Universidade Federal de Pelotas
Centro de Desenvolvimento Tecnológico
Bacharelado em Ciência da Computação
Engenharia de Computação

Arquitetura e Organização de Computadores I

Prática

Aula 5

Jumps, Branches e Sets

Prof. Guilherme Corrêa
gcorrea@inf.ufpel.edu.br

Revisão

MIPS: estrutura básica

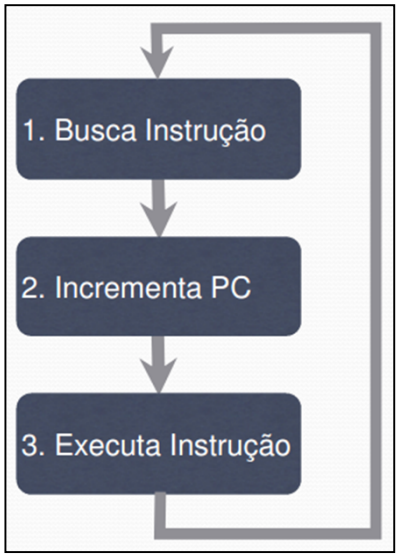
- Banco de registradores (32 registradores)
- Program Counter (PC)
- Memória (dados, instruções)
- Unidade Lógica e Aritmética (ULA)

Revisão

MIPS: estrutura básica

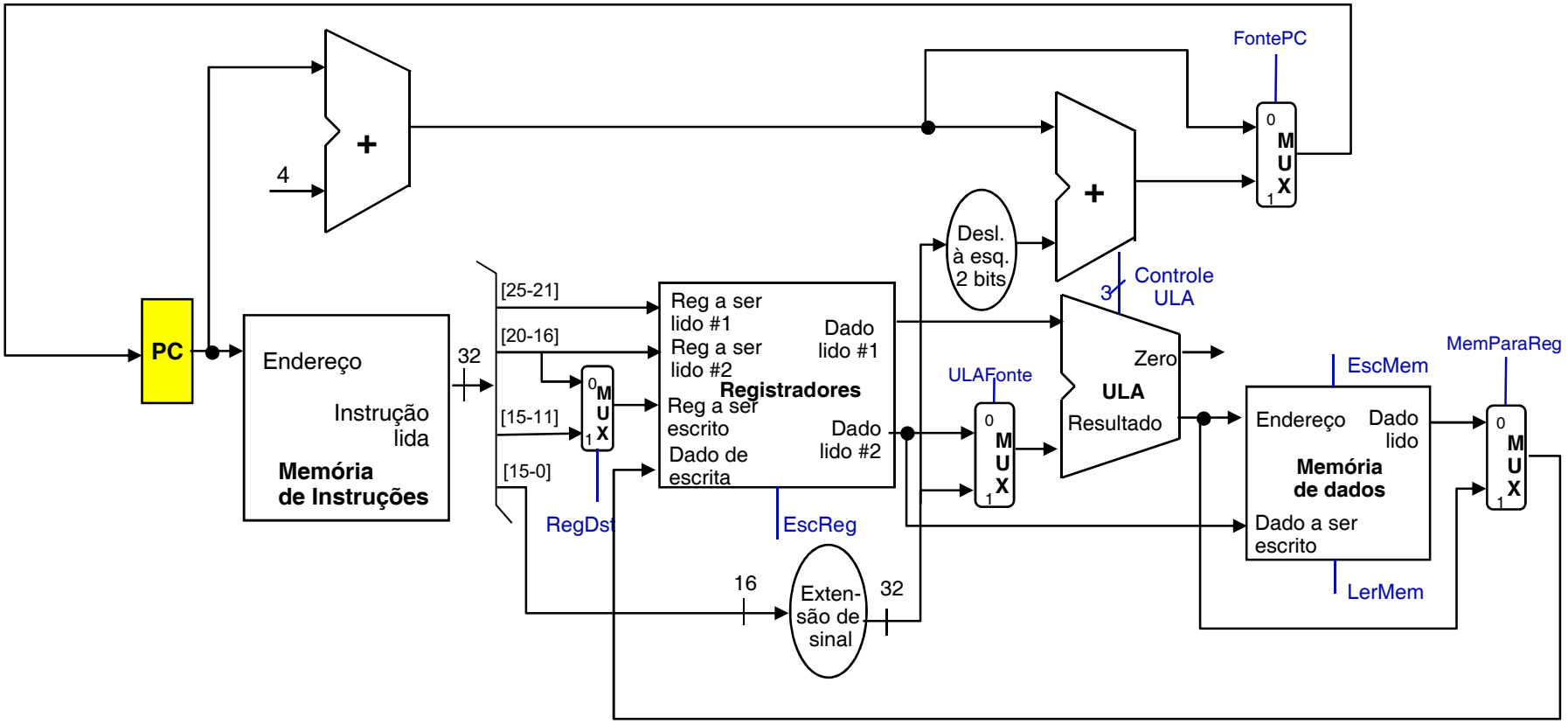
- Banco de registradores (32 registradores)
- Program Counter (PC)
- Memória (dados, instruções)
- Unidade Lógica e Aritmética (ULA)

