1 Introduction

In this program, we are required to write a program with LC-3 assembly language, which can convert the decimal numbers to hexadecimal one.For example,

if the user input

$$123$$

The program should print

$$007B$$

2 Algorithm Specification

Initially, we devide the program into four functional block:

- store the input characters
- convert ASCII characters into decimal digit
- convert the decimal digit into hexadecimal digit
- convert hexadecimal digit into ASCII characters and output

Firsty, in the input part, use **TRAP x20 and TRAP x21** to read the input character, compare it with CR, if it's not CR, then continue to read. Also the program will record the number of characters inputted.

Next, in the ASCII-to-decimal-digit conversion, we use $result := 10result + number$ to calculate the decimal digit

Then, the program converts the dicimal digit to hexadecimal digit by dividing it with 16 four times. Each time, we store the remainder as the result hexadecimal digit, and assign the quotien as the dicimal digit in the next loop. In the dividing procedure, the sign of the digit should be paid attention to. So in the program, we divide it into two cases. After the decimal digit been transfered into hexadecimal digit, we store the result into memory.

Finally, we check each four hexadecimal digit wheteher they are less than 10, then transfer it into corresponding ASCII characters and use **TRAP x21** to output them.

The peseudocode is follow:

```
n ← 0
while(input != \n)
    c[n++] = input // array storing characters
result ← 0
for i = 0 to n - 1
    result ← result<<3 + result<<1 + c[i] - x0030
if(result>32767) // the first bit is 1
    result ← result - 32768
    quotient ← result/16 + 1
    result ← result - quotien*16 + x7fff + x0001
    quotient ← quotient + result/16
    hex[3] ← result - quotient * 16
    result ← quotient
    for i = 1 to 3
        quotient ← result/16
        hex[3-i] ← result - quotient*16
```

```
                result ← quotient

        else            //the first bit is 0
            for i = 1 to 4
                quotient ← result/16
                hex[4-i] ← result - quotient*16
                result ← quotient
        for i = 1 to 4
            if hex[i-1]<10
                output ← hex[i-1] + x0030
            else
                output ← hex[i-1] + x0041



```

In the peseudocode, the / operator is implement as follow:

```
        quotient ← 0
        remainder ← 0
        while  dividend > 0
            dividend ← dividend - 16
            quotient++
        remainder ← dividend + 16

```

3 Q and A

- Q: what the procedure of your input and output block?

  A: In the input procedure, we use trap x20 and trap x21 to read the character and echo it. Then we store the character into memory.

  In the output procedure, we use the hexadecimal digit calculated before, transfer it into ascii character and use trap x21 to output it.

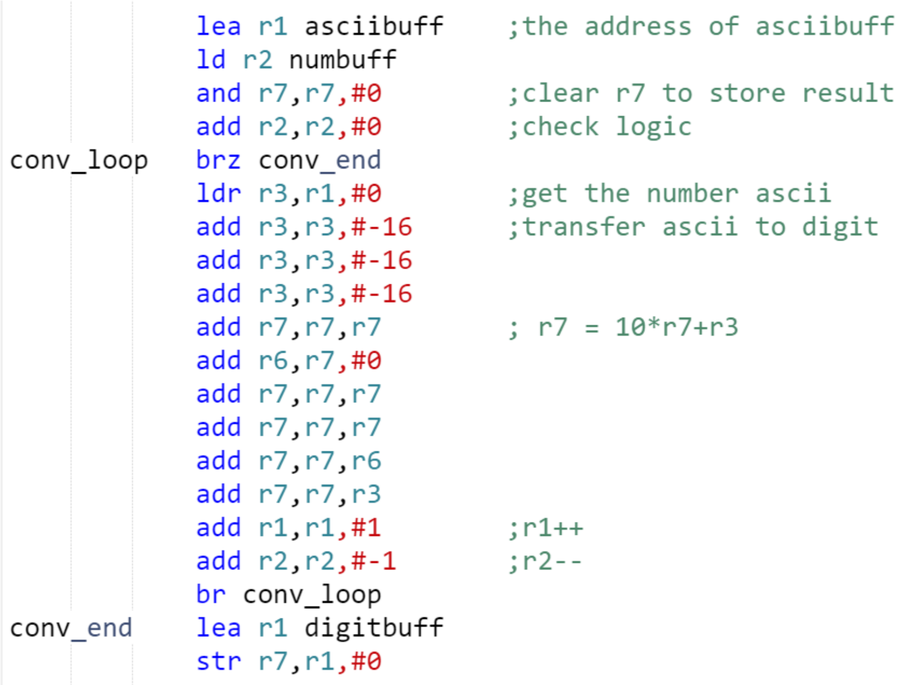
- Q: Is your program use divide operation?

