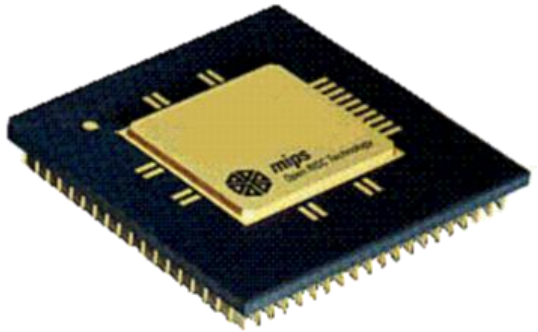
Arquitetura do Processador MIPS

Introdução

- **MIPS = Microprocessor without Interlocked Pipeline Stages**
- **Desenvolvido pela MIPS Technologies**
- **Usado nas estações de trabalho da Silicon Graphics e em sistemas embarcados**
  - TiVO
  - Dispositivos Windows CE
  - Roteadores CISCO
  - Consoles de videogame (Nintendo 64, Sony PlayStation)



Introdução

- **O MIPS é um processador do tipo RISC** (Reduced Instruction-set Processor)
- **Requisitos de processadores RISC**
  - Executar uma operação em um único ciclo de relógio
  - Todas as instruções com mesmo tamanho
  - Acesso à memória apenas por instruções load e store
  - Poucos formatos de instrução
  - Poucas instruções
  - Poucos modos de endereçamento
  - Muitos registradores

Histórico do MIPS

- **1981**: projeto pela equipe do Prof. John Hennessy de Stanford
- **1984**: John Hennessy funda a MIPS Computer Systems
- **1985**: lançado o R2000, 1o modelo comercial de 32 bits
- **1988**: lançado o R3000, 2o modelo comercial
- **1991**: lançado o R4000, 1o modelo de 64 bits
- **1992**: SGI compra a empresa que se torna uma subsidiária denominada MIPS Technologies
- **Início dos 90s**: licenciamento para terceiros
- **1998**: MIPS Technologies torna-se uma empresa separada
- **1999**: sistema de licenciamento formalizado com dois projetos básicos: MIPS32 e MIPS64

Saiba mais em: http://en.wikipedia.org/wiki/MIPS_architecture

Produtos que usam MIPS

- Gravadores de DVD
- Telefones IP
- Telefones sem fio
- Roteadores p/ Internet
- Gateways p/ Internet
- Tocador de MP3
- Periféricos para PC
- Câmeras digitais
- PDAs
- Impressoras
- Copiadoras
- HDTV
- Projetores multimídia
- PlayStation, ...

MIPS products are designed into:
- 72% of VoIP applications
- 76% of cable set-top boxes
- 70% of DVD recorders
- 95% of cable modems

Visão geral da arquitetura do MIPS (de 32 bits)

- Palavra de dados de 32 bits
- Palavra de instrução de 32 bits
- 3 formatos de instrução
- 5 modos de endereçamento
- 32 registradores de uso geral
- Sub-conjunto básico de instruções
  - Aritméticas — add, addi, sub
  - Lógicas — and, or, xor, andi, ori, xori
  - Memória — lw, sw
  - Comparação — slt, slti
  - Desvio condicional — beq, bne
  - Desvio incondicional — j, jal, jr

Revisão de conceitos de arquitetura de processador

- **Arquitetura**
  - Refere-se a atributos que são visíveis ao programador do processador (programação em linguagem de montagem)
- **Organização**
  - Refere-se a atributos que não são visíveis ao programador, sendo foco da atenção do engenheiro de computação (projetista de hardware)

Revisão de conceitos de arquitetura de processador

- **Tamanho da palavra de dados**
  - Número de bits do dado manipulado pelo processador
- **Tipos de dados**
  - Tipos de dados manipulados pelo processador: inteiro (com ou sem sinal), real,...
- **Tamanho da palavra de instrução**
  - Número de bits usados para representar uma instrução de programa
- **Formato das instruções**
  - Estrutura utilizada para organização das instruções
  - Largura (em bits) do código da operação (OpCode)
  - Número de operandos
  - Largura dos operandos (em bits)

Revisão de conceitos de arquitetura de processador

- **Modos de endereçamento**
  - Métodos de acesso aos dados processados pelas instruções
- **Registradores**
  - Unidades de armazenamento internas da CPU
  - Registradores de uso específico
  - Registradores de uso geral
- **Conjunto de instruções**
  - Vocabulário de instruções
  - Códigos das operações das instruções

Visão geral da arquitetura do MIPS (de 32 bits)

- Palavra de dados de 32 bits
- Palavra de instrução de 32 bits
- 32 registradores de uso geral
- 3 formatos de instrução
- 5 modos de endereçamento
- Sub-conjunto básico de instruções
  - Aritméticas — add, addi, sub
  - Lógicas — and, or, xor, andi, ori, xori
  - Memória — lw, sw
  - Comparação — slt, slti
  - Desvio condicional — beq, bne
  - Desvio incondicional — j, jal, jr

Operandos usados nas instruções

- **Imediato**
  - Constante de dado
  - Deslocamento de endereço da memória de dados (para acesso a variáveis)
  - Deslocamento de endereço da memória de instruções (para desvios condicionais ou incondicionais)
- **Registrador**
  - Dado
  - Endereço base da memória de dados
  - Endereço da memória de instruções
- **O significado depende da classe da instrução**

Registradores

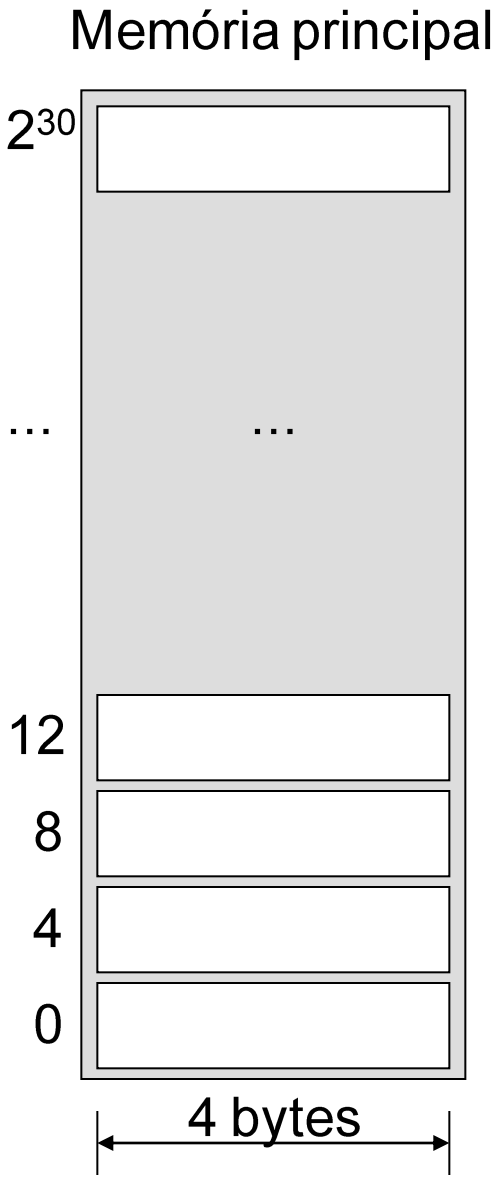
- **32 registradores**
  - Identificados pelo nome ou pelo endereço (de 5 bits)

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

Memória

- **A memória do MIPS é endereçada em nível de byte**
- **Palavras (words) de 32 bits consecutivas diferem de 4 unidades de endereço**
- **A memória principal do MIPS pode ter até**
  - 2^{32} Bytes = 4 Gbytes

