JinR:LC-3 Divide by 10 Calculator

I. Getting & Storing User Input

```
.ORIG x3000
UserInputPtr .BLKW 6
ResultPtr .BLKW 6

Begin LEA R0, Prompt          ; R0 = beginning of prompt
      puts
      LD R5, AsciiN           ; R5 = -x30
      LEA R6, UserInputPtr    ; R6 = beginning of UserInputPtr

READ  getc
      out
      ADD R3, R0, x-A         ; Check Enter Key
      BRz ADDNull
      ADD R0, R0, R5          ; Ascii --> Decimal (UserInput-x30)
      STR R0, R6, #0          ; Store R0 at UserInputPtr
      ADD R6, R6, #1          ; Increase pointer by 1
      BRnzp READ        ; if no enter key, keep getting user input

ADDNull
      STR R0, R6, #0          ; R0 = xA. store xA at end of string
      AND R1, R1, #0
      AND R3, R3, #0
      BRnzp ParseDecimal
```

The above code will subtract x30 from user input ASCII code and store each digit into UserInputPtr. Because UserInputPtr is right below at .ORIG x3000, the values would be consecutively stored at x3000 to x3006. If the user press enter key (xA), the loop will end and xA would be stored at the end of string location

If the user input is 123, then the value converted is

User Input Ascii Code	ASCII Conversion ADD x-30	Memory Location	Value stored
x31		x3000	x1
x32		x3001	x2

x33		x3002	x3
Enter key	→	x3003	xA (Enter key)

II. Parsing individual character to integer

```
ParseDecimal
      JSR WIPE
      LEA R6, UserInputPtr    ; R6 = User Input Pointer
      BRnzp LOOP

LOOP
      LDR R1, R6, #0          ; R1 = first letter
      ADD R2, R1, x-A   ; Check whether the letter is enter key or not
      BRz Divide
      AND R2, R2, #0

      ADD R7, R3, R3          ; R7 = 2R3
      ADD R3, R7, R7          ; R3 = 4R3
      ADD R3, R3, R3          ; R3 = 8R3
      ADD R3, R3, R7          ; R3 = 8R3 + 2R3

      ADD R3, R3, R1
      ADD R6, R6, #1          ; Increase Pointer
      BRnzp LOOP        ; Result: R3
```

$$abc_{10} = 100 * a + 10 * b + 1 * c$$

Every time the ParseDecimal Loop starts, it will multiply the previously digit values by 10.
If the user input is 3 digit, then

$$Final\ Value\ stored\ at\ R3 = 10 * (10 * (M[x3000]) + M[x3001]) + M[x300$$

Memory	Value
x3000	x1
x3001	x2
x3002	x3
x3003	x4
x3005	x5

$$In\ this\ case,\ the\ final\ value\ of\ R3\ at\ the\ end\ of\ Loo$$

$$= 10 * (10 * (10 * (10 * (10 * 1) + 2) + 3) +$$

$$= 12345$$

x3006	xA (Enter)	

III. Divide by 10 Calculation

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Count how many times can 10 be subtracted
; It will discard the remainders

Divide
       AND R2, R2, #0
Subtr0 ADD R2, R2, #1          ; Subtract 10 Counter
       ADD R3, R3, x-A
       BRzp Subtr0
       ADD R2, R2, #-1
       BRnzp DecimalTemplate
```

For example,

- 128 = 10* 13 - 2
- 200 = 10*21 - 10
- 15 = 10*2 - 5
- 479 = 10*48 - 1

Because the subtract 10 counter always have 1 excess, the 1 should be subtracted from subtr10 counter by `ADD R2, R2, #-1` The Remainder will be discarded. The Counter value would be saved to R2.

IV. Displaying the value of Register into Decimal.

```
DecimalTemplate              ; Print Value of R2 in decimal

      LD R6, AsciiP          ; R6 = x30
      LEA R4, ResultPtr      ; R4 = Savepoint
                             ; R5 = digit counter
                             ; R7 = Base 10 constants

      LD R7, NTenThousand    ; R7 = -10,000
Subtr1ADD R5, R5, #1         ; R5 = n*10,000 digit
      ADD R2, R2, R7         ; Subtract 10,000 several times from R2
      BRzp Subtr1            ; if R2 is negative, end of loop
      ADD R5, R5, #-1   ; subtract excessive 1
      ADD R5, R5, R6         ; change n*10,000 digit into Ascii
      STR R5, R4, #0         ; Store R5 to R4
      ADD R4, R4, #1         ; increase savepoint by 1
```

```
      AND R5, R5, #0
      LD R7, PTenThousand
      ADD R2, R2, R7          ; ADD Positive 10,000 to compensate

