Laiba Fatima Khan (Lk04067)

**CA Lab 04 Report**

**Task 1**

addi x10,x0,12 #argument a set

addi x11, x0, 15 #argument b set

jal x1, sum

add x11,x0,x10 #sum result retrieved and stored as print argument

addi a0,x0,1 #print int ecall

ecall

beq x0,x0, Exit

sum:

add x10,x10,x11 #sum is stored in x10

jalr x0, 0(x1)

Exit:

**Task 2**

#initialize i in x22

add x22, x0, x0

#initialize j in x23

add x23, x0, x0

#initialize temp in x5

add x5,x0,x0

#store base memory address of array "a" (4-byte integers array)

addi x10, x0, 0x200

##stores len (needed to control loop)

addi x11, x0, 10

Loop:

slli x20, x22, 2 # saves i\*4 since 4bytes

add x20, x20, x10 #adress of a[i]

sw x22, 0(x20) #a[i]=i

add x21, x0, x22 #temp storage of a[i] value in register x2

beq x22, x11, EXIT #if i=10 loop exit

addi x22, x22, 1 #i=i+1

blt x22, x11,Loop

EXIT:

add x22,x0,x0 #i=0

jal x1, bubble

bubble:

beq x11,x0,return

beq x10,x0,return #if a=null return

Loop2outer:

slli x20, x22, 2 # saves i\*4 since 4bytes

add x20, x20, x10 #adress of a[i]

lw x9, 0(x20) #load a[i] in x9

add x23,x0, x22 #j=i

beq x0,x0,Loop2inner

Loop2inner:

slli x19, x23, 2 # saves j\*4 since 4bytes

add x19, x19, x10 #adress of a[j]

lw x8, 0(x19) #load a[j] in x8

lw x9,0(x20)

blt x9, x8, swapij

back:

addi x23, x23, 1 #j=j+1

blt x23, x11,Loop2inner

beq x22, x11, EXIT2 #if i=10 loop exit

addi x22, x22, 1 #i=i+1

blt x22, x11,Loop2outer

swapij:

add x5, x0, x9

sw x8, 0(x20)

sw x5, 0(x19)

beq x0,x0,back

EXIT2:

return:

jalr x1,0(x1)

**Task 3**

addi x10,x0,5

ntri:

addi sp, sp, -8 #adjust stack for 2 items

sw x1, 4(sp) #save the return address

sw x10, 0(sp) #save the argument n

addi x5, x10, -1 #x5 = n - 1

bltu x0, x5, L1 #if (n - 1) >= 0, go to L1

addi x10, x0, 1 # return 1

addi sp, sp, 8 #pop 2 items off stack

jalr x0, 0(x1) #return to caller

L1: addi x10, x10, -1 # n >= 1: argument gets (n − 1)

jal x1, ntri # call fact with (n − 1)

addi x6, x10, 0 #return from jal: move result of fact(n - 1) to x6:

lw x10, 0(sp) #restore argument n

lw x1, 4(sp) #restore the return address

addi sp, sp, 8 #adjust stack pointer to pop 2 items

add x10, x10, x6 #return n + ntri (n − 1)

jalr x0, 0(x1) #return to the caller