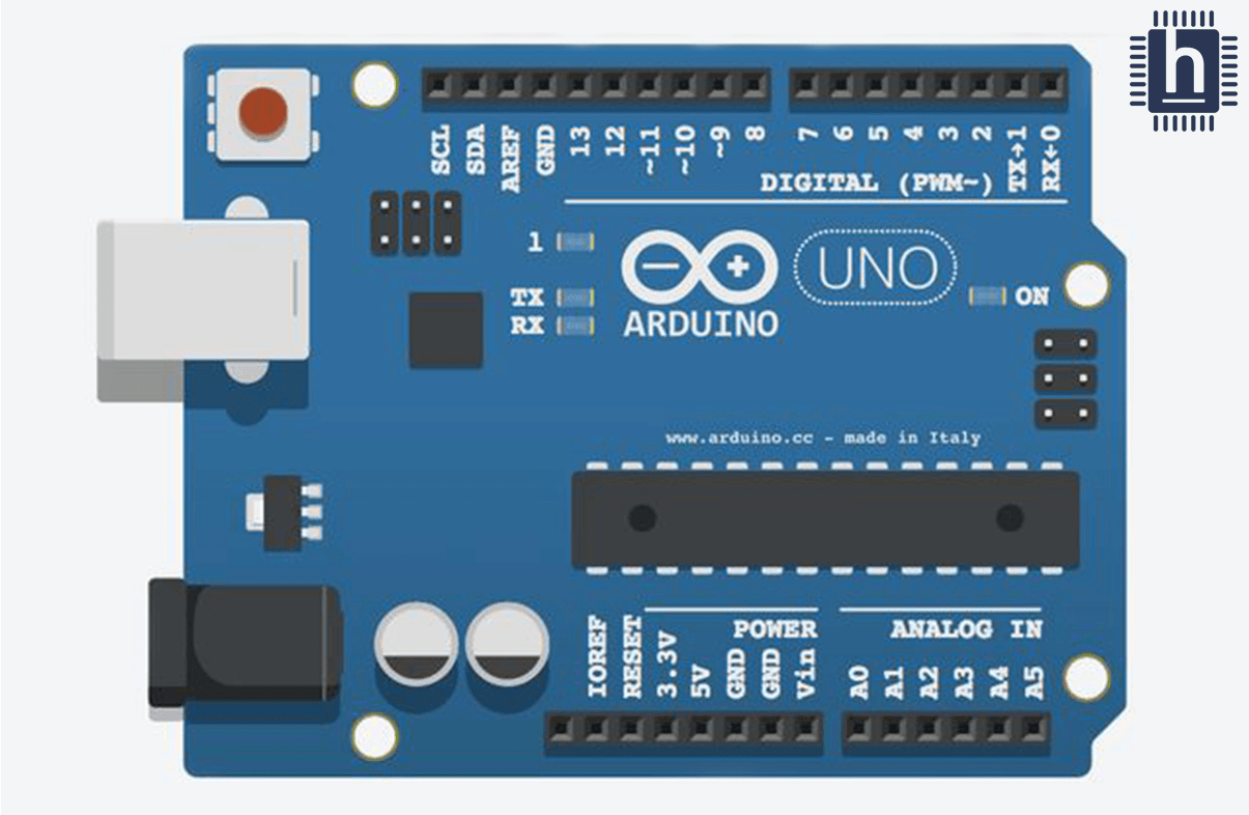
Authors: Max Finch, Tiger Slowinski
Team Members: Max Finch, Tiger Slowinski
ECE:3360 Embedded Systems
Post-Lab Report 2

