

Introduction to MARS

In order to edit and execute MIPS assembly programs we will use MARS (MIPS Assembler and Runtime Simulator). This is a short guide on how to launch and use MARS.

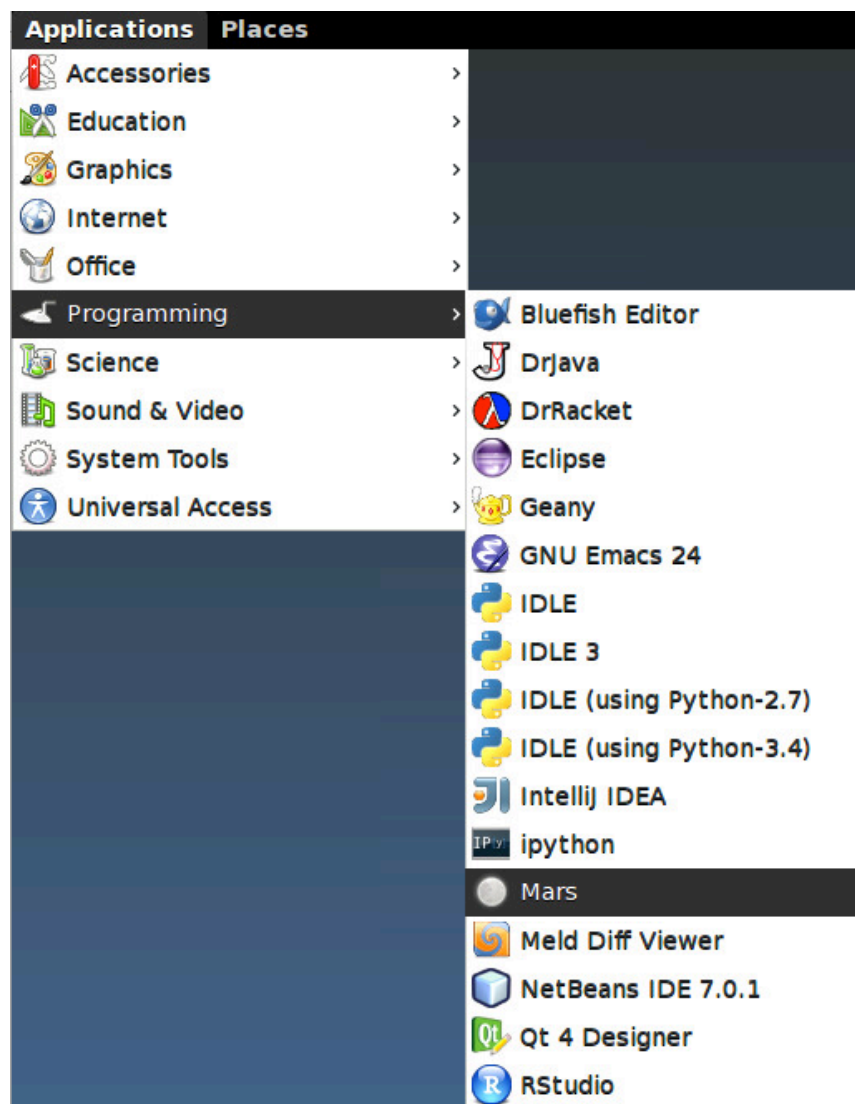
Launch MARS

MARS is available on the department Linux system. You may prefer to download, install and run MARS on your private computer (file area on the student portal).

