



Universidade Federal de Pelotas
Centro de Desenvolvimento Tecnológico
Bacharelado em Ciência da Computação
Engenharia de Computação

Arquitetura e Organização de Computadores I

Prática

Aula 6

Revisão

Prof. Guilherme Corrêa
gcorrea@inf.ufpel.edu.br

Revisão

► MIPS: Registradores

Registrador	Nome	Uso (convenção)
\$0	\$zero	Zero
\$1	\$at	<i>Assembler Temporary</i>
\$2, \$3	\$v0, \$v1	Valor de retorno de subrotina
\$4 – \$7	\$a0 – \$a3	Argumentos de subrotina
\$8 – \$15	\$t0 – \$t7	Temporários (locais à função)
\$16 – \$23	\$s0 – \$s7	Salvos (não alterados na função)
\$24, \$25	\$t8, \$t9	Temporários
\$26, \$27	\$k0, \$k1	Kernel (reservado para SO)
\$28	\$gp	<i>Global Pointer</i>
\$29	\$sp	<i>Stack Pointer</i>
\$30	\$fp	<i>Frame Pointer</i>
\$31	\$ra	Endereço de Retorno

Revisão

► Operações Lógicas (and, or, xor, nor)

Tipo R

31	26	25	21	20	16	15	11	10	6	5	0				
opcode						rs		rt		rd		shamt		funct	
6 bits						5 bits		5 bits		5 bits		5 bits		6 bits	

and \$t1, \$zero, \$t3

0H	0H	BH	9H	0H	24H
000000	00000	01011	01001	00000	100100

or \$t0, \$t1, \$t2

0H	9H	AH	8H	0H	25H
000000	01001	01010	01000	00000	100101

xor \$t1, \$t2, \$t3

0H	AH	BH	9H	0H	26H
000000	01010	01011	01001	00000	100110

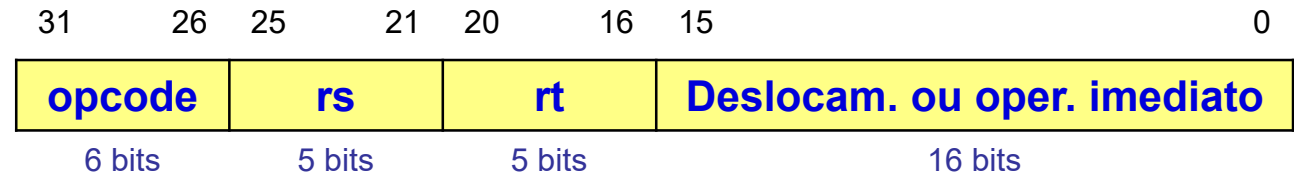
nor \$t0, \$t0, \$t3

0H	8H	BH	8H	0H	27H
000000	01011	01011	01001	00000	100111

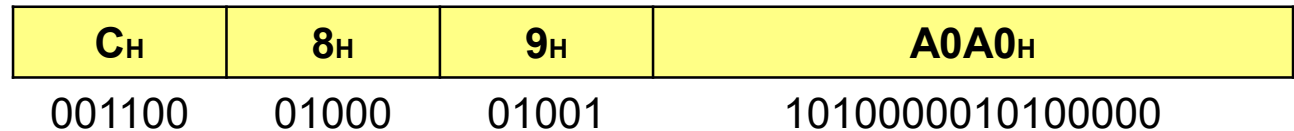
Revisão

► Operações Lógicas (andi, ori, xori)

