**MARS - Mips Assembly and Runtime Simulator**

**Release 4.5**

**August 2014**

**MARS Release History**

Mars 4.5 was released in August 2014. Enhancements and bug fixes include:

* The Keyboard and Display MMIO Simulator tool has been enhanced at the suggestion of Eric Wang at Washington State University. Until now, all characters written to the display via the Data Transmitter location (low order byte of memory word 0xFFFF000C) were simply streamed to the tools' display window. Mr. Wang requested the ability to treat the display window as a virtual text-based terminal by being able to programmatically clear the window or set the (x,y) position of a text cursor. Controlled placement of the text cursor (which is not displayed) allows you to, among other things, develop 2D text-mode games.
  + To clear the window, place ASCII/Unicode 12 decimal in the Data Transmitter byte. This is the non-printing Form Feed character.
  + To set the text cursor to a specified (x,y) position, where x is the column and y is the row, place ASCII/Unicode 7 in the Data Transmitter byte, and place the (x,y) position in the unused upper 24 bits of the Data Transmitter word. Place the X-position in bits 20-31 and the Y-position in bits 8-19. Position (0,0) is the upper-left corner of the display.
  + You can resize the display window to desired dimensions prior to running your MIPS program. Dimensions are dynamically displayed in the upper border. Note that the tool now contains a splitter between the display window and the keyboard window. Once the program is running, changes to the display size does not affect cursor positioning.

The Help window for this tool is no longer modal, so you can view it while working in other windows. The Help window contains a lot of information so you will find it useful to be able to refer to it while working on your program.

* Installed the MIPS X-ray Tool developed by Marcio Roberto and colleagues at the Federal Center of Technological Education of Minas Gerais in Brazil. This tool animates a display of the MIPS datapath. The animation occurs while stepping through program execution. Search the Internet for "MIPS X-ray" to find relevant publications and other information.
* Context-sensitive help in the editor should now be easier to read. It was implemented as a menu of disabled items, which caused their text to be dimmed. The items are now enabled for greater visibility but clicking them will only make the list disappear.
* Bug Fix: Fixed an editor problem that affects certain European keyboards. The syntax-highlighting editor ignored the Alt key, which some European keyboards require to produce the # or $ characters in particular. I had no means of testing this, but Torsten Maehne in France send me a solution and Umberto Villano in Italy affirmed that it worked for him as well.
* Bug Fix: Source code references to Coprocessor 1 floating point registers (e.g. $f12) within macro definitions were erroneously flagged as syntax errors. MARS permits SPIM-style macro parameters (which start with $ instead of %) and did not correctly distinguish them from floating point register names. This has been fixed. Thanks to Rudolf Biczok in Germany for alerting me to the bug.
* Bug Fix: Corrected a bug that caused the Data Segment window to sometimes display incorrect values at the upper boundary of simulated memory segments. Thanks to Yi-Yu (James) Liu from Taiwan for alerting me to the bug, which was introduced in Mars 4.4.

Mars 4.4 was released in August 2013. Enhancements and bug fixes include:

* A feature to support self-modifying code has been developed by Carl Burch (Hendrix College) and Pete Sanderson. It is disabled by default and can be enabled through a Settings menu option. A program can write to the text segment and can also branch/jump to any user address in the data segments within the limits of the simulated address space. Text segment contents can also be edited interactively using the Data Segment window, and text segment contents within the address range of existing code can be edited interactively using the Text Segment window. In command mode, the smc option permits a program to write and execute in both text and data segments.
* Bug fix: An assembly error occurred when a line within a macro contained both a macro parameter and an identifier defined to have an .eqv substitution.
* Bug fix: If a macro name was used as a macro parameter, an assembly error occurred in some situations when a macro being used as an argument was defined following the macro that defined the parameter. The "for" macro described in the Macro help tab is an example.

