L3:

```
.section .data
.section .text
.globl _start
_start:
addi $s0, $s0, 0 # variavel a=0
addi $s1, $s1, 0 # variavel i=0
addi $s2, $s2, 15 # variavel k = 15 (valor qualquer)
while:
bge $s1, $s2, done # pular para rotulo done caso condição i < k seja falsa
add $s0, $s0, $s1
addi $s1, $s1, 1
j while
done:
```

Strcopy:

```
.data
        str1: .space 200
        str2: .space 200
        str3: .asciiz "A string copiada foi: "
.text
.globl _start
_start:
        li $v0, 8 #chamada de sistema para ler string
        la $a0,str1 #endereço de onde ira guardar a string lida
        addi $a1, $0, 200 #ler no maximo 200 characteres
        syscall
        la $a1,str2 # string a ser escrita
        ## str1(lida) em $a0 e str2(a ser escrita) em $a1
        jal Strcopy #jump and link, chama procedimento e salva em $ra pc+4
        la $a0, str3 # endereço da string em $a0
        addi $v0, $0, 4 # 4 em $v0 para imprimir na tela
        syscall
        la $a0, str2 # endereço da string em $a0
        addi $v0, $0, 4 # 4 em $v0 para imprimir na tela
        syscall
        li $v0, 10
        syscall
        Strcopy:
                li $t0, 0
                lb $s0, 0($a0) #pegar primeiro byte da palavra origem para ver se n é \0
                while:
                        #beqz $a0,fim_while
                        beq $s0, $0,fim_while
                        lb $s0, 0($a0)
                        sb $s0, 0($a1)
                        addi $a0, $a0, 1
                        addi $a1, $a1, 1
                        j while
                fim_while:
                addi $a1, $a1, 1 #incrementa o endereço da palavra destino
                sb $0, 0($a1) #coloca o \0 no final dessa palavra
```

```
        jr $ra #registrador com endereço de retorno
```

Ex: 2-a)

```
.data
    X: .word 0:2048 #aloca 2048*4 bytes para 2048 valores inteiros
    Y: .word 0:64 #aloca 64*4 bytes para 64 valores inteiros
.text
    li $s0,0 # $s0 = A = 0
    li $s1,1 # $s1 = I = 1
    while:
        bge $s1,2048,fim_while
        mul $t2,$s1,4 #$t2 = i * 4 (quando i mudar uma unidade, $t2 muda 4, para se
adequar ao tamanho da palavra no mips)
        lw $t0, X($t2) # $t0 = x[i]
        add $s0, $s0, $t0 # a = a + $t0
        rem $t0,$s1,64 # $t0 = I%64 (mod)
        mul $t0,$t0,4
        lw $t1, Y($t0) #$t1 = Y[$t0]
        rem $t3,$t1,2048 #$t3 = $t1%2048 (mod)
        mul $t4,$t3,$4
        lw $t3, X($t4)
        add $s0, $s0, $t3 # a = a + $t3
        sll $s1, $s1, 1 #(shift left logical) i = i*2
        j while
    fim_while:
    li $v0,10
    syscall
```

Ex: 2-c)

```
.data
.text
.globl main
main:
    addi $a0, $0, 2 # argument g = 2
    addi $a1, $0, 3 # argument h = 3
    addi $a2, $0, 4 # argument i = 4
    addi $a3, $0, 5 # argument j = 5
    addi $t0, $0, 3 # argument k = 3
    sub $sp, $sp, 4 #space on stack
    sw $t0, 0($sp) # save the 5th arg
    jal fun # call procedure
    addi $sp, $sp, 4 #restore old stack value
    add $s0, $v0, $0 # y = returned value
    addi $v0, $0, 10 #exit
    syscall
    fun:
        #g = $a0, h = $a1, i = $a2, j = $a3, k = 0($sp)
        add $t0, $a0, $a1 # $t0 = g + h
        add $t1, $a2, $a3 # $t1 = i + j
        mult $t1, $t4 # (i + j)*k
        mflo $t1 # $t1 = (i + j)*k
        sub $t2, $t0,  $t1 # $t2 = (g + h) - (i + j)*k
```

```
        addi $t3, $0, 4 # $t3 = 4
        mult $t2, $t3 # f*4
        mflo $v0 # return = f*4
        jr $ra # return to caller
```

Ex: 2-d)

```
.data
myArray: .word 0:256
size: .word 256
.text
.globl main
main:
la $a0,myArray
lw $a1, size
jal clear1
addi $v0, $0, 10 #exit
syscall
clear1:
    li $t0,0 #variavel i
    for:
        bge $t0,$a1,fim_for #sai do for se i >= size
        mul $t1,$t0,4  #multiplica I por 4 para se adequar ao tamanho da palavra no mips
        sw $0, myArray($t1)  # myArray[i] = 0
        addi $t0, $t0, 1 #i++
        j for #volta ao inicio do for
        fim_for: #saida do for
    jr $ra #return to caller
```