Ciclo de Máquina



Revisão

Exemplo: MIPS (monociclo)



Desvios

Desvios Condicionais

- Permitem a tomada de decisões;
- Funcionam como ***if*** nas linguagens de alto nível;
- Se o teste resulta em **true**, mudam o endereço do PC para o endereço de outra instrução (ao invés de PC+4).

Desvios Incondicionais

- Funcionam como ***go to*** nas linguagens de alto nível;
- Sempre mudam o endereço do PC para o endereço de outra instrução (ao invés de PC+4).

Desvios

Desvios Condicionais

- Permitem a tomada de decisões;

Felizmente podemos usar *labels* e não precisamos calcular o endereço dos desvios manualmente!

O *assembler* calcula os endereços apropriados

D

ra

outra instrução (ao invés de PC+4).

Desvios Condicionais

Instruções

- *Branch if equal* (beq)
  - Vai para a instrução L1 se o valor em $t1 for igual ao valor em $t2

    beq $t1, $t2, L1
- *Branch if not equal* (bne)
  - Vai para a instrução L1 se o valor em $t1 for diferente do valor em $t2

    bne $t1, $t2, L1

Desvios Condicionais

Instruções

- *Set on less than* (slt)
  - Se $t2 < $t3, coloca 1 em $t1; Senão, coloca 0 em $t1

    ```
    slt  $t1, $t2, $t3
    ```

- *Set on less than immediate* (slti)
  - Se $t2 < 100, coloca 1 em $t1; Senão, coloca 0 em $t1

    ```
    slti  $t1, $t2, 100
    ```

Desvios Incondicionais

Instruções

- *Jump* (j)
  - Vai para a instrução rotulada como L1

```
j  L1
```

- *Jump register* (jr)
  - Vai para a instrução cujo endereço está armazenado em $t2

```
j  $t2
```

Exemplo (1)

C/C++:

```
if (h == i)
    e = f + g;
else
    e = f - g;
```

Exemplo (1)

C/C++:

```
if (h == i)
    e = f + g;
else
    e = f - g;
```

Assembly:

```
        bne $s3,$s4,Else
        add $s0,$s1,$s2
        j Exit
Else:   sub $s0,$s1,$s2
Exit:   #continuação do programa
```

e: $s0	h: $s3
f: $s1	i: $s4
g: $s2	j: $s5

Exemplo (2)

C/C++:

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

Exemplo (2)

C/C++:

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

i: $s3
k: $s5
save[]: $s6

Assembly:

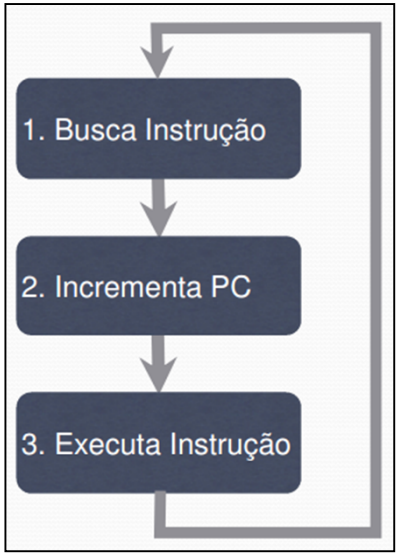
```
Loop: sll $t1,$s3,2
      add $t1,$t1,$s6
      lw $t0,0($t1)
      bne $t0,$s5,Exit
      addi $s3,$s3,1
      j Loop
Exit: #continuação do programa
```

Delayed Branching

ATENÇÃO!

- No MIPS pipeline, as instruções de desvio alteram o PC em um momento em que a instrução seguinte já foi buscada e está prestes a ser executada (passo 3 do Ciclo de Máquina);
- Portanto, mesmo que haja um desvio (*jump* ou *branch*), a instrução a seguir será executada!
- SOLUÇÃO: sempre usar uma instrução nop (*no operation*) após instruções de desvio.

