Course : Computer Organization – ENCM 369

Lab # : Lab 3

Student Name : Nimna Wijedasa

Lab Section : B04

EX A

```
Assemble: assembling
/Users/nimnawijedasa/Desktop/Winter/ENCM_369/lab3/encm369w23lab03/exA/bad-
align.asm
Assemble: operation completed successfully.
Go: running bad-align.asm
Error in /Users/nimnawijedasa/Desktop/Winter/ENCM_369/lab3/encm369w23lab03/exA/bad-
align.asm line 12: Runtime exception at 0x00400010: Load address not aligned to word
boundary 0x10010002
Go: execution terminated with errors.
```

```
Assemble: assembling
/Users/nimnawijedasa/Desktop/Winter/ENCM_369/lab3/encm369w23lab03/exA/null-ptr.asm
Assemble: operation completed successfully.
Go: running null-ptr.asm
Error in /Users/nimnawijedasa/Desktop/Winter/ENCM_369/lab3/encm369w23lab03/exA/null-
ptr.asm line 16: Runtime exception at 0x00400004: address out of range 0x00000000
Go: execution terminated with errors.
```

```
Assemble: assembling
/Users/nimnawijedasa/Desktop/Winter/ENCM_369/lab3/encm369w23lab03/exA/write-to-
text.asm
Assemble: operation completed successfully.
Go: running write-to-text.asm
Error in
/Users/nimnawijedasa/Desktop/Winter/ENCM_369/lab3/encm369w23lab03/exA/write-to-
text.asm line 10: Runtime exception at 0x00400008: Cannot write directly to text
segment!0x00400000
Go: execution terminated with errors.
```

EX C

```
# stub1.asm
# ENCM 369 Winter 2023
# This program has complete start-up and clean-up code, and a "stub"
# main function.

# BEGINNING of start-up & clean-up code.  Do NOT edit this code.
        .data
exit_msg_1:
        .asciz   "***About to exit. main returned "
exit_msg_2:
        .asciz   ".***\n"
main_rv:
        .word   0

        .text
        # adjust sp, then call main
        andi     sp, sp, -32                 # round sp down to multiple of 32
        jal      main

        # when main is done, print its return value, then halt the program
        sw       a0, main_rv, t0
        la       a0, exit_msg_1
        li    a7, 4
        ecall
        lw       a0, main_rv
        li       a7, 1
        ecall
        la       a0, exit_msg_2
        li       a7, 4
        ecall
     lw     a0, main_rv
        addi     a7, zero, 93     # call for program exit with exit status that is in a0
        ecall
# END of start-up & clean-up code.

# Below is the stub for main. Edit it to give main the desired behaviour.

        .data
        .globl train
train:  .word  0x20000
```

```
        .text
        .globl   main
main:
        li         s0,0xa000        # boat = 0xa000
        li         s1,0x3000        # plane = 0x3000
        la         s2,train         # s2 = train
        lw         s3,(s2)          # train = 0x20000

        addi       sp,sp,-8
        sw      ra,0(sp)  # save return value
        sw         s3,4(sp)

        addi       a1,a1,6          # first
        addi       a2,a2,4          # second
        addi       a3,a3,3          # third
        addi       a4,a4,2          # fourth
        jal        proc_A           # call proc A
        add        s0,s0,a0         # boat + procA return
        sub        t1,s0,s1         # temp = boat - plane


        lw         s3,4(sp)


        add        s3,s3,t1         # train = train + temp

        lw      ra,0(sp)   # return value
        addi       sp,sp,8

        li      a0, 0        # return value from main = 0
        jr         ra

proc_A:
        addi    sp, sp, -28         # extend stack pointer
        sw      ra, 0(sp) # save return value
        sw      a1, 4(sp)# save first
        sw         a2, 8(sp)        # save second
        sw         a3, 12(sp)       # save third
        sw         a4, 16(sp)       # save fourth
        sw      s0, 20(sp)          # save boat
        sw      s1, 24(sp)          # save plane

        lw         s0, 4(sp)        # s0 = first
        lw         s1, 8(sp)        # s1 = second
        lw      s2, 12(sp)      # s2 = third
        lw      s3, 16(sp)      # s3 = fourth
        add     s4,zero,zero    # s4 = alpha
        add     s5,zero,zero    # s5 = beta
        add     s6,zero,zero    # s6 = gamma

        mv      a1,s3           # a1 = fourth
        mv      a2,s2           # a2 = third

        jal     procB           # call procB
        mv      s5,a0           # beta = return from procB

        mv      a1,s1           # a1 = second
        mv      a2,s0           # a2 = first

        jal     procB           # call procB
        mv      s4,a0           # gamma = return value

        mv      a1,s2           # a1 = third
        mv      a2,s3           # a2 = fourth

        jal     procB           # call procB
        mv      s6,a0           # alpha = return value

        add     t1,s4,s5        # add temp = alpha + beta
        add     a0,t1,s6        # add return value a0 = temp + gamma

        lw    ra, 0(sp)  # reset return value
        lw    a1, 4(sp) # reset first
        lw      a2, 8(sp)       # reset second
        lw      a3, 12(sp)      # reset third
        lw      a4, 16(sp)      # reset fourth
        lw    s0, 20(sp)        # reset boat
        lw    s1, 24(sp)        # reset plane
        addi   sp, sp, 28       # reset stack pointer
        jr      ra              #

procB:
        slli    t1,a1,8         # first arg * 256
        add     a0,t1,a2        # value of previous + second arg
        jr      ra
```

