Shahin Saeidi

ECE9047/9407B

Student ID: 251324723

Statement of the issue:
In this project we are going to create a three-digit decimal counter that accurately counts 1s intervals. The count should be displayed as a decimal number on the 7-segment display. An interrupt-enabled push button should be used to reverse the direction of the counter. The count should roll over to zero after reaching 999 (if counting up), or vice versa if counting down.

Explanation of the solution:

Display:
We use register r0 to store our numbers that we want to show on our seven segments.

```
        @r0 for input of Seven Segment
        cmp r0, #0
        moveq r0, #63 @Decimal to Binary converter-0b00111111
        cmp r0, #1
        moveq r0, #6 @Decimal to Binary converter-0b00000110
        bxeq lr
        cmp r0, #2
        moveq r0, #91 @Decimal to Binary converter-0b01011011
        cmp r0, #3
        moveq r0, #79 @Decimal to Binary converter-0b01001111
        cmp r0, #4
        moveq r0, #102 @Decimal to Binary converter-0b01100110
        cmp r0, #5
        moveq r0, #109 @Decimal to Binary converter-0b01010101
        cmp r0, #6
        moveq r0, #125 @Decimal to Binary converter-0b01110110
        cmp r0, #7
        moveq r0, #7 @Decimal to Binary converter-0b00000111
        cmp r0, #8
        moveq r0, #127 @Decimal to Binary converter-0b01111111
        cmp r0, #9
        moveq r0, #111 @Decimal to Binary converter-0b01101111
        bx lr
```

Timer:
We use register r6, r7 and r8 for our timer:

```
timer:
        push {r6, r7, r8}
        b timer_loop
timer_loop:
        @ get the current count
        str r6, [r4, #16]
        @ now get the low count
   ldr r7, [r4, #16]
        @ get the high count
   ldr r8, [r4, #20]
        add r7, r8, lsl #16
        cmp r7, #0
   beq timer_end
   b timer_loop
timer_end:
        pop {r8, r7, r6}
        bx lr
```

Main loop:
In this part of the code, we first read our data at 0x3000. Then we add r0 with r1 to display our number on the 7-segment display that I set up for the ones place. We repeat the same process with r1 and r2 to display our numbers on the seven-segment displays that I set up for the tens and hundreds places.

```
main_loop:
        ldr r0, =0x3000 @ read the data in 0x3000
        ldr r7, [r0] @ store it in r7
        mov r0, r1
        bl display @ for display, r0 as the input
        add r8, r0
        mov r0, r2
        bl display @ for display
        add r8, r0, lsl  #8
        mov r0, r3
        bl display @ for display
        add r8, r0, lsl #16
        str r8, [r5] @ write data to 7-segment
```

Up counter:

The register used as the flag bit is r7. If r7 is 1, we know that our code is working as an up counter.

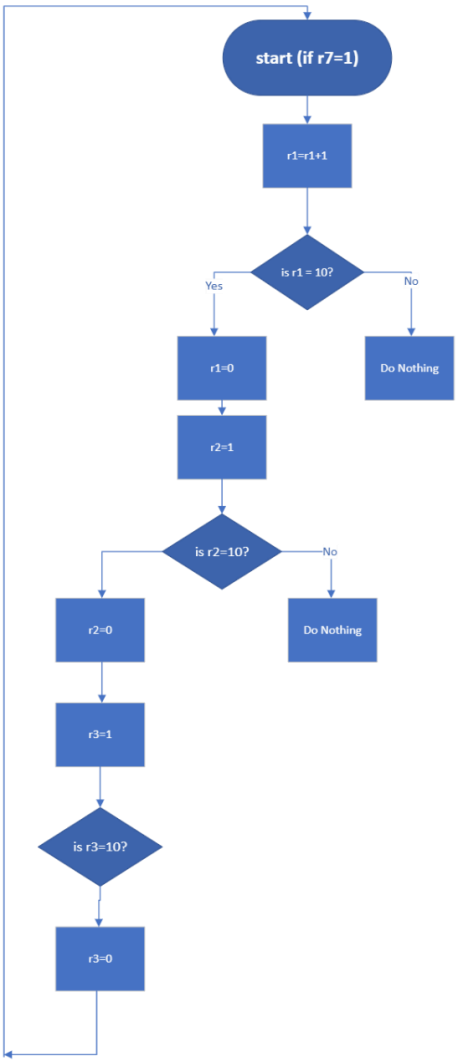
$if\ r1 = 10, \quad then: r1 = 0\ \&\ r2 = 1$

