

Conjunto de Instruções

Parte III

Histórico de revisões

2

| Revisão | Data | Responsável | Descrição |
|---------|---------|----------------------|----------------------------|
| 0.1 | -x- | Prof. Cesar Zeferino | Primeira versão |
| 0.2 | 03/2017 | Prof. Cesar Zeferino | Revisão do modelo |
| 0.3 | 08/2017 | Prof. Cesar Zeferino | Revisão do conteúdo |
| 0.4 | 03/2023 | Prof. Felski | Revisão Arquitetura RISC-V |

Observação: Este material foi produzido por pesquisadores do Laboratório de Sistemas Embarcados e Distribuídos (LEDS – Laboratory of Embedded and Distributed Systems) da Universidade do Vale do Itajaí e é destinado para uso em aulas ministradas por seus pesquisadores.

Introdução

3

❑ Objetivo

- ❑ Saber programar instruções de desvio utilizando a linguagem de montagem do RISC-V

❑ Conteúdo

- ❑ Instruções de desvio
- ❑ Modos de endereçamento (cont.)
- ❑ Comparação *menor que*

Introdução

4

❑ Bibliografia

- ❑ PATTERSON, David A.; HENNESSY, John L. Abstrações e tecnologias computacionais. *In*: _____. **Organização e projeto de computadores: a interface hardware/software**. 4. ed. Rio de Janeiro: Campus, 2014. cap. 2. Disponível em:
<<http://www.sciencedirect.com/science/book/9788535235852>>. Acesso em: 13 mar. 2017.

- ❑ Edições anteriores
 - ❑ Patterson e Hennessy (2005, cap. 3)
 - ❑ Patterson e Hennessy (2000, cap. 3)

1 Instruções de desvio

❑ Desvio condicional

❑ Branch on equal (desvia se igual)

```
beq reg1, reg2, L1    #vai p/ L1 se reg1=reg2
```

❑ Branch on not equal (desvia se não igual)

```
bne reg1, reg2, L1    #vai p/ L1 se reg1/=reg2
```

❑ Desvio incondicional

❑ JAL

```
jal zero, L1          #vai p/ L1
```

L1 é um *label* (rótulo) que “abstrai” o endereço de uma instrução. Quem determina o endereço associado a L1 é o compilador (ou o montador).

1 Instruções de desvio: if

❑ Código em C

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

Onde:

```
f => s0
g => s1
h => s2
i => s3
j => s4
```

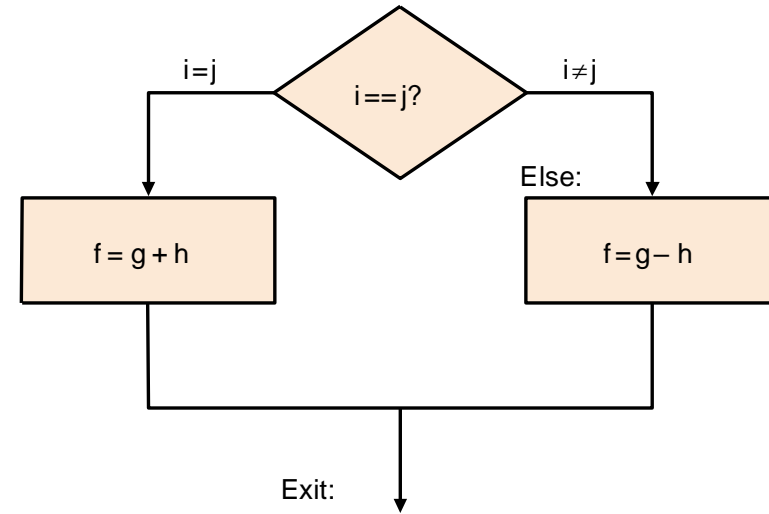
❑ Código em ASM (assembly)

```
    beq s3, s4, L1      # se i==j goto L1
    add s0, s1, s2      # f = g + h
L1:  sub s0, s0, s3      # f = f - i
```

1 Instruções de desvio: if-then-else

□ Código em C

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



□ Código em ASM (assembly)

```

    bne s3, s4, Else    # se i!=j vai p/ Else
    add s0, s1, s2      # f = g + h
    jal zero, Exit      # goto Exit
Else: sub s0, s1, s2    # f = g - h
Exit: nop               # não faz nada.
```

Desvio incondicional (não depende de nenhuma condição)

```
j label    # Vá para label
```

1 Instruções de desvio: Loop

❑ Código em C

```
Loop: g = g + A[i];
      i = i + j;
      if (i!=h) go to Loop;
```

Onde:

```
g => s1
h => s2
i => s3
j => s4
A => s5
```

❑ Código em ASM (assembly)

| | |
|----------------------|-------------------------|
| Loop: add t1, s3, s3 | #1: t1 = 2*i |
| add t1, t1, t1 | #2: t1 = 4*i |
| add t1, t1, s5 | #3: t1 = end.base + 4*i |
| lw t0, 0(t1) | #4: \$t0 = A[i] |
| add s1, s1, t0 | #5: g = g + A[i] |
| add s3, s3, s4 | #6: i = i + j |
| bne s3, s2, Loop | #7: Loop se i!=h |

1 Instruções de desvio: Laços while

❑ Código em C

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

Onde:

| | | |
|------|----|----|
| i | => | s3 |
| j | => | s4 |
| k | => | s5 |
| save | => | s6 |

❑ Código em ASM

```
Loop: add t1, s3, s3
      add t1, t1, t1
      add t1, t1, s6
      lw  t0, 0(t1)
      bne t0, s5, Exit
      add s3, s3, s4
      jal zero, Loop
Exit: nop
```

```
# t1 = 2*i
# t1 = 4*i
# t1 = end.base + 4*i
# t0 = save[i]
# Exit se save[i]!=k
# i = i + j
# desvia para Loop
```

1 Instruções de desvio: Laços for

❑ Código em C

```
for (i=0; i<8; i++) {
    (...)
}
```

Onde:

i => s1

❑ Código em ASM

```
add  s1, zero, $zero    # s1 = 0
```

```
addi t0, zero, 8        # t0 = 8
```

Loop:

```
beq  s1, t0, Exit        # Se s1=t0, sai
```

```
(...)
```

```
addi s1, s1, 1           # s1 = s1 + 1
```

```
jal  zero, Loop          # desvia para Loop
```

Exit:

```
nop
```

1 Instruções de desvio: Resumo

❑ Instruções de desvio condicional

❑ Baseadas no formato I

| Categoria | Instrução | Exemplo | Significado |
|--------------------|------------|----------------------------|---------------------------------|
| Desvio condicional | beq | <code>beq s1, s2, L</code> | Se $(s1 == s2)$ desvia para L |
| | bne | <code>bne s1, s2, L</code> | Se $(s1 != s2)$ desvia para L |
| | bge | <code>bge s1, s2, L</code> | Se $(s1 \geq s2)$ desvia para L |
| | blt | <code>bge s1, s2, L</code> | Se $(s1 < s2)$ desvia para L |


L é um número inteiro (positivo ou negativo) que será somado ao registrador (PC) de modo a determinar o endereço da próxima instrução a ser executada se houver desvio

1 Instruções de desvio: Resumo

❑ Instruções de desvio incondicional

❑ Baseadas no formato J

| Categoria | Instrução | Exemplo | Significado |
|----------------------|-----------|-------------|---------------|
| Desvio incondicional | jal | jal zero, L | Desvia para L |



Endereço de desvio
(endereço-relativo ao PC)

L é um número natural (não negativo) que será concatenado com os quatro bits mais significativos do PC para determinar o endereço da próxima instrução a ser executada