MARS Tutorial: Basic MARS Use

Created by Pete Sanderson & Ken Vollmar
Modified by Hyeran Jeon & Haonan Wang

The example program is **`Fibonacci.asm`** to compute everyone's favorite number sequence.

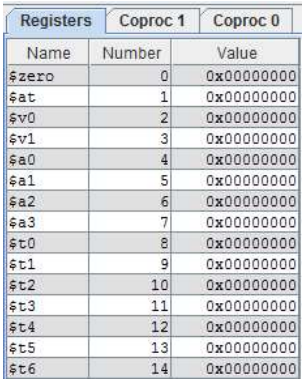
1. Start the MARS simulator by double clicking on the .jar file. **MARS requires Java J2SE 1.4.2 (or later) SDK installed on your computer (http://java.sun.com/javase/downloads/index.jsp)**

2. Use the menubar File→Open or the Open icon  to open Fibonacci.asm in the default folder. *(All icons have menubar equivalents; the remainder of these steps will use the icon whenever possible.)*

3. The provided assembly program is complete. Assemble the program using the icon 

4. Identify the location and values of the program's initialized data. Use the checkbox to toggle the display format between decimal and hexadecimal  .
   - The nineteen-element array **`fibs`** is initialized to zero, at addresses 0x10010000 to 0x10010048.
   - The data location **`size`** has value 19_{ten} at 0x1001004c.
   - The addresses 0x10010050 to 0x1001006c contain null-terminated ASCII strings.

5. Locate the Registers display, which shows the 32 common MIPS registers. Other tabs in the Registers display show the floating-point registers (Coproc 1) and status codes (Coproc 0).



Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000

6. Use the slider bar to change the run speed to about 10 instructions per second.

   This allows us to "watch the action" instead of the assembly program finishing directly.

7. Choose how you will execute the program:

   - The  icon runs the program to completion. Using this icon, you should observe the yellow highlight showing the program's progress and the values of the Fibonacci sequence appearing in the Data Segment display.

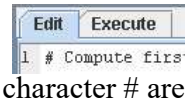
- The  icon resets the program and simulator to initial values. Memory contents are those specified within the program, and register contents are generally zero.
- The  icon is "single-step." Its complement is  , "single-step backwards" (undoes each operation).

8. Observe the output of the program in the Run I/O display window:
   The Fibonacci numbers are:
   1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
   -- program is finished running --

9. Modify the contents of memory. (Modifying a register value is exactly the same.)
   - Set a breakpoint at the first instruction of the subroutine which prints results. Use the checkbox at the left of the instruction whose address is 0x00400058 = 4194392_{ten}.
     

   - Reset  and re-run  the program, which stops at the breakpoint.
   - Double-click in one of the memory locations containing the computed Fibonacci numbers. The cell will be highlighted and will accept keyboard entry, similar to a spreadsheet. Enter some noticeably different value, and use the Enter key or click outside the cell to indicate that the change is complete. *Example: Memory address 0x10010020 = 268501024 $_{ten}$ presently contains data 0x00000022 = 34 $_{ten}$.*
   - Click  to continue from the breakpoint. The program output includes your entered value instead of the computed Fibonacci number.

10. Open the Help  for information on MIPS instructions, pseudoinstructions, directives, and syscalls.

11. Modify the program so that it prompts the user for the Fibonacci sequence length.
    - Select the Edit tab in the upper right to return to the program editor. 

    - The MIPS comment symbol is #. All characters on the line after the character # are ignored.
    - Un-comment lines 12-19. The newly exposed program fragment will prompt the user for the length of the Fibonacci sequence to generate, in the range $2 \le x \le 19$. (The length of the sequence must be limited to the size of the declared space for result storage.)
    - Determine the correct **`syscall`** parameter to perform "read integer" from the user, and insert the parameter at line The correct **`syscall`** parameter may be found at Help  → Syscall tab → read integer service. The completed line will have the form **`li $v0, 42`** (where in this case 42 is not the right answer).
    - Reset  and re-run  the program. The program will stop at the breakpoint you inserted previously. Continue and finish with  .