

1. Fundamentação

O MARS (*MIPS Assembler and Runtime Simulator*) é um programa simulador do processador MIPS 32 bits especialmente desenvolvido para fins educacionais. Este simulador caracteriza-se por ser uma máquina virtual, isto é, um programa que implementa a arquitetura de um determinado processador (no caso o MIPS32) em uma arquitetura distinta (em nosso caso o x86), sendo semelhante ao simulador SPIM. O simulador MARS pode ser obtido acessando o seguinte endereço: <http://courses.missouristate.edu/KenVollmar/MARS/>

Nas Figuras 1 e 2, as principais características da interface do MARS são ilustradas, sendo elas:

1. **Menus e Atalhos.**

2. **Modo de visualização:** edição ou execução.

3. **Editor:** para a programação da linguagem de montagem.

4. **Outros Atalhos:** principais menus e atalhos que possibilitam a montagem das instruções simbólicas, a execução do código binário e a configuração da velocidade de execução.

5. **Registers:** mostra os valores armazenados nos registradores de propósito geral do MIPS.

6. **Text Segment:** corresponde à área de memória onde é alocado o código do programa. Abaixo do título **Bkpt** (*breakpoint*), é possível selecionar a linha de código na qual deseja-se inserir uma parada de execução.

7. **Labels:** apresenta os endereços de memória que correspondem aos endereços simbólicos (labels) usados no código.

8. **Data Segment:** mostra os valores armazenados na memória. O conteúdo visualizado dependerá do contexto selecionado através da combo-box no rodapé da janela Data Segment. Por exemplo, a opção **.data** refere-se ao segmento de dados estáticos usados pelo programa aplicativo. Já a opção **sp** refere-se ao segmento de dados dinâmicos que armazena a estrutura de dados chamada de pilha (stack), que é crucial no suporte à chamada de procedimentos e, na opção **.kdata** refere-se ao segmento de dados de uso reservado para o núcleo (kernel) do sistema operacional. O usuário comum desenvolve programas aplicativos e, por isso, deve trabalhar na área de dados na região **.data**.

9. **Message:** contém as mensagens geradas pelo MARS para o usuário.

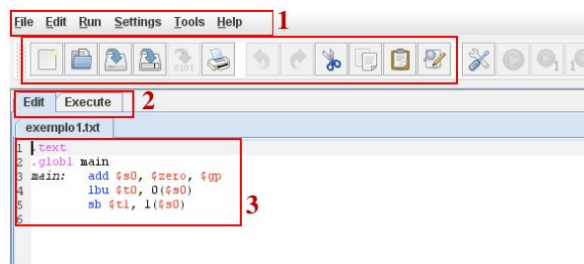


Figura 1. Janela de Edição do MARS.

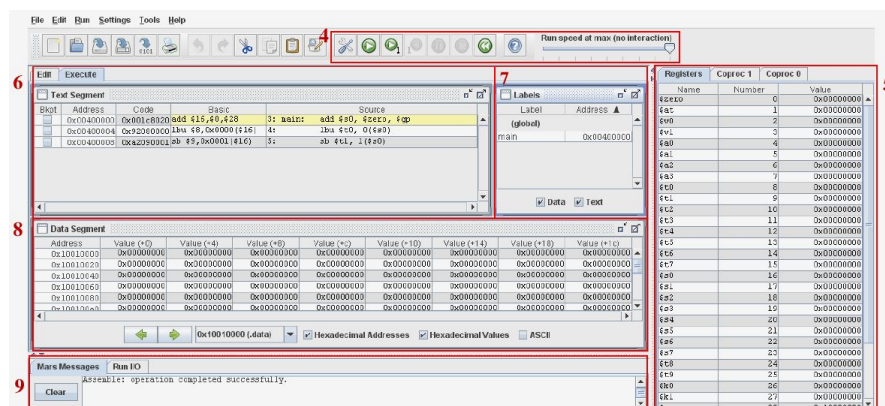


Figura 2. Janela de Execução do MARS.

2. Exercícios para o Entendimento do MARS

A seguir são propostos exercícios para entendimento da interface do MARS, fixação das instruções e aprendizagem de conceitos gerais de organização de computadores. Cabe salientar que os valores numéricos escritos em linguagem de montagem são, por default, decimais. Para a introdução de valores hexadecimais deve-se preceder o valor de “0x” (por exemplo, 0x89AACF1B).

Sugestão: abra uma pasta “Lab01” para armazenar os programas dos exercícios a serem resolvidos.

