

"Os humanos são pensadores lentos, desleixados e brilhantes. Computadores são rápidos, cuidadosos e estúpidos" (John Pfeiffer).

Programas em Assembly, I/O e exercícios

Paulo Ricardo Lisboa de Almeida

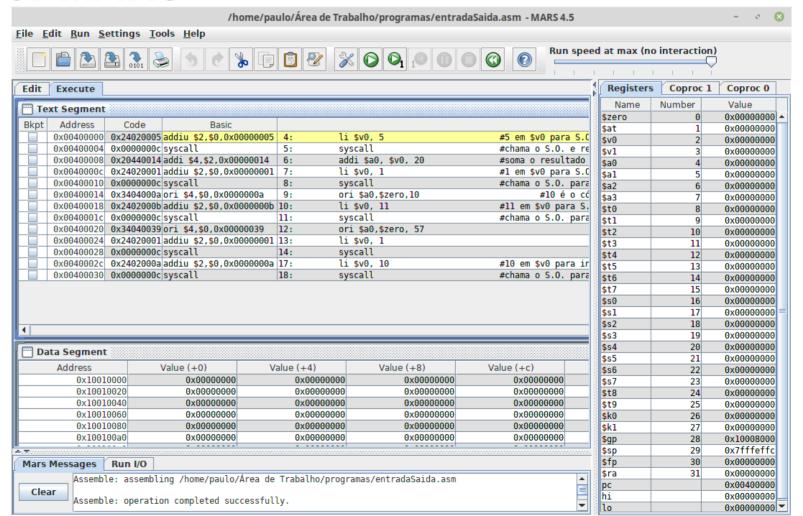


Simulador MARS

- MARS (MIPS Assembler and Runtime Simulator)
 - Simulador MIPS
 - Desenvolvido por Pete Sanderson e Kenneth Vollmar
- Gratuito e de código aberto
 - Licença MIT
 - Download: courses.missouristate.edu/kenvollmar/mars/license.htm
- O programa é um jar, então basta ter o Java instalado na sua máquina
- Maneira simples de rodar:
 - Abra um terminal no diretório onde o jar foi salvo
 - Digite (assumindo que o nome do jar baixado é Mars4_5.jar)
 - java -jar Mars4_5.jar



Simulador MARS





Simulador MARS

- Abra o MARS
- Crie um novo arquivo
- Salve como aula.asm



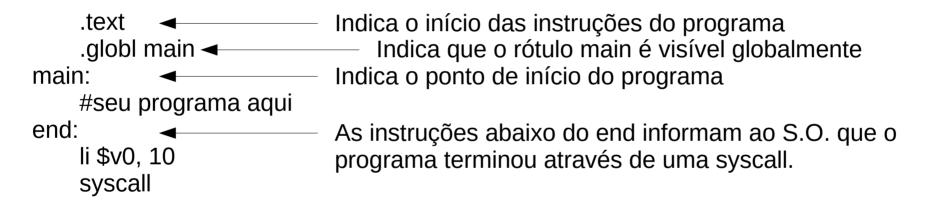
Programas em Assembly

- Programas escritos em assembly geralmente são salvos com a extensão .s ou .asm
- Colocamos apenas um nível de indentação
 - Mais a esquerda, colocamos definições de rótulos e importações
 - A uma tabulação de distância, colocamos todos os comandos
- Sempre comente o máximo possível em seus programas



Criando seu programa

• Inicialmente, nossos programas terão o seguinte formato:





Exemplo

• Escreva o seguinte no programa aula.asm

```
.text
.globl main
main:
li $s5, 15  #pseudoinstrução para carregar o imediato 15 em $s5
end:
li $v0, 10  #10 em $v0 para indicar ao S.O. que o programa acabou syscall #chamar o S.O.
```



Assemble

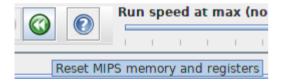
- No MARS, vá em File → open, e abra o arquivo que você salvou
- Para "montar" o código e torná-lo executável, clique no botão "Assemble"

- Aperte o "play" e veja o resultado do seu programa nos registradores
- Note que o li é uma pseudoinstrução, e foi traduzida para instruções reais

