MODOS DE ENDEREÇAMENTO NO MIPS E ENDEREÇOS DE 32 BITS

Arquitetura e Organização de Computadores I

Docente responsável: Dr. Leonardo Bidese de Pinho.
Mestrando: Luciano Moraes da Luz Brum.
Email: lucianobrum18@gmail.com

Roteiro

- Constantes.
- Endereços em desvios condicionais e incondicionais.
- Modos de endereçamento.
- Resumo e revisão geral das instruções do MIPS.
- Exercícios.

Constantes

- Constantes pequenas são usadas muito frequentemente (50% dos operandos).
- Exemplos:
  - *A = A + 1;*
  - *B = 0 + 1;*
- Soluções? Vale a pena?
  - Determinar constantes típicas na memória e carrega-las.
  - Criar registradores "*hard-wired*" (como $zero) para constantes como 0 ou 1.
- Princípio de projeto: agilizar o caso comum. Que formato?

Constantes

- Com instruções do formato I (imediato) podemos utilizar constantes em nossas soluções.

- Qual é a limitação dessa solução?
  - *Formato da instrução.*

- Como trabalhar com constantes de 32 bits?

Constantes

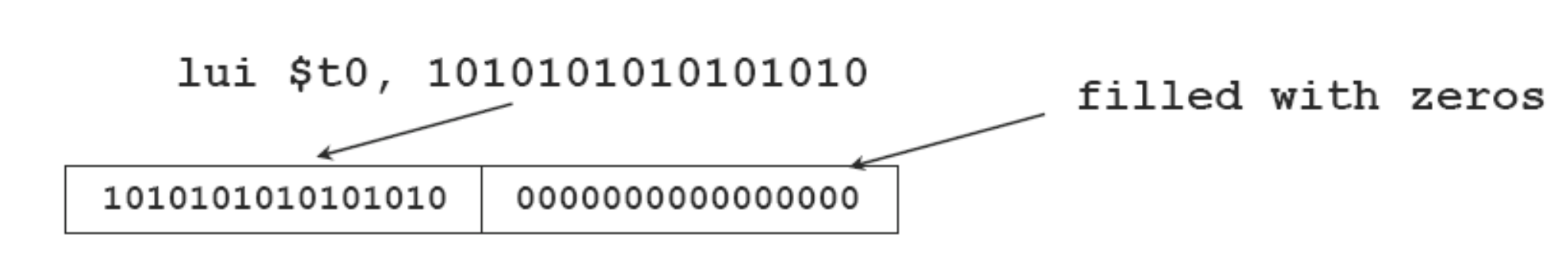
- Solução do MIPS: uma nova instrução que permitirá o carregamento de constantes maiores.

- Instrução *lui* (*load upper immediate):* carrega uma constante de 16 bits para os 16 bits mais significativos de um registrador.

- Estes 16 bits são os mais significativos da constante que desejamos carregar para um registrador.

Constantes

- Exemplo:



- A instrução *lui* deve ser utilizada em conjunto com outra instrução para carregar a constante de 32 bits.

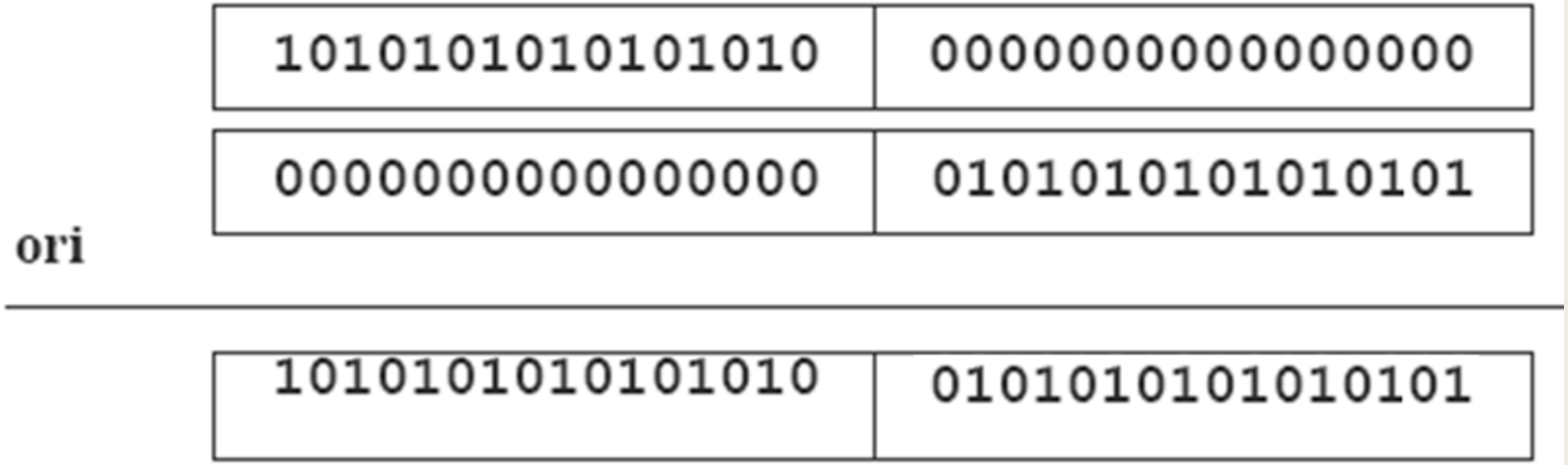
- Com qual instrução podemos preencher os 16 bits menos significativos do registrador $t0?