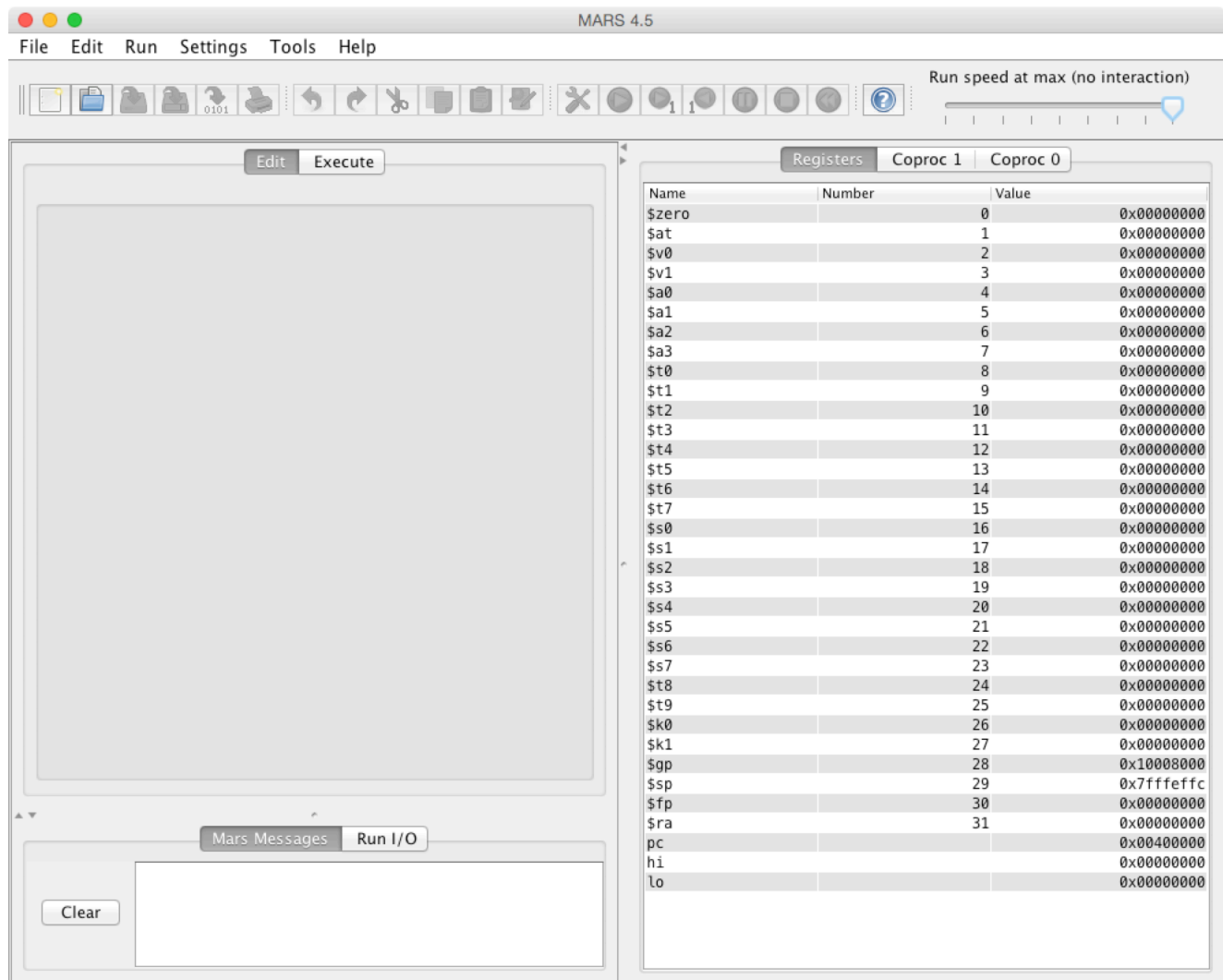
Department Linux system

Log in to the department Linux system.

From the Applications menu you find MARS under Programming.



MARS should now start and you should see something similar to this.



IDE overview







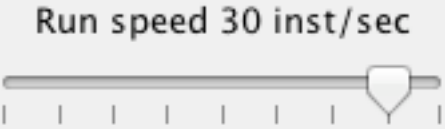
MARS is an Integrated Development Environment (IDE) for MIPS Assembly Language Programming

Top level menu

At the top you find the top level menu: **File**, **Edit**, **Run**, **Settings**, **Tools** and **Help**.

Tools and operations

Under the top level menu a collection of icons show some of the most commonly used tools and operations. The most important of these controls are described in the below table.

Control	Description
	Load a file
	Save the current file
	Assemble the program in the Edit tab
	Run the assembled program to completion (or breakpoint)
	Execute a single instruction (single-step)
	Undo the last instruction (single-step backwards)
 <p>Run speed 30 inst/sec</p>	Adjust the execution speed

Edit and Execute

In the middle left you find two tabs: **Edit** and **Execute**.

- The **Edit tab** will be used to **edit assembly code**.
- The **Execute tab** showd the **Text Segment** (machine instructions) and **Data Segment** during **execution**.

