Edgar Ruiz
CS 264
HW5
July 18, 2015

File.asm

```
.data 0x10000000
        a: .word 5      #int a = 5
        one: .word 1    #int const one = 1

.text 0x00400000
        main:   addi $t0, $zero, 30     #$t0 = int b = 30
                addi $t1, $zero, 75     #$t1 = int c = 75
                addi $sp, $sp, -4       #allocate space for int array [1] = {7}
                addi $t2, $zero, 7      #$t2 = 7
                sw $t2, 0x4($sp)        #int array [1] = {7}
                lw $t3, -0x8000($gp)#load int a = 5 -> $t3
                lw $t4, -0x7FFC($gp)#load int const one = 1 -> $t4
                add $t5, $t3, $t0       #d = a + b
                add $t5, $t5, $t1       #d = d + c
                add $t5, $t5, $t4       #d = d + one
                addi $sp, $sp, -40      # allocate space array = new int[10]
                sw $t5, 0x10($sp)       #array[4] = d
                sw $t3, 0x20($sp)       #array[8] = a
                sw $t1, 0x8($sp)        #array[2] = c
                add $a0, $t0, $zero     #store b -> $a0
                add $a1, $t1, $zero     #store c -> $a1
                jal delta               #delta(b, c)
                add $t3, $v0, $zero     #a = delta(b, c)
                addi $t3, $t3 , 15      #a = a +15
                blez $t3, else          #if a <= 0, do else
                lw $t6, 0x?($gp)        #load glob
                addi $t3, $t6, 1        #a = 1 + glob
                j for                   #jump to for loop
        else:   add $a0, $t0, 0$zero    #store b -> $a0
                jal factorial           #factorial(b)
                add $t1, $v0, $zero     #c = factorial(b)
        for:    add $t7, $zero, $zero#int f = 0
                addi $t8, $zero, 4      #4 used in f < 4
        loop:   bge $t7, $t8, nxtfor    #if f >= 4, exit for loop
                addi $t3, $t3, 1        #a = a + 1
                add $t5, $t0, $t1       #d = b + c
                addi $t7, $t7, 1        #f = f +1
                j loop                  #jump back to beginning of loop
        nxtfor: add $a0, $t3, $zero     #store a -> $a0
                jal mult                #mult(a)
                add $t5, $v0, $zero     #d = mult(a)
                addi $v0, $v0, 10       #$v0 = 10
                syscall                 #syscall code 10 to exit
        factorial:add $s0, $a0, $zero   #$s0 = n
                addi $t9, $zero, 1      #1 used for n < 1
                bge $s0, $t9, felse     #if n >= 1, do felse
                add $v0, $v0, 1         #return 1
                jr $ra                  #jump to return address
        felse:  addi $sp, $sp, -8       #allocate spade for $s0 & $ra
                sw $ra, 0($sp)          #store $ra
                sw $s0, 4($sp)          #store $s0
                addi $s1, $s0, -1       #n-1
                add $a0, $s1, $zero     #store n-1 -> $a0
                jal factorial           #factorial(n-1)
                mul $v0, $v0, $s0       #return (n*factorial(n-1))
                lw $ra, 0($sp)          #restore stack
                lw $s0, 4($sp)          #restore stack
                addi $sp, $sp, 8        #restore stack
                jr $ra                  #jump to return address
```

```
delta.asm
.data 0x10000000
    glob: .word 5               #int glob = 5

.text 0x00400000
    delta:  bge $a0, $a1, delse #b -> $a0, c -> $a1 in main before jal delta
            sub $v0, $a1, $a0   # return b-a
            jr $ra              # jump to return address
    delse:  sub $v0, $a0, $a1   #return a-b
            jr $ra              #jump to return address
```

```
library.asm
.data 0x10000000

.text 0x00400000
    mult:   mul $v0, $a0, $a0   #return a*a
            jr $ra              #jump to return address
```