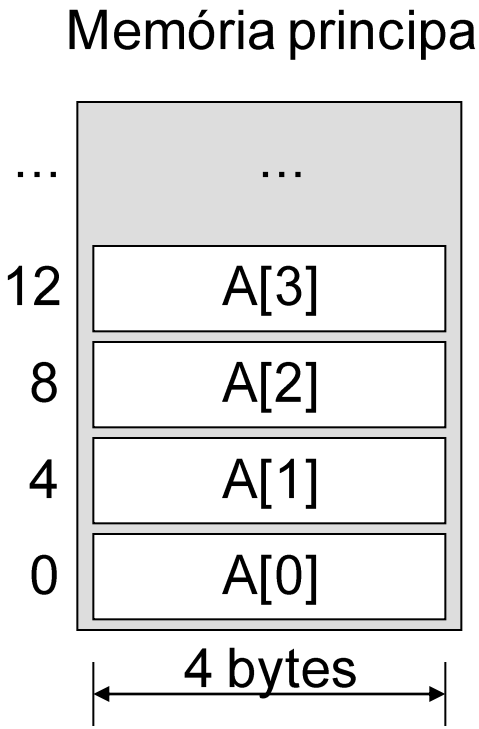
  ou
  - 2^{30} Words = 1 Gwords



Memória

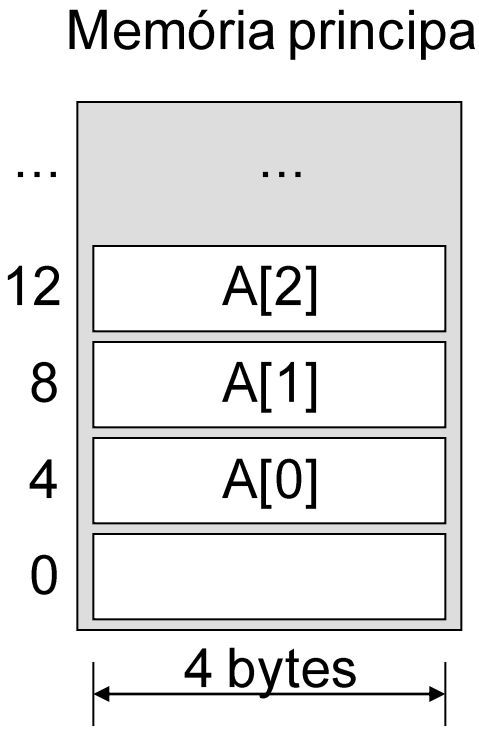
- **A memória principal pode ser vista como um grande array unidimensional (vetor) onde cada elemento do vetor ocupa uma palavra de 32 bits, ou seja, quatro posições de 1 byte**

- **Exemplo:**
  - Se o elemento A[0] é armazenado na posição **0**,
  - O elemento A[1] está armazenado na posição **0+1x4 = 4**
  - O elemento A[2] está armazenado na posição **0+2x4 = 8**

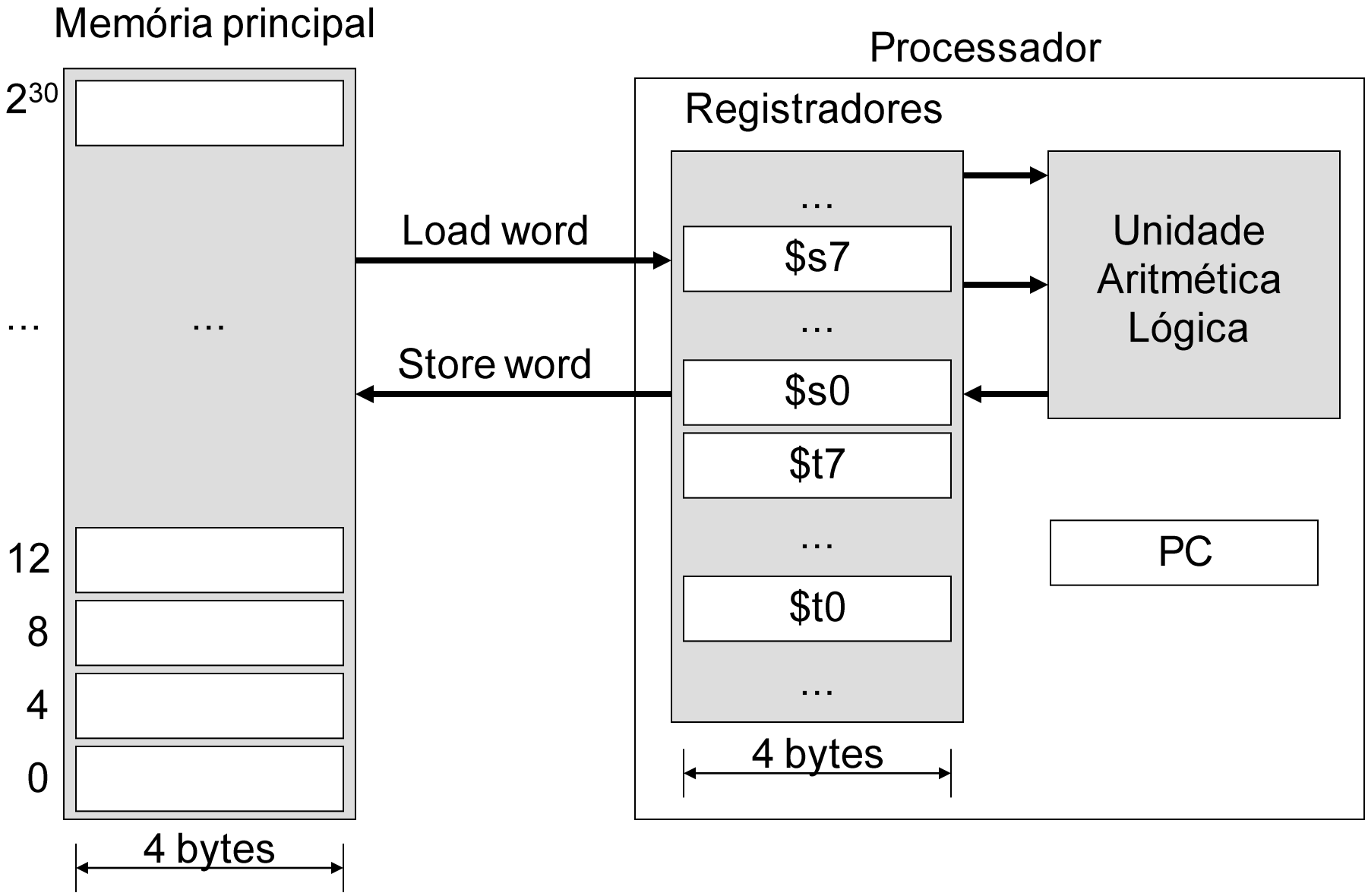


- **Exemplo:**
  - Se o elemento A[0] é armazenado na posição **4**,
  - O elemento A[1] está armazenado na posição **4+1x4 = 8**
  - O elemento A[2] está armazenado na posição **4+2x4 = 12**

A posição do elemento A[0] é considerada o endereço-base do vetor A

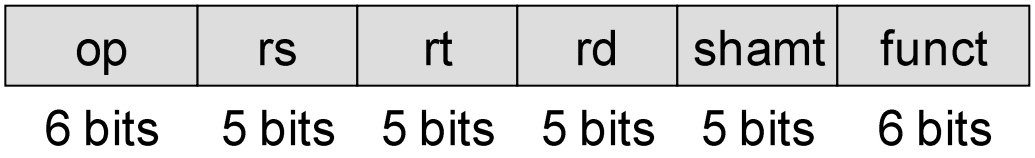


Visão simplificada do sistema “memória + CPU”

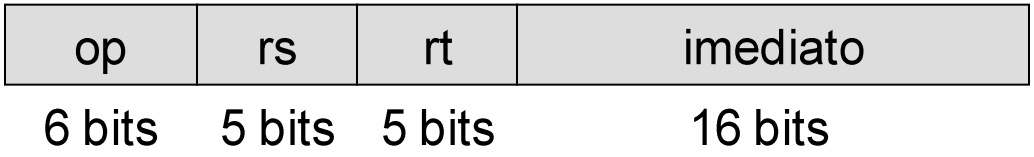


Formatos de instrução

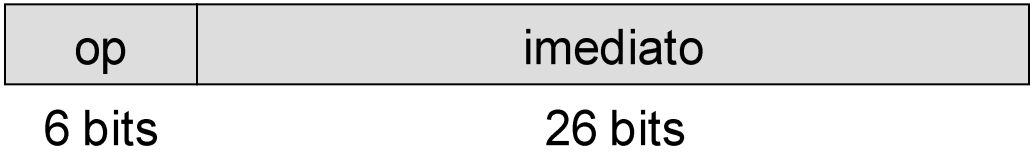
Formato R (registrador)