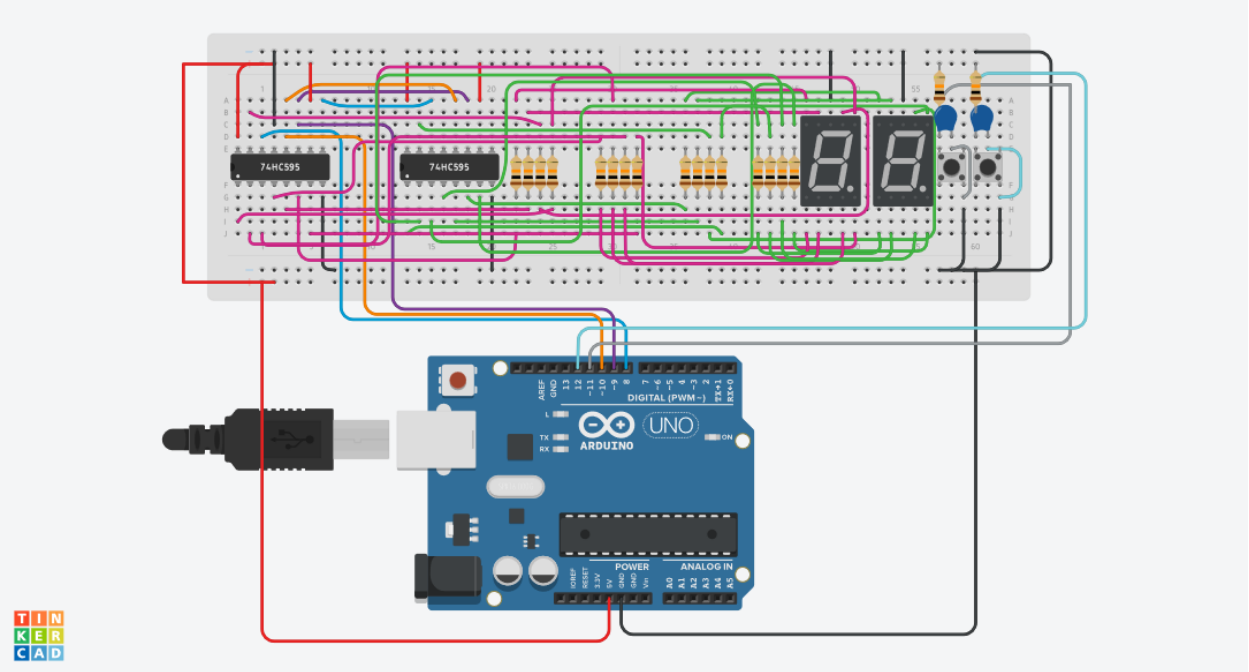


1. Introduction

The goal of this lab was to create a microcontroller based countdown timer that is able to countdown between 1 and 25 seconds. The timer utilizes two cascading 8 bit shift registers, two seven segment displays, and two pushbuttons. Operating the first pushbutton with a short press and release increments the displayed digit by one, and operating the same pushbutton with a long press and

release resets the display to zero. Operating the second pushbutton will countdown from the number displayed at a rate of 1 digit per second.

