Arquitetura de Computadores

Reduced Instruction Set Computer: Estudo sobre RISC-V e Introdução ao Rocket-Chip

Rodolfo Labiapari Mansur Guimarães

rodolfolabiapari@decom.ufop.br Lattes: http://goo.gl/MZv4Dc Departamento de Computação – Universidade Federal de Ouro Preto Ouro Preto - MG – Brasil

Última Atualização: 18 de julho de 2017.



- 🚺 Formato das Instruções Bases do RISC-V
 - RV32I
 - Comparação de Instruções entre diferentes Arquiteturas e Expansões

Formato das Instruções Bases

■ Na base do ISA RISC-V, existe quatro formato de instruções principais

31	25	5 24 2	0 19	15	14 12	2 11	7 6	0
func	t7	rs2	rs1		funct3	rd	opcode	R-type
	$\operatorname{imm}[11:$	0]	rs1		funct3	$_{ m rd}$	opcode	I-type
imm[1	[1:5]	rs2	rs1		funct3	imm[4:0]	opcode	S-type
		imm[31:12]	2]			rd	opcode	U-type

- Alguma observações
 - É possível ver que *rd*, *rs1* e *rs2* estão na mesma posição
 - Isso é um método para facilitação da decodificação.
 - Imediatos estão sempre no final da instrução;

- Formato das Instruções Bases do RISC-V
 - RV32I
 - Comparação de Instruções entre diferentes Arquiteturas e Expansões

Register-Immediate

31	20 19	15	14 1	2 11	7 6	0
$\mathrm{imm}[11:0]$		rs1	funct3	rd	opcode	
12		5	3	5	7	
I-immediate $[11:0]$		src	ADDI/SLTI[U]	dest	OP-IMM	
$\hbox{I-immediate} \hbox{\small [11:0]}$		src	ANDI/ORI/XC	$ m RI \ dest$	OP-IMM	

■ *Opcode*: Local onde fica a operação principal;

■ Funct3: Especificação da operação.

Register-Register

31	25	24 20	19 15	5 14 12	11 7	6	0
:	funct7	rs2	rs1	funct3	rd	opcode	
	7	5	5	3	5	7	
(0000000	src2	$\operatorname{src1}$	ADD/SLT/SLT	$_{ m U-dest}$	OP	
(0000000	${ m src2}$	$\operatorname{src1}$	AND/OR/XOR	$_{ m dest}$	OP	
(0000000	${ m src2}$	$\operatorname{src1}$	SLL/SRL	dest	OP	
(0100000	${ m src2}$	$\operatorname{src1}$	SUB/SRA	dest	OP	

Opcode: Local onde fica a operação principal;

■ Funct3: Especificação da operação;

Funct7: Especificação da operação;

31 2	0 19 15	5 14 12	11 7	6 0
$\mathrm{imm}[11:0]$	rs1	funct3	$^{\mathrm{rd}}$	opcode
12	5	3	5	7
0	0	\mathbf{ADDI}	0	OP-IMM

Jump and Link¹

31	30		21	20	19	12 11	7 6	0
imm[20]		imm[10:1]		imm[11]	imm[19:12]	rd	opcode	
1		10		1	8	5	7	_
		offset[2	20:1	.]		dest	${f JAL}$	



¹ Jump and Link é um jump com endereço de retorno.

Conditional Branches

31	30 25	24 20	19 15	14 12	2 11 8	7	6	0
imm[12]	$\mathrm{imm}[10.5]$	rs2	rs1	funct3	imm[4:1]	imm[11]	opcode	
1	6	5	5	3	4	1	7	
offset	[12,10:5]	src2	$\operatorname{src}1$	BEQ/BNE	offset[1]	1,4:1]	BRANCH	
offset	[12,10:5]	src2	$\operatorname{src1}$	BLT[U]	offset[1]	1,4:1]	BRANCH	
offset	[12,10:5]	src2	$\operatorname{src}1$	BGE[U]	offset[1]	1,4:1	BRANCH	

Load e Store

31	20 19	15 14	12 11	. 7	6	0
$\operatorname{imm}[11:0]$	rs1	fu	nct3	$^{\mathrm{rd}}$	opcode	
12	5		3	5	7	
$ ext{offset}[11:0]$	base	e wi	dth	dest	LOAD	

31	25 24	20	19	15	14	12	11	7 6		0
imm[11:5]		rs2	rs1		func	t3	imm[4:0]		opcode	
7		5	5		3		5		7	
offset[11:5]		src	base		widt	\mathbf{h}	offset[4:0]	S	STORE	

- 🚺 Formato das Instruções Bases do RISC-V
 - RV32I
 - Comparação de Instruções entre diferentes Arquiteturas e Expansões

