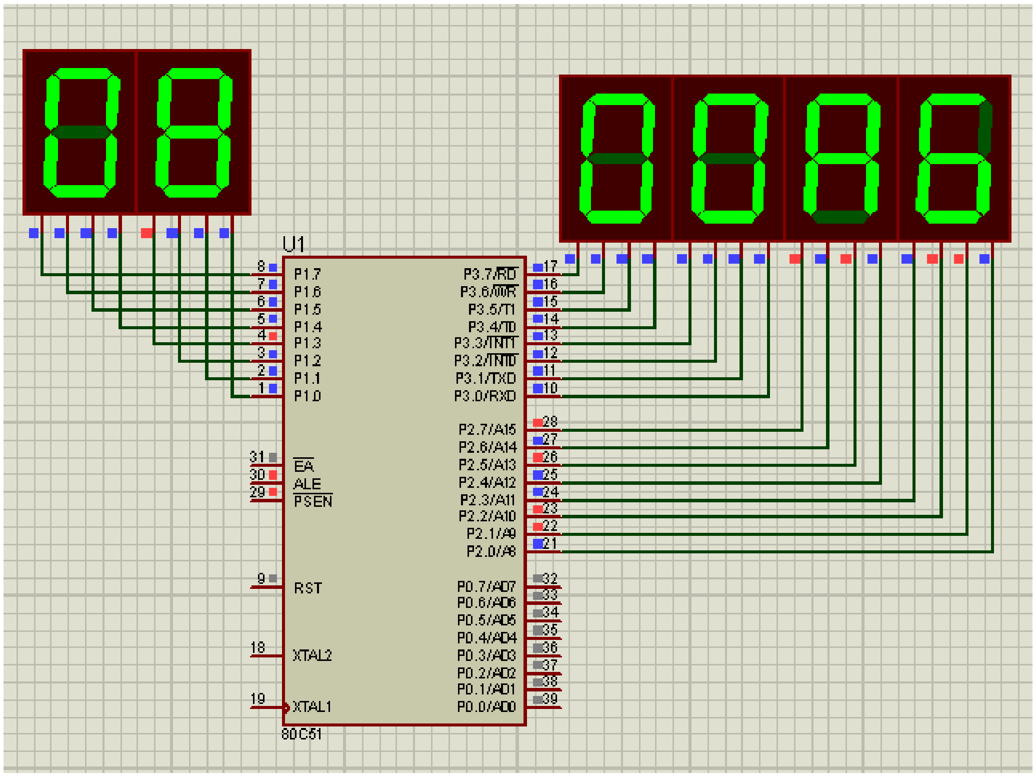
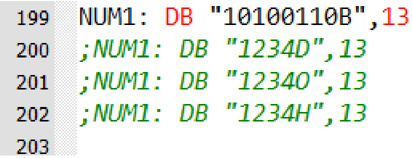
Problem 30



In this problem we will read an ascii number stored in the ROM. This number can be binary, hex, octal, hex, or decimal.

Here is the number stored in memory



```
NUM1: DB "10100110B",13
;NUM1: DB "1234D",13
;NUM1: DB "1234O",13
;NUM1: DB "1234H",13
```

The last character in the number defines its type

'B' → binary

'D' → decimal

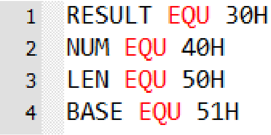
'O'→ Octal

'H'→ hex

CR code '13' defines the end of the number

The code will load this number into memory digit by digit until it find the CR. Then it will test the last character to determine the base for converting the number.

Variables



```
RESULT EQU 30H
NUM EQU 40H
LEN EQU 50H
BASE EQU 51H
```

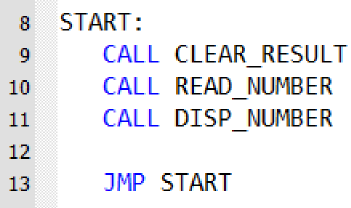
Result stores the result of conversion

LEN → number length

NUM → stores the digits of the number

Base → stors the number base

Main code



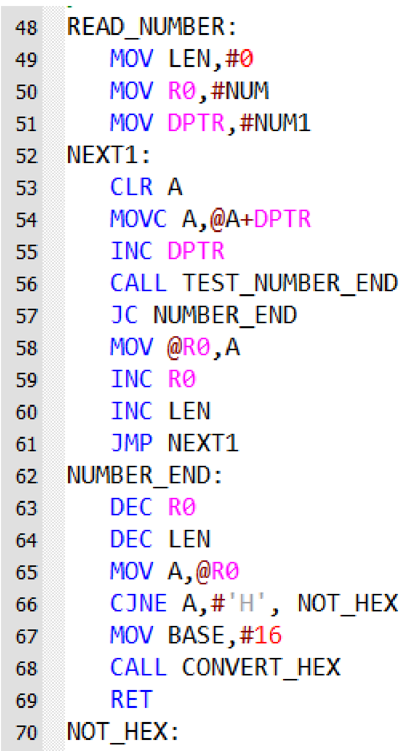
```
8  START:
9      CALL CLEAR_RESULT
10     CALL READ_NUMBER
11     CALL DISP_NUMBER
12
13     JMP START
```

