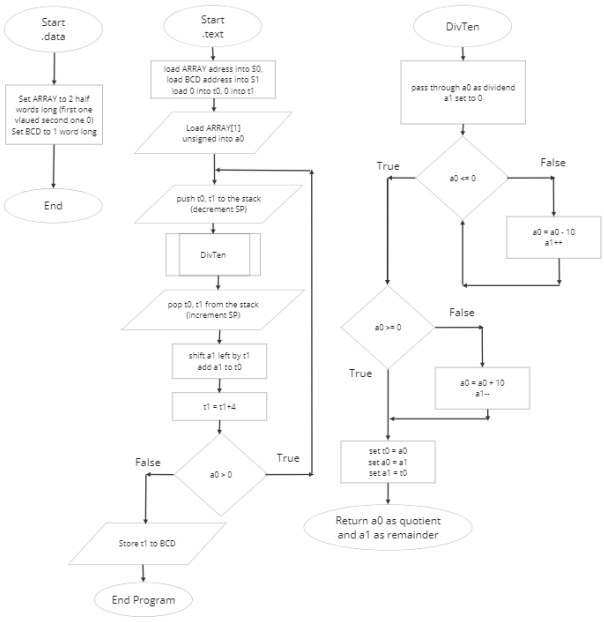
CPE 233 SW 6

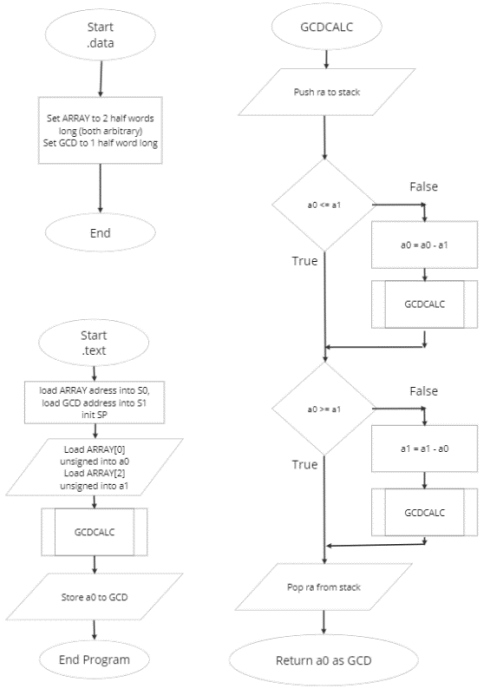
Wyatt Tack and Ethan Vosburg

1. Flowcharts :

   Part 1 :



Part 2 :



2. Table 1: Verification Part 1:

ARRAY[1]	BCD CODED	Reasoning
65535 0xffff	**0x00065535**	Tests high end of half word values
4 0x0004	**0x00000004**	Tests numbers below 16 (0x10)
0 0x0000	**0x00000000**	Tests 0
12345 0x3039	**0x00012345**	Standard test for middle of spectrum
70000 0x1_1170 (turned into 0x1170 = 4464)	**0x00004464**	Tests overflow in .data (assembler automatically truncates data past 2nd byte)

Table 2: Verification Part 2:

CONDITION: a0/a1 < 10238 && a1/a0 < 10238		
ARRAY	**GCD**	**Reasoning**
a0:60000 0x3a60 a1:3005 0x0bbd	5 0x0005	Tests standard operation with a0 > a1
a0: 65535 0xffff a1: 0 0x0000	Stack overflow error	Tests stack overflow and divide by 0
a0:3920 0x0f50 a1:5920 0x1720	80 0x0050	Tests standard operation with a0 < a1
a0:65320 0xff50 a1:63264 0xf720	8 0x0008	Tests standard operation with high numbers

3. Figure 1: Assembly Code Part 1:

```
.data
ARRAY:  .half 256 #number you want to be BCD coded
BCD:    .word 0
.text
        la s0, ARRAY    #load mem addresses
        la, s1, BCD
        li sp, 0x10000 #Stack Pointer
        li t0, 0        #set BCD total to 0
        li t1, 0        #start shift value to 0
        lhu a0, (s0)    #load array into a0
BCODED:
        addi sp, sp, -8 #push t0 and t1 to stack
        sw t0, (sp)
        sw t1, 4(sp)
        call DIVTEN     #divide a0 by 10, return a0 quotient, a1 remainder
        lw t0, (sp)     #pop t0 and t1 from stack
        lw t1, 4(sp)
        addi sp, sp, 8
        sll a1, a1, t1 #shift a1 by nibble count
        add t0, a1, t0 #add a1 to t0 --> add to BCD val
        addi t1, t1, 4 #increment nibble count
        bgt a0, x0 BCODED #loop until whole number is coded
        sw t0, (s1) #set BCD coded number into bcd
END:    j END
#DIVTEN Function (a0/10 = a0, a1 remainder)
DIVTEN: addi a1, x0, 0 #set a1 to 0
SUB:    ble a0, x0 DIVBY        #count how many times
        addi a0, a0, -10        #a0 can be subtracted by 10
        addi a1, a1, 1
        j SUB
DIVBY:  bge a0, x0 QTNT#if remainder, remove 1 from sub
        addi a0, a0, 10#count and send to remainder
        addi a1, a1, -1
QTNT:   addi t0, a0, 0 #swap a1 and a0 so
        addi a0, a1, 0 #quotient can be a0
        addi a1, t0, 0
        ret
```

Figure 2: Assembly Code Part 2:

```
.data
ARRAY:  .half 4         #const 1
        .half 16        #const 2
GCD:    .half
#GCDCALC can only run 10238 iterations, a0/a1 < 10238 && a1/a0 < 10238
.text

        la s0, ARRAY    #load mem addresses
        la, s1, GCD
        li sp, 0x10000 #Stack Pointer
        lhu a0, (s0)    #load ARRAY into a0 and a1
        lhu a1, 2(s0)
        call GCDCALC    #calculate GCD of a0 and a1, return to a0
        sw a0, (s1)     #store GCD to GCD array
END:    j END           #terminate program

#GCD Function: a0 = GCD(a0, a1)
GCDCALC:
        addi sp, sp, -4 #push ra to stack
        sw ra, (sp)

        ble a0, a1, LT #if a0 > a1,
        sub a0, a0, a1 #a0 = a0-a1 and loop
        call GCDCALC
LT:     bge a0, a1, GE #if a0 < a1,
        sub a1, a1, a0 #a1 = a1-a0 and loop
        call GCDCALC
GE:     lw ra, (sp)     #pop ra from stack
        addi sp, sp, 4
        ret
```