

# Arquitetura e Organização de Computadores

Universidade do Estado de Mato Grosso

Faculdade de Ciências Exatas e Tecnológicas

Prof Me. João Ricardo dos Santos Rosa

[joao.santos@unemat.br](mailto:joao.santos@unemat.br)

2024/02

# ASSEMBLY MIPS

# INTRODUÇÃO

# *Introdução*

- Instruções:
  - São as palavras da linguagem de máquina.
- Conjunto de instruções:
  - É o vocabulário da linguagem de máquina
- Todo o processador é construído com base nos mesmos princípios fundamentais , mesmas tecnologias de hardware e realiza as mesmas operações básicas.

# *Introdução*

- Objetivo no projeto de construção de microprocessadores
  - Encontrar um conjunto de instruções que facilite tanto a construção do hardware quanto a do compilador e, ao mesmo tempo, maximize a performance e minimize os custos

# OPERAÇÕES

# Operadores

- Dentro da IDE do Assembly do simulador MARS, você verá algumas cores diferenciadas cujo seu propósito são representar funções distintas. Por exemplo

Componente	Função
#Sim	Comentário
.data	Diretiva
li, la, add, sub, mpy	Instrução
\$v0, \$a0, \$t0, \$t1, \$s0	Registradores
Principal: main	Estiquetas
12.56, 7, 0XAFB	Dados
"Olá Mundo", João", "Oi"	String
'S', 'N'	Caractere

# IDE

EditExecute

Hello word 2\*

1 .data

2 segmento de dados:

3 .data Subsequent items stored in Data segment at next available address

4 #aqui são definidas as constantes e variáveis

5

6 .text

7 segmento de código:

8 #Aqui ficam as instruções mips

Line: 8 Column: 33Show Line Numbers

Mars MessagesRun I/O

Clear

RegistersCoprocc 1Coprocc 0

Name	Float	Double
\$f0	0x00000000	0x0000000000000000
\$f1	0x00000000	
\$f2	0x00000000	0x0000000000000000
\$f3	0x00000000	
\$f4	0x00000000	0x0000000000000000
\$f5	0x00000000	
\$f6	0x00000000	0x0000000000000000
\$f7	0x00000000	
\$f8	0x00000000	0x0000000000000000
\$f9	0x00000000	
\$f10	0x00000000	0x0000000000000000
\$f11	0x00000000	
\$f12	0x00000000	0x0000000000000000
\$f13	0x00000000	
\$f14	0x00000000	0x0000000000000000
\$f15	0x00000000	
\$f16	0x00000000	0x0000000000000000
\$f17	0x00000000	
\$f18	0x00000000	0x0000000000000000
\$f19	0x00000000	
\$f20	0x00000000	0x0000000000000000
\$f21	0x00000000	
\$f22	0x00000000	0x0000000000000000
\$f23	0x00000000	
\$f24	0x00000000	0x0000000000000000
\$f25	0x00000000	
\$f26	0x00000000	0x0000000000000000
\$f27	0x00000000	
\$f28	0x00000000	0x0000000000000000
\$f29	0x00000000	
\$f30	0x00000000	0x0000000000000000
\$f31	0x00000000	

Condition Flags

☐ 0☐ 4

☐ 1☐ 5

☐ 2☐ 6

☐ 3☐ 7

# Tipos de Operadores

- Os tipos de dados em Assembly trata-se de uma diretiva e uma diretiva inicia-se sempre com ponto (.)

Operadores	Tipo
<b>.ascii</b>	Usado para decalcar uma constante ou variável do tipo texto (String)
<b>.word</b>	Usado para declarar uma constante ou variável do tipo (Inteiro)
<b>.float</b>	Usado para declarar uma constante ou variável do tipo ponto flutuante (Float)
<b>.double</b>	Usado para declarar uma constante ou variável do tipo ponto flutuante mais abrangente (Double)
<b>.byte</b>	Usado para declarar uma constante ou variável do tipo caractere (char)
<b>.space&lt;qtde&gt;</b>	Usado para declarar uma constante ou variável do tipo string com espaços em branco



# *Tipos de Operadores*

## **.data**

mensagem: **.ascii** "Olá mundo" #declarando uma variável tipo String

dia: **.word** 22 #Declarando uma variável do tipo Int

Kg: **.float** 55.6 #Declarando uma variável do tipo float

Km: **.double** 10000.56 #Declarando uma variável tipo Double

Resposta: **.byte** 'S' #Declarando uma variável tipo caractere

nome: **.space** 40 #Declarando uma variável onde tem 40 espaços

## **.text**

# *Funções*

**la Load Adress** ( significa carregar, mandar, guardar algum determinado dado ou variável na memória)

**li Load Immediate** ( significa carregar, trazer, apresentar, mostrar para o usuário alguma determinada informação guardada na memória)