Mars 4.3 was released in January 2013. Enhancements and bug fixes include:

* A macro facility has been developed by Mr. Mohammad Sekhavat. It is documented in the MIPS help tab Macros.
* A text substitution facility similar to #define has been developed using the new .eqv directive. It is also documented in the MIPS help tab Macros.
* A text insertion facility similar to #include has been developed using the new .include directive. It is also documented in the MIPS help tab Macros. It permits a macro to be defined in one file and included wherever needed.
* Two new command mode options are now available: ic (Instruction Count) to display a count of statements executed upon program termination, and me (Messages to Error) to send MARS messages to System.err instead of System.out. Allows you to separate MARS messages from MIPS output using redirection, if desired. Redirect a stream in DOS with "1>" or "2>" for out and err, respectively. To redirect both, use "> filename 2>&1"
* Changed the default font family settings from Courier New to Monospaced. This was in response to reports of Macs displaying left parentheses and vertical bars incorrectly.
* Changed the way operands for .byte and .half directives are range-checked. It will now work like SPIM, which accepts any operand value but truncates high-end bits as needed to fit the 1 (byte) or 2 (half) byte field. We'll still issue a warning but not an error.
* For file reads, syscall 14, file descriptor 0 is always open for standard input. For file writes, syscall 15, file descriptors 1 and 2 are always open for standard output and standard error, respectively. This permits you to write I/O code that will work either with files or standard I/O. When using the IDE, standard input and output are performed in the Run I/O tab. File descriptors for regular files are allocated starting with file descriptor 3.

Mars 4.2 was released in August 2011. Enhancements and bug fixes include:

* Performing a Save operation on a new file will now use the file's tab label as the the default filename in the Save As dialog (e.g. mips1.asm). Previously, it did not provide a default name.
* When the "assemble all files in directory" setting is enabled (useful for multi-file projects), you can now switch focus from one editor tab to another without invalidating the current assemble operation. You can similarly open additional project files. Previously, the open or tab selection would invalidate the assemble operation and any paused execution state or breakpoints would be lost.
* The Read String syscall (syscall 8) has been fortified to prevent Java exceptions from occurring when invalid values are placed in $a1.
* Will now perform runtime test for unaligned doubleword address in 'ldc1' and 'sdc1' instructions and trap if not aligned.
* Basic statements in the Text Segment display now renders immediates using eight hex digits when displaying in hex. Previously it rendered only four digits to conserve space. This led to confusing results. For instance, -1 and 0xFFFF would both be displayed as 0xFFFF but -1 expands to 0xFFFFFFFF and 0xFFFF expands to 0x0000FFFF.

Mars 4.1 was released in January 2011. Enhancements and bug fixes include:

* The ability to view Data Segment contents interpreted as ASCII characters has been added. You'll find it on the bottom border of the Data Segment Window as the checkbox "ASCII". This overrides the hexadecimal/decimal setting but only for the Data Segment display. It does not persist across sessions. Cells cannot be edited in ASCII format.
* The Dump Memory feature in the File menu now provides an ASCII dump format. Memory contents are interpreted as ASCII codes.
* A command-mode option "ascii" has been added to display memory or register contents interpreted as ASCII codes. It joins the existing "dec" and "hex" options for displaying in decimal or hexadecimal, respectively. Only one of the three may be specified.
* The actual characters to display for all the ASCII display options (data segment window, dump memory, command-mode option) are specified in the config.properties file. This includes a "placeholder" character to be displayed for all codes specified as non-printing. ASCII codes 1-7, 14-31, and 127-255 are specified as non-printing, but this can be changed in the properties file.
* A new Help tab called Exceptions has been added. It explains the basics of MIPS exceptions and interrupts as implemented in MARS. It also includes tips for writing and using exception handlers.
* A new Tool called Bitmap Display has been added. You can use it to simulate a simple bitmap display. Each word of the specified address space represents a 24 bit RGB color (red in bits 16-23, green in bits 8-15, blue in bits 0-7) and a word's value will be displayed on the Tool's display area when the word is written to by the MIPS program. The base address corresponds to the upper left corner of the display, and words are displayed in row-major order. Version 1.0 is pretty basic, constructed from the Memory Reference Visualization Tool code.
* Additional operand formats were added for the multiplication pseudo-instructions 'mul' and 'mulu'.
* The editor's context-sensitive pop-up help will now appear below the current line whenever possible. Originally it appeared either above, centered to the side, or below, depending on the current line's vertical position in the editing window. Displaying the pop-up above the current line tended to visually block important information, since frequently a line of code uses the same operand (especially registers) as the one immediately above it.
* The editor will now auto-indent each new line when the Enter key is pressed. Indentation of the new line will match that of the line that precedes it. This feature can be disabled in the Editor settings dialog.
* Two new command-mode options have been added. The "aeN" option, where N is an integer, will terminate MARS with exit value N when an assemble error occurs. The "seN" option will similarly terminate MARS with exit value N when a simulation (MIPS runtime) error occurs. These options can be useful when writing scripts for grading. Thanks to my Software Engineering students Robert Anderson, Jonathan Barnes, Sean Pomeroy, and Melissa Tress for designing and implementing these options.
* An editor bug that affected Macintosh users has been fixed. Command shortcuts, e.g. Command-s for save, did not function and also inserted the character into the text.
* A bug in Syscall 54 (InputDialogString) has been fixed. This syscall is the GUI equivalent of Syscall 8 (ReadString), which follows the semantics of UNIX 'fgets'. Syscall 54 has been modified to also follow the 'fgets' semantics.
* A bug in the Cache Simulator Tool has been fixed. The animator that paints visualized blocks green or red (to show cache hit or miss) sometimes paints the wrong block when set-associated caching is used. The underlying caching algorithm is correct so the numeric results such as hit ratios have always been correct. The animator has been corrected. Thanks to Andreas Schafer and his student Carsten Demel for bringing this to my attention.