2. Schematic



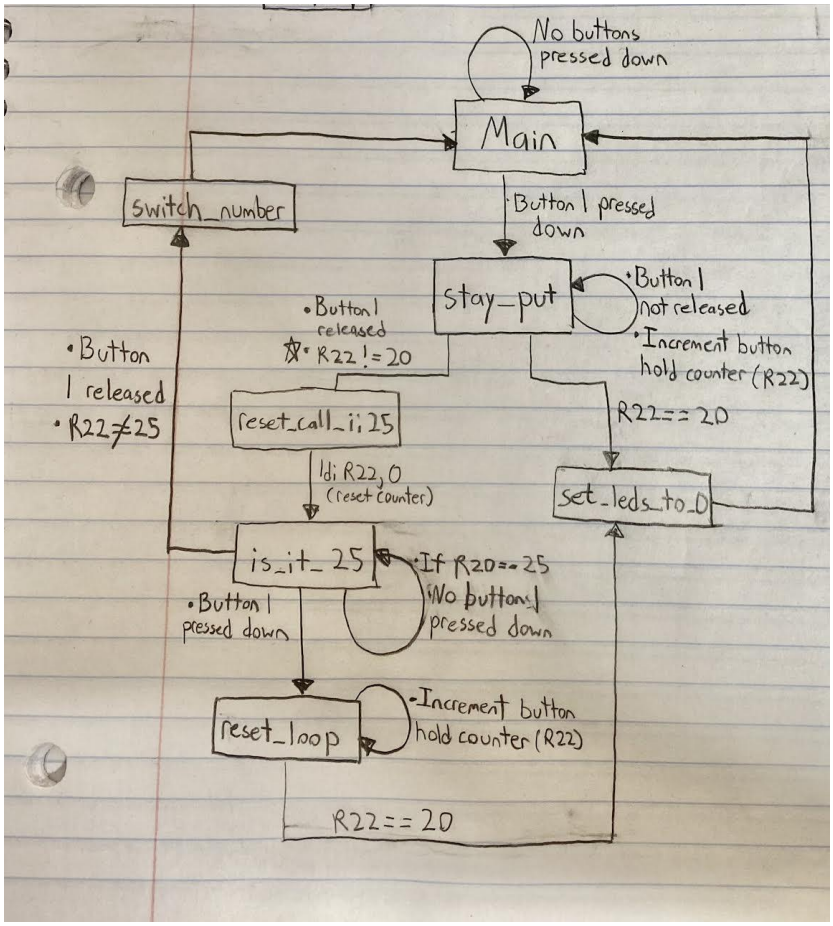
3. Discussion

The majority of the hardware work for this circuit was centered around the pushbuttons and seven segment displays. The pushbuttons are set to be actively high (5V), with a pullup resistor ensuring that the pin going to the microcontroller is only ever at one of two possible states. A 100pF capacitor is connected across the terminals of the pushbutton, which works as a filter for “noise”, in this case bounces from having a hardware pushbutton. A discharge resistor is not necessary here, as with such a small capacitance the rate of discharge is sufficient enough so that the switch is properly debounced, but the response of the button is not sluggish.

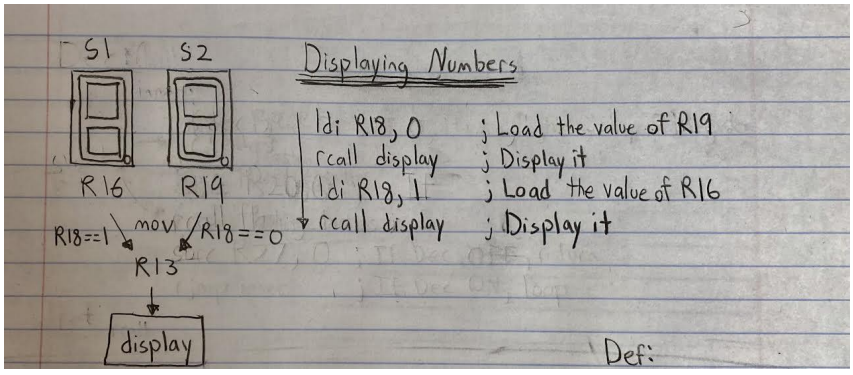
The software required first identifying the various states the timer could be in, as well as identifying the conditions necessary for a state change to occur. These states are summarized in the figure below.

State	Description	Conditions for next state
Initial	Displays “00”, awaits user input.	Interaction with pushbutton.
Displaying static digits “00” to “24”	Digits do not change unless one of the pushbuttons is interacted with.	Increment by one digit: - Pushbutton A is pressed and released within one second. If the number post increment is <25, the timer is still in this state. If the number post increment is 25, the state is now “displaying 25”. Resetting timer: - Pushbutton A is pressed and held for more than 1 second. Upon release, it goes to initial state. Countdown: - Pushbutton B is pressed and released. Enters countdown state.
Displaying “25”	Display can no longer be incremented any further. Awaits input from the user.	Same as above, except “Increment by one digit” is no longer possible.
Countdown	Counts down from the number originally shown at a rate of one per second until “00” is	No state change is possible at this time, aside from the display changing due to

