# RV32I Base Instruction Set + Zicsr +Zifencei + mret

RISC-V_Cheatsheet_RV32I.xlsx

basic_RV32s document by KHWL(Hyun Woo Kang) | 2025.10.14. | RISC-V Manual version 20240411

## RV32I (42) + Zicsr (6) + Zifencei (1) + mret (1) = 50

## R-Type (11) + I-Type (27) + S-Type (3) + B-Type (6) + U-Type (2) + J-Type (1) = 50

| Instruction Type | Name | Description | Mnemonics | Bit Field | | | | | | | | | | opcode |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 31 | 30 29 28 | 27 26 25 | 24 | 23 22 21 | 20 | 19 18 17 16 15 | 14 13 12 | 11 10 9 8 | 7 | 6 5 4 3 2 1 0 |
| [R-Type] | Instruction | 10 + 1 Instructions | - | funct7 | | | rs2 | | | rs1 | funct3 | rd | | opcode |
| | ADD | Addition | R[rd] = R[rs1] + R[rs2] | 0000000 | | | rs2 | | | rs1 | 000 | rd | | 0110011 |
| | SUB | Substraction | R[rd] = R[rs1] - R[rs2] | 0100000 | | | rs2 | | | rs1 | 000 | rd | | 0110011 |
| | SLL | Shift Left Logical | R[rd] = R[rs1] << R[rs2] | 0000000 | | | rs2 | | | rs1 | 001 | rd | | 0110011 |
| | SLT | Set Less Than | R[rd] = (R[rs1] < R[rs2]) ? 1 : 0 | 0000000 | | | rs2 | | | rs1 | 010 | rd | | 0110011 |
| | SLTU | Set Less Than Unsigned | R[rd] = (R[rs1] < R[rs2]) ? 1 : 0 | 0000000 | | | rs2 | | | rs1 | 011 | rd | | 0110011 |
| | XOR | bitwise XOR | R[rd] = R[rs1] ^ R[rs2] | 0000000 | | | rs2 | | | rs1 | 100 | rd | | 0110011 |
| | SRL | Shift Right Logical | R[rd] = R[rs1] >> R[rs2] | 0000000 | | | rs2 | | | rs1 | 101 | rd | | 0110011 |
| | SRA | Shift Right Arithmetic | R[rd] = R[rs1] >> R[rs2] | 0100000 | | | rs2 | | | rs1 | 101 | rd | | 0110011 |
| | OR | bitwise OR | R[rd] = R[rs1] \| R[rs2] | 0000000 | | | rs2 | | | rs1 | 110 | rd | | 0110011 |
| | AND | bitwise AND | R[rd] = R[rs1] & R[rs2] | 0000000 | | | rs2 | | | rs1 | 111 | rd | | 0110011 |
| [R-Type]<br>SYSTEM | MRET | RETurn from a trap taken into M-mode<br>(RISC-V Previleged Instruction Set) | next PC = CSR[mepc] | 0000000 | | | 00010 | | | 00000 | 000 | 00000 | | 1110011 |
| [I-Type] | Instruction | 14 + 1 + 2 + 3 + 6 + 1 Instructions | - | imm[11:0] | | | | | | rs1 | funct3 | rd | | opcode |
| | ADDI | Addition Immediate value | R[rd] = R[rs1] + imm | imm[11:0] | | | | | | rs1 | 000 | rd | | 0010011 |
| | ANDI | bitwise AND Immediate value | R[rd] = R[rs1] & imm | imm[11:0] | | | | | | rs1 | 111 | rd | | 0010011 |
| | ORI | bitwise OR Immediate value | R[rd] = R[rs1] \| imm | imm[11:0] | | | | | | rs1 | 110 | rd | | 0010011 |
| | XORI | bitwise XOR Immediate value | R[rd] = R[rs1] ^ imm | imm[11:0] | | | | | | rs1 | 100 | rd | | 0010011 |
| | SLLI | Shift Left Logical Immediate value | R[rd] = R[rs1] << imm | 0000000 | | | shamt | | | rs1 | 001 | rd | | 0010011 |