Constantes

- Alternativas:
  - Instrução *addi*.
  - Instrução *ori*.
- **Ambas funcionam em todos os casos?**

Constantes

```
ori $t0, $t0,0101010101010101
```



Constantes

```
addi $t0, $t0,0101010101010101
```

	1010101010101010	0000000000000000
addi	0000000000000000	0101010101010101

1010101010101010	0101010101010101

Constantes

- Exemplo no PCSPIM.

Roteiro

- Constantes.
- Endereços em desvios condicionais e incondicionais.
- Modos de endereçamento.
- Resumo e revisão geral das instruções do MIPS.
- Exercícios.

Endereços em Desvios

- Instruções:

  bne $t4,$t5,Label **##A próxima instrução está em Label se $t4 != $t5**

  beq $t4,$t5,Label **##A próxima instrução está em Label se $t4 == $t5**

  j Label **##A próxima instrução está em Label**

- Formato de instruções do tipo I e J:

I	op	rs	rt	endereço de 16 bits
J	op	endereço de 26 bits		

- Problema: se o programa precisasse caber nestes 16/26 bits, o tamanho máximo do programa ficaria limitado (2^{16}/2^{26}).
- Solução?

Endereços em Desvios

- Instruções do MIPS possuem endereço em *bytes*.
- 1 *word* = 1 instrução = 4 *bytes*.
- Para acessar uma posição de memória (*word*) do MIPS, o endereço deve ser múltiplo de 4.
- Se os endereços nos *jumps* e *branchs* considera o endereçamento por palavra, podemos aumentar 2 bits para endereçamento (multiplicar por 4 a capacidade de endereçamento).
- Porém, ainda não temos 32 bits de endereçamento. Como resolver?

Endereços em Desvios

- Poderíamos especificar um registrador base (como em *lw* e *sw*) e acrescentá-lo ao endereço.

- Detalhe: a maioria dos desvios é local (princípio da localidade).

- Escolha natural: usar o registrador de endereço de instrução (PC = contador do programa)

- Endereçamento relativo ao PC ($\pm 2^{15}$ instruções).

Endereços em Desvios

- E a solução para a instrução *jump*?

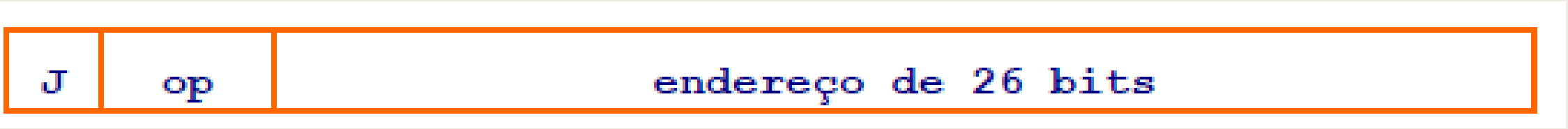
- Se os endereços nos *jumps* e *branchs* considera o endereçamento por palavra, podemos aumentar 2 bits para endereçamento (multiplicar por 4 a capacidade de endereçamento).

- 28 bits. Como complementar os outros 4 bits?

  - Solução: 4 bits mais significativos do PC são usados.

Endereços em Desvios

- Formato da instrução jump.



- Qual a faixa de endereços acessível pelo jump?
- Qual a faixa de endereços acessível pelos desvios condicionais?

Endereços em Desvios

- Problema: Como saltar para posições superiores?
  - Solução: Usar *jump register* depois de carregar uma constante de 32 bits no registrador.

