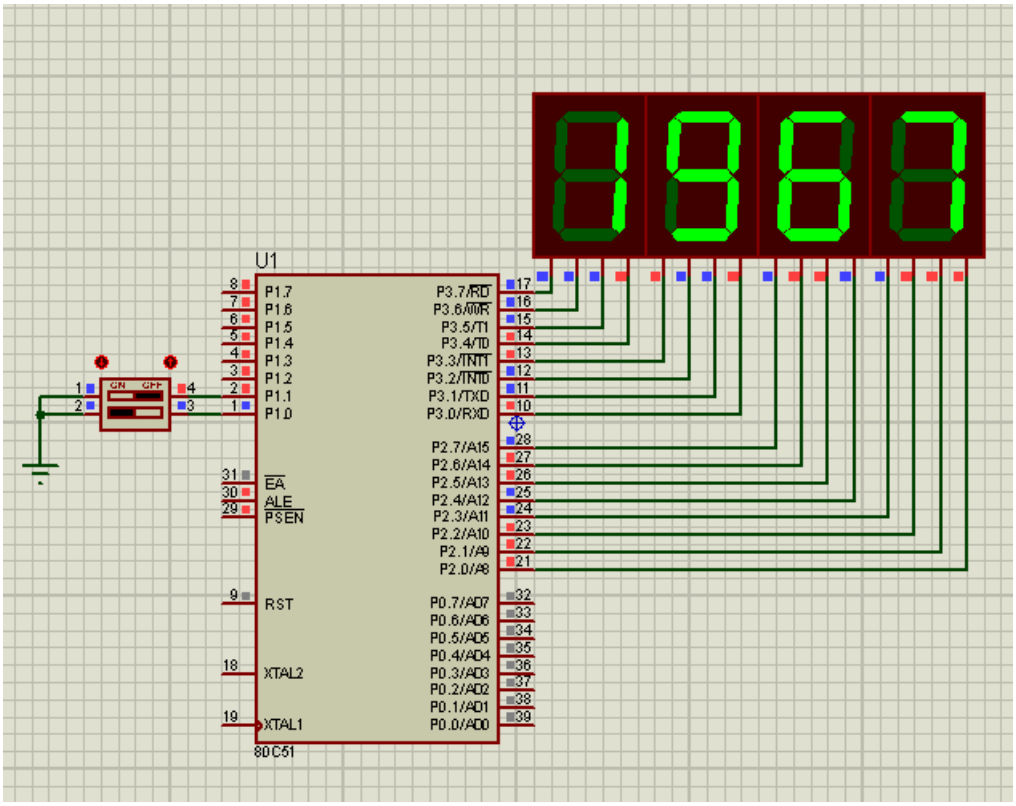
Problem 28



This version of the problem will convert from binary to one of the following base

0-> binary or two hex digts/byte

1-> one hex digit/byte

2-> one bcd digit/byte

3-> two bcd digits/byte

variables

```
LEN EQU 1
TEMP1 EQU 30H
RESULT EQU 38H
TEMP EQU 40H
RC EQU 00H

   ORG 0
```

Same as problem 27 but with only one number to convert

Main code

```
8  START:
9     CALL LOAD_NUMBER
10    MOV RESULT,#0
11    MOV RESULT+1,#0
12    MOV RESULT+2,#0
13    MOV RESULT+3,#0
14    MOV A,P1
15    ANL A,#00000011B
16    CJNE A,#0,NOT0
17    CALL BIN_BASE
18    JMP START
19 NOT0:
20    CJNE A,#1,NOT1
21    CALL ONE_HEX_BASE
22    JMP START
23 NOT1:
24    CJNE A,#2,NOT2
25    CALL ONE_BCD_BASE
26    JMP START
27 NOT2:
28    CJNE A,#3,NOT3
29    CALL TWO_BCD_BASE
30    JMP START
31 NOT3:
32    MOV P2,#0
33    MOV P3,#0
34    JMP START
```

First we load the number into ram at address temp1

Clear the initial result (10-13)

As before; read the first two bits of P1 and branch to one of the subroutine

0 → bin_base

1→ one_hex_base

2→ one_bcd_base

3→ two_bcd_base

Else clear display to 00 (32-33)

Functions

1-BIN_BASE:

```
;================================================
BIN_BASE:
   MOV R7,#LEN
   MOV R0,#TEMP1
   MOV R1,#RESULT
ALL_BIN:
   MOV A,@R0
   MOV @R1,A
   INC R0
   INC R1
   DJNZ R7,ALL_BIN

   CALL DISP_RESULT
RET
;================================================
```

It is very simple where the number is already stored in binary

We move byte by byte from temp1 to result

2-one_hex_base:

```
;================================================
ONE_HEX_BASE:
   MOV R7,#LEN
   MOV R0,#TEMP1
   MOV R1,#RESULT
ALL_HEX:
   MOV A,@R0
   ANL A,#0FH
   MOV @R1,A
   INC R1
   MOV A,@R0
   SWAP A
   ANL A,#0FH
   MOV @R1,A
   INC R1
   INC R0
   DJNZ R7,ALL_HEX

   CALL DISP_RESULT_HEX1
RET
;================================================
```

We make a conversion from one two hex digits/byte to one hex digit/byte

we separate one byte into two nibble with each nibble store in one byte in result as follows:

Read first byte (55); mask off upper nibble (56),and store it in current result byte

Read first byte again (59), move upper nibble to lower nibble position and mask off upper nibble (60-61) and store it in the next result byte (62)

Update pointers (63,64) and repeat

3-ONE_BCD_BASE:

```
70  ONE_BCD_BASE:
71     MOV R7,#LEN
72     MOV R0,#TEMP1
73     MOV R1,#RESULT
74  ;ALL_BCD1:
75     MOV A,@R0
76     MOV B,#10
77     DIV AB
78     MOV @R1,B
79     INC R1
80     MOV B,#10
81     DIV AB
82     MOV @R1,B
83     INC R1
84     MOV @R1,A
85     ;DJNZ R7,ALL_BCD1
86
87     CALL DISP_RESULT_HEX1
88  RET
```

We convert from binary to one bcd base

Read the binary byte (75)

Divide it by 10 (76-77) → B will have the remainder (first bcd digit) store it in the result (78)

Divide again by 10 (80-81) → B now has the 2nd bcd digit; store it (82)

A will have the 3rd digit ; store it (84)

4-TWO_BCD_BASE

```
90  TWO_BCD_BASE:
91     MOV R7,#LEN
92     MOV R0,#TEMP1
93     MOV R1,#RESULT
94  ;ALL_BCD1:
95     MOV A,@R0
96     MOV B,#10
97     DIV AB
98     MOV TEMP,B
99     MOV B,#10
100    DIV AB
101    PUSH ACC
102    MOV A,B
103    SWAP A
104    ADD A,TEMP
105    MOV @R1,A
106    INC R1
107    POP ACC
108    MOV @R1,A
109    ;DJNZ R7,ALL_BCD1
110
111    CALL DISP_RESULT_HEX1
112 RET
```

The same procedure as previous function but with packing two bcd/byte

So after getting the first bcd digit we store it temporary in TEMP (98); then we get the 2nd digit (99-100); we store it in the upper nibble of A (102-103); then we add the first digit to A (104) and store the packed bcd into result (105);

Finally we store the 3rd digit into next result byte (108)