"See, it's not magic. I'm like the anti-Harry Potter." – Craig Zilles

Preliminaries

1. You have the following files in your Lab7 github repository in a directory called `QtSpimExercise`: `max.c`, `max.s` and `max_test.s`.
2. Set your editor's tab width to 8 spaces, otherwise the above assembly files will not look at all as intended. Now would also be a good time to get MIPS syntax highlighting set up in your editor – see the MIPS editor support in the class website to see how to do so: `https://wiki.cites.illinois.edu/wiki/display/cs233fa18/MIPS+editor+support`

Example Code

```
unsigned max(unsigned *array, unsigned size) {
    unsigned currentMax = array[0];
    for (unsigned i = 1; i < size; ++ i) {
        if (array[i] > currentMax) {
            currentMax = array[i];
        }
    }
    return currentMax;
}
```

What you need to do

This is a short exercise intended to familiarize you with QtSpim, which you'll be using for the MIPS labs. Technically, you'll be using QtSpimbot for the SPIMbot parts, but the two programs have an almost identical interface.

Your task is to run and debug a small MIPS function intended to find the maximum element in an array of unsigned integers. The C code being translated can be found in `max.c`, and it should be pretty straightforward. The file `max.s` contains the translated assembly, and `max_test.s` contains a few test cases for it. Read through `max.s` to understand what it's doing.

Running QtSpim: To start with, navigate to the directory where the files for this exercise are located. Open the files in your editor of choice, and then run QtSpim with the following command:

```
QtSpim -file max.s max_test.s &
```

Dealing with syntax errors: The first thing you'll be greeted with is a syntax error. The message will tell you the file and line number, and when you dismiss the error alert, you'll notice the bottom of the QtSpim window also displays the error message, with a caret pinpointing the exact location of the error. What's the problem here? Fix it and re-open the files.

The easiest way to re-open is to close QtSpim and then rerun the QtSpim command. If you want to do it from within the GUI, you should use ***File>Reinitialize and Load File*** to load the first file and then ***File>Load File*** to load all subsequent files.

Once you've re-opened, the syntax error should have gone away, so we can actually try to run the code now.

Running your code: The test code in `max_test.s` simply calls `max` on the three arrays at the top of the file, so it's clear what the intended outputs are. There should be a toolbar at the top of the QtSpim window with a Play button on it; click this button to run the code. Unfortunately, all we get for our trouble is an ***Error*** dialog, so we need to do some fixing.

Fixing the exception: The first ***Error*** dialog informs us that an exception occurred at PC `0x004000fc`. Click ***OK*** to dismiss it, and we'll get another one explaining the cause of the exception: "Bad address in data/stack read: `0x00000000`". Click ***Abort*** this time to avoid triggering any more exceptions. You'll notice the bottom of the QtSpim window also displays the exception message, in case you missed it the first time around.

Those of you who have taken or are taking 225 will be intimately familiar with segfaults, and the exception we're getting is the QtSpim version of a segfault – we're trying to read from an invalid address, so it's yelling at us. The PC in the exception message tells us which instruction is causing the segfault, and if you look at the Text window, the leftmost column contains instruction addresses and the part after the semicolon contains the original instruction, so it's easy to map instruction addresses to file lines.

In this particular case, address `0x004000fc` corresponds to line 21 of `max.s`, and if you look at that line, we're using register `$s1` as the source of our load. The register window on the left tells us that `$s1` has the value 0, hence the exception. What are we doing wrong here, and how can we fix it?

Once you've fixed the problem, re-load the files and run the code again; this time, there should be no exception, and a console should pop up with the outputs from the code. Our function calculates the correct maximum for the first two arrays, but for the third, it returns `0x12345678` even though the maximum element is `0xdeadbeef`, so we're going to have to use the debugger to figure out what's going wrong.

Using the debugger: Since the first and second calls to `max` worked correctly, it makes sense to begin debugging with the third one. This is on line 117 of `max_test.s`, so let's put a breakpoint on that line. To do so, you'll have to locate it in QtSpim. Fortunately, QtSpim displays line numbers and their associated file names, so scroll down until you see `max_test.s:117` (i.e. line 117 of `max_test.s`). ***Run Simulator>Clear Registers*** to get QtSpim back to its initial state (or just close and re-open it), right click on this line and select ***Set Breakpoint***, and then run the code. You'll see a ***Breakpoint*** dialog pop up; click on ***Single Step*** to start stepping.

The second-last button on the toolbar should have three lines with 1 2 3 next to them; click on this button to step. Before executing an instruction while stepping, you should work out the correct behavior of the instruction mentally, so that you can check if it behaves as expected when executed. As you step, the register window on the left will get updated, so you can precisely track the execution of the code.

Start stepping through the function call, paying close attention to the control flow and the register

values, particularly `$v0`, which holds `currentMax`, and `$t1`, which holds `array[i]`. Does the first iteration of the loop behave like you expect it to? What about the second and the third? What's going on here, and how can we fix it?

Once you've corrected the problem, re-open the files and run the code again. All three tests should produce the correct values now. Hooray, we're done!

`max_test.s` uses some neat MIPS and QtSpim constructs which are worth reading and understanding. Feel free to post on Piazza if you need clarification.