Microprocessadores e Microcontroladores (27146)

Conjunto de instruções (RISC-V)

Prof. Ricardo Menotti (menotti@ufscar.br)

Atualizado em: 14 de maio de 2021

Departamento de Computação
Centro de Ciências Exatas e de Tecnologia
Universidade Federal de São Carlos

Computadores RISC

David Patterson and John Hennessy in their text Computer Organization and Design:

(1) regularity supports simplicity;

(2) make the common case fast;

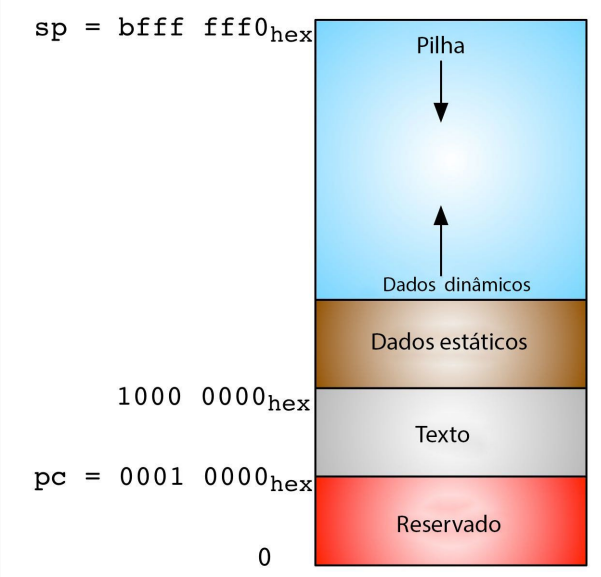
(3) smaller is faster; and

(4) good design demands good compromises.

Objetivos do processador RISC-V [Patterson and Waterman(2017)]

- Atender a todos os tamanhos de processadores, desde o minúsculo controlador embarcado até o computador de alto desempenho mais rápido;
- Funcionar bem com uma grande variedade de software e linguagens de programação populares;
- Acomodar todas as tecnologias de implementação: FPGAs, ASICs, chips customizados e até mesmo futuras tecnologias de dispositivos.
- Ser eficiente para todos os tipos de microarquitetura: controle microcodificado ou hardwired; pipelines em ordem, desacoplados ou desordenados; emissão de instrução única ou superescalar; e assim por diante;
- Apoiar ampla especialização para atuar como uma base para aceleradores customizados, que aumentam em importância à medida que a Lei de Moore se desvanece;
- Ser estável, ou seja, a ISA base não deve mudar. Mais importante, a ISA não pode ser descontinuada, como aconteceu no passado com as ISAs proprietárias, como a AMD Am29000, a Digital Alpha, a Digital VAX, o Hewlett Packard PA-RISC, Intel i860, Intel i960, Motorola 88000, e a Zilog Z8000.

Modelo de memória do RISC-V



Olá mundo!

```
#include <stdio.h>
int main()
{
   printf("Hello, %s\n", "world");
   return 0;
}
```

Figura 3.5: Programa Hello World escrito em C (`hello.c`).

Olá mundo!

```
  .text                   # Diretiva: insere a seção de texto
  .align 2                # Diretiva: alinha o código a 2 ^ 2 bytes
  .globl main             # Diretiva: declara o símbolo global principal
main:                     # rótulo para início da função main:
  addi sp,sp,-16          # aloca quadro de pilha
  sw   ra,12(sp)          # salva o endereço de retorno
  lui  a0,%hi(string1)    # endereço de computação de
  addi a0,a0,%lo(string1) #   string1
  lui  a1,%hi(string2)    # endereço de computação de
  addi a1,a1,%lo(string2) #   string2
  call printf             # chama a função printf
  lw   ra,12(sp)          # restaura o endereço de retorno
  addi sp,sp,16           # desaloca o quadro de pilha
  li   a0,0               # dá load no valor de retorno 0
  ret                     # retorna
  .section .rodata        # Diretiva: insira a seção de dados somente leitura
  .balign 4               # Diretiva: alinha a seção de dados a 4 bytes
string1:                  # rótulo para primeira string
  .string "Hello, %s!\n"  # Diretiva: string terminada com nulo
string2:                  # rótulo para segunda string
  .string "world"         # Diretiva: string terminada com nulo
```

Olá mundo!

```
00000000 <main>:
 0: ff010113  addi  sp,sp,-16
 4: 00112623  sw    ra,12(sp)
 8: 00000537  lui   a0,0x0
 c: 00050513  mv    a0,a0
10: 000005b7  lui   a1,0x0
14: 00058593  mv    a1,a1
18: 00000097  auipc ra,0x0
1c: 000080e7  jalr  ra
20: 00c12083  lw    ra,12(sp)
24: 01010113  addi  sp,sp,16
28: 00000513  li    a0,0
2c: 00008067  ret
```

Figura 3.7: Programa Hello World na linguagem de máquina do RISC-V (`hello.o`). As seis instruções que são posteriormente corrigidas pelo *linker* (locais 8 a 1c) têm zero em seus campos de endereço. A tabela de símbolos registra os rótulos e endereços de todas as instruções que necessitam ser editadas pelo *linker*.

