

CENTRO TECNOLÓGICO  
DEPARTAMENTO DE INFORMÁTICA

Arquitetura de Computadores I – Turmas 01 e 02 (EARTE) – 2021/2  
Prof. Rodolfo da Silva Villaca – [rodolfo.villaca@ufes.br](mailto:rodolfo.villaca@ufes.br)

**Laboratório I – Iniciando com o MARS e a  
Linguagem Assembly MIPS<sup>1</sup>**

**Objetivos:**

- Carregar e executar programas em assembly MIPS;
- Examinar posições de memória;
- Examinar registradores;
- Executar programas passo-a-passo.

**Descrição:**

O MARS<sup>2</sup> é um simulador escrito em Java que roda programas para as arquiteturas baseadas em MIPS. O simulador pode carregar e executar programas em assembly MIPS. O processo de translação do código MIPS é transparente para o usuário. Isto significa que você não tem que tratar com o assembler, linker e loader diretamente. Após você escrever seu programa em assembly, o simulador vai se encarregar de executá-lo e apresentar o resultado da execução em uma das saídas disponíveis (em geral, a saída padrão definida em uma das janelas do MARS).

**Atividades:**

**1. Primeiros passos com o MARS**

Usando a janela de edição do MARS, edite o programa p1.asm a seguir. No editor, o caractere (#) marca o início de um comentário; um nome seguido (:) é um label (identificador) e os nomes que se iniciam com (.) são diretivas para o montador (assembler).

p1.asm:

```
.data
msg1: .asciiz "Digite um valor inteiro: "

.text
# print message on shell
li $s0, 0x00400000      # save return address in $s0
li $v0, 4               # system call for print_str
la $a0, msg1            # address of string to print
syscall

# now get an integer from the user
li $v0, 5               # system call for read_int
syscall                 # the integer is placed in $v0

# do some computation here with the number
addu $t0, $v0, $0       # move the number to $t0
sll $t0, $t0, <digit>    # change <digit> with the last digit of your UFES id (matricula)
# print the result in shell
li $v0, 1               # system call for print_int
```

1 Material adaptado das apostilas do curso do Prof. Virgil Bistriceanu

Disponível em: [www.cs.iit.edu/~virgil/cs470/Labs/](http://www.cs.iit.edu/~virgil/cs470/Labs/)

2 Disponível em: <http://courses.missouristate.edu/kenvollmar/mars/download.htm>

**CENTRO TECNOLÓGICO**  
**DEPARTAMENTO DE INFORMÁTICA**

```
addu $a0, $t0, $0      # move number to be printed in $a0
syscall

# restore now the return address in $ra and return from main
addu $ra, $0, $s0      # return address back in $ra
jr $ra                 # return to the main label
```

Monte o programa usando o menu Run → Assemble, ou F3 e execute o programa. Não se esqueça de trocar o dígito da linha indicada para o seu último dígito da sua matrícula. Observe o resultado para diferentes números digitados como entrada. Preencha a tabela a seguir com diversos valores entrada e de saída (inteiros, positivos e negativos) para a execução do programa.

Entrada	Saída

Responda:

- Qual é a equação que define a transformação dos valores de entrada para saída?
- Qual o valor armazenado em \$s0 e por que ele foi passado a \$ra?

## 2. Usando o MARS para entender a arquitetura

Usando o MARS, é possível observar a ocupação da memória e dos registradores, assim como o endereçamento de instruções e dados de um programa MIPS. Também é possível fazer a execução de um programa passo a passo, permitindo a observação detalhada da variação dos valores dos registradores e do fluxo de execução dos programas.

Modifique o programa p1.asm, incluindo um label L1: na 2ª linha do segmento de texto, que passará a ser:

```
L1: li $v0, 4      # system call for print_str
```

Observe a tabela de símbolos do programa. Dois labels devem aparecer (L1 e msg1). Anote os valores desses labels (eles estão em hexadecimal, prefixados sempre com 0x) e explique como esses valores foram definidos.

Label	Valor	Justificativa

Neste ponto, você já deverá entender como um programa é armazenado, como os valores dos labels são definidos e como os valores dos registradores são modificados. Importantes questões podem ser respondidas agora:

- Qual o tamanho das instruções do programa? Quantos bytes no total ele ocupa?
- A instrução li \$s0, 0x00400000 tem qual propósito? E se esse valor fosse mudado, o que aconteceria?
- Essa instrução foi substituída por duas instruções. Quais? Por que?

**CENTRO TECNOLÓGICO**  
**DEPARTAMENTO DE INFORMÁTICA**

- A instrução `la $a0, msg1` também foi substituída por 2 instruções. Quais? Por que?

É possível fazer a execução passo-a-passo do programa usando o botão correspondente na interface ou a tecla F7. Execute o programa passo-a-passo e observe a mudança nos valores do registrador PC, assim como a instrução executada no momento, além das mudanças nos valores dos registradores utilizados no código.

Em 3 pontos do programa a instrução `syscall` foi utilizada. Note que essa instrução faz a chamada de um serviço (I/O) e que alguns argumentos devem ser passados antes da chamada. Anote na tabela a seguir os registradores que são alterados em cada uma das instruções `syscall` do programa, assim como os valores dos argumentos usados como entrada e saída nas chamadas.

Endereço	Tipo de Serviço	Entrada (registrador)	Saída (registrador)
0x00400014			
0x0040001C			
0x00400030			

Por opção de projeto, a CPU MIPS foi projetada com:

- Uma unidade de inteiros (ULA dentro da CPU);
- Um coprocessador 0, com suporte ao tratamento de interrupções, exceções e memória virtual;
- Um coprocessador 1, correspondendo a uma unidade de operações em ponto flutuante.

Estes tópicos ainda não foram vistos no curso, porém os coprocessadores já podem ser testados:

- Escolha a aba Coproc 0 na janela direita do simulador. Digite um valor de entrada “muito grande” e observe o erro de execução e sua indicação em um dos registradores. Qual registrador indica o erro? Qual o erro foi indicado? Como essa indicação poderia ser usada pelo SO?
- Inclua uma declaração da variável `x` no segmento de dados com o comando `x: .float 1.5674`. Observe a tabela de labels do programa. O valor da variável `x` está correto? Que representação é essa?
- Inclua as instruções a seguir no segmento de código após o último `syscall`:

```
l.s $f0, x  
add.s $f2, $f0, $f0
```

- Escolha a aba Coproc 1 na janela direita do simulador. O que representam os valores que foram gerados nos registradores do Coproc 1?

#### 4. Execução

- Grupos de até 3 (três) alunos;
- Submissão até 02/12 (9h);