Ciclo de Máquina



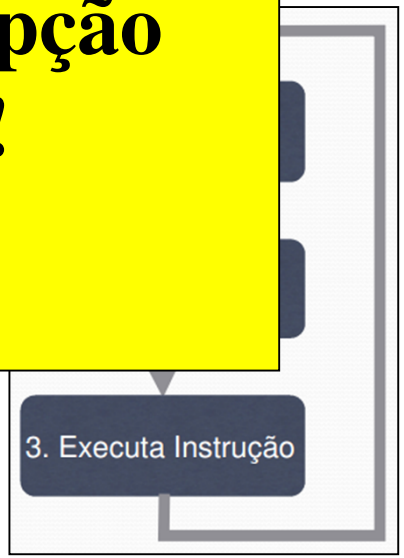
Delayed Branching

ATENÇÃO!

- No MIPS pipeline, as instruções de desvio alteram o PC
- clo
- uina

A partir de agora, todos os exercícios feitos no MARS devem ter a opção *Delayed Branching* ligada!

- ***nop** (no operation)* após instruções de desvio.



Exemplo (1)

C/C++:

```
if (h == i)
    e = f + g;
else
    e = f - g;
```

e: $s0	h: $s3
f: $s1	i: $s4
g: $s2	j: $s5

Assembly:

```
        bne $s3,$s4,Else
        sll $0,$0,0
        add $s0,$s1,$s2
        j Exit
        sll $0,$0,0
Else:   sub $s0,$s1,$s2
Exit:   #continuação do programa
```

Exemplo (2)

C/C++:

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

i: $s3
k: $s5
save[]: $s6

Assembly:

```
Loop: sll $t1,$s3,2
      add $t1,$t1,$s6
      lw $t0,0($t1)
      bne $t0,$s5,Exit
      sll $0,$0,0
      addi $s3,$s3,1
      j Loop
      sll $0,$0,0
Exit: #continuação do programa
```

Exemplo (3)

- O que faz?

Assembly:

```
  .text

main: sll $0,$0,0
      sll $0,$0,0
      sll $0,$0,0
      sll $0,$0,0
      j main
      sll $0,$0,0
```

Exemplo (4)

- O que faz?

```
##
##
##
## Registradores:
## $8  --- contador
## $9  --- flag
## $10 --- soma
##
        ori   $10,$0,0
        ori   $8,$0,0
teste:  slti  $9,$8,101
        beq   $9,$0,fim
        sll   $0,$0,0

        addu  $10,$10,$8

        addiu $8,$8,1
        j     teste
        sll   $0,$0,0

fim:    sll   $0,$0,0
```

Exemplo (4)

- O que faz?

```
##
## Soma de inteiros 0..100 (loop while)
##
## Registradores:
## $8  --- contador
## $9  --- flag
## $10 --- soma
##
        ori   $10,$0,0          # soma = 0
        ori   $8,$0,0           # contador = 0
teste:  slti  $9,$8,101         # contador < 101 ?
        beq   $9,$0,fim         # se contador > 101, sai do loop
        sll   $0,$0,0           # delay

        addu  $10,$10,$8        # soma += contador

        addiu $8,$8,1           # contador++
        j     teste             # volta pro inicio do loop
        sll   $0,$0,0           # delay

fim:    sll   $0,$0,0
```

Exemplo (5)

- O que faz?

```
## temperatura.asm
## Checar que 30 <= temperatura <= 55
## flag := 1 se a temperatura está no intervalo, 0 se está fora
##
## Registradores:
## $2 --- temperatura
## $3 --- dentro/fora do range (flag)
## $8 --- rascunho

        .text
        .globl main

main:   ori    $3,$0,1

# Testa se temperatura <= 55
        slti   $8,$2,56
        beq    $8,$0,fora
        sll    $0,$0,0

# Testa se temperatura >= 30
        slti   $8,$2,30
        beq    $8,$0,dentro
        sll    $0,$0,0

# Fora do intervalo
fora:   ori    $3,$0,0
        ...
dentro: sll    $0,$0,0
        ...
```

Exemplo (5)

- O que faz?

```
## temperatura.asm
## Checar que 30 <= temperatura <= 55
## flag := 1 se a temperatura está no intervalo, 0 se está fora
##
## Registradores:
## $2 --- temperatura
## $3 --- dentro/fora do range (flag)
## $8 --- rascunho

        .text
        .globl main

main:   ori   $3,$0,1                   # flag := 1

# Teste 30 <= temperatura <= 55
        slti  $8,$2,56                  # rascunho=1 se temp < 56
        beq   $8,$0,fora                # 0? fora do intervalo
        sll   $0,$0,0                   # delay

        slti  $8,$2,30                  # rascunho=1 se temp < 30
        beq   $8,$0,dentro              # 0? no intervalo
        sll   $0,$0,0                   # delay

# Fora do intervalo
fora:   ori   $3,$0,0                   # flag := 0
        ...
dentro:sll    $0,$0,0                   # alvo do jump
        ...
```