

Simuladores MIPS

Chr stofer Caetano de Oliveira

SPIM

- 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

- A entrada e saída padrão do sistema é o console, através dele o usuário pode interagir com o programa lendo as mensagens e escrevendo dados.
- Para se utilizar o console é necessário o conhecimento de algumas “syscalls” (as básicas são compatíveis com o SPIM).

O que são *syscalls*?

- São serviços de sistema, principalmente para I/O.
- Para a chamada destes serviços é necessário indicar qual o serviço, através do registrador \$v0. E passar seus parâmetros, geralmente pelos registradores \$a0 .. \$a3.

Syscalls básicas

Nome	ID	Parâmetros (Entradas)	Saídas
print_int	1	\$a0: Inteiro à ser escrito	-
print_str	4	\$a0: Ponteiro para string	-
read_int	5	-	\$v0: Valor lido
Exit	10	-	-
print_char	11	\$a0: Caractere à ser escrito	-
read_char	12	-	\$v0: Caractere lido

Lista completa:

<http://courses.missouristate.edu/KenVollmar/MARS/Help/SyscallHelp.html>

Exemplo de uso das syscalls

...

```
move $a0, $t0 # Valor à ser impresso no console
```

```
li $v0, 1 # ID syscall (1 = print_int)
```

```
syscall # Dispara syscall
```

...

