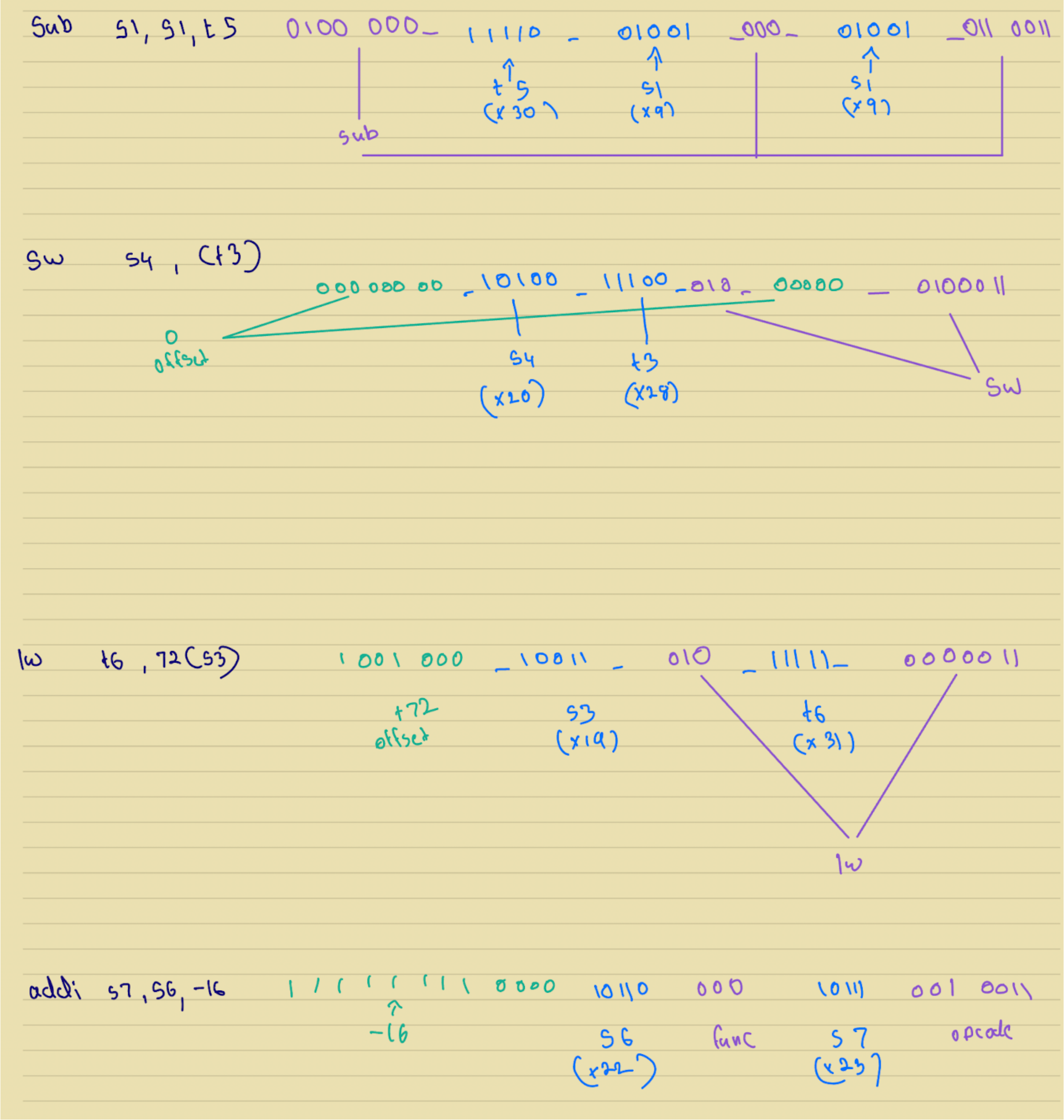
Course : Computer Organization – ENCM 369

