

Mars MIPS Editor Link →

<https://courses.missouristate.edu/KenVollmar/mars/download.htm>

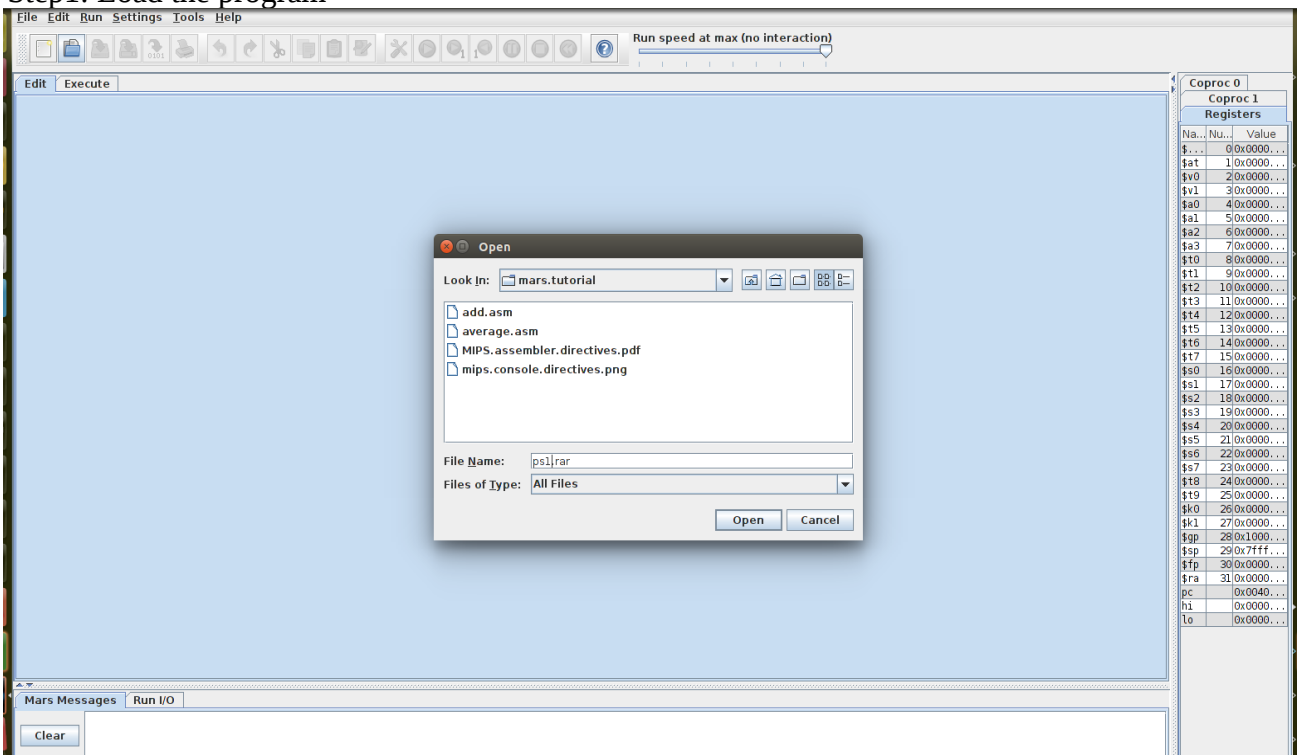
For running Mars Editor

- **Option A: Desktop.** Save the jar file on the desktop. Run MARS by double-clicking the icon.
- **Option B: DOS shell using jar file.** Save the jar file in some folder. Open a DOS shell in that folder. Rename the jar file to "Mars.jar" for convenience. Run MARS with the DOS command `java -jar Mars.jar`
- **Option C: DOS shell using Java classes.** Save the jar file in some folder. Open a DOS shell in that folder. Rename the jar file to "Mars.jar" for convenience. Extract MARS files with the DOS command `jar -xf Mars.jar` Run MARS with the DOS command `java Mars`

Mars Config → Default Mode is enough.

Mars Usage →

Step1: Load the program



Step2: Run the program

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

add.asm

```
3 # of two numbers specified at runtime by the user.
4 # Registers used:
5 # $t0 - used to hold the first number.
6 # $t1 - used to hold the second number.
7 # $t2 - used to hold the sum of the $t1 and $t2.
8 # $v0 - syscall parameter and return value.
9 # $a0 - syscall parameter.
10
11 main:
12     ## Get first number from user, put into $t0.
13     li $v0, 5 # load syscall read_int into $v0.
14     syscall # make the syscall.
15     move $t0, $v0 # move the number read into $t0.
16
17     ## Get second number from user, put into $t1.
18     li $v0, 5 # load syscall read_int into $v0.
19     syscall # make the syscall.
20     move $t1, $v0 # move the number read into $t1.
21
22     add $t2, $t0, $t1 # compute the sum.
23
24     ## Print out $t2.
25     move $a0, $t2 # move the number to print into $a0.
26     li $v0, 1 # load syscall print_int into $v0.
27     syscall # make the syscall.
28
29     li $v0, 10 # syscall code 10 is for exit.
30     syscall # make the syscall.
31
32 # end of add2.asm.
33
```