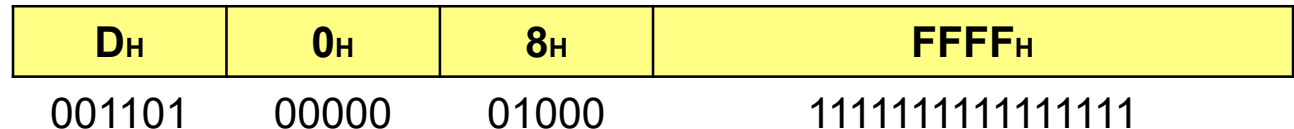
Tipo I



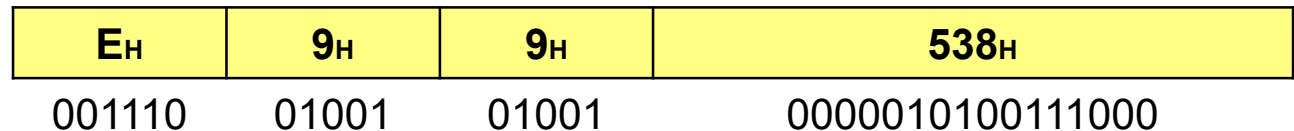
andi \$t1, \$t0, 0xA0A0



ori \$t0, \$zero, 0xFFFF



xori \$t1, \$t1, 0x538



Revisão

► Operações Lógicas (sll, slr)

Tipo R

31	26	25	21	20	16	15	11	10	6	5	0
opcode			rs		rt		rd		shamt		funct
6 bits			5 bits		5 bits		5 bits		5 bits		6 bits

sll \$t0, \$t1, 12

0 _H	0 _H	9 _H	8 _H	C _H	0 _H
000000	00000	01001	01000	01100	000000

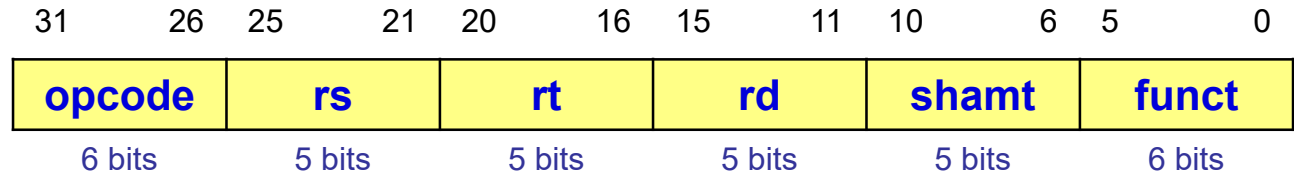
srl \$t5, \$t5, 28

0 _H	0 _H	D _H	D _H	1C _H	2 _H
000000	00000	01101	01101	11100	000010

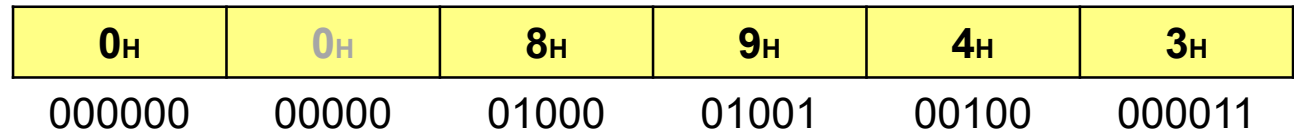
Revisão

► Operações Aritméticas (sra)

Tipo R



sra \$t1, \$t0, 4



Revisão

► Operações Aritméticas (add, sub, addu, subu)

Tipo R

31	26	25	21	20	16	15	11	10	6	5	0				
opcode						rs		rt		rd		shamt		funct	
6 bits						5 bits		5 bits		5 bits		5 bits		6 bits	

add \$t2, \$t0, \$t1

0H	8H	9H	AH	0H	20H
000000	01000	01001	01010	00000	100000

sub \$t2, \$t0, \$t1

0H	8H	9H	AH	0H	22H
000000	01000	01001	01010	00000	100010

addu \$t2, \$t0, \$t1

0H	8H	9H	AH	0H	21H
000000	01000	01001	01010	00000	100001

subu \$t2, \$t0, \$t1

0H	8H	9H	AH	0H	23H
000000	01000	01001	01010	00000	100011

Revisão

► Operações Aritméticas (mult, multu, mfhi, mflo)