$else:\ do\ nothing$

$if\ r2 = 10, \quad then: r2 = 0\ \&\ r3 = 1$

$else: do\ nothing$

$if\ r3 = 10, then: r3 = 0$




```
add_loop:
        add r1, #1 @ increase counter
```

```
@ if r1>10, then r1=0, r2++
cmp r1, #10
moveq r1, #0
addeq r2, #1
@ if r2>10, then r2=0, r3++
cmp r2, #10
moveq r2, #0
addeq r3, #1
cmp r3, #10
moveq r3, #0
b main_loop
```

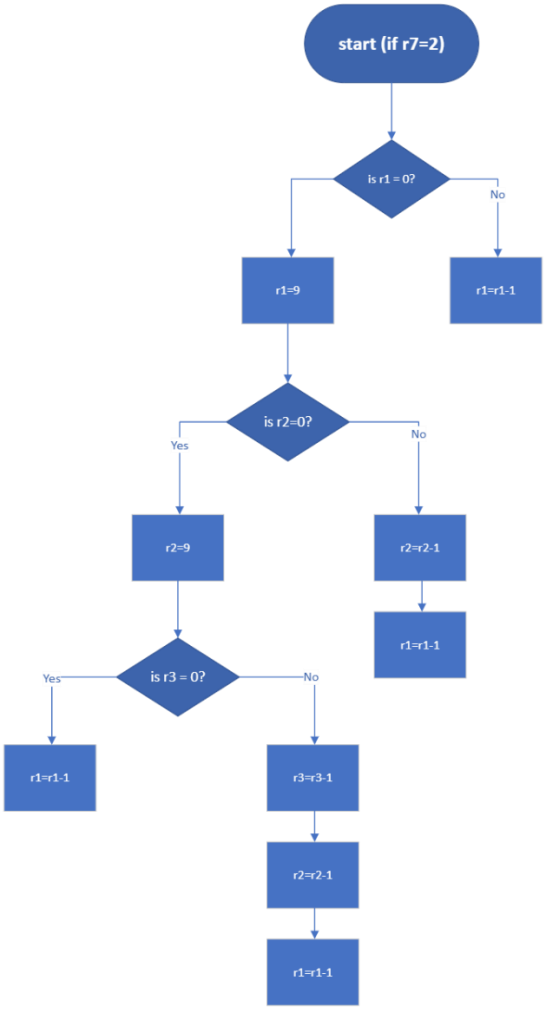
Down counter:

The register used as the flag bit is r7. If r7 is 2, we know that our code is working as a down counter.

$$\{ \quad if\ r1 \neq 0 \rightarrow \text{then } r1 = r1 - 1\ (\text{sub } r1, \#1)$$

$$\begin{cases} if\ r1 = 0\ \&\ r2 \neq 0 \rightarrow r1 = 10\ (mov\ r1, \#10) \\ \qquad\qquad\qquad\qquad \rightarrow r2 = r2 - 1\ (\text{sub } r2, \#1) \end{cases}$$

$$\begin{cases} if\ r1, r2 = 0\ \&\ r3 \neq 0 \rightarrow r1 = 10, \\ \qquad\qquad\qquad\qquad \rightarrow r2 = 9\ (mov\ r2, \#9) \\ \qquad\qquad\qquad\qquad \rightarrow r3 = r3 - 1\ (sub\ r3, \#1) \end{cases}$$




```
subtract_loop:
        @ if r1=0, then r1=10, r2--
```

```
        cmp r1, #0
        bne subtract_end
first:
        moveq r1, #10
        cmp r2,#0
        subne r2, #1
        bne subtract_end
second:
        @ if r2=0, then r2=9, r3--
        moveq r2, #9
        cmp r3, #0
        subne r3, #1
        moveq r3, #9
        b subtract_end
subtract_end:
        sub r1, #1 @ decrease counter-change it r--
        b main_loop
```

