

# Sistemas Processadores e Periféricos

## Aula 2 - Revisão

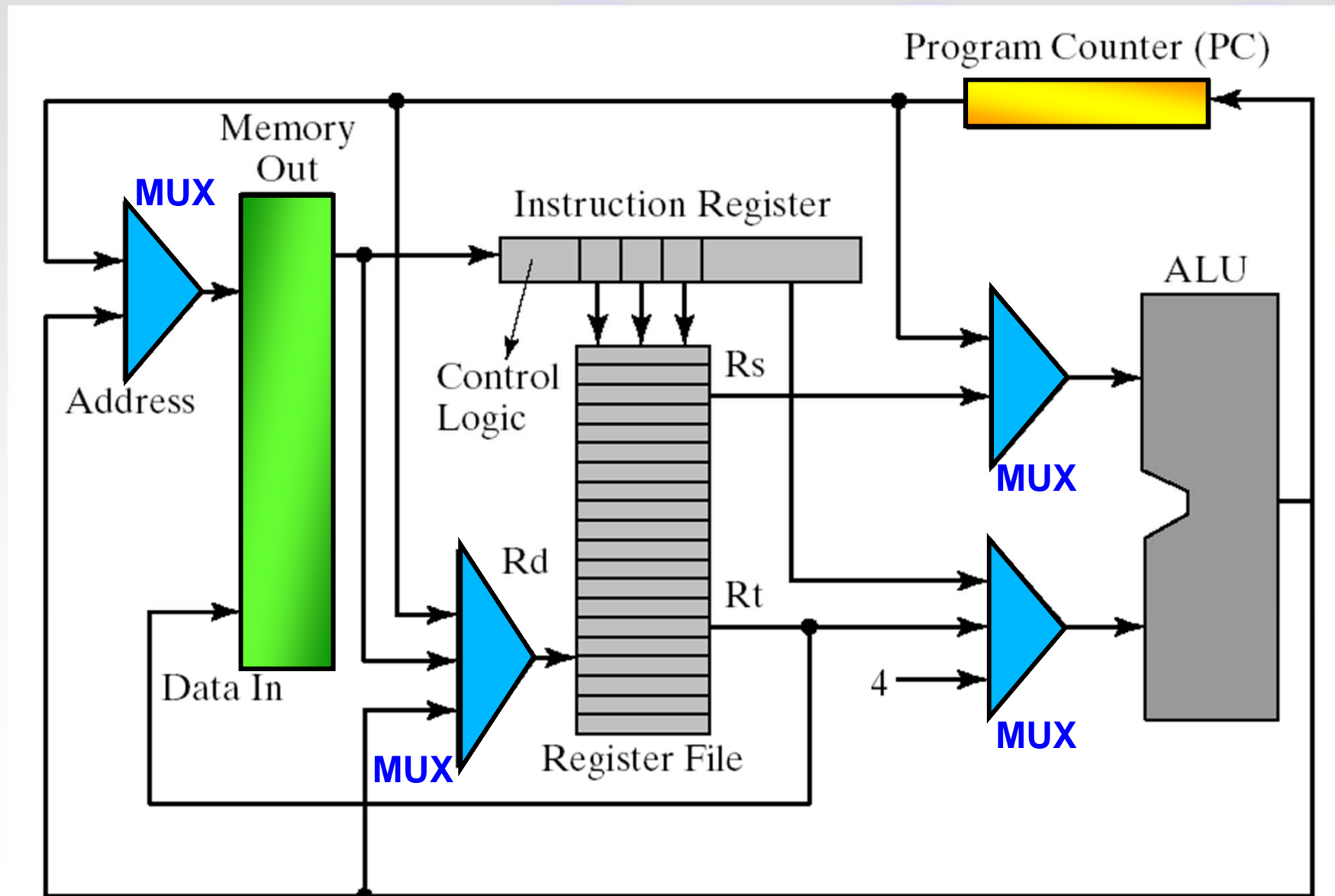
Prof. Frank Sill Torres  
DELT – Escola de Engenharia  
UFMG

Adaptado a partir dos Slides de Organização de Computadores 2006/02 do professor

Leandro Galvão DCC/UFAM - [galvao@dcc.ufam.edu.br](mailto:galvao@dcc.ufam.edu.br) pelo Prof. Ricardo de Oliveira Duarte

# Arquitetura MIPS

## :: Diagrama simplificado



# Convenção de uso dos registradores

Registrador	Número	Uso
\$zero	0	Valor constante igual a zero
\$v0-\$v1	2-3	Retorno de funções
\$a0-\$a3	4-7	Argumentos de funcoes
\$t0-\$t7	8-15	Temporários, não precisam ser salvos
\$s0-\$s7	16-23	Salvos por uma função chamada
\$t8-\$t9	24-25	Mais temporários
\$gp	28	Apontador global
\$sp	29	Apontador de pilha
\$fp	30	Apontador de quadro
\$ra	31	Endereço de retorno

# Princípios de projeto MIPS

1. **Simplicidade favorece regularidade**
2. **Menor significa mais rápido**
3. **Agilize os casos mais comuns**

# Instruções MIPS

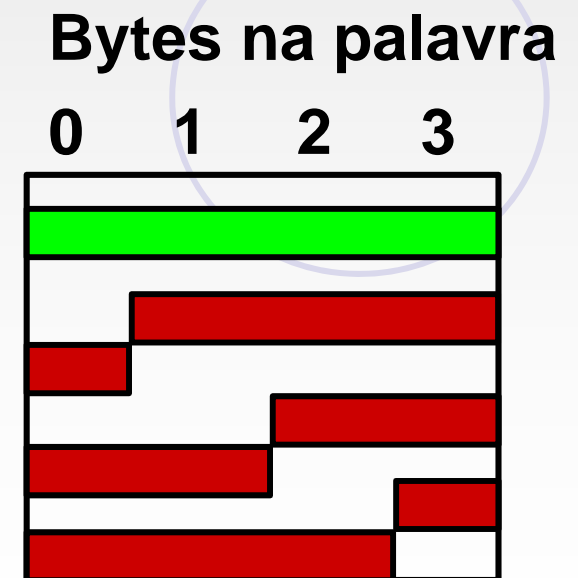
## :: Armazenamento na memória

- O espaço de endereçamento de memória do MIPS é de  $2^{30}$  palavras (de 32 bits)
- MIPS exige que todas as **palavras** comecem em endereços que são **múltiplos de 4 bytes**

endereço	dados
0	100
4	10
8	101
12	1
⋮	⋮
4294967292	77

*Alinhado*

*Não Alinhado*



# Instruções MIPS

## :: Armazenamento na memória

- **Big endian** - byte mais à **esquerda** marca endereço da palavra
- **Little endian** – byte mais à **direita** marca endereço da palavra
- Exemplo: palavra = **6151CE94<sub>h</sub>**, endereço = **0F40**

1) Big Endian:

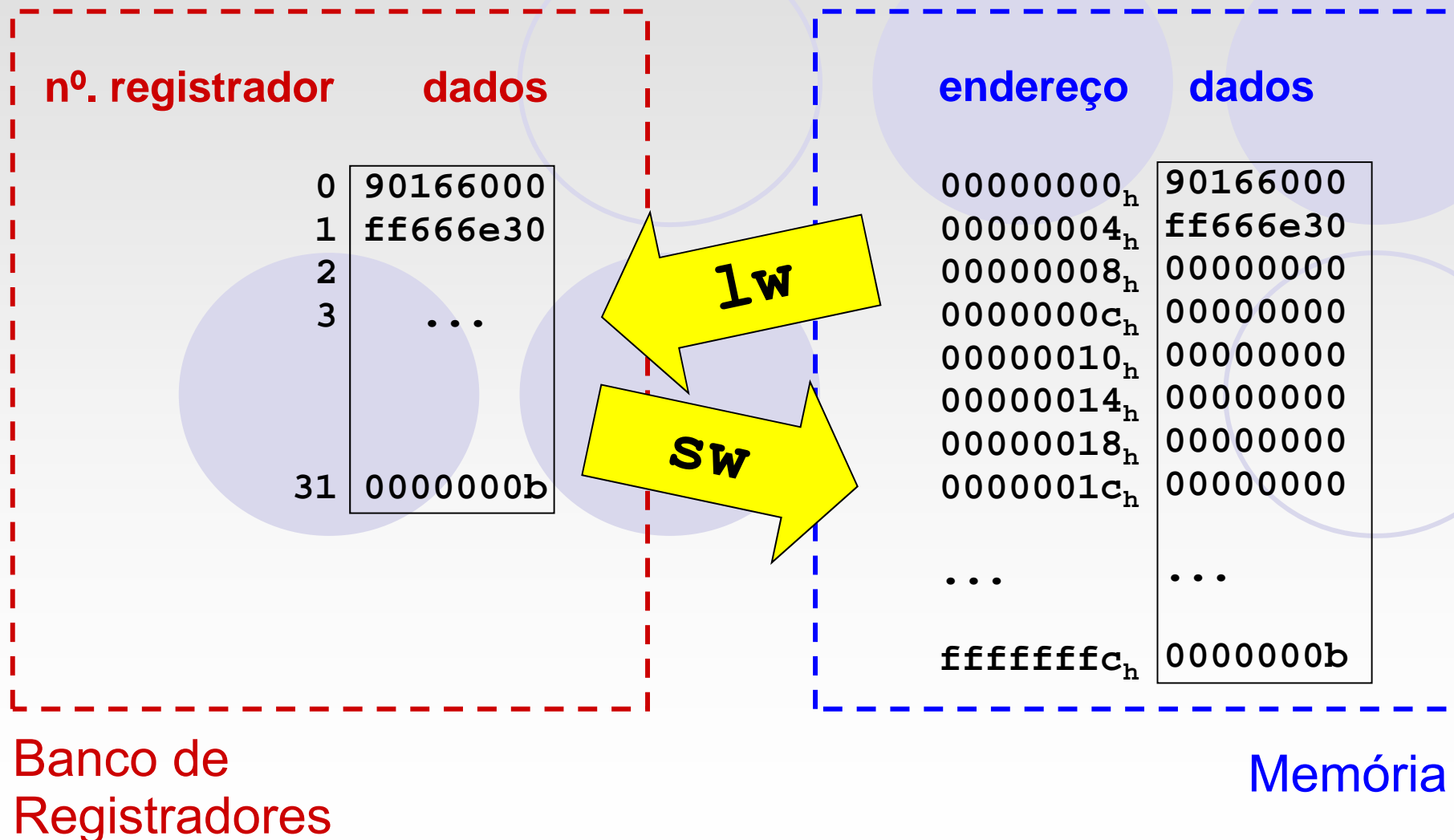
<b>0F40</b>	<b>61</b>	<b>51</b>	<b>CE</b>	<b>94</b>
<b>0F44</b>	<b>00</b>	<b>00</b>	<b>00</b>	<b>00</b>

2) Little Endian:

<b>0F40</b>	<b>94</b>	<b>CE</b>	<b>51</b>	<b>61</b>
<b>0F44</b>	<b>00</b>	<b>00</b>	<b>00</b>	<b>00</b>

# Instruções MIPS

## :: Instruções de transferência de dados



# Instruções MIPS

## :: Instruções de transferência de dados

### ● Load word (lw)

#### Banco de registradores

nº. registrador	dados
0	9016 6000
1	ff66 6e30
2	0000 000c
3	...
...	...
30	0000 012f
31	0000 000b

lw \$30, 4(\$2)

+

#### Memória

endereço	dados
00000000 <sub>h</sub>	9016 6000
00000004 <sub>h</sub>	ff66 6e30
00000008 <sub>h</sub>	0000 0000
0000000c <sub>h</sub>	0000 0000
00000010 <sub>h</sub>	0000 012f
00000014 <sub>h</sub>	0000 0000
00000018 <sub>h</sub>	0000 0000
0000001c <sub>h</sub>	0000 0000
...	...
ffffffff <sub>h</sub>	0000 000b



# Representando instruções no computador

- **Formato registrador (R)**

Op-Code	Rs	Rt	Rd	Shamt	Function Code
000000	sssss	ttttt	ddddd	00000	ffffff

- **Formato immediato (I)**

Op-Code	Rs	Rt	Immediate
ffffff	sssss	ttttt	iiiiiiiiiiiiiiiiiii

- **Formato jump (J)**

[illegible]

# Lei de Amdahl

“O maior aumento de desempenho possível introduzindo melhorias numa determinada característica é limitado pela percentagem em que essa característica é utilizada”

$$\text{Tempo de execução } (T_{ex}) \text{ após melhoria} = \frac{T_{ex} \text{ afetado pela melhoria}}{\text{Quantidade da melhoria}} + T_{ex} \text{ não-afetado}$$

$$T_{novo} = \frac{F \cdot T_{velho}}{S} + (1 - F) \cdot T_{velho} = T_{velho} \left[ \frac{F}{S} + (1 - F) \right]$$

$$\text{Melhoria}_{global} = \frac{T_{velho}}{T_{novo}} = \frac{1}{\frac{F}{S} + (1 - F)}$$

# Sistemas Processadores e Periféricos

## Aula 3 - Conjunto de Instruções MIPS II

Prof. Frank Sill Torres  
DELT – Escola de Engenharia  
UFMG

Adaptado a partir dos Slides de Organização de Computadores 2006/02 do professor

Leandro Galvão DCC/UFAM - [galvao@dcc.ufam.edu.br](mailto:galvao@dcc.ufam.edu.br) pelo Prof. Ricardo de Oliveira Duarte

# Instruções MIPS

---

- Transferência de Dados
- Lógicas
- Controle
- Suporte a procedimentos

# Instruções MIPS

## :: Instruções Lógicas

Operação lógica	Instrução MIPS	Significado
Shift à esquerda	<b>sll</b>	<i>Shift left logical</i>
Shift à direita	<b>srl</b>	<i>Shift right logical</i>
AND bit a bit	<b>and, andi</b>	<i>E</i>
OR bit a bit	<b>or, ori</b>	<i>OR</i>
NOR bit a bit	<b>nor</b>	<i>NOR</i>

# Instruções MIPS

## :: Instruções Lógicas

---

- Operações de deslocamento (shift)
  - Deslocam todos os bits de uma palavra para esquerda ou direita, **preenchendo os bits vazios com zero** (não cíclico)
  - São instruções do **tipo R** (registrador)
  - A **quantidade** de bits a serem deslocados é especificada pelo campo **shamt** (shift amount)

# Instruções MIPS

## :: Instruções Lógicas

- **shift left logical (sll)**

**sll \$t2, \$s0, 4      # \$t2 ← \$s0 << 4**

Conteúdo:

\$s0	0110	1000	1111	0000	0111	0110	1111	1111
	↙	↙	↙	↙	↙	↙	↙	
\$t2	1000	1111	0000	0111	0110	1111	1111	0000

# Instruções MIPS

## :: Instruções Lógicas

- shift left logical (sll)

sll \$t2, \$s0, 4      # \$t2 ← \$s0 << 4

Instrução (decimal):	op	rs	rt	rd	shamt	funct
	0	0	16	10	4	0
	sll	--	\$s0	\$t2	shamt	sll

Instrução (binário):	000000	00000	10000	01010	00100	000000
-------------------------	--------	-------	-------	-------	-------	--------



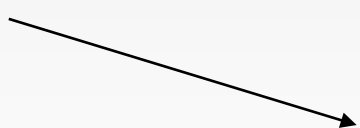
# Instruções MIPS

## :: Instruções Lógicas

- **shift right logical (srl)**

```
srl  $t2, $s0, 8      # $t2 ← $s0 >> 8
```

Conteúdo:

\$s0	0110	1000	1111	0000	0111	0110	1111	1111
								
\$t2	0000	0000	0110	1000	1111	0000	0111	0110

# Instruções MIPS

## :: Instruções Lógicas

- shift right logical (srl)

```
srl  $t2, $s0, 8      # $t2 ← $s0 >> 8
```

Instrução (decimal):	op	rs	rt	rd	shamt	funct
	0	0	16	10	8	2
	srl	--	\$s0	\$t2	shamt	srl

Instrução (binário):	000000	00000	10000	01010	01000	000010
-------------------------	--------	-------	-------	-------	-------	--------

# Instruções MIPS

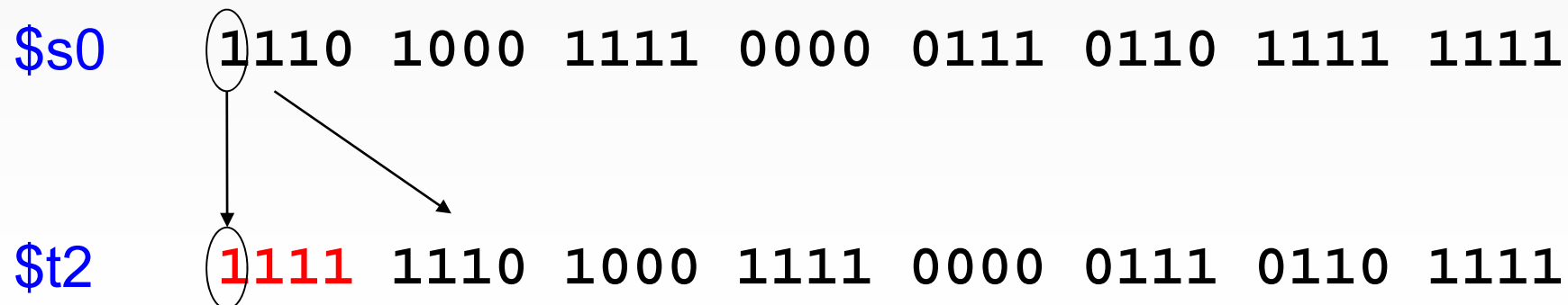
## :: Instruções Lógicas

- **shift right arithmetic (sra)**

- Desloca bits à direita, **preservando** o sinal (compl. a 2)
- Deslocamento de  $n$  bits corresponde à divisão por  $2^n$
- **Dica:** para divisões com valores de tipo integer

