Universidade Federal de Pelotas
Instituto de Física e Matemática
Departamento de Informática
Bacharelado em Ciência da Computação

Arquitetura e Organização de Computadores II

Aula 1

Arquitetura do Processador MIPS: características gerais, registradores, formatos de instrução, modos de endereçamento

Prof. José Luís Güntzel

guntzel@ufpel.edu.br

www.ufpel.edu.br/~guntzel/AOC2/AOC2.html

Arquitetura do MIPS

Registradores

- 32 registradores de propósito geral de 32 bits
  - $0, $1, …, $31
  - operações inteiras
  - endereçamento
- $0 tem sempre valor 0
- $31 guarda endereço de retorno de sub-rotina

Arquitetura do MIPS

Tipos de Dados

- **dados inteiros disponíveis em instruções load e store**
  - bytes
  - meias-palavras de 16 bits
  - palavras de 32 bits
- **dados inteiros disponíveis em instruções aritméticas e lógicas**
  - meias-palavras de 16 bits (estendidos para 32 bits)
  - palavras de 32 bits

Arquitetura do MIPS

Modos de Endereçamento

- **acessos à memória devem ser alinhados**
  - dados de 32 bits precisam iniciar em endereços múltiplos de 4
  - dados de 16 bits precisam estar em endereços múltiplos de 2
- **modo registrador**
  - para instruções aritméticas e lógicas: dado está em registrador
  - para instruções de desvio incondicional: endereço está em registrador

Arquitetura do MIPS

Modos de Endereçamento

- **modo base e deslocamento**
  - para instruções load e store
  - base é registrador inteiro de 32 bits
  - deslocamento de 16 bits contido na própria instrução
- **modo relativo ao PC**
  - para instruções de branch condicional
  - endereço é a soma do PC com deslocamento contido na instrução
  - deslocamento é dado em palavras e precisa ser multiplicado por 4

Arquitetura do MIPS

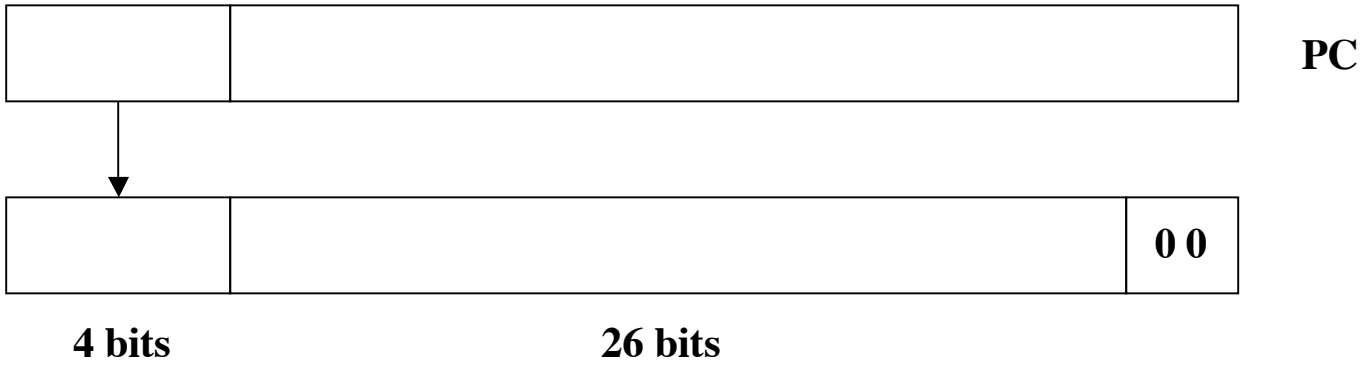
Modos de Endereçamento

- **modo imediato**
  - para instruções aritméticas e lógicas
  - dado imediato de 16 bits contido na própria instrução
  - dado é estendido para 32 bits
    - extensão com sinal nas instruções aritméticas
    - extensão sem sinal nas instruções lógicas
- **para que se possa especificar constantes de 32 bits**
  - instrução lui (load upper immediate)
  - carrega 16 bits imediatos na parte superior do registrador
  - parte inferior do registrador é zerada
  - instrução seguinte precisa fazer soma imediata do registrador com 16 bits da parte inferior

Arquitetura do MIPS

Modos de Endereçamento

- **modo absoluto**
  - para instruções de desvio incondicional
  - instrução tem campo com endereço de palavra com 26 bits
  - endereço de byte obtido com dois bits menos significativos iguais a 0
  - 4 bits mais significativos obtidos do PC
  - só permite desvios dentro de uma área de 256 Mbytes



Arquitetura do MIPS

Instruções Principais

- **Instruções de referência à memória (tipo I):**

Load word (lw) e store word (sw)

- **Instruções aritméticas e lógicas (tipo R):**

(add, sub, and, or, slt)

