Fibonacci sequence

```
.data
argument: .word 10
str: .string “th number in the Fibonacci sequence is ” # 宣告

.text
main:
                        # initial reg t0 = 0, reg t1 = 1
                        li      t0, 0 # fib(0)
                        li      t1, 1 # fib(1)

                        lw      t3, argument  # reg t3 = 10

                        # prints int 10
                        li      a0, 1
                        mv      a1, t3
                        ecall

                        # fibonnaci loop
fib:
                        beq     t3, zero, End # if reg t3 == 0 go to label End
                        add     t2, t1, t0       # fib(x) = fib(x-2) + fib(x-1)
                        # 更新數值
                        mv      t0, t1 # reg t0 = t1
                        mv      t1, t2 # reg t1 = t2
                        addi    t3, t3, -1 # reg t3--
                        j fib # loop, go back to fib

End:
                        # prints str
                        la      a1, str
                        li      a0, 4
                        ecall

                        # prints the result in reg t0
                        li a0, 1
                        mv a1, t0
                        ecall

                        # ends the program
                        li a0, 10
                        ecall
```

GCD

```
.data
argument1: .word 512 #宣告
argument2: .word 480
str1: .string "GCD value of "
str2: .string " and "
str3: .string " is "

.text
main:
                    lw      t0, argument1 # reg t0 = m
                    lw      t1, argument2 # reg t1 = n
                    jal     ra, gcd  # jump to label gcd

                     # ends the program
                    li      a0, 10
                    ecall
gcd:
                    beq     t1, zero, printResult # if n == 0 go to label printResult

                    rem     t3, t0, t1 # reg t3 = r , r = m%n
                    mv      t0, t1 # m = n
                    mv      t1, t3 # n = r
                    j       gcd # loop
printResult:
                    # print str1
                    la      a1, str1
                    li      a0, 4
                    ecall
                    # print int 512
                    lw      a1, argument1
                    li      a0, 1
                    ecall
                    # print str2
                    la      a1, str2
                    li      a0, 4
                    ecall
                    # print int 480
                    lw      a1, argument2
                    li      a0, 1
                    ecall
                    # print str3
                    la      a1, str3
                    li      a0, 4
                    ecall
                    # print gcd in t0
                    mv      a1, t0
                    li      a0, 1
                    ecall
                    # return to main function
                    ret
```

###bubble_sort###

Bubble Sort

```
.data
argument: .word 10 # 宣告
arr: .word 5, 3, 6, 7, 31, 23, 43, 12, 45, 1
str1: .string "Array: "
str2: .string "Sorted: "
space: .string " "
new_line: .string "\n"

.text
main:
                addi    sp, sp, -64 # make room on stack
                sw      ra, 60(sp) # save ra on stack
                sw      s0, 56(sp) # save s0 on stack
                addi    s0, sp, 64 # let s0 = sp + 64

                la      t0, arr # load arr on t0
                addi    t2, s0, -48 # set t2 = s0 -48
                addi    t3, zero, 0 # set t3 = 0
                lw      t4, argument # t4 = 10
load:
                bge     t3, t4, conti # if t3 >= 10, go to label conti
                # load elements on arr to a0
                lw      t1, 0(t0)
                mv      a0, t1
                sw      a0, 0(t2) # store a0 to stack

                addi    t0, t0, 4 # i = i + 1
                addi    t2, t2, 4 # renew location on stack
                addi    t3, t3, 1 # counter++
                j       load
conti:
                # print str1
                la      a1, str1
                li      a0, 4
                ecall
                la      a1, new_line
                li      a0, 4
                ecall

                addi    t2, s0, -48
                addi    t3, zero, 0
                jal     ra, printArray

                mv      a0, zero # inti i
                sw      a0, -52(s0)
```

```
                    # sorting
outerloop:
                    lw      a0, -52(s0)   # load i
                    # reg a1 = 9
                    lw      a1, argument
                    addi     a1, a1, -1
                    blt     a1, a0, exit1 # go to exit1 if 9 < i

                    addi    a0, a0, -1  # j = i - 1
                    sw      a0, -56(s0) # save j in location s0 – 56 pointing

inner_loop:
                    lw      a0, -56(s0) # j
                    mv      a1, zero
                    blt     a0, a1, exit2 # if j < 0 go to label exit2

                    # if(data[j] > data[j+1])
                    lw      a0, -56(s0) # j
                    slli    a0, a0, 2
                    addi    a1, s0, -48
                    add     a0, a0, a1
                    lw      a1, 0(a0) # reg a1 = data[j]
                    lw      a0, 4(a0) # reg a0 = data[j+1]
                    bge     a0, a1, update_j # if(data[j+1] >= data[j]), go to update_j

                    # swap function
                    lw      a0, -56(s0) # j
                    slli    a0, a0, 2
                    addi    a1, s0, -48
                    add     t1, a0, a1 # reg t1 = data[j]

                    lw      t0, 0(t1) # reg t0 (temp) = data[j]
                    lw      t2, 4(t1) # reg t2 = data[j+1]
                    sw      t2, 0(t1) # data[j] = data[j+1]
                    sw      t0, 4(t1) # data[j+1] = temp

update_j:
                    lw      a0, -56(s0)  # j
                    addi    a0, a0, -1   # j--
                    sw      a0, -56(s0) # save j
                    j       inner_loop   # jump to label inner_loop

exit2:

                    lw      a0, -52(s0)  # i
                    addi    a0, a0, 1    # i++
                    sw      a0, -52(s0) # save i
                    j       outerloop    # jump to label outerloop
```

```
exit1:
                    # print str2 and print "\n"
                    la      a1, str2
                    li      a0, 4
                    ecall
                    la      a1, new_line
                    li      a0, 4
                    ecall

                     # initial reg t2 = s0 – 48, which is data[0]'s location
                    addi    t2, s0, -48
                    addi    t3, zero, 0 # initial reg t3 = 0
                    jal     ra, printArray # jump to label printArray

                    # End program
                    li      a0, 10
                    ecall

printArray:
                    lw      t4, argument # reg t4 = 10
                    bge     t3, t4, printNextLine # if t3 >= 10, go to printNextLine
                    lw      t0, 0(t2) # data[k]
                    # print data[k]
                    li      a0, 1
                    mv      a1, t0
                    ecall
                    # print space
                    la      a1, space
                    li      a0, 4
                    ecall

                    addi    t2, t2, 4 # 更新t2(data[k])
                    addi    t3, t3, 1 # reg t3++, as a counter
                    j       printArray # loop

printNextLine:      # print "\n"
                    la      a1, new_line
                    li      a0, 4
                    ecall

                    ret # return
```