EECS 314 - Project Report

Haley Eisenshtadt, Jason Kuster, Ian Lavelle, Joseph Satterfield, James Wright

April 29, 2015

1 Problem Statement

While learning MIPS in EECS 314, our group noticed a lack of fun, interactive tools to make learning MIPS easier. There are helpful demonstrations in the textbook and you can always open up QtSpim and play around, but there wasn't anything particularly graphical and interactive to help people learn MIPS. Our group wanted to put together an Android app to make the process of learning MIPS easier. The tool we settled on to build this was Unity, a cross-platform game engine which several of us had used before which made game development easy.

2 Major Challenges

There were a couple of challenges we ran up against while developing this set of games.

1. User interface design in Unity. Unity recently switched to using a new user interface system, and setting up all of the click interactions within MIPS simulator was non-trivial in the new system, requiring nesting function definitions within functions. Also, setting up a system to let the various operands know what types they can accept was a challenge as it essentially involved keeping lists of which slots would accept various operand types, and then forcing them to correctly highlight.

2. In the original game Bejeweled (and its clone Candy Crush), the goal is to simply match lines of identical pieces, regardless of order. Our initial thought was to just make Bejeweled and then change the sprites to make it MIPS themed, but becasue MIPS relies on a specific order of different items, we were forced to rethink large sections of code. Now, there are specific OP pieces which tell the game how many pieces to check and in what order the different inputs should be.

3 Key Components

Our project has three key components which come together to make the MIPS learning process easier. We have the MIPS Simulator, which is a graphical MIPS simulator in the style of ALICE for Java, we have MIPS Bejeweled, a twist on the classic Bejeweled game

which helps teach people how MIPS instructions work, and we have the Pipeline Bakery which presents the MIPS pipeline in a fun, graphical way.

3.1 MIPS Simulator

The MIPS Simulator allows the user to use an intuitive user interface to create a sequence of instructions and view the results of those instructions. You start by adding new instructions via the instructions panel and then filling in the operands using the various operand kind panels. Operands will only let you place themselves in valid instruction positions to help reduce the amount of error that can occur.

3.2 MIPS Bejeweled

MIPS Bejeweled takes the classic bejeweled game and adds the twist of only completing rows when those rows create a valid MIPS instruction. In order to make this work we needed to remove the requirement in Bejeweled of reversing a move if it doesn't make a valid combination, because with the added complexity of MIPS instructions the game was impossible to play. We feel that this is an acceptable compromise in the name of usability. We also changed out the typical falling physics of classic Bejeweled for a swapping mechanism that encourages the player to think ahead.

3.3 Pipeline Bakery

4 Integration

The integration of the three games was relatively simple. Each game was made seperately and then merged with GitHub into one folder. James went through and resolved any file conflicts in that arose from the merge. When together, each game takes place in what Unity calls a scene, including the main game-select screen. When the player selects a game to play, the scene of the desired game is loaded up, and the user plays it as normal. To go back, each game has another button that returns the user to the game-select screen.

5 User Interface

This project has three main UIs, one for each game, in addition to a main menu which is used to navigate to the individual projects.