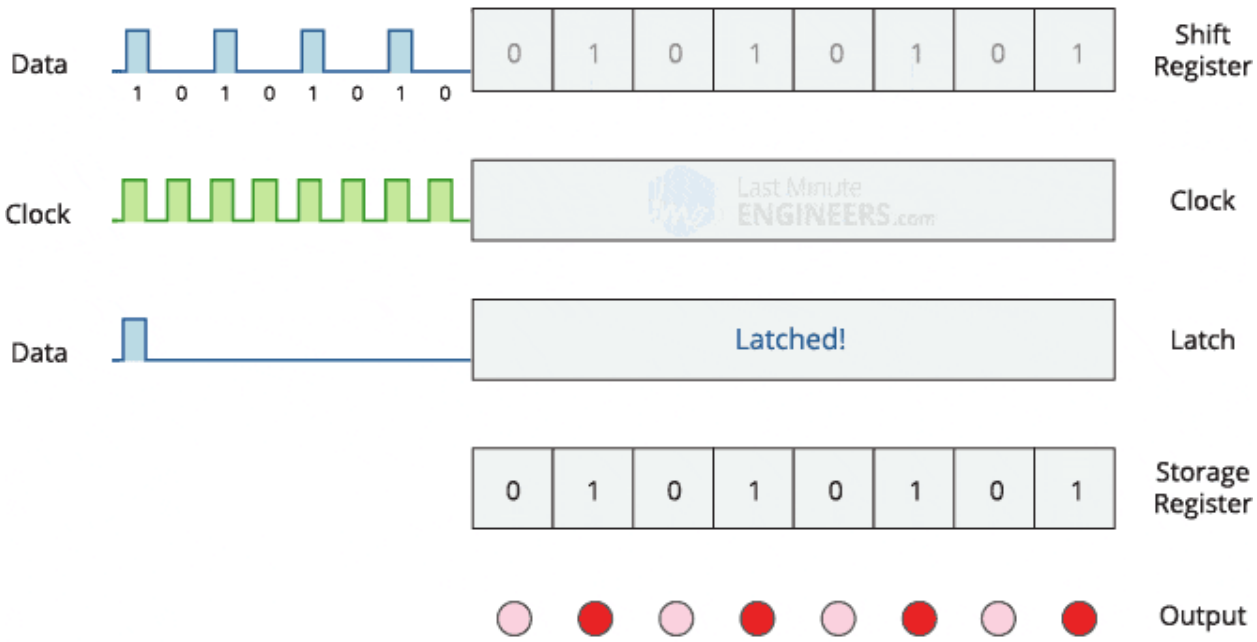
	displayed. “--” is displayed, and flashes on and off in succession four times.	countdown. Pushbuttons will be unresponsive until flashing is complete and the timer goes to initial state.



[1]



[1] **Figure 1**: Top image (Increment action as state diagram)
Figure 2: Bottom image (Logic for displaying numbers)
Figure 3: Timing diagram
Figure 4: Register Purposes



Register	Purpose
R16	**Holds value intended to be loaded to the left display.**
R19	**Holds the value intended to be loaded to the right display.**
R18	**Determines whether the value of R16 or R19 is loaded into R13. This can be seen in *display*.**
R17	**Value that pushes all 8 bits of R13 value into SER of the shift register.**
R13	**Contains the value that *display* loads into SER of the shift register.**
R20	**Holds the decimal value equivalent to the number represented by both displays.**

R22	**Counts the amount of 10ms sampled collected by r-calling *delay_long*. (Ex: R22 = = 20 means 2 seconds have been recorded)**
R24	**Holds the number four and decrements so that *flash* runs 4 times. It is reset when it re-enters *main*.**

4. Conclusion

This lab provided experience working with I/O registers and problem solving how to output lots of data using shift registers when it is not practical or not possible to directly output all of that data from the ports of the microcontroller. The lab also provided experience with planning and implementing both software and hardware components in incremental steps. “Software design” practices specific to assembly were learned as well, as operations basic to higher level languages (if statements, loops, function calls) must be implemented creatively and efficiently on a case by case basis. Overall, the lab provided a helpful framework for problem solving at the intersection of software and hardware, and with a variety of components.