- Problema: Como desviar para um ponto mais distante em relação ao PC?
  - Solução: Inverter a condição de desvio e acrescentar um *jump*.
  - Ex: beq $s0,$s1, L1 -> bne $s0,$s1, L2

    J L1

    L2:

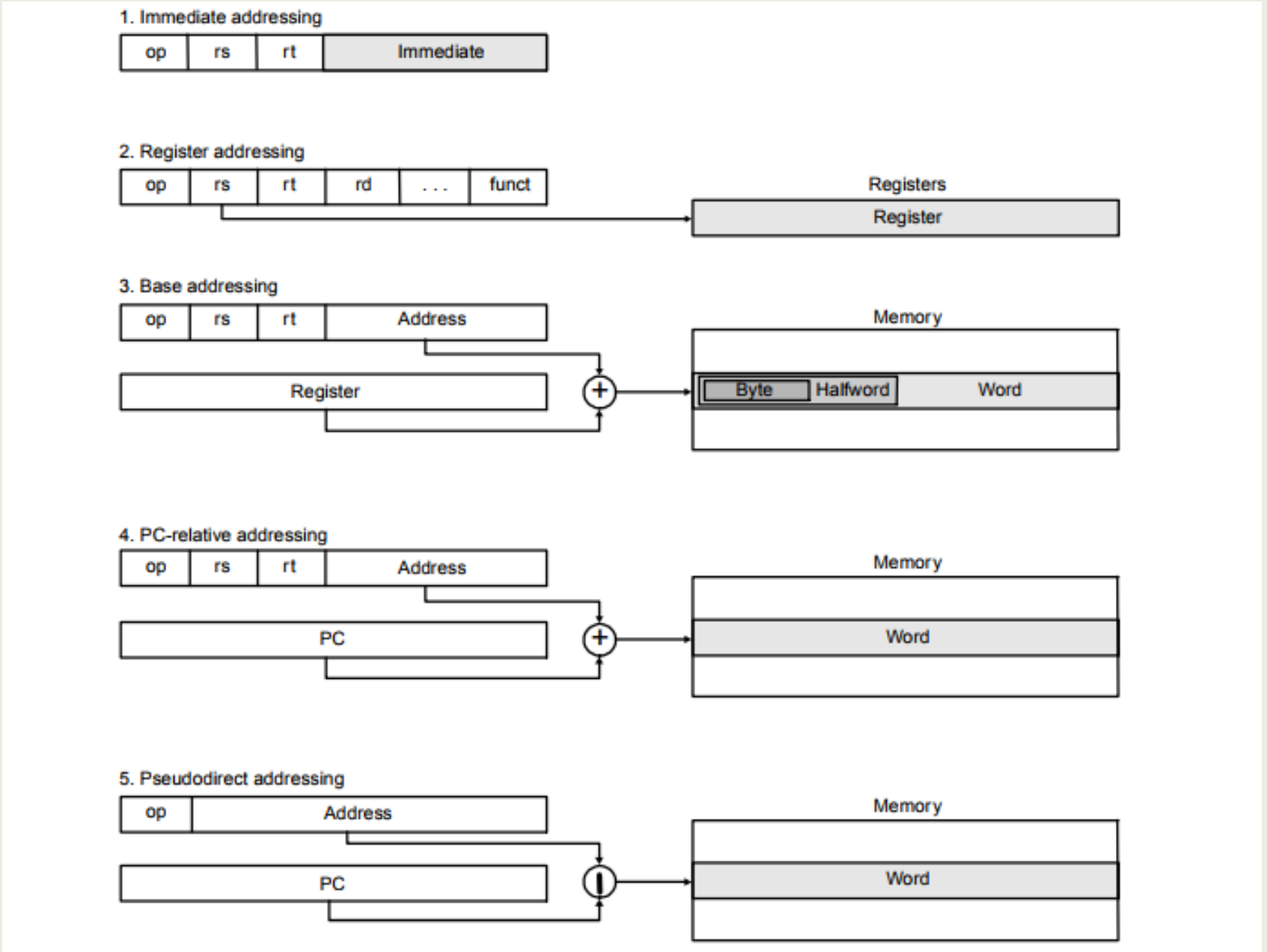
Roteiro

- Constantes.
- Endereços em desvios condicionais e incondicionais.
- Modos de endereçamento.
- Resumo e revisão geral das instruções do MIPS.
- Exercícios.

Modos de Endereçamento

- Endereçamento imediato: Operando é uma constante na instrução.
- Endereçamento em registrador: Operando é um registrador.
- Endereçamento de base ou deslocamento: Registrador base + constante na instrução.
- Endereçamento relativo ao PC: PC + constante na instrução.
- Endereçamento pseudodireto: Constante das instruções de *jump* concatenada com os 4 bits mais altos do PC.

