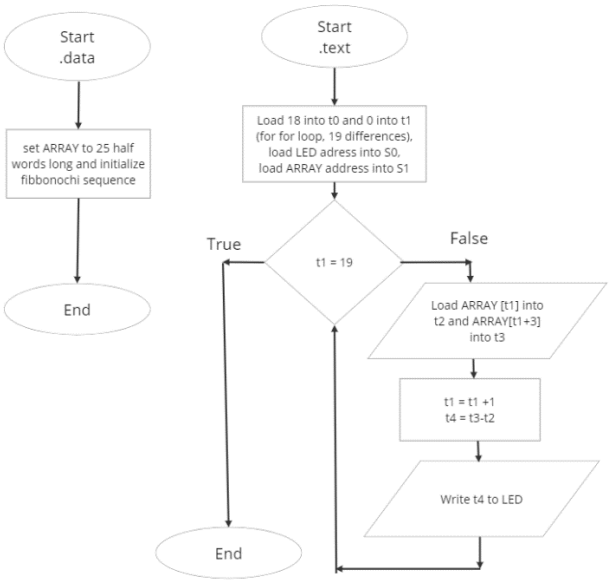
CPE 233 SW 4

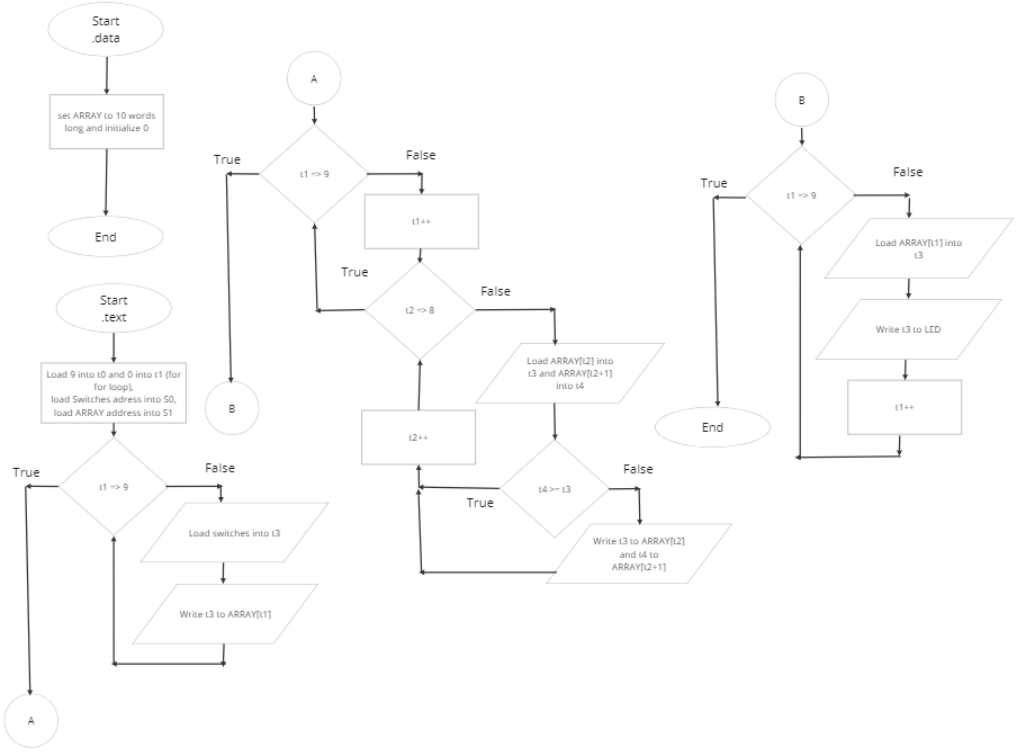
Wyatt Tack and Thomas Hong

1. Flowcharts :

   Part 1 :



Part 2 :



2. Table 1: Verification Part 1:

No data input, program should run consistently every time…			
Calculated Differences		**Output Differences**	
2	**288**	**2**	**288**
2	**466**	**2**	**466**
4	**754**	**4**	**754**
6	**1220**	**6**	**1220**
10	**1974**	**10**	**1974**
16	**3194**	**16**	**3194**
26	**5168**	**26**	**5168**
42	**8362**	**42**	**8362**
68	**13530**	**68**	**13530**
110	**21892**	**110**	**21892**
178	**35422**	**178**	**35422**

Table 2: Verification Part 2:

