Brian Sherman & Nathaniel Leclerc

EECS 314: Computer Architecture

Final Project Report

1 May 2015

**Problem Statement:**

Our project involved the use of MIPS to program a simple robot to respond to external commands for movement. The program we are using uses an USB Microsoft Xbox 360 Gamepad Controller as the input device, and outputs commands to be read by a simple robot. We have successfully managed to implement simple movement controls, but we have a good framework set up to increase the functionality of the robot if necessary in the future.

**Major Challenges:**

One of the biggest challenges for our project was the cost of creating a robot. After doing some estimates, we found that, unfortunately, neither of us were able to afford the cost of creating a robot in ThinkBox. Because of this, we could only simulate sending commands to a robot, and were unable to test these commands on an actual, working prototype. This was a disastrous setback, but we pressed on regardless.

Another challenge for our project was in finding a way to receive commands sent via the USB Microsoft Xbox 360 Gamepad Controller. MIPS, being a low-level language, does not have an easy way to capture USB data, and attempts to find and interpret the raw data sent by commands from the USB Microsoft Xbox 360 Gamepad Controller were unsuccessful.

**Key Components:**

To address the issue of not having a prototype robot, we created a simulation of sending and interpreting commands. Based on the button(s) pressed, MIPS generates a command or series of commands in a file. This file would have been sent to the robot, if we had been able to create a working prototype. To simulate receiving input commands, we created another MIPS program to read the information from the file, consume the commands it reads, and print it to the console.

To solve the issue of receiving controller input, we utilized a C# library to capture the controller commands. We were hesitant at first to use C# code, as this was primarily a MIPS project. In the end, we decided to go through with it, but took care to minimize the impact of the C# code. The C# code does nothing more than interpret commands from the USB Microsoft Xbox 360 Gamepad Controller and call the appropriate functions in SPIM.

These key components comprise an effective framework for interpreting other functions and providing ways to signal the robot to perform other tasks. Because we have everything needed for the basic functionality of robot control (save for the robot itself), this project has been a success.

**Integration of Components**

These components are integrated through MIPS syscalls and a C# library for interpreting commands. The C# program will create a command line call to execute the appropriate MIPS programs, depending on the input given by the user. The MIPS functions use syscalls to provide data flow to the hypothetical robot, both in writing to the file to be sent and interpreting the file (on the user end, as the robot itself does not exist).

**User Interface**

The primary user interface for this project is a USB Microsoft Xbox 360 Gamepad Controller. After running the executable provided, the program will listen for commands on the USB Microsoft Xbox 360 Gamepad Controller. By receiving move commands input into the controller’s directional pad, the program will run appropriately. Using the Back/Select button will exit the program.

