```
Bubble sort:
.data
n: .word 10  //設定大小
str1: .string "Array: " //宣告string1
str2: .string "Sorted: "//宣告string2
space: .string " "//宣告space
array: .word 5, 3, 6, 7, 31, 23, 43, 12, 45, 1 //宣告array
endl: .string "\n"//宣告換行的string

.text
main:
        la a1, str1      //print str1 將str1的address load 至a1
        li a0, 4                 //將a0的數值存為4
        ecall            //執行系統呼叫

        la t0, array     //將array的address load 至t0
        lw t2, n         //將n load 至t2
        jal ra, loop1    //跳到loop1並將instruction存在ra
        la t0, array     //將array的address load 至t0
        lw t2, n         //將n load 至t2
        addi t1, zero, 0        //zero+0存在t1
        addi t3, t1, -1                  //t1加(-1) 存在t3
        addi s5, zero, 1        // zero+1 存在s5
        bge t2, s5, bubble   //t2大於或等於s5時 跳到bubble
         li      a0, 10   //將a0的數值存為10
        ecall            //程式終止

bubble:
        bge  t1, t2, exit        //t2大於或等於t1時 跳到bubble
        addi t3, t1, -1                  //t1加(-1) 存在t3

bubble2:
        blt t3, zero, exit2      //t3小於zero時 跳到exit2
        addi s2, zero, 4        //zero+4存在s2
        mul s2 ,s2, t3                   //s2*t3存在s2
        add s2, t0, s2                   //s2+t0 存在s2
        lw s3, 0(s2)             //s2(0)的值load到s3
        lw s4, 4(s2)             //s2的(4)值load到s4
        ble s3, s4, exit2        //s3如果小於s4跳到exit2
        j swap                   //跳到swap
        addi t3, t3 ,-1                  //t3加(-1)存在t3
        j bubble2                //跳到bubble2

exit2:
```

```
        addi t1, t1, 1          //t1+1存到t1
        j bubble                //跳到bubble

exit: //結果
        la a1, str2             //print str2 將str2的address load 至a1
        li a0, 4                //將a0的數值存為4
        ecall                   //執行系統呼叫
        jal ra, loop1           //跳到loop1並將instruction存在ra
        li  a0, 10              //將a0的數值存為10
        ecall                   //程式終止

swap: //交換
        addi s2, zero, 4        //zero+4存在s2
        mul s2 ,s2, t3                  //s2*t3存在s2
        add s2, t0, s2                  //s2+t0 存在s2
        lw s3, 0(s2)            //s2(0)的值load到s3
        lw s4, 4(s2)            //s2(4)的值load到s4
        sw s3, 4(s2)            //s2(4)的值存到s3的位置
        sw s4, 0(s2)            //s2(0)的值存到s4的位置
        jalr x0, x1, 0          //跳到x1的instruction 將next存在x0


loop1:
        li a0, 4                        //將a0的數值存為4
        la a1, space            //print space 將space的address load 至a1
        ecall                   //執行系統呼叫

        li a0, 1                //將a0的數值存為1
        lw t1, 0(t0)            //t0(0)的值load到t1
        mv a1, t1               //將t1移到a1
        ecall

        addi t0, t0, 4          //t0加4存到t0
        addi t2, t2, -1                 //t2加(-1)存到t2
        bne t2, zero, loop1     //t2如果不等於zero 就跳到loop1
        la a1, endl             //print endl 將endl的address load 至a1 換行
        li a0, 4                //將a0的數值存為4
        ecall                   //執行程式終止
        ret                     //return
```

```
GCD:
.data                           //宣告輸入的兩個數字以及字串
N1: .word 512
N2: .word 480
str1: .string "GCD value of "
str2: .string " and "
str3: .string " is "

.text
main:
    lw      a0, N1              //將N1 load到a0
    lw      a1, N2              //將N2 load到a1
    jal     ra, gcd             //跳到gcd並將instruction存在ra

    add     a2, a0, zero        //a0加zero存在a2
    lw      a0, N1              //將N1 load到a0
