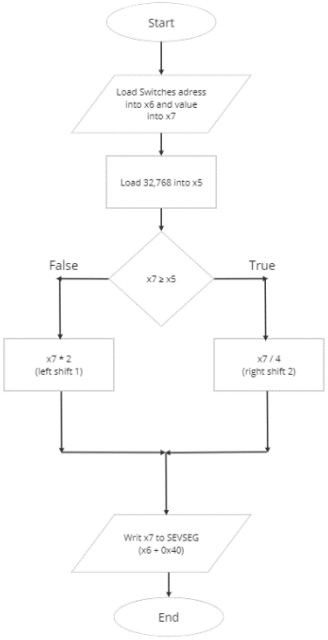
CPE 233 SW 2

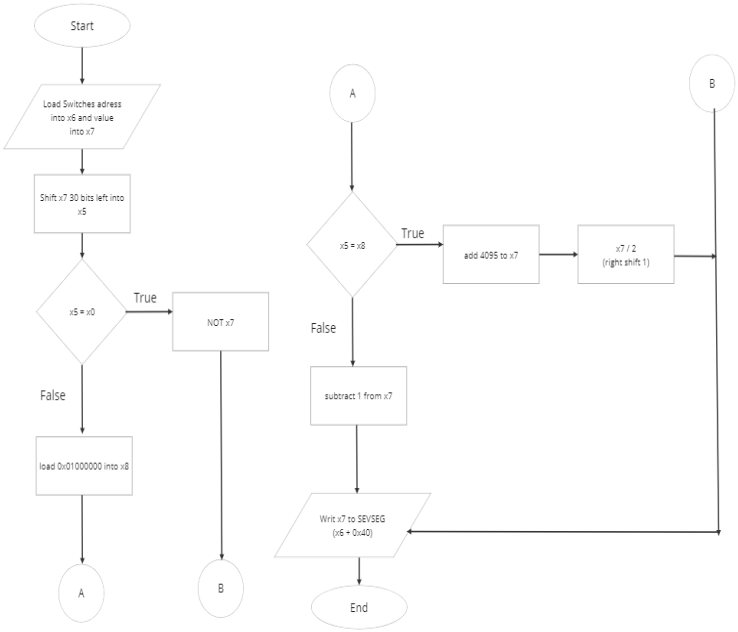
Wyatt Tack

1. Flowcharts :

Part 1 :



Part 2 :



2. Table 1: Simulation Table Part 1:

Reasoning	Switches	SevSeg Memory
Tests middle case, tests equal to in ≥	**32768 0x00008000**	**8192** 0x00002000
Tests high max for overflow	**4294967295 0xffffffff**	**1073741823** 0x3fffffff
Tests 0 (lower bound)	**0 0x00000000**	**0** 0x00000000
Tests standard for less than	**30000 0x00007530**	**60000** 0x0000ea60
Tests standard for greater than	**40000 0x00009c40**	**10000** 0x00002710

Table 2: Simulation Table Part 2:

Reasoning	Switches	SevSeg Memory
Tests number divisible by 4	**2048** **0x00000800**	**4294965247** 0xfffff7ff
Tests odd number	**31923** **0x00007cb3**	**18009** 0x00004659
Tests number neither divisible by 4 or odd	**218** **0x000000DA**	**217** 0x000000D9
Tests 0 case for underflow (not a multiple of 4)	**0** **0x00000000**	**-1 in RC** 0xffffffff
Tests max overflow	**4294967295** **0xffffffff**	**2047** 0x000007ff

3. Figure 1: Assembly Code Part 1:

```
      lui x6, 0x11000 #set x6 as value for Switches memory address
      lw x7, 0(x6) #fill x7 with value in switches
      li x5, 32768 #set x5 as temp value to compare switches to
      bgeu x7, x5, TRUE #conditional statement
      slli x7, x7, 1 #if false, x7*2 (shift left by 1 to *2)
      j END #skip past true statement
TRUE: srli x7, x7, 2 #if true, shift right 2
END:  sw x7, 0x40(x6) #load into sevseg
```

Figure 2: Assembly Code Part 2:

```
            lui x6, 0x11000 #set x6 as value for Switches memory adress
            lw x7, 0(x6) #fill x7 with value in switches
            beq x7, x0, SUB #zero case special, not /4 so default
            slli x5, x7, 30 #set top 2 digits of x5 to 1's and 2's place
            beq x5, x0, FOURTHS #if divisible by 4, 1's and 2's both =0
            slli x5, x5, 1 #if not /4, set 1's place to MSB to compare
            bne x5, x0, ODD #if odd then not all bits of x5 are 0
SUB:        addi x5, x0, 1 #if not odd, set x5 to 1 and subtract from x7
            sub x7, x7, x5
            j END
FOURTHS:    not x7, x7 #invert x7
            j END
ODD:        addi x7, x7, 2000 #add 4095 and divide by 2
            addi x7, x7, 2000 #must split 4095 into 2 too large for imm
            addi x7, x7, 95
            srli x7, x7, 1
END:        sw x7, 0x40(x6) #load into sevseg
```