Tipo R

31	26	25	21	20	16	15	11	10	6	5	0
opcode			rs		rt		rd		shamt		func
6 bits			5 bits		5 bits		5 bits		5 bits		6 bits

mult \$t0, \$t1

0H	8H	9H	0H	0H	18H
000000	01000	01001	00000	00000	011000

multu \$t0, \$t1

0H	8H	9H	0H	0H	19H
000000	01000	01001	00000	00000	011001

mfhi \$t2

0H	0H	0H	AH	0H	10H
000000	00000	00000	01010	00000	010000

mflo \$t3

0H	0H	0H	BH	0H	12H
000000	00000	00000	01011	00000	010010

Revisão

► Operações Aritméticas (div, divu, mfhi, mflo)

Tipo R

31	26	25	21	20	16	15	11	10	6	5	0
opcode			rs		rt		rd		shamt		funct
6 bits			5 bits		5 bits		5 bits		5 bits		6 bits

div \$t0, \$t1

0H	8H	9H	0H	0H	1AH
000000	01000	01001	00000	00000	011010

divu \$t0, \$t1

0H	8H	9H	0H	0H	1BH
000000	01000	01001	00000	00000	011011

mfhi \$t2

0H	0H	0H	AH	0H	10H
000000	00000	00000	01010	00000	010000

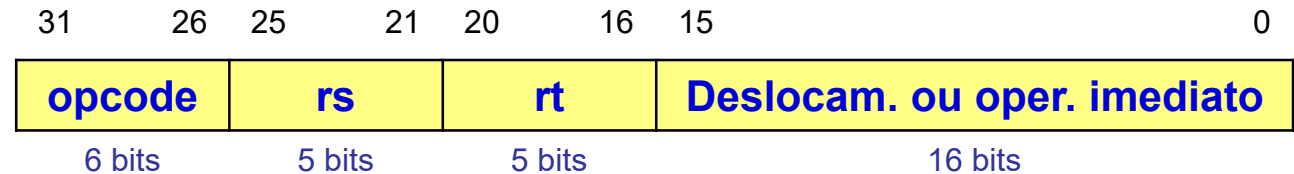
mflo \$t3

0H	0H	0H	BH	0H	12H
000000	00000	00000	01011	00000	010010

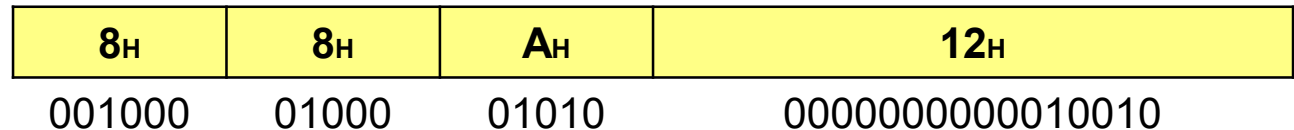
Revisão

► Operações Aritméticas (addi, addiu)