# REGISTRADORES

# Registradores

- Registradores são pequenas memórias dentro da CPU
  - O processador MIPS tem 32 registradores;
  - Cada registrador suporta a capacidade de até 32 bits;
  - Os registradores são altamente velozes;
  - Podem executar instruções em até 2 **Nanosegundos**;
- **Na Arquitetura MIPS** os registradores são representados por um símbolo de \$ e depois seu nome e seu numero

# Tipos de Registradores

- **\$zero** = atribui um valor constante igual a zero
- **\$at** = Registrador temporário utilizado para assembler
- **\$v0** e **\$v1** = registradores que recebem as funções chamadas do sistema
- **\$a0** até **\$a3** = registradores para passagem de argumentos
- **\$t0** até **\$t7** = são os registradores temporários, esses não preservam os valores. Ou seja, se apagam quando o processador é desligado.
- **\$s0** até **\$s7** = São registradores que preservam os valores
- **\$t8** e **\$t9** = Também são registradores temporários
- **\$k0** e **\$k1** = São registradores reservados para o Kernel do Sistema Operacional.
- **\$gp** = registrador para ponteiro global.
- **\$sp** = registrador para apontar os elementos da pilha
- **\$fp** = registrador para apontador de frames
- **\$ra** = registrador para guardar um endereço de retorno
- **\$pc** = registrador para contar as funções do pc
- **\$hi** e **\$lo** = registradores para guardar os resultados de uma multiplicação ou divisão

# *Funções Registradores*

- Os registradores **\$v0** e **\$v1** são registradores que se comportam como registradores de controle e de dados pois recebem os valores de uma expressão e os resultados de funções
- Servem como memórias para armazenar os valores de retorno de chamadas do sistema instrução chamada em Assembly como **syscall**
- O registrador **\$v0** é uma espécie de chaveador (decodificador), alguns números recebidos por ele farão com que o processador MIPS executem algumas funções específicas

# Registradores

Serviço	Código em \$V0	argumento
imprimir inteiro	1	\$a0 = inteiro a imprimir
imprimir Float	2	\$f12 = float a imprimir
imprimir Double	3	\$f12 = double a imprimir
Imprimir string	4	\$a0 = endereço da string terminada em nulo para imprimir
Ler Inteiro	5	\$v0 contém inteiro lido
Ler Float	6	\$f0 contém inteiro lido
Ler Double	7	\$f0 contém inteiro lido
Ler String	8	\$a0 = endereço do buffer de entrada \$a1 = número máximo de caracteres a serem lidos
Finalizar programa	10	Li \$v0,10 finaliza o programa