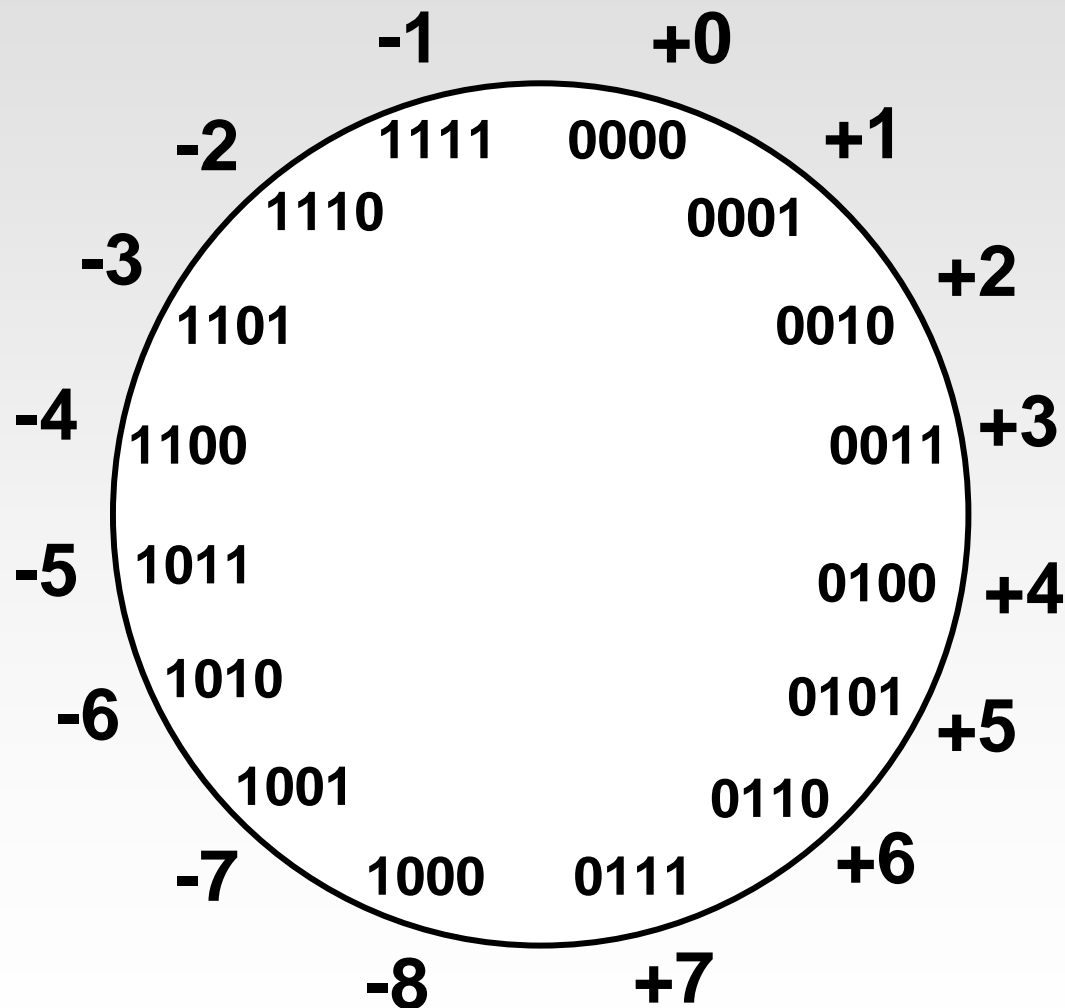
```
sra    $t2, $s0, 4    # $t2 ← $s0 >> 4
```

Conteúdo:



# Instruções MIPS

## :: Revisão – Complemento a 2



+

0 100 = + 4

-8 421

1 100 = - 4

-

# Instruções MIPS

## :: Instruções Lógicas

- **shift right arithmetic (sra)**

```
sra    $t2, $s0, 4    # $t2 ← $s0 >> 4
```

Instrução (decimal):	op	rs	rt	rd	shamt	funct
	0	0	16	10	4	3
	sra	--	\$s0	\$t2	shamt	sra

Instrução (binário):	000000	00000	10000	01010	00100	000011
-------------------------	--------	-------	-------	-------	-------	--------

# Instruções MIPS

## :: Instruções Lógicas

- **AND bit a bit (and)**

`and $t0, $t1, $t2`      #  $\$t0 \leftarrow \$t1 \& \$t2$

Conteúdo:

<code>\$t1</code>	0000	0000	1111	0000	0000	0110	0000	1111
<code>\$t2</code>	0000	0000	0110	0000	0000	0000	0000	1110
<code>\$t0</code>	0000	0000	0110	0000	0000	0000	0000	1110

# Instruções MIPS

## :: Instruções Lógicas

- **AND bit a bit (and)**

`and $t0, $t1, $t2`      #  $\$t0 \leftarrow \$t1 \& \$t2$

	op	rs	rt	rd	shamt	funct
Instrução (decimal):	0	9	10	8	0	(24) <sub>h</sub>
	and	\$t1	\$t2	\$t0	--	and

Instrução (binário):	000000	01001	01010	01000	00000	100100
-------------------------	--------	-------	-------	-------	-------	--------

# Instruções MIPS

## :: Instruções Lógicas

- **OR bit a bit (or)**

`or $t0, $t1, $t2      # $t0 ← $t1 | $t2`

### Conteúdo:

\$t1	0000	0000	1111	0000	0000	0110	0000	1111
\$t2	0000	0000	0110	0000	0000	0000	0000	1110
\$t0	0000	0000	1111	0000	0000	0110	0000	1111



# Instruções MIPS

## :: Instruções Lógicas

- **OR bit a bit (or)**

`or $t0, $t1, $t2      # $t0 ← $t1 | $t2`

Instrução (decimal):	op	rs	rt	rd	shamt	funct
	0	9	10	8	0	(25) <sub>h</sub>
	or	\$t1	\$t2	\$t0	--	or

Instrução (binário):	000000	01001	01010	01000	00000	100101
-------------------------	--------	-------	-------	-------	-------	--------

# Instruções MIPS

## :: Instruções Lógicas

---

- **NOR bit a bit (nor)**

- A operação **NOT** (negação) tem apenas um operando
- Para acompanhar o formato de dois operandos, a instrução **NOR** foi projetada no lugar do **NOT**
- Se um operando for zero, NOR é **equivalente** ao NOT:
  - $A \text{ NOR } 0 = \text{NOT} (A \text{ OR } 0) = \text{NOT} (A)$

# Instruções MIPS

## :: Instruções Lógicas

- NOR bit a bit (nor)**

`nor $t0, $t1, $zero      # $t0 ← ~ $t1`

Instrução (decimal):	op	rs	rt	rd	shamt	funct
	0	9	0	8	0	(27) <sub>h</sub>
	nor	\$t1	\$zero	\$t0	--	nor

Instrução (binário):	000000	01001	00000	01000	00000	100111
-------------------------	--------	-------	-------	-------	-------	--------

# Instruções MIPS

## :: Instruções Lógicas

- XOR bit a bit (xor)**

**xor**    \$t0, \$t1, \$t2        # \$t0  $\leftarrow$  \$t1  $\oplus$  \$t2

	op	rs	rt	rd	shamt	funct
Instrução (decimal):	0	9	10	8	0	(26) <sub>h</sub>
	xor	\$t1	\$t2	\$t0	--	xor

Instrução (binário):	000000	01001	01010	01000	00000	100110
-------------------------	--------	-------	-------	-------	-------	--------

# Instruções MIPS

## :: Instruções Lógicas

- **AND imediato (andi)**

`andi $t0, $t1, 40`      #  $\$t0 \leftarrow \$t1 \& 40$

Instrução  
(decimal):

op	rs	rt	immediate
$(C)_h$	9	8	40
<code>andi</code>	<code>\$t1</code>	<code>\$t0</code>	<code>constante</code>

Instrução  
(binário):

<code>001100</code>	<code>01001</code>	<code>01000</code>	<code>0000000000101000</code>
---------------------	--------------------	--------------------	-------------------------------

# Instruções MIPS

## :: Instruções Lógicas

- **OR imediato (ori)**

`ori $t0, $t1, 40`      #  $\$t0 \leftarrow \$t1 \mid 40$

Instrução  
(decimal):

op	rs	rt	immediate
$(D)_h$	9	8	40
<code>ori</code>	<code>\$t1</code>	<code>\$t0</code>	<code>constante</code>

Instrução  
(binário):

<code>001101</code>	<code>01001</code>	<code>01000</code>	<code>0000000000101000</code>
---------------------	--------------------	--------------------	-------------------------------

# Instruções MIPS

## :: Instruções Lógicas :: Resumo

Categoria	Nome	Exemplo	Operação	Comentários
lógicas	or	or \$8, \$9, \$10	$\$8 = \$9 \text{ or } \$10$	
	and	and \$8, \$9, \$10	$\$8 = \$9 \text{ and } \$10$	
	xor	xor \$8, \$9, 40	$\$8 = \$9 \text{ xor } 40$	
	nor	nor \$8, \$9, \$10	$\$8 = \$9 \text{ nor } \$10$	
	andi	andi \$8, \$9, 5	$\$8 = \$9 \text{ and } 5$	Imediato em 16 bits
	ori	ori \$8, \$9, 40	$\$8 = \$9 \text{ or } 40$	Imediato em 16 bits
	sll	sll \$8, \$9, 10	$\$8 = \$9 \ll 10$	Desloc. $\leq 32$
	srl	srl \$8, \$9, 5	$\$8 = \$9 \gg 5$	Desloc. $\leq 32$
	sra	sra \$8, \$9, 5	$\$8 = \$9 \gg 5$	Desloc. $\leq 32$ , preserva sinal

**Tabela 2.3: Operações Aritméticas do MIPS**

# Instruções MIPS

---

- Transferência de Dados
- Lógicas
- Controle
- Suporte a procedimentos



# Instruções MIPS

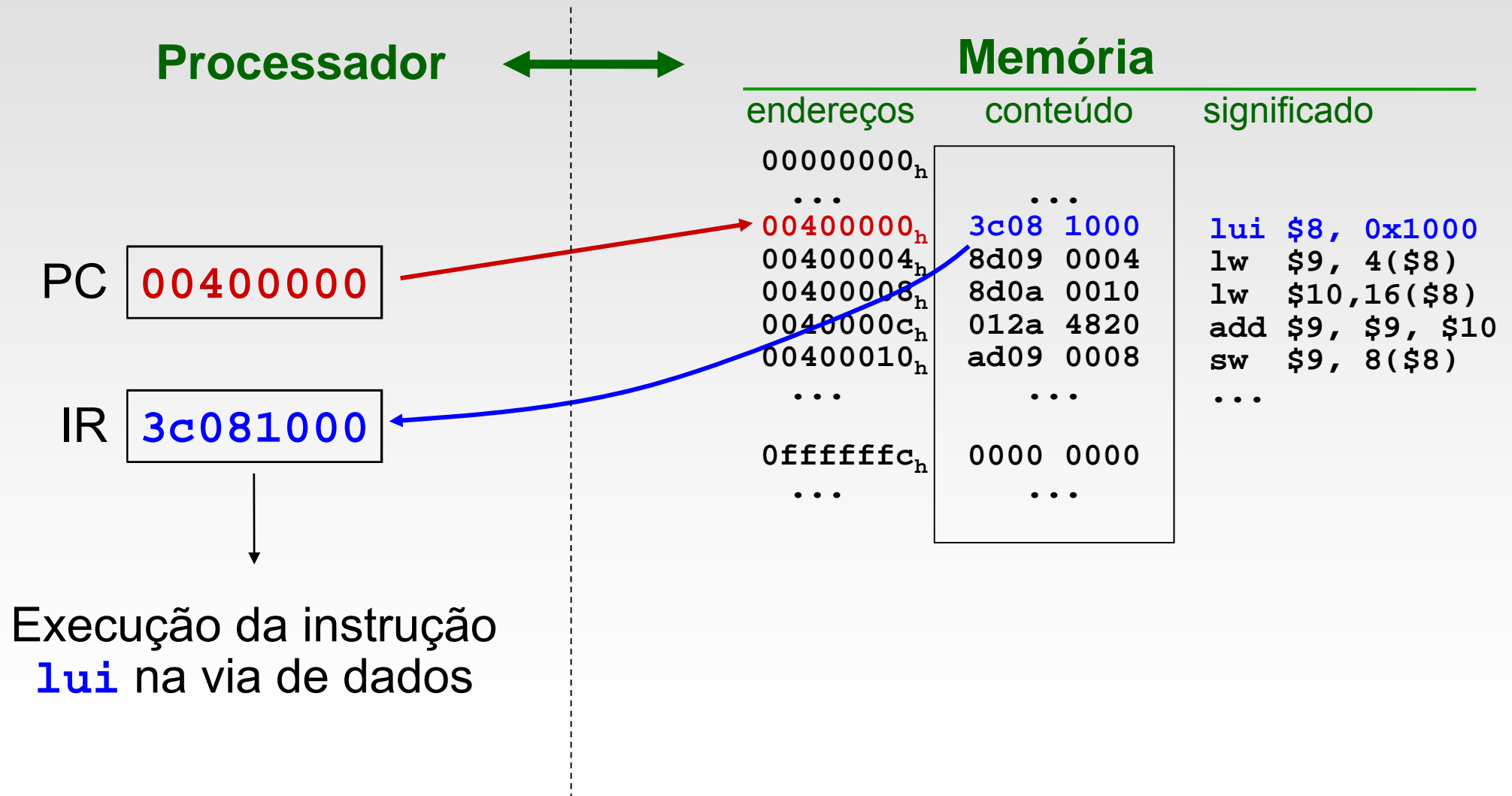
## :: Processando instruções sequenciais

### Memória

	endereços	conteúdo	significado	
	00000000 <sub>h</sub>			
	...	...		
Região de Códigos	00400000 <sub>h</sub>	3c08 1000	lui \$8, 0x1000	instruções
	00400004 <sub>h</sub>	8d09 0004	lw \$9, 4(\$8)	
	00400008 <sub>h</sub>	8d0a 0010	lw \$10, 16(\$8)	
	0040000c <sub>h</sub>	012a 4820	add \$9, \$9, \$10	
	00400010 <sub>h</sub>	ad09 0008	sw \$9, 8(\$8)	
	...	...	...	
Região de Dados	0ffffffc <sub>h</sub>	0000 0000		variáveis
	10000000 <sub>h</sub>	0000 0003	x	
	10000004 <sub>h</sub>	ffff fff0	y	
	10000008 <sub>h</sub>	0000 0000	n[0]	
	1000000c <sub>h</sub>	0000 0000	n[1]	
	10000010 <sub>h</sub>	0000 0003	n[2]	
	...	...	...	
	10007ffc <sub>h</sub>			
	10008000 <sub>h</sub>			
	10008004 <sub>h</sub>			
	10008008 <sub>h</sub>			
	...			
	ffffffffc <sub>h</sub>			

