Computer Architecture
SimpleRISC Specification

Memory Model

The memory is byte-addressable. Multi-byte entities are stored in little endian form. The memory contains the program instructions, the static data, and the stack.

Registers

There are a total of 16 registers: `r0` to `r15`. Each register is 4 bytes wide. 16 registers require 4 bits for encoding. `r0` is encoded as `0000`, `r1` as `0001`, and so on.

Table 1: Registers in the custom ISA

Register	Purpose
`r0` to `r13`	General purpose
`r14` / `sp`	Stack Pointer
`r15` / `ra`	Return Address Register
`PC`	Program Counter

Instruction Formats

Table 2: Instruction formats in the SimpleRISC ISA

register type

`opcode`	`immediate`	`rd`	`rs1`	`rs2`	unused
5 bits	1 bit	4 bits	4 bits	4 bits	14 bits

immediate type

`opcode`	`immediate`	`rd`	`rs1`	immediate
5 bits	1 bit	4 bits	4 bits	18 bits

branch type

`opcode`	offset
5 bits	27 bits

31st bit	0th bit

Instructions

Table 3: Instructions in the *SimpleRISC* ISA

Operation	Opcode	Format
add	00000	add rd, rs1, (rs2 / imm)
sub	00001	sub rd, rs1, (rs2 / imm)
mul	00010	mul rd, rs1, (rs2 / imm)
div	00011	div rd, rs1, (rs2 / imm)
mod	00100	mod rd, rs1, (rs2 / imm)
cmp	00101	cmp rs1, (rs2 / imm)
and	00110	and rd, rs1, (rs2 / imm)
or	00111	or rd, rs1, (rs2 / imm)
not	01000	not rd, (rs2 / imm)
mov	01001	mov rd, (rs2 / imm)
lsl	01010	lsl rd, rs1, (rs2 / imm)
lsr	01011	lsr rd, rs1, (rs2 / imm)
asr	01100	asr rd, rs1, (rs2 / imm)
nop	01101	nop
ld	01110	ld rd, imm[rs1] (rd ← [rs1+imm])
st	01111	st rd, imm[rs1] ([rs1+imm] ← rd)
beq	10000	beq offset
bgt	10001	bgt offset
b	10010	b offset
call	10011	call offset
ret	10100	ret
end	10101	end

Example Program

Factorial by Iteration

Consider the following program to compute the factorial of a number stored in memory at address `r0`. Assume that the number is greater than 2. Save the result in memory at the address `r0` + 4.

```
.main:
        ld r2, [r0]
        mov r1, 1
.loop:
        mul r1, r1, r2
        sub r2, r2, 1
        cmp r2, 1
        bgt .loop
        st r1, 4[r0]
        end
```