Formato I (imediato)



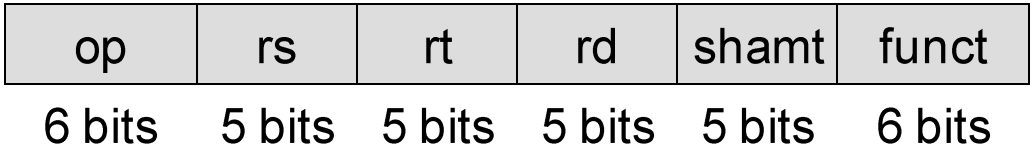
Formato J (*jump*)



Conjunto de instruções básico

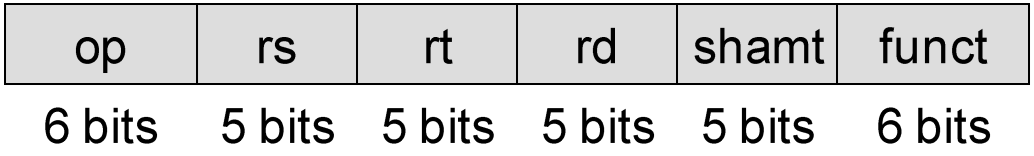
Aritméticas	add, addi, sub
Lógicas	and, or, xor, andi, ori, xori nor, sll, srl
Memória	lw, sw
Comparação	slt, slti
Desvio condicional	beq, bne
Desvio incondicional	j, jal, jr

Conjunto de instruções básico: Formato R



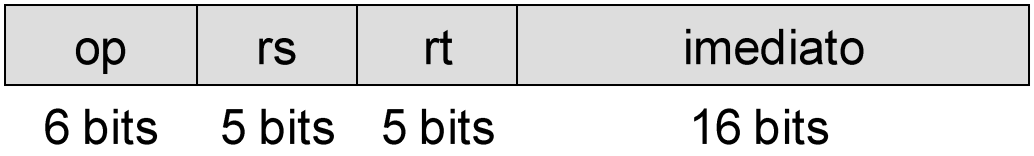
- `add  rd, rs, rt    # rd <- rs + rt`
- `sub  rd, rs, rt    # rd <- rs - rt`
- `and  rd, rs, rt    # rd <- rs & rt`
- `or   rd, rs, rt    # rd <- rs | rt`
- `xor  rd, rs, rt    # rd <- rs ^ rt`
- `nor rd, rs, rt     # rd <- ~(rs | rt)`

Conjunto de instruções básico: Formato R



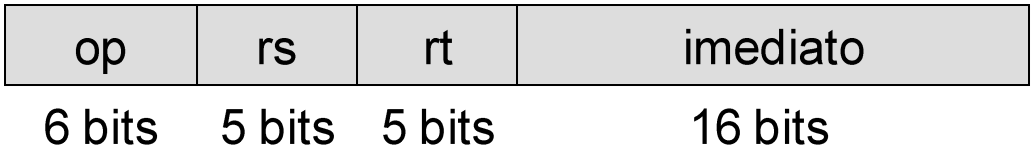
- `sll rd, rs, shamt # rd <- rt << shamt`
- `srl rd, rs, shamt # rd <- rt >> shamt`
- `slt rd, rs, rt # rd <- 1, se rs < rt`
  `# 0, se não`
- `jr rs # PC <- rs`

Conjunto de instruções básico: Formato I



- `addi  rt, rs, imed # rt <- rs + imed`
- `andi  rt, rs, imed # rt <- rs & imed`
- `ori   rt, rs, imed # rt <- rs | imed`
- `xori  rt, rs, imed # rt <- rs ^ imed`
- `slti  rt, rs, imed # rt <- 1,se rs<imed`
  `#          0,se não`

Conjunto de instruções básico: Formato I



- `lw  rt, imed (rs)  # rt <- Mem[imed+rs]`
- `sw  rt, imed (rs)  # Mem[imed+rs] <- rt`
- `beq rs, rt, imed  # Se (rs = rt) então`
  `# PC <- PC+(imed<<2)`
- `bne rs, rt, imed  # Se (rs != rt) então`
  `# PC <- PC+(imed<<2)`

Conjunto de instruções básico: Formato J

6 bits 26 bits

- `j    imed   # PC  <- PC[31..28]:(imed<<2)`
- `jal  imed   # PC  <- PC[31..28]:(imed<<2)`
  `# $ra <- PC ou $ra <- PC+4`

Códigos de operação (campo *op*)

	000	001	010	011	100	101	110	111
000	Formato R		j	jal	beq	bne		
001	addi		slti		andi	ori	xori	
010								
011								
100				lw				
101				sw				
110								
111								

Funções do formato R (campo *function*)

	000	001	010	011	100	101	110	111
000	sll		srl					
001	jr							
010								
011								
100	add		sub		and	or	xor	nor
101			slt					
110								
111								

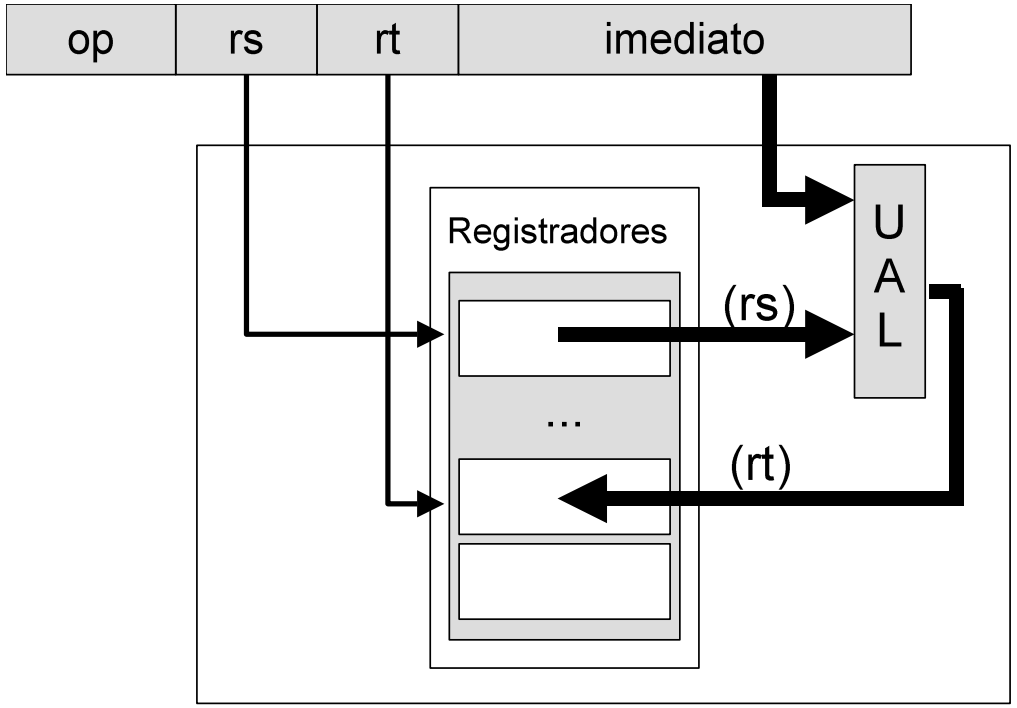
Modos de endereçamento

- **O modo de endereçamento especifica como é feito o acesso aos operandos da instrução**

- **No MIPS, existem cinco modos de endereçamento**
  - Imediato
  - Via registrador
  - Via endereço-base
  - Relativo ao PC
  - Pseudo-direto