Modos de Endereçamento



Roteiro

- Constantes.
- Endereços em desvios condicionais e incondicionais.
- Modos de endereçamento.
- Resumo e revisão geral das instruções do MIPS.
- Exercícios.

Resumo

- Instruções de 32 bits.
- 3 formatos de instrução:

R	op	rs	rt	rd	shamt	funct
I	op	rs	rt	endereço de 16 bits		
J	op	endereço de 26 bits				

Resumo

Nome	Número do registrador	Uso
$zero	0	O valor constante 0
$v0-$v1	2-3	Valores para resultados e avaliação de expressões
$a0-$a3	4-7	Argumentos
$t0-$t7	8-15	Temporários
$s0-$s7	16-23	Valores salvos
$t8-$t9	24-25	Mais temporários
$gp	28	Ponteiro global
$sp	29	Ponteiro de pilha
$fp	30	Pointeiro de quadro
$ra	31	Endereço de retorno

Registrador 1 ($at) reservado para o assembler, 26-27 para o sistema operacional

Assembly do MIPS				
Categoria	Instrução	Exemplo	Significado	Comentários
Aritmética	add	add $s1,$s2,$s3	$s1 = $s2 + $s3	Três operandos; dados nos registradores
	subtract	sub $s1,$s2,$s3	$s1 = $s2- $s3	Três operandos; dados nos registradores
	add immediate	addi $s1,$s2,100	$s1=$s2 + 100	Usada para somar constantes
Transferência de dados	load word	lw $s1,100($s2)	$s1 = Memória[$s2 + 100]	Dados da memória para o registrador
	store word	sw $s1,100($s2)	Memória[$s2 + 100] = $s1	Dados do registrador para a memória
	load byte	lb $s1,100($s2)	$s1 = Memória[$s2 + 100]	Byte da memória para registrador
	store byte	sb $s1,100($s2)	Memória[$s2+100] = $s1	Byte de um registrador para memória
	load upper immed.	lui $s1,100	$s1 = 100 * 216	Carrega constante nos 16 bits mais altos
Desvio condicional	branch on equal	beq $s1,$s2,25	if ($s1 == $s2) go to PC + 4 + 100	Testa igualdade; desvio relativo ao PC
	branch on not equal	bne $s1,$s2,25	if ($s1 != $s2) go to PC + 4 + 100	Testa desigualdade; relativo ao PC
	set on less than	slt $s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0	Compara menor que; usado com beq, bne
	set less than immediate	Slti$s1,$s2,100	if ($s2 < 100) $s1 = 1; else $s1 = 0	Compara menor que constante
Desvio incondicional	jump	j 2500	go to 10000	Desvia para endereço de destino
	jump register	jr $ra	go to $ra	Para switch e retorno de procedimento
	jump and link	jal 2500	$ra = PC + 4. go to 10000	Para chamada de procedimento

Roteiro

- Constantes.
- Endereços em desvios condicionais e incondicionais.
- Modos de endereçamento.
- Resumo e revisão geral das instruções do MIPS.
- Exercícios.

Lista de Exercícios

1. Construa um programa *assembly* MIPS que carregue duas constantes de 32 bits nos registradores $s1 e $s2 e realize a soma destes valores. Jogue o resultado no registrador $s3. Realize no caderno ou utilizando o simulador PCSPIM.

2. Realize o exercício anterior com duas instruções adicionais: uma utilizando *addi* e outra com *ori*. Teste as duas versões com as mesmas constantes: 1° caso com constantes positivas e 2° caso com constantes negativas (representação é em complemento de 2). Descreva qual a diferença nos resultados em utilizar *addi* e *ori*.

3. Classifique o modo de endereçamento das seguintes instruções e descreva o motivo.

- *add*
- *addi*
- *lw*
- *beq*
- *jal*
- *jr*
- *and*

Lista de Exercícios

- Entrega por email ou escrito até o dia 2 de outubro (segunda-feira).
- Dúvidas?
  - *Email: lucianobrum18@gmail.com*
  - *Sala: 3143.*
- **Leitura complementar:** Seção 2.9 do livro Organização e Projeto de Computadores – Pattersson e Hennessy.
- **Vídeo-aulas sobre PCSPIM - Canal do professor Sandro Camargo:** https://www.youtube.com/watch?v=1dWhY2brCUs

Referências Bibliográficas

- PATTERSON, D. A., HENNESSY, J, L. **Organização e Projeto de Computadores - a interface Hardware/Software**. 3. Ed., Campus. 2005.

Obrigado !