5. Appendix A: Source Code

File 1 (main.asm):

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Assembly Language file (1/2) for Lab 2 in ECE:3360
; Spring 2023, The University of Iowa
; Created: 2/4/2023 10:56:48 AM
; Author : Max Finch, Tiger Slowinski
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


; put code here to configure I/O lines ; as output &
.include "m328Pdef.inc"
.cseg
.org 0


; Configure I/O lines.
sbi   DDRB,0      ; PB0 is now output (SER)
sbi   DDRB,1      ; PB1 is now output (SRCLK)
sbi   DDRB,2      ; PB2 is now output (RCLK)
cbi   DDRB,3      ; PB3 is now input (BUTTON)
cbi   DDRB,4      ; PB4 is now input (BUTTON)

;R18 trigger bit
;R19 left bit
;R16 right bit

;set up limit
ldi R20, 0x00

ldi R24, 4
;set up 00
ldi R16, 0x3F ; right bit
ldi R19, 0x3F ; left bit


main:
ldi R24, 4
ldi R18, 0
rcall display ; call display subroutine
ldi R18, 1
rcall display

rcall delay_long

sbis PINB,3
rjmp stay_put

sbis PINB,4
rjmp stay_put_B

rjmp main

switch_left: ;Left digit
    ;If 0 make 1
    cpi R16, 0x3F
    breq become_left_1
    ;If 1 make 2
    cpi R16, 0x06
    breq become_left_2

become_left_1:
    ldi R16, 0x06
    rjmp main
become_left_2:
    ldi R16, 0x5B
    rjmp main

flash:

    ldi R20, 0
    cpi R24, 0
    breq main_set
    rcall flashing
    dec R24
    rjmp flash

main_set:
    ldi R16, 0x3F
    ldi R19, 0x3F
    rcall display_nums
    rjmp main


switch_number:

    ;increment register limit
    inc R20

    ;If 0 make 1
    cpi R19, 0x3F
    breq become_1

    ;If 1 make 2
    cpi R19, 0x06
    breq become_2

    ;If 2 make 3
    cpi R19, 0x5B
    breq become_3

    ;If 3 make 4
    cpi R19, 0x4F
    breq become_4
```

```
103
104         ;If 4 make 5
105         cpi R19, 0x66
106         breq become_5
107
108         ;If 5 make 6
109         cpi R19, 0x6D
110         breq become_6
111
112         ;If 6 make 7
113         cpi R19, 0x7D
114         breq become_7
115
116         ;If 7 make 8
117         cpi R19, 0x07
118         breq become_8
119
120         ;If 8 make 9
121         cpi R19, 0x7F
122         breq become_9
123
124         ;If 9 make 0
125         cpi R19, 0x67
126         breq become_0
127
128
129     become_0:
130         ldi R19, 0x3F
131         rjmp switch_lef
132         rjmp main
133
134     become_1:
135         ldi R19, 0x06
136         rjmp main
137
138     become_2:
139         ldi R19, 0x5B
140         rjmp main
141
142     become_3:
143         ldi R19, 0x4F
144         rjmp main
145
146     become_4:
147         ldi R19, 0x66
148         rjmp main
149
150     become_5:
151         ldi R19, 0x6D
152         rjmp main
153
154     become_6:
155         ldi R19, 0x7D
156         rjmp main
157
158     become_7:
159         ldi R19, 0x07
160         rjmp main
161
162     become_8:
163         ldi R19, 0x7F
164         rjmp main
165
166     become_9:
167         ldi R19, 0x67
168         rjmp main
169
170     stay_put:
171         rcall delay_long      ;10ms delay
172         inc R22               ;Register for samples
173         cpi R22, 20           ; If 20 reset
174         breq set_leds_to_0    ;set leds back to 0
175         sbic PINB,3           ;If button released continue to check if 25
176         rjmp reset_call_ii25  ; If its released reset the counter and go to is_it_25
177         rjmp stay_put         ;If no button released loop stay_put
178
179     stay_put_B:
180         sbis PINB,4
181         rjmp stay_put_B
182         jmp check_25
183
184     reset_call_ii25:       ;reset the counter and go to is_it_25
185         ldi R22, 0         ;reset counter
186         rjmp is_it_25
187
188     is_it_25:              ;check if leds are at 25
189         cpi R20, 25        ;if its not 25, go to switch number
190         brne switch_number
191         sbis PINB,3        ;If button pressed, go to 2 second reset loop
192         rjmp reset_loop
193         sbic PINB,4
194         rjmp is_it_25      ;If no button pressed, infinitely loop
195         rjmp stay_put_B
196
197     reset_loop:
198         sbic PINB,3        ;If button released, go back to is_it_25
199         rjmp reset_call_ii25
200         rcall delay_long   ;10ms delay
201         inc R22            ;Register for samples
202         cpi R22, 20        ; If 20 reset
203         breq set_leds_to_0 ;set leds back to 0
204         rjmp reset_loop    ;continue to increment counter
```

