Arquitectura de Computadores I

Código Máquina RISC-V

Miguel Barão

ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Resumo

Formatos binários RISC-V

Formatos das instruções

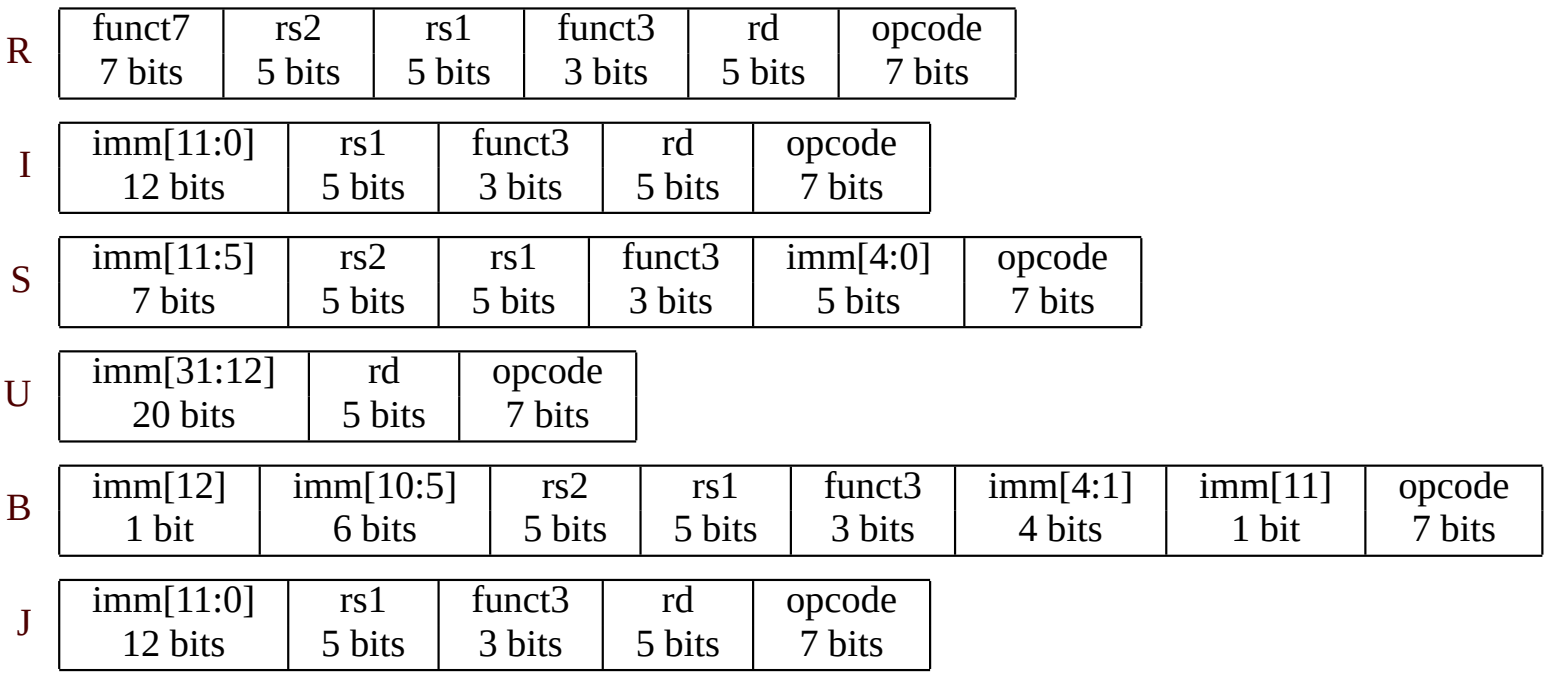
Existem 6 formatos binários:

Tipo R	Instruções *Register-Register*
Tipo I	Instruções *Register-Immediate* e *Load*
Tipo S	Instruções de *Store*
Tipo U	Instruções *Upper Immediate*
Tipo B	Instruções de *Branch*
Tipo J	Instrução *Jump and link*

O código máquina tem 32 bits e tem obrigatoriamente de ocupar posições de memória com endereços múltiplos de 4.

Formatos binários das instruções

Todas as instruções têm 32 bits



A posição dos registos `rd`, `rs1` e `rs2` é sempre a mesma para facilitar a descodificação pelo processador.