Modos de endereçamento

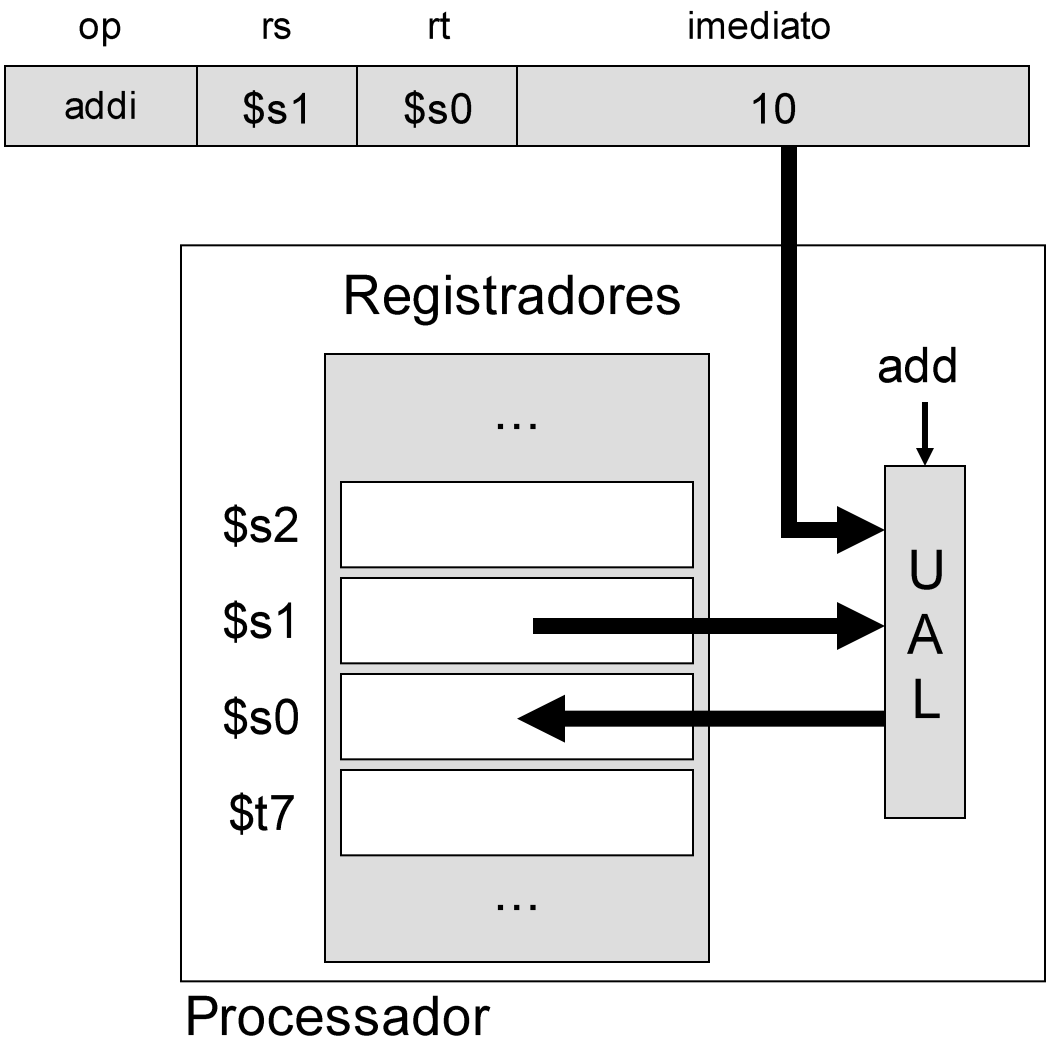
- **Endereçamento imediato**
  - Um dos operandos é uma constante (imediato) incluída na instrução e os outros dois são registradores (fonte e destino)
  - Usado na instrução `addi`



Modos de endereçamento

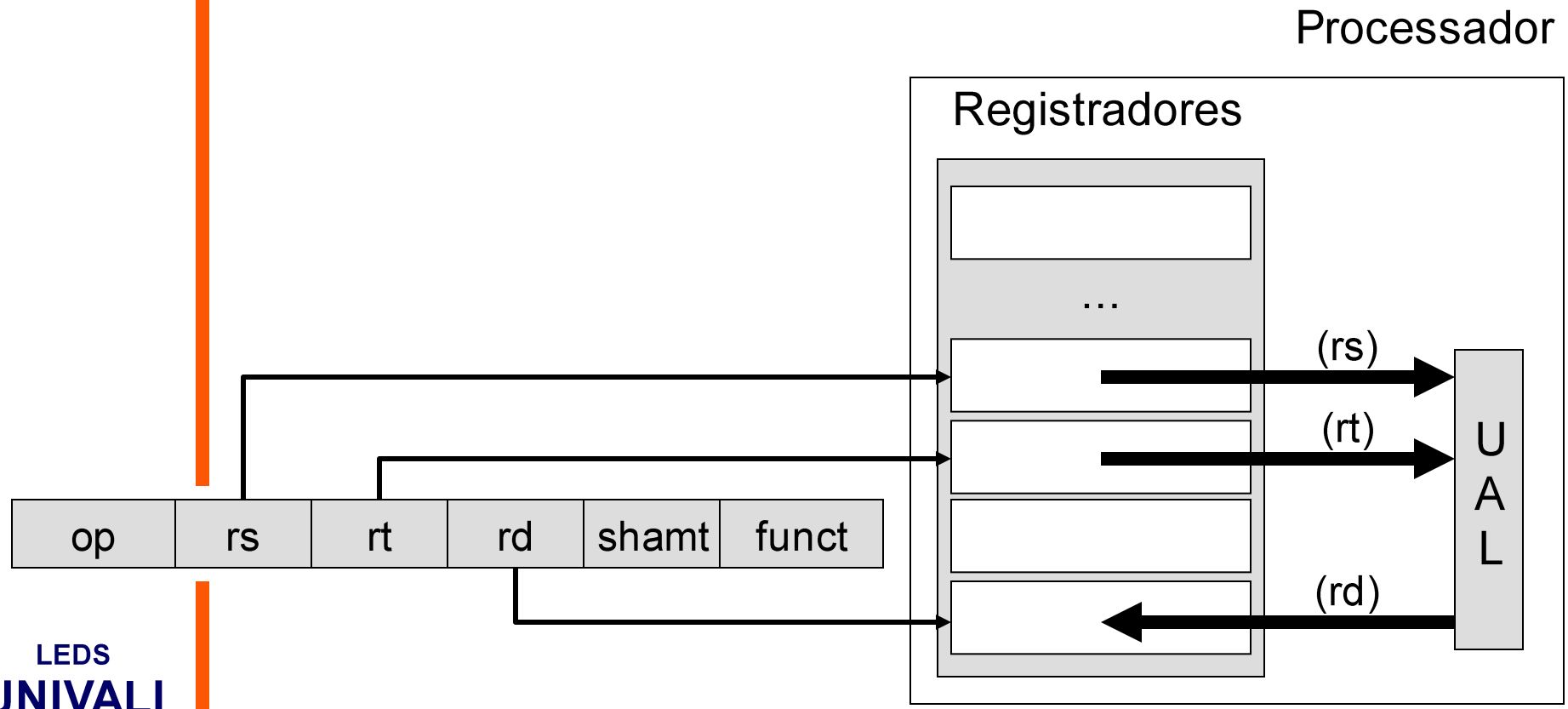
- **Endereçamento imediato – exemplo**

```
addi $s0, $s1, 10
```



Modos de endereçamento

- **Endereçamento via registrador**
  - Todos os operandos da instrução estão armazenados em registradores são registradores apontados pelos campos *rs*, *rt* e *rd*
  - Usado nas instruções `add` e `sub`

