Report microprocessor architecture

Yannis Kedadry   Oscar Garnier   Antoine Anastassiades

January 20, 2023

1 Architecture choice

We've chosen to make our project using the 32 bits RISC-V's architecture. We'll use 32 registers.

2 Instruction Set

We'll use a total of 28 instructions to make operations between unsigned integers. We'll encode our instructions in different ways according to the instruction's format (operation with immediates I, with jumps U, only on registers R). The different operations we'll use are:

- *add*, *sub* (type R) and *addi* (type I). Our arithmetical instructions.

- *sll*, *srl*, *sra* Of type R and their equivalent of type I to make bitwise shifts.

- logical operations *and*, *or* and *xor* of type R and their equivalent of type I.

- tests operations *slt* and *slti*.

- *lui* and *auipc* are special instructions of type U, used to move in the stack.

- *beq*, *bne*, *blt*, *blti* (type I) and *bge* for conditional branching. Except for *blti*, they're of type R.

- *jal* and *jalr* of type U for unconditional jumps.

- *lw* (type I) and *sw* (type R) load and store datas from and to memory.

Here are the different encoding for the different instruction formats. These formats help us manipulate immediates of big size. It's probably not mandatory but it can come handy in some particular cases.

rd represents the destination register and *rsX* the sources ones.

	31 20	19 15	14 10	9 5	4 0
R	immédiat	rs2	rs1	rd	opcode
I	immédiat		rs	rd	opcode
U	immédiat			rd	opcode

3 Opcodes

Our instructions will have the following opcodes:

	000	001	010	011	100	101	110	111
00	Addi		Srli	Srai	Slli	Andi	Ori	Xori
01	Add	Sub	Srl	Sra	Sll	And	Or	Xor
10	Lw	Sw		Beq	Bne	Blt	Blti	Bge
11	Lui	Auipc	Jal	Jalr			Slt	Slti

4 TLDR

instruction	format	opcode	usage	result
Addi	I	00000	addi x0 x1 1	x0 = x1 + 1
Srli	I	00010	srli x0 x1 1	x0 = x1 >> 1 (0 extended)
Srai	I	00011	srai x0 x1 1	x0 = x1 >> 1 (sign extended)
Slli	I	00100	slli x0 x1 1	x0 = x1 << 1
Andi	I	00101	andi x0 x1 1	x0 = x1 & 1
Ori	I	00110	ori x0 x1 1	x0 = x1 | 1
Xori	I	00111	xori x0 x1 1	x0 = x1 ⊕ 1
Add	R	01000	add x0 x1 x2 1	x0 = x1 + x2
Sub	R	01001	sub x0 x1 x2 1	x0 = x1 − x2
Srl	R	01010	srl x0 x1 x2 1	x0 = x1 >> x2 (0 extended)
Sra	R	01011	sra x0 x1 x2 1	x0 = x1 >> x2 (sign extended)
Sll	R	01100	sll x0 x1 x2 1	x0 = x1 << x2
And	R	01101	and x0 x1 x2 1	x0 = x1 & x2
Or	R	01110	or x0 x1 x2 1	x0 = x1 | x2
Xor	R	01111	xor x0 x1 x2 1	x0 = x1 ⊕ x2
Lw	I	10000	lw x0 x1 1	x0 = Ram(x1)
Sw	R	10001	sw x0 x1 x2 1	Ram(x1) = x2
Beq	R	10011	beq x0 x1 x2 1	rdi = (x1 = x2) ? rdi + 1 : rdi + 4
Bne	R	10100	bne x0 x1 x2 1	rdi = (x1 ≠ x2) ? rdi + 1 : rdi + 4
Blt	R	10101	blt x0 x1 x2 1	rdi = (x1 < x2) ? rdi + 1 : rdi + 4
Bge	R	10111	bge x0 x1 x2 1	rdi = (x1 ≥ x2) ? rdi + 1 : rdi + 4
Jal	U	11010	jal x0 1	rdi = 1
Slt	R	11110	slt x0 x1 x2 3	x0 = (x1 < x2) ? 1 : 0
Slti	I	11111	slti x0 x1 3	x0 = (x1 < 3) ? 1 : 0