RV32I vs. RV64I vs. Extensão C (compacta)

Register-Immediate

31		20 19		15 1	14	12 1	11	7 6		0
	$\mathrm{imm}[11:0]$		rs1		funct3		$^{\mathrm{rd}}$		opcode	
	12		5		3		5		7	
	I-immediate [11:0]		src	1	ADDI/SLTI[U	U]	dest		OP-IMM	
	I-immediate [11:0]		src	A	NDI/ORI/X	OR	$_{ m I}$ dest		OP-IMM	
31		20 19		15	14	12	11	7 6		0
	$\mathrm{imm}[11:0]$		rs1		${ m funct3}$		$^{ m rd}$		opcode	
	12		5		3		5		7	
	I-immediate [11:0]		src		ADDIW		dest		OP-IMM-32	
31		20 19		15	14	12	11	7 6		0
	imm[11:0]		rs1		funct3		$^{\mathrm{rd}}$		opcode	
	12		5		3		5		7	
	I-immediate [11:0]		src		ADDIW		dest		OP-IMM-32	

RV32I vs. Extensão F

Load e Store

3	1	:	20 19		15	14	12	11	7 6		0
	$_{ m imm}[11:$	0]		rs1		fun	ct3	rd		opcode	
	12			5		3		5		7	
	offset[1]	1:0]		base		wid	$^{\mathrm{th}}$	dest		LOAD	
3	1 25	24 2	0 19		15	14	12	11	7 6		0
3	$\frac{1}{\text{imm}[11:5]}$	rs2	0 19	rs1	15	func		imm[4:0]	7 6	opcode	0
3			0 19	rs1	15				7 6	opcode 7	0

31		20 19)	15	14 12	11	7 6		0
$_{ m imm}[11:$	0]		rs1		width	$^{\mathrm{rd}}$		opcode	
12			5		3	5		7	
offset[1]	1:0]		base		\mathbf{W}	dest		LOAD-FP	
31 25	24	20 19		15	14 12	11	7 6		0
$_{ m imm[11:5]}$	rs2		rs1		width	$_{ m imm}[4:0]$		opcode	
7	5		5		3	5		7	
offset[11:5]	src		base		\mathbf{W}	offset[4:0]	4 1	STORE-FP	<u>-</u> - <u>-</u> -

- 🚺 Formato das Instruções Bases do RISC-V
 - RV32I
 - Comparação de Instruções entre diferentes Arquiteturas e Expansões

Rocket Chip

Introdução

- Rocket Chip permite gerar diferentes configurações de SoC².
- Essas configurações são especificadas por meio de parâmetros em linguagem Chisel³ que podem ser alterados livremente:
 - E com isso, pode-se selecionar o que quer gerar como chip.
- Também possui um emulador C++ de RLT⁴.

² System-on-a-chip: se refere a todos os componentes de um computador/sistema eletrônico, em um circuito integrado.

³ Constructing Hardware In a Scala Embedded Language.

⁴Register transfer level.

Rocket Chip

Introdução

- Rocket Chip é formado por vários submódulos:
 - Chisel: Linguagem HDL para desenvolvimento de RTL;
 - Rocket: Código-fonte dos *cores* e *caches* do Rocket;
 - Dramsim2: Simulador de tempos de acesso à DRAM;
 - Entre outros.
- Algumas configurações possíveis:
 - Quantidade memória física;
 - Quantidade de bits do endereço virtual;
 - Parâmetros de interface de memória;
 - E outros.



Rocket Chip

Simulando uma Configuração Predefinida

- Para uma simulação, deve-se ir ao respectivo diretório.
- O emulador C++ RTL é executado do diretório /emulator.
- Neste diretório, um comando de execução default já é conhecido para verificar se a ferramenta foi instalada com sucesso. Possui a seguinte sintaxe make run-asm-tests.
- Podemos também construir itens com configurações de pequeno porte utilizando a especificação CONFIG=ExampleSmallConfig
 - E para testar, utiliza-se
 make CONFIG=ExampleSmallConfig run-asm-tests.

Dúvidas, Sugestões ou Reclamações?

■ rodolfolabiapari@decom.ufop.br

https://www.guerrillamail.com/

Arquitetura de Computadores

Reduced Instruction Set Computer: Estudo sobre RISC-V e Introdução ao Rocket-Chip

Rodolfo Labiapari Mansur Guimarães

rodolfolabiapari@decom.ufop.br Lattes: http://goo.gl/MZv4Dc Departamento de Computação – Universidade Federal de Ouro Preto Ouro Preto - MG – Brasil

Última Atualização: 18 de julho de 2017.