Olá mundo!

```
000101b0 <main>:
   101b0: ff010113 addi sp,sp,-16
   101b4: 00112623 sw   ra,12(sp)
   101b8: 00021537 lui  a0,0x21
   101bc: a1050513 addi a0,a0,-1520 # 20a10 <string1>
   101c0: 000215b7 lui  a1,0x21
   101c4: a1c58593 addi a1,a1,-1508 # 20a1c <string2>
   101c8: 288000ef jal  ra,10450 <printf>
   101cc: 00c12083 lw   ra,12(sp)
   101d0: 01010113 addi sp,sp,16
   101d4: 00000513 li   a0,0
   101d8: 00008067 ret
```

Figura 3.8: Programa Hello World na linguagem de máquina do RISC-V após linkagem. Em sistemas Unix, o arquivo seria nomeado `a.out`.

Registradores de inteiros do RISC-V

Registrador	Nome ABI	Descrição	Preservado em toda a chamada?
x0	zero	Hard-wired zero	—
x1	ra	Endereço de retorno	Não
x2	sp	Ponteiro de pilha	Sim
x3	gp	Ponteiro global	—
x4	tp	Ponteiro de Thread	—
x5	t0	Registrador de link temporário/alternativo	Não
x6–7	t1–2	Temporários	Não
x8	s0/fp	Registrador salvo/Ponteiro de quadro	Sim
x9	s1	Registrador salvo	Sim
x10–11	a0–1	Argumentos de função / valores de retorno	Não
x12–17	a2–7	Argumentos de função	Não
x18–27	s2–11	Registradores salvos	Sim
x28–31	t3–6	Temporários	Não

Formatos de instrução do RISC-V

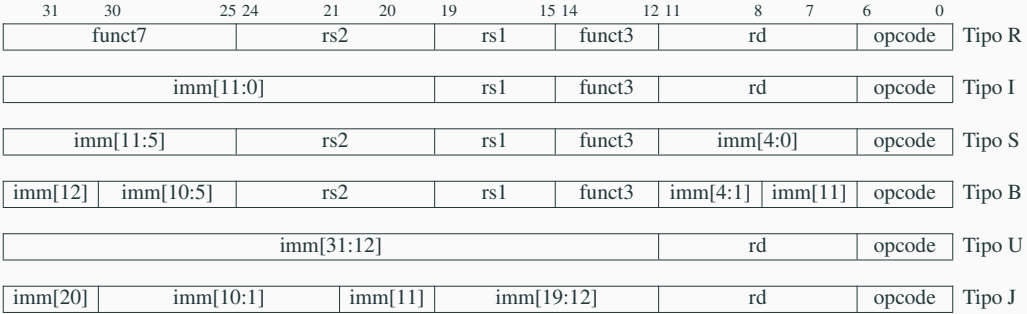


Figura 2.2: Formatos de instrução RV32I Nós rotulamos cada subcampo imediato com a posição do bit (imm $[x]$) no valor sendo produzido em imediato, ao invés da posição do bit no campo imediato da instrução como normalmente é feito. O capítulo 10 explica como as instruções de registrador de status de controle utilizam o formato Tipo-I de maneira um pouco diferente. (A Figura 2.2 de Waterman and Asanović 2017 é a base desta figura).

Formatos de instrução do RISC-V (Tipo R)

0000000	rs2	rs1	000	rd	0110011	R add
0100000	rs2	rs1	000	rd	0110011	R sub
0000000	rs2	rs1	001	rd	0110011	R sll
0000000	rs2	rs1	010	rd	0110011	R slt
0000000	rs2	rs1	011	rd	0110011	R sltu
0000000	rs2	rs1	100	rd	0110011	R xor
0000000	rs2	rs1	101	rd	0110011	R srl
0100000	rs2	rs1	101	rd	0110011	R sra
0000000	rs2	rs1	110	rd	0110011	R or
0000000	rs2	rs1	111	rd	0110011	R and

Formatos de instrução do RISC-V (Tipo I)

imm[11:0]		rs1	000	rd	0010011	I addi
imm[11:0]		rs1	010	rd	0010011	I slti
imm[11:0]		rs1	011	rd	0010011	I sltiu
imm[11:0]		rs1	100	rd	0010011	I xori
imm[11:0]		rs1	110	rd	0010011	I ori
imm[11:0]		rs1	111	rd	0010011	I andi
0000000	shamt	rs1	001	rd	0010011	I slli
0000000	shamt	rs1	101	rd	0010011	I srli
0100000	shamt	rs1	101	rd	0010011	I srai

Formatos de instrução do RISC-V (Tipo I)

imm[11:0]	rs1	000	rd	0000011	I lb
imm[11:0]	rs1	001	rd	0000011	I lh
imm[11:0]	rs1	010	rd	0000011	I lw
imm[11:0]	rs1	100	rd	0000011	I lbu
imm[11:0]	rs1	101	rd	0000011	I lhu

Formatos de instrução do RISC-V (Tipo S)

imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	S sb
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	S sh
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	S sw

Formatos de instrução do RISC-V (Tipo B)

imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	B beq
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	B bne
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	B blt
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	B bge
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	B bltu
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	B bgeu

Formatos de instrução do RISC-V (Tipo J/U)

imm[20|10:1|11|19:12]	rd	1101111	J jal

imm[31:12]	rd	0110111	U lui
imm[31:12]	rd	0010111	U auipc

Para saber mais e praticar...

- Simulador RARS com várias ferramentas
- Simulador online animado
- RISC-V International
  - Specifications
  - Cores & SoCs
  - Available Software

Bibliografia

Edson Borin.

An Introduction to Assembly Programming with RISC-V.

2021.

URL `https://riscv-programming.org/`.

David Patterson and Andrew Waterman.

The RISC-V Reader: an open architecture Atlas.

Strawberry Canyon, 2017.

URL `http://riscvbook.com/`.