Mars 4.0.1 was released in October 2010. It is a bug fix release to address three bugs.

* The Edit and Execute tabs of the IDE, which were relocated in 4.0 from the top to the left edge and oriented vertically, have been moved back to the top edge because Macintosh running Java 1.6 does not correctly render vertically-oriented tabs.
* An exception may be thrown in multi-file assembles when the last file of the assembly is not the longest. This occurs only when using the IDE, and has been corrected.
* If an assemble operation fails due to a non-existing exception handler file (specified through the IDE Settings menu), unchecking the exception handler setting does not prevent the same error from occuring on the next assemble. This has been corrected.

Mars 4.0 was released in August 2010. Enhancements and bug fixes include:

* *New Text Editor:* Mars features an entirely new integrated text editor. It creates a new tab for each file as it is opened. The editor now features language-aware color highlighting of many MIPS assembly language elements with customizable colors and styles. It also features automatic context-sensitive popup instruction guides. There are two levels: one with help and autocompletion of instruction names and a second with help information for operands. These and other new editor features can be customized or disabled through the expanded Editor Settings dialog. You can even revert to the previous notepad-style editor if you wish (multi-file capability is retained). The language-aware editor is based on the open source *jEdit Syntax Package* (syntax.jedit.org). It is separate from the assembler, so any syntax highlighting quirks will not affect assembly.
* *Improved Instruction Help:* All the instruction examples in the help tabs (and new popup instruction guides) now use realistic register names, e.g. $t1, $t2, instead of $1, $2. The instruction format key displayed above the MIPS help tabs has been expanded to include explanations of the various addressing modes for load and store instructions and pseudo-instructions. Descriptions have been added to every example instruction and pseudo-instruction.
* *Improved Assembly Error Capability:* If the assemble operation results in errors, the first error message in the Mars Messages text area will be highighted and the corresponding erroneous instruction will be selected in the text editor. In addition, you can click on any error message in the Mars Messages text area to select the corresponding erroneous instruction in the text editor. The first feature does not select in every situation (such as when assemble-on-open is set) but in the situations where it doesn't work no harm is done plus the second feature, clicking on error messages, can still be used.
* Console input syscalls (5, 6, 7, 8, 12) executed in the IDE now receive input keystrokes directly in the Run I/O text area instead of through a popup input dialog. Thanks to Ricardo Pascual for providing this feature! If you prefer the popup dialogs, there is a setting to restore them.
* The floor, ceil, trunc and round operations now all produce the MIPS default result 2^31-1 if the value is infinity, NaN or out of 32-bit range. For consistency, the sqrt operations now produce the result NaN if the operand is negative (instead of raising an exception). These cases are all consistent with FCSR (FPU Control and Status Register) Invalid Operation flag being off. The ideal solution would be to simulate the FCSR register itself so all MIPS specs for floating point instructions can be implemented, but that hasn't happened yet.
* The Basic column in the Text Segment Window now displays data and addresses in either decimal or hexadecimal, depending on the current settings. Note that the "address" in branch instructions is actually an offset relative to the PC, so is treated as data not address. Since data operands in basic instructions are no more than 16 bits long, their hexadecimal display includes only 4 digits.
* The Source column in the Text Segment Window now preserves tab spacing for a cleaner appearance (tab characters were previously not rendered).
* Instruction mnemonics can now be used as labels, e.g. "b:".
* New syscall 36 will display an integer as an unsigned decimal.
* A new tool, Digital Lab Sim, contributed by Didier Teifreto (dteifreto@lifc.univ-fcomte.fr). This tool features two seven-segment displays, a hexadecimal keypad, and a counter. It uses MMIO to explore interrupt-driven I/O in an engaging setting. More information is available from its Help feature. Many thanks!
* MARS 4.0 requires Java 1.5 (5.0) instead of 1.4. If this is an issue for you, let me know.