**Contributions**

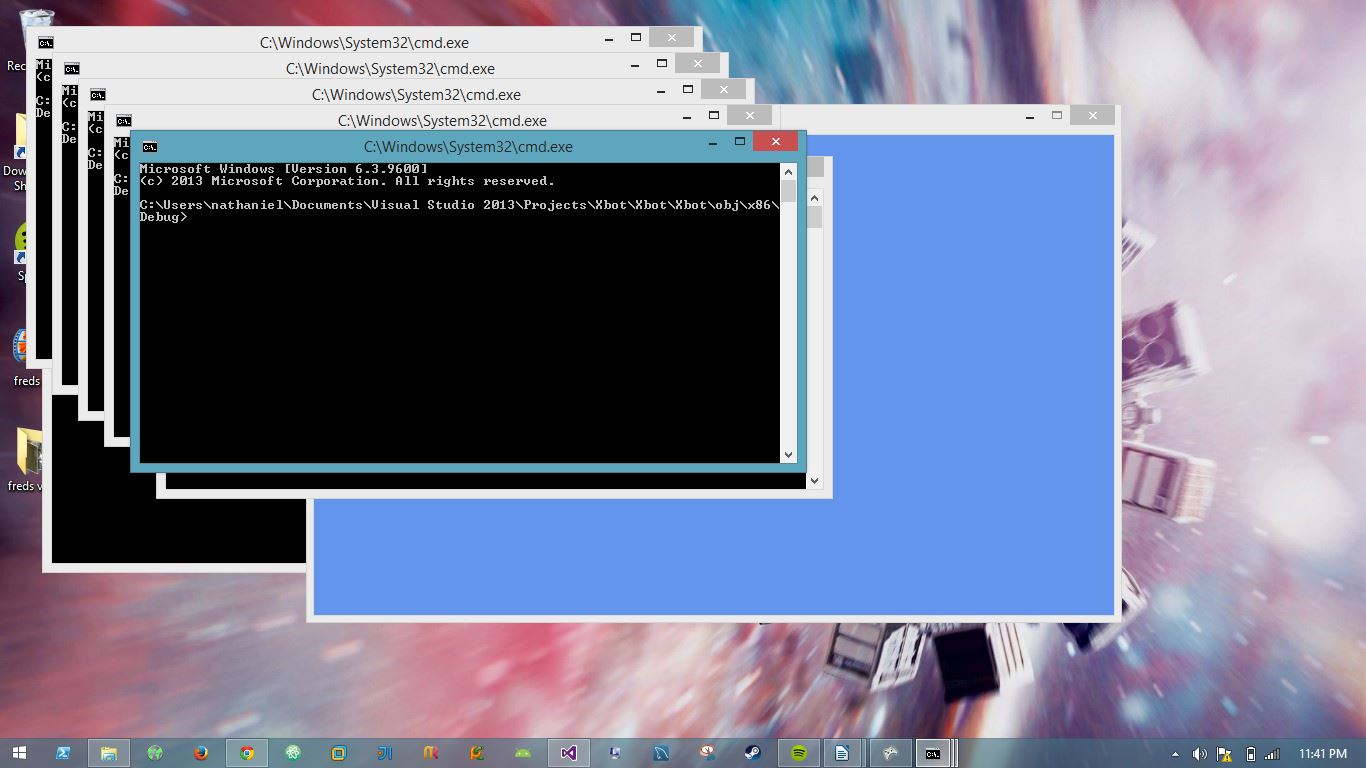
Nathaniel Leclerc

* C# interface
  + Program, Game1

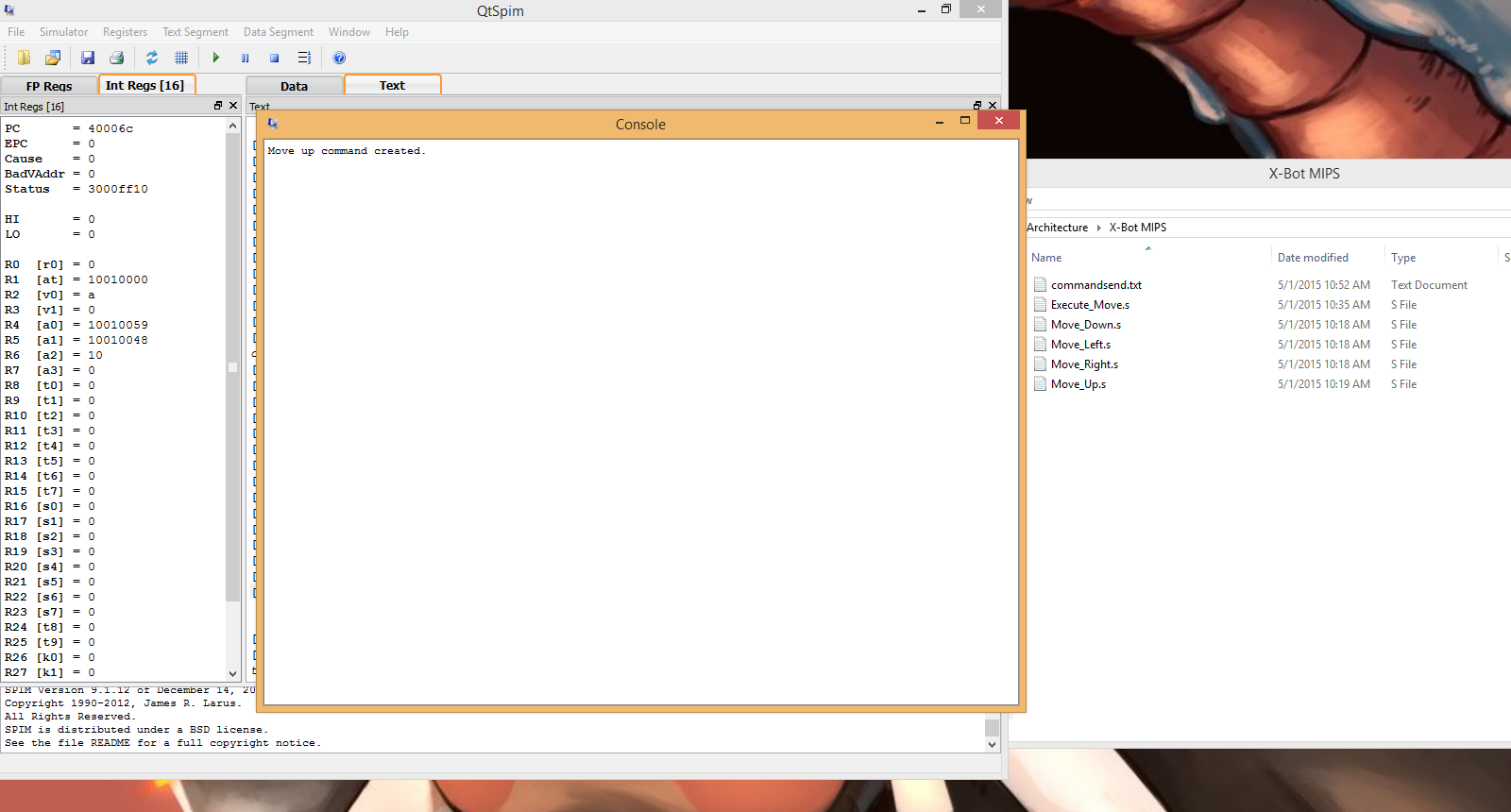
Brian Sherman

* MIPS Programs
  + Move\_Up, Move\_Down,…
  + Execute\_Move

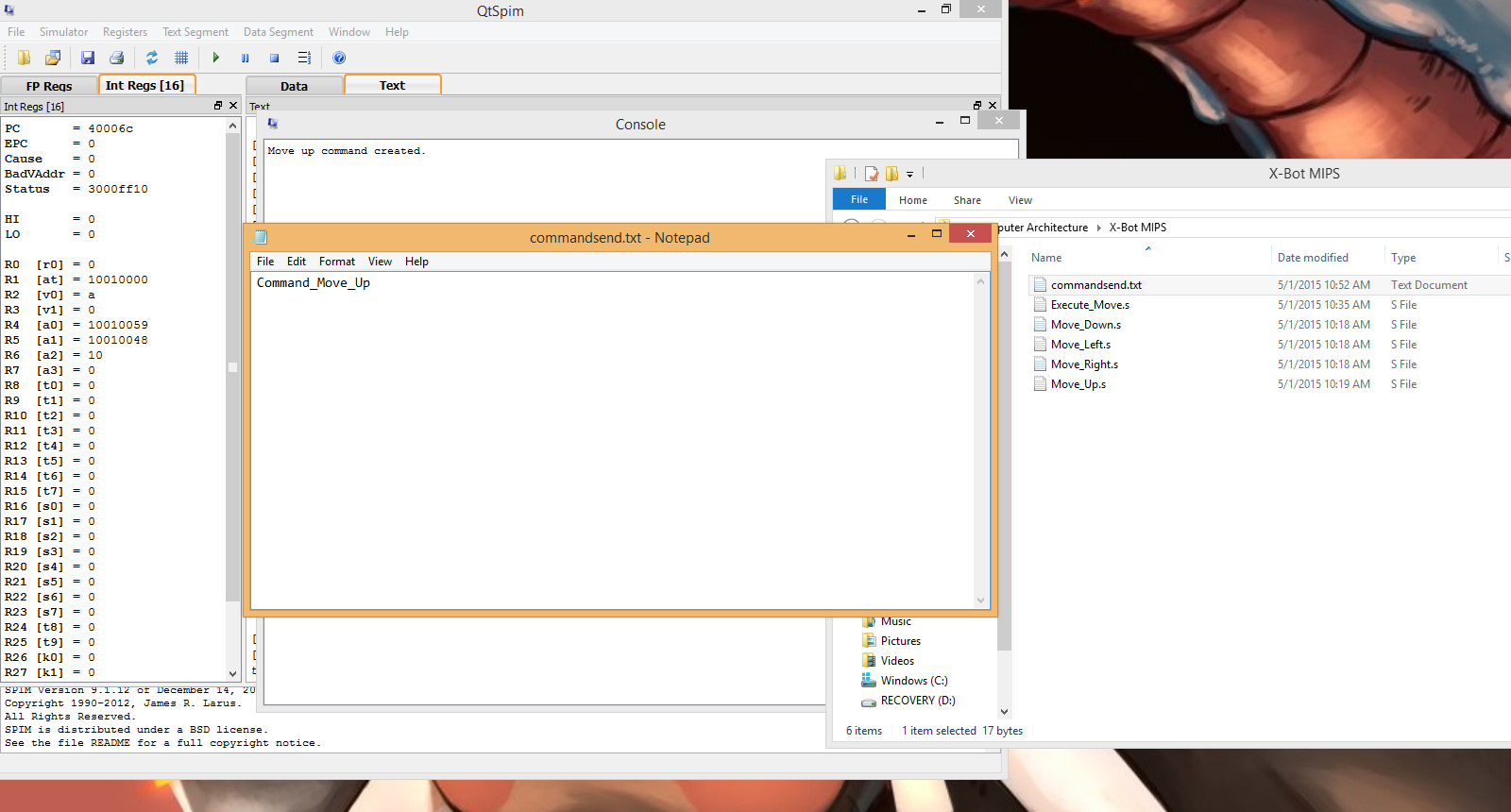
**Sample Runs:**



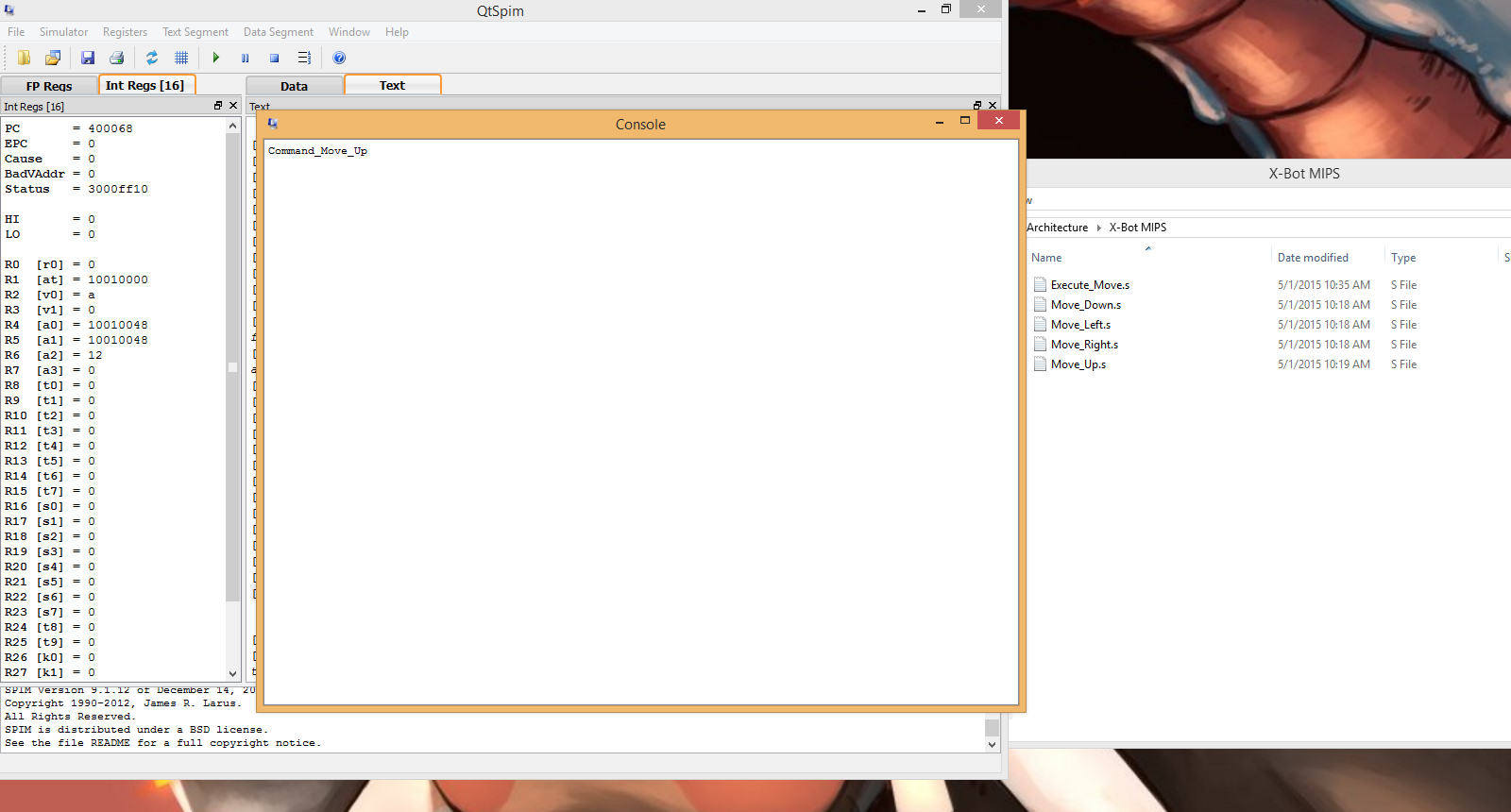
This is the interface that is opened when the program is first being run. A command prompt is opened (normally this would be hidden so it doesn’t keep popping up and annoying the user, but we left it visible for demonstration purposes) whenever a button on the controller is captured. This prompt calls a MIPS function.



This is a screen showing the output of one of the commands from one of the MIPS functions. Here, we are telling the robot to move up by generating a txt file with the command to move up. We are also printing to the console for demonstration purposes.



This is a screenshot of the text file that is being generated. This text file is fed to another mips program on board the robot which moves it according to the commands here.



This is a screenshot of Execute\_Move being run. Here you can see that it is consuming the text file. We are also printing to console here, just for testing purposes.