**MARS - Mips Assembly and Runtime Simulator**

**Release 4.5**

**August 2014**

**Introduction**

MARS, the **M**ips **A**ssembly and **R**untime **S**imulator, will assemble and simulate the execution of MIPS assembly language programs. It can be used either from a command line or through its integrated development environment (IDE). MARS is written in Java and requires at least Release 1.5 of the J2SE Java Runtime Environment (JRE) to work. It is distributed as an executable JAR file. The MARS home page is **http://www.cs.missouristate.edu/MARS/**. This document is available for printing there.

As of Release 4.0, MARS assembles and simulates 155 basic instructions of the MIPS-32 instruction set, approximately 370 pseudo-instructions or instruction variations, the 17 syscall functions mainly for console and file I/O defined by SPIM, and an additional 22 syscalls for other uses such as MIDI output, random number generation and more. These are listed in separate help tabs. It supports seven different memory addressing modes for load and store instructions: label, immed, label+immed, ($reg), label($reg), immed($reg), and label+immed($reg), where immed is an integer up to 32 bits. A setting is available to disallow use of pseudo-instructions and extended instruction formats and memory addressing modes.

Our guiding reference in implementing the instruction set has been *Computer Organization and Design, Fourth Edition* by Patterson and Hennessy, Elsevier - Morgan Kaufmann, 2009. It summarizes the MIPS-32 instruction set and pseudo-instructions in Figures 3.24 and 3.25 on pages 279-281, with details provided in the text and in Appendix B. MARS Releases 3.2 and above implement all the instructions in Appendix B and those figures except the delay branches from the left column of Figure 3.25. It also implements all the system services (syscalls) and assembler directives documented in Appendix B.

The MARS IDE provides program editing and assembling but its real strength is its support for interactive debugging. The programmer can easily set and remove execution breakpoints or step through execution forward or backward (undo) while viewing and directly editing register and memory contents.

**Questions and Comments**

Send MARS questions and comments to Dr. Pete Sanderson at PSanderson@otterbein.edu or Dr. Ken Vollmar at KenVollmar@missouristate.edu. We will respond as quickly as we can but as teaching professors do not have as much time to work on this project as we would like during the school year. We presented papers on MARS at the 2005 CCSC:MW conference and the 2006 SIGCSE Technical Symposium. We presented a tutorial session on MARS at the 2007 CCSC:CP conference and the Tutorial handout is available from the MARS homepage.

This document is available for printing on the MARS home page **http://www.cs.missouristate.edu/MARS/**.