

Programa de Pós-Graduação em Computação

MATE48 – Arquitetura de Computadores

MARS **Simulador MIPS**

Prof. Marcos E. Barreto

marcoseb@dcc.ufba.br

<http://www.dcc.ufba.br/~marcoseb>

2015.2

MIPS

- **Conjunto de registradores**

Name	Number	Use
\$zero	0	The constant value 0
\$at	1	Assembler temporary
\$v0-\$v1	2-3	Values for function results and expression evaluation
\$a0-\$a3	4-7	Arguments
\$t0-\$t7	8-15	Temporaries
\$s0-\$s7	16-23	Saved temporaries
\$t8-\$t9	24-25	Temporaries
\$k0-\$k1	26-27	Reserved for OS kernel
\$gp	28	Global pointer
\$sp	29	Stack pointer
\$fp	30	Frame pointer
\$ra	31	Return address

MIPS

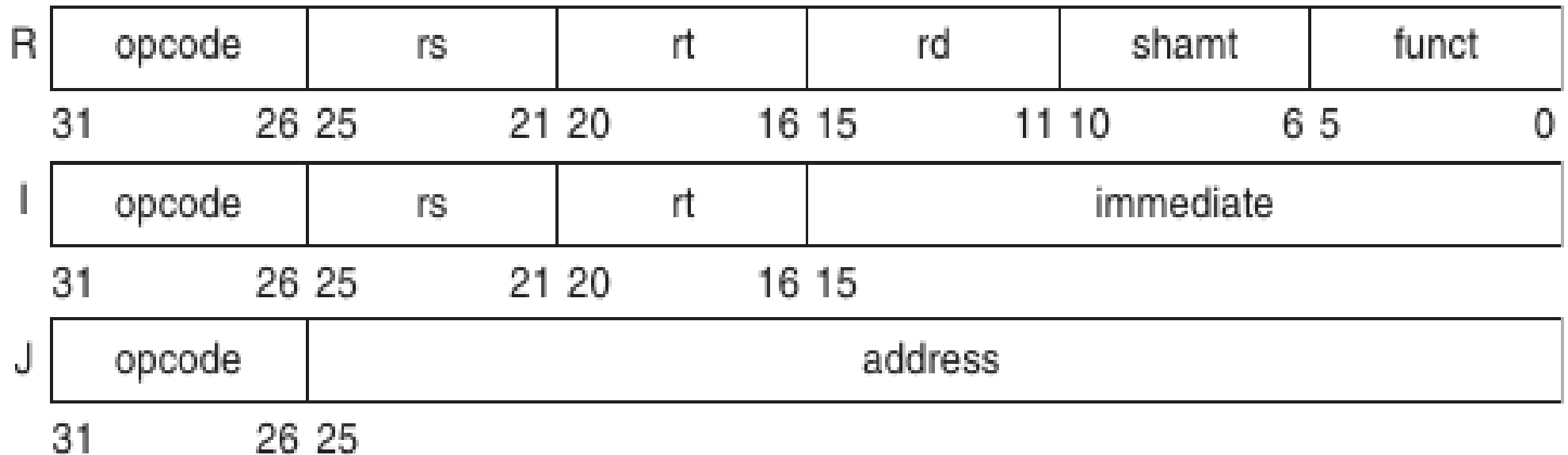
- **Conjunto de instruções**

- Aritméticas (*add, sub, mul, div* e variações)
- Lógicas (*and, or, xor, nor, slt*)
- Transferência de dados (*lw, sw* e variações, *mflo, mfhi*)
- Deslocamento (*sll, srl, sra* e variações)
- Desvio condicional (*beq, bne*)
- Desvio incondicional (*j, jr, jal*)
- Lista completa em:
 - http://www.cs.cornell.edu/courses/cs3410/2008fa/MIPS_Vol2.pdf
 - <https://www.lri.fr/~de/MIPS.pdf>
 - http://en.wikipedia.org/wiki/MIPS_instruction_set