Exercício 1: Leitura/Escrita de um byte na memória

a) Crie um novo arquivo no editor do MARS (selecione o menu File -> New) com o seguinte código:

```
.text           # Define o início do Text Segment
.globl main     # Define o início do código do usuário
main:          # Label que define o início do código do usuário
add $s0, $zero, $gp # Copia o valor de $gp no registrador $s0
lbu $t0, 0($s0)  # Copia o byte da posição de memória [$s0+0] p/ $t0
sb $t1, 1($s0)   # Copia o byte menos significativo (à direita) de $t1 para a posição de
                # memória indicada por [$s0+1]
```

Observações:


O símbolo “#” indica que o conteúdo que segue à direita é comentário e será ignorado pelo montador.

O registrador \$gp (global pointer) é utilizado para o acesso ao segmento de dados estáticos do programa aplicativo (normalmente ele é inicializado pelo SO para apontar para o meio do segmento de dados estáticos).

Os registradores de uso geral do MIPS são todos de 32 bits. Os bytes nestes registradores são organizados da seguinte forma:

Byte mais significativo			Byte menos significativo
Byte 3	Byte 2	Byte 1	Byte 0

Cabe salientar que a instrução lbu transfere o byte residente em um endereço de memória para o byte 0 de um registrador e a instrução sb transfere o byte 0 de um registrador para um endereço de memória.

b) Salve o seu programa com a extensão .s, .asm ou .txt usando o menu File -> Save As. Por exemplo, *Exemplo1.txt*. A seguir, selecione a opção **Assemble** . Nesse momento, o simulador MARS muda para o modo execução, mostrando a janela **Text Segment** com o conteúdo digitado na coluna mais à direita, conforme mostrado na Figura 3.

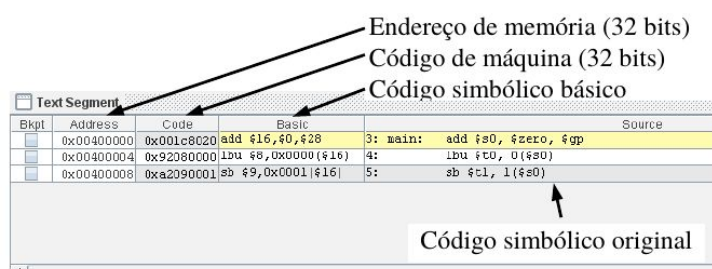


Figura 3 – Detalhamento da janela Text Segment (código).

Como indicado na Figura 3, o MARS apresenta, na coluna **Address**, os endereços de memória (em hexadecimal) onde estão armazenadas as instruções. As instruções em código de máquina (hexadecimal) encontram-se na coluna **Code**, seguidas do código simbólico correspondente nas colunas **Basic** e **Source**. Note que, no código original, os registradores são identificados por seus nomes simbólicos, enquanto que no código básico eles são identificados por seus números.

O conteúdo dos registradores pode ser apresentado no formato decimal ou hexadecimal. Caso você prefira, poderá visualizar o conteúdo dos registradores em hexadecimal ativando as opções **Hexadecimal Addresses** e **Hexadecimal Values**.

c) Agora, na janela **Registers** veja o valor presente no registrador \$gp (\$28). Memorize este valor.

\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0

Figura 4. Visualização do valor inicial atribuído ao registrador \$gp.

d) Agora, na janela **Data Segment**, selecione a opção current \$gp no combo-box (veja a Figura 5) e selecione o endereço que estava armazenado no registrador \$gp (268 468 224 = 0x1000 8000). Clique duas vezes para atribuir um novo valor para esta posição de memória, adicione o valor **VALOR1** (disponível na folha do relatório) e pressione Enter (veja a Figura 6 - o processo é semelhante a uma alteração em uma planilha do Excel).

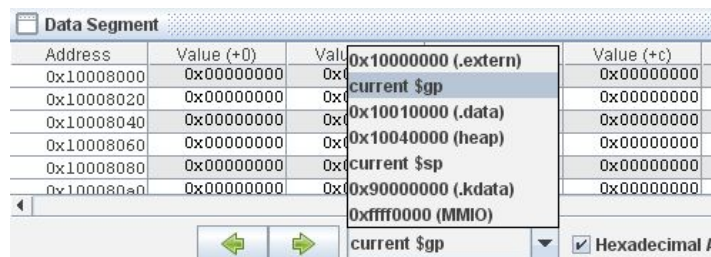


Figura 5. Seleção da seção current \$gp no Data Segment.

