Course: ENCM 369
Lab Section: B03
Lab 4
Student Name: Mitchell Sawatzky
Date Submitted: Feb 12, 2016

Exercise C

toLower.asm

```
# tolower.asm
# ENCM 369 Winter 2016 Lab 4 Exercise C

# 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.

# int lower_char(int c)
# (Code for this procedure is complete and correct.)
        .text
        .globl  lower_char
lower_char:
        add     $v0, $a0, $zero         # result = c
        slti    $t1, $v0, 65            # $t1 = result < 65
        bne     $t1, $zero, L1          # if ( $t1 ) goto L1
        slti    $t2, $v0, 91            # $t2 = result <= 90
        beq     $t2, $zero, L1          # if ( !$t2 ) goto L1
        addi    $v0, $v0, 32            # result += 32
L1:

        # The remaining code does not help in the translation of lower_char.
        # It is here to make sure you are very careful with
        # use of a-registers and t-registers when translating lower_string.
        # Do not modify any of the code up to and including
        # the jr instruction.  (When coding a nonleaf procedure,
        # it is useful to remember that any jal to a callee might
        # result in destruction of data in all of the a-registers and
        # t-registers!)
        li      $t0, 0x0bad0008
        addi    $t1, $t0, 1
        addi    $t2, $t0, 2
        addi    $t3, $t0, 3
        addi    $t4, $t0, 4
        addi    $t5, $t0, 5
        addi    $t6, $t0, 6
        addi    $t7, $t0, 7
        addi    $t8, $t0, 16
        addi    $t9, $t0, 17
        addi    $a0, $t0, -4
        addi    $a1, $t0, -3
        addi    $a2, $t0, -2
        addi    $a3, $t0, -1
        jr      $ra

# void lower_string(char *to, const char *from)
#

        .text
        .globl lower_string
lower_string:
        addi    $sp, $sp, -16
        sw      $ra, 12($sp)
        sw      $s0, 8($sp)
        sw      $a0, 4($sp)
        sw      $a1, 0($sp)

L2:     lw      $t0, 0($sp)
        lb      $a0, ($t0)
        jal     lower_char
        add     $s0, $v0, $zero
        lw      $t0, 4($sp)
        sb      $s0, ($t0)
        beq     $s0, $zero, L3
        lw      $t0, 0($sp)
        addi    $t0, $t0, 1
        sw      $t0, 0($sp)
        lw      $t0, 4($sp)
        addi    $t0, $t0, 1
        sw      $t0, 4($sp)
        j       L2

L3:     lw      $s0, 8($sp)
        lw      $ra, 12($sp)
        addi    $sp, $sp, 16
        jr      $ra

.data
.globl  result
result: .space  40              # char result[40]

NEWLINE:.asciiz "\n"
S1:     .asciiz "Exercise 4C result is ..."
S2:     .asciiz "ENCM 369 Winter 2015 AZ az [ ] @ !!!"

.text
main:
        addi    $sp, $sp, -4
        sw      $ra, 0($sp)

        la      $a0, S1
        li      $v0, 4
        syscall                 # puts("Exercise4C result is...")
        la      $a0, NEWLINE
        li      $v0, 4
        syscall                 # puts("\n")

        la      $a0, result     # $a0 = result
        la      $a1, S2         # $a1 = "ENCM 369 Winter 2015 AZ az [ ] @ !!!"
        jal     lower_string    # lower_string()

        la      $a0, result
        li      $v0, 4
        syscall                 # puts(result)
        la      $a0, NEWLINE
        li      $v0, 4
        syscall                 # puts("\n")

        add     $v0, $zero, $zero

        lw      $ra, 0($sp)
        addi    $sp, $sp, 4
        jr      $ra
```

Exercise D

append.asm

```
# ENCM 369 Winter 2016 Lab 4 Exercise D
#
# Simple example of allocation and use of an array of chars within the stack
# frame of a procedure.

# 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.


        .data
S1:     .asciiz ""
S2:     .asciiz "W"
S3:     .asciiz "inter "
S4:     .asciiz "2"
S5:     .asciiz "016"
S6:     .asciiz " ENCM 369"
NEWLINE:.asciiz "\n"
        .text

# int main(void)
main:
        addi    $sp, $sp, -32
        sw      $ra, 28($sp)
        sw      $s0, 24($sp)

        add     $s0, $sp, $zero
        sb      $zero, ($s0)    # str[0] = '\0'
        add     $a0, $s0, $zero
        la      $a1, S1
        jal     append

        add     $a0, $s0, $zero
        la      $a1, S2
        jal     append

        add     $a0, $s0, $zero
        la      $a1, S3
        jal     append

        add     $a0, $s0, $zero
        la      $a1, S4
        jal     append

        add     $a0, $s0, $zero
        la      $a1, S5
        jal     append

        add     $a0, $s0, $zero
        la      $a1, S1
        jal     append

        add     $a0, $s0, $zero
        la      $a1, S6
        jal     append

        add     $a0, $sp, $zero
        li      $v0, 4
        syscall
        la      $a0, NEWLINE
        li      $v0, 4
        syscall

        add     $v0, $zero, $zero

        lw      $s0, 24($sp)
        lw      $ra, 28($sp)
        add     $sp, $sp, 32
        jr      $ra
```

```
# void append(char *dest, const char *src)
append:
        add     $t0, $zero, $zero       # i = 0
L1:     add     $t3, $a0, $t0           # $t3 = dest + i
        lb      $t3, ($t3)              # $t3 = dest[i]
        beq     $t3, $zero, L2          # if ($t3 == 0) goto L2
        addi    $t0, $t0, 1             # i++
        j       L1
L2:     add     $t1, $zero, $zero       # j = 0
L3:     add     $t3, $a1, $t1           # $t3 = src + j
        lb      $t2, ($t3)              # c = src[j]
        add     $t3, $a0, $t0           # $t3 = dest + i
        sb      $t2, ($t3)              # dest[i] = c
        addi    $t0, $t0, 1             # i++
        addi    $t1, $t1, 1             # j++
        beq     $t2, $zero, L4          # if (c == 0) goto L4
        j       L3
L4:     jr      $ra
```