Reasoning	Switches (First to last)	LEDS (First to last)
Tests standard 10-1 oriented in opposite direction, tests if enough loops will be needed for maximum number of loops	0x0000_000a 0x0000_0009 0x0000_0008 0x0000_0007 0x0000_0006 0x0000_0005 0x0000_0004 0x0000_0003 0x0000_0002 0x0000_0001	0x0000_0001 0x0000_0002 0x0000_0003 0x0000_0004 0x0000_0005 0x0000_0006 0x0000_0007 0x0000_0008 0x0000_0009 0x0000_000a
Tests maximum for unsigned equating, along with values in middle to test all around working	0x1010_a0a0 0xFFFF_FFFF 0x2020_b0b0 0x3030_c0c0 0x1234_abcd 0x4040_d0d0 0xabcd_1234 0x5050_e0e0 0x0000_0000 0x6060_f0f0	0x0000_0000 0x1010_a0a0 0x1234_abcd 0x2020_b0b0 0x3030_c0c0 0x4040_d0d0 0x5050_e0e0 0x6060_f0f0 0xabcd_1234 0xffff_ffff

3. Figure 1: Assembly Code Part 1:

```
.data
#initialize the sequence, each number is 2 bytes away from eachother
ARRAY:  .half 0, 1, 1, 2, 3, 5
        .half 8, 13, 21, 34, 55
        .half 89,144, 233, 377, 610
        .half 987, 1597, 2584, 4181
        .half 6765, 10946, 17711, 28657
        .half 46368
.text
        li t0, 42               #t0 for for loop comparison
        li t1, 0                #t1 for for loop incriment
        li s0, 0x11000020       #addresses saved
        la s1, ARRAY
COUNT:  bge t1, t0, END         #for loop initialize (all for loop variables are *2
                                #due to each ARRAY index being 2 bytes from eachother

        add t1, t1, s1          #load ARRAY[t1] into t2
        lhu t2, (t1)            #load ARRAY[t1+3] into t3
        lhu t3, 6(t1)

        sub t4, t3, t2          #take difference 3 numbers apart

        sub t1, t1, s1          #reverse t1+ARRAY_ADRESS
        addi t1, t1, 2          #increment t1 for count

        sw t4, (s0)             #store difference to LEDs
        j COUNT
END:    #end
```

Figure 2: Assembly Code Part 2:

```
.data
ARRAY:    .space 40          #40 bytes = 10 words saved

.text
          li t0, 40          #t0 for for loop comparison
          li t1, 0           #t1 for for loop incriment
          li s0, 0x11000000  #addresses saved
          la s1, ARRAY
LOAD_IN:  bge t1, t0, LOADED #for loop initialize (all for loop variables are *4
                                    #due to each ARRAY index being 4 bytes from eachother

          lw t4, 0(s0)               #load switches into t4
          add t1, t1, s1
          sw t4, (t1)                #store t4 into ARRAY[t1]
          sub t1, t1, s1
          addi t1, t1, 4             #increment t1 4 (for 4 bytes over)
          j LOAD_IN
LOADED:   #finished loading values, start sorting

          li t0, 10          #t0 for compare and t1 for i++ in main loop
          li t1, 0

SORTING:  bge t1, t0, SORTED #t1 is i++ for main loop (10x)
          li t2, 36          #t2 for compare and t3 for j+4 in sub loop
          li t3, 0

COMP:     bge t3, t2, COMPEND#t3 is j+4 for sub loop (9x)

          add t3, t3, s1
          lw t4, (t3)                #load ARRAY[t3] to t4
          lw t5, 4(t3)               #load ARRAY[t3+1] to t5

          bgeu t5, t4 NOSWAP #if ARRAY[t3]>ARRAY[t3+1] swap values
          sw t5, (t3)                #UNSIGNED
          sw t4, 4(t3)
NOSWAP:
          sub t3, t3, s1             #return t3 to normal count variable
          addi t3, t3, 4             #increment t3
          j COMP
COMPEND:  #finished sub loop, branch to main loop
          addi t1, t1, 1             #increment t1
          j SORTING
SORTED:   #finished sorting, on to outputting

          li t0, 40          #t0 for compare and t1 for i++ in loop
          li t1, 0

LEDLOOP:  bge t1, t0 END

          add t1, t1, s1
          lw t3, (t1)                #load ARRAY[t1] into t3
          sw t3, 0x20(s0)            #store t3 into LEDs
          sub t1, t1, s1
          addi t1, t1, 4             #increment t1
          j LEDLOOP
END:      #end
```