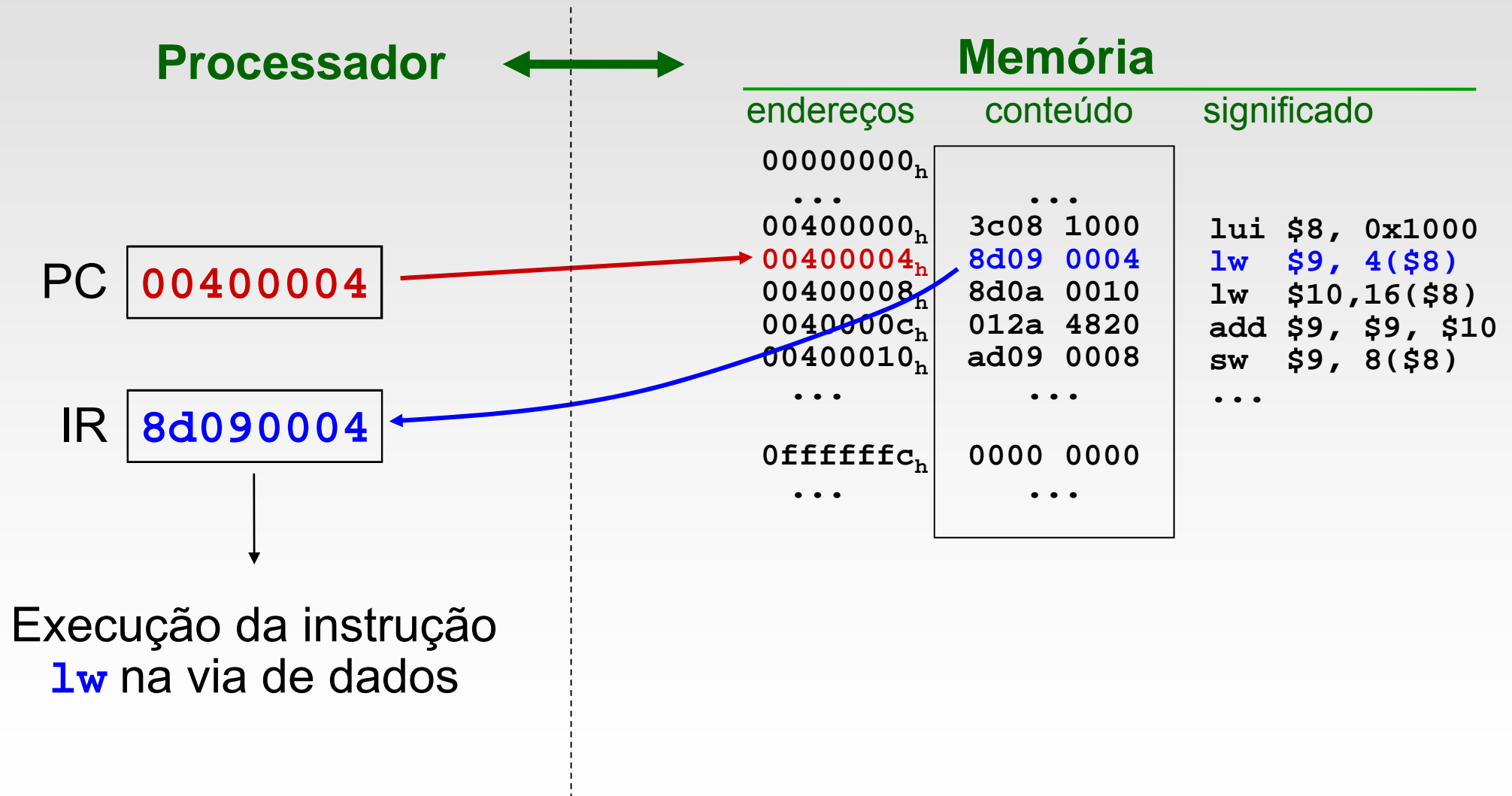
# Instruções MIPS

## :: Processando instruções seqüenciais



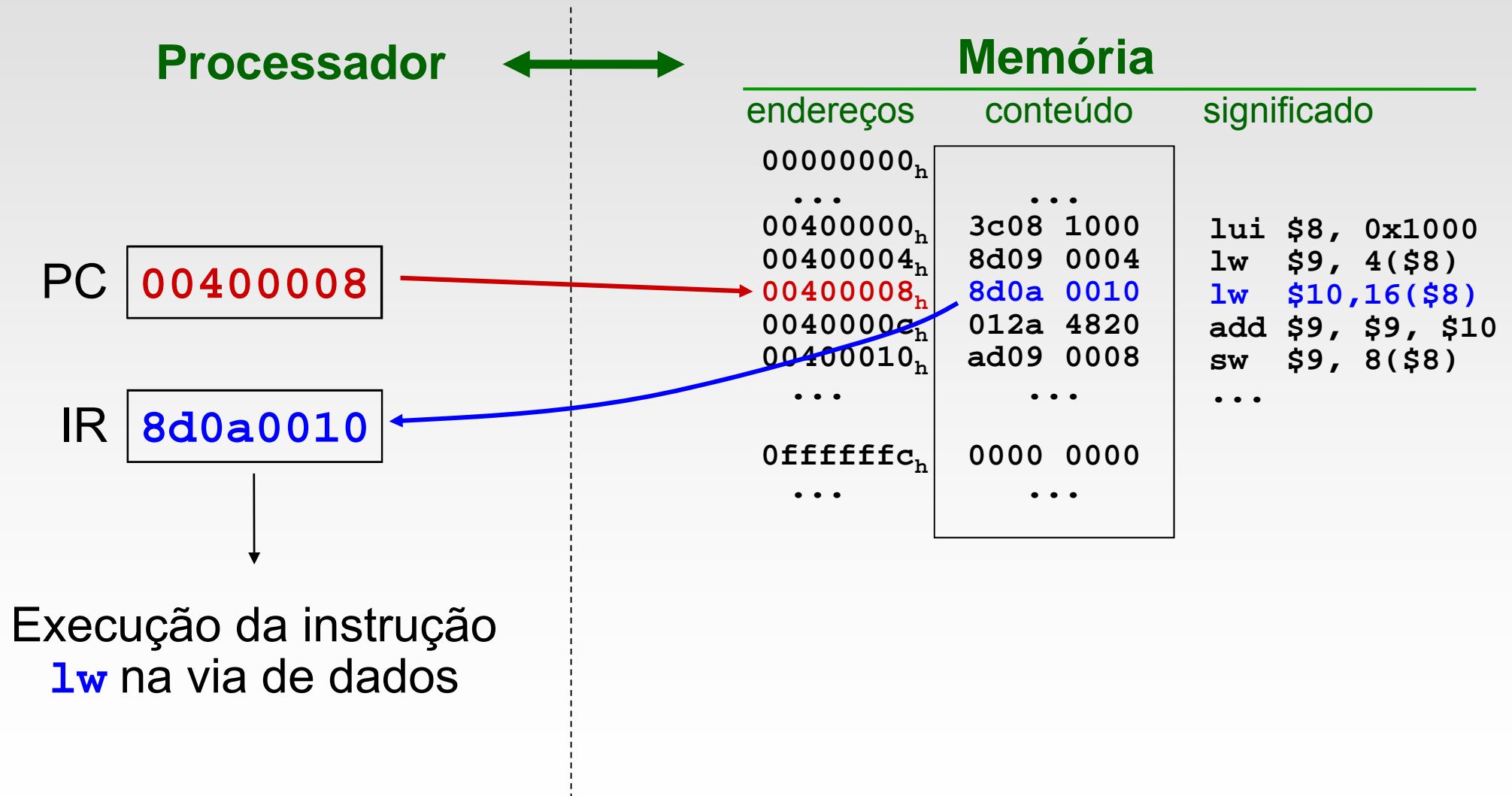
# Instruções MIPS

## :: Processando instruções seqüenciais



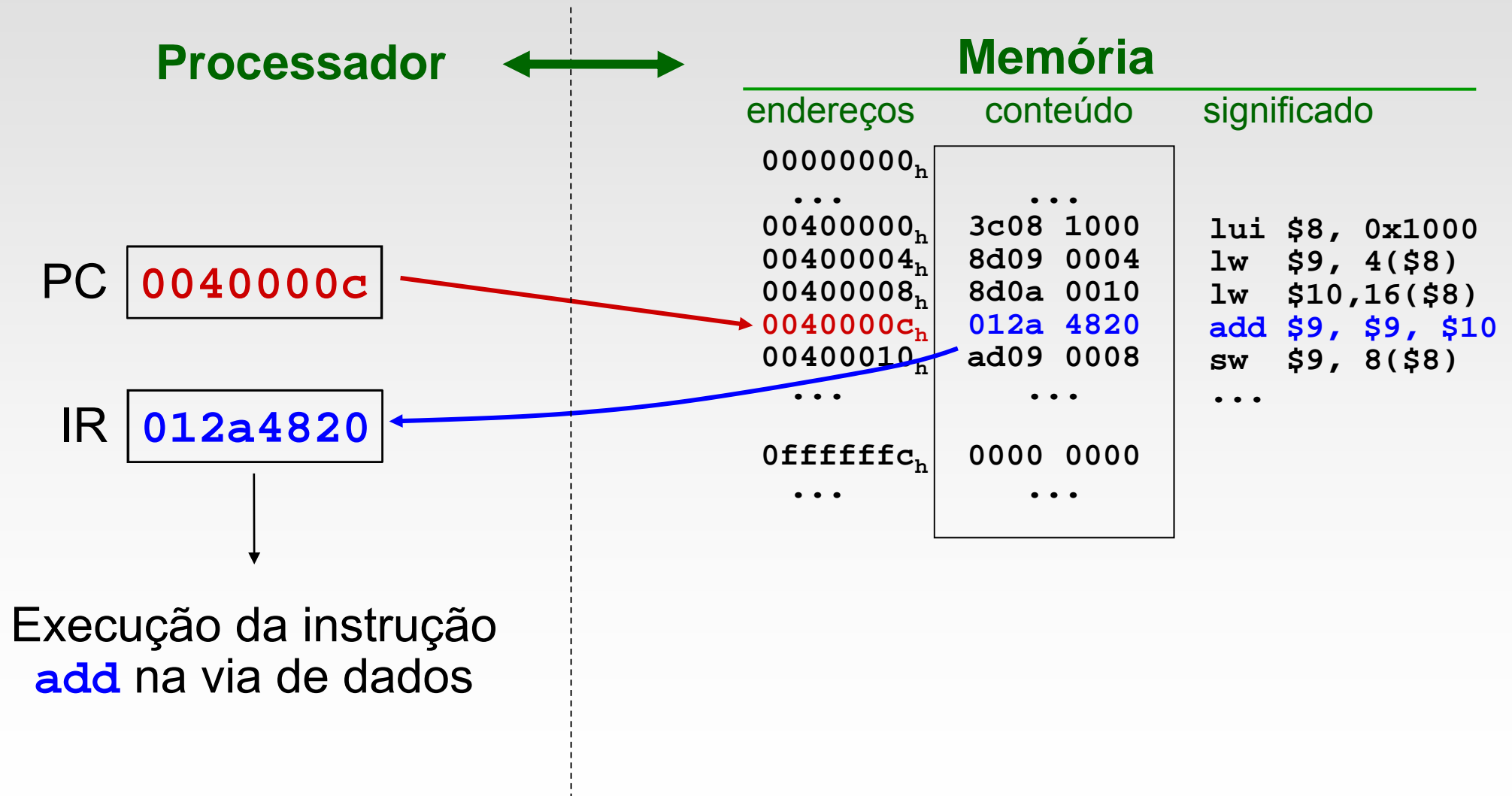
# Instruções MIPS

## :: Processando instruções seqüenciais



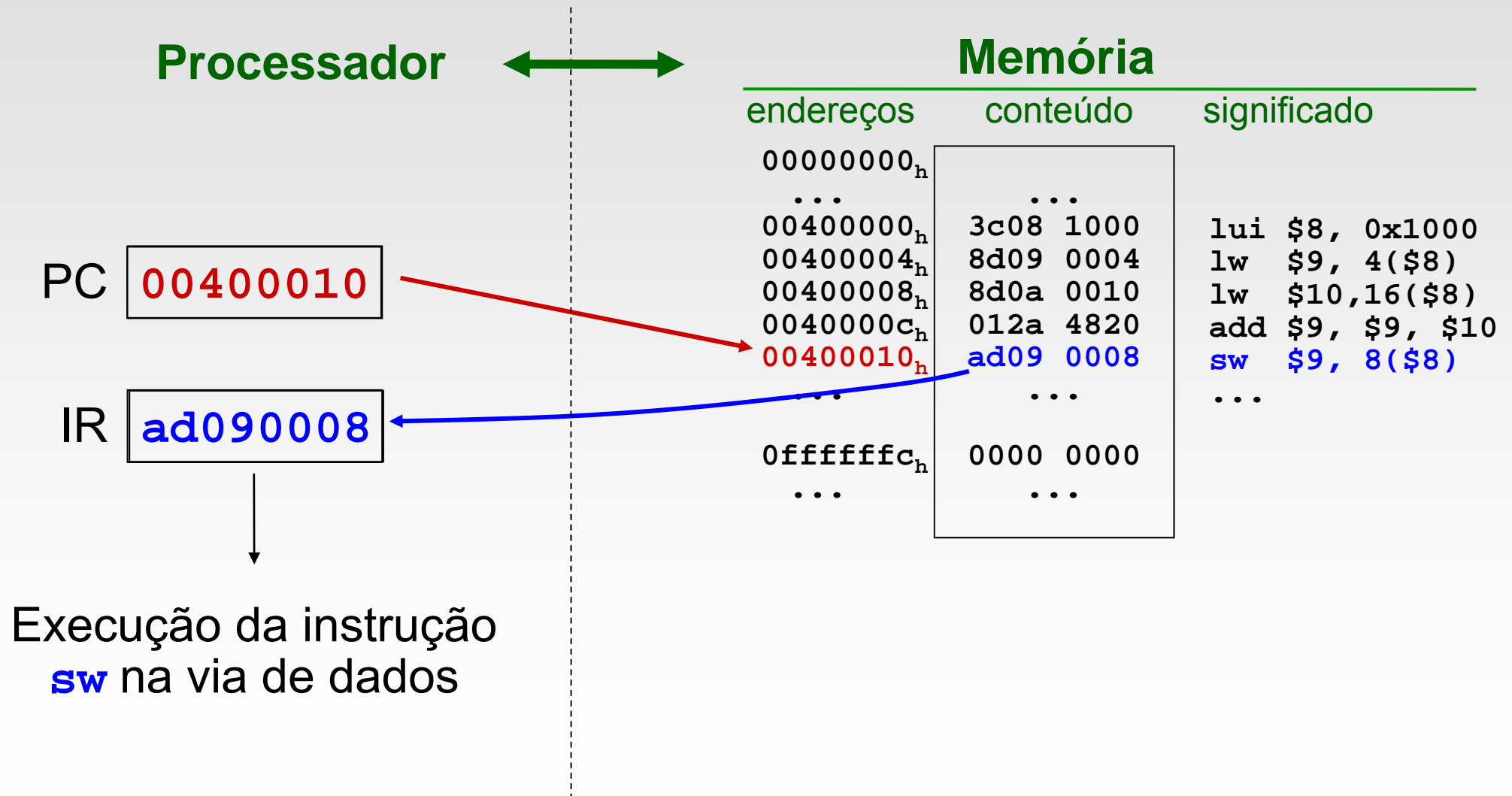
# Instruções MIPS

## :: Processando instruções seqüenciais



# Instruções MIPS

## :: Processando instruções seqüenciais



# Instruções MIPS

## :: Instruções de controle

---

- Instruções para tomada de decisão
  - Alteram o fluxo de controle do programa
  - Alteram a “próxima” instrução a ser executada
- Instruções de controle:
  - Salto condicional
  - Salto incondicional

# Instruções MIPS

## :: Instruções de controle

---

- Instruções MIPS para salto **condicional**:
  - Branch on equal (**beq**)
  - Branch on not equal (**bne**)
  - Set on less than (**slt**)
  - Set on less than immediate (**slti**)
- Instruções MIPS para salto **incondicional**:
  - jump (**j**)



# Instruções MIPS

## :: Instruções de controle

- **Branch on not equal (bne)**

- Desvia o programa para <label1> se \$t0 != \$t1

```
bne    $t0, $t1, label1    #if ($t0 != $t1) goto label1
```

- **Branch on equal (beq)**

- Desvia o programa para <label2> se \$t0 == \$t1

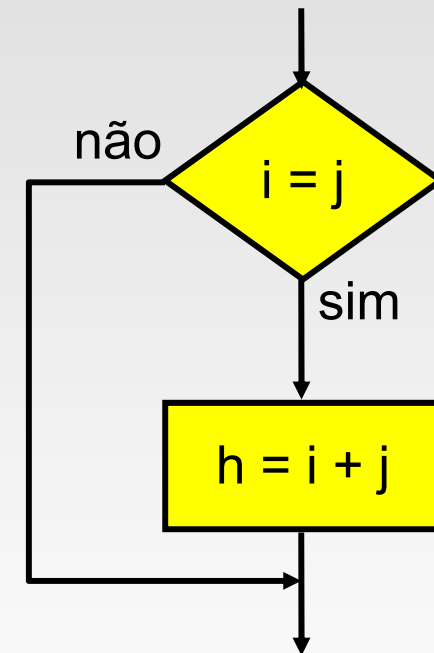
```
beq    $t0, $t1, label2    #if ($t0 == $t1) goto label2
```

# Instruções MIPS

## :: Instruções de controle :: Ex 01

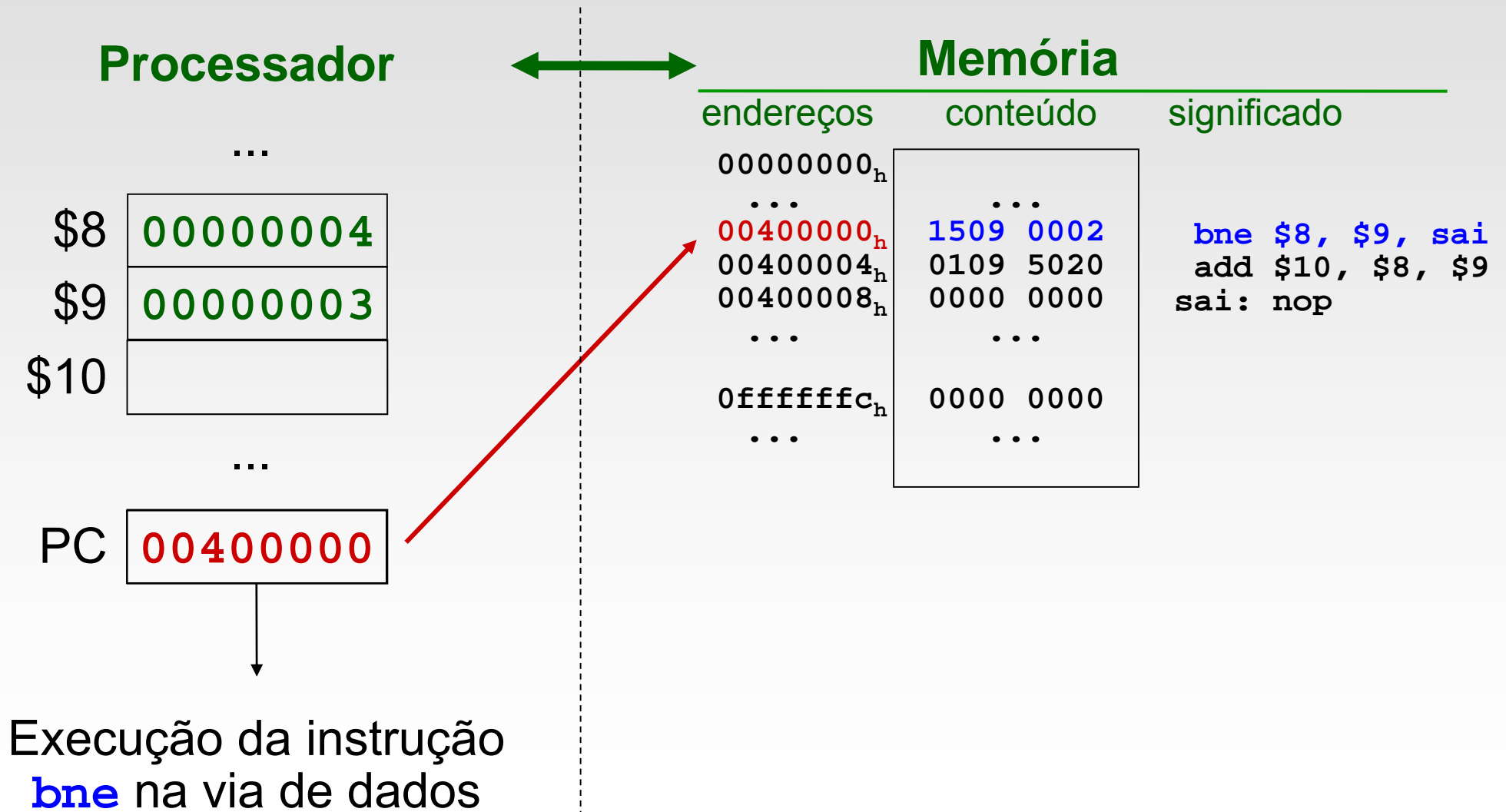
- Exemplo

```
bne $8, $9, sai  
add $10, $8, $9  
sai:  nop
```