- **Instruções de salto:**

Branch on equal (beq) e jump (j)

Arquitetura do MIPS

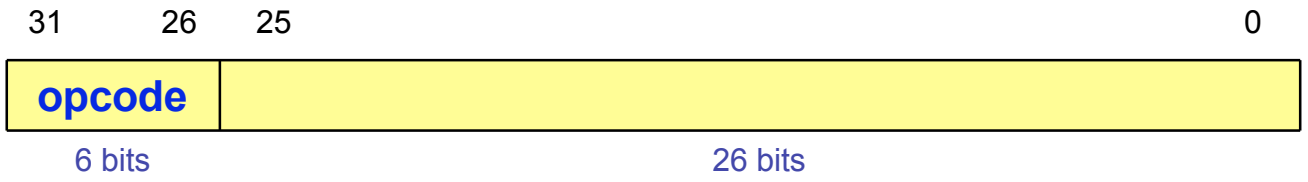
Instruções Principais

tipo	Instrução	exemplo	significado
Aritmética	Add immediate	addi $s1, $s2, 100	$s1 = $s2 + 100
Aritmética	Add	add $s1, $s2, $s3	$s1 = $s2 + $s3
Aritmética	Subtract	sub $s1, $s2, $s3	$s1 = $s2 - $s3
Aritmética	OR	or $s1, $s2, $s3	$s1 = $s2 OR $s3
Aritmética	AND	and $s1, $s2, $s3	$s1 = $s2 AND $s3
Transf. de dados	Load word	lw $s1, 100($s2)	$s1 = Mem[$s2+100]
Transf. de dados	Store word	sw $s1, 100($s2)	Mem[$s2+100] = $s1
Transf. de dados	Load Upper Immediate	lui $s1, 100	$s1 = 100 * 2^{16}
Desvio cond.	Branch on equal	beq $s1, $s2, L	if($s1 == $s2) go to L
Desvio cond.	Branch on not equal	bne $s1, $s2, L	if($s1 != $s2) go to L
Desvio cond.	Set on less than	slt $s1, $s2, $s3	if($s2 < $s3) $s=1; else $s1=0
Desvio cond.	Set less than immediate	slti $s1, $s2, 100	if($s2 < 100) $s=1; else $s1=0
Desvio incond.	Jump	j 2500	Desvia para 10000
Desvio incond.	Jump register	jr s$1	Desvia para $t1
Desvio incond.	Jump and link	jal 2500	$ra = PC+4; desvia para 10000

Arquitetura do MIPS

Instruções

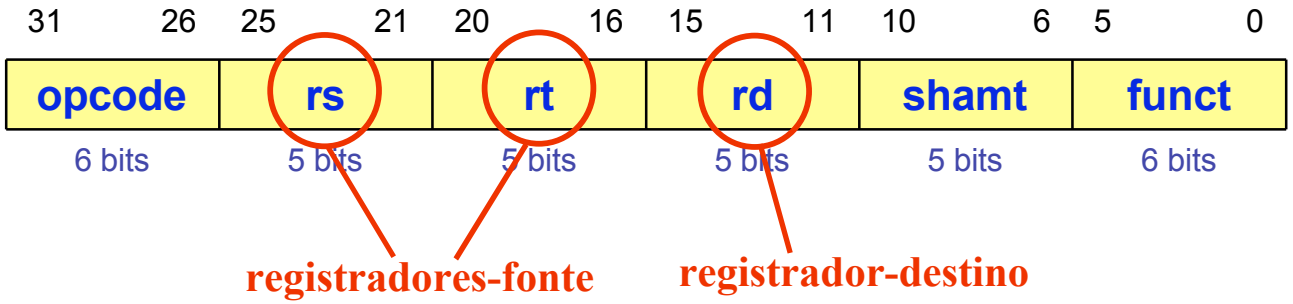
- todas as instruções têm 32 bits
- todas têm opcode de 6 bits
- o modo de endereçamento é codificado juntamente com o opcode



Arquitetura do MIPS

Instruções formato R: add, sub, or, and

- opcode = 0
- "funct" define a operação a ser feita pela ALU
- "shamt" (shift amount) é usado em instruções de deslocamento