    lw      a1, N2              //將N2 load到a1
    jal     ra, Result          //跳到Result並將instruction存在ra

    li      a0, 10              //將a0的數值存為10
    ecall                       //執行程式終止



gcd:
    addi    sp, sp, -24         //將sp-32後再存在sp
    sw      ra, 16(sp)          //將16(sp)存在ra
    sw      a1, 8(sp)           //將8(sp)存在a1
    sw      a0, 0(sp)           //將0(sp)存在a0
    bne     zero, a1, gcd2      //如果zero不等於a1則跳到gcd2
    addi    sp, sp, 24          //將sp+24後再存在sp
    jalr    x0, x1, 0           //跳到x1的instruction 將next存在x0

gcd2:
    rem     t0, a0, a1          //a0和a1取餘數存在t0
    add     a0, a1, zero        //將a1加zero存在a0
    add     a1, t0, zero        //將t0加zero存在a1
    bne zero, a1, gcd2          //如果zero不等於a1則跳到gcd2
    addi    sp, sp, 24          //將sp+24後再存在sp
    jalr    x0, x1, 0           //跳到x1的instruction 將next存在x0


Result:
    mv      t0, a0              //將a0移到t0
    mv      t1, a1              //將a1移到t1
```

```
    mv      t2, a2          //將a2移到t2

    la      a1, str1        //print str1 將str1的address load 至a1
    li      a0, 4           //將a0的數值存為4
    ecall

    mv      a1, t0          //將t0移到a1
    li      a0, 1           //將a0的數值存為1
    ecall

    la      a1, str2        //print str2 將str2的address load 至a1
    li      a0, 4           //將a0的數值存為4
    ecall

    mv      a1, t1          //將t1移到a1
    li      a0, 1           //將a0的數值存為1
    ecall

    la      a1, str3        //print str3 將str3的address load 至a1
    li      a0, 4           //將a0的數值存為4
    ecall

    mv      a1, t2          //將t2移到a1
    li      a0, 1           //將a0的數值存為1
    ecall

    ret                     //return


fibonacci
.data                       //宣告字串跟運行的次數
N: .word 10
str1: .string "th number in the Fibonacci sequence is "

.text
main:
    lw      a0, N           //將N load到a0
    jal     ra, fibonacci   //跳到fibonacci並將instruction存在ra

    mv      a1, t2          //將t2移到a1（列印結果）
    lw      a0, N           //將N load到a0
    jal     ra, Result      //跳到Result並將instruction存在ra（列印結果）

# Exit program
    li      a0, 10          //將a0的數值存為10
    ecall
```

```
fibonacci:
        addi    a3, zero, 1     //zero加1存到a3
        bgt     a0,a3,fib2      //如過a0大於等於a3則跳到fib2
        mv      t2,a0           //將a0移到t2
        ret
fib2:
        addi    sp,sp,-12       //將sp-12後再存在sp
        sw      ra,0(sp)        //將0(sp)的值存到ra

        sw      a0,4(sp)        //將4(sp)的值存到a0
        addi    a0,a0,-1        //a0加(-1)存到a0
        jal     fibonacci       //跳到fibonacci
        lw      a0,4(sp)        //4(sp)的值load到a0
        sw      t2,8(sp)        //將8(sp)的值存到t2

        addi    a0,a0,-2        //a0加(-2)存到a0
        jal     fibonacci       //跳到fibonacci

        lw      t0,8(sp)        //8(sp)的值load到t0
        add     t2,t0,t2        //t0+t2存到t2

        lw      ra,0(sp)        //0(sp)的值load到ra
        addi    sp,sp,12        //將sp+12後再存在sp
        ret

Result:
        mv      t0, a0          //將a0移到t0
        mv      t1, a1          //將a1移到t1

        mv      a1, t0          //將t0移到a1
        li      a0, 1           //將a0的數值存為1
        ecall

        la      a1, str1        //print str1 將str1的address load 至a1
        li      a0, 4           //將a0的數值存為4
        ecall


        mv      a1  t1          //將t1移到a1
        li      a0, 1           //將a0的數值存為1
        ecall

        ret                     //return
```

遭遇困難：

組合語言好難，花了好多時間才研究懂程式碼在說什麼。

解決方法：

詢問同學和爬文找資料。

作業心得：

因為這學期才修數位電路設計，有許多相關的知識都不太清楚，所以讀的有點吃力，希望之後會好一點。