Fibonacci Sequence:

```
.data
argument: .word 10      //argument=10
i:.word 1                    //i=1
j:.word 1                  //j=1
k:.word 2                  //k=2
str1: .string "th number in the Fibonacci sequence is "      //str1= th number in the
Fibonacci sequence is

.text
main:
        lw        a0, argument               // a0 = argument
        lw      t0,i                         // t0 = i
        lw      t1,j                         // t1 = j
        lw        t3,k                               // t3 = k
        jal       ra, fib                    // jump and link to the  'fib'  lable


          li       a0,10                     // exit program
          ecall



fib:
        addi      a0,a0,-1                           // a0 = a0 - 1
        add       t2,t0,t1                   // t2 = t0 + t1
        mv        t0,t1                      // t0 = t1
        mv        t1,t2                      // t1 = t2
        bne       a0,t3,fib                  // if ( a0 ! = t3 ), go to the  'fib'  lable
        beq       a0,t3,printResult          // if ( a0 == t3), go to the  'printResult'  lable


printResult:

        lw        a1,argument                // print argument
        li        a0,1
        ecall

        la        a1,str1                    // print str1
        li       a0,4
        ecall

        mv        a1,t2                    // print t2, t2=result
        li       a0,1
        ecall

        ret
```

Greatest Common Divisor :

```
.data
argument1: .word 512               // argument1 = 512
argument2: .word 480               // argument2 = 480
i:.word 0                          // i = 0
str1: .string "GCD value of "      // str1 = GCD value of
str2: .string " and "              // str2 = and
str3: .string " is "               // str3 = is

.text
main:
lw a0, argument1                   // a0 = argument1
lw a1,argument2                    // a1 = argument2
lw t0,i                            // t0 = i
jal ra, gcd                        // jump and link to 'gcd' label


li a0,10                           // exit program
ecall



gcd:
rem t1,a0,a1                       // t1 = a0 % a1
mv a0,a1                           // a0 = a1
mv a1,t1                           // a1 = t1
beq t1,t0,printResult              // if ( t1 == t0 ), go to printResult
```

```
bne t1,t0,gcd                   // if ( t1 ! = t0 ), go to  ‘gcd’ label

printResult:
                                //因為 print 需要用到 a0,a1，所以需要先將 a0,a1
                                裡面存的東西暫存到其他暫存器
lw t2,argument1                 // t2 = argument1
lw t3,argument2                 // t3 = argument2
mv t4,a0                        // t4 = a0

la a1,str1                      // print str1
li a0,4
ecall

mv a1,t2                        // print t2
li a0,1
ecall

la a1,str2                      // print str2
li a0,4
ecall

mv a1,t3                        // print t3
li a0,1
ecall

la a1,str3                      // print str3
li a0,4
ecall

mv a1,t4                        // print t4
li a0,1
ecall

ret
```

Bubble sort :

```
.data
n:.word 10                              //n =110
str1:.string "Array: "                  // str1 = Array
str2:.string "Sorted: "                 // str2 = Sorted
space:.string " "
endl:.string "\n"                       // endl = newline
array:.word 5,3,6,7,31,23,43,12,45,1    //初始化 array

.text
main:
lw a5,n                                 // a5 = n
la s5, array                            //s5 指到 array 一開始的位置
jal ra,initialArray                     // jump and link to ‘initialArray’ label
la a1,endl                              // 換行
li a0,4
ecall
la a1,str2                              // print str2
li a0,4
ecall
la a1,endl                              // 換行
li a0,4
ecall
lw a5,n                                 // a5 = n
jal ra,sort                             // jump and link to ‘sort’ label

#exit program
li a0, 10                               // exit program
ecall
ret

initialArray:
la a1,str1                              // print str1
li a0,4
ecall
la a1,endl                              //換行
li a0,4
ecall
```

```
beq zero,zero,print0            //  印出 sort  之前的 array
ret

swap:
mv s6,s4                        // s6 = s4
slli s6,s6,2                    //為了留 4byte
add s6,s5,s6                    //指向 array[j]的位置
lw t4,0(s6)                     // t4 = array[j]
lw t5,4(s6)                     // t5 = array[j+1]
sw t5,0(s6)                     // array[j] = t5
sw t4,4(s6)                     // array[j+1] = t4
ret                             //return

sort:
li s3,0                         // i=0
beq zero,zero,loop1             //跳到 loop1

loop1:
bge s3,a5,print0                // if ( i > n ), go to  ‘print0’  label
addi s4,s3,,-1                  // s4 = s3 -1, j = i - 1
beq zero,zero,loop2             //跳到  loop2
loop2:
blt s4,zero,exit2 #if j<0       // if ( j < 0 ), go to  ‘exit2’  label
slli t0,s4,2                    // s4 = j , t0 = j + 1
add t0,s5,t0                    // t0  指向 array[j]的位置
lw t1,0(t0)                     // t1 = a[j]
lw t2,4(t0)                     // t2 = a[j+1]
ble t1,t2,exit2                 // if a[j]<a[j+1], go to  ‘exit2’  label
mv t3,ra                        //先將 ra 暫存，以免 call swap 時 ra 被改到
jal ra,swap                     // jump and link to  ‘swap’  label
mv ra,t3                        //將 ra 的值移回來
addi s4,s4,-1                   // s4 = s4  –  1, j--
beq zero,zero,loop2             //跳到 loop2
print0:
mv s7,zero                      // s7 = 0, Jo6 為 index k
beq zero,zero,print             //  印出 array  中的 element
print:
add s8,s5,s7                    // s7=k, s8 指到目前要印出的 element
```

```
lw a1,0(s8)                 // 將目前的 element 放到 a1 後印出
li a0,1
ecall
la a1,space                 //印出 element 與 element 之間的空白
li a0,4
ecall
addi s7,s7,4                //為了 k++
addi a5,a5,-1               //計算還有多少個 element 沒印出來
bne a5,zero,print           //如果還沒印完就繼續印
ret                         // return
exit2:
addi s3,s3,1                // s3 = s3 + 1, i++
beq zero,zero,loop1         //回到 loop1
```