  A: yes. In the program, divide operation is used to transform the decimal digit into hexadecimal digit.

4 essential parts of code

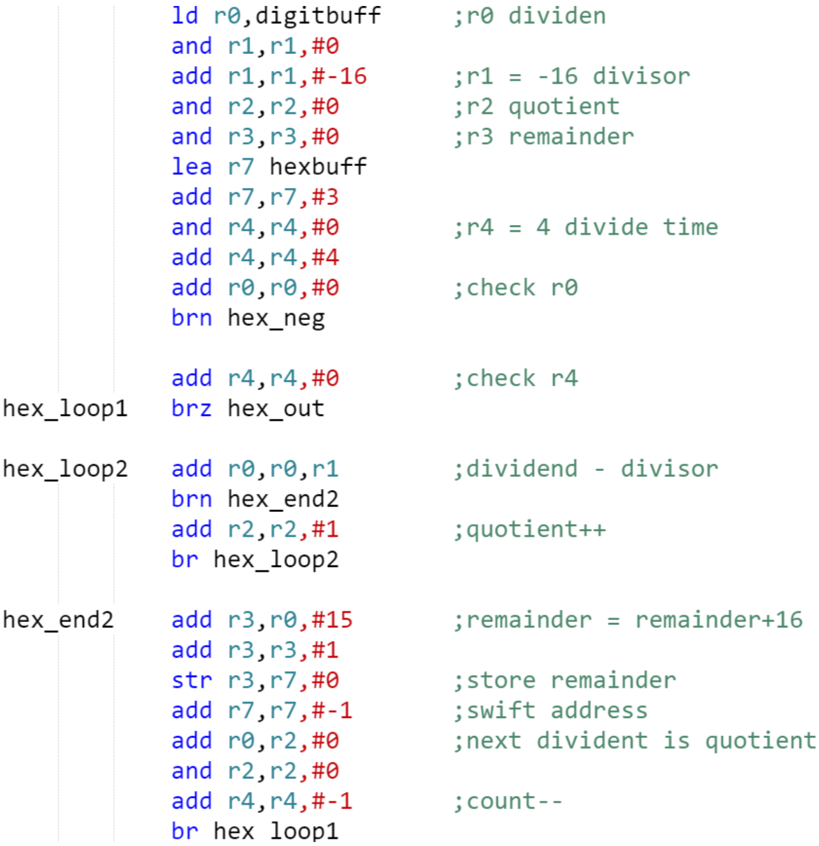
Fig 1 is the code to transfer the ascii character to digit

Fig 2 is transfer the decimal digit to hexadecimal digit code(positive case)



```
                lea r1 asciibuff    ;the address of asciibuff
                ld r2 numbuff
                and r7,r7,#0        ;clear r7 to store result
                add r2,r2,#0        ;check logic
conv_loop       brz conv_end
                ldr r3,r1,#0        ;get the number ascii
                add r3,r3,#-16      ;transfer ascii to digit
                add r3,r3,#-16
                add r3,r3,#-16
                add r7,r7,r7        ; r7 = 10*r7+r3
                add r6,r7,#0
                add r7,r7,r7
                add r7,r7,r7
                add r7,r7,r6
                add r7,r7,r3
                add r1,r1,#1        ;r1++
                add r2,r2,#-1       ;r2--
                br conv_loop
conv_end        lea r1 digitbuff
                str r7,r1,#0
```

Figure 1: Fig 1



```
            ld r0,digitbuff     ;r0 dividen
            and r1,r1,#0
            add r1,r1,#-16      ;r1 = -16 divisor
            and r2,r2,#0        ;r2 quotient
            and r3,r3,#0        ;r3 remainder
            lea r7 hexbuff
            add r7,r7,#3
            and r4,r4,#0        ;r4 = 4 divide time
            add r4,r4,#4
            add r0,r0,#0        ;check r0
            brn hex_neg

            add r4,r4,#0        ;check r4
hex_loop1   brz hex_out

hex_loop2   add r0,r0,r1        ;dividend - divisor
            brn hex_end2
            add r2,r2,#1        ;quotient++
            br hex_loop2

hex_end2    add r3,r0,#15       ;remainder = remainder+16
            add r3,r3,#1
            str r3,r7,#0        ;store remainder
            add r7,r7,#-1       ;swift address
            add r0,r2,#0        ;next divident is quotient
            and r2,r2,#0
            add r4,r4,#-1       ;count--
            br hex loop1
```

Figure 2: Fig 2