MIPS

- **Formato das instruções**

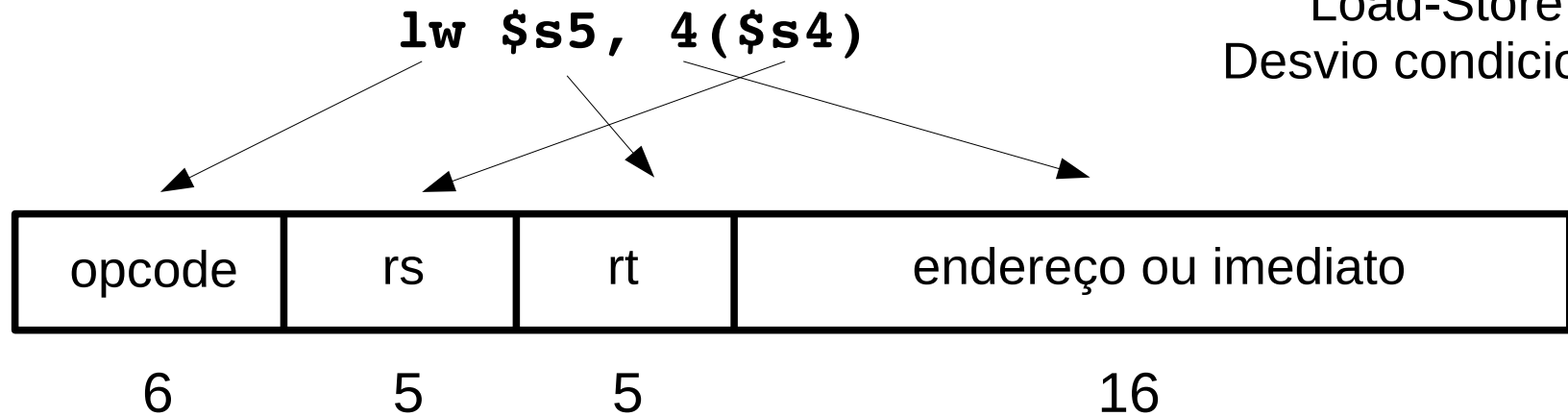
- Todas com 32 bits
- 3 tipos:
 - R: instruções aritméticas e lógicas (operandos em registradores)
 - I: instruções de acesso a memória (load/store) e desvios condicionais.
 - J: instruções para desvio