EX E

```
# stub1.asm
# ENCM 369 Winter 2023
# This program has complete start-up and clean-up code, and a "stub"
# main function.

# BEGINNING of start-up & clean-up code.  Do NOT edit this code.
        .data
exit_msg_1:
        .asciz   "***About to exit. main returned "
exit_msg_2:
        .asciz   ".***\n"
main_rv:
        .word  0

        .text
        # adjust sp, then call main
        andi     sp, sp, -32                   # round sp down to multiple of 32
        jal      main

        # when main is done, print its return value, then halt the program
        sw       a0, main_rv, t0
        la       a0, exit_msg_1
        li    a7, 4
        ecall
        lw       a0, main_rv
        li       a7, 1
        ecall
        la       a0, exit_msg_2
        li       a7, 4
        ecall
     lw     a0, main_rv
        addi     a7, zero, 93     # call for program exit with exit status that is in a0
        ecall
# END of start-up & clean-up code.

#
#
#
#
#
# Below is the stub for main. Edit it to give main the desired behaviour.
```

```
        .data
        .global aaa
aaa:    .word 11,11,3,-11 #array aaa
        .global bbb
bbb:    .word 200,-300,400,500 #array bbb
        .global ccc
ccc:    .word -2,-3,2,1,2,3 #array ccc
        .text
        .global main

special_sum:
# s0 = bound
# s1 = x
# s2 = n
# s3 = result
# s4 = i
        addi sp, sp, -24
        sw ra, 20(sp)
        sw s4,16(sp)# i
        sw s3,12(sp)# result
        sw s2, 8(sp)# n
        sw s1, 4(sp)# x
        sw s0,0(sp) #bound
        mv s0,a0
        mv s1,a1
        mv s2,a2
        li s3,0
        li s4,0 # i = 0
        j loop
loop:

        slli t0,s4,2
        add t1,s1,t0 #&x[i]
        lw a1,(t1) # x[i]
        mv a0,s0 # a0 = bound
        jal clamp #jump to clamp
        add s3,s3,a0 #store argument 0 to result
        addi s4,s4,1     #i += 1
        blt s4,s2,loop   #if i is bigger than n go to loop label
        mv a0,s3         # move
        lw s0,0(sp)
        lw s1,4(sp)
        lw s2,8(sp)
        lw s3,12(sp)
        lw s4,16(sp)
        lw ra,20(sp)
        addi sp,sp,24
        jr ra

clamp:
        sub t0,zero,a0
        blt a1,t0,returnsub
        bgt a1,a0,returnpos
        mv a0,a1
        jr ra
returnsub:
        mv a0,t0
        jr ra

returnpos:
        jr ra



#red = s0
#green = s1
#blue = s2

main:
        addi sp,sp,-16 #allocate 5 words
        sw ra,12(sp)
        sw s2, 8(sp)
        sw s1, 4(sp)
        sw s0,0(sp)
        li s2,1000 # blue = 1000
        li a0,10 #first argument = 10
        la a1,aaa #second argument holds the address of aaa[0]
        li a2,4 # third argument = 4
        jal special_sum #jump to special_sum label
        add s0,zero,a0 # red = returm from labek special_sum
        li a0,200       # first argument = 200
        la a1,bbb       #second argument hold address bbb
        li a2,4         #third argument = 4
        jal special_sum         #jump to special_sum
        add s1,zero,a0
        li a0,500
        la t0,ccc
        add a1,t0,zero
        li a2,6
        jal special_sum
        add s2,s2,a0
        add s2,s2,s1
        add a0,s2,s0
        lw s0,0(sp)
        lw s1,4(sp)
        lw s2,8(sp)
        lw ra,12(sp)
        addi sp,sp,16
        jr      ra
```