Mars 3.8 was released in January 2010. Enhancements and bug fixes include:

* A new feature to temporarily suspend breakpoints you have previously set. Use it when you feel confident enough to run your program without the breakpoints but not confident enough to clear them! Use the Toggle Breakpoints item in the Run menu, or simply click on the "Bkpt" column header in the Text Segment window. Repeat, to re-activate.
* Two new Tools contributed by Ingo Kofler of Klagenfurt University in Austria. One generates instruction statistics and the other simulates branch prediction using a Branch History Table.
* Two new print syscalls. Syscall 34 prints an integer in hexadecimal format. Syscall 35 prints an integer in binary format. Suggested by Bernardo Cunha of Portugal.
* A new Setting to control whether or not the MIPS program counter will be initialized to the statement with global label 'main' if such a statement exists. If the setting is unchecked or if checked and there is no 'main', the program counter will be initialized to the default starting address. Release 3.7 was programmed to automatically initialize it to the statement labeled 'main'. This led to problems with programs that use the standard SPIM exception handler exceptions.s because it includes a short statement sequence at the default starting address to do some initialization then branch to 'main'. Under 3.7 the initialization sequence was being bypassed. By default this setting is unchecked. This option can be specified in command mode using the 'sm' (Start at Main) option.
* Mars Tools that exist outside of Mars can now be included in the Tools menu by placing them in a JAR and including it in a command that launches the Mars IDE. For example: java -cp plugin.jar;Mars.jar Mars Thanks to Ingo Kofler for thinking of this technique and providing the patch to implement it.
* Corrections and general improvements to the MIDI syscalls. Thanks to Max Hailperin of Gustavus Adolphus College for supplying them.
* Correction to an assembler bug that flagged misidentified invalid MIPS instructions as directives.

Mars 3.7 was released in August 2009. Enhancements and bug fixes include:

* A new feature for changing the address space configuration of the simulated MIPS machine. The 32-bit address space configuration used by all previous releases remains the default. We have defined two alternative configurations for a compact 32KB address space. One starts the text segment at address 0 and the other starts the data segment at address 0. A 32KB address space permits commonly-used load/store pseudo-instructions using labels, such as lw $t0,increment, to expand to a single basic instruction since the label's full address will fit into the 16-bit address offset field without sign-extending to a negative value. This was done in response to several requests over the years for smaller addresses and simplified expansions to make assembly programs easier to comprehend. This release does not include the ability to define your own customized configuration, although we anticipate adding it in the future. It is available both through the command mode (option mc) and the IDE. See Memory Configuration... at the bottom of the Settings menu.
* Related to the previous item: load and store pseudo-instructions of the form lw $t0,label and lw $t0,label($t1) will expand to a single instruction (addi for these examples) if the current memory configuration assures the label's full address will fit into the low-order 15 bits. Instructions for which this was implemented are: la, lw, lh, lb, lhu, lbu, lwl, lwr, ll, lwc1, ldc1, l.s, l.d, sw, sh, sb, swl, swr, sc, swc1, sdc1, s.s, and s.d.
* If a file contains a global statement label "main" (without quotes, case-sensitive), then execution will begin at that statement regardless of its address. Previously, program execution always started at the base address of the text segment. This will be handy for multi-file projects because you no longer need to have the "main file" opened in the editor in order to run the project. Note that main has to be declared global using the .globl directive.
* We have added a Find/Replace feature to the editor. This has been another frequent request. Access it through the Edit menu or Ctrl-F. Look for major enhancements to the editor in 2010!
* The syscalls for Open File (13), Read from File (14), and Write to File (15) all now place their return value into register $v0 instead of $a0. The table in *Computer Organization and Design*'s Appendix B on SPIM specifies $a0 but SPIM itself consistently uses $v0 for the return values.
* Pseudo-instructions for div, divu, mulo, mulou, rem, remu, seq, sne, sge, sgeu, sgt, sgtu, sle, sleu now accept a 16- or 32-bit immediate as their third operand. Previously the third operand had to be a register.
* Existing Tools were tested using reconfigured memory address space (see first item). Made some adaptations to the Keyboard and Display Simulator Tool that allow it to be used for Memory Mapped I/O (MMIO) even under the compact memory model, where the MMIO base address is 0x00007f00 instead of 0xffff0000. Highlighting is not perfect in this scenario.
* Bug Fix: The syscall for Open File (13) reversed the meanings of the terms *mode* and *flag*. Flags are used to indicate the intended use of the file (read/write). Mode is used to set file permissions in specific situations. MARS implements selected flags as supported by Java file streams, and ignores the mode if specified. For more details, see the Syscalls tab under Help. The file example in that tab has been corrected.
* Bug Fix: The assembler incorrectly generated an error on Jump instructions located in the kernel text segment.
* Bug Fix: The project (p) option in the command interface worked incorrectly when MARS was invoked within the directory containing the files to be assembled.
* Acknowledgement: The development of Release 3.7 was supported by a SIGCSE Special Projects Grant.