Instruções do tipo R

Instruções do tipo R

As instruções do tipo **R** são instruções aritméticas e lógicas que operam entre dois registos (*rs1 e rs2* - *register source* 1 e 2) e guardam o resultado num registo de destino (*rd* - *register destination*).

```
mnemónica rd, rs1, rs2
```

funct7 7 bits	rs2 5 bits	rs1 5 bits	funct3 3 bits	rd 5 bits	opcode 7 bits

- O número dos registos *rs1*, *rs2* e *rd* é codificado em 5 bits.
- O *opcode*, *funct3* e *funct7* estão na tabela do código máquina.

Código máquina de instruções do tipo R

funct7	rs2	rs1	funct3	rd	opcode	Assembly
0000000	rs2	rs1	000	rd	0110011	add rd, rs1, rs2
0100000	rs2	rs1	000	rd	0110011	sub rd, rs1, rs2
0000000	rs2	rs1	010	rd	0110011	slt rd, rs1, rs2
0000000	rs2	rs1	011	rd	0110011	sltu rd, rs1, rs2
0000000	rs2	rs1	111	rd	0110011	and rd, rs1, rs2
0000000	rs2	rs1	110	rd	0110011	or rd, rs1, rs2
0000000	rs2	rs1	100	rd	0110011	xor rd, rs1, rs2
0000000	rs2	rs1	001	rd	0110011	sll rd, rs1, rs2
0000000	rs2	rs1	101	rd	0110011	srl rd, rs1, rs2
0100000	rs2	rs1	101	rd	0110011	sra rd, rs1, rs2

	+0	+1	+2	+3	+4	+5	+6	+7
0	zero	ra	sp	gp	tp	t0	t1	t2
8	s0	s1	a0	a1	a2	a3	a4	a5
16	a6	a7	s2	s3	s4	s5	s6	s7
24	s8	s9	s10	s11	t3	t4	t5	t6

Código máquina de instruções do tipo R

funct7	rs2	rs1	funct3	rd	opcode	Assembly
0000000	rs2	rs1	000	rd	0110011	add rd, rs1, rs2
0100000	rs2	rs1	000	rd	0110011	sub rd, rs1, rs2
0000000	rs2	rs1	010	rd	0110011	slt rd, rs1, rs2
0000000	rs2	rs1	011	rd	0110011	sltu rd, rs1, rs2
0000000	rs2	rs1	111	rd	0110011	and rd, rs1, rs2
0000000	rs2	rs1	110	rd	0110011	or rd, rs1, rs2
0000000	rs2	rs1	100	rd	0110011	xor rd, rs1, rs2
0000000	rs2	rs1	001	rd	0110011	sll rd, rs1, rs2
0000000	rs2	rs1	101	rd	0110011	srl rd, rs1, rs2
0100000	rs2	rs1	101	rd	0110011	sra rd, rs1, rs2

	_000	_001	_010	_011	_100	_101	_110	_111
00	zero	ra	sp	gp	tp	t0	t1	t2
01	s0	s1	a0	a1	a2	a3	a4	a5
10	a6	a7	s2	s3	s4	s5	s6	s7
11	s8	s9	s10	s11	t3	t4	t5	t6

Exemplo: instruções do tipo R

```
mnemónica rd, rs1, rs2
```

funct7	rs2	rs1	funct3	rd	opcode
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

Exemplo: instruções do tipo R

```
mnemónica rd, rs1, rs2
```

funct7 7 bits	rs2 5 bits	rs1 5 bits	funct3 3 bits	rd 5 bits	opcode 7 bits

Exemplo

```
sub t2, t0, t1
```

funct7 0100000	rs2=t1=x6 00110	rs1=t0=x5 00101	funct3 000	rd=t2=x7 00111	opcode 0110011

- código máquina: `0100 0000 0110 0010 1000 0011 1011 0011`
- em hexadecimal: `0x406283b3`

Instruções do tipo I

Instruções do tipo I

Uma instrução do tipo **I** opera com um registo (*rs1* - *register source*) e um número fixo de 12 bits (*immediate*) que faz parte do código máquina da instrução. O resultado é guardado no registo de destino (*rd* - *register destination*).

```
mnemónica rd, rs1, imm
mnemónica rd, imm(rs1)
```

imm[11:0]	rs1	funct3	rd	opcode
12 bits	5 bits	3 bits	5 bits	7 bits

- Os registos são codificados em números de 5 bits.
- O *opcode* e *funct3* estão na tabela do código máquina.
- Os 12 bits da esquerda são o *immediate*.