      LD R7, NThousand  ; Loop for 1000
Subtr2ADD R5, R5, #1
      ADD R2, R2, R7
      BRzp Subtr2
      ADD R5, R5, #-1
      ADD R5, R5, R6
      STR R5, R4, #0
      ADD R4, R4, #1

      AND R5, R5, #0
      LD R7, PThousand
      ADD R2, R2, R7

      LD R7, NHundred   ; Loop for 100
Subtr3ADD R5, R5, #1
      ADD R2, R2, R7
      BRzp Subtr3
      ADD R5, R5, #-1
      ADD R5, R5, R6
      STR R5, R4, #0
      ADD R4, R4, #1          ; increase pointer by 1

      AND R5, R5, #0
      LD R7, PHundred   ; R7 = #100
      ADD R2, R2, R7

Subtr4ADD R5, R5, #1
      ADD R2, R2, x-A   ; Subtract 10
      BRzp Subtr4
      ADD R5, R5, #-1
      ADD R5, R5, R6          ; Convert to Ascii Template
      STR R5, R4, #0

      ADD R2, R2, #10

      ADD R2, R2, R6
      ADD R4, R4, #1
      STR R2, R4, #0          ; The last digit is R1.

      BRnzp PrintDecimal

PrintDecimal
```

```
    LEA R0, Prompt2
    puts
    LEA R0, ResultPtr
    puts
    BRnzp Begin
```

This is probably the hardest part of this code. In order to display the value in the register by decimal, each digits were saved in ResultPtr.

By counting how many times we can subtract the base 10 exponent, we can get the each digit ASCII codes from the Register value. The maximum positive integer that LC-3 can have is $2^{16-1} - 1$, which is 32767. Thus, 4 loops are required in order to find out each digits

- 10^4 subtract loop
- 10^3 subtract loop
- 10^2 subtract loop
- 10^1 subtract loop
- 10^0 subtract loop is not required because it will be same as remainder

For example,

- $4728 = 0 * 10^4 + 4 * 10^3 + 7 * 10^2 + 2 * 10^1 + 8 * 10^0$
- $17349 = 1 * 10^4 + 7 * 10^3 + 3 * 10^2 + 4 * 10^1 + 9 * 10^0$
- $12 = 0 * 10^4 + 0 * 10^3 + 0 * 10^2 + 1 * 10^1 + 2 * 10^0$

If 15 goes through subtraction loop,
$15 + (-10) * 2 = -5.$
$-5 + 10 = 5.$
Thus, we know that the 10^1 digit is $2 - 1 = 1$and 10^0 digit is 5.

If 243 goes through subtraction loop,
$243 + (-100) * 3 = -57.$
$-57 + 100 = 43.$
$43 + (-10) * 5 = -7$
$-7 + 10 = 3$
Thus, we know that 10^2digit is $3 - 1 = 1$and 10^1digit is$5 - 1 = 4$ and 10^0digit is3.

If 17456 goes through subtract loop,
$17456 = [[[\{(17456 - 10,000 * 2) + 10,000\} - 1000 * 8 + 1000] - 100 * 5 + 100] - 10 * 6 + 10]$

17456 Subtraction loop	
10,000	(-10,000)*2 until it becomes negative
1000	(-1000)*8 until it becomes negative
100	(-100)*5 until it becomes negative

10	(-10)*6 until it becomes negative
1	Remainder

In order to subtract them easily, an constant table was made

```
AsciiN .fill x-30 ; Ascii conversion
AsciiP .fill x30  ; Ascii conversion

NTenThousand .fill #-10000
PTenThousand .fill #10000
NThousand .fill #-1000
PThousand .fill #1000
NHundred .fill #-100
PHundred .fill #100
```

By using those constants, the loops are made more easily.

```
LD R7, NTenThousand     ; R7 = -10,000
Subtr1ADD R5, R5, #1          ; R5 = n*10,000 digit
      ADD R2, R2, R7          ; Subtract 10,000 several times from R2
      BRzp Subtr1             ; if R2 is negative, end of loop
      ADD R5, R5, #-1   ; subtract excessive 1
      ADD R5, R5, R6          ; change n*10,000 digit into Ascii
      STR R5, R4, #0          ; Store R5 to R4
      ADD R4, R4, #1          ; increase savepoint by 1

      AND R5, R5, #0
      LD R7, PTenThousand
      ADD R2, R2, R7          ; ADD Positive 10,000 to compensate
```

This is 10,000 loop.

If 42,000 goes through this loop,

$$4,2000 + NTenThousand * 5 = -8000$$
$$-8000 + PTenThousand = 2000$$

By counting how many times we can subtract NTenThousand, we can find out the digit of 10,000 by subtracting 1 to the subtr1 loop counter.

V. Final Result