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

7 May 2014

**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. Once our application was developed, we tested it's time efficiency against a MIPS program that takes in a decimal and converts it to other bases and the C equivalent of our application.

Keywords: MIPS, SPIM, C, base, conversion, binary, octal, hexadecimal, efficiency, comparison

**1. Introduction**

By working on this project, we hoped to develop a better understanding of assembly programming and computer architecture through the development of a fully functional binary conversion program, as well as to test the efficiency of our code against other previously implemented approaches.

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].

Once the code was completed and fully operational, we ran multiple timed tests using our code, equivalent C code, and outside code that converted from decimal instead of binary. Using this data, we hoped to judge the efficiency and usefulness of our application.

**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 at a basic level using computer logic [4].

During our initial research, we quickly discovered that MIPS programs dealing with base conversion are extremely commonplace and make for relatively common college-level 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.

In addition to MIPS code, we also found C code that serves a similar function to our application. This code gave us a basis of comparison for testing our program [6].

**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 later combined them into a single source file. Version control was managed through GitHub [7].

Due to the fact that base conversion is by no stretch of the imagination a novel problem, we elected to use established algorithms rather than attempt to develop our own. For each conversion, we used standard algorithms described in Alexander Bogolmony's "Implementation of Base Conversion" [8].

**4. Discussion**

Before testing anything else, we had to ensure our program's functionality. In order to do this, we used it to convert various binary numbers, ranging from single digits all the way up to sixteen digits, then checked the results using an online calculator [9]. We also made sure to test for error checking, proper looping, and exit conditions.

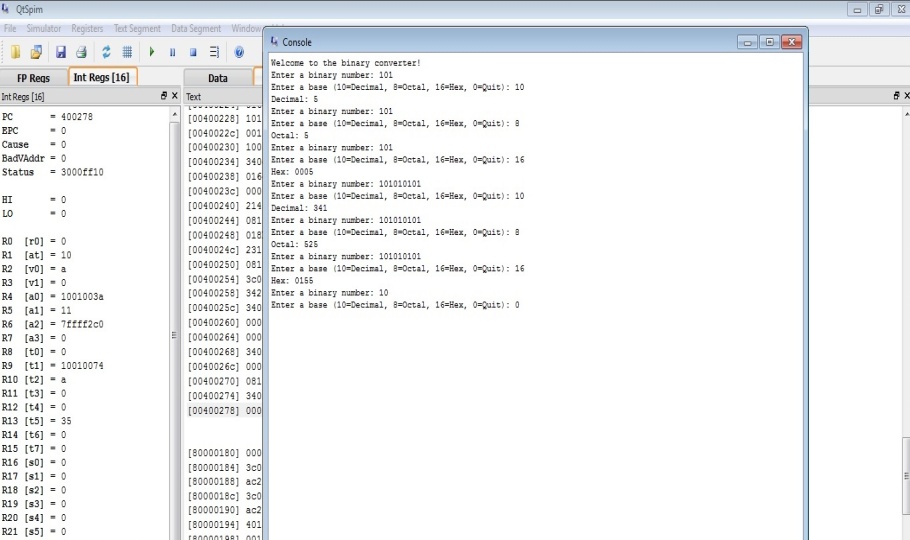


Figure 1: Example output

For our first analytical test, we compared our application with one that converts from decimal to other bases. Although the other program is capable of converting to any base, we only tested conversion to binary, octal, and hexadecimal Additionally, we slightly modified the program so that it looped and exited in a similar manner as ours. This was done to ensure a fair comparison. Our initial hypothesis was that the conversion from decimal would be faster, due to the fact that the code used fewer instructions.

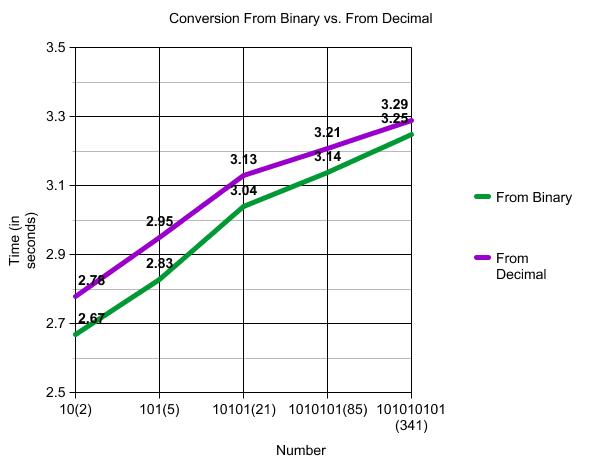


Figure 2: Conversion From Binary vs. From Decimal

Contrary to our prediction, our program completed its conversions in less time. Our theory as to why this occurred is that it is due to the fact that the computer architecture naturally stores everything in binary.

Next, we tested our application against the C language program. Again, we predicted that our program would be less time efficient, this time because C is a high-level language that uses an advanced compiler. In this case, the C code was compiled using the Gnu Compiler Collection, more commonly known as GCC. GCC works by compiling C code directly into assembly code [10]. In the case of our test machine's architecture, the assembly language was Intel x86. Like in the previous test, the C code was modified slightly so that it would run in a comparable fashion to our MIPS code.

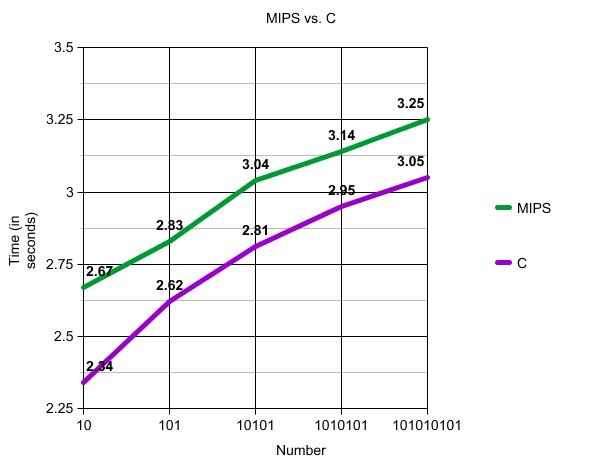


Figure 3: MIPS vs. C

This time, our hypothesis was correct. The C program proved to be faster than our equivalent MIPS application.

**5. Conclusion**

Upon completion of our project, we can proudly say that our goals were met. We successfully created a fully-functional binary conversion application, becoming more well-rounded software engineers during the development process. We also succeeded in designing and executing trials that pitted our program against others in simulations of real-world use. In the end, we concluded that our program is more efficient then code that converts from decimal, but less efficient then an equivalent C program.

**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] "C > Beginners Lab Assignments Code Examples." C code example. N.p., n.d. Web. 7 May 2014. <http://c.happycodings.com/beginners-lab-assignments/code36.html>.

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

[8] Bogomolny, Alexander. "Implementation of Base Conversion." Algorithms from Interactive Mathematics Miscellany and Puzzles. N.p., n.d. Web. 7 May 2014. <http://www.cut-the-knot.org/recurrence/conversion.shtml>.

[9] "Number base converter." - Online tools. N.p., n.d. Web. 7 May 2014. <http://www.tools4noobs.com/online\_tools/base\_convert/>.

[10] "GCC, the GNU Compiler Collection." - GNU Project. Free Software Foundation, Inc, 5 May 2014. Web. 7 May 2014. <http://gcc.gnu.org/>.