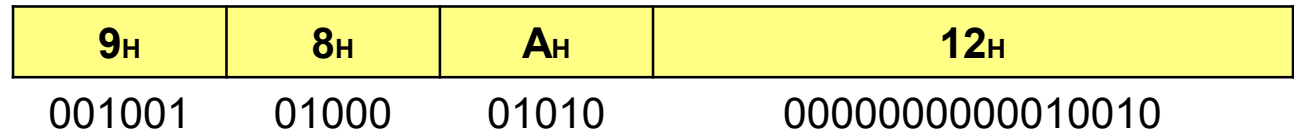
Tipo I



addi \$t2, \$t0, 0x12



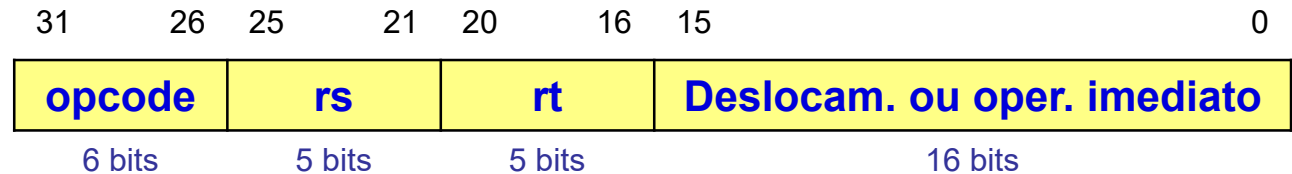
addiu \$t2, \$t0, 0x12



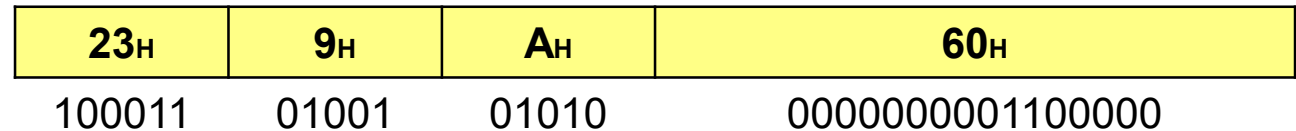
Revisão

► Acesso à Memória (lw, sw)

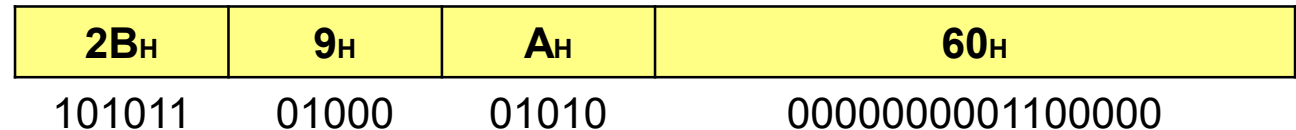
Tipo I



lw \$t2, 0x60(\$t1)



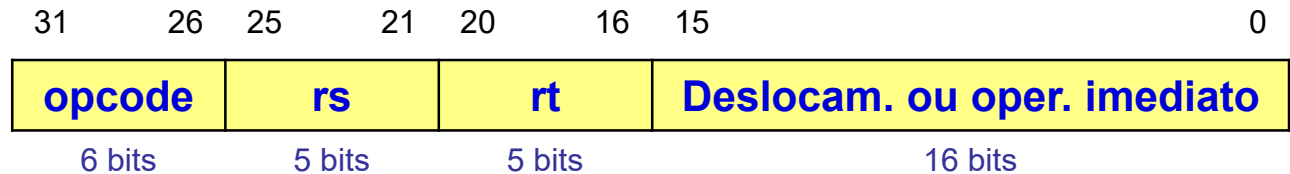
sw \$t2, 0x60(\$t1)



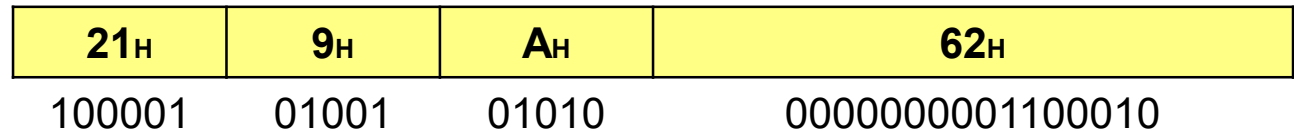
Revisão

► Acesso à Memória (lh, sh)

