# CS/COE 0447 Fall 2009
# Lab 3: Strings, Loops and Look up Tables
# Due Date: October 1, 2009

To get started on this lab, attend recitation on 9/25. Each of you should submit your own solution, according to these instructions: http://www.cs.pitt.edu/~sab104/teaching/cs447/submission.html. You may collaborate with your partner, but each person must turn in their own copy of the lab, with the name of their partner. The lab is due on 10/1 at 11:59pm.

## Part 1: Strings and Loops

1) The following data segment defines a null-terminated string:

```
.data
str:      .asciiz        "Hello World!"
```

Each byte of memory stores the ASCII code of the corresponding character in the string. For example, the first byte (address 0x10010000, right part of the box in MARS) contains the value 0x48 (72 decimal), which corresponds to the letter "H" in ASCII code (you can find a list of the ASCII codes in the book or online, for example at http://www.asciitable.com). The last byte allocated to the string (address 0x10010011) contains the value 0x00, which identifies the end of the string.

Write a MIPS program that transforms each of the characters in the string to its corresponding capital letter and then prints the string to standard output.

Hint: before converting a character, make sure it is actually a lowercase letter, by testing that it is in the corresponding range.

2) The following data segment defines 2 different 50-byte long regions of data, which we will use as buffers to store strings:

```
.data
buf1:     .space 50
buf2:     .space 50
```

We will write a MIPS program that copies a string from one buffer to another, ignoring whitespaces. First, write MIPS code that prompts the user for a string and stores it in buf1. Then, write MIPS code that copies each of the characters in buf1 to buf2, but ignoring whitespaces. You will have to keep two separate pointers to each buffer, because one might be moving faster than the other one. Finally, write MIPS code that prints the contents of buf2.

## Part 2: Look up Tables

To do this part of the lab, you need a special version of the MARS simulator, which is available here: http://www.cs.pitt.edu/~childers/CS0447/secure/Mars-3.7-jvm1.5-LED.jar (this is the same version required for the project).

The goal of this part of the lab is to read a number between 0 and 8 and then turn on some of the LEDs on the display. The progam should print the first *n* LEDs of the display, where *n* is the number read. The program quits when the users enters a negative number.

The following data segment defines a table of values, each holding the bit pattern to display:

```
.data
table: .word   0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8, 0xFC, 0xFE, 0xFF
```

If we write one of these values, we'll see that some of the leds on the display light up:

```
.text
li $t0, 0xffff0000  #LED memory begins at this address
li $t1, 0xC0        #0xC0 is the second element, so 2 LEDs should
                    #light up
sw $t1, 0($t0)      #Store the pattern in memory
```

First, write MIPS code that reads a number and checks that it is in range. Then, write MIPS code that looks up a value in the table to determine the bit pattern of the LEDs to turn on and stores that value in the memory location that corresponds to the first word of the LED display. Finally, modify your MIPS code so that it asks for another number, until the user enters a negative number.