Lab # : Lab 2

Student Name : Nimna Wijedasa

Lab Section : B04

Ex A



Sub s1, s1, t5 0100 000_ 11110 _ 01001 _000_ 01001 _011 0011

sw s4 , (t3) 000 000 00 _10100 _ 11100 _010 _ 00000 _ 0100011

lw t6 , 72(s3) 1001 000 _10011 _ 010 _11111_ 0000011

addi s7, s6, -16 111111111 0000 10110 000 10111 001 0011

Ex C

```
# array-sum2C.asm
# ENCM 369 Winter 2023 Lab 2 Exercise C Part 3

# Start-up and clean-up code copied from stub1.asm

# 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.

# Global variables
        .data
        # int abc[ ] = {-32, -8, -4, -16, -128, -64}
        .globl      abc
abc:    .word       -32, -8, -4, -16, -128, -64

# Hint for checking that the original program works:
# The sum of the six array elements is -252, which will be represented
# as 0xffffff04 in a RISC-V GPR.

# Hint for checking that your final version of the program works:
# The maximum of the four array elements is -4, which will be represented
# as 0xfffffffc in a RISC-V GPR.


# int main(void)
#
# local variable    register
#   int *p          s0
#   int *end                s1
#   int sum                 s2
#   int max                 s3  (to be used when students enhance the program)
```

```
        .text
        .globl  main
main:
        la      s0, abc                 # p = abc
        add     s3, zero, zero                  # max = 0
        lw      s3, (s0)                # max = abc[0]
        addi    s1, s0, 24              # end = p + 6
        add     s2, zero, zero              # sum = 0
L1:
        beq     s0, s1, L2              # if (p == end) goto L2
        lw      t0, (s0)                # t0 = *p
        add     s2, s2, t0              # sum += t0
        lw      t0,(s0)
        bgt     t0,s3,UP                # if abc > max go to update
        addi    s0, s0, 4               # p++
        j       L1

UP:
        lw      s3,(s0)                         # max = abc[i]
        addi    s0, s0, 4               # p++
        j       L1
L2:
        add     a0, zero, zero              # return value from main = 0
        jr      ra
```

Ex D

```
# stub1.asm
# ENCM 369 Winter 2023 Lab 2
# 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.
#
# Local Variable    Register
# int *p            s1
# int *guard                  s2
# int min                     s3
# int j                       s4
# int k                       s5
# int *alpha                  s6
# int *beta                   s7
# int compare                 t1
# Below is the stub for main. Edit it to give main the desired behaviour.
          .data
    .globl  alpha
alpha:  .word 0xb1, 0xe1, 0x91, 0xc1, 0x81, 0xa1, 0xf1, 0xd1
    .globl  beta
beta:   .word 0x0, 0x10, 0x20, 0x30, 0x40, 0x50, 0x60, 0x70

          .text
          .globl    main

main:
          la s7,alpha                   #p1=alpha
          la s8,beta                    #p2=beta
```

```
        addi s3,s7,32                   #guard = p+8
        lw s4,(s7)                      #min = *p
L0:
        addi s7,s7,4                    #p++
        beq s7,s3,L1                    #if (p==guard)goto l1
        lw t0,(s7)                      #t0 = *p
        bge t0,s4,L3                    #if (*p>=min) goto L3
        lw s4,(s7)                      #min = *p
        addi s7,s7,4                    #p++
L3:
        j L0                            #jump to L0

L1:

        li   s5, 0              # j = 0
        li   s6, 7              # k = 7
        la s7,alpha
        la s8,beta
        addi s9,s9,8

L5:
        bge s5,s9,L4                    #if (j>=8) goto L4
        slli t1,s6,2                    #shift right by 2
        add t2,s8,t1
        lw t3,(t2)
        slli t4,s5,2
        add t5,s7,t4
        sw t3,(t5)
        addi s5,s5,1
        addi s6,s6,-1
        j    L5                 # goto loop

L4:
        add     a0, zero, zero          # return value from main = 0
        jr      ra
```