Resumo - Aula 4

Arquitetura MIPS

Prof. Sérgio Ronaldo

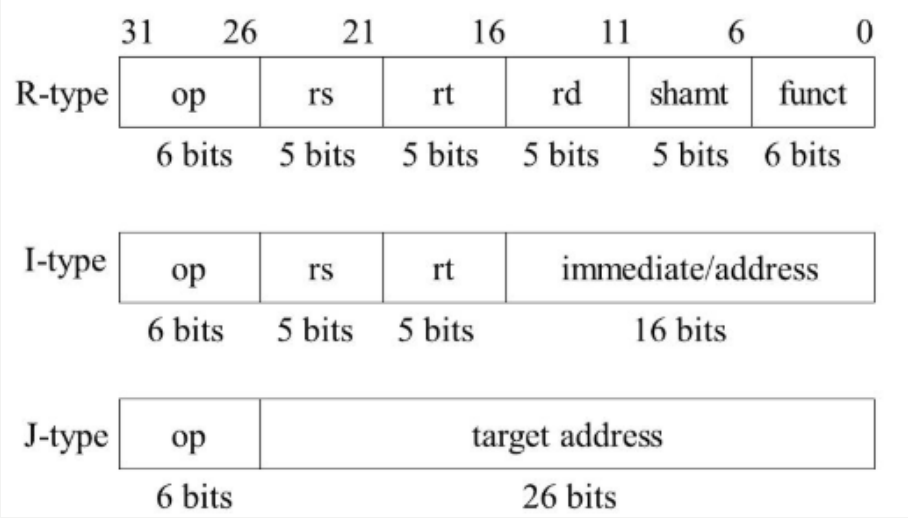
Sumário

- Arquitetura MIPS (Cap. 2 - Patterson)

Conjunto de Instruções

- Os três formatos usados para o conjunto de instrução do núcleo:

Tipo R - opcode (6) rs (5) rt (5) rd (5) shamt (5) funct (6)
Tipo I - opcode (6) rs (5) rt (5) imediato (16)
Tipo J - opcode (6) endereço (26)



Conjunto de Instruções

- Instrução é uma palavra da linguagem de máquina
- ISA (Instruction Set Architecture)
  - Conjunto de instruções de uma máquina
- ISA MIPS
  - 3 formatos de instruções
  - instruções de 3 operandos

Programa em C	Assembly MIPS
a = b + c; d = a – c;	add a,b,c sub d,a,c

f = (g + h) – (i + j);	add t0,g,h
	add t1,i,j
	sub f,t0,t_1
	o compilador cria t0 e t1 .

Conjunto de Instruções

- Operandos
  - No MIPs os operandos das instruções são registradores
    - 32 registradores de 32 bits

Programa em C	Assembly MIPS

f = (g + h) – (i + j);	add $t0,$s1,$s2
	add $t1,$s3,$s4
	sub $s0,$t0,$t1

Conjunto de Instruções

- Load e Store
  - lw : instrução de movimentação de dados da memória para registrador (load word)
    - Sintaxe: lw $destino, deslocamento($origem)

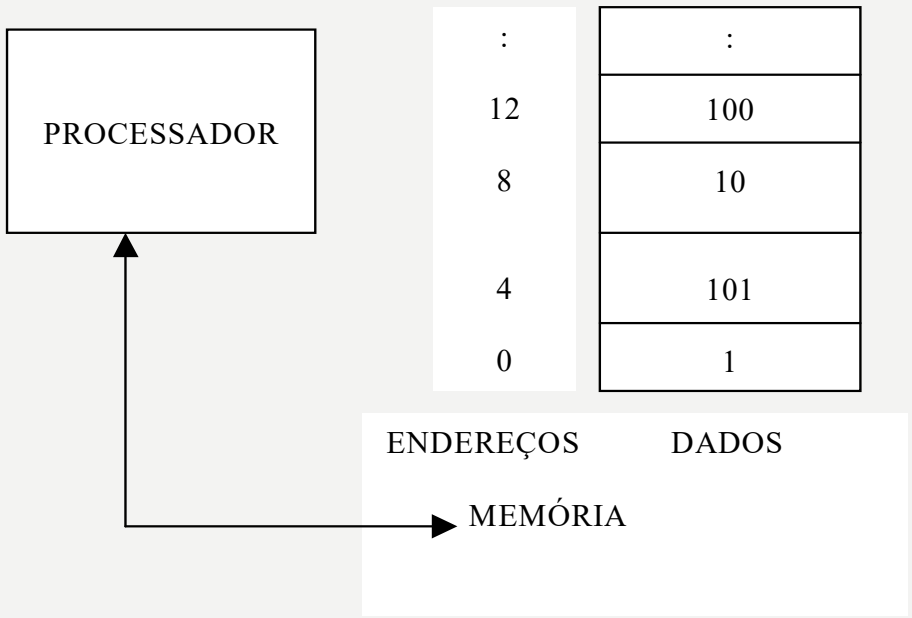
```
lw $r1, 4($r2) # Load Word: Esta instrução carrega uma palavra (estrutura de 4 bytes)
               # localizada no endereço representado pela soma do valor
               # armazenado no registrador $r2 mais 4. O resultado é armazenado em $r1.
```

