1 Instructions

You may work in pairs with a partner on this assignment if you wish or you may work alone. If you work with a partner, only submit one zip archive with both of your names in the document and source code file; you will each earn the same number of points. Your solution file must be uploaded to Blackboard by the assignment deadline. Section 3 describes what to submit and by when. Download the *Homework Assignment 3 Zip Archive* file from the course website and extract it.

2 Exercises

1. **(25 pts) MIPS32 Assembly Language Programming Problem 1 (h3_1.s)** *The source code file h3_1.s in the Homework Assignment 3 zip archive is a template for the complete MIPS program. Implement your program in this file using the comments as a guide. You will submit hw3_1.s for grading.* Write a complete MIPS32 assembly language program that operates as shown below, where user input is shown in **bold**. That is, the program prompts for and reads two integers a and b. It then calculates and displays the sum of a and b, the difference between a and b, the product of a and b, the quotient from dividing a by b, and the remainder from dividing a by b.

```
Enter a? 11
Enter b (not zero)? 3
a + b = 14
a - b = 8
a * b = 33
a / b = 3
a % b = 2
```

   A quick glance at **h3_1.s** makes it appear to be a quite lengthy program (there are 206 lines of code, but close to half of that is comments) but implementing much of the common code can be done by copying-and-pasting text and then modifying it. For examples, lines 78-80 perform the necessary instructions to print the prompt for a. Copy-and-paste lines 78-80 to replace line 89 and then modify the text to print the prompt for b. Copy-and-paste lines 83-86 to replace line 92 to read the integer b. Also, look at lines 125-138 which prints the output string "a + b = ", the value of *sum*, and then a newline character. Copy-and-paste these lines to replace lines 141-154 (for a - b), lines 157-170 (for a * b), lines 173-186 (for a / b), and lines 189-202 (for a % b).

   When typing assembly language code, it is important for the labels, mnemonics, operands, and comments of each instruction to be aligned in columns for maximum readability. The penalty for improper alignment or poorly formatted code will be 25% of the exercise points, i.e., the assigned points will be the points earned multiplied by 0.75. Also, update the header comment block with your name, email address, and section meeting time.

2. **(2 pts) Pseudoinstructions.** We discussed commonly-used pseudoinstructions in §2.3.8 of the Ch. 2 lecture notes. Remember that pseudoinstructions are software-emulated instructions (or *virtual* instructions) in the sense that the assembler replaces pseudoinstructions with one or more hardware instructions (or *physical* instructions) that achieve the desired effect. Suppose you are writing a MIPS assembler and you can create your own pseudoinstructions. Two common instructions found in many different instruction sets are one that increments an integer value in a register (to increment means add one) and one that decrements the integer value (subtracts one). You decide to implement two pseudoinstructions

   ```
   inc reg     # reg ← reg + 1
   dec reg     # reg ← reg - 1
   ```

   For both `inc` and `dec`, what would be the equivalent hardware instruction(s)?

3. **(5 pts) Instruction Encoding.** What would be the MIPS32 encoding of this instruction: `or $t8, $s3, $t5`? Write your answer as an 8-hexdigit integer, e.g., 0x12345678, and for full credit show/explain how your answer was obtained.

4. **(5 pts) Instruction Encoding.** What would be the encoding of this instruction: `lw $a1, -16($fp)`? Write your answer as an 8-hexdigit integer, e.g., 0x12345678, and for full credit show/explain how your answer was obtained.

5. **(13 pts)** You are hacking away on a MIPS32 system trying to understand how the code in a MIPS binary executable file works. You have loaded the binary into your debugger and are focused on the following memory addresses and the 8-hex digit values stored in them,

```
0x0040_0000:  0x3C011001
0x0040_0004:  0x34300068
0x0040_0008:  0x8E080000
0x0040_000C:  0x21080001
0x0040_0010:  0xAE080000
```

You know these values represent instructions because you know that 0x00400000 is the starting address of the text segment in MIPS programs. Decode each of these instructions and write the corresponding symbolic assembly language instructions. While playing around with the program in the debugger, you discover that memory addresses 0x10010000–0x10010003 appear to be allocated to an integer variable; let us call this variable x. Knowing that, please explain with one or two English sentences what the above five instructions are doing.

3 Submission Instructions

Create an empty folder named **cse230-f16-h3-*asurite*** where *asurite* is your ASU user name. If you worked with a partner, name your folder **230-f16-h3-*asurite1*-*asurite2***, where *asurite1* and *asurite2* are the ASURITE user names of both partners. Type the solutions to exercises 2–5 in a word processing document or text editor and convert the file into PDF format. Copy this PDF into the new folder, along with **h3_1.s**. There should only be two files in this folder. Then compress the folder creating a **.zip** archive named either **cse230-h3-*asurite*.zip** or **cse230-h3-*asurite1*-*asurite2*.zip**. Upload the zip archive to Blackboard using the homework submission link by the deadline, which is **4:00am Fri 30 Sep**. Consult the online syllabus for the late and academic integrity policies.