Team Doge: Binary Converter

Micah Hutchens Kellie LaFlamme

hmi1@umbc.edu kl4@umbc.edu

Bryan Stetson Alexander Wooten

stetson2@umbc.ed awooten1@umbc.edu

Department of Computer Science and Electrical Engineering

University of Maryland, Baltimore County

30 April 2014

Note: Report is pretty short as of this draft, but will be more in depth once we're done and have gathered test data, figures, and what not.

**Abstract**

The goal of this project was to create an assembly language program that takes in a binary number and converts it to decimal, octal, and hexadecimal. (placeholder for actual abstract once project is done)

**1. Introduction**

By working on this project, we hope to develop a better understanding of assembly programming and computer architecture through the development of a fully functional binary conversion program.

For this project, we elected to use the reduced Microprocessor without Interlocked Pipeline Stages (MIPS) instruction set [1]. In order to simulate a MIPS-backed machine, we used SPIM, a MIPS simulator that mimics the architecture used by R2000/R3000 processors. In particular, we used the latest SPIM build, QtSpim, due to the fact that it runs cleanly on Windows, Mac, and Linux systems [2].

**2. Previous Work**

Prior to taking this course, each of us had developed assembly programs in x86 for CMSC313 [3]. While MIPS does have some fundamental differences when compared to x86, our prior experience served to provide us with a general understanding of how to program in assembly and how to think like a computer [4].

During our initial research, we quickly discovered that MIPS programs dealing with base conversion are extremely common place and make for relatively common school assignments. For example, a quick Google search reveals a program developed in 2008 that takes in a decimal number and converts it to a user-specified base [5]. What makes our project unique is that we start off with a binary number that is then converted to other bases as opposed to starting with a decimal number. This presents some unique challenges regarding user input and storing the initial value.

**3. Methodology**

Our methodology for this project was fairly straight forward. We divided up the different bases that we were required to convert the binary input into amongst our team members and plan on combining them into a single source file. Version control was managed through GitHub [6].

More details about the specific methodology of the code itself will be added once the code is actually complete.

**4. Discussion**

Not much to discuss at this stage. This section will be completed once the code is up and running.

**5. Conclusion**

Conclusion will be added once we've concluded working on the project.

**References**

[1] Frenzel, James. "MIPS Instruction Reference." *uidaho.edu*. University of Idaho, 10 Sep 1998. Web. 30 Apr 2014. <http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html>.

[2] Larus, James. "SPIM A MIPS32 Simulator." *cs.wisc.edu*. University of Wisconson-Madison, n.d. Web. 30 Apr 2014. <http://pages.cs.wisc.edu/~larus/spim.html>.

[3] Chang, Richard. "CMSC313, Computer Organization & Assembly Language Programming, Spring 2013." *csee.umbc.edu*. University of Maryland Baltimore County, 28 Jan 2013. Web. 30 Apr 2014. <http://www.csee.umbc.edu/~chang/cs313/>.

[4] Toal, Ray. "x86 Assembly Language Programming." *cs.lmu.edu*. Loyola Marymount University. Web. 30 Apr 2014. <http://cs.lmu.edu/~ray/notes/x86assembly/>.

[5] Papadopoulos, Andreas. "MIPS Convert a number to another base ." *http://akomaenablog.blogspot.com*. N.p., 18 Oct 2008. Web. 30 Apr 2014. <http://akomaenablog.blogspot.com/2008/10/mips-convert-number-to-another-base.html>.

[6] Stetson, Bryan. "CMSC-411-Project." *github.com*. Team Doge, 30 Apr 2014. Web. 30 Apr 2014. <https://github.com/bryanstetson/CMSC-411-Project>.