Registers

To the left you find the registers pane. Here the contents of all registers are shown. There are three register tabs:

- 1 General purpose registers.
- 2 Coprocessor 0 register.
- 3 Coprocessor 1 registers.

Mars Messages and Run I/O

In the lower left corner there are two tabs: **Mars Messages** and **Run I/O**.

The **Mars Messages** tab is used for displaying assembly or runtime errors and informational messages. You can click on assembly error messages to highlight and set focus on the corresponding line of code in the editor.

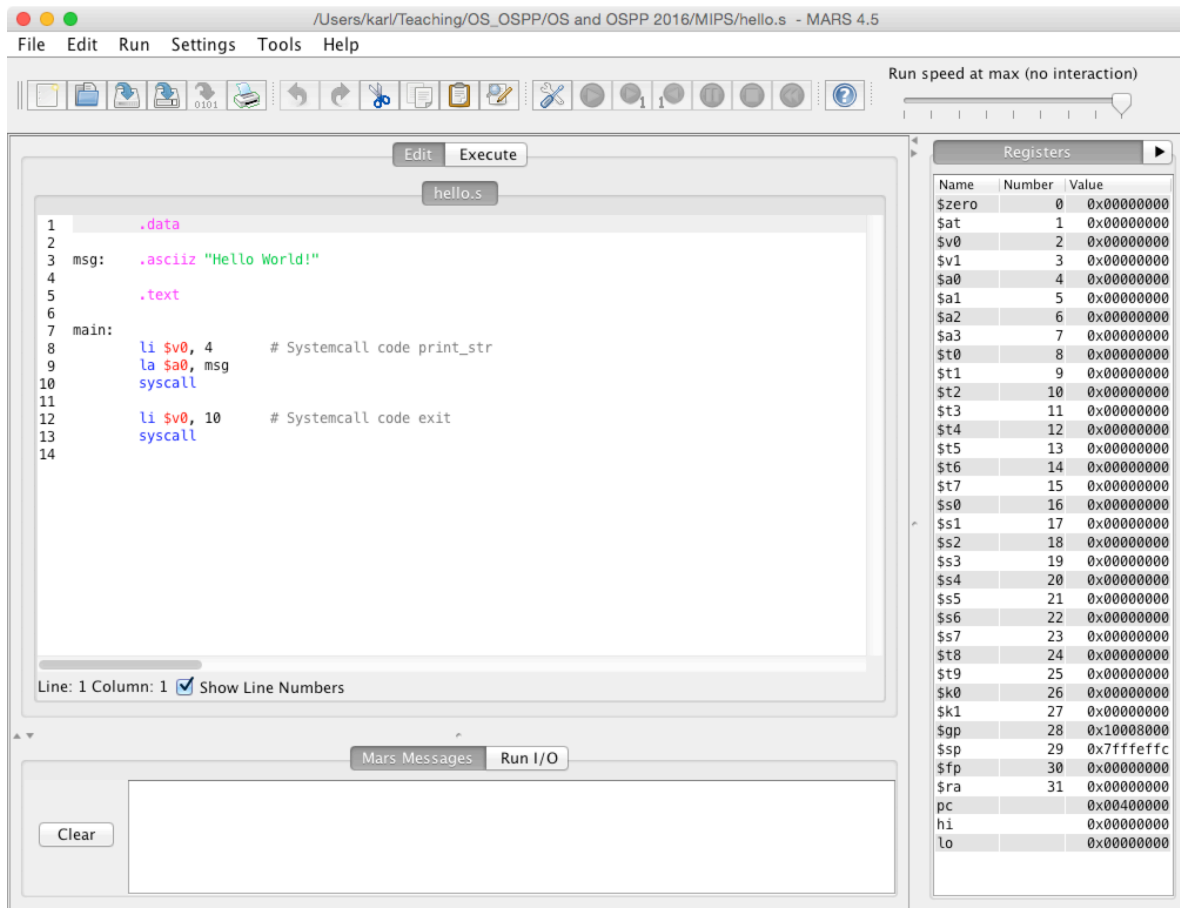
The **Run I/O** tab is used at runtime for displaying console output and entering console input as program execution progresses.