| | SRAI | Shift Right Arithmetic Immediate value | R[rd] = R[rs1] >> imm | 0100000 | | | shamt | | | rs1 | 101 | rd | | 0010011 |
| | SRLI | Shift Right Logical Immediate value | R[rd] = R[rs1] >> imm | 0000000 | | | shamt | | | rs1 | 101 | rd | | 0010011 |
| | SLTI | Set Less Than Immediate value | R[rd] = (R[rs1] < imm) ? 1 : 0 | imm[11:0] | | | | | | rs1 | 010 | rd | | 0010011 |
| | SLTIU | Set Less Than Immediate value Unsigned | R[rd] = (R[rs1] < imm) ? 1 : 0 | imm[11:0] | | | | | | rs1 | 011 | rd | | 0010011 |
| | LB | Load Byte | R[rd] = {24'bM[](15), M[R[rs1] + imm] (7:0)} | imm[11:0] | | | | | | rs1 | 000 | rd | | 0000011 |
| | LBU | Load Byte Unsigned | R[rd] = {24'b0, M[R[rs1] + imm] (7:0)} | imm[11:0] | | | | | | rs1 | 100 | rd | | 0000011 |
| | LH | Load Half-word | R[rd] = {24'bM[](15), M[R[rs1] + imm] (7:0)} | imm[11:0] | | | | | | rs1 | 001 | rd | | 0000011 |
| | LHU | Load Half-word Unsigned | R[rd] = {16'b0, M[R[rs1] + imm] (15:0)} | imm[11:0] | | | | | | rs1 | 101 | rd | | 0000011 |
| | LW | Load Word | R[rd] = {M[R[rs1] + imm] (31:0)} | imm[11:0] | | | | | | rs1 | 010 | rd | | 0000011 |
| | JALR | Jump And Link Register | R[rd] = PC + 4;<br>PC = R[rs1] + imm | imm[11:0] | | | | | | rs1 | 000 | rd | | 1100111 |
| [I-Type]<br>SYSTEM | Instruction | - | - | funct12 | | | | | | rs1 | funct3 | rd | | opcode |
| | ECALL | Environment CALL | CALL service request to the execution Environment | 000000000001 | | | | | | 00000 | 000 | 00000 | | 1110011 |
| | EBREAK | Environment BREAK | return control to a debugging Environment | 000000000000 | | | | | | 00000 | 000 | 00000 | | 1110011 |
| | Instruction | Predecessor, Successor = P, S | Input, Output, Read, Write = I, O, R, W | fm | | PI PO PR | PW | SI SO SR | SW | rs1 | funct3 | rd | | opcode |
| | FENCE | Order device I/O and memory accesses | Read Unprivileged Manual Chapter 2.7 | fm | | predecessor | | successor | | rs1 | 000 | rd | | 0001111 |
| | FENCE.TSO | Pre-encoded FENCE | 2.7 Memory Ordering Instructions | 1000 | | 0011 | | 0011 | | 00000 | 000 | 00000 | | 0001111 |
| | PAUSE | HINT(NOP) | Read Unprivileged Manual Chapter 10. Zihintpause | 0000 | | 0000 | | 0000 | | 00000 | 000 | 00000 | | 0001111 |
| [I-Type]<br>Zicsr | Instruction | - | - | source / destination | | | | | | rs1 | funct3 | rd | | opcode |
| | CSRRW | CSR Read and Write | R[rd] = CSR; CSR = R[rs1] | csr | | | | | | rs1 | 001 | rd | | 1110011 |
| | CSRRS | CSR Read and Set | R[rd] = CSR; CSR = CSR \| R[rs1] | csr | | | | | | rs1 | 010 | rd | | 1110011 |
