COE 147 Spring 2013
Lab 3 Solution: Endianness, Bit Manipulation, Strings, Loops

Part 1: Endianness
part a:

```
.data
a: .byte 0x62, 0xA4, 0x19, 0x6C

Q1: What is the address of byte 0x19?

Answer: 0x10010002

.data
a: .word    0x62A4196zC

Q2: What is now the address of byte 0x19?

Answer: 0x10010001

Q3: Is the simulator little endian or big endian? How can you tell?

Answer: Simulator is little-endian. Because it is organizing lower bytes of a word
in lower memory addresses and higher bytes in higher memory addresses. Because of
this little-endianness, 'a' described as a four-byte sequence matches 'a' described
as a single-word.
```

part b:

```
.data

str1:       .asciiz           "Please enter your integer: "
str2:       .asciiz           "Here is the output: "

.text
# print string
la    $a0, str1
li    $v0, 4
syscall

# read integer
li    $v0, 5
syscall

# save integer from v0 to t0
addi  $t0, $v0, 0

# isolate bits 24,25 and 26
sll   $t0, $t0, 6
srl   $t0, $t0, 29

# print string
la    $a0, str2
li    $v0, 4
syscall

# print isolated bits in t0 as an integer
```

```
addi  $a0, $t0, 0
li    $v0, 1
syscall
```

Part 2: Strings (Modifying In Place)

```
.data
some_str:   .space      64

str1:       .asciiz     "Please enter your string: "
str2:       .asciiz     "Here is the output: "

.text
# print string
la    $a0, str1
li    $v0, 4
syscall

# get the string input
la    $a0, some_str
li    $a1, 64
li    $v0, 8
syscall

# now traverse through the bytes of the strings
# if the byte is in the range 0x61-0x7a (A-Z) then add 0x20 to it
la    $t0, some_str
li    $t6, 0x61         # 'a'
li    $t7, 0x7a         # 'z'
li    $s0, 0x41         # 'A'
li    $s1, 0x5a         # 'Z'
li    $t5, 0x00         # null terminator

LOOP:
lb    $t1, 0($t0)

# check null terminator
beq   $t1, $t5, END_OF_PROCESSING

# check start of range
# if t1<t6 then do nothing
blt   $t1, $t6, CMP_UPPER

# else if t1>t7, then do nothing
bgt   $t1, $t7, DO_NOTHING

# else add 0x20 to t1 and store it into byte addressed by t0
addi  $t1, $t1, -32
sb    $t1, 0($t0)
j DO_NOTHING

CMP_UPPER:
# if t1<s0 then do nothing
blt   $t1, $s0, DO_NOTHING

# else if t1>s1, then do nothing
bgt   $t1, $s1, DO_NOTHING

addi  $t1, $t1, 32
sb    $t1, 0($t0)
```

```
DO_NOTHING:
addi  $t0, $t0, 1 # next byte

j     LOOP

END_OF_PROCESSING:
# print string
la    $a0, str2
li    $v0, 4
syscall

# print output string
la    $a0, some_str
li    $v0, 4
syscall
```

Part 3: Strings (Modifying a Copy)

```
.data
buf1:       .space      64
buf2:       .space      64

str1:       .asciiz     "Please enter your string: "
str2:       .asciiz     "Here is the output: "

.text
# print string
la    $a0, str1
li    $v0, 4
syscall

# get the string input
la    $a0, buf1
li    $a1, 64
li    $v0, 8
syscall

# now get the length of buf1
la    $t0, buf1
li    $t2, 0x00              # null byte
li    $t4, 0x20              # SPACE byte
la    $s0, buf2         # start copy place
add   $t7, $t0,$zero    # copy start address

LOOP1:
lb    $t1, 0($t0)

# check null terminator
beq   $t1, $t2, END_OF_BUF1
beq   $t1, $t4, END_OF_BUF1  #space
addi  $t0, $t0, 1          # next byte
j     LOOP1

END_OF_BUF1:
# fix the tail pointer in buf1
addi $t6, $t0, 0               #store last end space
beq   $t1, $t4, SUB_ADDR     #add space is detected
subi  $t0, $t0, 1          #last byte of the string (past null byte and the
newline)
SUB_ADDR:
```

```
subi  $t0, $t0, 1          # space: just minus 1
# copy bytes from buf1 into buf2 in reverse
# t0 already pointing to last byte of buf1
# let t1 point to start of buf2

LOOP2:
# get the byte at t0 (on buf1)
lb    $t3, 0($t0)
sb    $t3, 0($s0)
# as copying is valid, so increment t1 (on buf2)
addi  $s0, $s0, 1
beq   $t0, $t7, END_COPYING
subi  $t0, $t0, 1

j     LOOP2

END_COPYING:
# copying done... put the null terminator at the end of buf2
addi $t7, $t6, 1           #store start space
addi $t0, $t6, 1
beq   $t1, $t4, LOOP1        #space
li    $t2, 0x00
sb    $t2, 0($s0)

# DONE
# print string
la    $a0, str2
li    $v0, 4
syscall

# print output string
la    $a0, buf2
li    $v0, 4
syscall
```