Código máquina de instruções do tipo I

immediate	rs1	funct3	rd	opcode	Assembly
imm[11:0]	rs1	000	rd	1100111	jalr rd, imm(rs1)
imm[11:0]	rs1	010	rd	0000011	lw rd, imm(rs1)
imm[11:0]	rs1	001	rd	0000011	lh rd, imm(rs1)
imm[11:0]	rs1	101	rd	0000011	lhu rd, imm(rs1)
imm[11:0]	rs1	000	rd	0000011	lb rd, imm(rs1)
imm[11:0]	rs1	100	rd	0000011	lbu rd, imm(rs1)
imm[11:0]	rs1	000	rd	0010011	addi rd, rs1, imm
imm[11:0]	rs1	010	rd	0010011	slti rd, rs1, imm
imm[11:0]	rs1	011	rd	0010011	sltiu rd, rs1, imm
imm[11:0]	rs1	111	rd	0010011	andi rd, rs1, imm
imm[11:0]	rs1	110	rd	0010011	ori rd, rs1, imm
imm[11:0]	rs1	100	rd	0010011	xori rd, rs1, imm
0000000 | shamt	rs1	001	rd	0010011	slli rd, rs1, shamt
0000000 | shamt	rs1	101	rd	0010011	srli rd, rs1, shamt
0100000 | shamt	rs1	101	rd	0010011	srai rd, rs1, shamt

Exemplo 1: instrução do tipo I

```
mnemónica rd, rs1, imm
mnemónica rd, imm(rs1)
```

imm[11:0]	rs1	funct3	rd	opcode
12 bits	5 bits	3 bits	5 bits	7 bits

Exemplo 1: instrução do tipo I

```
mnemónica rd, rs1, imm
mnemónica rd, imm(rs1)
```

imm[11:0]	rs1	funct3	rd	opcode
12 bits	5 bits	3 bits	5 bits	7 bits

Exemplo

```
slti t2, t0, -1
```

imm=(-1)	rs1=t0=x5	funct3	rd=t2=x7	opcode
111111111111	00101	010	00111	0010011

- código máquina: `1111 1111 1111 0010 1010 0011 1001 0011`
- em hexadecimal: `0xfff2a393`

Exemplo 2: instrução do tipo I

```
mnemónica rd, rs1, imm
mnemónica rd, imm(rs1)
```

imm[11:0]	rs1	funct3	rd	opcode
12 bits	5 bits	3 bits	5 bits	7 bits

Exemplo

```
lw ra, 12(s1)
```

imm=12	rs1=s1=9	funct3	rd=ra=x1	opcode
000000001100	01001	010	00001	0000011

- código máquina: `1111 1111 1111 0010 1010 0011 1001 0011`
- em hexadecimal: `0xfff2a393`

Instruções do tipo S

Instruções do tipo S

Uma instrução do tipo **S** opera com um registo (*rs1* - *register source*) e um número fixo de 12 bits (*immediate*) que faz parte do código máquina da instrução. O valor do registo *rs2* é guardado em memória.

```
mnemónica rs2, imm(rs1)
```

imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
12 bits	5 bits	5 bits	3 bits	5 bits	7 bits

- Os registos são codificados em números de 5 bits.
- O *opcode* e *funct3* estão na tabela do código máquina.
- O *immediate* está divido em duas partes no código máquina.

Código máquina de instruções do tipo S

immediate	rs2	rs1	funct3	immediate	opcode	Assembly
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	sw rs2, imm(rs1)
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	sh rs2, imm(rs1)
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	sb rs2, imm(rs1)

Os 12 bits do *immediate* estão divididos em duas partes:

- os 5 bits menos significativos ocupam o espaço do *rd*;
- os 7 mais significativos do *immediate* estão do lado esquerdo do código máquina.

