EE 108B Lab Assignment #1
MIPS Assembly Programming
Due: Tuesday, January 29, 2013

1. Introduction

Throughout these labs, you will be designing a processor that can execute programs written in MIPS assembly. Once the hardware support for software has been built, a large variety of tasks can be performed without having to constantly modify the hardware design as in EE108A. To demonstrate how software can perform tasks previously done with hardware, you will be implementing the classic game of Pongin MIPS assembly.

2. Requirements

You will implement a self-playing version of the game "Pong" using the SPIM simulator.

IMPORTANT: Please implement Lab 1 on the **corn** machines. The corn cluster has all the software that you will need to implement the labs, (SPIM and python-tk).
You may also install these packages on your own computer if you are running a Linux variant or OS X. They are available with apt-get and MacPorts, and the lab's functionality has been verified on OS X and Ubuntu. Come to office hours if you would like help in doing this.

To log into the corn machines use:

```
ssh yoursunetid@corn.stanford.edu -Y
```

(-Y enables X window forwarding for the display)

There should be one paddle at the left edge of the screen that can move vertically. Meanwhile, a ball will be bouncing off the edges of the screen, and the paddle must move so that it prevents the ball from reaching the left edge of the screen. The game ends when the ball reaches the left edge. Your implementation must use no more than 256 assembly instructions, which will be enforced by a special command line argument to the SPIM simulator.

After completing these minimum requirements, you may choose to implement any of the following extensions:

- Have paddles on all sides of the screen, all tracking the ball.
- Implement your own version of Breakout.
- Allow the user to control the paddle(s) by typing on `stdin`.

Review the comments in the starter code thoroughly for more information on these extensions and other details.
If you do not implement the user control extension, your game should play itself indefinitely until the user forces it to close; however, you should still implement the logic that causes the game to end when the ball hits the left edge (which it never will).

3. Implementation Details

The file pong.s contains a skeleton implementation that demonstrates how to draw on the screen and how to end the game. You should read all comments in this file before beginning your implementation. Put your group members' names and email addresses in the file where indicated. You may want to keep a copy of the original file to refer to later. (You should delete the explanatory comments in the file you submit, adding your own comments that describe your implementation.)

Also included is a Makefile with a few convenient options for debugging your code:

`make` – this will just run your MIPS program and pipe its output to the display

`make debug` – this will run your MIPS program, pipe its output to the display, and print a line every time your program draws a block on the screen showing the coordinates and color

`maketofile` – this will run your MIPS program and store its output to a text file so you can inspect its contents

`makefromfile` – this will send the text file generated by "make tofile" to the display

NOTE: Some systems will disable execute permissions when copying files from a remote source. To enable execute permissions for the display program, use:

```
chmod +x pong_display.py
```

This allows the display program to be executed with:

```
./pong_display.py
```

You may modify the Makefile and pong_display.py if you like, but you must demonstrate your implementation using the original Makefile and pong_display.py. Specifically, you must still use the restriction that the text segment is limited to 1024 bytes (256 instructions).

4. Submission

You must demonstrate your implementation of Pong during office hours on the due date. A sign-up sheet will be posted a few days before this date. You must also submit a lab report and your source code at 11:59PM on the due date on CCNet. Guidelines for lab reports in this class are included. For this lab, you do not need an appendix 1 (simulations), just submit your pong.s file and your report PDF in a single zip archive.

The course staff will read the code you submit, and may run it to verify its functionality, but only the functionality as demonstrated during office hours will be graded. Extra features implemented after the demonstration will not be considered. This is to give you time to focus solely on producing a polished lab report by the deadline. Adding comments to your code after the demo is also acceptable.

Additional Question for the Lab 1 Report:

What are the advantages and disadvantages of the software approach to this specific project? Give specific examples of things that are more and less efficient in the software implementation. Even if you did not implement any extensions, think about what types of extensions would be easier to do with the software approach, and what types would be easier to do with the hardware approach. (This should be qualitative, you do not need to demonstrate with calculations why some feature is more efficiently implemented in software or hardware, just explain your reasoning.)

5. Grading Rubric

Lab Report:
5 points – following report guidelines (no missing sections, title page, etc.)
10 points – well thought out answer to hardware vs. software question
10 points – thorough design discussion

Code:
5 points – turning in your code
10 points – following MIPS conventions (clearly documenting any deviations)
10 points – clear comments where needed (make sure the code is easy to understand)

Demonstration:
50 points – working implementation of minimum functionality (no bugs or visual errors)
+2 points – paddles on all sides
+4 points – Breakout
+4 points – user control

Total: 100 points (110 possible with extensions)