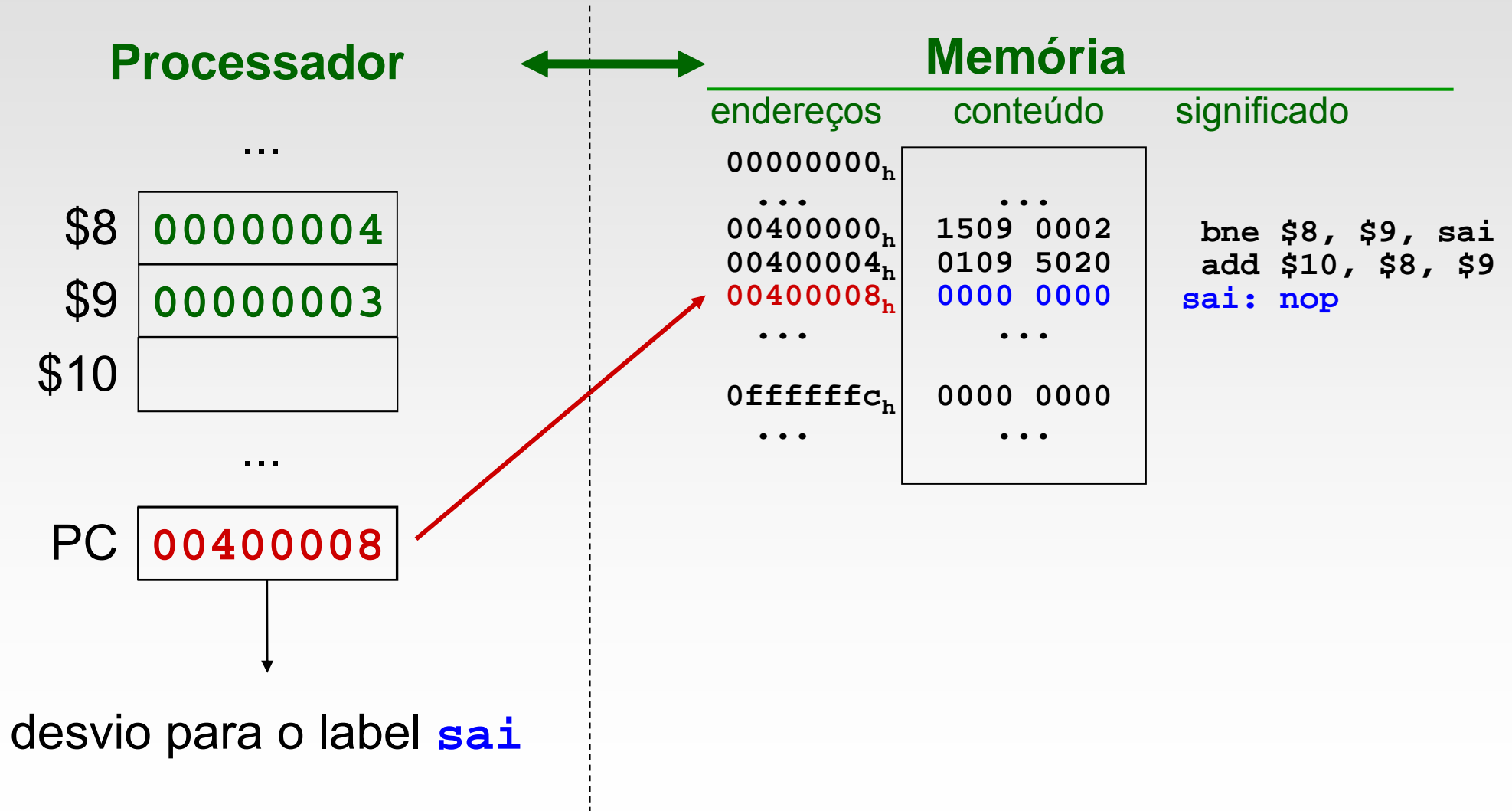
# Instruções MIPS

## :: Instruções de controle :: Ex 01.1



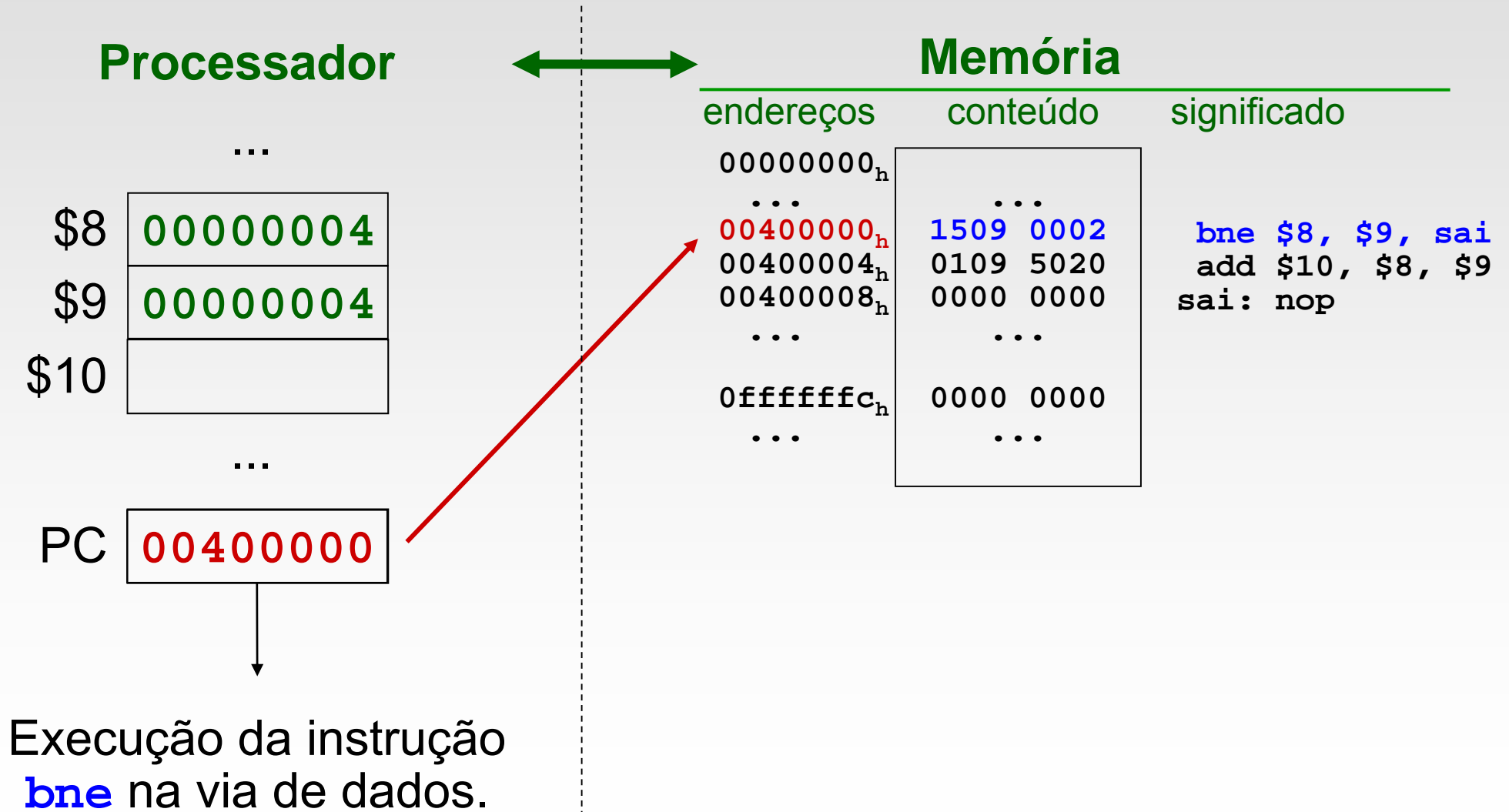
# Instruções MIPS

## :: Instruções de controle :: Ex 01.1



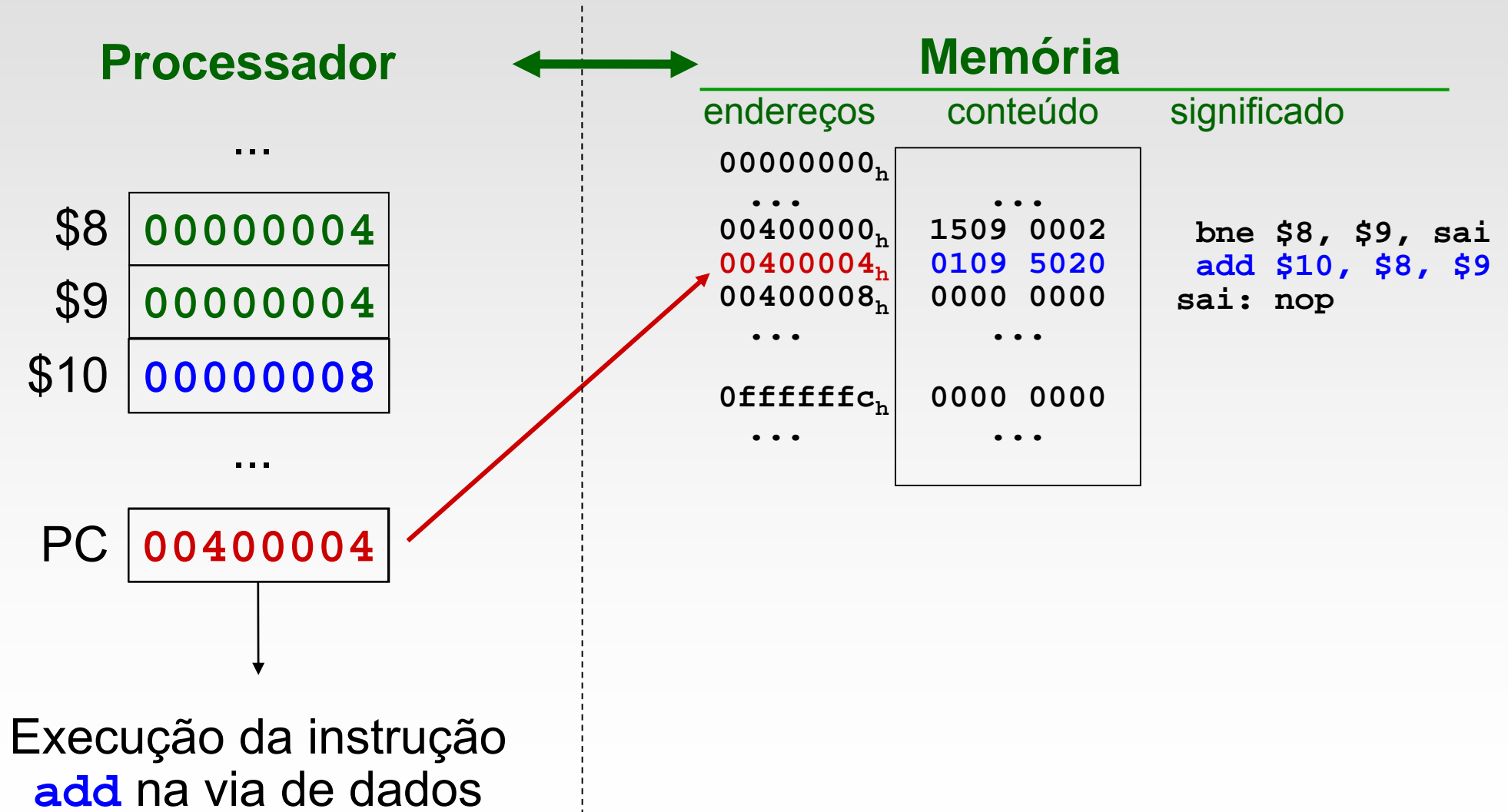
# Instruções MIPS

## :: Instruções de controle :: Ex 01.2



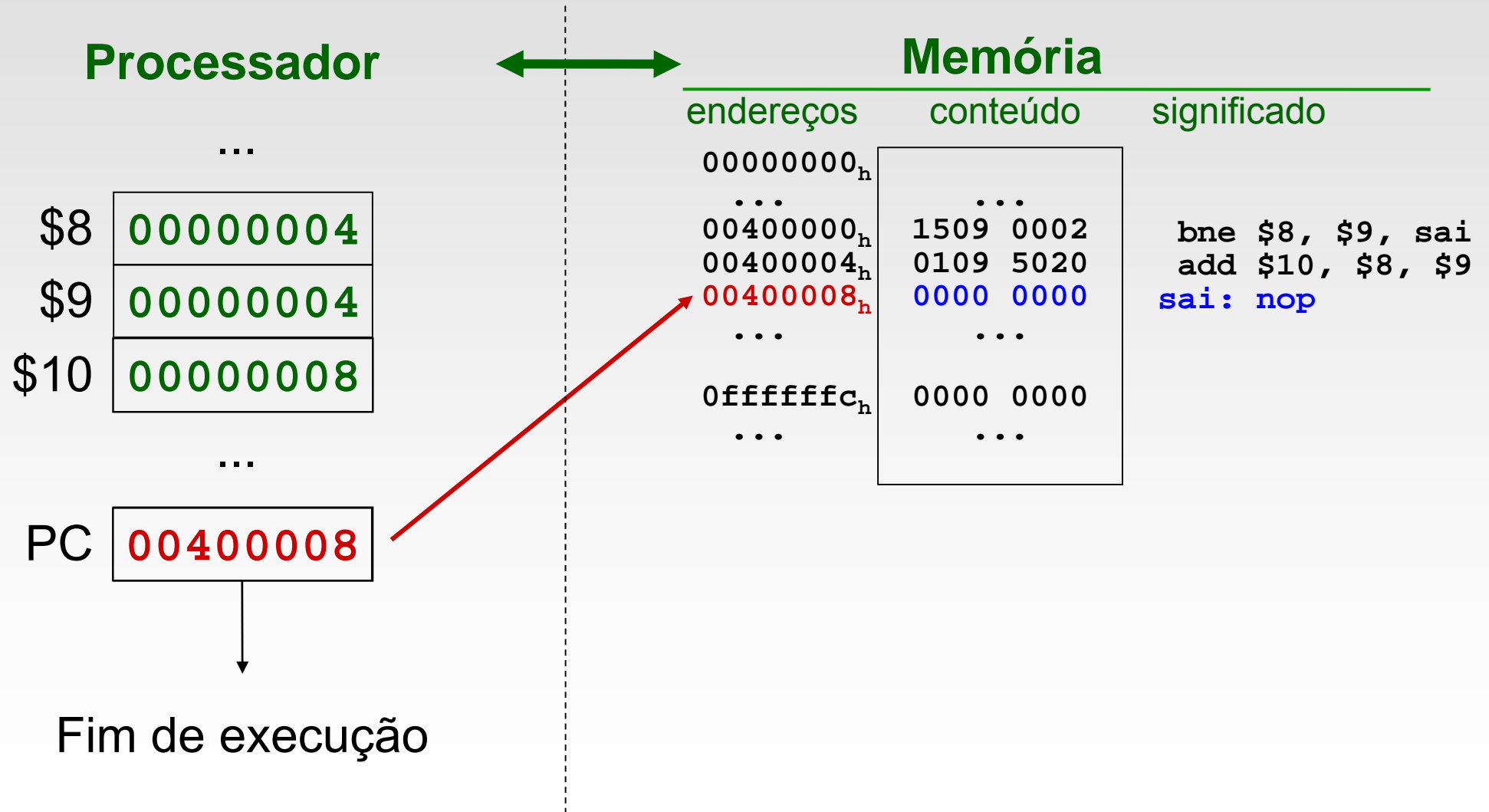
# Instruções MIPS

## :: Instruções de controle :: Ex 01.2



# Instruções MIPS

## :: Instruções de controle :: Ex 01.2



# Instruções MIPS

## :: Instruções de controle :: Formato

- Branch on not equal (bne)

```
bne $t0, $t1, label
```

Instrução  
(decimal):

op	rs	rt	immediate
5	8	9	xx
bne	\$t0	\$t1	nº. words saltadas

Instrução  
(binário):