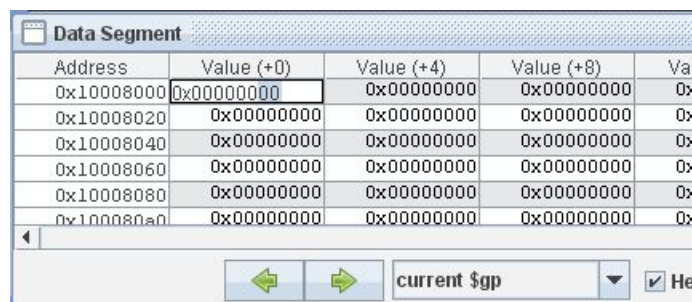



Figura 6. Atribuição do valor **VALOR1** ao endereço 268 468 224 (0x1000 8000 -> valor inicialmente atribuído a \$gp).

e) Da mesma forma, atribua o valor **VALOR2** ao registrador \$t1 (\$9). Clique duas vezes para atribuir um novo valor a esta posição de memória, adicione o valor **VALOR2** e pressione Enter (veja Figura 7). Assim, você atribuiu um valor de 8 bits para a posição de memória apontada por \$gp (**VALOR1**) e outro valor de 8 bits para o registrador \$t1 (**VALOR2**).

Registers			Coproc 1	Coproc 0
Name	Number	Value		
\$a0	4	0x00000000		
\$a1	5	0x00000000		
\$a2	6	0x00000000		
\$a3	7	0x00000000		
\$t0	8	0x00000000		
\$t1	9	0x00000000		
\$t2	10	0x00000000		

Figura 7. Atribuição do valor **VALOR2** ao o registrador \$t1 (\$9).

f) Agora aperte a tecla F7 ou o atalho . Esse comando permite rodar o programa instrução-por-instrução. Aperte F7 e veja a alteração no registrador \$16 (\$s0); aperte F7 novamente e veja a alteração no registrador \$8 (\$t0); aperte F7 mais uma vez e observe a alteração na posição de memória correspondente (no **Data Segment**).

g) Agora reinicialize a execução (Run->Reset), repita os passos dos itens “b” até “e”, e selecione no menu Run -> Go (F5)



. Compare com o que ocorreu no item “f”.

h) Responda às Questões 1.1 a 1.3 na folha do relatório de aula.

Exercício 2: Leitura e escrita de múltiplos bytes na memória

O objetivo deste exercício é verificar como o MARS mostra a sequência de bytes na memória.

a) Crie o seguinte arquivo no editor do MARS:

```
.text                # Define o início do Text Segment
.globl main          # Define início do código do usuário
main:                # Label que define do código do usuário
add $s0, $zero, $gp  # Copia o valor de $gp para $s0
lbu $t0, 0($s0)      # Copia o byte da posição de memória [$s0+0] p/ $t0
lbu $t1, 1($s0)      # Copia o byte da posição de memória [$s0+1] p/ $t1
lbu $t2, 2($s0)      # Copia o byte da posição de memória [$s0+2] p/ $t2
lbu $t3, 3($s0)      # Copia o byte da posição de memória [$s0+3] p/ $t3
lbu $t4, 4($s0)      # Copia o byte da posição de memória [$s0+4] p/ $t4
lbu $t5, 5($s0)      # Copia o byte da posição de memória [$s0+5] p/ $t5
lbu $t6, 6($s0)      # Copia o byte da posição de memória [$s0+6] p/ $t6
lbu $t7, 7($s0)      # Copia o byte da posição de memória [$s0+7] p/ $t7
lw  $t8, 0($s0)      # Copia a word da posição de memória [$s0+0] p/ $t8
```

b) Salve o seu programa (por exemplo, *Exemplo2.txt*) e altere a configuração para hexadecimal ativando as opções *Hexadecimal Addresses* e *Hexadecimal Values*.

c) Inicialize o valor **VALOR3** na posição de memória apontada por \$gp (0x1000 8000) e o valor **VALOR4** na posição apontada por \$gp+4 (0x1000 8004).

d) Utilize o comando F7 e execute o programa passo-a-passo. A cada passo, avalie a modificação dos conteúdos dos registradores.

e) Reinicialize a execução (Run->Reset), repita o passo “c”, configure a velocidade de execução para uma instrução por segundo e re-execute (F5).

f) Note que, antes de executar o programa acima, você armazenou uma *word* (palavra de 32 bits) no endereço apontado por \$gp e uma segunda *word* no endereço apontado por \$gp+4. Com base no código do programa e nos valores finais dos registradores, **responda às Questões 2.1 a 2.3 no relatório de aula.**