Line: 21 Column: 1 ☒ Show Line Numbers

Mars Messages Run I/O

Clear

Registers	Coproc 1	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
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute Run the current program

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24020005	addiu \$2,\$0,0x00000005	13: li \$v0, 5 # load syscall read_int into \$v0.
	0x00400004	0x0000000c	syscall	14: syscall # make the syscall.
	0x00400008	0x00024021	addiu \$8,\$0,\$2	15: move \$t0, \$v0 # move the number read into \$t0.
	0x0040000c	0x24020005	addiu \$2,\$0,0x00000005	18: li \$v0, 5 # load syscall read_int into \$v0.
	0x00400010	0x0000000c	syscall	19: syscall # make the syscall.
	0x00400014	0x00024821	addiu \$9,\$0,\$2	20: move \$t1, \$v0 # move the number read into \$t1.
	0x00400018	0x01095020	add \$10,\$8,\$9	22: add \$t2, \$t0, \$t1 # compute the sum.
	0x0040001c	0x000a2021	addiu \$4,\$0,\$10	25: move \$a0, \$t2 # move the number to print into \$a0.
	0x00400020	0x24020001	addiu \$2,\$0,0x00000001	26: li \$v0, 1 # load syscall print_int into \$v0.
	0x00400024	0x0000000c	syscall	27: syscall # make the syscall.
	0x00400028	0x2402000a	addiu \$2,\$0,0x0000000a	29: li \$v0, 10 # syscall code 10 is for exit.
	0x0040002c	0x0000000c	syscall	30: syscall # make the syscall.

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☒ Hexadecimal Addresses ☒ Hexadecimal Values ☐ ASCII