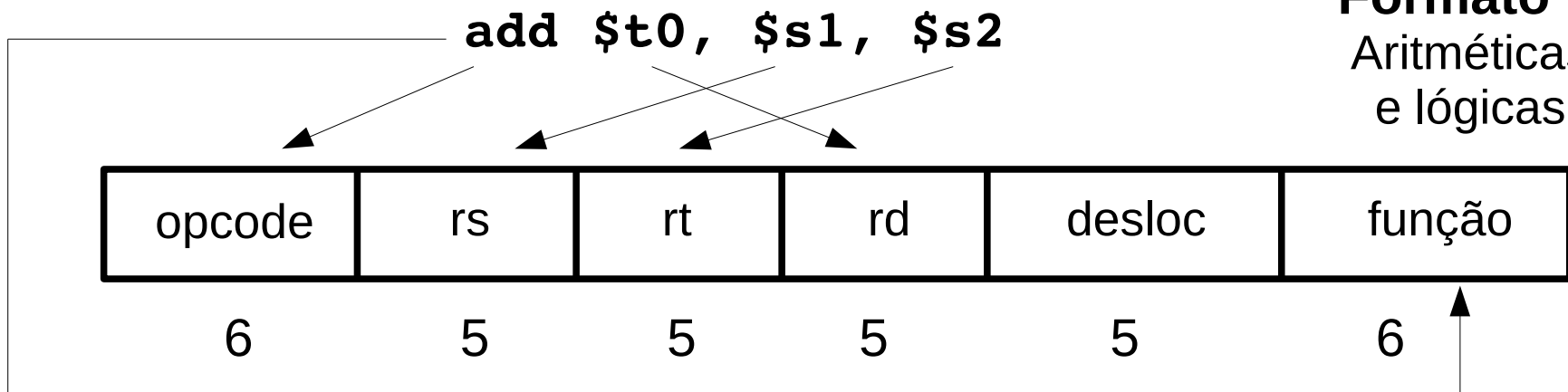
MIPS

- Formato das instruções

Formato I
Load-Store
Desvio condicional



Formato R
Aritméticas
e lógicas



MIPS

- **Formato das instruções**

```

Loop:    sll  $t1, $s3, 2      # $t1 = i * 4 (alinhamento)
         add  $t1, $t1, $s6    # $t1 = endereço de vetor[i]
         lw   $t0, 0($t1)      # $t0 = valor de vetor[i]
         bne  $t0, $s5, Exit   # termina laço se vetor[i]!=k
         addi $s3, $s3, 1      # i = i + 1
         j    Loop            # retorna para início do laço

```

Exit: ...

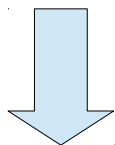
8000	0	0	19	9	4	0	R R I I I J
8004	0	9	22	9	0	32	
8008	35	9	8	0			
8012	5	8	21	2			
8016	8	19	19	1			
8020	2	2000					
8024						

MIPS



- ✓ Antes de calcular a expressão, deve-se associar registradores às variáveis,

```
la $s0, f // load address
la $s1, g
la $s2, h
la $s3, i
la $s4, j
```



- ✓ carregar o valor dos operandos nos registradores,

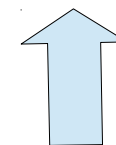
```
lw $s1, 0($s1) // load word
lw $s2, 0($s2)
lw $s3, 0($s3)
lw $s4, 0($s4)
```



$$f = (g + h) - (i + j)$$

- ✓ gravar o resultado.

```
sw $t2, 0($s0)
// store word
```



- ✓ calcular a expressão e

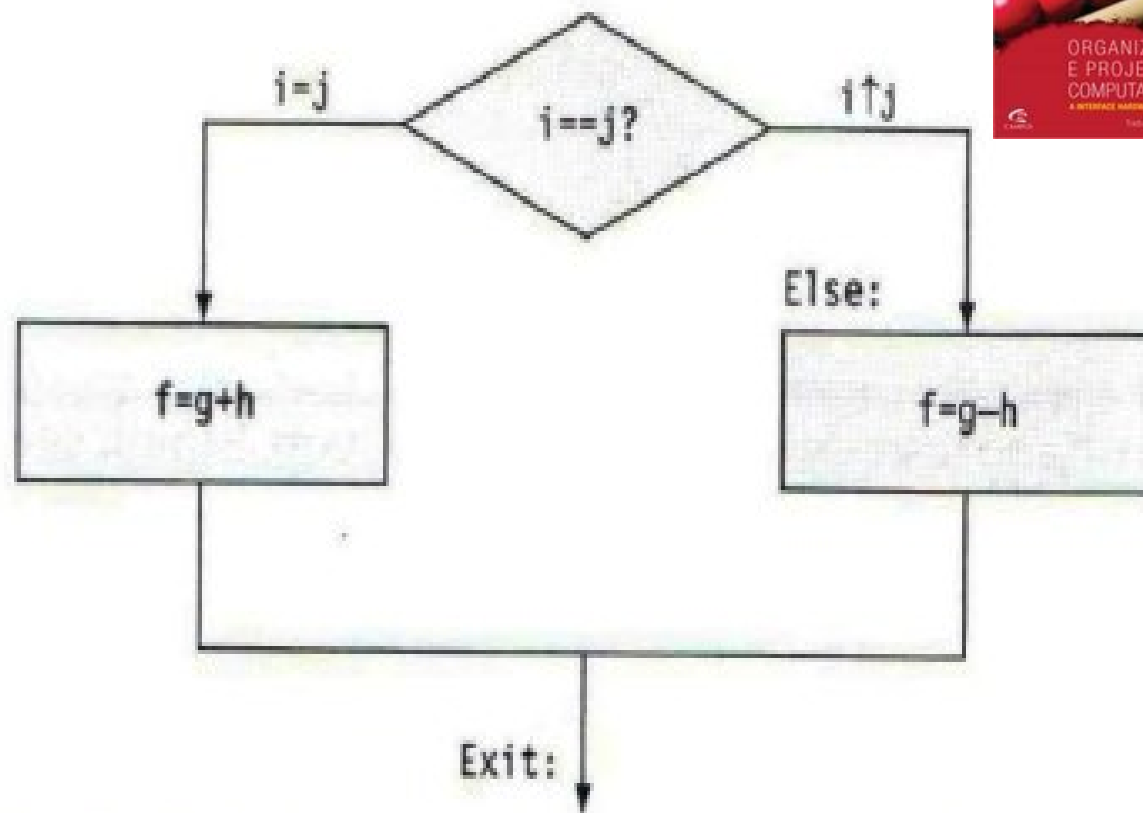
```
add $t0, $s1, $s2
add $t1, $s3, $s4
sub $t2, $t0, $t1
```