000101	01000	01001	xxx (16 bits)
--------	-------	-------	---------------



# Instruções MIPS

## :: Instruções de controle :: Formato

- Branch on equal (beq)

```
beq $t0, $t1, label
```

Instrução  
(decimal):

op	rs	rt	immediate
4	8	9	xx
beq	\$t0	\$t1	nº. words saltadas

Instrução  
(binário):

000100	01000	01001	xxx (16 bits)
--------	-------	-------	---------------

# Instruções MIPS

## :: Instruções de controle :: Formato

---

- As instruções de desvio condicional armazenam, no campo immediate, a **quantidade de palavras** (words) que devem ser saltadas para chegar 'a instrução marcada pelo label (rótulo)
- Um número **positivo** indica que o salto deve ser no sentido **para frente** (até o fim do código)
- Um número **negativo** indica que o salto deve ser no sentido **para trás** (até o início do código)

# Instruções MIPS

## :: Instruções de controle

- **Set on less than (slt)**

- Compara dois registradores

```
slt    $s1, $s2, $s3    #if ($s2 < $s3) $s1 = 1  
                                #else $s1 = 0
```

- **Set on less than immediate (slti)**

- Compara um registrador e uma constante

```
slti   $s1, $s2, 100    #if ($s2 < 100) $s1 = 1  
                                #else $s1 = 0
```

# Instruções MIPS

## :: Instruções de controle

---

- Instruções do tipo “Set on less than” combinadas com instruções do tipo “Branch” permitem criar **todas as condições relativas**:
  - igual
  - diferente
  - maior
  - maior ou igual
  - menor
  - menor ou igual

# Instruções MIPS

## :: Instruções de controle

---

- **Jump (j)**

- Desvio incondicional para um endereço de memória apontado por um label

`j     label`

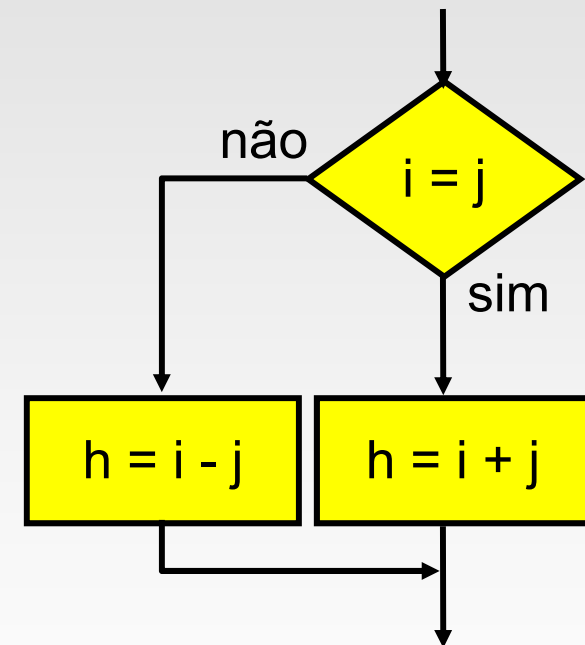
- Instruções do tipo **branch** indicam desvio da sequência do programa mediante **análise de uma condição**
- Instruções do tipo **jump** indicam desvio **incondicional** da sequência do programa

# Instruções MIPS

## :: Instruções de controle :: Ex 02

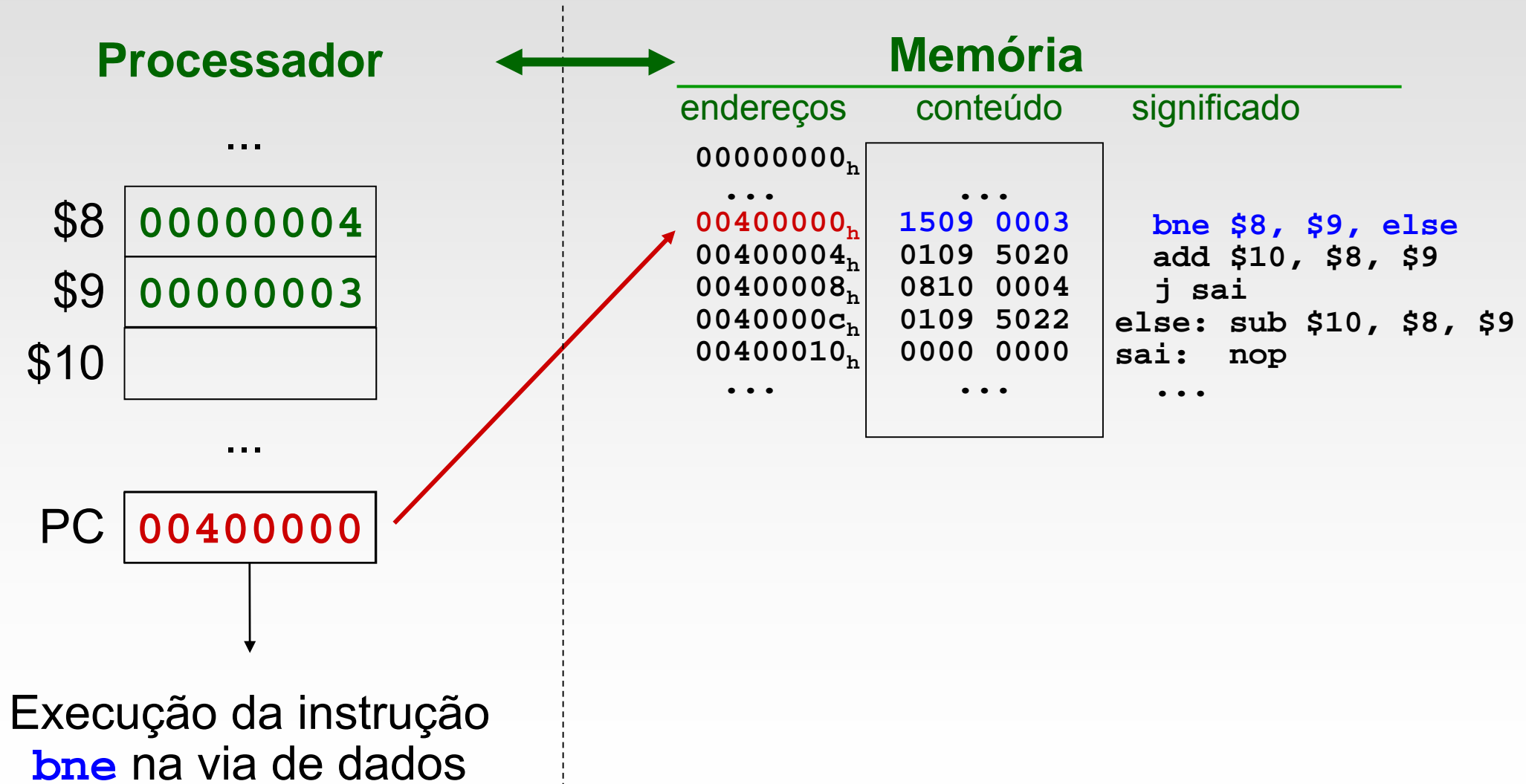
- Exemplo

```
        bne    $8, $9, else
        add    $10, $8, $9
        j      sai
else:    sub    $10, $8, $9
sai:    nop
```