  - sw: instrução de movimentação de dados do registrador para a memória (store word)
    - Sintaxe: sw $fonte, deslocamento($origem)

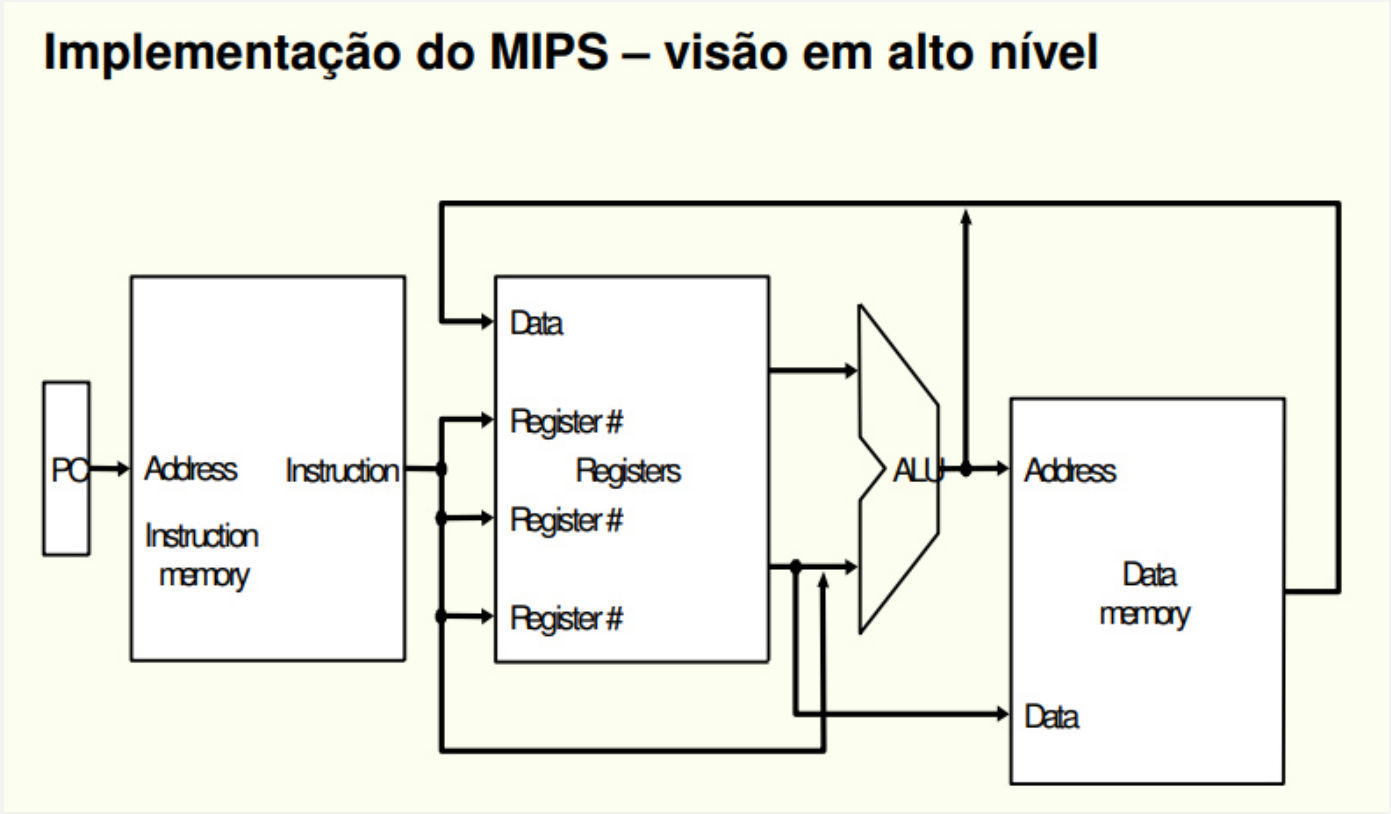
```
sw $r1, 4($r2) # Store Word: Esta instrução carrega uma palavra (estrutura de 4 bytes)
               # localizada no registrador $r1 e armazena no endereço representado
               # pela soma do valor armazenado no registrador $r2 mais 4.
```