```
206  set_leds_to_0:
207      ldi R16, 0x3F ; right bit
208      ldi R19, 0x3F ; left bit
209      ldi R20, 0 ;reset counter
210      ldi R22, 0 ;reset limit
211      rcall delay_extra_long_2
212      rjmp main
213
214
215  check_25:
216      ldi R21,25
217      cpse R20,R21
218      rjmp check_24
219      jmp _25
220  check_24:
221      ldi R21,24
222      cpse R20,R21
223      rjmp check_23
224      jmp _24
225  check_23:
226      ldi R21,23
227      cpse R20,R21
228      rjmp check_22
229      jmp _23
230  check_22:
231      ldi R21,22
232      cpse R20,R21
233      rjmp check_21
234      jmp _22
235  check_21:
236      ldi R21,21
237      cpse R20,R21
238      rjmp check_20
239      jmp _21
240  check_20:
241      ldi R21,20
242      cpse R20,R21
243      rjmp check_19
244      jmp _20
245  check_19:
246      ldi R21,19
247      cpse R20,R21
248      rjmp check_18
249      jmp _19
250  check_18:
251      ldi R21,18
252      cpse R20,R21
253      rjmp check_17
254      jmp _18
255  check_17:
256      ldi R21,17
257      cpse R20,R21
258      rjmp check_16
259      jmp _17
260  check_16:
261      ldi R21,16
262      cpse R20,R21
263      rjmp check_15
264      jmp _16
265  check_15:
266      ldi R21,15
267      cpse R20,R21
268      rjmp check_14
269      jmp _15
270  check_14:
271      ldi R21,14
272      cpse R20,R21
273      rjmp check_13
274      jmp _14
275  check_13:
276      ldi R21,13
277      cpse R20,R21
278      rjmp check_12
279      jmp _13
280  check_12:
281      ldi R21,12
282      cpse R20,R21
283      rjmp check_11
284      jmp _12
285  check_11:
286      ldi R21,11
287      cpse R20,R21
288      rjmp check_10
289      jmp _11
290  check_10:
291      ldi R21,10
292      cpse R20,R21
293      rjmp check_9
294      jmp _10
295  check_9:
296      ldi R21,9
297      cpse R20,R21
298      rjmp check_8
299      jmp _9
300  check_8:
301      ldi R21,8
302      cpse R20,R21
303      rjmp check_7
304      jmp _8
305  check_7:
306      ldi R21,7
307      cpse R20,R21
308      rjmp check_6
309      jmp _7
310  check_6:
311      ldi R21,6
312      cpse R20,R21
313      rjmp check_5
314      jmp _6
315  check_5:
316      ldi R21,5
317      cpse R20,R21
318      rjmp check_4
319      jmp _5
320  check_4:
321      ldi R21,4
322      cpse R20,R21
323      rjmp check_3
324      jmp _4
325  check_3:
326      ldi R21,3
327      cpse R20,R21
328      rjmp check_2
329      jmp _3
330  check_2:
331      ldi R21,2
332      cpse R20,R21
333      rjmp check_1
334      jmp _2
335  check_1:
336      ldi R21,1
337      cpse R20,R21
338      jmp flash
339      jmp _1
340
341
342
343  display_nums:
344      ldi R18,0
345      rcall display
346      ldi R18,1
347      rcall display
348      ret
349
350  _25:
351      ldi R16,0x5B
352      ldi R19,0x6D
353      call display_nums
354      rcall delay_extra_long_2
355      rjmp _24
```