First we reset the result to zero (9)

Call the main function that will convert the number (10)

Display the number (11)

Here is the basic function for this problem



```
48 READ_NUMBER:
49    MOV LEN,#0
50    MOV R0,#NUM
51    MOV DPTR,#NUM1
52 NEXT1:
53    CLR A
54    MOVC A,@A+DPTR
55    INC DPTR
56    CALL TEST_NUMBER_END
57    JC NUMBER_END
58    MOV @R0,A
59    INC R0
60    INC LEN
61    JMP NEXT1
62 NUMBER_END:
63    DEC R0
64    DEC LEN
65    MOV A,@R0
66    CJNE A,#'H', NOT_HEX
67    MOV BASE,#16
68    CALL CONVERT_HEX
69    RET
70 NOT_HEX:
```

R0 points to address of storing number in ram

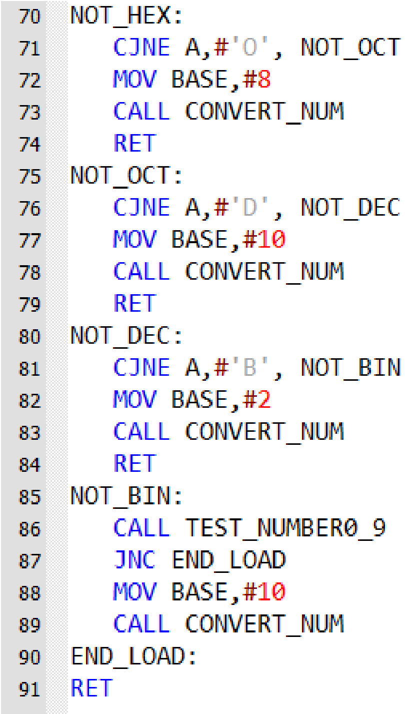
DPTR points to address of original number in ROM

53-55 → read character from ROM

56 → call the function to test if this is the CR character to end the read process

If no carry is returned, we will save the number into RAM (58-60) and read another

If carry → end process of reading (62) where we decrement the pointer and the length (63-64) to point to the last character of the number and make a series of comparison to 'H', 'O', 'D', 'B' to make the appropriate conversion (65-66)



```
NOT_HEX:
   CJNE A,#'O', NOT_OCT
   MOV BASE,#8
   CALL CONVERT_NUM
   RET
NOT_OCT:
   CJNE A,#'D', NOT_DEC
   MOV BASE,#10
   CALL CONVERT_NUM
   RET
NOT_DEC:
   CJNE A,#'B', NOT_BIN
   MOV BASE,#2
   CALL CONVERT_NUM
   RET
NOT_BIN:
   CALL TEST_NUMBER0_9
   JNC END_LOAD
   MOV BASE,#10
   CALL CONVERT_NUM
END_LOAD:
RET
```

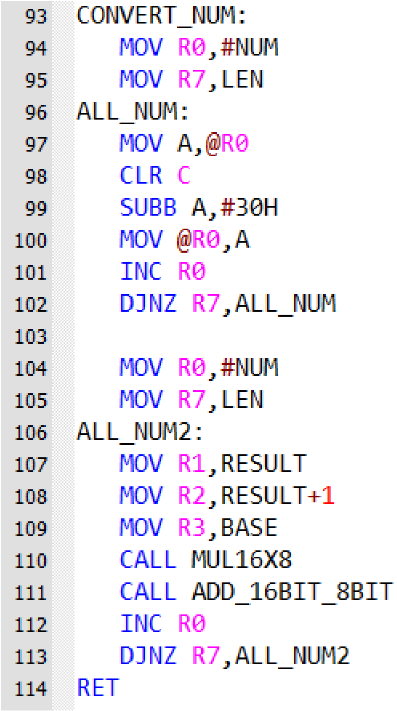
In each case we set the variable base to 2 or 8 or 10 or 16 corresponding to (binary, octal, decimal, hex)

For all number bases except hex, we will call the convert_num function to make the conversion.