Push buttons:

I used this piece of code to enable the use of push buttons to change the counting direction. If one of the buttons is pressed once, the direction changes. It is already known that the address 0xFF200050 is reserved for push buttons.

```
.global KEY_ISR
KEY_ISR:
        @ if any key is pressed, read the data in 0x2000
        MOV     R0, #0x3000
        LDR     R1, [R0]
        @ if data in 0x3000 = 1, then wirte 2 to 0x3000
        @ if data in 0x3000 = 2, then wirte 1 to 0x3000
        cmp     R1,#1
        moveq   r2, #2
        movne   r2, #1
        STR     R2, [R0]
        LDR     R0, =0xFF200050  // base address of pushbutton KEY port
        LDR     R1, [R0, #0xC]   // read edge capture register
        MOV     R2, #0xF
        STR     R2, [R0, #0xC]   // clear the interrupt

END_KEY_ISR:
        BX LR
```

Going through some issues:

I began writing my code by starting with the counter part. Firstly, I wrote the code for a single 7-segment display and used it to display numbers ranging from 0 to 9. After that, I modified the code to display three numbers on the 7-segment display, which represent the hundreds, tens, and ones places. Following this, I proceeded to write the code for the down counter, which starts from 999 and counts downwards. However, I noticed that the numbers on the 7-segment display crash when the counter reaches 000 for the down counter and 999 for the up counter. To address this issue, I added an if loop for each register (r1, r2, r3).

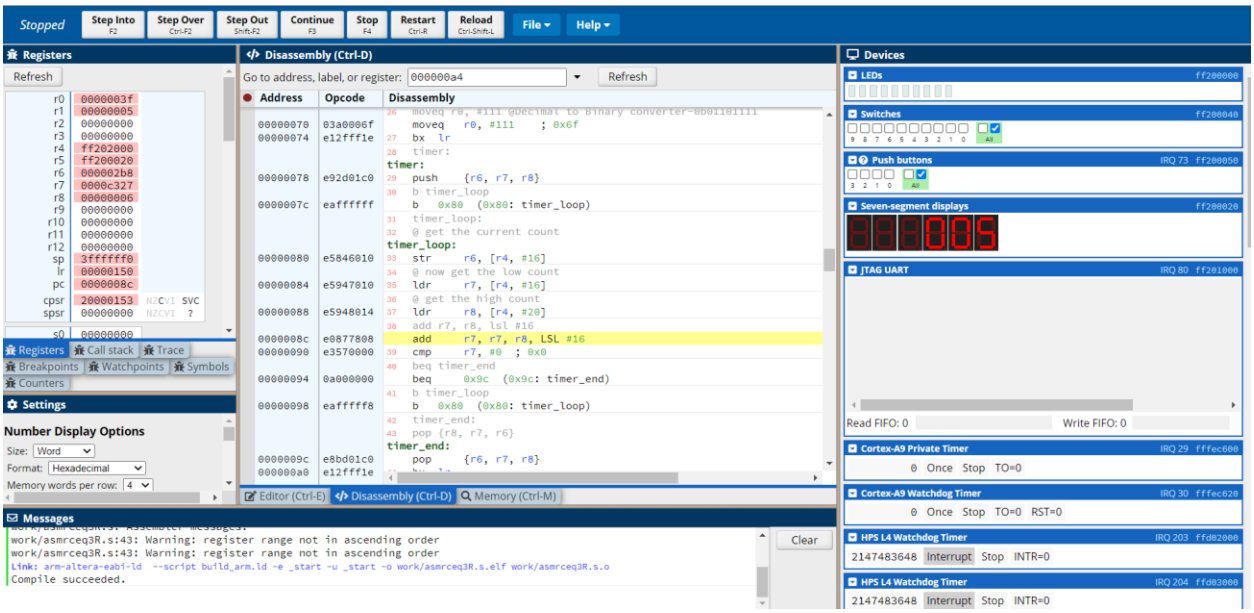
Then I combined these two codes with a flag bit (r7). In the if loop, I explained that if r7 equals 1, it should branch to the add_loop, which is the loop for the up counter. If r7 equals 2, it should branch to the subtract_loop. However, I encountered an issue where if the number went down to 000 in the down counter mode, it would display 8899 instead of 999. To solve this, I added another if statement to compare r3 with 0.