MIPS

```
if (i == j)
    f = g + h
else f = g - h
```

✓ Supondo o mapeamento

```
la $s0, f
la $s1, g
la $s2, h
la $s3, i
la $s4, j
```



✓ Código para o comando IF

```
    bne $s3, $s4, ELSE    # vai para ELSE se i!=j
    add $s0, $s1, $s2     # f = g+h, se i==j
    j    EXIT             # vai para Exit (pula Else)
ELSE:  sub $s0, $s1, $s2  # f = g-h, se i!=j
EXIT:  ...

// bne – branch if not equal
// j   – jump
```



MIPS



```
while (vetor[i] == k)
    i += 1;
```

✓ Supondo o mapeamento

```
la $s3, i
la $s5, k
la $s6, vetor
```

✓ Código para o comando WHILE


```
Loop:    sll $t1, $s3, 2      # $t1 = i * 4 (alinhamento)
        add $t1, $t1, $s6    # $t1 = endereço de vetor[i]
        lw  $t0, 0($t1)      # $t0 = valor de vetor[i]
        bne $t0, $s5, Exit   # termina laço se vetor[i] != k
        addi $s3, $s3, 1     # i = i + 1
        j   Loop            # retorna para início do laço

Exit:    ...
```

```
// sll – shift left logical immediate
// addi – add immediate
```

MARS

- Simulador da arquitetura MIPS desenvolvido na Universidade de Missouri.
- Simulador multi-plataforma, desenvolvido em Java.
- Download e documentação disponível em
<http://courses.missouristate.edu/KenVollmar/MARS/index.htm>
- Versão atual: 4.5
- Para executar:
 - Linux: `java -jar Mars4_5.jar &`



Run speed at max (no interaction)

Para compilar

Edit Execute

mips1.asm Exemplo1.asm

```
1 # Implementa C = A + B
2
3 .data
4 A:      .word 40
5 B:      .word 0xffab3100
6 C:      .word 0
7
8 .text
9 .globl main
10
11 main:   la      $t0,A      # load address - carrega endereço de A em $t0
12         lw      $t0,0($t0) # load word - lê valor de A para $t0
13         la      $t1,B      # carrega endereço de B em $t1
14         lw      $t1,0($t1) # lê valor de B para $t1
15         addu    $t0,$t0,$t1 # $t0 recebe A + B
16         la      $t2,C      # carrega endereço de C em $t2
17         sw      $t0,0($t2) # C recebe A+B
18
19 # Finalização do programa.
20         li      $v0,10      # chamada de sistema para encerrar o programa (código 10 em $v0)
21         syscall
22
23
24
25
26
27
```

registradores

Área de edição

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	0
\$t1	9	0
\$t2	10	0
\$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		4194304
hi		0
lo		0

Mars Messages Run I/O

Reset: reset completed.

