```
; read a decimal number from the keyboard,
; convert it from ASCII to 2's complement, and
; store it in a predefined memory location.  If
; any non-numeric character is pressed, or the
; number overflows, store a 0 and print an error
; message.

; R0 holds the value of the last key pressed
; R1 holds the current value of the number being input
; R2 holds the additive inverse of ASCII '0' (0xFFD0)
; R3 is used as a temporary register

00110000 00000000     ; starting address is x3000

0010 010 000010100    ; LD R2,x14      (put the value -x30 in R2)
0101 001 001 1 00000  ; AND R1,R1,#0   (clear the current value)
1111 0000 00100000    ; TRAP x20       (read a character)
1111 0000 00100001    ; TRAP x21       (echo it back to monitor)
0001 011 000 1 10110  ; ADD R3,R0,#-10 (compare with ENTER)
0000 010 000010001    ; BRz x11        (ENTER pressed, so done)
0001 000 000 0 00 010 ; ADD R0,R0,R2   (subtract x30 from R0)
0000 100 000010001    ; BRn x11        (smaller than '0' means error)
0001 011 000 1 10110  ; ADD R3,R0,#-10 (check if > '9')
0000 011 000001111    ; BRzp xF        (greater than '9' means error)
0001 011 001 0 00 001 ; ADD R3,R1,R1   (sequence of adds multiplies R1 by 10)
0000 100 000010101    ; BRn x15        (overflow, but not really necessary here)
0001 011 011 0 00 011 ; ADD R3,R3,R3
0000 100 000010011    ; BRn x13        (overflow, but not really necessary here)
0001 001 001 0 00 011 ; ADD R1,R1,R3
0000 100 000010001    ; BRn x11        (overflow)
0001 001 001 0 00 001 ; ADD R1,R1,R1
0000 100 000001111    ; BRn xF         (overflow)
0001 001 001 0 00 000 ; ADD R1,R1,R0   (finally, add in new digit)
0000 100 000001101    ; BRn xD         (overflow)
0000 111 111101101    ; BRnzp 0x1ED    (get another digit)

11111111 11010000     ; the additive inverse of ASCII '0'
11111111 11111111     ; storage for the result

; done
0011 001 111111110    ; ST R1,x1FE
1111 0000 00100101    ; TRAP x25

; print error message: "non-digit pressed"
; R4 holds pointer to character
; R0 used to pass output character to trap
; end of string marked with negative value
1110 100 000001001    ; LEA R4,x9      (point R4 to the start of the string)
0110 000 100 000000   ; LDR R0,R4,#0   (read character pointed to by R4)
0000 100 000000011    ; BRn x3         (done printing)
1111 0000 00100001    ; TRAP x21       (print the character)
0001 100 100 1 00001  ; ADD R4,R4,#1   (point to next character)
0000 111 111111011    ; BRnzp x1FB     (loop to read string)
0101 001 001 1 00000  ; AND R1,R1,#0   (clear the current value)
0000 111 111110110    ; BRnzp x1F6     (store the zero and end)

; print error message: "overflow"
1110 100 000010101    ; LEA R4,x15     (point R4 to the start of the string)
0000 111 111110111    ; BRnzp x1F7     (branch back to the string-printing code)

; first error message (all in ASCII)
00000000 00001010     ; LF (line feed)
00000000 01101110     ; 'n'
00000000 01101111     ; 'o'
00000000 01101110     ; 'n'
00000000 00101101     ; '-'
00000000 01100100     ; 'd'
00000000 01101001     ; 'i'
00000000 01100111     ; 'g'
00000000 01101001     ; 'i'
00000000 01110100     ; 't'
00000000 00100000     ; ' '
00000000 01110000     ; 'p'
00000000 01110010     ; 'r'
00000000 01100101     ; 'e'
00000000 01110011     ; 's'
00000000 01110011     ; 's'
00000000 01100101     ; 'e'
00000000 01100100     ; 'd'
00000000 00001010     ; LF (line feed)
11111111 11111111     ; end of string marker

; second error message (all in ASCII)
00000000 00001010     ; LF (line feed)
00000000 01101111     ; 'o'
00000000 01110110     ; 'v'
00000000 01100101     ; 'e'
00000000 01110010     ; 'r'
00000000 01100110     ; 'f'
00000000 01101100     ; 'l'
00000000 01101111     ; 'o'
00000000 01110111     ; 'w'
00000000 00001010     ; LF (line feed)
11111111 11111111     ; end of string marker
```