SI100B EE Part Project • Tutorial 2

RISC-V Instruction Overview

@Yunxiang He, heyx1@shanghaitech.edu.cn
@Quanyu Chen, chenqy2022@shanghaitech.edu.cn
@Yongzhi Zhang, zhangyz2022@shanghaitech.edu.cn
@Chaofan Li, lichf1@shanghaitech.edu.cn
@Qihan Ding, dingqh@shanghaitech.edu.cn

Contents

1. Document Overview ->
2. RISC-V Instruction ->
3. R-Type Instructions ->
4. I-Type Instructions ->
5. Example ->

1. Document Overview

1. Document Overview

- Document will be released in the week!
- Let's have a look at the document first.

☒ Document Statement

这是我们第一次撰写类似文档。如果你认为有任何描述不清晰的部分（不包括知识性问题），你可以直接联系助教，我们会尽快进行调整和修改。
在第三周之后，我们会发布最终文档，**一切以最终文档为准**。

2. RISC-V Instruction

2. RISC-V Instruction

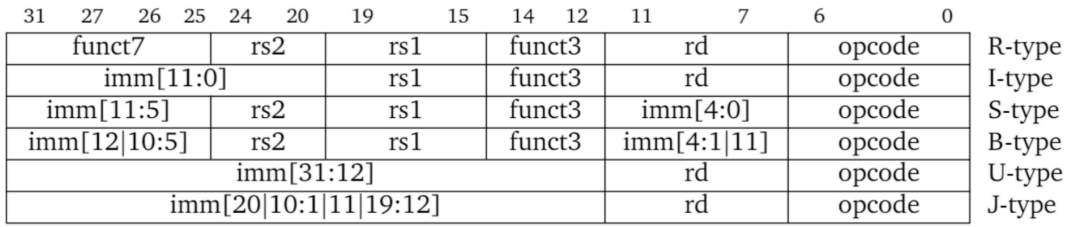
这节课我们首先介绍 RISC-V 指令集中最基本的两种指令格式：R-Type 和 I-Type。

- R-Type 指令：Register Type，用于寄存器之间的运算。
- I-Type 指令：Immediate Type，用于立即数和寄存器之间的运算。

这节课中，我们会举一些例子，带大家熟悉这两种指令的使用方式和格式。

2. RISC-V Instruction

- 下图位 RISC-V 的指令格式。
  - 每一行表示一个 32 位的指令类型。
  - 上面标的数字是指令的每个字段和其对应的位数。
- 右图为 RISC-V 的指令集。
  - 每一行表示一个指令。
  - 这部分表格则主要描述了指令的具体功能，和其对应的字段的值。