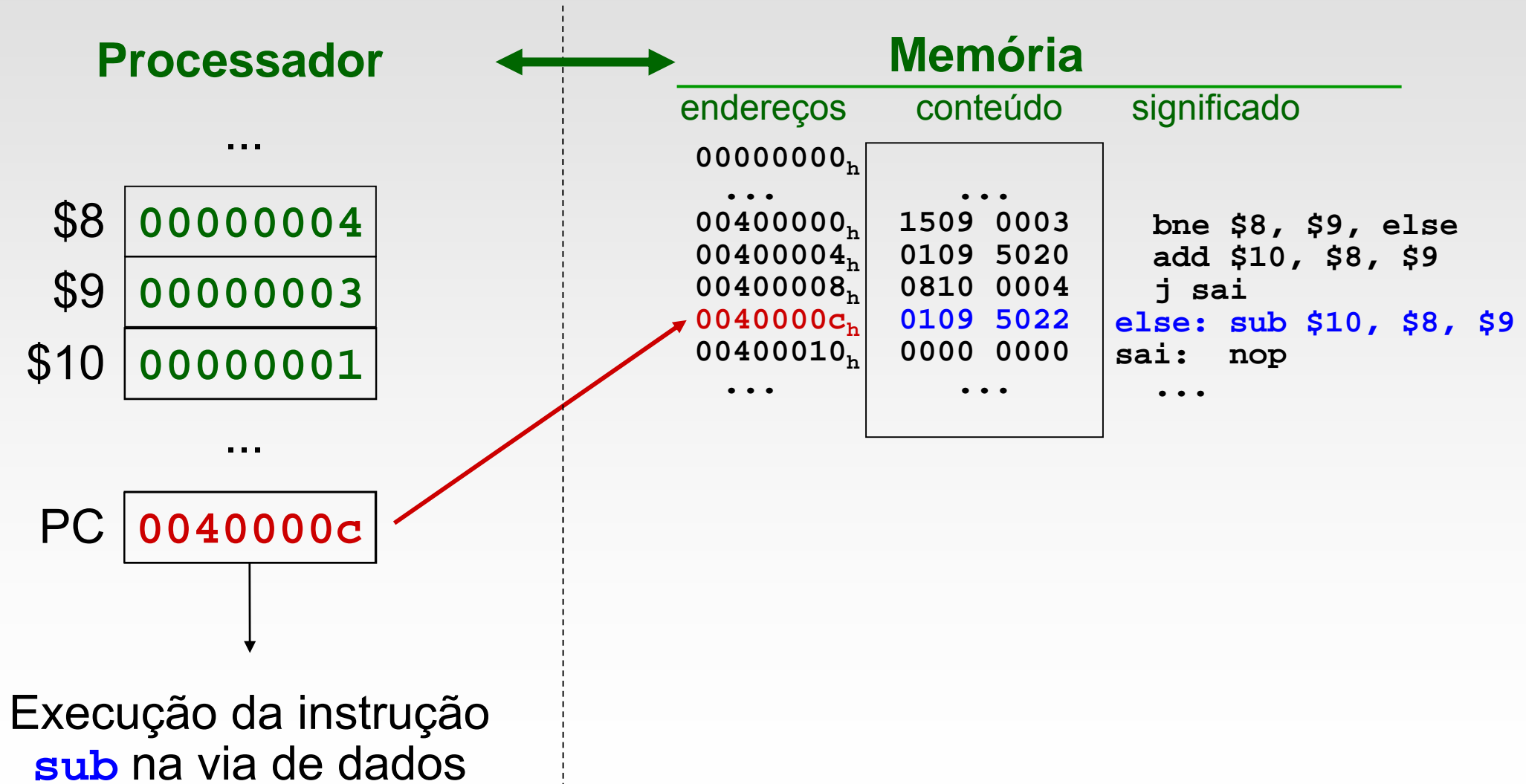
# Instruções MIPS

## :: Instruções de controle :: Ex 02.1



# Instruções MIPS

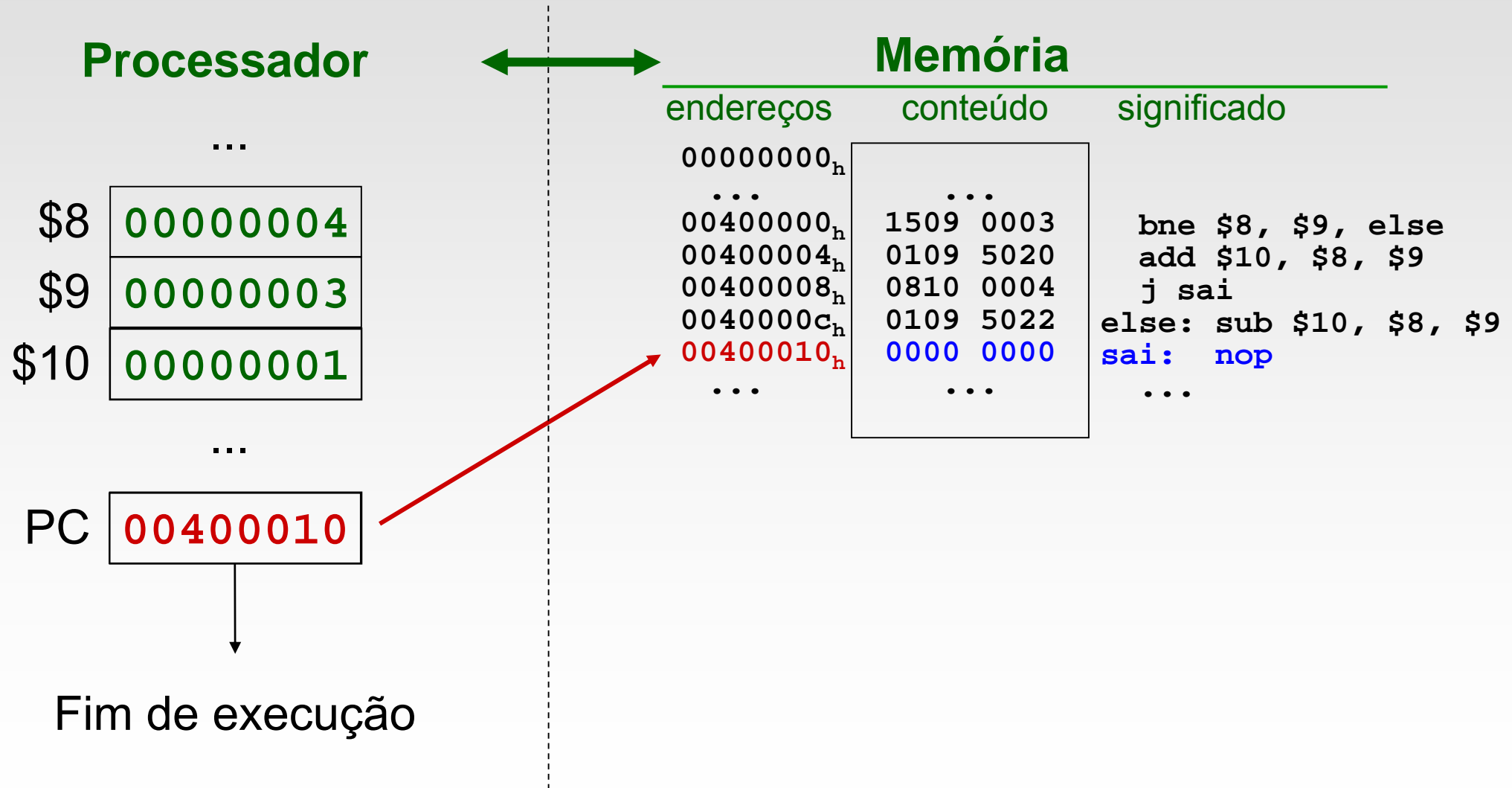
## :: Instruções de controle :: Ex 02.1





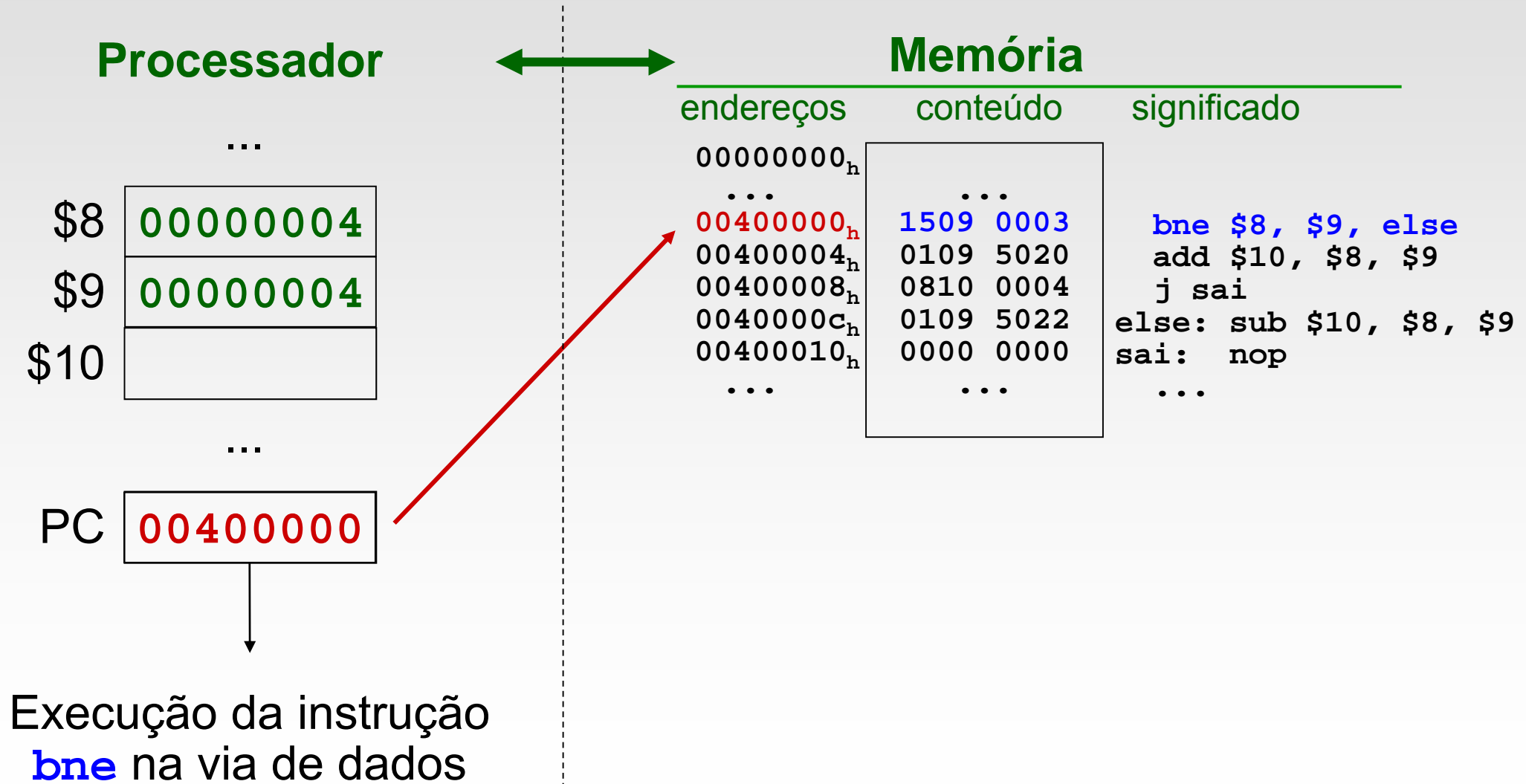
# Instruções MIPS

## :: Instruções de controle :: Ex 02.1



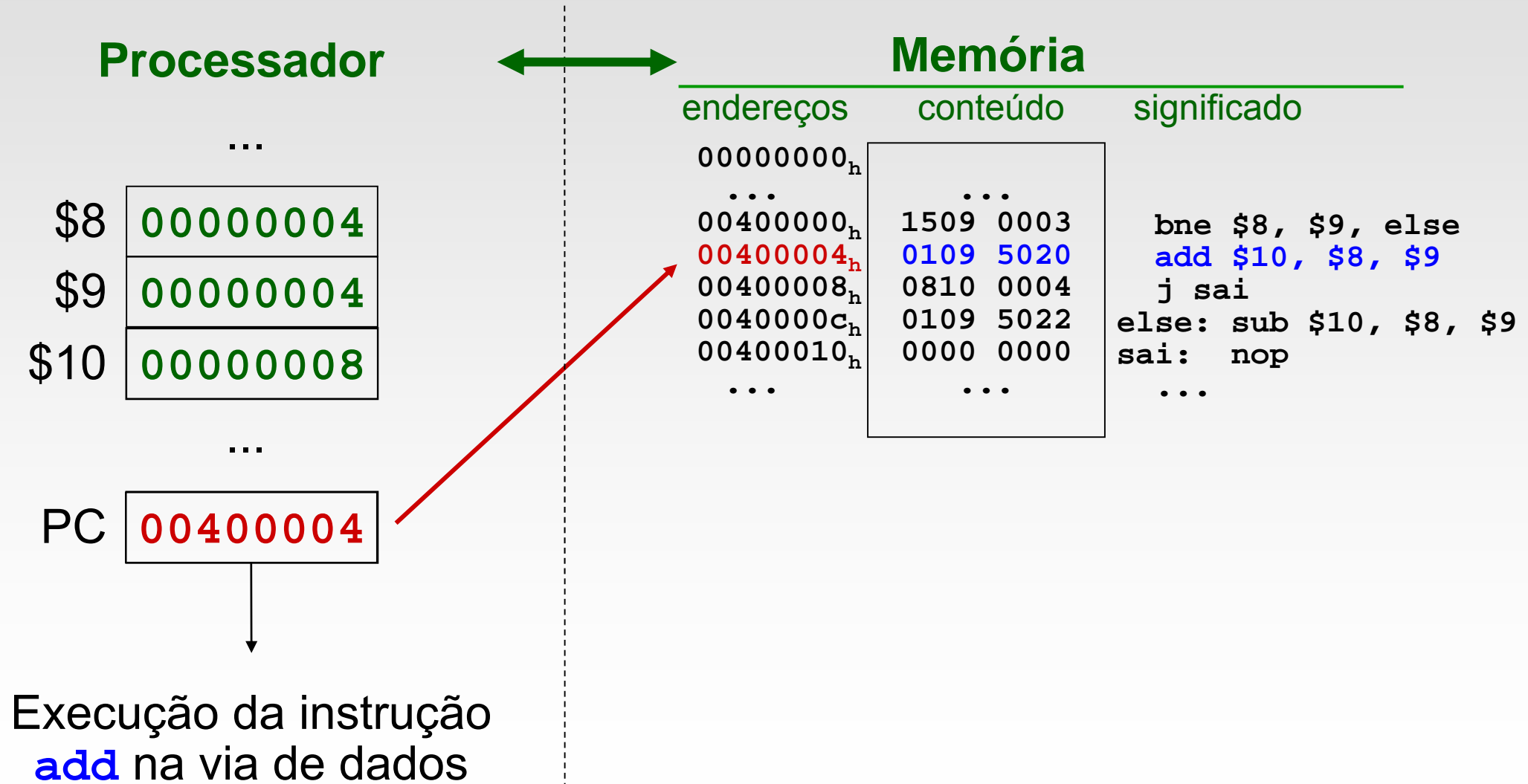
# Instruções MIPS

## :: Instruções de controle :: Ex 02.2



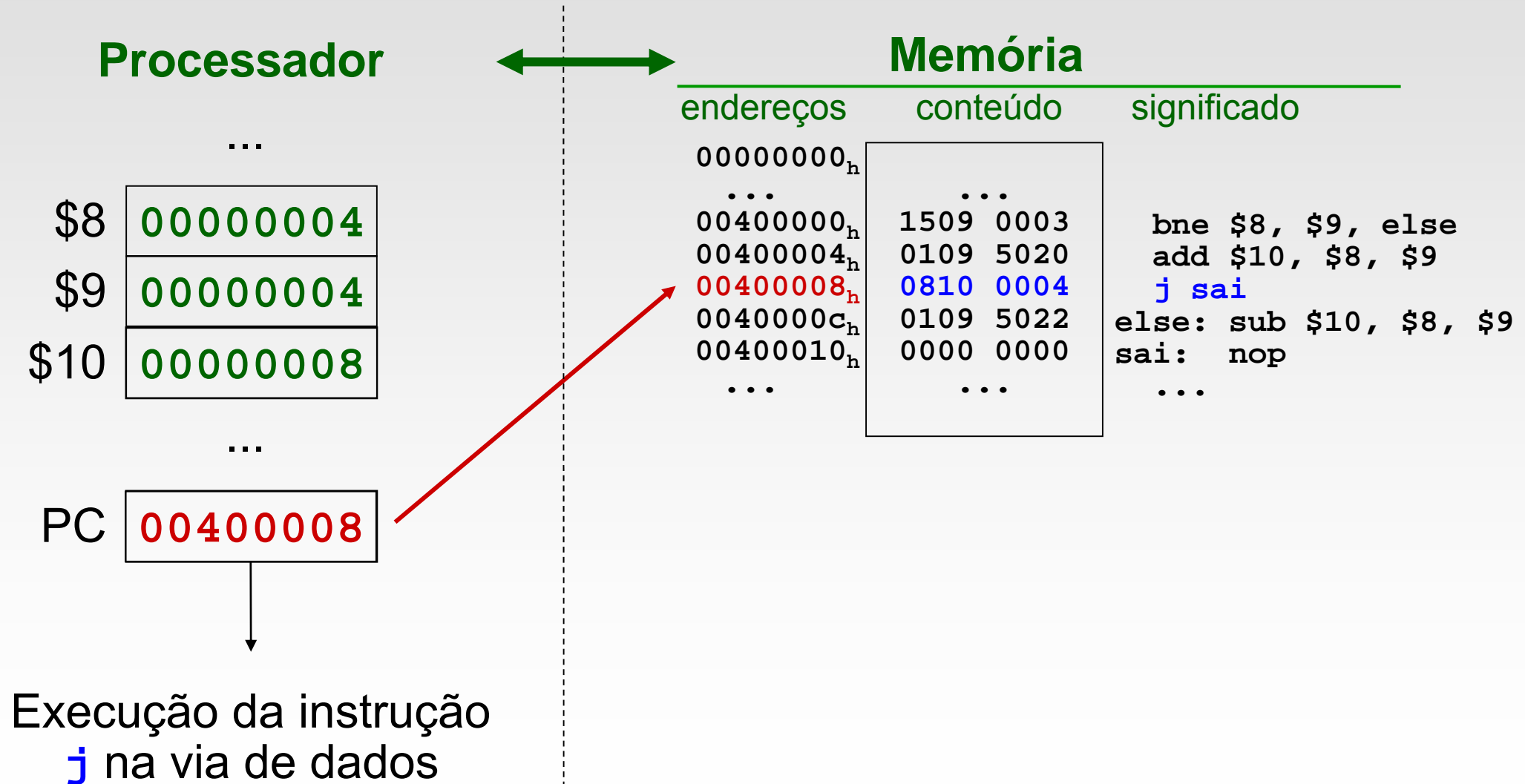
# Instruções MIPS

## :: Instruções de controle :: Ex 02.2



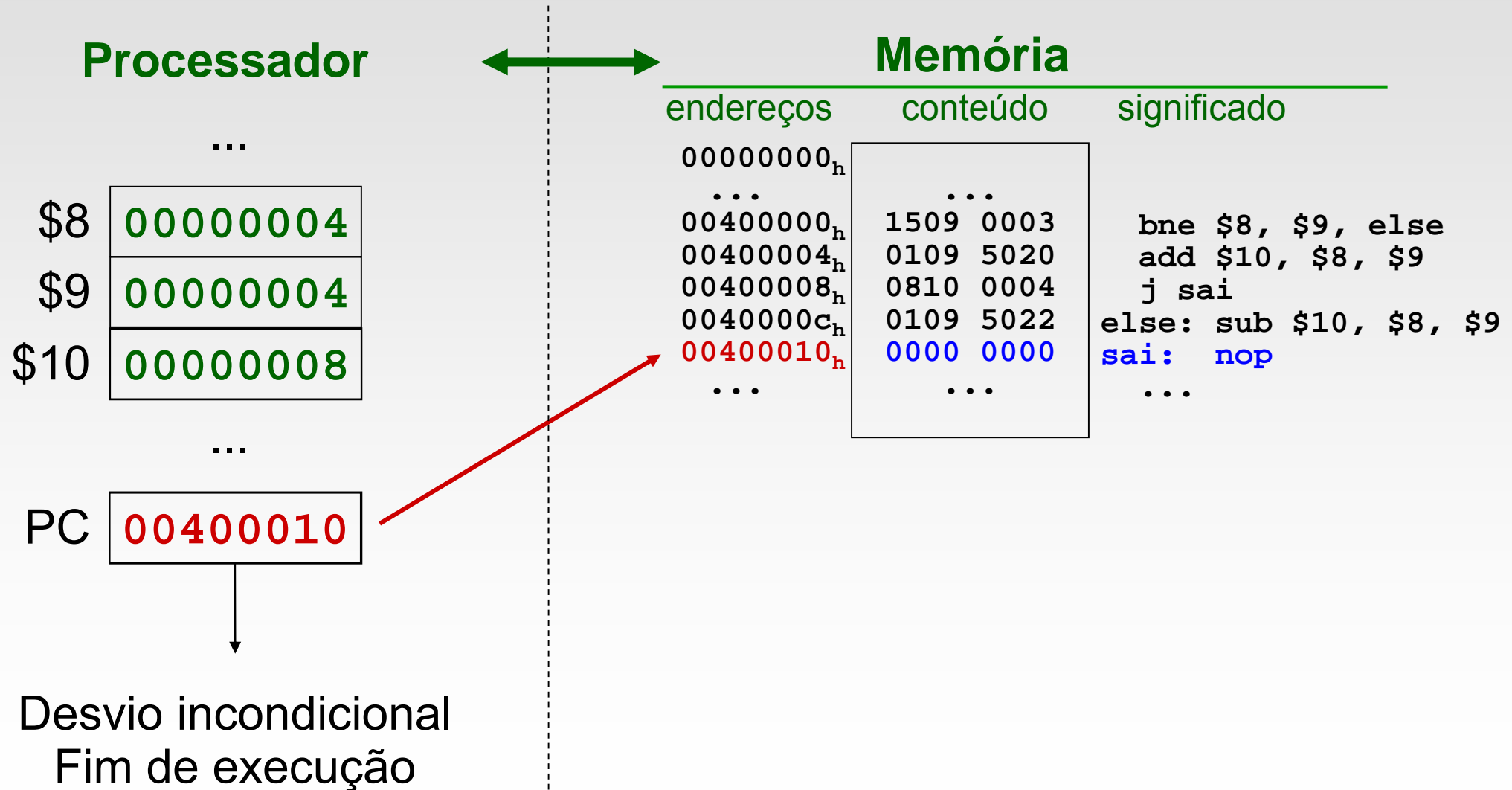
# Instruções MIPS

## :: Instruções de controle :: Ex 02.2



# Instruções MIPS

## :: Instruções de controle :: Ex 02.2

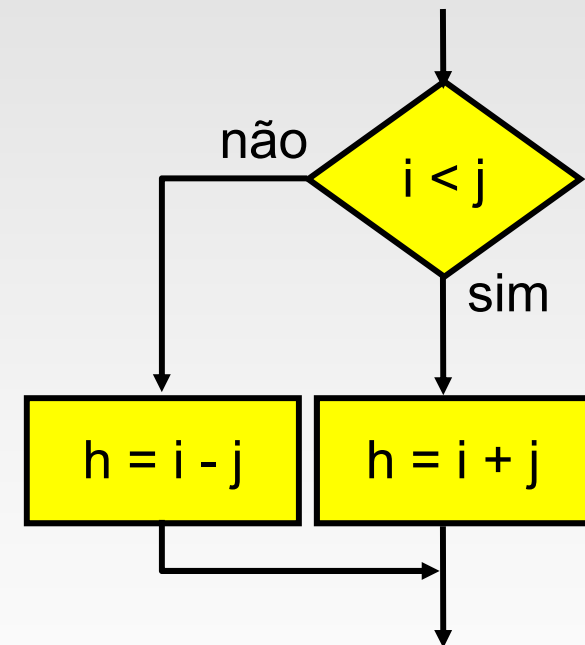


# Instruções MIPS

## :: Instruções de controle :: Ex 3

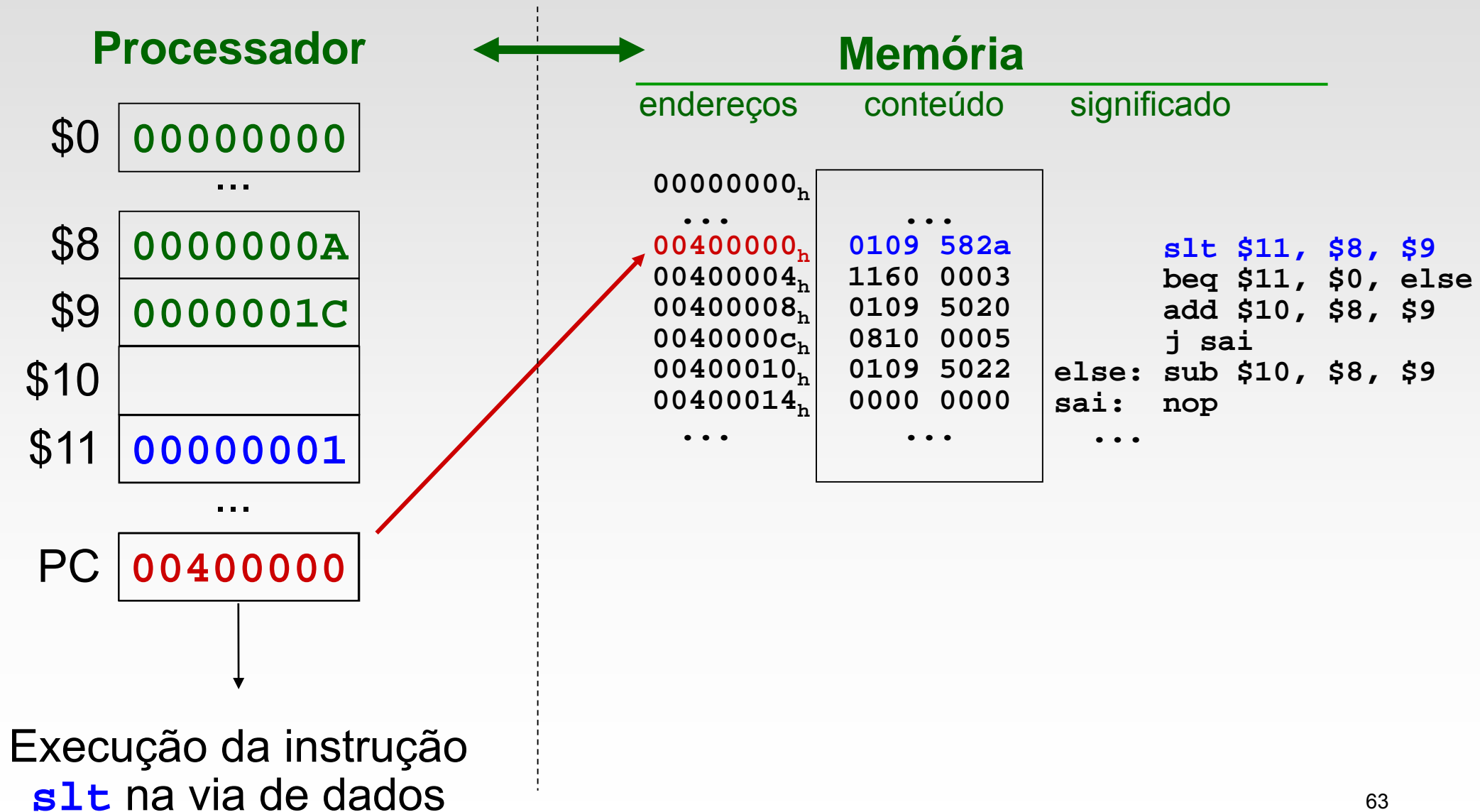
- Exemplo

```
    slt $11, $8, $9
    beq $11, $0, else
    add $10, $8, $9
    j    sai
else:  sub $10, $8, $9
sai:   nop
```



