



# SIMULADORES MIPS

Prof. Tiago Gonçalves Botelho

# SPIN

- Simulador “clássico” multi-plataformas open-source, suas versões mais novas são desenvolvidas em Qt.
- Download: <http://spimsimulator.sourceforge.net/>
- Documentação:  
<http://pages.cs.wisc.edu/~larus/spim.html#information>



# MARS

- Simulador “moderno” multi-plataformas, desenvolvido em JAVA é open-source.
- Download:  
<http://courses.missouristate.edu/KenVollmar/MARS/download.htm>
- Documentação:  
<http://courses.missouristate.edu/KenVollmar/MARS/Help/MarsHelpIntro.html>



# MARS

- Fazendo o primeiro código assembly no MARS.

The screenshot shows the MARS 4.4 interface. The main window displays assembly code for a file named 'primeiro.asm'. The code includes data declarations, a main routine, and several instructions for loading, adding, and subtracting values from registers. The status bar at the bottom indicates 'Line: 16 Column: 1' and 'Show Line Numbers' is checked.

**Assembly Code:**

```
1 .data
2 .text
3 .main:
4     #Primeiro Código
5     li $s0,0      #a
6     li $s1,100    #b
7     li $s2,50     #c
8     li $s3,25     #d
9     li $s4,10     #e
10    li $s5,5       #f
11
12    add $t0, $s1, $s2    # $t0=b+c
13    add $t1, $s3, $s4    # $t1=d+e
14    add $t1, $t1, $s5     # $t1=$t1+f
15    sub $s0, $t0, $t1     # a=$t0-$t1 (a=(b+c)-(d+e+f))
16
```

**Registers Table:**

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194304
hi		0
lo		0

# MARS

## ○ INICIANDO A SIMULAÇÃO:

- Para se realizar a simulação é necessário ‘montar’ o programa, para isso deve-se clicar no ícone, localizado no Menu RUN->ASSEMBLE:



- Se o programa contiver pseudoinstruções é necessário habilitá-las.
- O Mars NÃO tem suporte a pipeline, porém pode simular o atraso nos saltos.



# MARS

The screenshot shows the MARS MIPS simulator interface. The main window is titled "/home/wagnerprates/Documents/Mips/Exemplos.asm/Fibonacci.asm - MARS 4.1". The interface includes a menu bar (File, Edit, Run, Settings, Tools, Help), a toolbar, and several panels:

- Text Segment:** Displays the assembly code. A red box highlights the first few lines, and a red arrow points to the instruction `la $a0, fibs` with the text "Habilita breakpoints." (Enables breakpoints).
- Labels:** A table showing labels and their addresses. The table has columns for Label, Address, and Value. The labels are: `Fibonacci.asm`, `loop`, `print`, `out`, `fibs`, `size`, `prompt`, `space`, and `head`.
- Registers:** A table showing the state of registers. The table has columns for Name, Number, and Value. The registers are: `$zero`, `$at`, `$v0`, `$v1`, `$a0`, `$a1`, `$a2`, `$a3`, `$t0`, `$t1`, `$t2`, `$t3`, `$t4`, `$t5`, `$t6`, `$t7`, `$s0`, `$s1`, `$s2`, `$s3`, `$s4`, `$s5`, `$s6`, `$s7`, `$s8`, and `$s9`.
- Data Segment:** A table showing the state of data. The table has columns for Address, Value (+0), Value (+4), Value (+8), Value (+c), Value (+10), Value (+14), Value (+18), and Value (+1c). The data is organized into rows, with the first row starting at address 0x10010000.
- Mars Messages:** A panel at the bottom showing the output of the simulation. It includes a "Run" button and a "Clear" button.

**Text Segment:**  
Mostra a memória de programa, com os opcodes. Destacando a instrução atual. Nesta tela são inseridos os breakpoints.

# MARS

**Data Segment:**  
Mostra a memória de dados, é possível alterar seus valores e o formato no qual são exibidos.

**Formatos de exibição.**

The screenshot displays the MARS MIPS assembler simulator interface. The top menu bar includes File, Edit, Run, Settings, Tools, and Help. Below the menu is a toolbar with various icons. The main window is divided into several panes. The 'Program Arguments' pane shows a list of instructions with their addresses, codes, and comments. The 'Data Segment' pane is highlighted, showing a table of memory addresses and their corresponding values. The 'Mars Messages' pane at the bottom shows the assembly process and a successful completion message.

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010004	0	0	0	0	0	0	0	0
0x10010008	1847617891	1700949265	1948282762	1701257327	1634887022	541025552	1008742952	544743485
0x1001000c	824196412	536	0	0	0	0	1700949265	1629516658
0x10010010	171599218	0	0	0	0	0	0	0
0x10010014	0	0	0	0	0	0	0	0
0x10010018	0	0	0	0	0	0	0	0
0x1001001c	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010024	0	0	0	0	0	0	0	0
0x10010028	0	0	0	0	0	0	0	0
0x1001002c	0	0	0	0	0	0	0	0
0x10010030	0	0	0	0	0	0	0	0
0x10010034	0	0	0	0	0	0	0	0
0x10010038	0	0	0	0	0	0	0	0
0x1001003c	0	0	0	0	0	0	0	0

Mars Messages: Run ID: 1  
Assemble: assembling /home/wagnerprates/Documents/Programs/Exemplos/.asm/Fibonacci.asm  
Assemble: operation completed successfully.

# MARS

The screenshot displays the MARS MIPS simulator interface. The main window shows the assembly code for a Fibonacci program. A red box highlights the 'Labels' window, which lists the labels and their corresponding addresses. A red arrow points from the 'Labels' window to a text box that explains its function.

