**MARS - Mips Assembly and Runtime Simulator**

**Release 4.5**

**August 2014**

**Interactive Debugging Features**

MARS provides many features for interactive debugging through its Execute pane. Features include:

* In *Step* mode, the next instruction to be simulated is highlighted and memory content displays are updated at each step.
* Select the *Go* option if you want to simulate continually. It can also be used to continue simulation from a paused (step, breakpoint, pause) state.
* Breakpoints are easily set and reset using the check boxes next to each instruction displayed in the Text Segment window. *New in Release 3.8:* You can temporarily suspend breakpoints using Toggle Breakpoints in the Run menu or by clicking the "Bkpt" column header in the Text Segment window. Repeat, to re-activate.
* When running in the *Go* mode, you can select the simulation speed using the Run Speed slider. Available speeds range from .05 instructions per second (20 seconds between steps) up to 30 instructions per second, then above this offers an "unlimited" speed. When using "unlimited" speed, code highlighting and memory display updating are turned off while simulating (but it executes really fast!). When a breakpoint is reached, highlighting and updating occur. Run speed can be adjusted while the program is running.
* When running in the *Go* mode, you can pause or stop simulation at any time using the *Pause* or *Stop* features. The former will pause execution and update the display, as if you were stepping or at a breakpoint. The latter will terminate execution and display final memory and register values. If running at "unlimited" speed, the system may not respond immediately but it will respond.
* You have the ability to interactively step "backward" through program execution one instruction at a time to "undo" execution steps. It will buffer up to 2000 of the most recent execution steps (this limit is stored in a properties file and can be changed). It will undo changes made to MIPS memory, registers or condition flags, but not console or file I/O. This should be a great debugging aid. It is available anytime execution is paused and at termination (even if terminated due to exception).
* When program execution is paused or terminated, select *Reset* to reset all memory cells and registers to their initial post-assembly values. In fact, Reset is implemented by re-assembling the program.
* Memory addresses and values, and register values, can be viewed in either decimal or hexadecimal format. All data are stored in little-endian byte order (each word consists of byte 3 followed by byte 2 then 1 then 0). Note that each word can hold 4 characters of a string and those 4 characters will appear in the reverse order from that of the string literal.
* Data segment contents are displayed 512 bytes at a time (with scrolling) starting with the data segment base address (0x10010000). Navigation buttons are provided to change the display to the next section of memory, the previous, or back to the initial (home) range. A combo box is also provided to view memory contents in the vicinity of the stack pointer (contents of MIPS $sp register), global pointer (contents of MIPS $gp register), the heap base address (0x10040000), .extern globals (0x10000000), the kernel data segment (0x90000000), or memory-mapped IO (MMIO, 0xFFFF0000). *Starting with Mars 4.4,* raw text segment contents can also be displayed.
* Contents of any data segment memory word and almost any MIPS register can be modified by editing its displayed table cell. Double-click on a cell to edit it and press the Enter key when finished typing the new value. If you enter an invalid 32-bit integer, the word INVALID appears in the cell and memory/register contents are not affected. Values can be entered in either decimal or hexadecimal (leading "0x"). Negative hexadecimal values can be entered in either two's complement or signed format. Note that three of the integer registers (zero, program counter, return address) cannot be edited.
* *New in 4.4* If the setting for Self-Modifying Code is enabled (disabled by default, look in the Settings menu), text segment binary code can be modified using the same technique described above. It can also be modified by double-clicking on a cell in the Text Segment display's Code column.
* Contents of cells representing floating point registers can be edited as described above and will accept valid hexadecimal or decimal floating point values. Since each double-precision register overlays two single-precision registers, any changes to a double-precision register will affect one or both of the displayed contents of its corresponding single-precision registers. Changes to a single-precision register will affect the display of its corresponding double-precision register. Values entered in hexadecimal need to conform to IEEE-754 format. Values entered in decimal are entered using decimal points and E-notation (e.g. 12.5e3 is 12.5 times 10 cubed).
* Cell contents can be edited during program execution and once accepted will apply starting with the next instruction to be executed.
* Clicking on a Labels window entry will cause the location associated with that label to be centered and highlighted in the Text Segment or Data Segment window as appropriate. Note the Labels window is not displayed by default but can be by selecting it from the Settings menu.

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