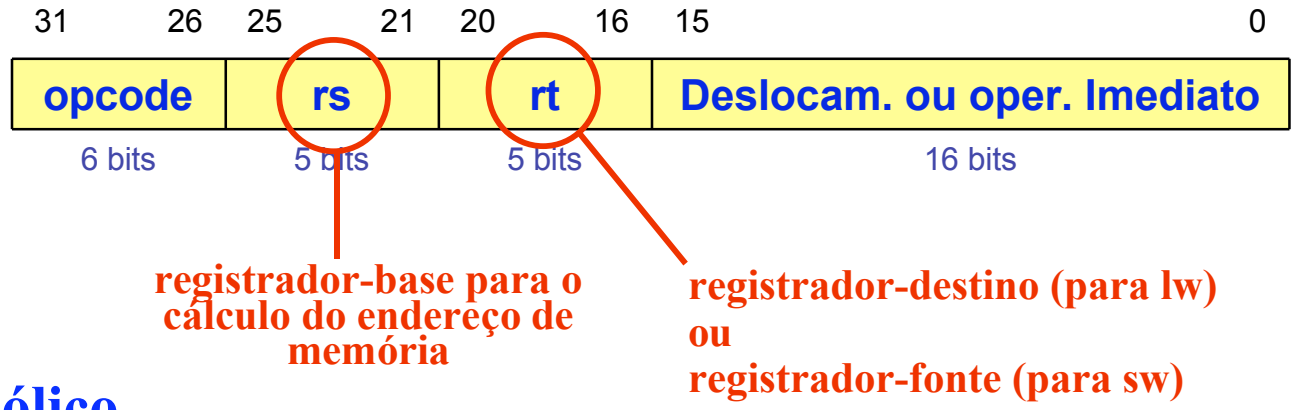


Simbólico (exemplo): `add $s1,$s2, $s3` `($s1 ← $s2 + $s3)`

Arquitetura do MIPS

Instruções formato I: load word (ld) e store word (sw)

- load word (lw): opcode = 35
- store word (sw): opcode = 43



Simbólico

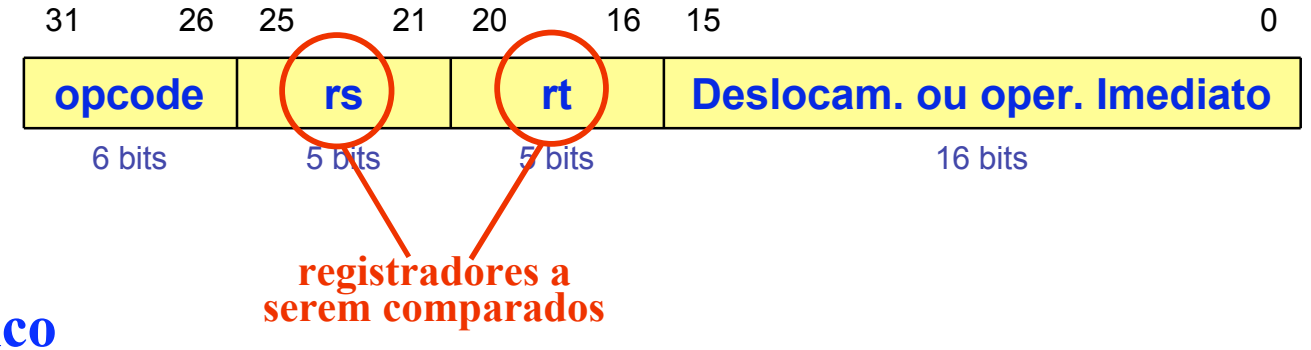
```
lw $s1, deslocam($s2)   ($s1 ← Mem[$s2 + deslocam] )
sw $s1, deslocam($s2)   (Mem[$s2 + deslocam] ← $s1 )
```

Arquitetura do MIPS

Instrução formato I: Desvio Condicional

beq: branch on equal

- Opcode = 4
- Campo deslocamento usado para calcular o endereço-alvo
- Se o conteúdo do registrador cujo endereço está no campo rs for igual ao conteúdo do registrador cujo endereço está em rt, então salta para a posição endereço+PC+4



Simbólico

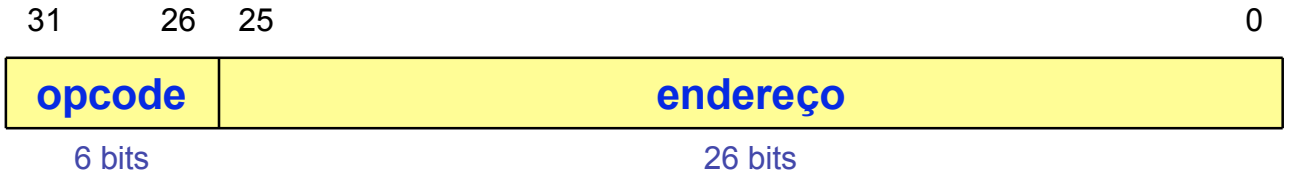
```
beq $s1, $s2, deslocm  ( if ($s1== $s2) then PC←PC+4+deslocam)
```

Arquitetura do MIPS

Instrução formato J: Desvio Incondicional

j: jump

- Opcode = 2
- Campo deslocamento usado para calcular o endereço-alvo
- Se o conteúdo do registrador cujo endereço está no campo rs for igual ao conteúdo do registrador cujo endereço está em rt, então salta para a posição endereço+PC+4