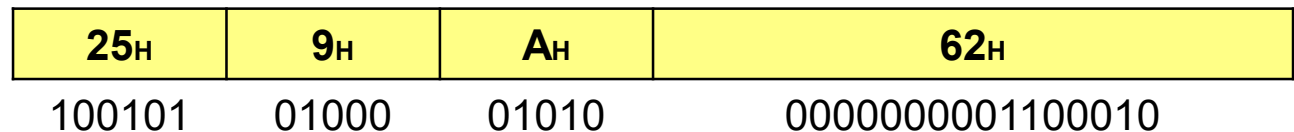
Tipo I



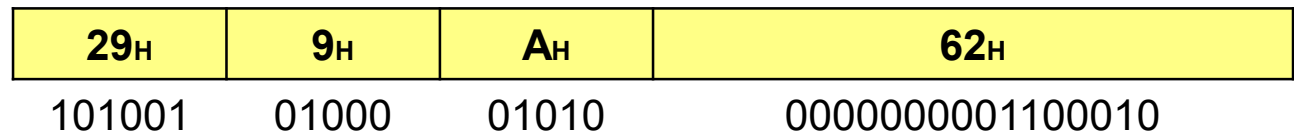
lh \$t2, 0x62(\$t1)



lhu \$t2, 0x62(\$t1)



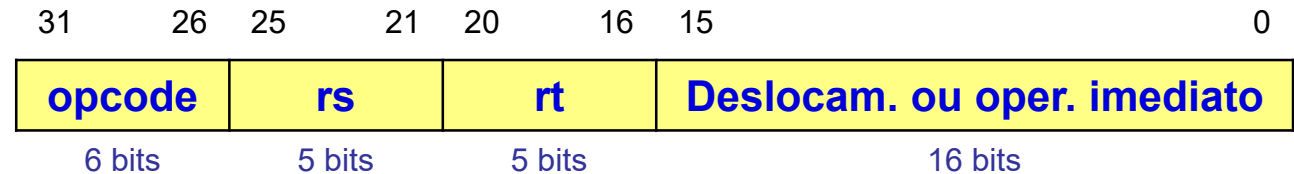
sh \$t2, 0x62(\$t1)



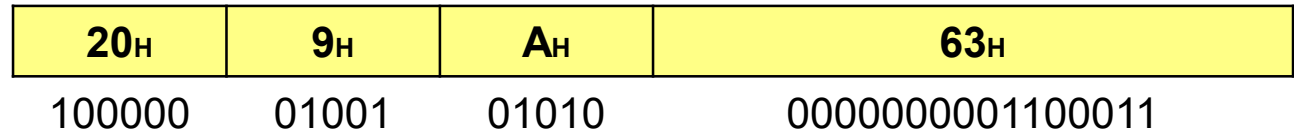
Revisão

► Acesso à Memória (lb, sb)

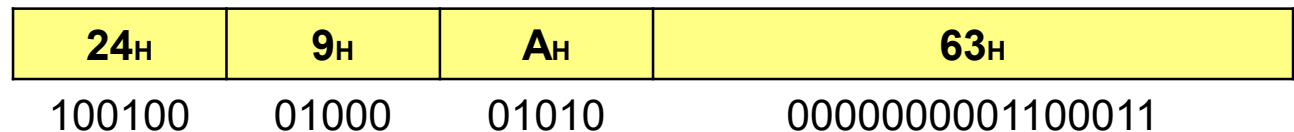
Tipo I



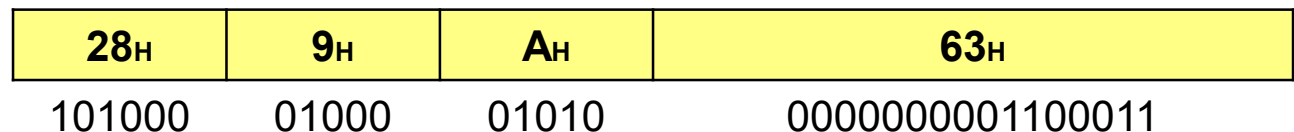
lb \$t2, 0x63(\$t1)



lbu \$t2, 0x62(\$t1)