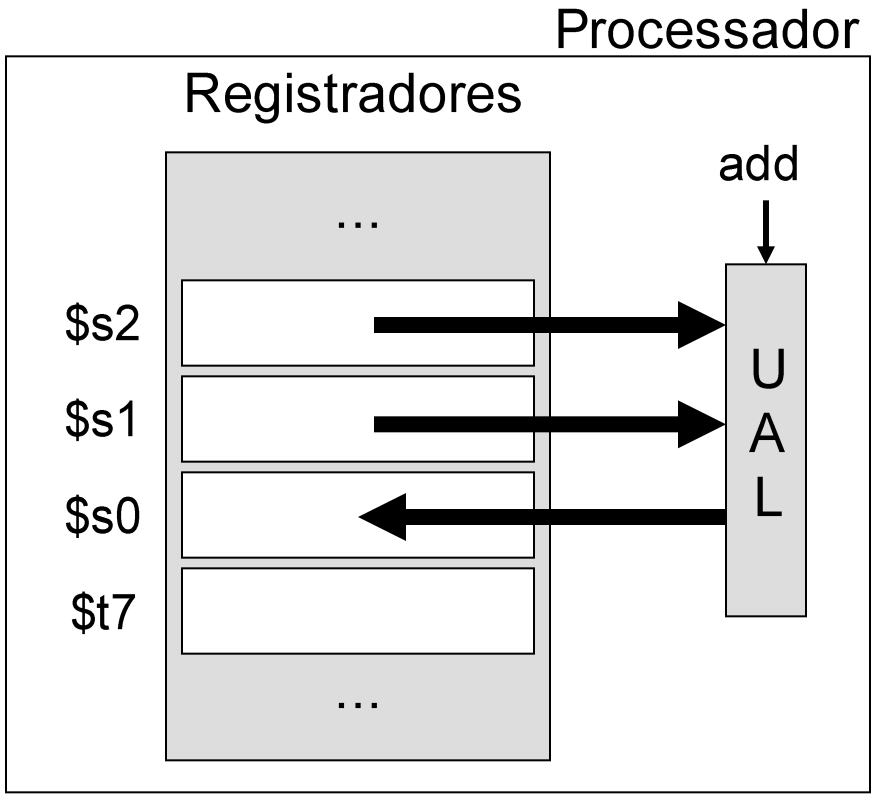


Modos de endereçamento

- **Endereçamento via registrador – exemplo**

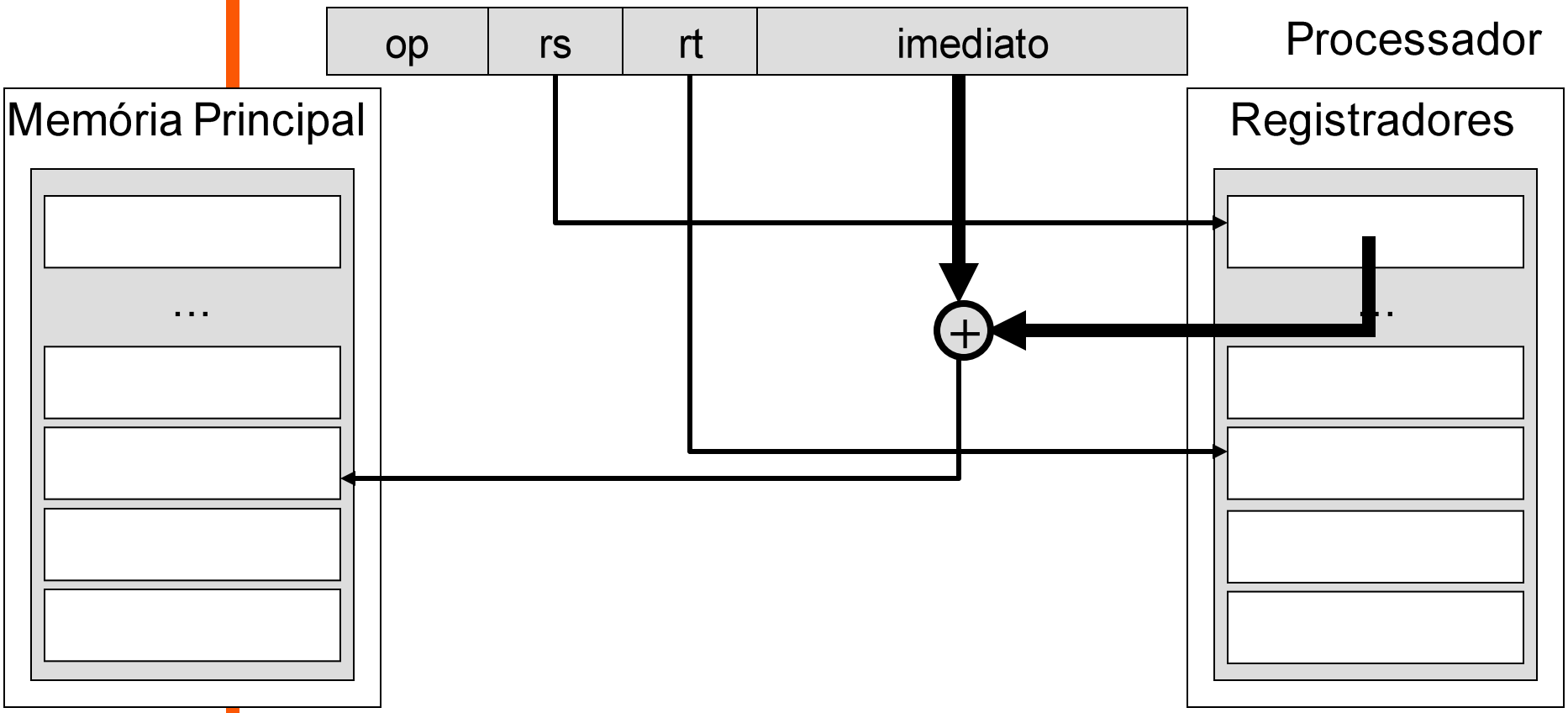
```
add $s0, $s1, $s2
```

op	rs	rt	rd	shamt	funct
formato R	$s1	$s2	$s0	0	add



Modos de endereçamento

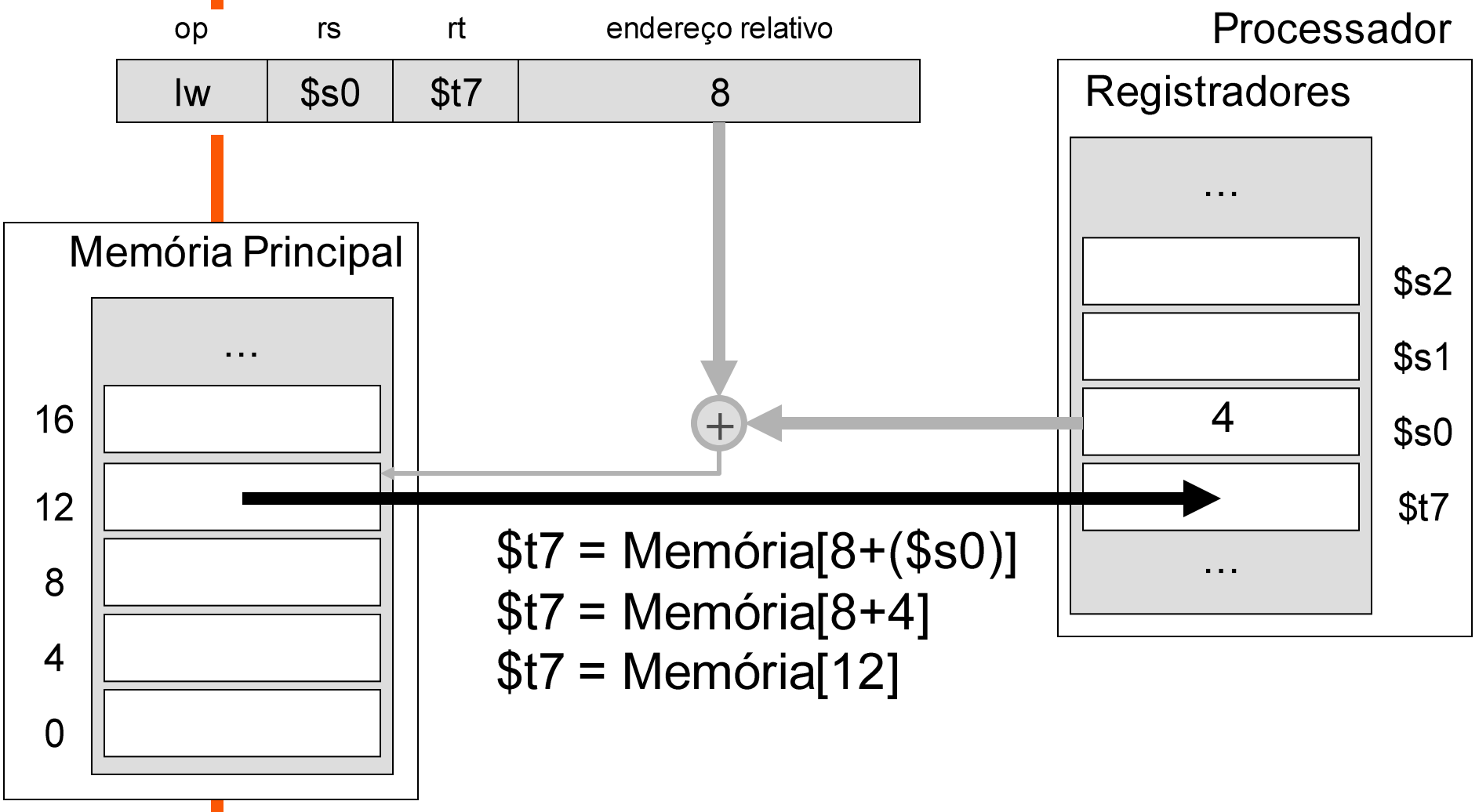
- **Endereçamento via registrador-base**
  - Um dos operandos está armazenado na memória e seu endereço é indicado pela soma do campo imediato ao conteúdo de um registrador-base (campo rs)
  - Usado nas instruções lw e sw