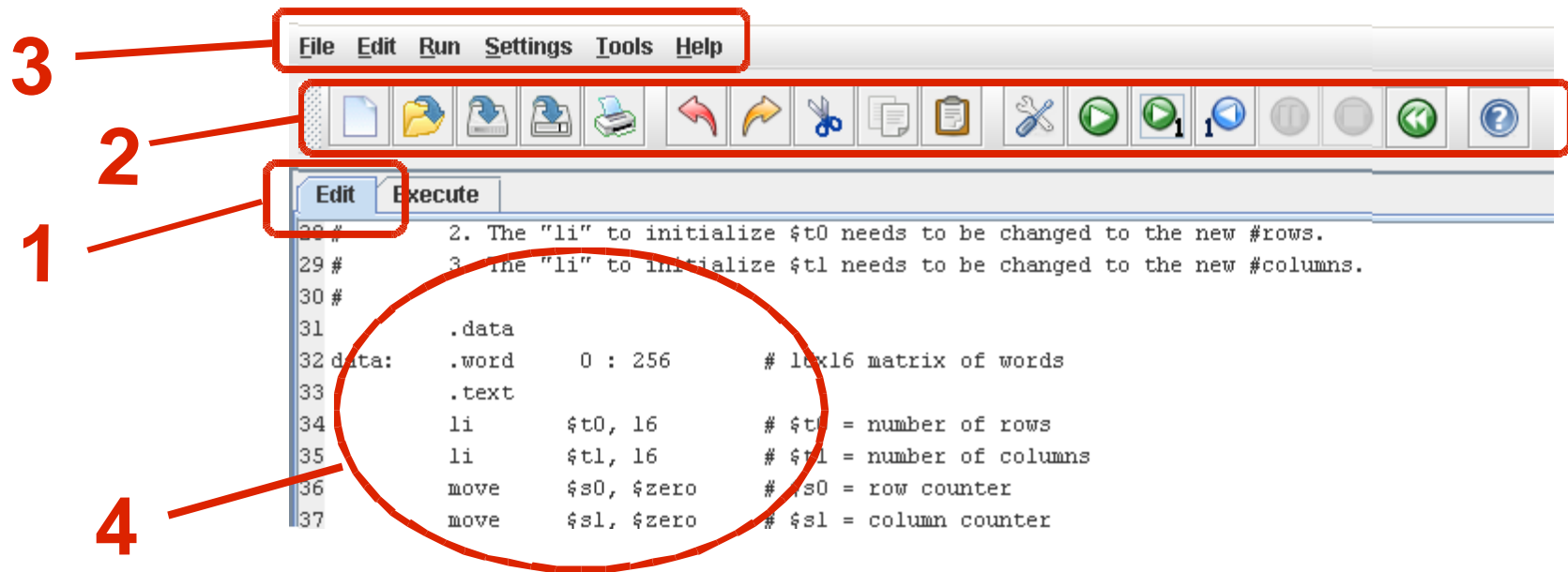
Mars Messages Run I/O

Assemble: assembling /home/fesirci/Documents/ps2/mars.tutorial/add.asm

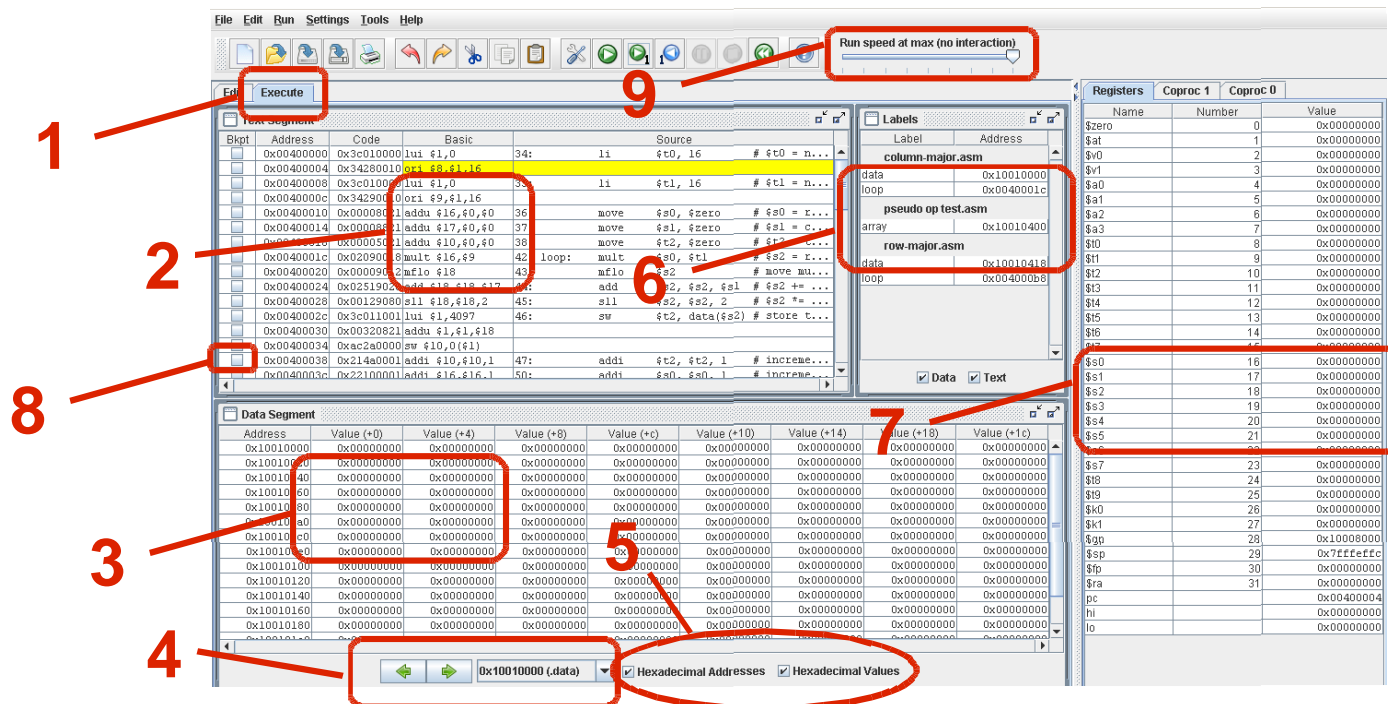
Assemble: operation completed successfully.

Clear

Registers	Coproc 1	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
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000



1. Edit display is indicated by highlighted tab.
- 2, 3. Typical edit and execute operations are available through icons and menus, dimmed-out when unavailable or not applicable.
4. WYSIWYG editor for MIPS assembly language code.



1. Execute display is indicated by highlighted tab.
2. Assembly code is displayed with its address, machine code, assembly code, and the corresponding line from the source code file. (Source code and assembly code will differ when pseudoinstructions have been used.)
3. The values stored in Memory are directly editable (similar to a spreadsheet).
4. The window onto the Memory display is controlled in several ways: previous/next arrows and a menu of common locations (e.g., top of stack).
5. The numeric base used for the display of data values and addresses (memory and registers) is selectable between decimal and hexadecimal.
6. Addresses of labels and data declarations are available. Typically, these are used only when single-stepping to verify that an address is as expected.
7. The values stored in Registers are directly editable (similar to a spreadsheet).
8. Breakpoints are set by a checkbox for each assembly instruction. These checkboxes are always displayed and available.
9. Selectable speed of execution allows the user to "watch the action" instead of the assembly program finishing directly.