Saida de dados

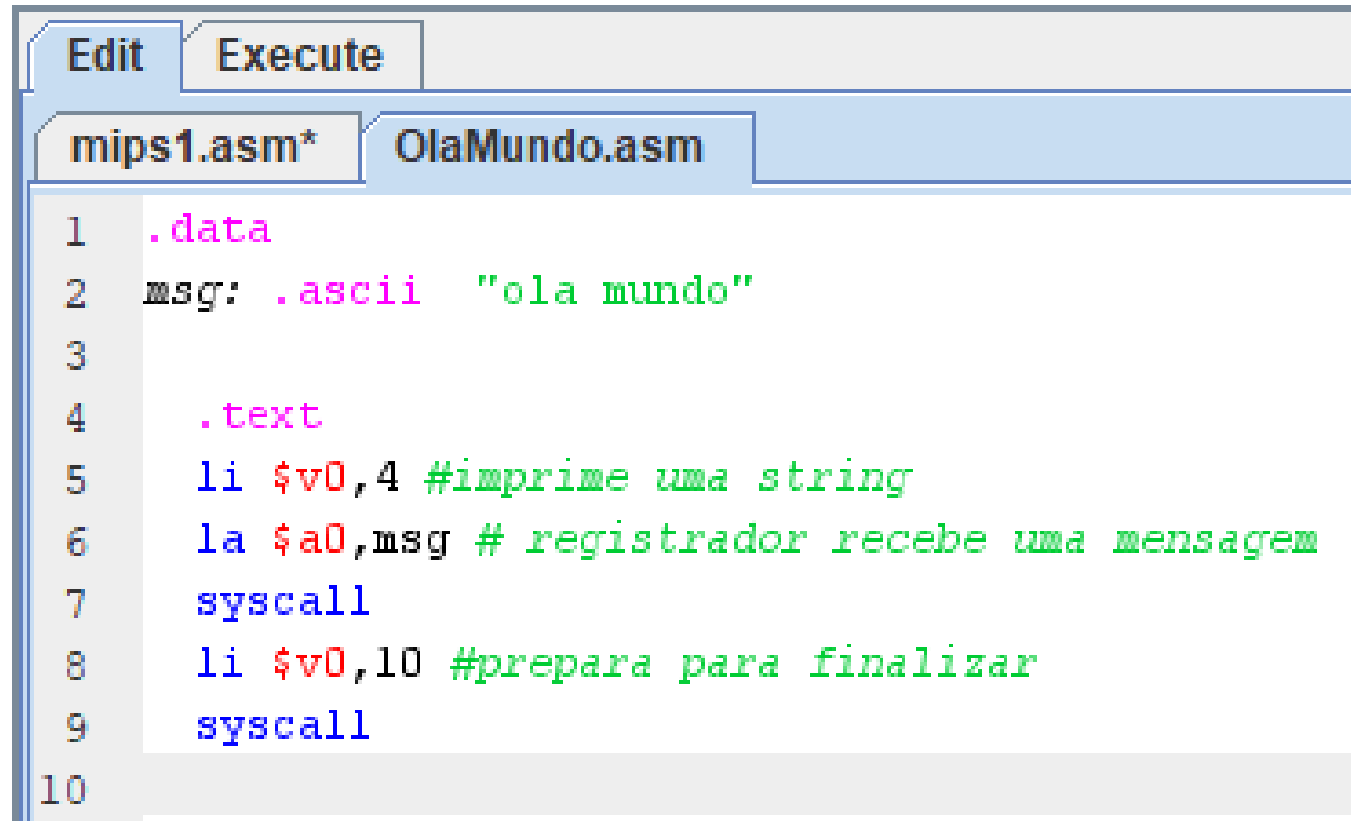


Ritual de iniciação

“HELLO WORD”



# Hello word



The image shows a screenshot of a MIPS assembly editor window. The window has a menu bar with 'Edit' and 'Execute' tabs. Below the menu bar, there are two tabs for the assembly files: 'mips1.asm\*' and 'OlaMundo.asm'. The 'OlaMundo.asm' tab is active, displaying the following assembly code:

```
1  .data
2  msg: .ascii  "ola mundo"
3
4  .text
5  li $v0,4 #imprime uma string
6  la $a0,msg # registrador recebe uma mensagem
7  syscall
8  li $v0,10 #prepara para finalizar
9  syscall
10
```

# Hello word 2 variáveis

```
1  .data
2  msg1: .ascii "Hello "    # primeira string
3  msg2: .ascii " world"    # segunda string
4
5  .text
6
7      # carrega endereço de str1 no registrador $a0
8      la $a0, msg1
9
10     # imprime str1
11     li $v0, 4
12     syscall
13
14     # carrega endereço de str2 no registrador $a0
15     la $a0, msg2
16
17     # imprime str2
18     li $v0, 4
19     syscall
20
21     # saída do programa
22     li $v0, 10
23     syscall
```

# Imprimindo um inteiro

Hello word 2\*

mips1.asm

```
1  .data
2  numero: .word 2023 #numero a ser impresso
3  .text
4      li $v0,1 #comando para imprimir um inteiro
5      lw $a0,numero #movendo um inteiro para o registrador $a0
6      syscall
7
8      li $v0,10 #finalizando o programa
9      syscall
```

10  
11  
12

# ATIVIDADES

# *Atividade 01*

- Implemente um algoritmo no MIPS que apresente o seu nome completo, seu endereço, a cidade onde você nasceu e o País de origem. Obrigatório o uso de uma variável para cada ação

.data

```
nome: .ascii "joão ricardo dos santos rosa\n"  
endereco: .ascii "R. Benedito Americo\n"  
cidade: .ascii "Curitiba\n"  
pais: .ascii "Brasil\n"
```

.text

```
li $v0,4 #comando para imprimir um inteiro  
la $a0, nome #movendo um inteiro para o registrador $ao  
syscall  
li $v0,4 #comando para imprimir um inteiro  
la $a0, endereco #movendo um inteiro para o registrador $ao  
syscall  
li $v0,4 #comando para imprimir um inteiro  
la $a0, cidade #movendo um inteiro para o registrador $ao  
syscall  
li $v0,4 #comando para imprimir um inteiro  
la $a0, pais #movendo um inteiro para o registrador $ao  
syscall  
  
li $v0,10 #finalizando o programa  
syscall
```

## *Atividade 02*

- Implemente um algoritmo no MIPS que apresente o nome de dez pessoas. Obrigatório o uso de uma variável para cada nome de pessoa



# *Atividade 03*

- Implemente um algoritmo no MIPS que apresente o nome de uma pessoa o dia e o ano de aniversário