EX F

```
# swap.asm

# ENCM 369 Winter 2023 Lab 3 Exercise F



# BEGINNING of start-up & clean-up code.  Do NOT edit this code.

        .data

exit_msg_1:

        .asciz  "***About to exit. main returned "

exit_msg_2:

        .asciz  ".***\n"

main_rv:

        .word   0



        .text

        # adjust sp, then call main

        andi    sp, sp, -32             # round sp down to multiple of 32
        jal     main

        # when main is done, print its return value, then halt the program
        sw      a0, main_rv, t0
        la      a0, exit_msg_1
        li  a7, 4
        ecall
        lw      a0, main_rv
        li      a7, 1
        ecall
        la      a0, exit_msg_2
        li      a7, 4
        ecall
    lw    a0, main_rv
        addi    a7, zero, 93    # call for program exit with exit status that is in a0
        ecall
# END of start-up & clean-up code.

# int foo[] =  { 0x600, 0x500, 0x400, 0x300, 0x200, 0x100 }
        .data
    .globl      foo
foo:    .word   0x600, 0x500, 0x400, 0x300, 0x200, 0x100
```

```
# int main(void)
        .text
        .globl  main
main:
    addi    sp, sp, -32
     sw     ra, 0(sp)

    la    t0, foo        # t0 = &foo[0]
    addi    a0, t0, 20    # a0 = &foo[5]
    addi    a1, t0, 0    # a1 = &foo[0]
    jal    swap

    la    t0, foo        # t0 = &foo[0]
    addi    a0, t0, 16    # a0 = &foo[4]
    addi    a1, t0, 4    # a1 = &foo[1]
    jal    swap

    la    t0, foo        # t0 = &foo[0]
    addi    a0, t0, 12    # a0 = &foo[3]
    addi    a1, t0, 8    # a1 = &foo[2]
    jal    swap



    add    a0, zero, zero
    lw    ra, 0(sp)
    addi    sp, sp, 32
    jr    ra

# void swap(int *left, int *right)
#
    .text
    .globl  swap
swap:
    # Students: Replace this comment with code to make swap
    # do its job correctly.

    add     t0, zero, zero    # old_star_left
    add    t1, a0, zero    # address of left
    add    t2, a1, zero    # address of right
```

```
lw    t3, (t1)    # *left
lw    t4, (t2)    # *right

add    t0, t0, t3

sw    t4, (t1)
sw    t0, (t2)

jr    ra
```