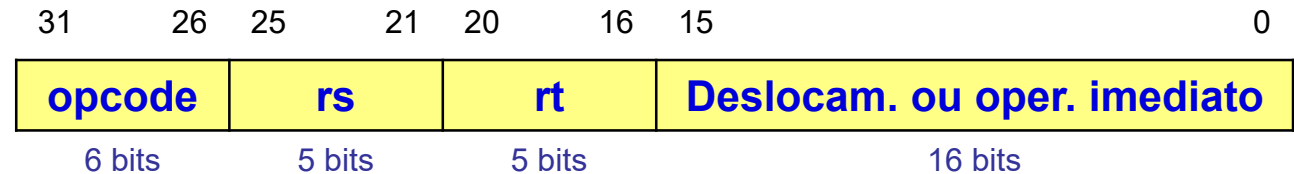
sb \$t2, 0x63(\$t1)



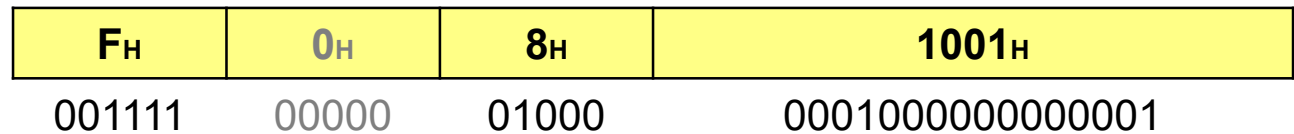
Revisão

► Acesso à Memória (lui)

Tipo I



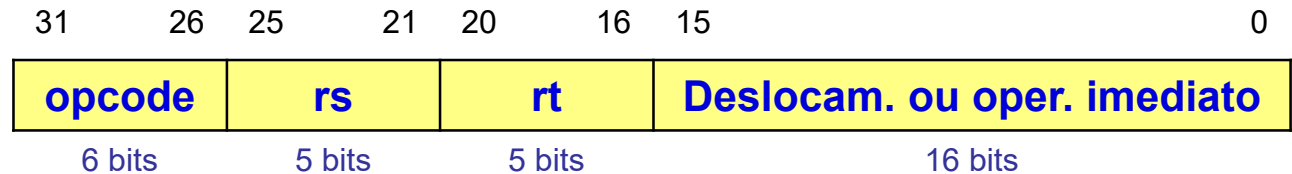
lui \$t0, 0x1001



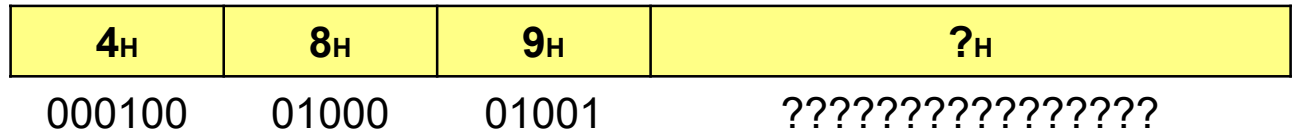
Revisão

► Desvios Condicionais (beq, bne)