Mars 3.6 was released in January 2009. Enhancements and bug fixes include:

* We've finally implemented the most requested new feature: memory and register cells will be highlighted when written to during timed or stepped simulation! The highlighted memory/register cell thus represents the result of the instruction just completed. During timed or stepped execution, this is NOT the highlighted instruction. During back-stepping, this IS the highlighted instruction. The highlighted instruction is the next one to be executed in the normal (forward) execution sequence.
* In conjunction with cell highlighting, we've added the ability to customize the highlighting color scheme and font. Select Highlighting in the Settings menu. In the resulting dialog, you can select highlight background color, text color, and font for the different runtime tables (Text segment, Data segment, Registers). You can also select them for normal, not just highlighted, display by even- and odd-numbered row but not by table.
* Cool new Labels Window feature: the table can be sorted in either ascending or descending order based on either the Label (alphanumeric) or the Address (numeric) column. Just click on the column heading to select and toggle between ascending (upright triangle) or descending (inverted triangle). Addresses are sorted based on unsigned 32 bit values. The setting persists across sessions.
* The Messages panel, which includes the Mars Messages and Run I/O tabs, now displays using a mono-spaced (fixed character width) font. This facilitates text-based graphics when running from the IDE.
* The Mars.jar distribution file now contains all files needed to produce a new jar file. This will make it easier for you to expand the jar, modify source files, recompile and produce a new jar for local use. CreatMarsJar.bat contains the jar instruction.
* The Help window now includes a tab for Acknowledgements. This recognizes MARS contributors and correspondents.
* We've added a new system call (syscall) for generating MIDI tones synchronously, syscall 33. The original MIDI call returns immediately when the tone is generated. The new one will not return until the tone output is complete regardless of its duration.
* The Data Segment display now scrolls 8 rows (half a table) rather than 16 when the arrow buttons are clicked. This makes it easier to view a sequence of related cells that happen to cross a table boundary. Note you can hold down either button for rapid scrolling. The combo box with various data address boundaries also works better now.
* Bug Fix: Two corrections to the Keyboard and Display Simulator Tool. Transmitter Ready bit was not being reset based on instruction count when running in the kernel text segment, and the Status register's Exception Level bit was not tested before enabling the interrupt service routine (could lead to looping if interrupts occur w/i the interrupt service routine). Thanks to Michael Clancy and Carl Hauser for bringing these to my attention and suggesting solutions.
* Bug Fix: Stack segment byte addresses not on word boundaries were not being processed correctly. This applies to little-endian byte order (big-endian is not enabled or tested in MARS). Thanks to Saul Spatz for recognizing the problem and providing a patch.
* Minor Bug Fixes include: Correcting a fault leading to failure when launching MARS in command mode, clarifying assembler error message for too-few or too-many operands error, and correcting the description of lhu and lbu instructions from "unaligned" to "unsigned".

Mars 3.5 was released in August 2008. Major enhancements and bug fixes include:

