ECE 550
Homework 3
Fall 2017

For this homework you will be answering 2 "pencil and paper" questions, and doing one programming questions. Your submission should the following files:

- A pdf or plain text file with your answers to questions 1–2.
- The finished `strsort.c` and `strsort.s` file for question 3.

Please remember that **no Word Documents or rtf files** will be accepted.

Question 1: Assembly bits

For each of the following tasks (a) write a MIPS assembly instruction to perform the task, and (b) convert the MIPS assembly you wrote to its numeric encoding.

- Add `$a1 + $s2` and store the result in register `$t2`

- Load an unsigned byte from the address `$sp + 32` into register `$v0`

- Branch if `$t0` is not equal to `$s0` to label `endIf` (which is 88 instructions away from the current instruction).

- Call the function `edgarware`, whose address is `0x04238270`

Question 2: Stack Frame

Suppose you have a function which you want to write in MIPS assembly. This function writes to the following registers: `$s0`, `$s1`, `$s2`, `$t0`, `$t1`, `$t2`, `$a0`, `$a1`, `$v0`. Of these, `$s0`, `$t0`, and `$t1` are live across calls to other functions.

1. Write the MIPS assembly code to setup your stack frame, and save the registers you need to save at the start of function.

2. Write the MIPS assembly code to restore the registers you need to and clean up the stack frame at the end of this function.

3. Describe where else you need to save and restore registers, and state which ones you need to save and restore.

4. Draw a diagram of your stack frame (as you have set it up). Include the stack pointer and frame pointer, and show what register is saved in each slot. Finally, indicate where the caller's stack frame exists.

Question 3: Assembly Programming

For this problem, you will be writing MIPS assembly to sort an array of strings.

The first step in this task is to **write the code in C.** Open the included `strsort.c` file, and fill in the `strsort` function. This function takes an array of strings (i.e., a `char**`), and a count of how many strings are in the array. You should sort the array "in place" (that is, modifying ordering of the `char*`s in the array) into ascending order (smallest string first). You can use the `strcmp` function to compare two strings, and figure out what order they are in. See `man strcmp` for details.

Our provided code also has a main function which will help you get started testing your code (feel free to add more test cases as you feel you need).

The second step in this task is to **write the code in MIPS Assembly.** Open the included `strsort.s` file, and fill in the `strsort` function. This function should work exactly like your C version, which you should use as a guide. Note: You may NOT use a MIPS C compiler to do this translation, you must write the assembly yourselves. We have provided an implementation of `strcmp` as well as a `main` to test your code.

You will need to follow the MIPS calling conventions (which were discussed in lecture, and are described in your book) to make your function work properly with the other function (main and strcmp).

You should not change the `strcmp` function that we provided.