Course: ENCM 369
Lab Section: B03
Lab 2
Student Name: Mitchell Sawatzky
Date Submitted: Jan 29, 2016

Exercise A

Instruction	Machine Code						But how?
	OP	RS	RT	RD	SHAMT	FUNCT	
slt $t0, $s3, $s3 R: slt rd, rs, rt	000000	10011	10011	01000	00000	101010	According to APP B, slt has an op code of 0 and a funct code of 42. $t0 is register 8 and $s3 is register 19 (table 6.1). The slt mnemonic always has a SHAMT of 0.
	OP	RS	RT	IMM			
addi $s0, $s0, -8 I: addi rt, rs, imm	001000	10000	10000	1111 1111 1111 1000			According to p. 307, addi has an op code of 8. $s0 is register 16 (table 6.1), and the immediate is -8 in twos compliment.
	OP	RS	RT	IMM			
lw $t8, 40($s5) I: rt, imm(rs)	100011	10101	11000	0000 0000 0010 1000			According to p. 307, lw has an op code of 35. $t8 is register 24 and $s5 is register 21 (table 6.1), and the immediate is 40.
	OP	RS	RT	IMM			
sw $s1, ($t3) I: rt, imm(rs)	101011	01011	10001	0000 0000 0000 0000			According to p. 307, sw has an op code of 43. $s1 is register 17 and $t3 is register 11 (table 6.1). The immediate is 0.

Exercise C

array-sum.asm

```
# array-sum.asm
# ENCM 369 Winter 2016 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:
        .asciiz "***About to exit. main returned "
exit_msg_2:
        .asciiz ".***\n"
main_rv:
        .word   0

        .text
        # adjust $sp, then call main
        addi    $t0, $zero, -32         # $t0 = 0xffffffe0
        and     $sp, $sp, $t0           # round $sp down to multiple of 32
        jal     main
        nop

        # when main is done, print its return value, then halt the program
        sw      $v0, main_rv
```

```
        la      $a0, exit_msg_1
        addi    $v0, $zero, 4
        syscall
        nop
        lw      $a0, main_rv
        addi    $v0, $zero, 1
        syscall
        nop
        la      $a0, exit_msg_2
        addi    $v0, $zero, 4
        syscall
        nop
        addi    $v0, $zero, 10
        syscall
        nop
# END of start-up & clean-up code.


# Global variables
        .data
        # int xyz[] = { -8, 16, -32, 64, -128, 256 }
        .globl  xyz
xyz:    .word   -8, 16, -32, 64, -128, 256

# Hint for checking that the original program works:
# The sum of the six array elements is 168, which will be represented
# as 0x000000a8 in a MIPS GPR.

# Hint for checking that your final version of the program works:
# The minimum of the six array elements is -128, which will be represented
# as 0xffffff80 in a MIPS GPR.


# int main(void)
#
# local variable register
#   int *p              $s0
#   int *end            $s1
#   int min             $s2  (to be used when students enhance the program)
#   int total           $s3
#
        .text
```

```
        .globl  main
main:
        la      $s0, xyz        # p = foo
        addi    $s1, $s0, 24            # end = p + 6
        add     $s3, $zero, $zero       # total = 0
        lw      $s2, ($s0)              # min = *p
L1:
        beq     $s0, $s1, L2            # if (p == end) goto L2
        lw      $t0, ($s0)              # $t0 = *p
        add     $s3, $s3, $t0           # total += $t0
        addi    $s0, $s0, 4             # p++
        slt     $t1, $t0, $s2           # if ($t0 < min) $t1 = 1
        bne     $t1, $zero, L3          # if ($t1 != 0) goto L3
        j       L1
L3:
        add     $s2, $t0, $zero         # min = $t0 + 0
        j       L1                      # goto L1
L2:
        add     $v0, $zero, $zero       # return value from main = 0
        jr      $ra
```

exD.asm

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

        .text
        # adjust $sp, then call main
        addi    $t0, $zero, -32         # $t0 = 0xffffffe0
        and     $sp, $sp, $t0           # round $sp down to multiple of 32
        jal     main
        nop

        # when main is done, print its return value, then halt the program
        sw      $v0, main_rv
        la      $a0, exit_msg_1
        addi    $v0, $zero, 4
```

```
        syscall
        nop
        lw      $a0, main_rv
        addi    $v0, $zero, 1
        syscall
        nop
        la      $a0, exit_msg_2
        addi    $v0, $zero, 4
        syscall
        nop
        addi    $v0, $zero, 10
        syscall
        nop
# END of start-up & clean-up code.

# GLOBAL variables
        .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

# Register Allocations
# Register              Variable
# $s0                   pa
# $s1                   pb
# $s3                   guard
# $s4                   i
# $s5                   min
# $s6                   imin

        .text
        .globl  main
# Main Entry Point
main:
        la      $s0, alpha              # pa = alpha
        addi    $s3, $s0, 32            # guard = pa + 8
        la      $s1, beta               # pb = beta
        addi    $s1, $s1, 32            # pb += 8
L1:
        beq     $s0, $s3, L2            # if (pa == guard) goto L2
        addi    $s1, $s1, -4            # pb--
```

```
        lw      $t0, ($s0)              # $t0 = *pa
        sw      $t0, ($s1)              # *pb = $t0
        addi    $s0, $s0, 4             # pa++
        j       L1                      # goto L1
L2:
        add     $s6, $zero, $zero       # imin = 0
        la      $t1, alpha              # $t1 = alpha
        lw      $s5, ($t1)              # min = alpha[0]
        addi    $s4, $zero, 1           # i = 1
L3:
        addi    $t3, $zero, 8           # $t3 = 8
        slt     $t4, $s4, $t3           # $t4 = (i < $t3)
        beq     $t4, $zero, L6          # if ($t4 != 0) goto L6

        sll     $t5, $s4, 2             # $t5 = i * 4
        add     $t5, $t5, $t1           # $t5 += alpha
        lw      $t6, ($t5)              # $t6 = *$t2
        slt     $t7, $t6, $s5           # $t7 = ($t6 < min)
        bne     $t7, $zero, L5          # if ($t7 != 0) goto L5
L4:
        addi    $s4, $s4, 1             # i++
        j       L3                      # goto L3
L5:
        sll     $t2, $s4, 2             # $t2 = i * 4
        add     $t2, $t2, $t1           # $t2 += alpha
        lw      $s5, ($t2)              # min = *$t2
        add     $s6, $zero, $s4         # imin = i
        j       L4                      # goto L4
L6:
        jr      $ra                     # return
```