Inst	Name	FMT	Opcode	funct3	funct7	Description (C)	Note
add	ADD	R	0110011	0x0	0x00	rd = rs1 + rs2	
sub	SUB	R	0110011	0x0	0x20	rd = rs1 - rs2	
xor	XOR	R	0110011	0x4	0x00	rd = rs1 ^ rs2	
or	OR	R	0110011	0x6	0x00	rd = rs1 | rs2	
and	AND	R	0110011	0x7	0x00	rd = rs1 & rs2	
sll	Shift Left Logical	R	0110011	0x1	0x00	rd = rs1 << rs2	
srl	Shift Right Logical	R	0110011	0x5	0x00	rd = rs1 >> rs2	
sra	Shift Right Arith*	R	0110011	0x5	0x20	rd = rs1 >> rs2	msb-extends
slt	Set Less Than	R	0110011	0x2	0x00	rd = (rs1 < rs2)?1:0	
sltu	Set Less Than (U)	R	0110011	0x3	0x00	rd = (rs1 < rs2)?1:0	zero-extends
addi	ADD Immediate	I	0010011	0x0		rd = rs1 + imm	
xori	XOR Immediate	I	0010011	0x4		rd = rs1 ^ imm	
ori	OR Immediate	I	0010011	0x6		rd = rs1 | imm	
andi	AND Immediate	I	0010011	0x7		rd = rs1 & imm	
slli	Shift Left Logical Imm	I	0010011	0x1	imm[5:11]=0x00	rd = rs1 << imm[0:4]	
srli	Shift Right Logical Imm	I	0010011	0x5	imm[5:11]=0x00	rd = rs1 >> imm[0:4]	
srai	Shift Right Arith Imm	I	0010011	0x5	imm[5:11]=0x20	rd = rs1 >> imm[0:4]	msb-extends
slti	Set Less Than Imm	I	0010011	0x2		rd = (rs1 < imm)?1:0	
sltiu	Set Less Than Imm (U)	I	0010011	0x3		rd = (rs1 < imm)?1:0	zero-extends
lb	Load Byte	I	0000011	0x0		rd = M[rs1+imm][0:7]	
lh	Load Half	I	0000011	0x1		rd = M[rs1+imm][0:15]	
lw	Load Word	I	0000011	0x2		rd = M[rs1+imm][0:31]	
lbu	Load Byte (U)	I	0000011	0x4		rd = M[rs1+imm][0:7]	zero-extends
lhu	Load Half (U)	I	0000011	0x5		rd = M[rs1+imm][0:15]	zero-extends
sb	Store Byte	S	0100011	0x0		M[rs1+imm][0:7] = rs2[0:7]	
sh	Store Half	S	0100011	0x1		M[rs1+imm][0:15] = rs2[0:15]	
sw	Store Word	S	0100011	0x2		M[rs1+imm][0:31] = rs2[0:31]	
beq	Branch ==	B	1100011	0x0		if(rs1 == rs2) PC += imm	
bne	Branch !=	B	1100011	0x1		if(rs1 != rs2) PC += imm	
blt	Branch <	B	1100011	0x4		if(rs1 < rs2) PC += imm	
bge	Branch ≥	B	1100011	0x5		if(rs1 >= rs2) PC += imm	
bltu	Branch < (U)	B	1100011	0x6		if(rs1 < rs2) PC += imm	zero-extends
bgeu	Branch ≥ (U)	B	1100011	0x7		if(rs1 >= rs2) PC += imm	zero-extends
jal	Jump And Link	J	1101111			rd = PC+4; PC += imm	
jalr	Jump And Link Reg	I	1100111	0x0		rd = PC+4; PC = rs1 + imm	
lui	Load Upper Imm	U	0110111			rd = imm << 12	
auipc	Add Upper Imm to PC	U	0010111			rd = PC + (imm << 12)	
ecall	Environment Call	I	1110011	0x0	imm=0x0	Transfer control to OS	
ebreak	Environment Break	I	1110011	0x0	imm=0x1	Transfer control to debugger	

3. R-Type Instructions

3. R-Type Insturctions

R-Type 的字段格式

31-25	24-20	19-15	14-12	11-7	6-0
`funct7`	`rs2`	`rs1`	`funct3`	`rd`	`opcode`
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

作为执行寄存器之间的运算的指令，R-Type 指令的指定 `rs1` 和 `rs2` 两个寄存器，然后将运算结果存储到 `rd` 寄存器中。

R-Type 的字段内容细节

字段	位数	描述
funct7	7 bits	功能码，用于指定具体的操作类型
rs2	5 bits	源寄存器2
rs1	5 bits	源寄存器1
funct3	3 bits	功能码，用于指定操作类型
rd	5 bits	目标寄存器，存储运算结果
opcode	7 bits	操作码，指定指令类型

3. R-Type Instructions

Example: `and` 指令

```
and x10, x1, x7
```

- `opcode`: `0110011` (R-Type); `funct3`: `111` (and); `funct7`: `0000000`
- `rs1`: `00001` (x1); `rs2`: `00111` (x7); `rd`: `01010` (x10)