Clear

Console I/O

File Edit Settings Tools Help

**Programa em execução**

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	4194304	0x3c011001	lui \$1,4097	11: main: la \$t0,A # load address - carrega endereço de A em \$t0
<input type="checkbox"/>	4194308	0x34280000	ori \$8,\$1,0	
<input type="checkbox"/>	4194312	0x8d080000	lw \$8,0(\$8)	12: lw \$t0,0(\$t0) # load word - lê valor de A para \$t0
<input type="checkbox"/>	4194316	0x3c011001	lui \$1,4097	13: la \$t1,B # carrega endereço de B em \$t1
<input type="checkbox"/>	4194320	0x34290004	ori \$9,\$1,4	
<input type="checkbox"/>	4194324	0x8d290000	lw \$9,0(\$9)	14: lw \$t1,0(\$t1) # lê valor de B para \$t1
<input type="checkbox"/>	4194328	0x01094021	addu \$8,\$8,\$9	15: addu \$t0,\$t0,\$t1 # \$t0 recebe A + B
<input type="checkbox"/>	4194332	0x3c011001	lui \$1,4097	16: la \$t2,C # carrega endereço de C em \$t2
<input type="checkbox"/>	4194336	0x342a0008	ori \$10,\$1,8	
<input type="checkbox"/>	4194340	0xad480000	sw \$8,0(\$10)	17: sw \$t0,0(\$t2) # C recebe A+B
<input type="checkbox"/>	4194344	0x2402000a	addiu \$2,\$0,10	20: li \$v0,10 # chamada de sistema para encerrar o programa (código 10 em \$v0)
<input type="checkbox"/>	4194348	0x0000000c	syscall	21: syscall

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	40	-5558016	0	0	0	0	0	0
268501024	0	0	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0

Memória

Mars Messages Run I/O

Assemble: assembling /home/marcos/UFBA/Disiplinas/MATA48/Simuladores/MIPS/Mars/Exemplos/Exemplo1.asm

Assemble: operation completed successfully.

Clear

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	0
\$t1	9	0
\$t2	10	0
\$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		4194304
hi		0
lo		0

MARS – sintaxe do assembler

- Comentários iniciam com #
- Identificadores são sequências de caracteres alfanuméricos
- Rótulos são colocados no início da linha, seguidos de :
- Números são representados em base decimal (padrão), mas são interpretados em hexadecimal se precedidos de 0x.
- Strings são delimitados por “ ”
- Exemplos:
 - A: .word 40
 - B: .word 0xffab3100
 - Nome: .ascii “Marcos”

MARS – diretivas de tradução

- Não geram instruções de máquinas. Apenas informam ao tradutor como ele deve traduzir o programa
- *.asciiZ*: strings terminadas em caractere nulo.
- *.ascii*: strings sem terminação de caractere nulo.
- *.word*: inteiros de 32 bits
- *.data <seg>*: indica os dados a serem armazenados na memória. Se <seg> for especificado, inicia o armazenamento a partir desta posição.
- *.text*: indica o segmento de código/instruções.
- *.globl <rótulo>*: define <rótulo> como global, podendo ser acesso de outros programas.

MARS – chamadas ao sistema

- Comando *syscall*

Serviço	Código em \$v0	Argumentos	Resultados
Print integer	1	\$a0 = integer to print	
Print float	2	\$f12 = float to print	
Print double	3	\$f12 = double to print	
Print string	4	\$a0 = address of null-terminated string to print	
Read integer	5		\$v0 contains integer read
Read float	6		\$f0 contains float read
Read double	7		\$f0 contains double read
Read string	8	\$a0 = address of input buffer \$a1 = maximum number of characters to read	
Exit (terminate execution)	10		
Print character	11	\$a0 = character to print	
Read character	12		\$v0 contains character read

MARS – chamadas ao sistema

- Comando *syscall*

Serviço	Código em \$v0	Argumentos	Resultados
Open file	13	\$a0 = address of null-terminated string containing filename \$a1 = flags \$a2 = mode	\$v0 contains file descriptor (negative if error).
Read from file	14	\$a0 = file descriptor \$a1 = address of input buffer \$a2 = maximum number of characters to read	\$v0 contains number of characters read (0 if end-of-file, negative if error).
Write to file	15	\$a0 = file descriptor \$a1 = address of output buffer \$a2 = number of characters to write	\$v0 contains number of characters written (negative if error).
Close file	16	\$a0 = file descriptor	

Lista completa em <http://courses.missouristate.edu/KenVollmar/MARS/Help/SyscallHelp.html>

MARS

- Exemplos 1 a 9 na página da disciplina no NovoMoodle.

<http://www.novomoodle.ufba.br/course/view.php?id=1328>