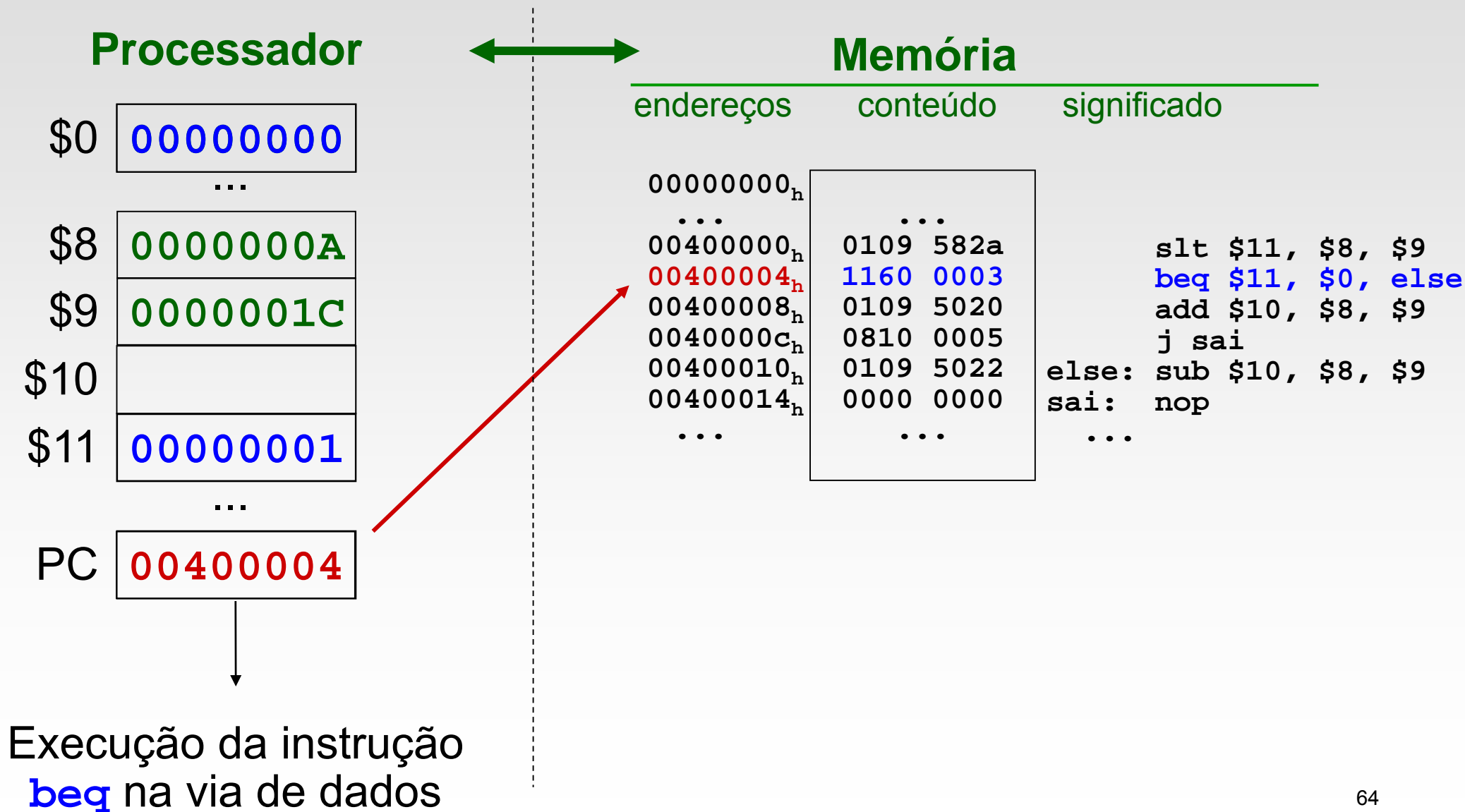
# Instruções MIPS

## :: Instruções de controle :: Ex 03.1



# Instruções MIPS

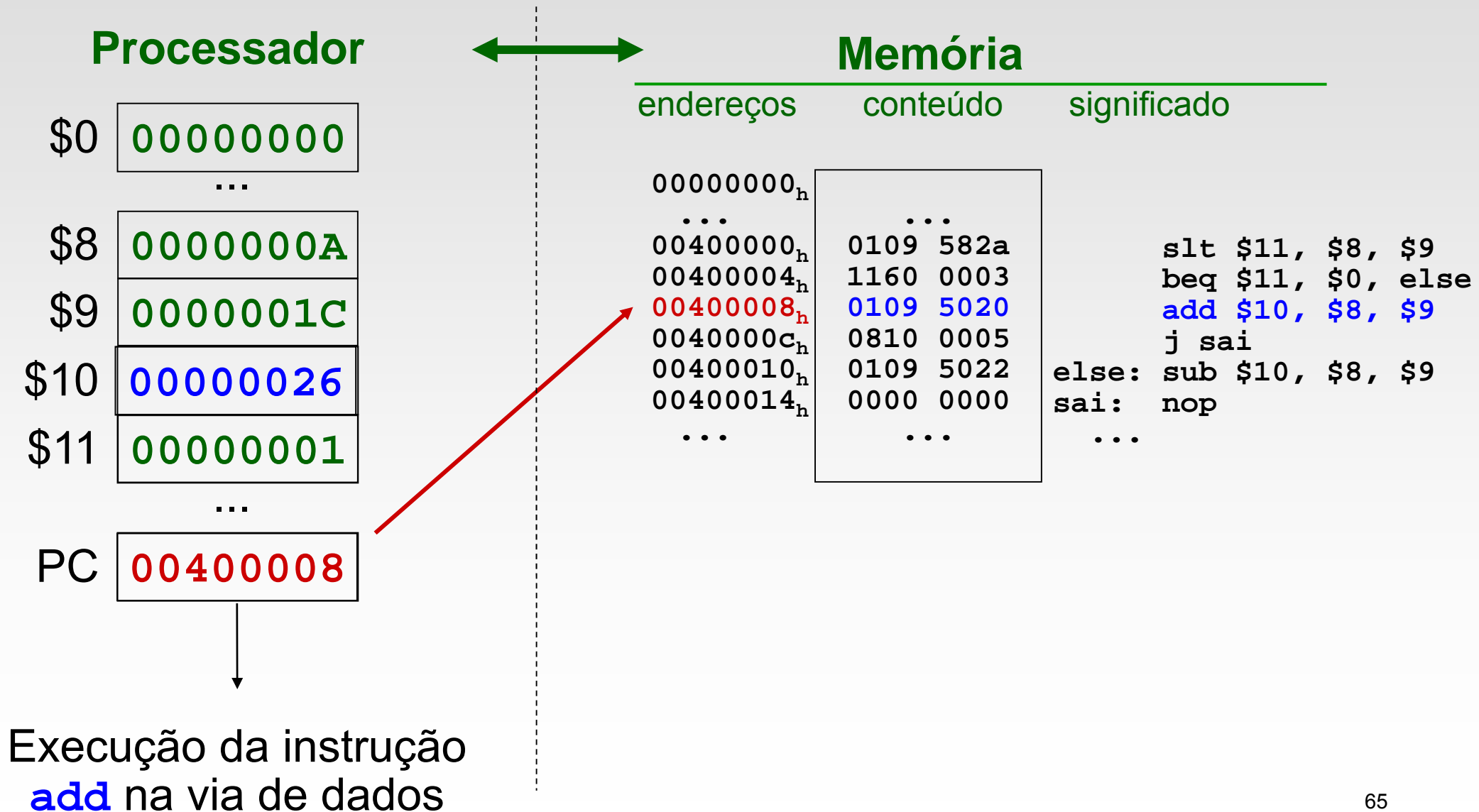
## :: Instruções de controle :: Ex 03.1





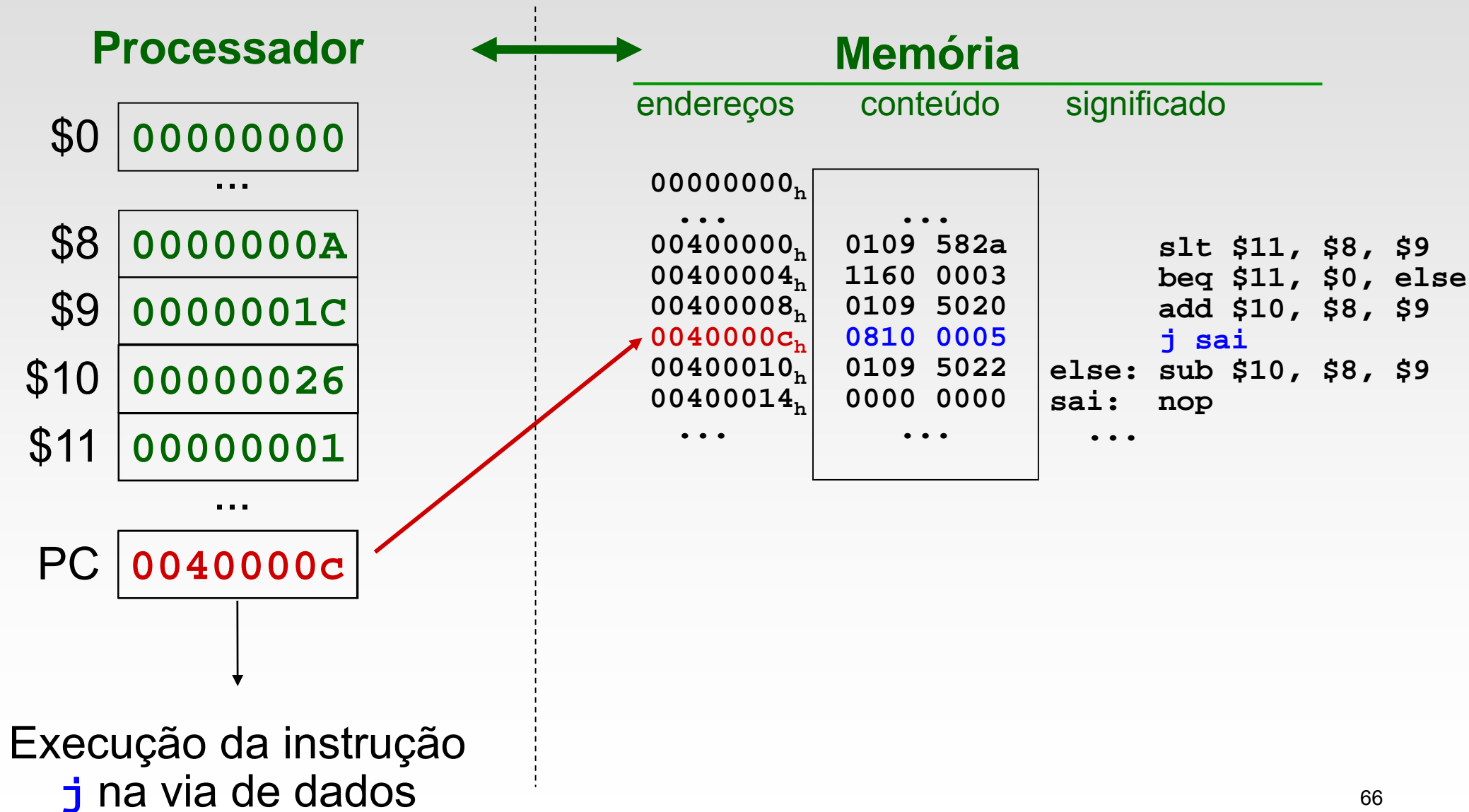
# Instruções MIPS

## :: Instruções de controle :: Ex 03.1



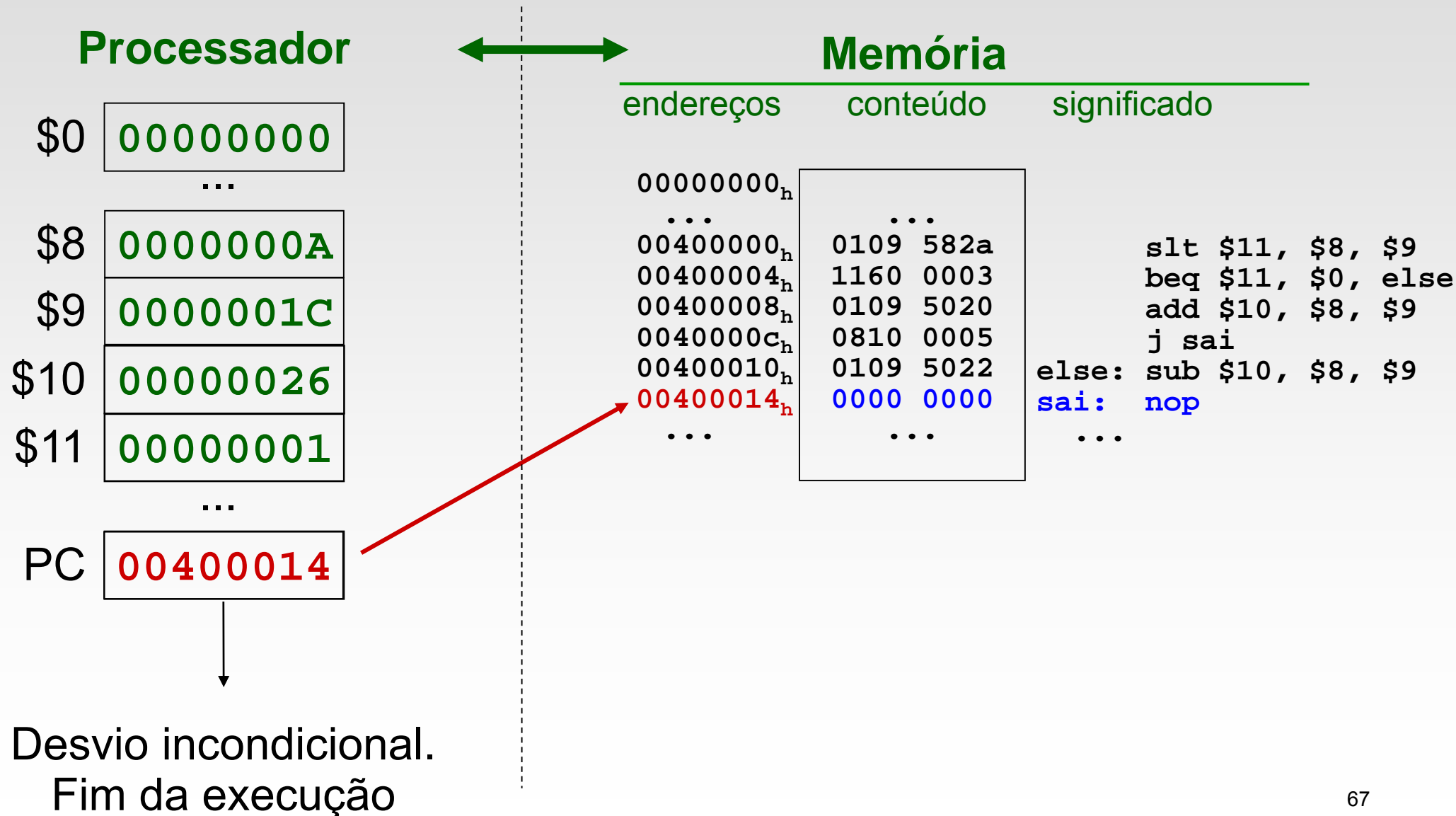
# Instruções MIPS

## :: Instruções de controle :: Ex 03.1



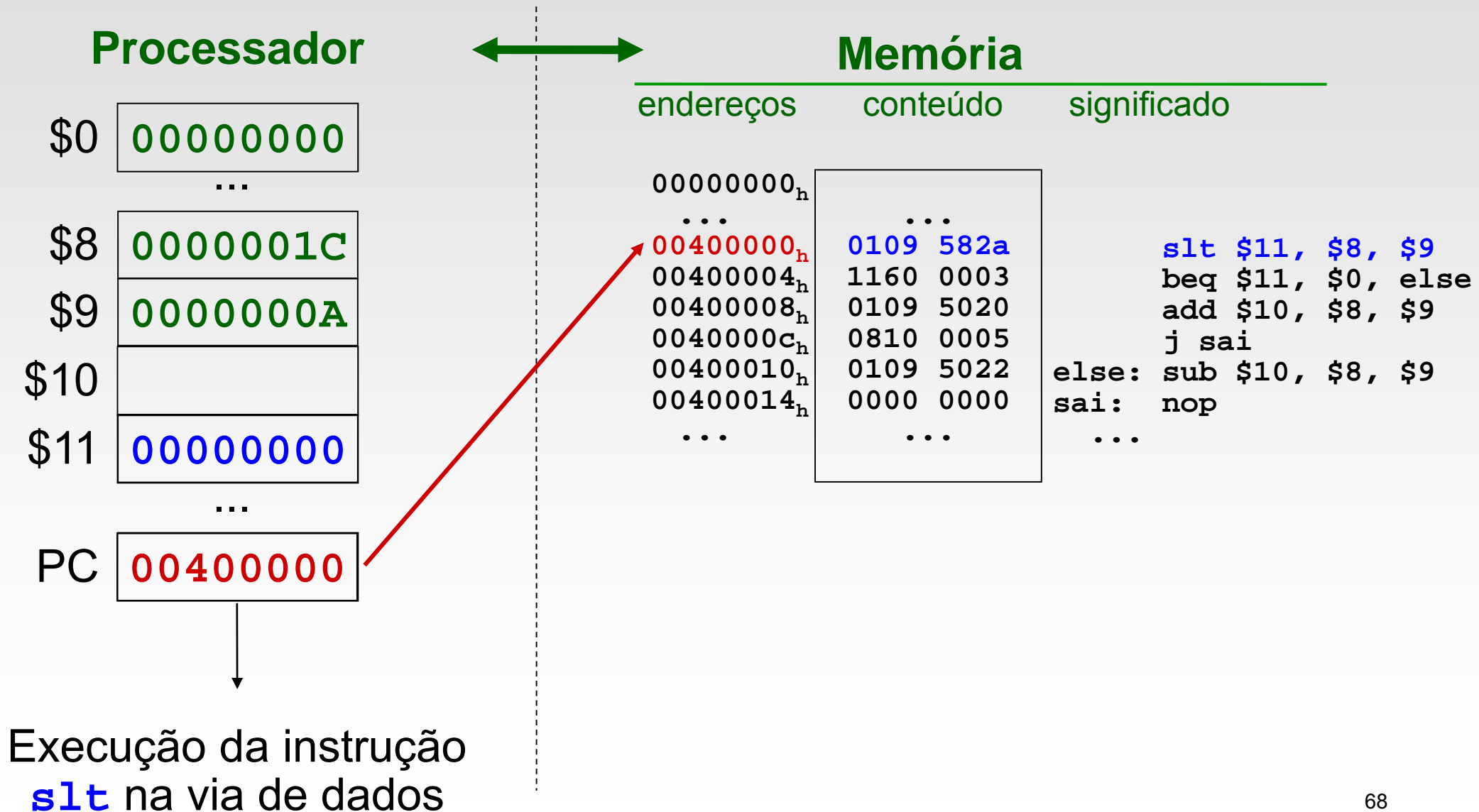
# Instruções MIPS

## :: Instruções de controle :: Ex 03.1



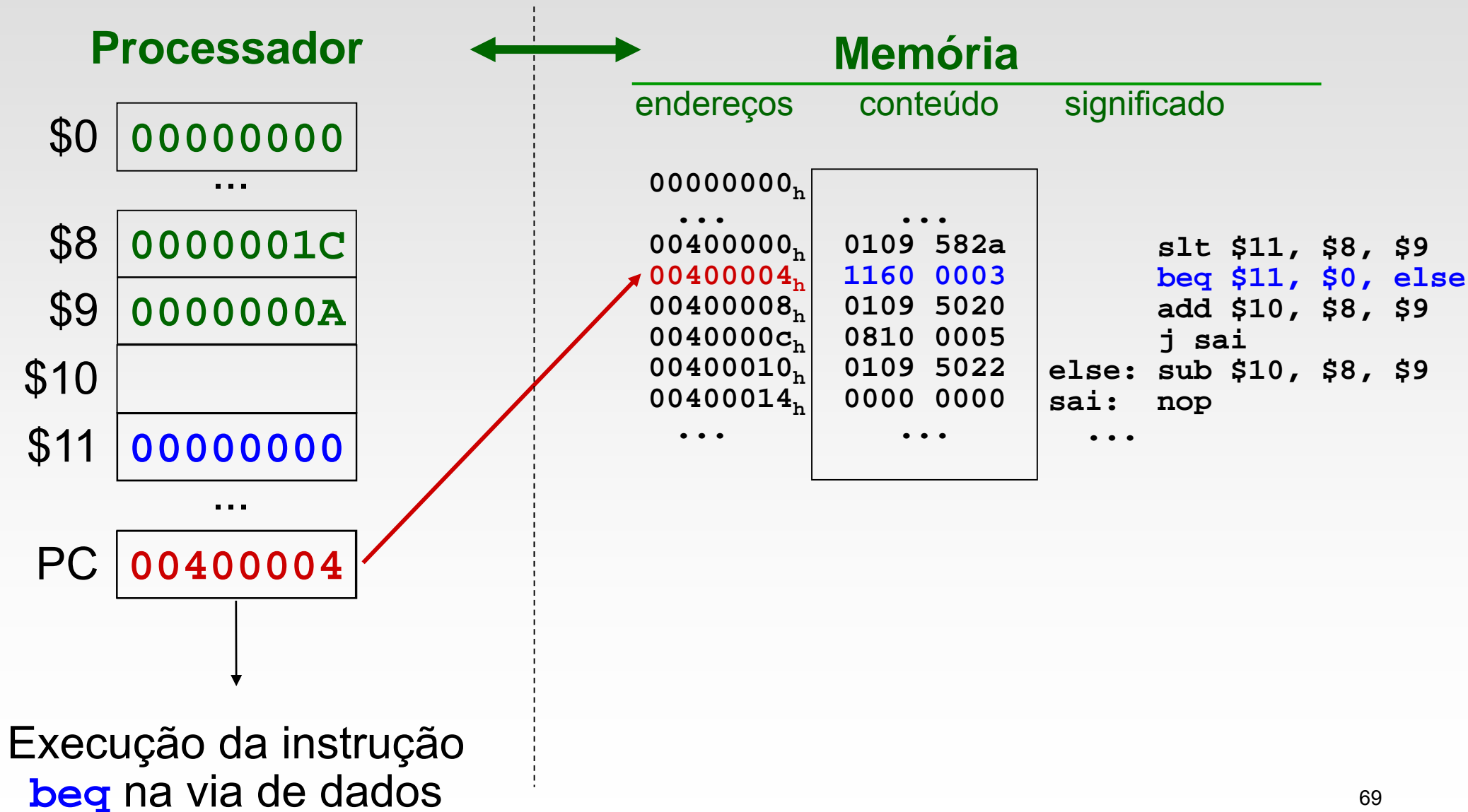
# Instruções MIPS

## :: Instruções de controle :: Ex 03.2



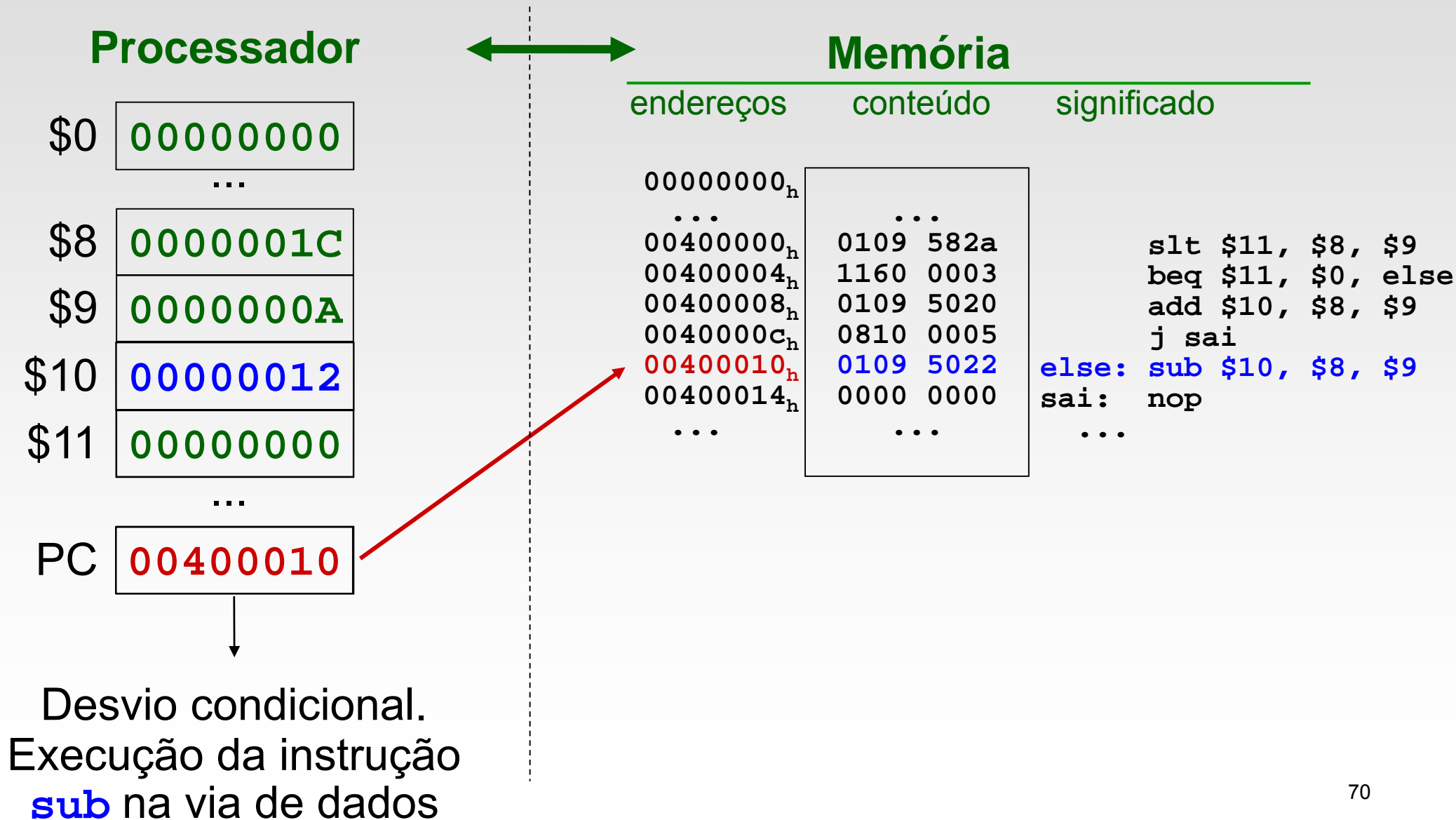
# Instruções MIPS

## :: Instruções de controle :: Ex 03.2



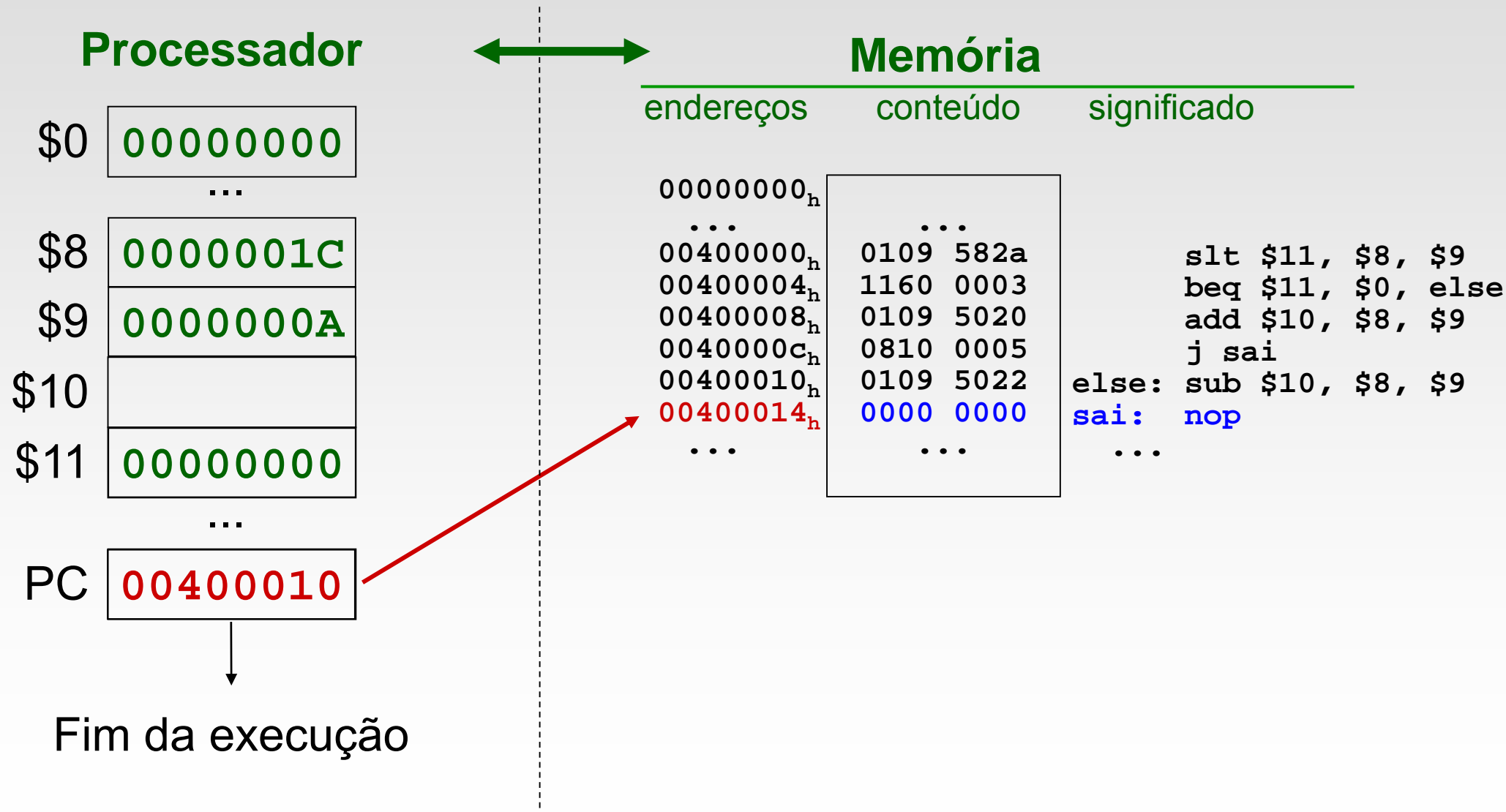
# Instruções MIPS

## :: Instruções de controle :: Ex 03.2



# Instruções MIPS

## :: Instruções de controle :: Ex 03.2



# Instruções MIPS

## :: Instruções de controle :: Formato

- Set on less than (**slt**)

```
slt    $t0, $t1, $t2    #if ($t1 < $t2) $t0 = 1  
                        #else $t0 = 0
```

Instrução (decimal):	op	rs	rt	rd	shamt	funct
	0	9	10	8	0	(2A) <sub>h</sub>
	slt	\$t1	\$t2	\$t0	--	slt

Instrução (binário):	000000	01001	01010	01000	00000	101010
-------------------------	--------	-------	-------	-------	-------	--------



# Instruções MIPS

## :: Instruções de controle :: Formato

- Set on less than immediate (**slti**)

```
slti    $t0, $t1, 100 #if ($t1 < 100) $t0 = 1  
                        #else $t0 = 0
```

Instrução  
(decimal):

op	rs	rt	immediate
(A) <sub>h</sub>	9	8	100
slti	\$t1	\$t0	valor

Instrução  
(binário):

001010	01001	01000	0000000001100100
--------	-------	-------	------------------

# Instruções MIPS

## :: Instruções de controle :: Formato

- **Jump (j)**

**j label      # PC ← endereço[label]**

Instrução  
(decimal):

op	target
<b>(2)<sub>h</sub></b>	<b>xxx</b>

**j**

**endereço da instrução**

Instrução  
(binário):

<b>000010</b>	<b>xxx (26 bits)</b>
---------------	----------------------

# Instruções MIPS

## :: Instruções de controle :: Formato

---