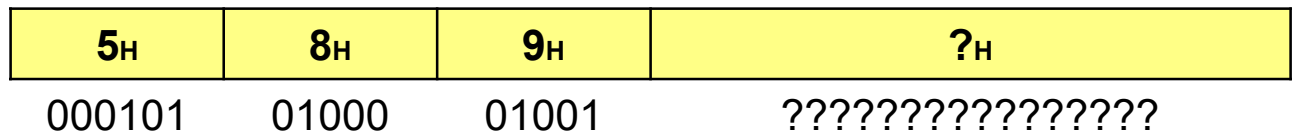
Tipo I



beq \$t0, \$t1, *Label*



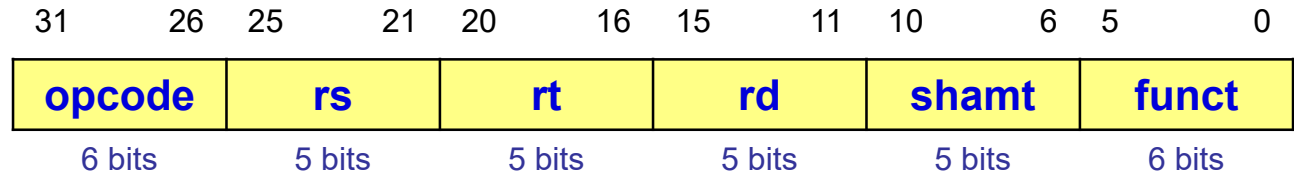
bne \$t0, \$t1, *Label*



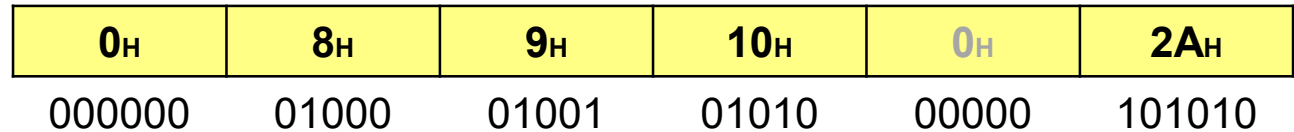
Revisão

► Desvios Condicionais (slt, sltu)

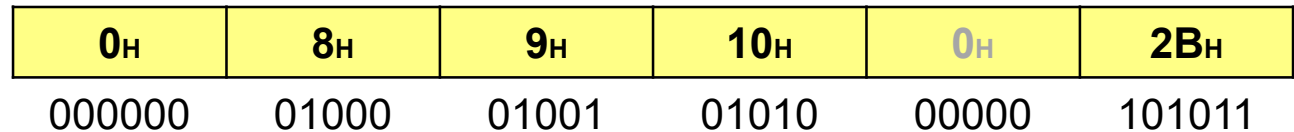
Tipo R



slt \$t0, \$t1, \$t2



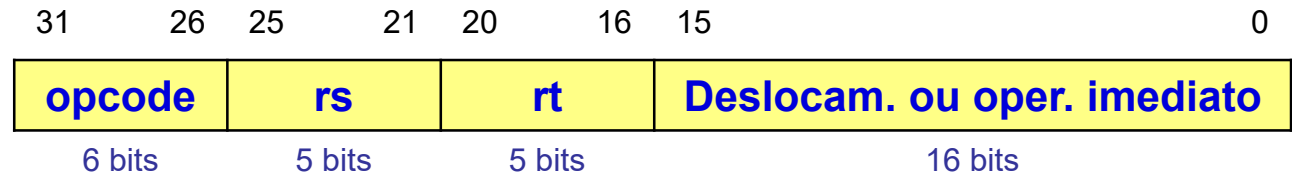
sltu \$t0, \$t1, \$t2



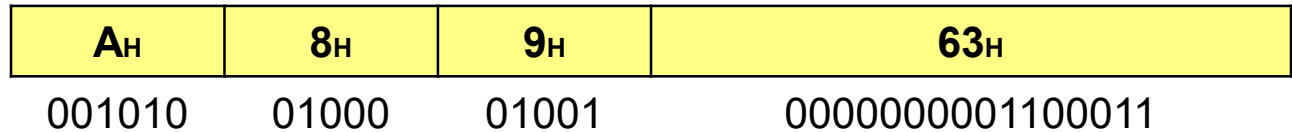
Revisão

► Desvios Condicionais (slti, sltiu)