**Labels Window:**

Label	Address
Fibonacci.asm	
loop	0x0040002d
print	0x00400058
out	0x00400070
fib	0x10010000
size	0x1001004c
prompt	0x10010050
space	0x10010057
head	0x10010059

**Text Box:**

Labels: Esta janela contém os labels declarados associado aos seus endereços.



# MARS

The screenshot displays the MARS MIPS simulator interface. The main window is titled "/home/wagnerprates/Documents/Mips/Exemplos.asm/Fibonacci.asm - MARS 4.1". The interface includes a menu bar (File, Edit, Run, Settings, Tools, Help) and a toolbar with various icons. The main area is divided into several panels:

- Text Segment:** Displays assembly code with columns for Dispt, Address, Code, Break, and Source. The code is for a Fibonacci calculator. A red box highlights the instruction `7: la $s0, fibs` with the comment `# load address of array`.
- Data Segment:** Displays memory addresses and their corresponding values in decimal, hexadecimal, and ASCII. A red box highlights the value `1667329647` at address `0x10010080`.
- Registers:** A table showing the state of MIPS registers. A red box highlights the `$s0` register, which contains the value `1667329647`. An arrow points from the highlighted value in the Data Segment to the `$s0` register.

**Registers:** Mostra os registradores do MIPS, e seus conteúdos. É possível alterá-los.

Register	Value
\$zero	0
\$at	1
\$v0	0
\$v1	0
\$a0	0
\$a1	0
\$a2	0
\$a3	0
\$t0	0
\$t1	0
\$t2	0
\$t3	0
\$t4	0
\$t5	0
\$t6	0
\$t7	0
\$s0	1667329647
\$s1	0
\$s2	0
\$s3	0
\$s4	0
\$s5	0
\$s6	0
\$s7	0
\$s8	0
\$s9	0
\$k0	0
\$k1	0
\$gp	268468224
\$sp	2147479548
\$fp	0
\$ra	0
\$pc	4194304
\$hi	0
\$lo	0

# MARS

Esta janela mostra as mensagens do compilador e do simulador.

É o console no qual os programas podem imprimir mensagens e receber dados pelo usuário.

The screenshot displays the MARS interface with the following components:

- Assembly Code:** A list of instructions with addresses, such as `0x00400010: addi $t0, $0, 1` and `0x00400020: sw $t0, 0($t0)`.
- Data Segment:** A table showing memory addresses and their corresponding values in hexadecimal and decimal.
- Registers:** A table on the right showing the state of various registers (e.g., \$t0, \$t1, \$t2) and their values.
- Mars Messages:** A console window at the bottom showing the output of the assembler and simulator. It displays the message: `Assemble: assembling /home/wagnerprates/Documents/FPs/Exemplos_asm/Fibonacci.asm` and `Assemble: operation completed successfully.`

A red arrow points from the "Mars Messages" window to the text above it, indicating that this is the console where programs can print messages and receive user input.

# MARS - SIMULAÇÃO



Executa o programa até o fim ou próximo breakpoint.



Executa a instrução atual.



Desfaz última instrução.

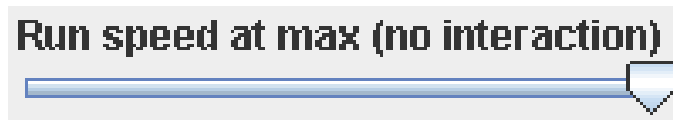


Reinicia o simulador, registradores e memória.



# MARS - SIMULAÇÃO

- A barra deslizante controla a velocidade de execução.



- Permitindo assim “ver a ação” ao invés do resultado final somente se a velocidade for reduzida.



# MIPS EM RESUMO – (1/2)

```
li    $r0, C
lui   $r0, C
move  $r0, $r1
```

```
$r0 <- C
$r0 <-  $2^{16} * C$ 
$r0 <- $r1
```

```
add   $r0, $r1, $r2
addi  $r0, $r1, C
sub   $r0, $r1, $r2
div   $r0, $r1, $r2
div   $r1, $r2
mul   $r0, $r1, $r2
neg   $r0, $r1
```

```
$r0 <- $r1 + $r2
$r0 <- $r1 + C
$r0 <- $r1 - $r2
$r0 <- $r1 / $r2
$lo <- $r1 / $r2, $hi <- $r1 mod $r2
$r0 <- $r1 * $r2 (sem overflow)
$r0 <- -$r1
```

```
slt   $r0, $r1, $r2
slti  $r0, $r1, C
sle   $r0, $r1, $r2
seq   $r0, $r1, $r2
sne   $r0, $r1, $r2
```

```
$r0 <- 1 se $r1 < $r2, $r0 <- 0 senão
$r0 <- 1 se $r1 < C, $r0 <- 0 senão
$r0 <- 1 se $r1 <= $r2, $r0 <- 0 senão
$r0 <- 1 se $r1 = $r2, $r0 <- 0 senão
$r0 <- 1 se $r1 <> $r2, $r0 <- 0 senão
```



# MIPS EM RESUMO – (2/2)

la	\$r0, adr	\$r0 <- adr
lw	\$r0, adr	\$r0 <- mem[adr]
sw	\$r0, adr	mem[adr] <- \$r0
beq	\$r0, \$r1, label	salto se \$r0 = \$r1
beqz	\$r0, label	salto se \$r0 = 0
bgt	\$r0, \$r1, label	salto se \$r0 > \$r1
bgtz	\$r0, label	salto se \$r0 > 0
beqzal	\$r0, label	salto se \$r0 = 0, \$ra <- \$pc + 1
bgtzal	\$r0, label	salto se \$r0 > 0, \$ra <- \$pc + 1
j	label	salto para label
jal	label	salto para label, \$ra <- \$pc + 1
jr	\$r0	salto para \$r0
jalr	\$r0	salto para \$r0, \$ra <- \$pc + 1