- Ao contrário das instruções de desvio condicional (**beq** e **bne**), o campo target da instrução jump armazena o **endereço da memória** correspondente à instrução marcada pelo label
- Como o endereço de memória é um **múltiplo de 4 bytes** ( $100_b$ ), seus **dois últimos bits não são representados** na instrução jump
- O novo endereço consiste de 4 Bits mais significativos do PC atual e de 26 (+2) Bits da instrução

# Instruções MIPS

## :: Instruções de controle :: Resumo

Categoria	Nome	Exemplo	Operação
Suporte a decisão	bne	bne \$8, \$9, rotulo	se $\$8 \neq \$9$ então $PC \leftarrow \text{endereço}[\text{rotulo}]$
	beq	beq \$8, \$9, rotulo	se $\$8 = \$9$ então $PC \leftarrow \text{endereço}[\text{rotulo}]$
	j	J rotulo	$PC \leftarrow \text{endereço}[\text{rotulo}]$
	slt	slt \$10, \$8, \$9	se $\$8 < \$9$ então $\$10 \leftarrow 1$ senão $\$10 \leftarrow 0$
	slti	slti \$10, \$8, 100	se $\$8 < 100$ então $\$10 \leftarrow 1$ senão $\$10 \leftarrow 0$

# O que você aprenderam hoje?

---

- Instruções lógicas
  - Formas de shift
  - AND, OR, NOR, XOR e versões imediatas
- Processo de instruções sequenciais
- Instruções de controle
  - Condicional, incondicional
  - Formato

# Questões

## Como fazer?

- “while loop”

```
while (save[i] != k) {  
    i += 1;  
}
```

## Código de assembly

*// \$s3 = i, \$s5 = k, \$s6 = base of save[]*

```
Loop: sll $t1, $s3, 2    #t1 = 4*i  
      add $t1, $t1, $s6  #t1 = end. de save[i]  
      lw $t0, 0($t1)     # t0 = valor de save[i]  
      beq $t0, $s5, Exit # if save[i] = k vai p. Exit  
      addi $s3, $s3, 1   # i = i + 1  
  
      j Loop  
  
Exit: nop
```

# Questões

## Como fazer?

- “for loop”

```
for ( $j < k$ ) {  
     $j += 1$ ;  
}
```

## Código de assembly

```
//  $\$s1 = j$ ,  $\$s2 = k$ 
```

```
Loop: slt  $\$t1, \$s1, \$s2$     # $t1 = 1$  if  $j < k$ , else  $t1 = 0$ 
```

```
    beq  $\$t1, \$0, Exit$       # if  $j \geq k$  then Exit
```

```
    addi  $\$s1, \$s1, 1$        #  $j = j + 1$ 
```

```
    j Loop
```

```
Exit: nop
```