Hex conversion is done with the convert_hex function

Functions

1-convert_num



```
 93 CONVERT_NUM:
 94     MOV R0,#NUM
 95     MOV R7,LEN
 96 ALL_NUM:
 97     MOV A,@R0
 98     CLR C
 99     SUBB A,#30H
100     MOV @R0,A
101     INC R0
102     DJNZ R7,ALL_NUM
103 
104     MOV R0,#NUM
105     MOV R7,LEN
106 ALL_NUM2:
107     MOV R1,RESULT
108     MOV R2,RESULT+1
109     MOV R3,BASE
110     CALL MUL16X8
111     CALL ADD_16BIT_8BIT
112     INC R0
113     DJNZ R7,ALL_NUM2
114 RET
```

The first for loop with counter = number length (95)

read ascii number (97)

subtract 30H (99) to convert it to real number (for example '0' has ascii code of 30H, so subtracting 30H → we get 0 as number not ascii)

then we save it again

the 2nd for-loop will accumulate the digits of the number according to its base using this formula

Result = (Result * base) + digit(i)

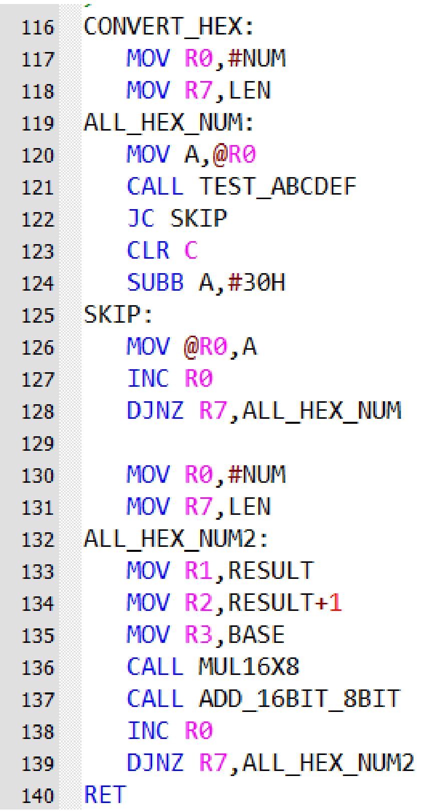
Starting with Result = 0

For this accumulation, we need a multiplication function and addition function

Multiplication is 8bit by 16 bit number "MUL16X8" (110)

Addition of 8bit number to 16 bit result 'ADD_16BIT_8BIT (111)

2- convert_hex



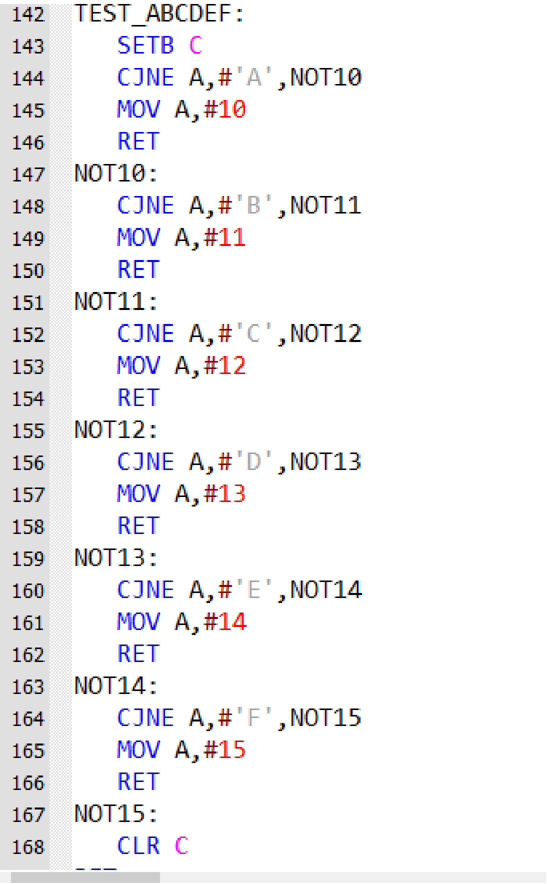
```
116 CONVERT_HEX:
117    MOV R0,#NUM
118    MOV R7,LEN
119 ALL_HEX_NUM:
120    MOV A,@R0
121    CALL TEST_ABCDEF
122    JC SKIP
123    CLR C
124    SUBB A,#30H
125 SKIP:
126    MOV @R0,A
127    INC R0
128    DJNZ R7,ALL_HEX_NUM
129
130    MOV R0,#NUM
131    MOV R7,LEN
132 ALL_HEX_NUM2:
133    MOV R1,RESULT
134    MOV R2,RESULT+1
135    MOV R3,BASE
136    CALL MUL16X8
137    CALL ADD_16BIT_8BIT
138    INC R0
139    DJNZ R7,ALL_HEX_NUM2
140 RET
```

Convert to hex is the same as conver_num except that it will test for non number characters that represents the values from 10 to 15 (A,B,C,D,E,F)→ line 121→ this function will convert each of these characters to the numbers from 10 to 15

If it is not a character, the function will indicate that by clearing the carry, and we make the usual procedure by subtracting 30H to get the number from ascii

