

# Laboratório 2

## Assembly MIPS

Prof. Gustavo Girão

# 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/>

# 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:
  - Carregar no registrador \$v0 o código da operação de entrada/saída
  - 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:
  - Qual o código deve ser passado para \$v0 para imprimir um inteiro no terminal?
  - Qual o código deve ser passado para \$v0 para imprimir uma string?
  - Qual o código deve ser passado para \$v0 para LER um inteiro?
  - 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:
- O que esse programa faz?
- Onde está armazenada a *str1*?

# Questões da lista

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);
```



20 min

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



# Salto Incondicional - JUMP

- *jump to Label*

- j **LABEL**

O que é LABEL?

- Ex:

...

operação 0

j **PARA\_AQUI**

operação 1

operação 2

**PARA\_AQUI**: operação 3

operação 1 e operação 2 não serão executadas, pois serão puladas

# Salto Incondicional - JUMP

- Baixe todos os programas da aula de hoje, disponível na turma virtual numa pasta em seu computador
- Abra o programa lab4.asm.
- Simule a execução, utilizando o painel de simulação e responda as questões a seguir:



# Questões da lista

Demonstre, em linguagem C, uma implementação em em alto nível que faça a mesma coisa que o código assembly lab4.asm.

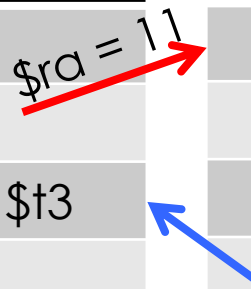


15 min

# JUMP com retorno

- *Jump and link*
  - jal **EndereçoProcedimento**
  - Pula para a instrução em **EndereçoProcedimento** e grava o endereço da próxima instrução em **\$ra**
- *Jump register*
  - jr **\$ra**
  - Pula de volta para seguir o fluxo de instruções anterior

Endereço	Instrução	Endereço	Instrução
...	...	35	<b>Aqui:</b> add \$t1, \$t2, \$t3
10	jal <b>Aqui</b>	36	add \$t1, \$t1, \$t1
11	add \$t1, \$t2, \$t3	37	sub \$t1, \$t1, \$t4
...	...	38	jr \$ra



# Instruções MIPS

## Desvio incondicional

Endereço	Instrução
...	...
10	jal <b>Aqui</b>
11	add \$t1, \$s4, \$t3
...	...
...	...
...	...
...	...
78	jal <b>Aqui</b>
79	
...	...


Endereço	Instrução
35	<b>Aqui:</b> add \$t1, \$t2, \$t3
36	add \$t1, \$t1, \$t1
37	sub \$s4, \$t1, \$t4
38	jr \$ra

# Instruções MIPS

## Desvio incondicional

Endereço	Instrução
...	...
10	jal <b>Aqui</b>
11	add \$t1, \$s4, \$t3
...	...
...	...
...	...
...	...
78	jal <b>Aqui</b>
79	
...	...

$\$ra = 11$



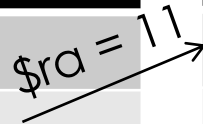
Endereço	Instrução
35	<b>Aqui:</b> add \$t1, \$t2, \$t3
36	add \$t1, \$t1, \$t1
37	sub \$s4, \$t1, \$t4
38	jr \$ra

# Instruções MIPS

## Desvio incondicional

Endereço	Instrução
...	...
10	jal <b>Aqui</b>
11	add \$t1, \$s4, \$t3
...	...
...	...
...	...
...	...
78	jal <b>Aqui</b>
79	
...	...

\$ra = 11



Endereço	Instrução
35	<b>Aqui:</b> add \$t1, \$t2, \$t3
36	add \$t1, \$t1, \$t1
37	sub \$s4, \$t1, \$t4
38	jr \$ra



# Instruções MIPS

## Desvio incondicional

Endereço	Instrução	Endereço	Instrução
...	...	35	<b>Aqui:</b> add \$t1, \$t2, \$t3
10	jal <b>Aqui</b>	36	add \$t1, \$t1, \$t1
11	add \$t1, \$s4, \$t3	37	sub \$s4, \$t1, \$t4
...	...	38	jr \$ra
...	...		
...	...		
...	...		
78	jal <b>Aqui</b>		
79			
...	...		

*\$ra = 11*

*\$ra = 79*



# Instruções MIPS

## Desvio incondicional

Endereço	Instrução	Endereço	Instrução
...	...	35	<b>Aqui:</b> add \$t1, \$t2, \$t3
10	jal <b>Aqui</b>	36	add \$t1, \$t1, \$t1
11	add \$t1, \$s4, \$t3	37	sub \$s4, \$t1, \$t4
...	...	38	jr \$ra
...	...		
...	...		
...	...		
78	jal <b>Aqui</b>		
79			
...	...		

# Questões da lista

- a) O que faz o programa lab5.asm?
- b) Explique a diferença desse programa para o lab4.asm.



10 min

# Exercício 4

Implemente e simule em assembly MIPS o algoritmo abaixo:

```
int num1, num2, resultado;  
printf("Digite o primeiro numero: \n");  
scanf("%d", &num1);  
printf("Digite o segundo numero: \n");  
scanf("%d", &num2);  
if (num1==0) faça  
    num1 = num1+num2-1;  
else  
    num1 = num1-num2+1;  
printf("O resultado e: %d\n", num1);
```



20 min

# 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>**