* A new Tool, the Keyboard and Display MMIO Simulator, that supports polled and interrupt-driven input and output operations through Memory-Mapped I/O (MMIO) memory. The MIPS program writes to memory locations which serve as registers for simulated devices. Supports keyboard input and a simulated character-oriented display. Click the tool's Help button for more details.
* A new Tool, the Instruction Counter, contributed by MARS user Felipe Lessa. It will count the number of MIPS instructions executed along with percentages for R-format, I-format, and J-format instructions. Thanks, Felipe!
* Program arguments can now be provided to the MIPS program at runtime, through either an IDE setting or command mode. See the command mode "pa" option for more details on command mode operation. The argument count (argc) is placed in $a0 and the address of an array of null-terminated strings containing the arguments (argv) is placed in $a1. They are also available on the runtime stack ($sp).
* Two related changes permit MARS to assemble source code produced by certain compilers such as gcc. One change is to issue warnings rather than errors for unrecognized directives. MARS implements a limited number of directives. Ignore these warnings at your risk, but the assembly can continue. The second change is to allow statement labels to contain, and specifically begin with, '$'.
* In command mode, final register values are displayed by giving the register name as an option. Register names begin with '$', which is intercepted by certain OS command shells. The convention for escaping it is not uniform across shells. We have enhanced the options so now you can give the register name without the '$'. For instance, you can use t0 instead of $t0 as the option. You cannot refer to registers by number in this manner, since an integer option is interpreted by the command parser as an instruction execution limit. Thanks to Lucien Chaubert for reporting this problem.
* Minor enhancements: The command mode dump feature has been extended to permit memory address ranges as well as segment names. If you enter a new file extension into the Open dialog, the extension will remain available throughout the interactive session. The data segment value repetition operator ':' now works for all numeric directives (.word, .half, .byte, .float, .double). This allows you to initialize multiple consecutive memory locations to the same value. For example:   
  ones: .half 1 : 8 # Store the value 1 in 8 consecutive halfwords
* Major change: Hexadecimal constants containing less than 8 digits will be interpreted as though the leading digits are 0's. For instance, 0xFFFF will be interpreted as 0x0000FFFF, not 0xFFFFFFFF as before. This was causing problems with immediate operands in the range 32768 through 65535, which were misinterpreted by the logical operations as signed 32 bit values rather than unsigned 16 bit values. Signed and unsigned 16 bit values are now distinguished by the tokenizer based on the prototype symbols -100 for signed and 100 for unsigned (mainly logical operations). Many thanks to Eric Shade of Missouri State University and Greg Gibeling of UC Berkeley for their extended efforts in helping me address this situation.
* Minor Bug Fixes: round.w.s and round.w.d have been modified to correctly perform IEEE rounding by default. Thanks to Eric Shade for pointing this out. Syscall 12 (read character) has been changed to leave the character in $v0 rather then $a0. The original was based on a misprint in Appendix A of *Computer Organization and Design*. MARS would not execute from the executable Mars.jar file if it was stored in a directory path those directory names contain any non-ASCII characters. This has been corrected. Thanks to Felipe Lessa for pointing this out and offering a solution. MARS will now correctly detect the EOF condition when reading from a file using syscall 14. Thanks to David Reimann for bringing this to our attention.

Mars 3.4.1 was released on 23 January 2008. It is a bug fix release to address two bugs.

* One bug shows up in pseudo-instructions in which the expansion includes branch instructions. The fixed branch offsets were no longer correct due to changes in the calculation of branch offsets in Release 3.4. At the same time, we addressed the issue of expanding such pseudo-instructions when delayed branching is enabled. Such expansions will now include a nop instruction following the branch.
* We also addressed an off-by-one error that occurred in generating the lui instruction in the expansion of conditional branch pseudo-instructions whose second operand is a 32 bit immediate.
* The expansions for a number of pseudo-instructions were modified to eliminate internal branches. These and other expansions were also optimized for sign-extended loading of 16-bit immediate operands by replacing the lui/ori or lui/sra sequence with addi. Pseudo-instructions affected by one or both of these modifications include: abs, bleu, bgtu, beq, bne, seq, sge, sgeu, sle, sleu, sne, li, sub and subi. These modifications were suggested by Eric Shade of Missouri State University.

Mars 3.4 was released in January 2008. Major enhancements are:

