Spring 2016 **ECE 3056**

Assignment 1

Due Dates:

- **Part I MIPS Program: 11:55 pm, Thursday January 21st, 2016**
- **Part II Homework Problems: Beginning of class, Thursday January 21st, 2016**

Part I: MIPS Programming (100 pts)

Purpose: To reacquaint you with the MIPS-32 ISA and the control flow behavior captured in the ISA. This will guide your subsequent understanding of the data path implementations and the problems/issues that must consequently be solved in high performance data path implementations.

SPIM Warm-Up:

We will be using the QTSPIM simulator in this class. You can use any simulator you wish but make sure it is compliant with the MIPS ISA from the text. The TA is not responsible for making changes to get your program to work in the simulator.

Information about the QTSPIM simulator can be found linked off of the class Resources page. The following warm-up should not take more than 30 minutes if you have no problems. *If you hit an hour, talk to the TA or me*. Of course you may choose to spend more time working with several of the examples that are provided on the class Resources page. You can work with each other on this part of the assignment.

1. Following instructions from the Resources page linked on the class webpage (not T-square) download and install QTSPIM. The following steps are simply to familiarize you with the SPIM simulator.
2. Select Qtspim → Preferences and the MIPS tab. Make sure *Accept Pseudoinstructions* is selected. *Bare Machine* should not be selected.
3. Select Window→ Tile. This will create tabs for multiple windows in SPIM.
4. Set starting address of the code to 0x00400000 by going to Simulator → Run Parameters
5. Set register display to hexadecimal by Registers→Hex
6. Select all windows under the Text Segment
7. Make sure all three structures are selected under the Data Segment and the Hex display is selected.
8. Make sure all elements are selected under Windows.
9. Download the example SPIM program *constant.example.s* from the example programs linked off of the Resources page on the class website.
10. Load program: File → Load File. If there are any errors you should see this in the bottom windowpane.
11. Select the Int Regs tab. This should display all of the registers. Note the value of register $t4. It is 0x00000000. From the SPIM program you know that this is the register used to construct a 32-bit constant.

12. Execute this program by Simulator → Run/Continue
13. Now look at $t4. It will contain the constant value as specified in the original program. Study the program to refresh your memory about MIPS instructions.
14. Reload the program by File → Reinitialize and Load File
15. Select the Int Regs tab. This should display all of the registers. Note the value of register $t4. It is 0x00000000. Now single step the program with Simulator → Single Step. At step 2 you will see the upper bits set. At step 3 you will see the lower bits set. At the top of the window you will see the Program Counter advance by 4 with the execution of each instruction step.
16. Note the last two instructions: This is the proper way to terminate a SPIM program. Every program should end this way.
17. Load and execute the other example programs provided via the Resources page. Try and make some simple changes to these programs and execute them correctly before writing your own. Explore the data segment and text segment with these examples.
18. In particular, look carefully at *io.example.s*. This shows you how to read integers from the console and how to print characters to the console window.
19. Finally, the sample SPIM template on the Resources page has a more comprehensive I/O example.
20. As soon as the TA is assigned, the TA will be giving demos of the SPIM simulator during office hours for groups of students. Attendance is highly recommended. Appointments are not necessary. Specifically you should get comfortable with the following steps.

- Learn to single the step program execution and to set breakpoints. This is invaluable for debugging.
- Learn to read the text and data segments

Main Assignment:

This programming assignment is just to reacquaint you with the MIPS instruction set. Hence specific details other than that listed below are unimportant. For example, you can assume only 4x4 matrices are supported.

1. Write a SPIM program that will compute the maximum value of each column of the matrix stored in row major order starting at the location labeled Original in the starter template below. The results should be stored starting at the location labeled Max. The arrays we use will be 4x4 elements. Here is the program template to help you get started.

```
          .data
strA:     .asciiz “Original Array:\n “
strB:     .asciiz “Second Array:\n: “
newline:  .asciiz “\n”
space :   .asciiz “   “
```

```
# This is the start of the original array.
Original:   .word  200,  270,  250,  100
            .word  205,  230,  105,  235
            .word  190,   95,   90,  205
            .word   80,  205,  110,  215

# The next statement allocates room for the results in 4*4 = 16 bytes.
#
Max:        .space 16

.align 2

            .globl main
            .text

main:       # Your fully commented program starts here.
```

2. The program should include a block of code that prints both the original matrix in row major order and the maximum values (one for each column) to console. See example programs on the class resources page for code examples to help you with this.
3. Your program should be well commented.

Requirements and Suggestions:

The following steps are recommended to complete the assignment.

- You are required to use iteration.
- You are required to use at least one procedure.
- Some suggestions.
  - First understand the layout of the matrix in memory. How do addresses differ between elements of a row or column?
  - Perform register allocation: decide which registers you will use and for what purpose, e.g., loop count. Look at the examples.
  - Refine the algorithm description in terms of registers and memory addresses
  - Write and test blocks of code first. For example,
    - A block of code that prints a row to the screen (do not forget the spaces!)
    - A block of code that prints the matrix to the screen (row at a time)
    - Writing these loops is a good precursor for the remaining loops.
    - Test a loop for finding the maximum in a column.
- Use breakpoints to help in debugging.
- Use single step mode as the final resort in debugging. It is time consuming but gives you the most information.
- First load and execute the simpler programs available from the class webpage.

Grading Guidelines:

For your information here are the grading guidelines for the SPIM component of the assignment.

- **30 points:** Program compiles without errors on SPIM (and *appears* on the surface to be correct).
- **45 points:** Program executes correctly: This means that the memory locations contain appropriate values and the correct answers are printed out to console as well. A different 4x4 matrix than the one in the starter code might be used to test your code.
- **25 points:** Documentation and description of the program:
  - Have a commented program header with your name, class, and assignment (look at the starter program on the class resources page)
  - The program should well commented so much so the TA can determine what you are doing without you there to explain it.

Submission Guidelines:

Your code should be submitted electronically via t-square, by **11:55 pm** on the due date. Homework problems (below) should be submitted on paper at the beginning of class on the due date.

Part II: Problem Set (50 pts)

1. (**12 pts**) Consider a byte addressed memory. For the following memory sizes, how many bits do you need to specify a memory address?
   a. 31 bytes
   b. 128 Kbytes
   c. 24 Mbytes
   d. 32 Gbytes

2. (**10 pts**) Encode the following text strings in ASCII. Show your answers as a sequence of bytes where each byte is encoded in hexadecimal notation (do not forget the spaces!)
   a. First Test
   b. SPIM

3. (**12 pts**) Decode the following set of encoded instructions

```
8c290004
01095024
3402000a
```

4. (**16 pts**) Consider the following sequence of directives.

```
        .data
start:  .word 0x3, 18, str
str:    .asciiz "Test"
        .align 2
end:    .byte 0x7, 0x11
        .align 2
        .word 22
```

   a. Show the hexadecimal contents of the data segment assuming that it starts at 0x10010000. Show the address and contents of each location.
   b. What is the address of the byte storing the value 0x7.

<u>Note:</u> **<u>No late assignments will be accepted</u>**. Remember, you must make a passing grade of 50% in the assignments to pass the course!