```
;***************************************************************************
;*
;* Title:           multiplex_display.asm
;* Author:          Jason Chen
;* Version:         1
;* Last updated:    10/24/2022
;* Target:          AVR128DB48
;*
;* DESCRIPTION
;*          Design Task 2:
;*          Allocate the memory for led_display and digit_num and configures
;*          PORTD and PORTC. The main loop of the program consists of a call
;*          to subroutine multiplex_display.
;*
;* VERSION HISTORY
;* 1.0 Original version
;***************************************************************************

start:
    ldi r16, 0x00
    out VPORTC_DIR, r16             ; VPORTC - all pins configured as input
    ldi r16, 0xFF
    out VPORTC_DIR, r16             ; VPORTD - all pins configured as output
    ldi XH, HIGH(PORTC_PIN0CTRL)    ; X points to PORTC_PIN0CTRL
    ldi XL, LOW(PORTC_PIN0CTRL)
    ldi r17, 8                      ; loop control variable, 8 step counter

    .dseg               ; start of data segment
    led_display: .byte 4
    digit_num: .byte 1

/*pullups:
    ld r16, X           ; load value of PORTC_PINnCTRL
    ori r16, 0x88       ; enable input bits invert and pullup resistors
    st X+, r16          ; store results
    dec r17             ; decrement lcv
    brne pullups*/

    .cseg               ; start of code segment

main_loop:
    rcall multiplex_display
    rjmp main_loop

;***************************************************************************
;*
;* "multiplex_display" - Multiplex the Four Digit LED Display
;*
;* DESCRIPTION
;*          Updates a single digit of the display and increments the
;*          digit_num to the digit position to be displayed next.
;*
```

```
;* Author:          Jason Chen
;* Version:         1
;* Last Updated:    10/24/2022
;* Target:          AVR128DB48
;* Number of words:
;* Number of cycles:
;* Low registers modified:  none
;* High registers modified  none
;*
;* Parameters:
;*      led_display: a four byte array that holds the segment values
;*          for each digit of the display. led_display[0] holds the
;*          segment patter for digit 0 (the rightmost digit) and so on.
;*      digit_num: byte variable, the least significant two bits are the
;*          index of the last digit displayed.
;*
;* Returns: Outputs segment pattern and turns on digit driver for the next
;*          position in the display to be turned ON.
;* Notes:   The segments are controlled by PORTD - (dp, a through g), the
;*          digit drivers are controlled by PORTA (PA7 - PA4, digit 0 - 3).
;***************************************************************************

multiplex_display:
    ldi r16, 0xFF           ; turn all segments OFF
    out VPORTD_OUT, r16
;   in r16, VPORTA_OUT      ; get current value of VPORTA
;   ori r16, 0xF0           ; turn all digits OFF
; necessary if PA0 - PA3 have a purpose, otherwise treat as don't care
    out VPORTA_OUT, r16

    ldi XH, HIGH(led_display)   ; set pointer X to start of led_display array
    ldi XL, LOW(led_display)

    lds r16, digit_num      ; get current display number
    inc r16
    andi r16, 0x03          ; mask for two least significant bits
    sts digit_num, r16

    add XL, r16             ; add digit number to offset to array pointer

;   brcc PC + 2             ; if no carry skip next instruction
;   inc XH                  ; increment high pointer byte because carry occurred
; i think this is for cases where digit_num is allocated a certain memory
; address causing the addition to create a carry bit.

    ld r17, X
    out VPORTD_OUT, r17     ; output to segment display driver port

    in r17, VPORTA_OUT      ; get current digit driver port value
    ldi r18, 0x10           ; for next PORTA value via bit shift

    digit_pos:
        cpi r16, 0          ; if digit number is 0, use pattern in r18
        breq digit_on
        lsl r18             ; r18 shifted left if not 0
        dec r16             ; decrement digit number offset
        rjmp digit_pos
    digit_on:
        eor r17, r18        ; complement digit driver position indicated by r18
        out VPORTA_OUT, r17 ; turn selected digit ON
    ret
```