Similarly, there was a problem in the up counter mode where when it reaches 999, the 7-segment display for the hundreds place shows an odd number. I solved this issue by adding a line to compare r3 with 10.

Qualitative cost/benefit:

I spent 2 hours writing the code for the up counter part and about 4 hours for the down counter part. To combine these two parts and debug the problems, I spent approximately 8 hours designing the flag bit to switch between the add_loop and subtract_loop. Additionally, I spent approximately 6 hours adding the interrupt part to my code.

Regarding the report, I spent approximately 4 hours thoroughly covering all the work I had done so far and ensuring the coherence and grammar were correct.


Stopped
Step Into F2
Step Over Ctrl-F2
Step Out Shift-F2
Continue F3
Stop F4
Restart Ctrl-R
Reload Ctrl-Shift-L
File
Help
Registers
Refresh
r0 0000003f
r1 00000005
r2 00000000
r3 00000000
r4 ff202000
r5 ff200020
r6 000002b8
r7 0000c327
r8 00000006
r9 00000000
r10 00000000
r11 00000000
r12 00000000
sp 3ffffff0
lr 00000150
pc 0000008c
cpsr 20000153 NZCVI SVC
spsr 00000000 NZCVI ?
Registers
Call stack
Trace
Breakpoints
Watchpoints
Symbols
Counters
Settings
Number Display Options
Size: Word
Format: Hexadecimal
Memory words per row: 4
Disassembly (Ctrl-D)
Go to address, label, or register: 000000a4
Refresh
Address
Opcode
Disassembly
00000070 03a0006f moveq r0, #111 ; 0x6f
00000074 e12fff1e bx lr
timer:
00000078 e92d01c0 push {r6, r7, r8}
b timer_loop
0000007c eafffffd b 0x80 (0x80: timer_loop)
timer_loop:
@ get the current count
timer_loop:
00000080 e5846010 str r6, [r4, #16]
@ now get the low count
00000084 e5947010 ldr r7, [r4, #16]
@ get the high count
00000088 e5948014 ldr r8, [r4, #20]
add r7, r8, lsl #16
0000008c e0877808 add r7, r7, r8, LSL #16
00000090 e3570000 cmp r7, #0 ; 0x0
beq timer_end
00000094 0a000000 beq 0x9c (0x9c: timer_end)
b timer_loop
00000098 eafffff8 b 0x80 (0x80: timer_loop)
timer_end:
pop {r8, r7, r6}
timer_end:
0000009c e8bd01c0 pop {r6, r7, r8}
000000a0 e12fff1e
Editor (Ctrl-E)
Disassembly (Ctrl-D)
Memory (Ctrl-M)
Messages
Clear
work/asmrceq3R.s:43: Warning: register range not in ascending order
work/asmrceq3R.s:43: Warning: register range not in ascending order
Link: arm-altera-eabi-ld --script build_arm.ld -e _start -u _start -o work/asmrceq3R.s.elf work/asmrceq3R.s.o
Compile succeeded.
Devices
LEDs ff200000
Switches ff200040
Push buttons IRQ 73 ff200050
Seven-segment displays ff200020
JTAG UART IRQ 80 ff201000
Read FIFO: 0
Write FIFO: 0
Cortex-A9 Private Timer IRQ 29 fffec600
0 Once Stop TO=0
Cortex-A9 Watchdog Timer IRQ 30 fffec620
0 Once Stop TO=0 RST=0
HPS L4 Watchdog Timer IRQ 203 ffd02000
2147483648 Interrupt Stop INTR=0
HPS L4 Watchdog Timer IRQ 204 ffd03000
2147483648 Interrupt Stop INTR=0