.data

nome: .ascii "joão ricardo dos santos rosa\n"

dia: .word 31

ano: .word 1995

.text

li \$v0,4 #comando para imprimir um inteiro

la \$a0, nome #movendo um inteiro para o registrador \$ao  
syscall

li \$v0,1 #comando para imprimir um inteiro

lw \$a0, dia #movendo um inteiro para o registrador \$ao  
syscall

li \$v0,1 #comando para imprimir um inteiro

lw \$a0, ano #movendo um inteiro para o registrador \$ao  
syscall

li \$v0,10 #finalizando o programa  
syscall

# Entrada de datos

# INSTRUÇÕES EXECUTADAS PELO HARDWARE

# *Instruções executadas pelo hardware*

- AFIRMAÇÃO
  - Todo processador deve ser capaz de executar instruções aritméticas
- Exemplo:
  - **add** a, b, c
- Essa instrução representa a soma de B com C, armazenando o resultado em A.

# *Instruções executadas pelo hardware*

- Solução:
  - Conjunto de instruções
  - Cada instrução aritmética executa apenas uma operação
  - Precisa sempre estar referenciada a três constantes
- Exemplo:
  - Somar A,B,C,D,E
  - Armazenar o resultado em A

# *Instruções executadas pelo hardware*

- Solução: somar A,B,C,D,E
  - **add a, b, c**
  - **add a, a, d**
  - **add a, a, e**
- Na primeira linha soma-se B com C e o resultado é armazenado em A:
- Na segunda linha soma-se o resultado que foi armazenado em A com o valor de D.
- Na terceira linha soma-se o resultado armazenado em A com o valor de E.

# *Instruções executadas pelo hardware*

- Para realizar a soma de quatro valores, foi necessário três instruções
- Em MIPS cada linha da linguagem pode conter apenas uma instrução
- ADIÇÃO
  - Sempre possuirá três operandos
  - Dois números a serem somados (origem)
  - Um local de armazenamento (destino)



# *Instruções executadas pelo hardware*

- PERGUNTA:
  - Porque três operandos? Porque não dois operandos? Ou porque não instruções com tamanhos de operandos variados ?
- RESPOSTA
  - Objetivo: manter o Hardware tão simples quanto possível
  - Realizar operações aritméticas com um número variável de operandos é mais complexo
  - Realizar operações aritméticas com um numero fixo de operandos é mais simples

# *Compilação de comandos C para MIPS*

- EXEMPLO
  - Dado o seguinte código em linguagem de alto nível (linguagem C). Traduza-o para linguagem de máquina MIPS
    - $a = b + c;$
    - $d = a - e;$
    - $b = a * e;$
    - $f = a / b;$
- SOLUÇÃO
  - `add a, b, c`
  - `sub d, a, e`
  - `mpy b, a, e`
  - `div f, a, e`

# *Compilação de comandos C para MIPS*

- EXEMPLO
  - Dado o seguinte código em linguagem de alto nível (linguagem C). Traduza-o para linguagem de máquina MIPS
    - $y = x * w$
    - $j = a - b$
    - $p = g + f$
    - $r = z / q$
- SOLUÇÃO
  - ????

# Instruções inteiros no Assembly MIPS



# *Soma de inteiros MIPS*

- `add $t0, $t1, $t2`    `#t0 = t1 + t2`
- `addi $t0, $t1, 15`    `#to = t1 + 15`

# *Soma de inteiros instrução add MIPS*

```
1  .text
2
3      li $t0, 75    #primeiro numero da soma
4      li $t1, 25    # segundo numero da soma
5      add $s0, $t0, $t1
```

# Soma de inteiros MIPS

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	75
\$t1	9	25
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	100
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194316
hi		0
lo		0

# *Realize as seguintes adições em MIPS*

- A)  $569 + 789$
- B)  $334 + 45$
- C )  $3 + 152$
- E)  $86 + 14$
- F)  $14 + 23$
- G)  $132 + 18$



# *Subtração de inteiros*

- `sub $t2, $t0, $t1`     $\#t2 = t0 - t1$
- `subi $t2, $t0, 15`     $\#t2 = t1 - 15$

# *Soma de inteiros instrução add MIPS*

```
.text
```

```
li $t0, 500    #primeiro numero da subtração  
li $t1, 200    # segundo numero da subtração  
sub $t2, $t0, $t1
```

# Soma de inteiros MIPS

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	500
\$t1	9	200
\$t2	10	300
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194316
hi		0
lo		0

stack pointer

# *Realize as seguintes subtrações em MIPS*

- **A) 789 - 186**
- **B) 334 - 45**
- **C) 3 - 152**
- **E) 2023 - 1350**
- **F) 50 - 23**
- **G) 132 - 18**