Modos de endereçamento

- **Endereçamento via registrador-base – exemplo 1**

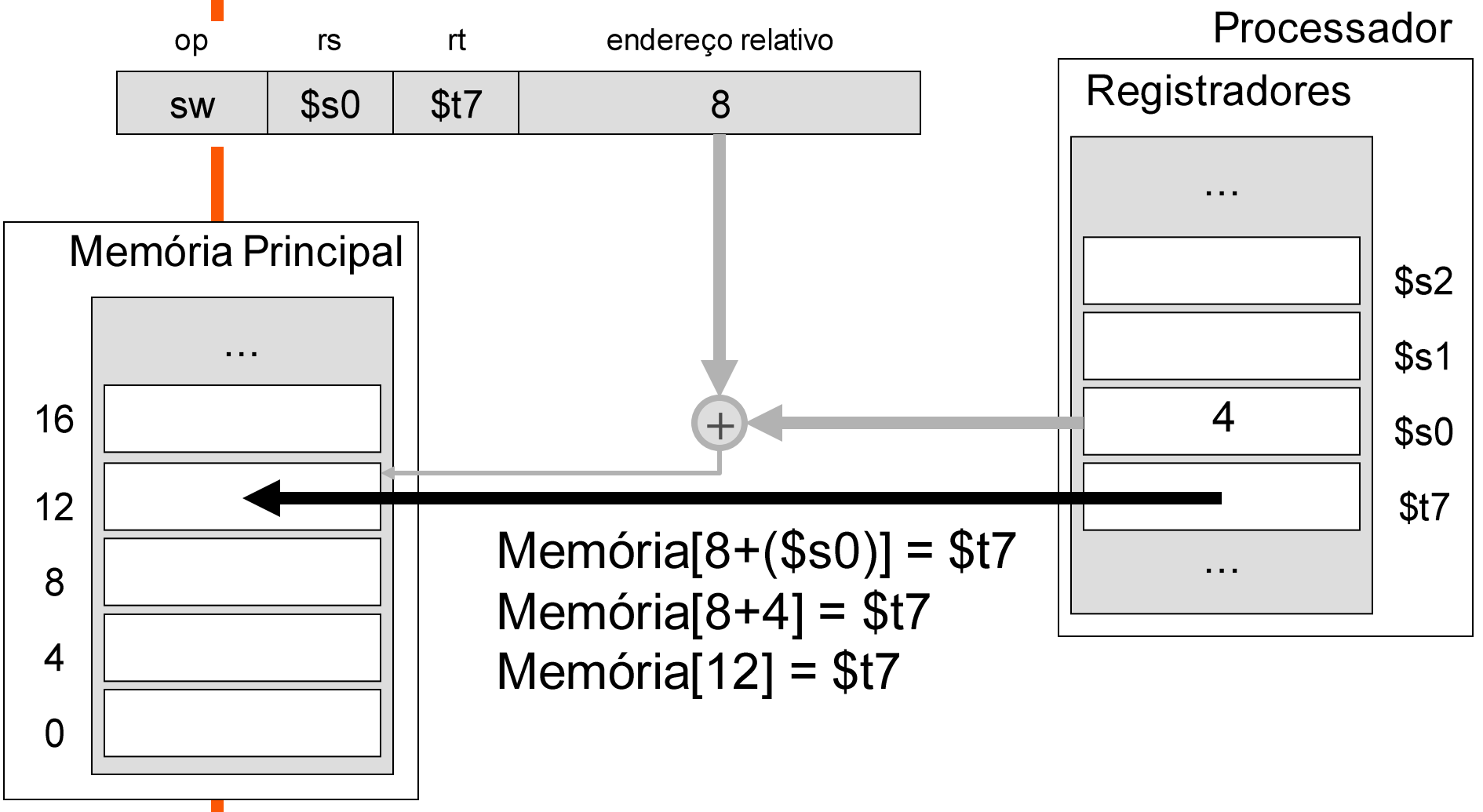
```
lw $t7, 8($s0)
```



Modos de endereçamento

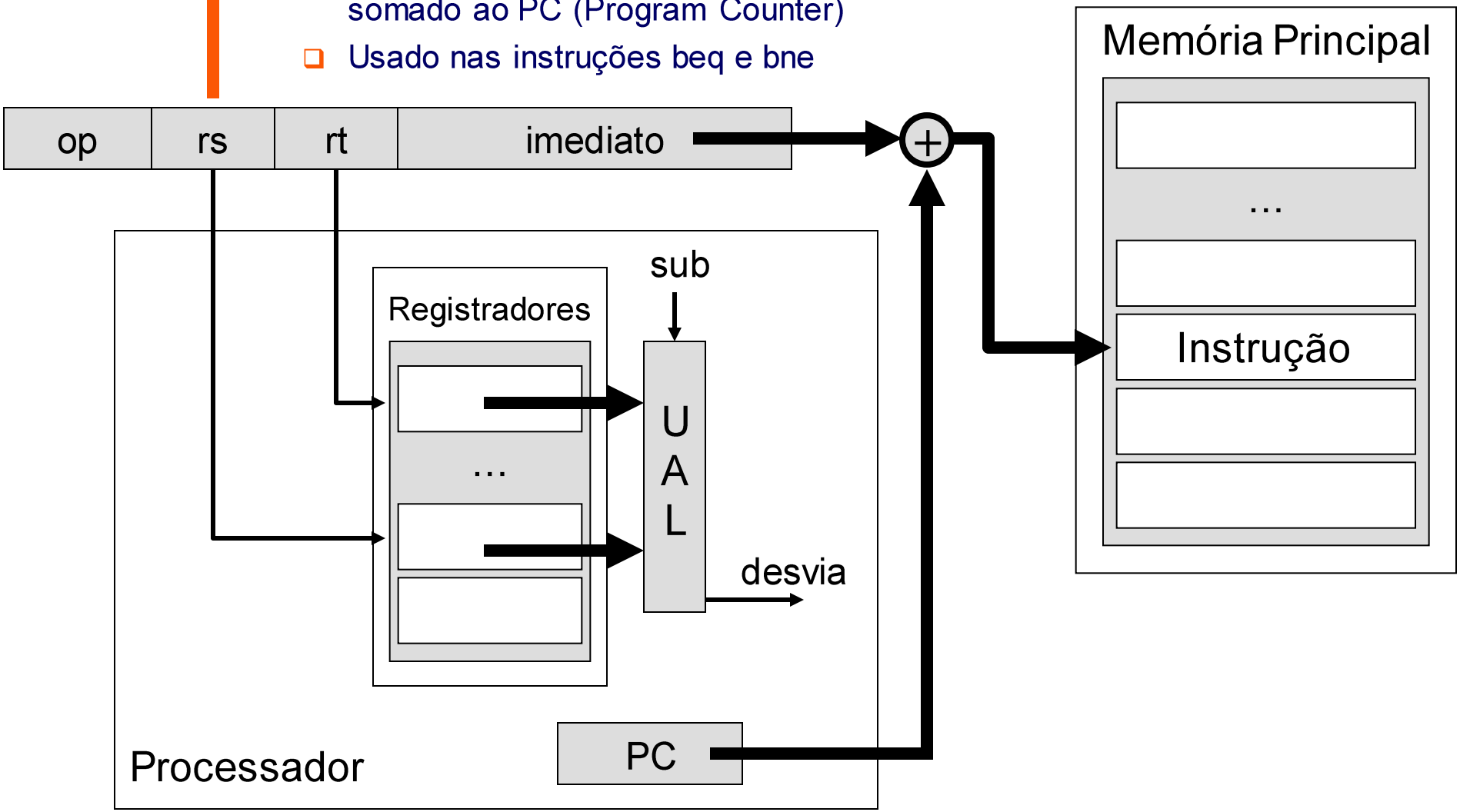
- **Endereçamento via registrador-base – exemplo 2**

```
sw $t7, 8($s0)
```