Download and load program into MARS

Download the file hello.s (file area on the student portal)

Open the file hello.s in the MARS simulator by selecting **Open** from the **File** menu.

You should now see something similar to this.



After you loaded a program, the source code is view in the **Edit pane**.

Change font colors

The default font colors might not be the most pleasing on your eyes. Especially the font color for comments may be hard to read.

Follow these steps to change the font colors:

0 **Top menu**

1 **Settings**

2 **Editor ...**

3 Now a window named **Text editor settings** will open.

4 To the right you find the **Syntax styling** options.

5 Un-check the checkbox to **override the default font**.

6 **Click** on the **button** to the right of the font preview to change the color of the font.

7 Click on the button **Apply and close** in the lower left corner of the window to apply your settings.

- 5 Uncheck this checkbox
- 6 Click here to change color


Syntax Styling

☒ = use defaults (disables buttons)

Comment	# Load	6	B	I	<input type="checkbox"/> 5
String literal	"First"		B	I	<input checked="" type="checkbox"/>
Character literal	'\n'		B	I	<input checked="" type="checkbox"/>
Label	main:		B	I	<input type="checkbox"/>
MIPS instruction	lui		B	I	<input checked="" type="checkbox"/>
Assembler directive	.text		B	I	<input checked="" type="checkbox"/>
Register	\$zero		B	I	<input checked="" type="checkbox"/>
In-progress, invalid	"Regi		B	I	<input checked="" type="checkbox"/>
Macro parameter	%arg		B	I	<input checked="" type="checkbox"/>

Assemble

To translate the MIPS assembly source code to executable machine instructions an assembler is used.

Assemble the file by clicking on the icon  (screwdriver and wrench).

Mars Messages

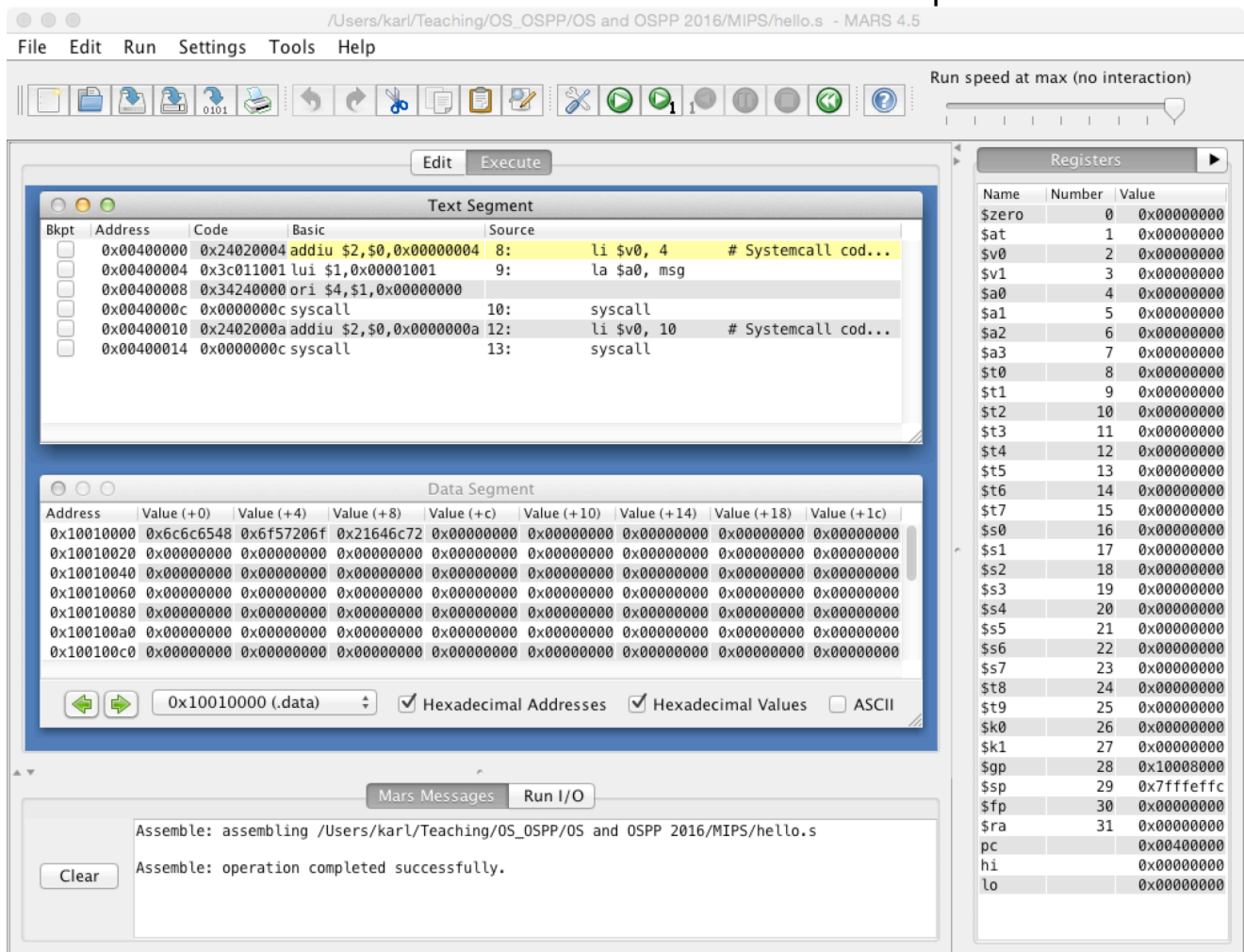
You should see something similar to the following in the Mars Messages display pane.

```
Assemble: assembling /path/to/file/hello.s
```

```
Assemble: operation completed successfully.
```

