```
.define HEX_ADDRESS 0x2000
.define SW_ADDRESS 0x3000
.define MAX_SPEED 0x1000
.define STACK 255
.define COUNT_STORAGE 250

// This program displays a counter on the 7 segments
// The speed of the counter is controlled by the switches
            mvi     r0, #0                  // Used for counting
            mvi     r1, #1                  // Used for add/sub 1
MAIN:       mvi     r4, #SW_ADDRESS         // Point to switches
            ld      r6, [r4]                // Read SW values
            add     r6, r1                  // Add 1 for minimum delay

// Count down delay until it reaches 0
DELAY1:     mvi     r5, #MAX_SPEED          // Reset max speed delay counter
            mvi     r3, #DELAY2             // Point to inner delay loop

// Each delay counter will count MAX_SPEED times
DELAY2:     sub     r5, r1                  // Count down by 1's
            mvnz    r7, r3                  // Continue inner delay loop
// End of DELAY2

            sub     r6, r1                  // Count down by 1's
            mvi     r3, #DELAY1             // Point to outer delay loop
            mvnz    r7, r3                  // Continue outer delay loop
// End of DELAY1

            add     r0, r1                  // Increment counter
            mvi     r3, #COUNT_STORAGE      // Store the counter because r0 is going to be modified
            st      r0, [r3]

// Display the counter in decimal
            mvi     r4, #HEX_ADDRESS        // Point to HEX port
            mvi     r6, #6                  // Used to count number of times DISPLAY must loop
DISPLAY:    mv      r5, r7                  // Return address for DIV10
            mvi     r7, #DIV10              // Call DIV10 subroutine

            mvi     r5, #DATA               // Used to get display pattern

            add     r5, r0                  // Point to correct display pattern
            ld      r3, [r5]                // Load display pattern
            st      r3, [r4]                // Light up HEX display
            add     r4, r1                  // Go to next HEX display

            mv      r0, r2                  // Move quotient to r0 for next division

            sub     r6, r1                  // Decrement number of times DISPLAY still need to loop
            mvi     r3, #DISPLAY            // Point to display loop
            mvnz    r7, r3                  // Keep looping until DISPLAY has looped 6 times (r6=0)
// End of DISPLAY

            mvi     r3, #COUNT_STORAGE      // Get the actual counter back
            ld      r0, [r3]
            mvi     r7, #MAIN               // Endless looping

// subroutine DIV10
// This subroutine divides the number in r0 by 10
// The algorithm subtracts 10 from r0 until r0 < 10, and keeps count in r2
// input: r0
// returns: quotient Q in r2, remainder R in r0
DIV10:      mvi     r1, #1
            mvi     r3, #STACK              // Save registers on stack
            sub     r3, r1                  // save registers that are modified
            st      r6, [r3]
            sub     r3, r1
            st      r4, [r3]            // end of register saving
            mvi     r2, #0              // init Q
            mvi     r6, RETDIV          // for branching

DLOOP:      mvi     r4, #9              // check if r0 is < 10 yet
            sub     r4, r0
            mvnc    r7, r6              // if so, then return

INC:        add     r2, r1              // but if not, then increment Q
            mvi     r4, #10
            sub     r0, r4              // r0 -= 10
            mvi     r7, DLOOP           // continue loop

RETDIV:     ld      r4, [r3]            // restore saved regs
            add     r3, r1
            ld      r6, [r3]            // restore the return address
            add     r3, r1
            add     r5, r1              // adjust the return address by 2
            add     r5, r1
            mv      r7, r5              // return results

// DATA for 7 segments
DATA:       .word 0b00111111            // '0'
            .word 0b00000110            // '1'
            .word 0b01011011            // '2'
            .word 0b01001111            // '3'
            .word 0b01100110            // '4'
            .word 0b01101101            // '5'
            .word 0b01111101            // '6'
            .word 0b00000111            // '7'
            .word 0b01111111            // '8'
            .word 0b01101111            // '9'
```