* A new syscall (32) to support pauses of specified length in milliseconds (sleep) during simulated execution.
* Five new syscalls (40-44) to support the use of pseudo-random number generators. An unlimited number of these generators are available, each identified by an integer value, and for each you have the ability to: set the seed value, generate a 32 bit integer value from the Java int range, generate a 32 bit integer value between 0 (inclusive) and a specified upper bound (exclusive), generate a 32-bit float value between 0 (inclusive) and 1 (exclusive), and generate a 64-bit double value between 0 (inclusive) and 1 (exclusive). All are based on the java.util.Random class.
* Ten new syscalls (50-59) to support message dialog windows and data input dialog windows. The latter are distinguished from the standard data input syscalls in that a prompting message is specified as a syscall argument and displayed in the input dialog. All are based on the javax.swing.JOptionPane class.
* The capability to dump .text or .data memory contents to file in various formats. The dump can be performed before or after program execution from either the IDE (File menu and toolbar) or from command mode. It can also be performed during an execution pause from the IDE. Look for the "Dump Memory" menu item in the File menu, or the "dump" option in command mode. A .text dump will include only locations containing an instruction. A .data dump will include a multiple of 4KB "pages" starting at the segment base address and ending with the last 4KB "page" to be referenced by the program. Current dump formats include pure binary (java.io.PrintStream.write() method), hexadecimal text with one word (32 bits) per line, and binary text with one word per line. An interface, abstract class, and format loader have been developed to facilitate development and deployment of additional dump formats. This capability was prototyped by Greg Gibeling of UC Berkeley.
* Changed the calculation of branch offsets when Delayed Branching setting is disabled. Branch instruction target addresses are represented by the relative number of words to branch. With Release 3.4, this value reflects delayed branching, regardless of whether the Delayed Branching setting is enabled or not. The generated binary code for branches will now match that of examples in the *Computer Organization and Design* textbook. This is a change from the past, and was made after extensive discussions with several MARS adopters. Previously, the branch offset was 1 lower if the Delayed Branching setting was enabled -- the instruction label: beq $0,$0,label would generate 0x1000FFFF if Delayed Branching was enabled and 0x10000000 if it was disabled. Now it will generate 0x1000FFFF in either case. The simulator will always branch to the correct location; MARS does not allow assembly under one setting and simulation under the other.
* Bug fix: The mars.jar executable JAR file can now be run from a different working directory. Fix was suggested by Zachary Kurmas of Grand Valley State University.
* Bug fix: The problem of MARS hanging while assembling a pseudo-instruction with a label operand that contains the substring "lab", has been fixed.
* Bug fix: No Swing-related code will be executed when MARS is run in command mode. This fixes a problem that occurred when MARS was run on a "headless" system (no monitor). Swing is the Java library to support programming Graphical User Interfaces. Fix was provided by Greg Gibeling of UC Berkeley.
* The '\0' character is now recognized when it appears in string literals.

MARS 3.3 was released in July 2007. Major enhancements are:

* Support for MIPS delayed branching. All MIPS computers implement this but it can be confusing for programmers, so it is disabled by default. Under delayed branching, the next instruction after a branch or jump instruction will *always* be executed, even if the branch or jump is taken! Many programmers and assemblers deal with this by inserting a do-nothing "nop" instruction after every branch or jump. The MARS assembler does *not* insert a "nop". Certain pseudo-instructions expand to a sequence that includes a branch; such instructions will not work correctly under delayed branching. Delayed branching is available in command mode with the "db" option.
* A new tool of interest mainly to instructors. The Screen Magnifier tool, when selected from the Tools menu, can be used to produce an enlarged static image of the pixels that lie beneath it. The image can be annotated by dragging the mouse over it to produce a scribble line. It enlarges up to 4 times original size.
* You now have the ability to set and modify the text editor font family, style and size. Select "Editor..." from the Settings menu to get the dialog. Click the Apply button to see the new settings while the dialog is still open. Font settings are retained from one session to the next. The font family list begins with 6 fonts commonly used across platforms (selected from lists found at [www.codestyle.org](http://www.codestyle.org/)), followed by a complete list. Two of the six are monospaced fonts, two are proportional serif, and two are proportional sans serif.
* The Labels window on the Execute pane, which displays symbol table information, has been enhanced. When you click on a label name or its address, the contents of that address are centered and highlighted in the Text Segment window or Data Segment window as appropriate. This makes it easier to set breakpoints based on text labels, or to find the value stored at a label's address.
* If you re-order the columns in the Text Segment window by dragging a column header, the new ordering will be remembered and applied from that time forward, even from one MARS session to the next. The Text Segment window is where source code, basic code, binary code, code addresses, and breakpoints are displayed.
* If a MIPS program terminates by "running off the bottom" of the program, MARS terminates, as before, without an exception, but now will display a more descriptive termination message in the Messages window. Previously, the termination message was the same as that generated after executing an Exit syscall.
* A new system call (syscall) to obtain the system time is now available. It is service 30 and is not available in SPIM. Its value is obtained from the java.util.Date.getTime() method. See the Syscall tab in MIPS help for further information.
* A new system call (syscall) to produce simulated MIDI sound through your sound card is now available. It is service 31 and is not available in SPIM. Its implementation is based on the javax.sound.midi package. It has been tested only under Windows. See the Syscall tab in MIPS help for further information.

