).

Then we use the following command "MOVC A, @A +DPTR"

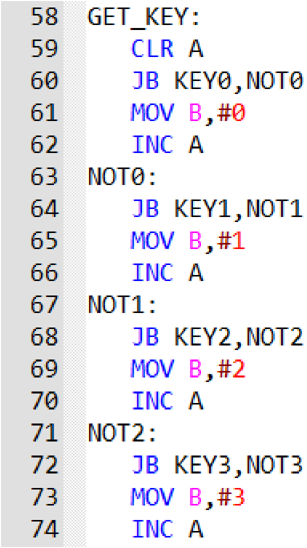
This command do the following A ← [DPTR+A]

For example if A = 2, DPTR = DIGIT_CODE, then A will be loaded by the content of location pointed to by "DIGIT_CODE + 2" or at offset of 2 bytes from "DIGIT_CODE" → this is the 7 segment code of 2

By this way, we can convert the number in A to its corresponding code from the table.

Note that at offset 10, we put the 7-segment code for the "-" which is used to indicate more than one key are pressed.

Subroutine GET_KEY

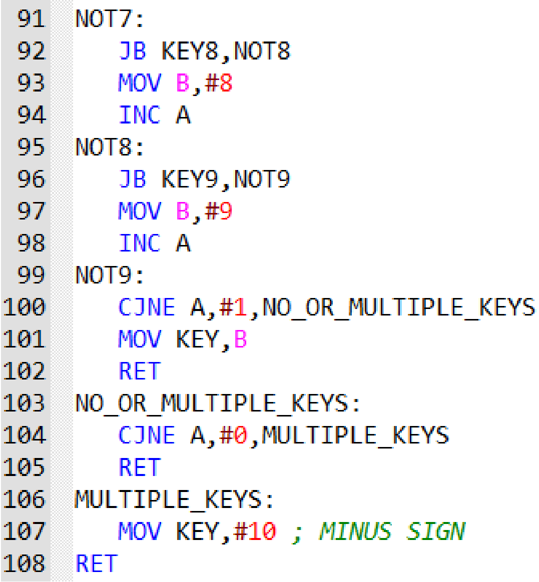


```
58  GET_KEY:
59      CLR A
60      JB KEY0,NOT0
61      MOV B,#0
62      INC A
63  NOT0:
64      JB KEY1,NOT1
65      MOV B,#1
66      INC A
67  NOT1:
68      JB KEY2,NOT2
69      MOV B,#2
70      INC A
71  NOT2:
72      JB KEY3,NOT3
73      MOV B,#3
74      INC A
```

This subroutine tests for each key starting from key0; if it is pressed it puts 0 in B register and increment the ACC to indicate that we have one key pressed(61,62)

Then we test for key1; if it is pressed we put 1 in B and increment A -→ So A will have 2 if the two keys are pressed at the same time. And by this technique we can detect multiple keys pressed (The value of A > 1)

The procedure repeats for all keys till key9



```
91  NOT7:
92      JB KEY8,NOT8
93      MOV B,#8
94      INC A
95  NOT8:
96      JB KEY9,NOT9
97      MOV B,#9
98      INC A
99  NOT9:
100     CJNE A,#1,NO_OR_MULTIPLE_KEYS
101     MOV KEY,B
102     RET
103 NO_OR_MULTIPLE_KEYS:
104     CJNE A,#0,MULTIPLE_KEYS
105     RET
106 MULTIPLE_KEYS:
107     MOV KEY,#10 ; MINUS SIGN
108 RET
```

At line 100, we have A contains the number of keys pressed, and B contains the key code for the last pressed one.

If A = 1, so only one key is pressed and we put it into variable "KEY" (101)

If A !=1 → either 0 (no key) or greater than 1 (multiple key).

So we test for 0 in line 104, and if it is zero, we just return without any updates

Else(107) we make key = 10 (to display '-') indicating multiple keys