RISC-V: The Free and Open RISC Instruction Set Architecture

Rodolfo Azevedo

MC404 – Organização Básica de Computadores e Linguagem de Montagem

http://www.ic.unicamp.br/~rodolfo/mc404

Conjuntos de Instruções - ISA

The portion of the computer that is visible to the programmer or the compiler writer.

Computer Architecture: A quantitative approach

An instruction set architecture (ISA) is an abstract model of a computer. It is also referred to as architecture or computer architecture.

Wikipedia

A contract HW and SW designers agreed to obey.

My one line definition

Arquitetura vs Microarquitetura

- Arquitetura é o modelo
  - X86, ARM, RISC-V, Power, Hack
- Microarquitetura é a implementação
  - Intel i7 geração 10, AMD Ryzen 3
- Conjunto de instruções pode ser visto como a borda
  - Pode facilitar ou dificultar a implementação!

Ambientes de Execução

- Plataforma *Bare metal*
- Sistema Operacional
- *Hypervisor*
- Emulador

RISC-V: Características Gerais

- Conjunto de instruções
  - Aberto
  - Modularizado
  - Virtualizável
- Espaço de endereçamento de 32, 64 ou 128 bits
- Registradores de 32, 64 ou 128 bits

Extensões

Extensão	Característica
RV32I	Conjunto base de instruções de inteiros de 32 bits
RV64I	Conjunto base de instruções de inteiros de 64 bits
M	Instruções de multiplicação e divisão de inteiros
A	Instruções atômicas
F	Instruções de ponto flutuante de precisão simples
D	Instruções de ponto flutuante de precisão dupla
G	Equivalente a IMAFD
Q	Instruções de ponto flutuante de precisão quádrupla
C	Instruções compactas

Os 32 registradores

Alias	Descrição
zero	Valor fixo em 0 - zero
t0-6	Valores temporários
s0-11	Valores salvos
a0-7	Parâmetros e valores de retorno de funções
ra	Endereço de retorno de função
sp	Apontador de pilha
gp	Apontador global
tp	Apontador de thread
pc	Contador de programa

Formatos Básicos das Instruções

- mnemônico rd, rs1, rs2
  - ADD s0, s1, s2
- mnemônico rd, rs1, imm
  - ADD s0, s1, 9
- mnemônico rd, imm
  - LUI s0, 9

Instruções Aritméticas

Instrução	Formato	Uso	
ADD	R	ADD	rd, rs1, rs2
ADD Immediate	I	ADDI	rd, rs1, imm
SUBtract	R	SUB	rd, rs1, rs2
Load Upper Imm	U	LUI	rd, imm
Add Upper Imm to PC	U	AUIPC	rd, imm

Exemplos

- x = y + z
  - add t0, t1, t2
- x = x + y
  - add t0, t0, t1
- x = y + 7
  - addi t0, t1, 7

- x = y – z
  - sub t0, t1, t2
- x = x – z
  - sub t0, t0, t2
- x = y – 7
  - addi t0, t1, -7

Instruções Lógicas

Instrução	Formato	Uso	
XOR	R	XOR	rd, rs1, rs2
XOR Immediate	I	XORI	rd, rs1, imm
OR	R	OR	rd, rs1, rs2
OR Immediate	I	ORI	rd, rs1, imm
AND	R	AND	rd, rs1, rs2
AND Immediate	I	ANDI	rd, rs1, imm

Exemplos

- x = y AND z
  - and t0, t1, t2
- x = x AND y
  - and t0, t0, t1
- x = y and 7
  - andi t0, t1, 7

- x = y XOR z
  - xor t0, t1, t2
- x = x OR z
  - or t0, t0, t2
- x = y OR 7
  - ori t0, t1, 7

Instruções de Deslocamento

Instrução	Formato	Uso	
Shift Left	R	SLL	rd, rs1, rs2
Shift Left Immediate	I	SLLI	rd, rs1, shamt
Shift Right	R	SRL	rd, rs1, rs2
Shift Right Immediate	I	SRLI	rd, rs1, shamt
Shift Right Arithmetic	R	SRA	rd, rs1, rs2
Shift Right Arith Imm	I	SRAI	rd, rs1, shamt

Exemplo

- x = 2; y = 16; z = -8
- x = x << 5
  - slli t0, t0, 5
- x = y >> 3
  - srli t0, t1, 3
- x = z >> 2
  - srai t0, t2, 2

Instruções de Memória

Instrução	Formato	Uso	
Load Byte	I	LB	rd, rs1, imm
Load Halfword	I	LH	rd, rs1, imm
Load Word	I	LW	rd, rs1, imm
Load Byte Unsigned	I	LBU	rd, rs1, imm
Load Half Unsigned	I	LHU	rd, rs1, imm
Store Byte	S	SB	rs1, rs2, imm
Store Halfword	S	SH	rs1, rs2, imm
Store Word	S	SW	rs1, rs2, imm

Exemplo

- Somar os dois primeiros elementos do vetor v e guardar na terceira posição do vetor
  - lw t1, t0, 0
  - lw t2, t0, 4
  - add t3, t1, t2
  - sw t3, t0, 8