Modos de endereçamento

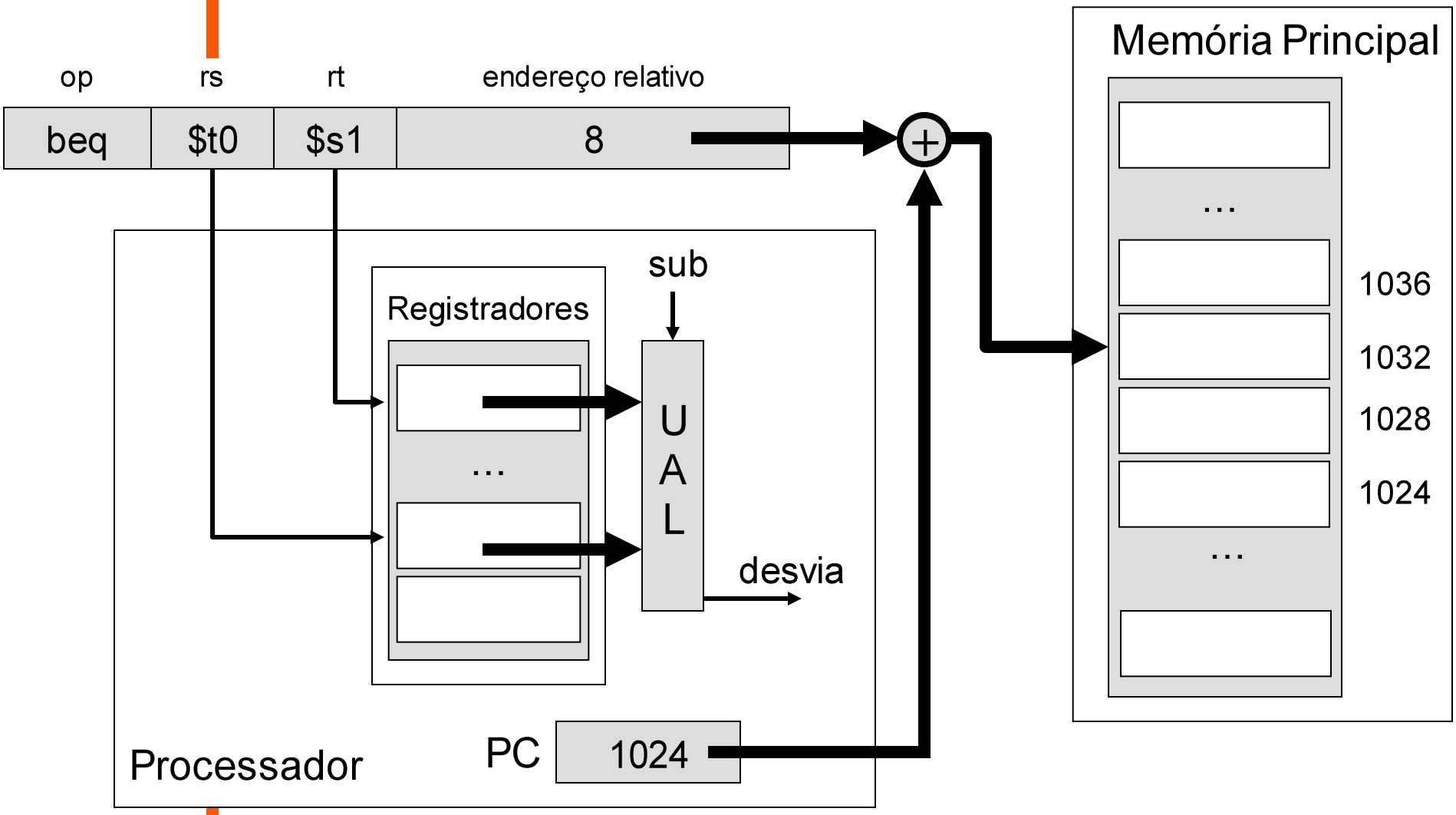
- **Endereçamento relativo ao PC**
  - O operando imediato (endereço-relativo) é somado ao PC (Program Counter)
  - Usado nas instruções beq e bne