Memória de Programa

- No MIPS a memória é organizada em bytes, embora o endereçamento seja em palavras de 4 bytes (32 bits)

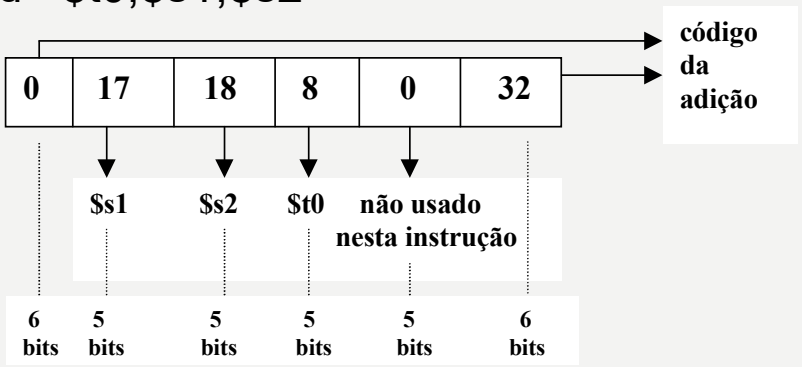


Implementação MIPS



Formato da Instrução R

- Formato da instrução add $t0,$s1,$s2



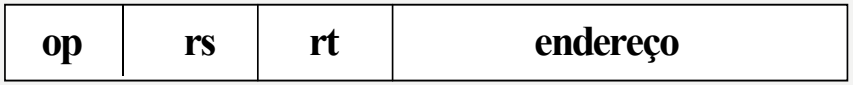
- Formato das instruções tipo R (R-type) e seus campos

op	rs	rt	rd	shamt	funct

op ➔ **operação básica da instrução (opcode)**
rs ➔ **o primeiro registrador fonte**
rt ➔ **o segundo registrador fonte**
rd ➔ **o registrador destino**
shamt ➔ **shift amount, para instruções de deslocamento**
funct ➔ **function. Seleciona variações das operação especificada pelo opcode**

Formato da Instrução I

- Formato das Instruções tipo I (I-type)



- Exemplo de instruções I-type
  - lw $t0, 32($s3)

- Codificação de Instruções MIPS

Instrução	Formato	Op	rs	rt	rd	Sham t	func	end.
Add	R	0	reg	reg	reg	0	32	n.d
Sub	R	0	reg	reg	reg	0	34	n.d
Lw	I	35	reg	reg	n.d.	n.d	n.d	end.
Sw	I	43	reg	reg	n.d	n.d	n.d	end.

Formato da Instrução I

- Mostre o código assembly do MIPS e o código de máquina para o seguinte comando em C: “A[300] = h + A[300];”, onde $t1 tem o endereço base do vetor A e $s2 corresponde a h.
- Considere que o dado ocupe

```
lw    $t0,1200($t1)   #  $t0 recebe A[300]
add   $t0,$s2,$t0     #  $t0 recebe h + A[300]
sw    $t0,1200($t1)   #  A[300] recebe h + A[300]
```

- **Linguagem de máquina**

Op	rs	rt	rd	end/shamt	funct
35	9	8	1200		
0	18	8	8	0	32
43	9	8	1200		

Formato da Instrução J

- **Instruções de desvio condicional**
- beq registrador1, registrador2, L1
  - se o valor do registrador1 for igual ao do registrador2 o programa será desviado para o label L1

    (beq = branch if equal).

```
beq $r1, $r2, DESTINO
```

- bne registrador1, registrador2, L1
  - se o valor do registrador1 não for igual ao do registrador2 o programa será desviado para o label L1
  - (bne = branch if not equal).

```
bne $r1, $r2, DESTINO
```

Formato da Instrução J

Exemplo - Compilando um comando IF.

Seja o comando abaixo:

```
      if ( i == j ) go to L1;
      f = g + h;
L1: f = f - i;
```

Supondo que as 5 variáveis correspondam aos registradores $s0..$s4, respectivamente, como fica o código MIPS para o comando?

Solução

```
      beq   $s3,$s4,L1     #   vá para L1 se i = j
      add   $s0,$s1,$s2    #   f = g + h, executado se i != j
L1:   sub   $s0,$s0,$s3    #   f = f – i, executado se i = j
```

Formato da Instrução J

- **J L1**
  - quando executado faz com que o programa seja desviado para L1

```
j ENDEREÇO
```

Exemplo : Seja o comando abaixo

```
if (i == j)
    f = g + h;
else
    f = g – h;
```

Solução

```
        bne $s3,$s4,Else    #  vá para Else se i != j
        add $s0,$s1,$s2     #  f = g + h, se i == j
        j Exit              #  vá para Exit
Else:   sub $s0,$s1,$s2     #  f = g – h, se i != j
Exit:
```