Universidade Federal de Santa Catarina - Centro Tecnológico – Departamento de Informática e Estatística
INE 5607 – Organização e Arquitetura de Computadores

Tutorial do Simulador MARS

(Adaptado da disciplina INE 5411 - Laboratório de Organização de Computador desenvolvido pelo Prof. Luiz Santos e Dr. Djones Lettnin a partir de aula congênere elaborada pela Prof. Lúcia Pacheco e pelo monitor Diogo Bratti)

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 MIPS) em uma arquitetura distinta (em nosso caso o IA-32), 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 (*break point*), 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.

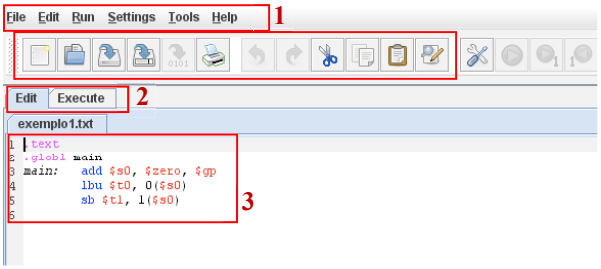


Figura 1. Janela de **Edição** do MARS

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
lw $t0, 0($s0)         # Copia a palavra da posição de memória [$s0] p/ $t0
sw $t1, 4($s0)         # Copia a o valor de $t1 para a posição de memória
                       indicada por [$s0+4]
```

Observações:
✓ O símbolo "#" indica que o conteúdo que segue naquela linha 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 sistema operacional para apontar para o meio do segmento de dados estáticos).

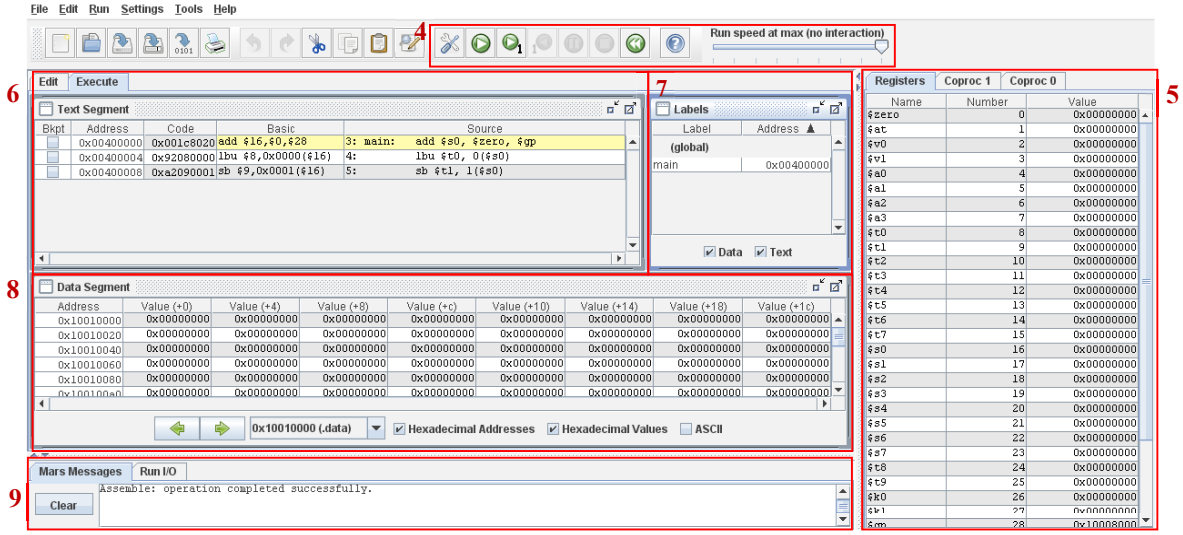


Figura 2. Janela de **Execução** do MARS.

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.

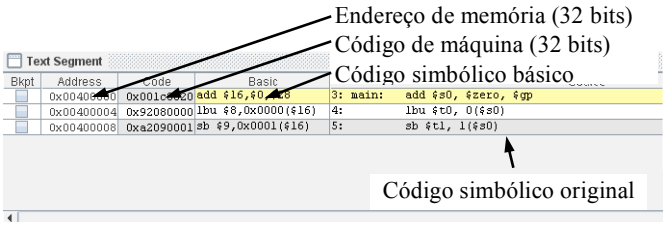
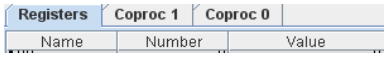


Figura 3 – Detalhamento da janela **Text Segment** (código).

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

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



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 16 (0x10) e pressione Enter (veja a Figura 6 - processo semelhante a uma alteração em uma planilha do Excel).

e) Da mesma forma, atribua o valor 0x20 (32) ao registrador $t1 ($9). Clique duas vezes para atribuir um novo valor a esta posição de memória, adicione o valor 0x20 e pressione Enter (veja Figura 7).

Assim, você atribuiu um valor de para a posição de memória apontada por $gp (0x0000 0010) e outro valor para o registrador $**t1** (0x0000 0020).

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**".

$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.

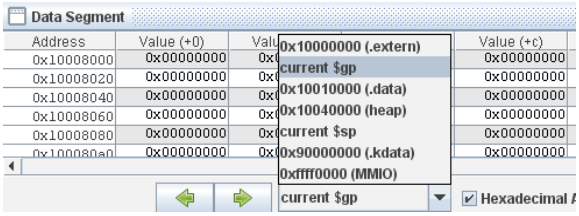


Figura 5. Seleção da seção current $gp no Data Segment.

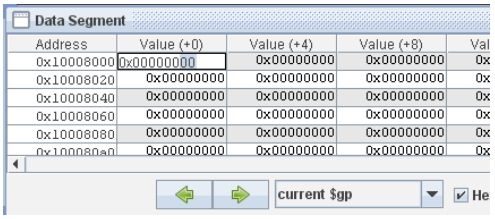


Figura 6. Atribuição do valor 16 (0x10) ao endereço 268 468 224 (0x1000 8000 => valor inicialmente atribuído a $gp).

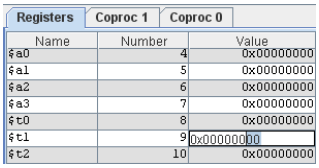


Figura 7. Atribuição do valor 0x20 (32) ao o registrador $t1 ($9)