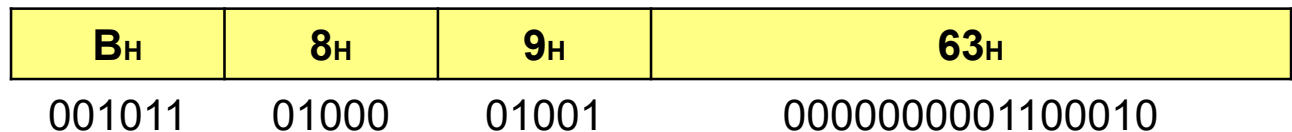
Tipo I



slti \$t0, \$t1, 0x63




sltiu \$t0, \$t1, 0x63



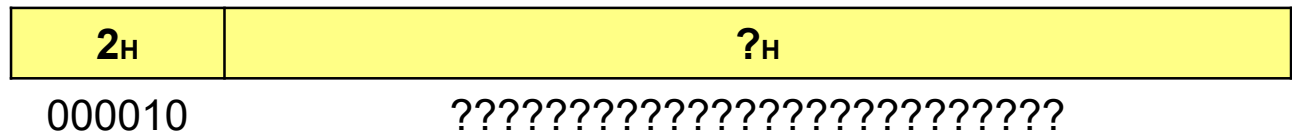
Revisão

► Desvios Incondicionais (j)

Tipo J 



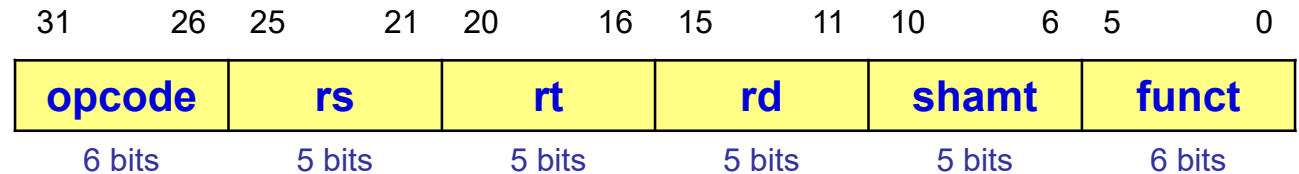
j *Label*



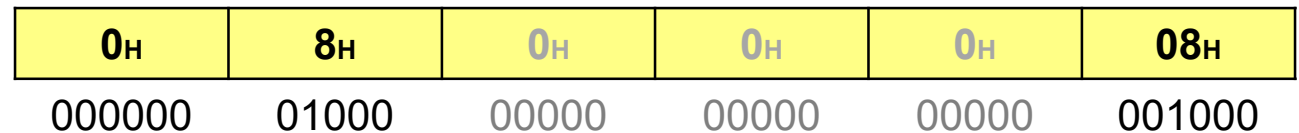
Revisão

► Desvios Incondicionais (jr)

Tipo R



jr \$t0



Revisão

► Definir seção de dados no MARS

- Utilizamos a diretiva *.data*
- O início da área de dados é a posição **0x10010000**
- **Atenção:** *.data* **NÃO É** uma instrução do MIPS; é apenas uma diretiva do montador!

<i>.data</i>	<i># inicia a seção de dados</i>
<i>.word 1</i>	<i># escreve 1 em 0x10010000</i>
<i>.word -3</i>	<i># escreve -3 em 0x10010004</i>
<i>.word 15</i>	<i># escreve 15 em 0x10010008</i>

Revisão

► Definir seção de dados no MARS

- A partir de agora, utilizaremos:
 - *.data* antes da seção de dados
 - *.text* antes da seção de instruções

<i>.data</i>	<i># inicia a seção de dados</i>
<i>.word 1</i>	<i># escreve 1 em 0x10010000</i>
<i>.word -3</i>	<i># escreve -3 em 0x10010004</i>
<i>.word 15</i>	<i># escreve 15 em 0x10010008</i>
<i>.text</i>	<i># inicia a seção de instruções</i>
<i>lui \$t0, 0x1001</i>	<i># carrega o reg \$t0 com 0x10010000</i>
<i>lw \$t1, 8(\$t0)</i>	<i># carrega \$t1 com a word em 0x10010008</i>
<i>add \$s2, \$t1, \$t1</i>	<i># soma \$t1 com \$t1 e coloca o resultado em \$s2</i>
<i>sw \$s2, 12(\$t0)</i>	<i># escreve a word \$s2 em 0x1001000C</i>

Revisão

► Definir seção de dados no MARS

- Outras diretivas para *.data*
 - .byte* reserva um *byte*
 - .half* reserva uma *halfword*
 - .space x* reserva *x bytes*