MARS 3.2.1 was released in January 2007. It is a bug fix release that addresses the following bug in 3.2: a NullPointerException occurs when MIPS program execution terminates by "dropping off the bottom" of the program rather than by using one of the Exit system calls.

MARS 3.2 was released in December 2006. Major enhancements are:

* It fixes several minor bugs, including one that could cause incorrect file sequencing in the Project feature.
* It includes the AbstractMarsToolAndApplication abstract class to serve as a framework for easily constructing "tools" and equivalent free-standing applications that use the MARS assembler and simulator engines (kudos to the SIGCSE 2006 audience member who suggested this capability!). A subclass of this abstract class can be used both ways (tool or application).
* The floating point and data cache tools were elaborated in this release and a new tool for animating and visualizing memory references was developed. All are AbstractMarsToolAndApplication subclasses.
* This release includes support for exception handlers: the kernel data and text segments (.kdata and .ktext directives), the MIPS trap-related instructions, and the ability to automatically include a selected exception (trap) handler with each assemble operation.
* Items in the Settings menu became persistent with this release.
* Added default assembly file extensions "asm" and "s" to the Config.properties file and used those not only to filter files for the File Open dialog but also to filter them for the "assemble all" setting.
* Implemented a limit to the amount of text scrollable in the Mars Messages and Run I/O message tabs - default 1 MB is set in the Config.properties file.
* For programmer convenience, labels can now be referenced in the operand field of integer data directives (.word, .half, .byte). The assembler will substitute the label's address (low order half for .half, low order byte for .byte).
* For programmer convenience, character literals (e.g. 'b', '\n', '\377') can be used anywhere that integer literals are permitted. The assembler converts them directly to their equivalent 8 bit integer value. Unicode is not supported and octal values must be exactly 3 digits ranging from '\000' to '\377'.
* Replaced buttons for selecting Data Segment display base addresses with a combo box and added more base addresses: MMIO (0xFFFF0000), .kdata (0x90000000), .extern (0x10000000).

MARS 3.1 was released in October 2006. The major issues and features are listed here:

* It addressed several minor limits (Tools menu items could not be run from the JAR file, non-standard shortcuts for Mac users, inflexible and sometimes inappropriate sizing of GUI components).
* It changed the way SYSCALLs are implemented, to allow anyone to define new customized syscall services without modifying MARS.
* It added a primitive Project capability through the "Assemble operation applies to all files in current directory." setting (also available as "p" option in command mode). The command mode also allows you to list several file names not necessarily in the same directory to be assembled and linked.
* Multi-file assembly also required implementing the ".globl" and ".extern" directives.
* And although "Mars tools" are not officially part of MARS releases, MARS 3.1 includes the initial versions of two tools: one for learning about floating point representation and another for simulating data caching capability.

MARS 3.0 was released in August 2006, with one bug fix and two major additions.

* The bug fix was corrected instruction format for the slti and sltiu instructions.
* One major addition is a greatly expanded MIPS-32 instruction set (trap instructions are the only significant ones to remain unimplemented). This includes, via pseudo-instructions, all reasonable memory addressing modes for the load/store instructions.
* The second major addition is 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).
* A number of IDE settings, described above, are now available through the Settings menu.

MARS 2.2 was released in March 2006 with additional bug fixes and implemented command line options (run MARS from command line with h option for command line help). This also coincides with our SIGCSE 2006 paper "MARS: An Education-Oriented MIPS Assembly Language Simulator".

MARS 2.1 was released in October 2005 with some bug fixes.

MARS 2.0 was released in September 2005. It incorporated significant modifications to both the GUI and the assembler, floating point registers and instructions most notably.

MARS 1.0 was released in January 2005 and publicized during a poster presentation at SIGCSE 2005.

Dr. Ken Vollmar initiated MARS development in 2002 at Missouri State University. In 2003, Dr. Pete Sanderson of Otterbein College and his student Jason Bumgarner continued implementation. Sanderson implemented the assembler and simulator that summer, and the basic GUI the following summer, 2004.

The development of Releases 3.1 and 3.2 in 2006 and 4.0 in 2010 were supported by the Otterbein College sabbatical leave program. The development of Release 3.7 during summer 2009 was supported by a SIGCSE Special Projects Grant.

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