Simbólico

```
j endereço  ( PC←endereço )
```

Arquitetura do MIPS

Instruções load/store

- são sempre tipo I
- qualquer registrador de propósitos gerais ou de ponto flutuante pode ser carregado ou armazenado da / na memória
- pode-se carregar ou armazenar bytes, meias palavras, palavras ou palavras em precisão dupla (ponto flutuante)
- endereçamento sempre por base e deslocamento

Arquitetura do MIPS

Instruções load/store

- lb, lh, lw – load byte, halfword, word
  - sinal é estendido em lb e lh
- lbu, lhu – load byte, halfword sem extensão de sinal
- sb, sh, sw – store byte, halfword, word

Arquitetura do MIPS

Instruções Aritméticas e Lógicas

- operação entre 2 registradores, resultado num terceiro registrador
  - tipo R
- add, and, nor, ori, or, sub, xor
- existem versões com endereçamento imediato
  - addi, andi, ori, xori
  - tipo I
- comparação – compara dois registradores e coloca valor 1 (ou 0) em registrador destino
  - slt, slti
- $0 usado para sintetizar operações populares
  - carga de constante = soma imediata onde $0 é um dos operandos
  - mover de registrador para registrador = soma com $0

Arquitetura do MIPS

Instruções Aritméticas e Lógicas

- instruções de deslocamento variável
  - tipo R
  - sllv, srlv – shift lógico (entra 0 na extremidade)
  - srav – shift aritmético (duplica sinal)
  - desloca registrador rt pela distância especificada no registrador rs e coloca resultado no registrador rd
- instruções de deslocamento constante
  - tipo I
  - sll, sra, srl
  - desloca registrador rt pela distância especificada no campo imediato e coloca resultado no registrador rd

Arquitetura do MIPS

Instruções Aritméticas e Lógicas

- instrução de multiplicação: mul
  - multiplica registradores rs e rt
  - resultado colocado em hi (32 msb) e lo (32 lsb)
- instrução de divisão: div
  - divide registrador rs pelo registrador rt
  - quociente colocado em lo
  - resto colocado em hi
- instruções de movimentação permitem transferir dados entre hi e lo e os demais registradores
  - mfhi Rd, mflo Rd
  - mthi Rs, mtlo Rs

Arquitetura do MIPS

Instruções Aritméticas e Lógicas

- instruções “unsigned” não geram overflow
  - addu, addiu, divu, multu, subu, sltu, sltiu
- instruções “não-unsigned” geram overflow
  - endereço da instrução que causou overflow é colocado no registrador EPC (em um co-processador)
  - instrução “mfc0” copia valor do EPC para outro registrador qualquer
  - instrução “jr” (jump para endereço especificado por registrador) é usada para desviar para rotina de atendimento

Arquitetura do MIPS

Instruções de Desvio Incondicional

- instrução j
  - tipo J
  - endereço destino = concatenação dos 4 msb do PC com endereço imediato de 28 bits
- instrução jr
  - tipo I: endereço destino contido em registrador
  - também serve para retornar de sub-rotina
- instrução jal (jump and link)
  - tipo J
  - desvio para sub-rotina, endereço especificado na instrução
  - endereço de retorno salvo em $31
- instrução jalr (jump and link register)
  - tipo R
  - desvio para sub-rotina, endereço especificado em rs
  - endereço de retorno salvo em rd

Arquitetura do MIPS

Instruções de Desvio Condicional

- são sempre tipo I
- endereço destino = soma do PC com offset imediato de 16 bits
- instruções que testam um único registrador
  - bgez, bgtz, blez, bltz – desvia se registrador é $\geq$, $>$, $\leq$, $<$ zero
  - bgezal, bltzal – como bgez e bltz, mas salva de endereço de retorno em $31
- instruções que comparam dois registradores
  - beq, bne – desvia se registradores são iguais (diferentes)

Arquitetura do MIPS

Instruções de Ponto Flutuante

- mover operandos de precisão simples ou dupla entre registradores
  - mov.d, mov.s
- soma, subtração, negação, multiplicação, divisão em precisão simples e dupla
  - add.s, sub.s, neg.s, mul.s, div.s
  - add.d, sub.d, neg.d, mul.d, div.d
- comparações em precisão simples e dupla
  - c.eq.s, c.le.s, c.lt.s,
  - c.eq.d, c.le.d, c.lt.d,
- conversão para ponto flutuante de precisão simples (ou dupla)
  - cvt.s.d, cvt.s.w – converter FP double (ou inteiro) para FP single
  - cvt.d.s, cvt.d.w – converter FP single (ou inteiro) para FP double
- conversão de ponto flutuante para inteiro
  - cvt.w.s, cvt.w.d – converter FP double (ou single) para inteiro