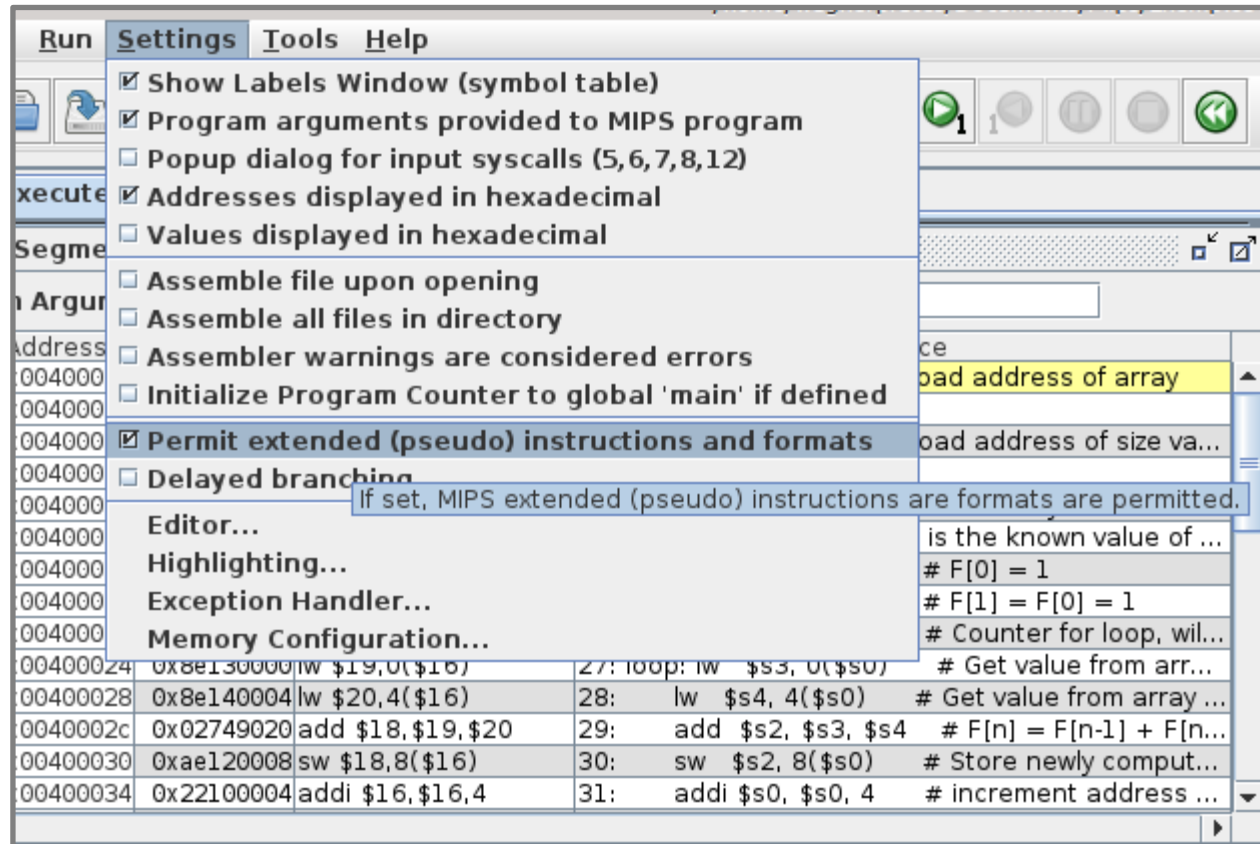
Pseudo Instructions

- São similares as macros, no momento da compilação são substituídas por uma ou mais instruções reais.
- Ex:
 - `li $t0, 0x80013002 # Load Immediate`
 - `lui $at, 0x8001 # $at ← 0x80010000`
 - `ori $t0, $at, 0x3002 # $t0 ← 0x80013002`

Pseudo Instructions

- Podem-se adicionar novas pseudo instruções facilmente no MARS.
- Podem ser incluídas no arquivo PseudoOps.txt. Este arquivo esta dentro do MARS 4_2.jar (abrir como .ZIP).
- Instruções no cabeçalho do arquivo, podem-se utilizar as pseudo instruções já inclusas como exemplo.

Habilitando Pseudo-Instructions



Iniciando a simulação

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



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

Interface do 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 menu bar includes File, Edit, Run, Settings, Tools, and Help. The toolbar contains various icons for file operations and execution. The main area is divided into several panels:

- Text Segment:** A table showing program arguments with columns for Bkpt, Address, Code, Basic, and Source. The current instruction is highlighted in yellow: `7: la $s0, fibs # load address of array`. A red box highlights the Bkpt column, and a text box with the text "Habilita breakpoints." points to it.
- Data Segment:** A table showing memory values at various addresses. A red box highlights the address `0x10010000`, and a text box with the text "Text Segment: Mostra a memória de programa, com os opcodes. Destacando a instrução atual. Nesta tela são inseridos os breakpoints." points to it.
- Labels:** A table showing labels and their addresses for the Fibonacci.asm file.
- Registers:** A table showing the state of registers, including Coproc 1 and Coproc 0.

The Mars Messages panel at the bottom shows the output of the assembly process: "Assemble: assembling /home/wagnerprates/Documents/Mips/Exemplos.asm/Fibonacci.asm" and "Assemble: operation completed successfully." A "Clear" button is visible below the messages.

Interface do Mars

The screenshot displays the Mars MIPS simulator interface. The main window is divided into several panes. The top pane shows the 'Text Segment' with a list of instructions and their addresses. The middle pane shows the 'Data Segment' with a table of memory addresses and their values. The bottom pane shows the 'Mars Messages' window with the output of the assembly process.

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

Formatos de exibição.

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
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	19	544698184	2037277037	1651066400	1667329647
0x10010060	1847617891	1700949365	1948283762	1701257327	1634887022	541025652	1008742952	544743485
0x10010080	824196412	536					1700949365	1629516658
0x100100a0	171599218							
0x100100c0	0							
0x100100e0	0							
0x10010100	0							
0x10010120	0							
0x10010140	0							
0x10010160	0							
0x10010180	0							
0x100101a0	0							

Mars Messages: Assemble: assembling /home/wagnerprates/Documents/Mips/Exemplos .asm/Fibonacci.asm
Assemble: operation completed successfully.

Interface do Mars

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

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

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

Interface do 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), a toolbar, and a status bar indicating "Run speed at max (no interaction)".

The central area is divided into several panels:

- Text Segment:** Shows assembly code with columns for Bkpt, Address, Code, Basic, and Source. The code is for a Fibonacci program, including instructions like `lui $t0, 0`, `li $t1, 1`, and a loop.
- Data Segment:** Shows memory addresses and their corresponding values in various registers.
- Registers:** A table showing the state of MIPS registers. The table has columns for Name, Coproc 1, and Coproc 0. The registers are listed from \$zero to \$lo.

An orange box highlights the Registers panel, and an arrow points from a text box to it.