31-25	24-20	19-15	14-12	11-7	6-0
funct7	rs2	rs1	funct3	rd	opcode
0000000	00111	00001	111	01010	0110011

4. I-Type Instructions

4. I-Type Instructions

I-Type 的字段格式

31-20	19-15	14-12	11-7	6-0
`imm [11:0]`	`rs1`	`funct3`	`rd`	`opcode`
12 bits	5 bits	3 bits	5 bits	7 bits

I-Type 指令用于立即数和寄存器之间的运算。这种指令的格式中，`rs1` 为源寄存器，`imm` 为立即数，`rd` 为目标寄存器。

- 立即数：立即数是在立即寻址方式指令中给出的数，通常用于指定一个常数。

I-Type 的字段内容细节

字段	位数	描述
imm	12 bits	立即数
rs1	5 bits	源寄存器1
funct3	3 bits	功能码，用于指定操作类型
rd	5 bits	目标寄存器，存储运算结果
opcode	7 bits	操作码，指定指令类型

4. I-Type Instructions

Example: `addi` 指令

```
addi x10, x1, 10
```

- `opcode`：`0010011` (I-Type); `funct3`：`000` (addi)
- `rs1`：`00001` (x1); `imm`：`000000000010` (10); `rd`：`01010` (x10)
- 注意：`imm` 是 12 位的有符号数，所以需要进行符号扩展，其二进制表示是 `000000000010`

31-20	19-15	14-12	11-7	6-0
imm [11:0]	rs1	funct3	rd	opcode
000000000010	00001	000	01010	0010011

5. Example

5. Example: An RISC-V Program

初始状态

按顺序执行以下指令：

```
addi x1, x2, -1
or x2, x2, x1
add x3, x1, x2
slt x4, x3, x1
sra x5, x3, x4
sub x0, x5, x4
```

右图为程序执行过程中寄存器的状态



5. Example: An RISC-V Program

`addi x1, x2, -1`

- 将 `x2` 的值加上 `-1`，存入 `x1` 中
- `x2` 的初始值为 `0`，所以 `x1` 的值为 `-1`
- `-1` 的三十二位二进制表示为 `0xFFFFFFFF`



5. Example: An RISC-V Program

`or x2, x2, x1`

- 将 `x2` 和 `x1` 的值进行`OR` 运算，存入 `x2` 中
- `x2` 的值为 `0`
- `x1` 的值为 `-1`
- 所以 `x2` 的值为 `0xFFFFFFFF`



5. Example: An RISC-V Program

`add x3, x1, x2`

- 将 `x1` 和 `x2` 的值相加，存入 `x3` 中
- `x1` 的值为 `-1`
- `x2` 的值为 `-1`
- 所以 `x3` 的值为 `-2`
- `-2` 的三十二位二进制表示为 `0xFFFFFFFE`，即 `x3` 的值



5. Example: An RISC-V Program

`slt x4, x3, x1`

- 将 `x3` 和 `x1` 的值进行有符号数的大小比较，如果 `x3` 小于 `x1`，则 `x4` 的值为 `1`，否则为 `0`
- `x3` 的值为 `-2`
- `x1` 的值为 `-1`
- 所以 `x4` 的值为 `1`
- 值得注意的是，如果是无符号数的比较，那么 `slt` 指令会被替换为 `sltu` 指令，此时结果为 `0`



5. Example: An RISC-V Program

`srl x5, x3, x4`

- 将 `x3` 的值**逻辑右移** `x4` 位，存入 `x5` 中
- `x3` 的值为 `-2`（`0xFFFFFFFE`）
- `x4` 的值为 `1`
- 所以 `x5` 的值为 `0x7FFFFFFF`
- 值得注意的是逻辑右移和算术右移的区别：
  - 逻辑右移在左边补 `0`
  - 算术右移在左边补符号位



SI100B EE Part Project · Tutorial 2

@ Presented by **Yunxiang He**
@ ShanghaiTech University