

# LABORATÓRIO ASSEMBLY MIPS

Prof<sup>a</sup> Monica Magalhães Pereira

# Plano de aula

- Nessa aula de laboratório, você será introduzido ao ambiente de programação MARS no qual você desenvolverá programas em linguagem assembly MIPS.
- Para fazer o download do MARS, acessar:  
<http://courses.missouristate.edu/KenVollmar/MARS/>

# CONHECENDO O SIMULADOR

# Conhecendo o simulador

- Faça o download do simulador MARS em:

<http://courses.missouristate.edu/KenVollmar/mars/>

- Faça o download do arquivo lab1.asm no SIGAA
- Abra o simulador MARS e o arquivo lab1.asm (**File...Open**)
- Você poderá visualizar e editar o programa em assembly na aba **Edit**
- Você poderá visualizar a simulação do programa, acompanhando o comportamento das memórias, do banco de registradores e a sequência de execução, na aba **Execute**
- Para iniciar a simulação, clique no botão



# Conhecendo o simulador

- Com o programa aberto, responda as seguintes questões:
  1. O que significa `.data` no código?
  2. O que significa `.text` no código?
  3. O que significa `.globl` no código?
  4. Qual o endereço de memória da primeira instrução?
  5. Em qual base está esse endereço?
  6. Qual o endereço de memória da segunda e da terceira instruções?
  7. Por que o endereço de cada instrução tem uma diferença de 4 unidades entre 1 e o outro?



# Conhecendo o simulador

8. Qual o primeiro endereço da memória de dados?
9. Para esse programa, o que existe na memória de dados?
10. Qual o valor armazenado no registrador PC?
11. Em que base está o valor do registrador PC?
12. O que representa esse valor?
13. Existe algum outro registrador com valores? Quais?
14. Qual a base dos valores observados no item 10? O que esses valores representam?

# Conhecendo o simulador

- Para simular o programa, você irá utilizar o painel de simulação



- O botão  serve para simular a execução do seu programa inteiro. O simulador só irá parar ao finalizar a execução
- O botão  serve para simular a execução passo a passo. Assim, você poderá acompanhar a execução.

# Conhecendo o simulador

- Inicie a simulação em apenas UMA etapa, utilizando o botão
- Responda as seguintes questões:



15. O que aconteceu com o registrador PC?

16. Algum outro registrador mudou de valor? Qual registrador? Qual o novo valor?



# Conhecendo o simulador

- Simule mais uma etapa e responda:

17. Qual o novo valor do PC?

18. Algum outro registrador mudou de valor? Qual registrador? Qual o novo valor?

# Conhecendo o simulador

- Simule uma terceira etapa e responda:

19. Qual o valor do PC?


20. Indique quais os registradores mudaram de valor e qual o valor de cada um.

21. O que esse programa faz?

22. O que acontece se você tentar mais uma etapa?

# CONHECENDO A MEMÓRIA DE DADOS

# Memória de dados

- Abra o programa lab2.asm
- “Assemble” o programa  , observe a memória de dados e responda as questões.
- Dica: para ajudar a entender a memória de dados, é possível mudar a exibição dos valores entre hexadecimal, decimal ou ASCII, no menu abaixo da memória.



0x10010000 (.data)



Hexadecimal Addresses



Hexadecimal Values



ASCII

# Memória de dados

23. Cada célula de memória contém quantos bytes?

24. No programa assembly, onde devem estar os valores que serão armazenados na memória?

- a. *.text*
- b. *.data*
- c. *.globl*

25. Qual o tamanho em byte dos valores das declarações:

- a. *.byte 4,3,2,1*
- b. *.half 8,7,6,5*
- c. *.word 1,2,3,4*
- d. *.ascii "EFG"*

# Memória de dados

No menu **Execute**, procure o painel Labels e responda:

26. Quais os labels que aparecem no painel?

27. O que significam esses labels?

28. Além do label, o painel mostra uma segunda. O que tem nessa coluna?

# USANDO ENTRADA E SAÍDA DE DADOS

# Entrada e Saída de dados

- Para operações de entrada e saída de dados é necessário utilizar instruções especiais para chamar o sistema operacional
- A instrução de chamada do SO é o SYSCALL
- Porém, além do SYSCALL, são necessárias outras informações adicionais que indicam para que o SO está sendo chamado



# Entrada e Saída de dados

- Para fazer a entrada/saída os passos são:

Passo 1) Carregar no registrador \$v0 o código da operação de entrada/saída

Passo 2) Em caso de saída, carregar o valor do argumento de saída em algum dos registradores \$a0, \$a1, \$a2 ou \$f12, conforme especificado pela operação

# Entrada e Saída de dados

- Para conhecer todas as possíveis chamadas de sistema que o MIPS pode realizar, abrir o *Help*, selecionar “Syscalls” e observar a tabela.
- Com o Help aberto em Syscalls responda:

29. Qual o código deve ser passado para \$v0 para imprimir um inteiro no terminal?

30. Qual o código deve ser passado para \$v0 para imprimir uma string?

31. Qual o código deve ser passado para \$v0 para LER um inteiro?

32. Qual o código deve ser passado para \$v0 para LER uma string?

# Treinando Saída

- Abra o programa lab3.asm, inicia a simulação e responda as seguintes questões:

33. O que esse programa faz?

34. Onde está armazenada a *str1*?

# ATIVIDADE FINAL

35. Modifique o programa lab3.asm para implementar em assembly MIPS o seguinte programa:

```
int num1, num2, resultado;  
printf("Digite o primeiro numero: \n");  
scanf("%d", &num1);  
printf("Digite o segundo numero: \n");  
scanf("%d", &num2);  
resultado = num1 - num2;  
printf("O resultado e: %d\n", resultado);
```

ATENÇÃO: VOCÊ DEVE ESCOLHER QUAIS REGISTRADORES IRÃO SER UTILIZADOS.  
CONTANTO QUE SEJAM REGISTRADORES \$t ou \$s

# Bibliografia

- PATTERSON, D. A. & HENNESSY, J. L.

**Organização e Projeto de Computadores** – A Interface  
Hardware/Software. 3ª ed. Campus,

CAPÍTULO 2

- MIPS Assembly Language

<http://www.inf.uni-konstanz.de/dbis/teaching/ws0304/computing-systems/download/rs-05.pdf>

Introdução Curta ao MIPS

<http://www.di.ubi.pt/~desousa/2011-2012/LFC/mips.pdf>