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

Name	Coproc 1	Coproc 0	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

Interface do 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 MIPS simulator interface. At the top, there is a list of assembly instructions with their addresses and comments. Below this is the 'Data Segment' table, which shows memory addresses and their corresponding values. At the bottom, the 'Mars Messages' window is visible, showing the output of the assembler.

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
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	19	544698184	2037277037	1651066400	1667329647
0x10010060	1847617891	1700949365	1948283762	1701257327	1634887022	541025652	1008742952	544743485
0x10010080	824196412	536881465	1701336064	1651066400	1667329647	1847617891	1700949365	1629516658
0x100100a0	171599218	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

Mars Messages Run I/O

```
Assemble: assembling /home/wagnerprates/Documents/Mips/Exemplos .asm/Fibonacci.asm
Assemble: operation completed successfully.
```

Clear

Simulação



Executa o programa até o fim ou próximo breakpoint.



Executa a instrução atual.




Desfaz última instrução.



Reinicia o simulador, registradores e memória.

Simulação

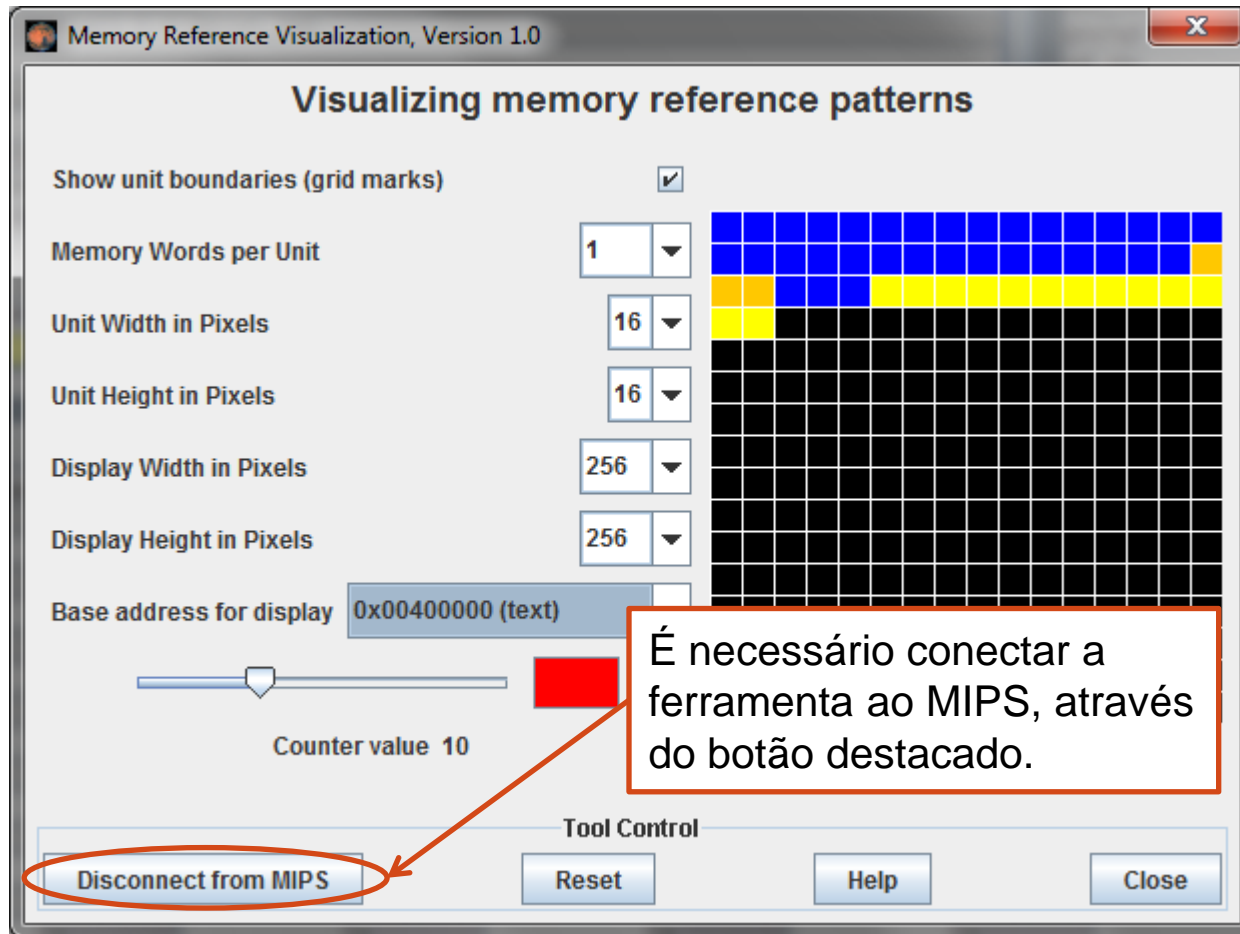
A barra deslizante  `Run speed at max (no interaction)` controla a velocidade de execução.

