The UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Comp 411 Computer Organization
Spring 2009

Problem Set #3
Issued Tuesday 10 Feb 09; Due Tuesday 17 Feb 09

Homework Information: Some of the problems are probably too long to be done the night before the due date, so plan accordingly. Late homework will not be accepted. Feel free to get help from others, but the work you hand in should be your own. Email your solutions as a pdf, txt, or doc file to Sang Woo Lee. Submit your assembly language program as an attachment as described below.

Problem 1. "MIPS Calisthenics" (25 points)

Write MIPS code fragments to perform the following simple tasks.

(A) [5 points] Clear registers \$t1-\$t4

(B) [5 points] Swap the contents of registers \$t1 and \$t2

(C) [7 points] Count the number of words in the memory address range from 0x100 to 0x147 (inclusive) that contain the value zero. Remember that all memory addresses are byte addresses, i.e., address 0x0 is the first byte in memory, address 0x1 is the second byte, etc.

(D) [8 points] Clear memory locations (bytes, not words) at byte addresses 0x100 to 0x145, inclusive. *Hint:* Use the store byte instruction (sb) instead of sw. You could use a combination of sw and sb here, but you can use only sb to keep things simple.

Problem 2. "Name that Loop" (25 points)

(A) [6 points] Describe in your own words the function performed by the following code fragment when it reaches the label `end`. Consider the following hints. The code fragment operates on the contents of register a0, and the loop will always complete.

```
          add   $t0,$0,$0
loop:     andi  $t1,$a0,1
          add   $t0,$t0,$t1
          srl   $a0,$a0,1
          bne   $a0,$0,loop
end:      add   $a0,$t0,$0
```

(B) [6 points] Describe how the operation of the above code fragment changes if the `srl` instruction is replaced with an `sra` instruction with the same arguments.

(C) [6 points] Describe the function of the following code fragment in your own words:

```
          la    $t1,a
loop:     lw    $t0,4($t1)
          sw    $t0,($t1)
          addi  $t1,$t1,4
          bne   $t0,$0,loop
```

(D) [7 points] Describe the function of the following code fragment in your own words:

```
          la    $t1,a
loop:     lw    $t0,($t1)
          addi  $t1,$t1,4
          bne   $t0,$0,loop
          la    $t0,a
          sub   $t0,$t1,$t0
          sra   $t0,$t0,2
```

Problem 3. "A Loop of Your Own" (50 points)

Download and install MARS, the MIPS instruction set simulator (see class website for download link). Then, write an assembly program to do the following.

You are given an array called Pixels, each element of which is a 32-bit word representing a color value. The lowest significant 8 bits of each color value denote an unsigned integer (from 0 to 255) representing the color's "blue" value, the next 8 bits are the "green" value, the next 8 bits are the "red" value, and the most significant 8 bits are all zeros. The size of the array is not known, but it is known that the last element of the array is a negative number (e.g., "-1"); this negative value should itself not be considered as a valid color value, of course. For example, the following array has three color values:

```
.data
Pixels:  .word   0x010101, 0x5, 0x1111, -1
```

Your program should read the Pixels array, one element at a time, until the end is reached, and for each element read, it should print the following line to the console:

Pixel *n* = *r g b*

where *n* is the number of the array element read (first one is 0th), and *r, g* and *b* are integers (from 0 to 255) representing the red, green and blue values for that pixel. You need not worry too much about formatting and white space etc., but please print only one element per line.

So, if you used the Pixels array given above, your output should be:

```
Pixel 0 = 1 1 1
Pixel 1 = 0 0 5
Pixel 2 = 0 17 17
```

Your code fragment will begin execution at the label “main”. You will use the syscall instruction (Appendix A. 43-45) to print the above output, and also to terminate your program.

Submission: Test your code using MARS and turn in a copy of the assembly language file (.asm) by email. Put “hw3” in the subject line, and be sure to email it by start of class.. Do not turn in hard copy.

Grading:

- If your program does not produce the correct result for the above example of Pixels array, you will receive zero credit.
- If your program works correctly on the above example, and on our ‘secret’ example of Pixels array, you will receive full credit (50 points). Don’t worry; we will not be trying to trick you!
- Otherwise, you will receive only half credit (25 points)

Collaboration: *For this problem on this homework assignment only* you are free to take as much help from other students as you need to write and debug your code. You must, however, understand everything you submit, and be able to explain if asked.