The 2nd loop is the same as in convert_num

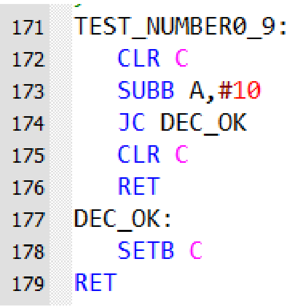
3-TEST_ABCDEF:



```
TEST_ABCDEF:
   SETB C
   CJNE A,#'A',NOT10
   MOV A,#10
   RET
NOT10:
   CJNE A,#'B',NOT11
   MOV A,#11
   RET
NOT11:
   CJNE A,#'C',NOT12
   MOV A,#12
   RET
NOT12:
   CJNE A,#'D',NOT13
   MOV A,#13
   RET
NOT13:
   CJNE A,#'E',NOT14
   MOV A,#14
   RET
NOT14:
   CJNE A,#'F',NOT15
   MOV A,#15
   RET
NOT15:
   CLR C
```

It makes direct comparison to hex characters and assign its corresponding values from 10 to 15

4-TEST_NUMBER0_9

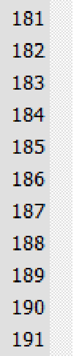


```
171 TEST_NUMBER0_9:
172    CLR C
173    SUBB A,#10
174    JC DEC_OK
175    CLR C
176    RET
177 DEC_OK:
178    SETB C
179 RET
```

This function is used to test if the last character in the ascii string is a number to consider it decimal number.

This is done simply by subtracting 10 from the number (173) and if a carry occurs → A <10 → set the carry as indication(178); else clear the carry(175)

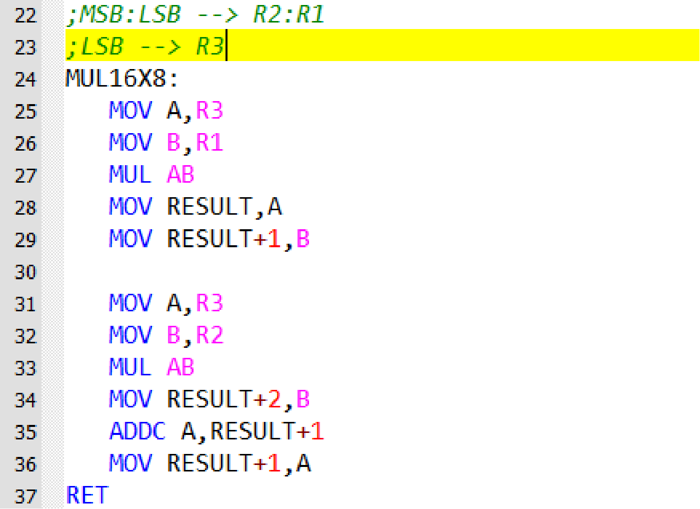
5- TEST_Number_END



```
TEST_NUMBER_END:
    CLR C
    CJNE A,#13,TEST2
    SETB C
    RET
TEST2:
    CJNE A,#10,TEST3
    SETB C
    RET
TEST3:
RET
```

It will compare with CR and LF character and set the carry if a match occurs

6-MUL16X8;



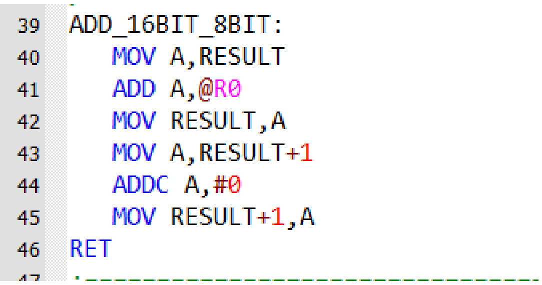
```
22  ;MSB:LSB --> R2:R1
23  ;LSB --> R3
24  MUL16X8:
25      MOV A,R3
26      MOV B,R1
27      MUL AB
28      MOV RESULT,A
29      MOV RESULT+1,B
30
31      MOV A,R3
32      MOV B,R2
33      MUL AB
34      MOV RESULT+2,B
35      ADDC A,RESULT+1
36      MOV RESULT+1,A
37  RET
```

This function will multiply a 16 bit number in R2:R1 by 8-bit number in R3

First we multiply R3 by R1 and save the result in Result, Result+1 (25-29)

31-36 multiply R3 by R2 and save 'B' Higher order to Result +2 (31-34) and add the lower order to Result+1

7-ADD_16BIT_8BIT



```
39  ADD_16BIT_8BIT:
40      MOV A,RESULT
41      ADD A,@R0
42      MOV RESULT,A
43      MOV A,RESULT+1
44      ADDC A,#0
45      MOV RESULT+1,A
46  RET
```

First we add the current digit to result (40-42), then accumulate the carry to result+1 (43-46)