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

Ferramentas do MARS

- O Mars possui algumas ferramentas já incluídas além de um modelo para ajudar na geração de novas ferramentas.
- Estas ferramentas possuem as mais diversas funcionalidades, como mostrar o padrão de acesso à memória de dados ou de programa.
- Podem ser acessadas através do menu *Tools*.

Ferramentas do MARS



Organização do programa

- Os programas são organizados em seções.
 - .text: Esta seção contém as instruções em ASM.
 - .data: Esta seção contém os dados, estes são inicializados na memória de dados.

Ex: `str_1: .asciiz "Simulador MARS\n"`

Tipos de dados

Tipo	Descrição
.word	Tipo de dados genérico de 32 bits.
.half	Tipo de dados genérico de 16 bits.
.byte	Tipo de dados genérico de 8 bits.
.ascii	Sequência de caracteres: Ex: "ABCD\n".
.asciiz	String com terminação nula: Ex: "ABCD\n".
.float	Ponto flutuante de precisão simples.
.double	Ponto flutuante de precisão dupla.
.space	Reserva espaço de 'n' bytes.
.align	Alinha o próximo dado para endereço múltiplo de 2 ⁿ

Exemplos:

```
prompt: .asciiz "Digite sua idade: "  
.align 2  
vetor_1: .word 1, 2, 3, 4
```

Chamadas de função

- Os programas ASM são sequenciais, por isso o programador deve tomar cuidado com os saltos (jumps).
- Os parâmetros podem ser passados através de registradores, ou através da pilha (stack).
- Funções recursivas devem salvar seus endereços de retorno (\$ra), antes de chamar outra função.

Chamadas de função

- Para chamar uma função devem-se utilizar as instruções de saltos com 'link', é comum utilizar a JAL, porém é possível utilizar-se também as condicionais BGEZAL ou BLTZAL.
- Para o retorno utiliza-se a instrução "jr", usualmente associada ao registrador \$ra.

Chamadas de função - Exemplo

main:

...

`move $a0, $t0` *# some parameter*

`jal func` *# call func and writes the return
address on \$ra*

...

func:

... Func statements ...

`jr $ra` *# return to caller*

...

Exemplos

- 01 Console.asm:
Este exemplo mostra como se utilizar o console.
- O que faz:
 - Lê dois números informados pelo usuário (\$s1 e \$s2)
 - Troca-os de registrador (swap)
 - Imprime no console

Exemplos

- 02 Factorial.asm:
Este exemplo demonstra o uso de funções recursivas e pilha (stack).
- O que faz:
 - Calcula o fatorial do número informado pelo usuário (\$s1).
 - Imprime no console o resultado (\$s0).

Exemplos

- 03 Fibonnaci.asm:
 - Este exemplo gera uma série de Fibonnaci
- O que faz:
 - Calcula os 12 primeiros itens da série de Fibonacci.
 - Imprime a série no console.

Links Úteis

- Site com instruções ASM do MIPS:
http://en.wikipedia.org/wiki/MIPS_architecture
- Site com vários exemplos de MIPS Assembly:
<http://pt.scribd.com/doc/50236313/10/MIPS-using-pseudo-instructions>

Exercício

- Modificar Fibonnaci.asm, indicado no link abaixo:
 - <http://courses.missouristate.edu/KenVollmar/MARS/Fibonacci.asm>
- De modo que o usuário indique o tamanho da série de 1 até 12.
 - Deve mostrar uma mensagem de erro para tamanho fora deste intervalo.

Obrigado.