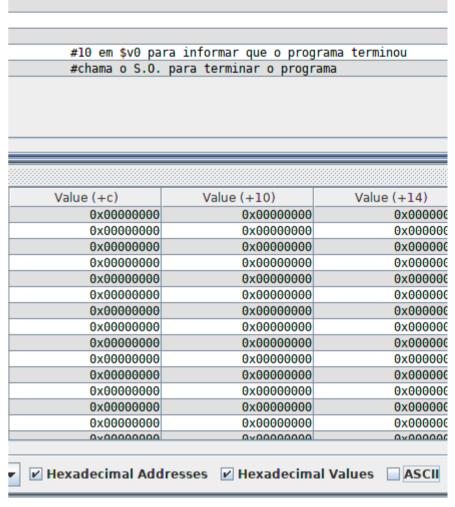


Dicas

- Sempre verifique o formato em que os valores estão sendo exibidos nos registradores
 - Binário ou hexa
- Você pode zerar os registradores na interface
 - Necessário se deseja rodar o programa novamente (para zerar o PC)



 O MARS segue uma implementação específica do MIPS, então alguns detalhes podem ser diferentes dos apresentados em Patterson e Henessy (2014).



#10 é o código do \n

#chama o S.O. para escrever

#11 em \$v0 para S.O escrever \$a0 na tela como char



Exercícios

1.Faça um programa que calcule 9! e armazene o resultado em \$s0 Veja como funciona a instrução mul. Exemplo: mul \$1,\$2,\$3



Entrada e Saída

- A entrada e saída é feita com ajuda do Sistema Operacional
 - O MARS inclui um Sistema Operacional minimalista para nossas simulações
- Precisamos fazer syscalls
 - Colocamos o código da operação desejada em \$v0
 - Instrução syscall devolve o controle ao S.O., que olha para \$v0 e faz o requisitado
 - Entradas e saídas são feitas na tela de RUN I/O
- Veja algumas chamadas de sistema válidas em
 - courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html



Exercícios

- 2.Modifique o exercício 1 para que seja calculado n!, onde n é solicitado do usuário. Além disso, utilize a instrução mult no lugar de mul. Como mult funciona? Por que o resultado é armazenado em dois registradores?
- 3.Faça um programa que solicita continuamente valores inteiros ao usuário. O programa termina quando o usuário digita -1. Ao final, o programa deve exibir a soma e a média dos valores digitados.
- 4. Modifique o programa do exercício anterior, de forma que o programa termina quando o usuário digita -1, **ou** quando a soma atingir um valor maior ou igual a 2048.
- 5.Faça um programa que calcule o vigésimo número da sequência de Fibonacci e o armazene o registrador \$s1.
- 6.Modifique o exercício anterior para que o índice do número de fibonacci a ser impresso seja requisitado do usuário, e o resultado seja impresso no console.



Exercícios

7.Crie um programa para um caixa eletrônico que calcula o menor número possível de cédulas que deve ser entregue a um usuário quando ele fizer um saque. Considere que a entrada do programa é o valor do saque, e a saída são as notas que o usuário receberá. Exiba as quantidades de notas como inteiros simples na tela, na seguinte ordem: notas de 50, 20, 10 e 5 reais, e moedas de 1 real. Exemplo se o usuário solicitar um saque de 87 reais:

11112

Dica. Para imprimir um espaço :

li \$a0, 32 #32 é o código do espaço

li \$v0, 11 #11 em \$v0 para S.O escrever \$a0 na tela como char

syscall #chama o S.O. para escrever



Referências

- D. Patterson; J. Henessy. Organização e Projeto de Computadores.
 4a Edição: Interface Hardware / Software. Elsevier Brasil, 2014.
- Andrew S. Tanenbaum. Organização estruturada de computadores.
 5. ed. São Paulo: Pearson, 2007.
- Harris, D. and Harris, S. **Digital Design and Computer Architecture**. 2a ed. 2012.
- courses.missouristate.edu/KenVollmar/mars/
- Bob Plantz. Introduction to Computer Organization: A Guide to X86-64 Assembly Language and GNU/Linux. 2019.