Modos de endereçamento

- **Endereçamento relativo ao PC – exemplo 1**

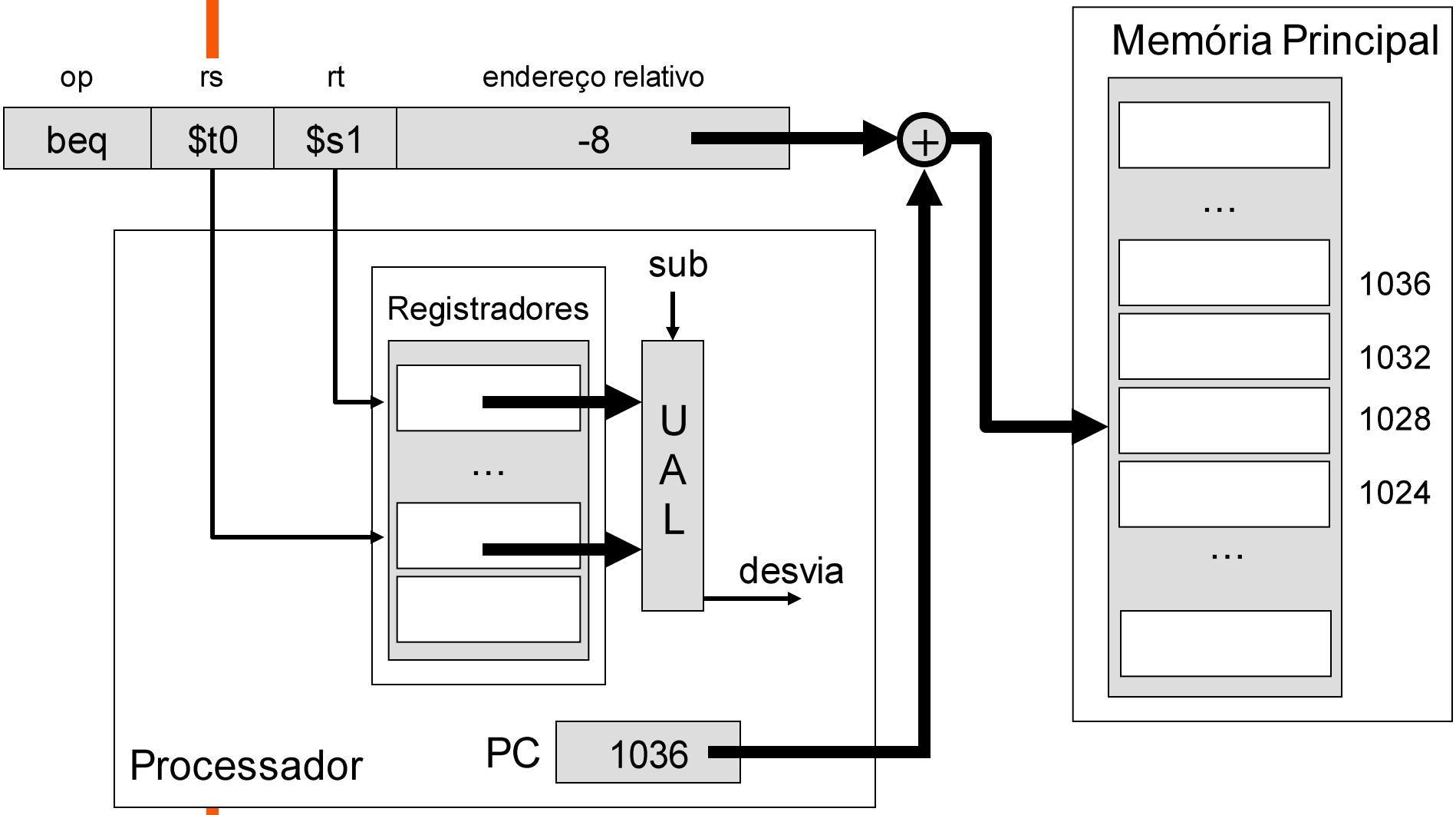
```
beq $t0, $s1, 8
```



Modos de endereçamento

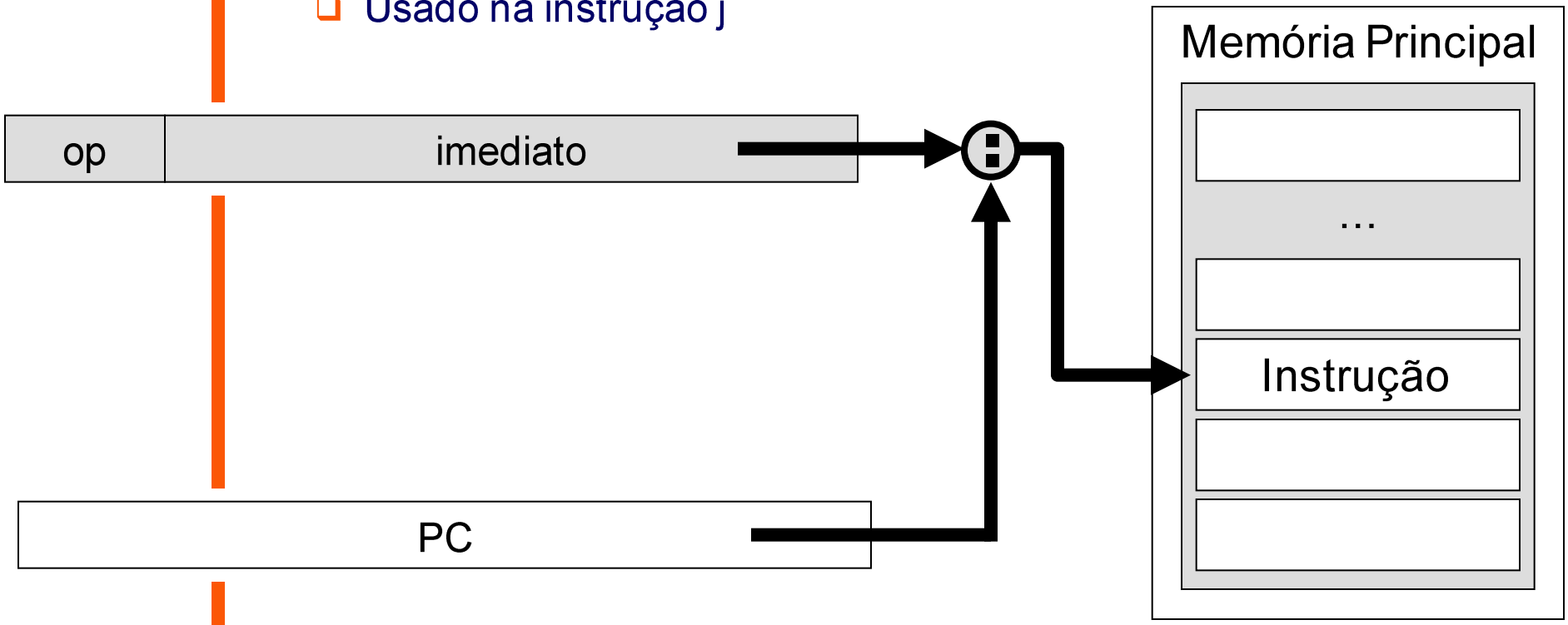
- **Endereçamento relativo ao PC – exemplo 2**

```
bne $t0, $s1, -8
```



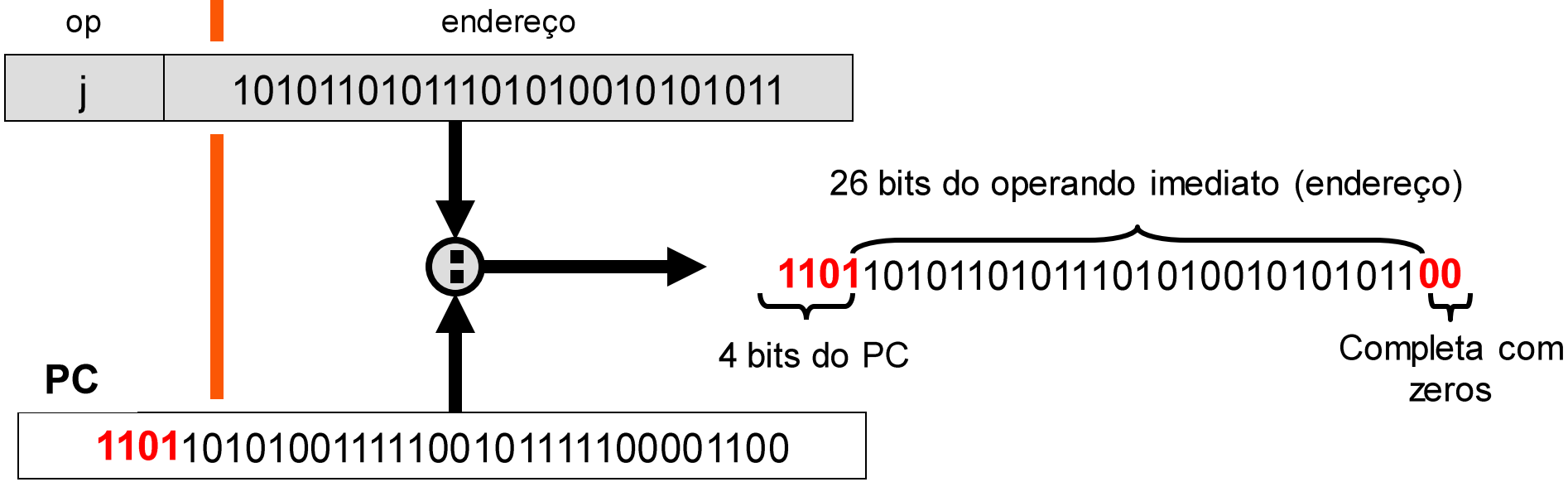
Modos de endereçamento

- **Endereçamento pseudodireto**
  - O operando imediato (endereço) é concatenado com os 4 bits mais significativos do PC
  - Usado na instrução j



Modos de endereçamento

- **Endereçamento pseudodireto - exemplo**
  - O operando imediato (endereço) é concatenado com os 4 bits mais significativos do PC
  - Usado na instrução j



Caracterização do conjunto de Instruções

Aritmética

Instrução	Formato	Modo de endereçamento
add	R	Via registrador
sub	R	Via registrador
addi	I	Imediato

Transferência

Instrução	Formato	Modo de endereçamento
lw	I	Via registrador-base
sw	I	Via registrador-base

Caracterização do conjunto de Instruções

Lógica

Instrução	Formato	Modo de endereçamento
sll	R	Via registrador
srl	R	Via registrador
and	R	Via registrador
andi	I	Imediato
or	R	Via registrador
ori	I	Imediato
xor	R	Via registrador
xori	I	Imediato
nor	R	Via registrador

Caracterização do conjunto de Instruções

Desvio

Instrução	Formato	Modo de endereçamento
beq	I	Relativo ao PC
bne	I	Relativo ao PC
j	J	Pseudodireto
jal	J	Pseudodireto
jr	R	

Comparação

Instrução	Formato	Modo de endereçamento
slt	R	Via registrador
slti	I	Imediato

Pseudo-instruções

- **Instruções que o montador converte em instruções do processador**
- **Pseudo-instruções de desvio condicional**
  - Desvia se igual a zero `beqz rsrc,label`
  - Desvia se maior que `bgt rsrc1,rsrc2,label`
  - Desvia se maior ou igual que `bge rsrc1,rsrc2,label`
  - Desvia se menor que `blt rsrc1,rsrc2,label`
  - Desvia se menor ou igual que `ble rsrc1,rsrc2,label`
  - Desvia se diferente de zero `bnez rsrc,label`
- **Outras pseudo-instruções**
  - Inversão `not rdest, rsrc`
  - Carga de vetor `la rdest, vetor`