# Reduced RISC-V ISA

## Storage space

In this exercise, RISC-V uses

- 4 Gigabytes ($2^{32}$ bytes) of memory
- A register file of 32 registers of 32 bits each. They are called x0, x1, up to x31. x0 always contains a 0 and cannot be changed.

## Reduced instruction set description

| Transfer instructions | | | |
|---|---|---|---|
| Instruction | Mnemonic | Operation | Example |
| Load word | lw rd, offset(rs1) | rd ← mem[rs1 + offset] | lw x5, 32(x0) x5 ← mem[32+0] |
| Store word | sw rs2, offset(rs1) | mem[rs1 + offset] ← rs2 | sw x4, 16(x2) mem[16+x2] ← x4 |
| Arithmetic-logical instructions | | | |
| Instruction | Mnemonic | Operation | Example |
| Add Immediate | addi rd, rs1, imm | rd ← rs1 + imm | addi x5, x2, -15 x5 ← x2 + (-15) |
| Add | add rd, rs1, rs2 | rd ← rs1 + rs2 | add x2, x3, x4 x2 ← x3 + x4 |
| Subtract | sub rd, rs1, rs2 | rd ← rs1 - rs2 | sub x4, x5, x6 x4 ← x5 - x6 |
| And Immediate | andi rd, rs1, imm | rd ← rs1 ∧ imm | andi x5, x2, 2 x5 ← x2 ∧ 2 |
| And | and rd, rs1, rs2 | rd ← rs1 ∧ rs2 | and x2, x3, x4 x2 ← x3 ∧ x4 |
| Or Immediate | ori rd, rs1, imm | rd ← rs1 ∨ imm | ori x5, x2, 2 x5 ← x2 ∨ 2 |
| Or | or rd, rs1, rs2 | rd ← rs1 ∨ rs2 | or x2, x3, x4 x2 ← x3 ∨ x4 |
| Shift Left Logical Immediate | slli rd, rs1, imm | rd ← rs1 ≪ imm | slli x2, x3, 2 x2 ← x3 ≪ 2 |
| Shift Right Logical Immed | srli rd, rs1, imm | rd ← rs1 ≫ imm | srli x2, x3, 2 x2 ← x3 ≫ 2 |

| Control instructions | | | |
|---|---|---|---|
| Instruction | Mnemonic | Operation | Example |
| Branch Equal | beq rs1, rs2, label | if rs1 = rs2 then<br>pc ← label | beq x1, x0, loop |
| Branch Less Than | blt rs1, rs2, label | if rs1 < rs2 then<br>pc ← label | blt x1, x0, loop |
| Jump and Link | jal rd, label | rd ← pc + length inst.<br>pc ← label | jal x1, loop |
| Jump and Link Register | jalr rs1 | pc ← rs1 | jalr x1 |

## Exercise

Given a list of common actions in assembly programming. Give for each action (a list of) instruction which would make it work. Describe the operands that are used.

| |
|---|
| Load a specific number into a register → Load 5 into register 5 |
| |
| Move the content of one register to another → Move the content of register 0 to register 6 |
| |
| Load a memory position content into a register → Load the content of memory position 64 into register 4 |
| |
| Store a register content into a specific memory position → Store the content of register 3 in memory position 32 |
| |