```
_24:
    ldi R16,0x5B
    ldi R19,0x66
    call display_nums
    rcall delay_extra_long_2
    rjmp _23
_23:
    ldi R16,0x5B
    ldi R19,0x4F
    call display_nums
    rcall delay_extra_long_2
    rjmp _22
_22:
    ldi R16,0x5B
    ldi R19,0x5B
    call display_nums
    rcall delay_extra_long_2
    rjmp _21
_21:
    ldi R16,0x5B
    ldi R19,0x06
    call display_nums
    rcall delay_extra_long_2
    rjmp _20
_20:
    ldi R16,0x5B
    ldi R19,0x3F
    call display_nums
    rcall delay_extra_long_2
    rjmp _19
_19:
    ldi R16,0x06
    ldi R19,0x67
    call display_nums
    rcall delay_extra_long_2
    rjmp _18
_18:
    ldi R16,0x06
    ldi R19,0x7F
    call display_nums
    rcall delay_extra_long_2
    rjmp _17
_17:
    ldi R16,0x06
    ldi R19,0x07
    call display_nums
    rcall delay_extra_long_2
    rjmp _16
_16:
    ldi R16,0x06
    ldi R19,0x7D
    call display_nums
    rcall delay_extra_long_2
    rjmp _15
_15:
    ldi R16,0x06
    ldi R19,0x6D
    call display_nums
    rcall delay_extra_long_2
    rjmp _14
_14:
    ldi R16,0x06
    ldi R19,0x66
    call display_nums
    rcall delay_extra_long_2
    rjmp _13
_13:
    ldi R16,0x06
    ldi R19,0x4F
    call display_nums
    rcall delay_extra_long_2
    rjmp _12
_12:
    ldi R16,0x06
    ldi R19,0x5B
    call display_nums
    rcall delay_extra_long_2
    rjmp _11
_11:
    ldi R16,0x06
    ldi R19,0x06
    call display_nums
    rcall delay_extra_long_2
    rjmp _10
_10:
    ldi R16,0x06
    ldi R19,0x3F
    call display_nums
    rcall delay_extra_long_2
    rjmp _9
_9:
    ldi R16,0x3F
    ldi R19,0x67
    call display_nums
    rcall delay_extra_long_2
    rjmp _8
_8:
    ldi R16,0x3F
    ldi R19,0x7F
    call display_nums
    rcall delay_extra_long_2
    rjmp _7
_7:
    ldi R16,0x3F
    ldi R19,0x07
    call display_nums
    rcall delay_extra_long_2
    rjmp  6
_6:
    ldi R16,0x3F
    ldi R19,0x7D
    call display_nums
    rcall delay_extra_long_2
    rjmp _5
_5:
    ldi R16,0x3F
    ldi R19,0x6D
    call display_nums
    rcall delay_extra_long_2
    rjmp _4
_4:
    ldi R16,0x3F
    ldi R19,0x66
    call display_nums
    rcall delay_extra_long_2
    rjmp _3
_3:
    ldi R16,0x3F
    ldi R19,0x4F
    call display_nums
    rcall delay_extra_long_2
    rjmp _2
_2:
    ldi R16,0x3F
    ldi R19,0x5B
    call display_nums
    rcall delay_extra_long_2
    rjmp _1
_1:
    ldi R16,0x3F
    ldi R19,0x06
    call display_nums
    rcall delay_extra_long_2
    rjmp _0
_0:
    ldi R16,0x3F
    ldi R19,0x3F
    call display_nums
    rcall delay_extra_long_2
    rjmp flash