Tamanhos de variáveis

Linguagem C	Tipo em RISC-V (32 bits)	Tamanho em bytes
bool	byte	1
char	byte	1
short	halfword	2
int	word	4
long	word	4
void *	unsigned word	4

Podem ser unsigned

- Memória endereçada em bytes

Instruções de Comparação

Instrução	Formato	Uso	
Set <	R	SLT	rd, rs1, rs2
Set < Immediate	I	SLTI	rd, rs1, imm
Set < Unsigned	R	SLTU	rd, rs1, rs2
Set < Imm Unsigned	I	SLTIU	rd, rs1, imm

Exemplo

- Como saber se i < j?
  - slt t0, t1, t2
- Se i < j
  - t0 = 1
- Caso contrário
  - t0 = 0

Instruções de Saltos Condicionais

Instrução	Formato	Uso	
Branch =	SB	BEQ	rs1, rs2, imm
Branch !=	SB	BNE	rs1, rs2, imm
Branch <	SB	BLT	rs1, rs2, imm
Branch >=	SB	BGE	rs1, rs2, imm
Branch < Unsigned	SB	BLTU	rs1, rs2, imm
Branch >= Unsigned	SB	BGEU	rs1, rs2, imm

Exemplo

- Se x == 0, some z = y+5, caso contrário z = y+7
  - beq t0, zero, e_zero
  - addi t1, t2, 7
  - j fim
  - e_zero: addi t1, t2, 5
  - fim

Instruções de Salto

Instrução	Formato	Uso	
J & L	UJ	JAL	rd, imm
J & Link Register	UJ	JALR	rd, rs1, imm

Instrução para constantes 32 bits

Instrução	Formato	Uso	
LUI	U	lui	rd, imm

`lui rd, constant`

- Copia a constante de 20 bits nos bits [31:12] de rd
- Estende o bit 31 para bits [63:32]
- Limpa os bits [11: 0] de rd para 0

```
lui x19, 976  // 0x003D0
addi x19,x19,128  // 0x500
```

0000 0000 0011 1101 0000	0000 0000 0000
0000 0000 0011 1101 0000	0101 0000 0000

Instruções de Sistema

Instrução	Formato	Uso
System CALL	I	SCALL
System BREAK	I	SBREAK

Chamadas de Sistema

Syscall	**id (coloque em t0)**	**Descrição**
Imprime inteiro	1	Imprime o valor de a0 no console como inteiro
Imprime caracter	2	Imprime o valor de a0 no console como caracter
Imprime string	3	Imprime string a0 com tamanho a1 no console
Lê inteiro	4	Lê inteiro do console e retorna em a0
Lê caracter	5	Lê caracter do console e retorna o valor ASCII
Lê string	6	Lê string do tamanho solicitado em a1 e armazena no endereço indicado em a0
SBRK	7	Aloca a0 bytes de memória e retorna ponteiro para o bloco de memória. Desalocar memória com a0 negativo.

Algumas pseudo-instruções

Pseudo Instrução	Descrição	Conversão
li a0, constante	Atribui uma constante ao registrador	Usa lui + addi para compor a constante
la a0, msg + 1	Carrega o endereço de msg + 1 no registrador	Usa auipc, mv e ld se necessário
mv a0, t0	Copia o valor de t0 em a0	addi a0, t0, zero
call função	Transfere o controle para a função	Converte para jal ou jalr conforme necessário
nop	Não realiza operação nenhuma	addi zero, zero, 0
j destino	Salta para o endereço destino	jal zero, destino
ret	Retorna de uma função	jalr zero, ra, 0
not rd, rs	Inverte os bits de um registrador	xori rd, x0, rs

if then else

```
if x == 5:
   a += 7
else:
   a += 15
```

if then else

```
if x == 5:
   a += 7
else:
   a += 15
```

```
main:
  addi t1, zero, 9          # x em t1
  add t2, zero, zero        # a em t2
  addi t0, zero, 5          # 5 em t0
  bne t1, t0, else

  addi t2, t2, 7
  j fim
else:
  addi t2, t2, 15

fim:
  jr        ra
```

Laço while

```
while x != y
{
   x += 2;
   y += 3;
}
```

Laço while

```
while x != y
{
    x += 2;
    y += 3;
}
```

```
main:
  addi t0, zero, 20        # x em t0
  addi t1, zero, 10        # y em t1

loop:
  beq t0, t1, fim
  addi t0, t0, 2
  addi t1, t1, 3
  j loop

fim:
  jr ra
```

Laço for

```
for (i = 0; i < 100; i ++)
  a += i
```

Laço for

```
for (i = 0; i < 100; i ++)
  a += i
```

```
main:
  addi t0, zero, 0        # a em t0
  addi t1, zero, 0        # i em t1
  addi t2, zero, 100      # 100 em t2

for:
  bge t1, t2, fim
  add t0, t0, t1
  addi t1, t1, 1
  j for

fim:
  jr ra
```