5.1 MIPS Simulator

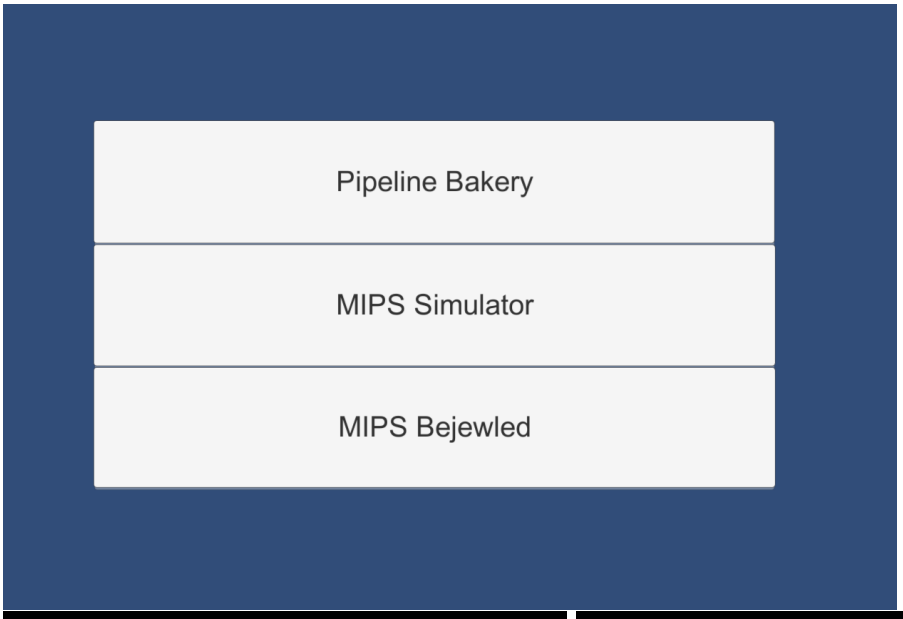
5.2 MIPS Bejeweled

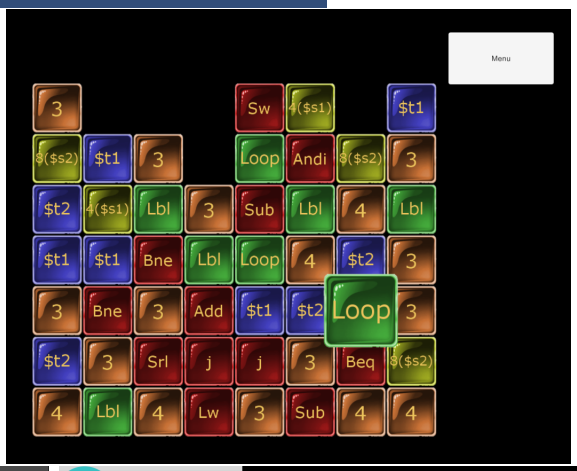
The UI for this was quite basic. Each gem has an input or an OP on it, and is colored according to what type it is. Blue is registers, yellow is a register with a shift, orange is an immediate value, red is an operation, and green is a jump address. The values on the pieces

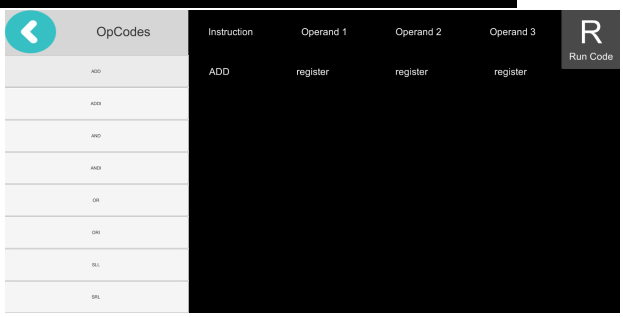
has no bearing on the game, but is just there as an example of what these values might look like in MIPS code. The first selected piece becomes larger than the others, indicating it has been selected and can be swapped with any adjacent piece.

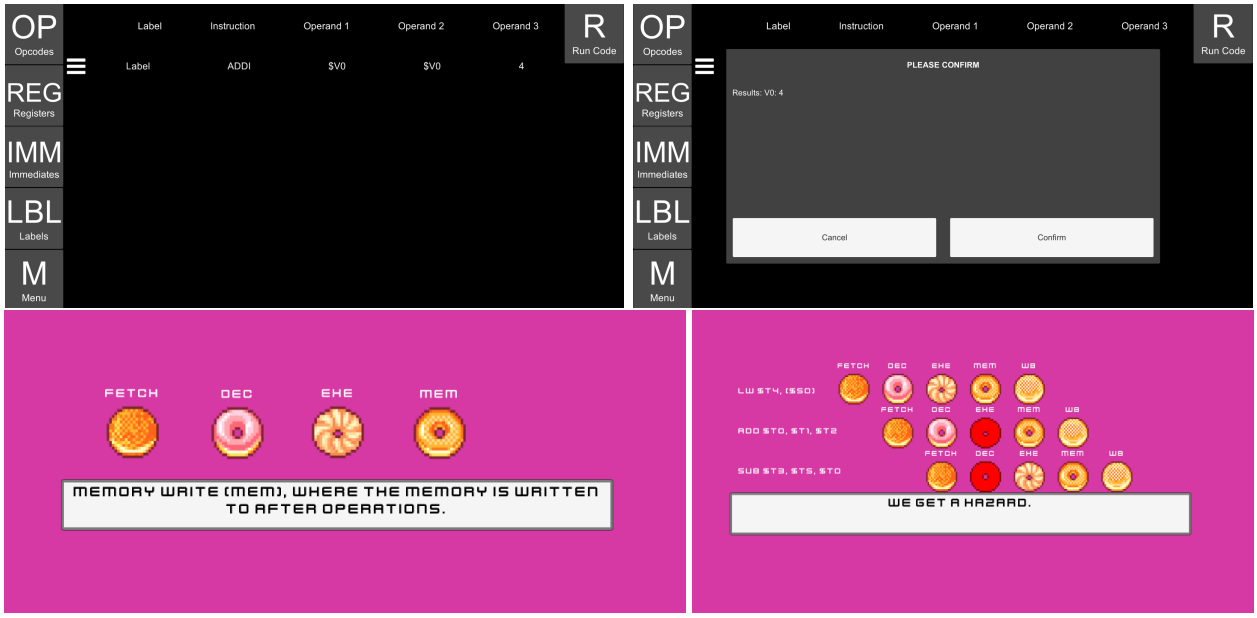
5.3 Pipeline Bakery

6 Snapshots









7 Individual Contributions

- Haley Eisenshtadt

  Pipeline Bakery Design/Implementation

- Jason Kuster

  Bejeweled Block Physics

  Project Report Lead/Compilation

- Ian Lavelle

  Bejeweled Base Implementation

  Bejeweled Instruction Logic

- Joe Satterfield

  MIPS Learner Simulator Back-End Implementer

- James Wright

  MIPS Learner User Interface Desgin and Implementer