.include "AsmFile1.asm"
.exit
```

File 2 (AsmFile1.asm):

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Assembly Language file (1/2) for Lab 2 in ECE:3360
; Spring 2023, The University of Iowa
; Created: 2/4/2023 10:56:48 AM
; Author : Max Finch, Tiger Slowinski
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


.equ count1 = 60000             ; assign a 16-bit value to symbol "count"
delay_extra_long:
    ldi r30, low(count)     ; r31:r30  <-- load a 16-bit value into counter register for outer loop
    ldi r31, high(count);
d11:
    ldi   r29, 255              ; r29 <-- load a 8-bit value into counter register for inner loop
    d22:
        nop                                         ; no operation
        dec   r29                   ; r29 <-- r29 - 1
        brne  d22                                   ; branch to d2 if result is not "0"
    sbiw r31:r30, 1                 ; r31:r30 <-- r31:r30 - 1
    brne d11                                    ; branch to d1 if result is not "0"
ret                                         ; return

flashing:

    ldi R16,0x00
    ldi R19,0x00
    call display_nums
    rcall delay_extra_long

    ldi R16,0x40
    ldi R19,0x40
    call display_nums
    rcall delay_extra_long

    ret

.equ count = 8502               ; assign a 16-bit value to symbol "count"

delay_long:
    ldi r30, low(count)     ; r31:r30  <-- load a 16-bit value into counter register for outer loop
    ldi r31, high(count);
d1:
    ldi   r29, 31               ; r29 <-- load a 8-bit value into counter register for inner loop
    d2:
        nop                                         ; no operation
        dec   r29                   ; r29 <-- r29 - 1
        brne  d2                                    ; branch to d2 if result is not "0"
    sbiw r31:r30, 1                 ; r31:r30 <-- r31:r30 - 1
    brne d1                                 ; branch to d1 if result is not "0"
ret                                         ; return
```

```
52  display: ; backup used registers on stack
53      sbrs R18, 0
54      mov R13, R19
55      sbrc R18, 0
56      mov R13, R16
57      ;Change all R16 below to R13
58
59      push R13
60      push R17
61      in R17, SREG
62      push R17
63      ldi R17, 8 ; loop --> test all 8 bits
64  loop:
65      rol R13 ; rotate left trough Carry
66      BRCS set_ser_in_1 ; branch if Carry is set
67      ; put code here to set SER to 0
68      cbi PORTB,0
69      rjmp end
70  set_ser_in_1:
71      ; put code here to set SER to 1...
72      sbi PORTB,0
73  end:
74      ; put code here to generate SRCLK pulse...
75      rcall pulse_clock
76      dec R17
77      brne loop
78      ; put code here to generate RCLK pulse
79      rcall pulse_latch
80      ; restore registers from stack
81      pop R17
82      out SREG, R17
83      pop R17
84      pop R13
85      ret
86
87
88  delay_extra_long_2:
89      rcall delay_long
90      rcall delay_long
91      rcall delay_long
92      rcall delay_long
93      rcall delay_long
94      rcall delay_long
95      rcall delay_long
96      rcall delay_long
97      rcall delay_long
98      rcall delay_long
99      rcall delay_long
100     rcall delay_long
101     rcall delay_long
102     rcall delay_long
103     rcall delay_long
104     ret
```

```
106 pulse_clock:
107     sbi   PORTB,1
108     cbi   PORTB,1
109 ret
110
111 pulse_latch:
112     sbi   PORTB,2
113     cbi   PORTB,2
114 ret
```

6. Appendix B: References

Atmel Corporation. *AVR Instruction Set Manual - Microchip Technology*. <https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-0856-AVR-Instruction-Set-Manual.pdf>.

Arduino. *Arduino UNO Rev3 with Long Pins.*<https://docs.arduino.cc/retired/boards/arduino-uno-rev3-with-long-pins>

Last Minute Engineers. “In-Depth: How 74HC595 Shift Register Works & Interface with Arduino.” *Last Minute Engineers*, Last Minute Engineers, 12 Feb. 2023, <https://lastminuteengineers.com/74hc595-shift-register-arduino-tutorial/>.

ES23_Lab02.pdf, provided by Professor Beichel