Execute pane

After a successful assembly, the generated machine code instructions together with the source code instructions are shown in the Execute pane.



Run to completion

Click on the  to run the program to completion.

Run I/O

In the the Run I/O display window you should see the following output.
Hello World!

-- program is finished running --

Symbol table

From the **Settings** menu, select **Show Label Window (symbol table)**. Now, the following window should appear.

Label	Address ▲
hello.s	
main	0x00400000
msg	0x10010000

☒ Data ☒ Text

The symbol table shows the actual memory addresses of all labels. For example, we see that the label `main` is a mnemonic for the memory address `0x00400000`.

Text segment

In the symbol table, click on the label `main`.

Now the following row should be **highlighted with a blue border** in the Source code column in the Text segment area in the Execute tab.

```
li $v0, 4 # Systemcall code print_str
```

If you look at the source code (press the Edit tab) you see that this is the instruction following directly after the label `main`.

Data segment

In the symbol table, click on the label `msg`.

Now address `0x10010000` should be **highlighted with a blue border** in the Data Segment area in the Execute tab. The value store at this address is `0x6c6c6548`.

If you study the data segment in detail you see that the string "Hello World!" is store byte for byte starting at address `0x10010000`.

Address	0x10010000				0x10010004				0x10010008			
Offset	3	2	1	0	3	2	1	0	3	2	1	0
Hex value	6c	6c	65	48	6f	57	20	6f	21	64	6c	72
ASCII Glyph	1	1	e	H	o	W	(space)	o	!	d	l	r

Breakpoints and debugging

A very usefull feature of MARS is the ability to set breakpoints. **Breakpoints** together with **single-stepping** and **backward single-stepping** are very powerfull when debugging a program.

Assemble the file.

Make sure to view the Execute tab. In the Text segment, click the **Bkpt** (breakpoint) checkbox at address 0x00400000.

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x24020004	addiu \$2,\$0,0x00000004	8: li \$v0, 4 # Systemcall code print_str
<input type="checkbox"/>	0x00400004	0x3c011001	lui \$1,0x00001001	9: la \$a0, msg
<input type="checkbox"/>	0x00400008	0x34240000	ori \$4,\$1,0x00000000	
<input checked="" type="checkbox"/>	0x0040000c	0x0000000c	syscall	10: syscall
<input type="checkbox"/>	0x00400010	0x2402000a	addiu \$2,\$0,0x0000000a	12: li \$v0, 10 # Systemcall code exit
<input type="checkbox"/>	0x00400014	0x0000000c	syscall	13: syscall

This will set a breakpoint on the syscall instruction.

Click on the **play** icon to start execute the program. The execution will now halt at the syscall instruction.

You can now **single-step** to continue to execution one instruction at the time.

Finally, test to **single-step-backwards** one or more instructions.