Exemplos de instruções do tipo S

```
mnemónica rs2, imm(rs1)
```

imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

Exemplos de instruções do tipo S

```
mnemónica rs2, imm(rs1)
```

imm[11:5] 7 bits	rs2 5 bits	rs1 5 bits	funct3 3 bits	imm[4:0] 5 bits	opcode 7 bits

Exemplo

```
sb t0, -3(t1)
```

onde o *immediate* é `-3 = 0b111111111101`.

imm[11:5]	rs2=t0=x5	rs1=t1=x6	funct3	imm[4:0]	opcode
1111111	00101	00110	000	11101	0100011

- código máquina: `1111 1110 0101 0011 0000 1110 1010 0011`
- em hexadecimal: `0xfe530ea3`

Instruções do tipo U

Instruções do tipo U

As instruções do tipo **U** têm como objectivo carregar valores grandes no registo de destino `rd`. `lui` carrega um valor absoluto e `auipc` carrega um valor relativo ao *program counter*.

upper-immediate	rd	opcode	Assembly
uimm[31:12]	rd	0110011	lui rd, uimm
uimm[31:12]	rd	0010011	auipc rd, uimm

Exemplo

```
lui t0, 0x12345
```

uimm[31:12]	rd	opcode
0001 0010 0011 0100 0101	00101	0110011

- código máquina: `0001 0010 0011 0100 0101 0010 1011 0011`
- em hexadecimal: `0x123452b3`

Instruções do tipo B

Instruções do tipo B

As instruções do tipo **B** realizam saltos condicionais dependendo do resultado de uma comparação entre dois registos.

1 bit	6 bits	rs2	rs1	funct3	4 bits	1 bit	opcode	Assembly
imm[12]	imm[10:5]	rs2	rs1	000	imm[4:1]	imm[11]	1100011	beq rs1, rs2, imm
imm[12]	imm[10:5]	rs2	rs1	001	imm[4:1]	imm[11]	1100011	bne rs1, rs2, imm
imm[12]	imm[10:5]	rs2	rs1	100	imm[4:1]	imm[11]	1100011	blt rs1, rs2, imm
imm[12]	imm[10:5]	rs2	rs1	101	imm[4:1]	imm[11]	1100011	bge rs1, rs2, imm
imm[12]	imm[10:5]	rs2	rs1	110	imm[4:1]	imm[11]	1100011	bltu rs1, rs2, imm
imm[12]	imm[10:5]	rs2	rs1	111	imm[4:1]	imm[11]	1100011	bgeu rs1, rs2, imm

- O *immediate* está separado em vários pedaços distribuídos pelos 32 bits do código máquina.
- O *immediate* corresponde ao valor que é necessário somar ao *program counter* para realizar o salto.
- O bit menos significativo do *immediate* é sempre `0` e não é representado no código máquina.

Exemplo de uma instrução do tipo B

Exemplo

```
LABEL:  addi t0, t0, 1
        sub t2, zero, t2
        blt t0, t1, LABEL    # ← qual o código máquina?
```

A instrução `blt` salta para 2 instruções atrás se a condição for verdadeira.
Como cada instrução ocupa 4 bytes, é necessário subtrair 8 bytes ao *program counter*, portanto o *immediate* é `-8`.
A instrução é `blt x5, x6, -8`.
`-8 = 0b11111111100`~~`0`~~, ou seja `1 1 111111 1100`.

imm[12]	imm[10:5]	rs2	rs1	funct3	imm[4:1]	imm[11]	opcode
1	111111	00110	00101	100	1100	1	1100011

- código máquina: `1111 1110 0110 0010 1100 1100 1110 0011`
- em hexadecimal: `0xfe62cce3`

Instruções do tipo J

Tipo J

Só existe uma instrução do tipo J.

1 bit	10 bits	1 bit	8 bits	rd	opcode	Assembly
imm[20]	imm[10:1]	imm[11]	imm[19:12]	rd	1101111	jal rd, imm

- O *immediate* está separado em vários pedaços distribuídos pelos 32 bits do código máquina.
- O *immediate* corresponde ao valor que é necessário somar ao *program counter* para realizar o salto.
- O bit menos significativo do *immediate* é sempre `0` e não é representado no código máquina.

Exemplo de uma instrução do tipo J

Exemplo

```
LABEL:  addi t0, t0, 1
        sub t2, zero, t2
        jal ra, LABEL   # ← qual o código máquina?
```

A instrução `jal` salta para 2 instruções atrás. Como cada instrução ocupa 4 bytes, é necessário subtrair 8 bytes ao *program counter*, portanto o *immediate* é `-8`. A instrução é `jal x1, -8`.

`-8 = 0b11111111111111111000`, ou seja `1 11111111 1 1111111100`.

imm[20]	imm[10:1]	imm[11]	imm[19:12]	rd	opcode
1	1111111100	1	11111111	00001	1101111

- código máquina: `1111 1111 1001 1111 1111 0000 1110 1111`
- em hexadecimal: `0xff9ff0ef`