| | CSRRC | CSR Read and Clear | R[rd] = CSR; CSR = CSR & ~R[rs1] | csr | | | | | | rs1 | 011 | rd | | 1110011 |
| | CSRRWI | CSR Read and Write Immediate value | R[rd] = CSR; CSR = imm | csr | | | | | | uimm[4:0] | 101 | rd | | 1110011 |
| | CSRRSI | CSR Read and Set Immediate value | R[rd] = CSR; CSR = CSR \| imm | csr | | | | | | uimm[4:0] | 110 | rd | | 1110011 |
| | CSRRCI | CSR Read and Clear Immediate value | R[rd] = CSR; CSR = CSR & ~R[rs1] | csr | | | | | | uimm[4:0] | 111 | rd | | 1110011 |
| [I-Type]<br>Zifencei | Instruction | - | - | funct12 | | | | | | rs1 | funct3 | rd | | opcode |
| | FENCE.I | Synchronize instruction and data streams | Read Unprivileged Manual Chapter 6. Zifencei | imm[11:0] | | | | | | rs1 | 001 | rd | | 0001111 |
| [S-Type] | Instruction | 3 Instructions | - | imm[11:5] | | | rs2 | | | rs1 | funct3 | imm[4:0] | | opcode |
| | SB | Store Byte | M[R[rs1] + imm] (7:0) = R[rs2] (7:0) | imm[11:5] | | | rs2 | | | rs1 | funct3 | imm[4:0] | | 0100011 |
| | SH | Store Halfword | M[R[rs1] + imm] (15:0) = R[rs2] (15:0) | imm[11:5] | | | rs2 | | | rs1 | funct3 | imm[4:0] | | 0100011 |
| | SW | Store Word | M[R[rs1] + imm] (31:0) = R[rs2] (31:0) | imm[11:5] | | | rs2 | | | rs1 | funct3 | imm[4:0] | | 0100011 |
| [B-Type] | Instruction | 6 Instructions | - | [12] | imm[10:5] | | rs2 | | | rs1 | funct3 | imm[4:1] | [11] | opcode |
| | BEQ | Branch EQual | if (R[rs1] == R[rs2]) PC = PC + {imm, 1'b0} | [12] | imm[10:5] | | rs2 | | | rs1 | 000 | imm[4:1] | [11] | 1100011 |
| | BNE | Branch Not Equal | if (R[rs1] != R[rs2]) PC = PC + {imm, 1'b0} | [12] | imm[10:5] | | rs2 | | | rs1 | 001 | imm[4:1] | [11] | 1100011 |
| | BLT | Branch Less Than | if (R[rs1] < R[rs2]) PC = PC + {imm, 1'b0} | [12] | imm[10:5] | | rs2 | | | rs1 | 100 | imm[4:1] | [11] | 1100011 |
| | BLTU | Branch Less Than Unsigned | if (R[rs1] < R[rs2]) PC = PC + {imm, 1'b0} | [12] | imm[10:5] | | rs2 | | | rs1 | 110 | imm[4:1] | [11] | 1100011 |
| | BGE | Branch Greater Equal | if (R[rs1] >= R[rs2]) PC = PC + {imm, 1'b0} | [12] | imm[10:5] | | rs2 | | | rs1 | 101 | imm[4:1] | [11] | 1100011 |
| | BGEU | Branch Greater Equal Unsigned | if (R[rs1] >= R[rs2]) PC = PC + {imm, 1'b0} | [12] | imm[10:5] | | rs2 | | | rs1 | 111 | imm[4:1] | [11] | 1100011 |
| [U-Type] | Instruction | 2 Instructions | - | imm[31:12] | | | | | | | | rd | | opcode |
| | LUI | Load Upper Immediate | R[rd] = {imm, 12'b0} | imm[31:12] | | | | | | | | rd | | 0110111 |
| | AUIPC | Add Upper Immediate to current Program Counter value | R[rd] = PC + {imm, 12'b0} | imm[31:12] | | | | | | | | rd | | 0010111 |
| [J-Type] | Instruction | 1 Instruction | - | [20] | imm[10:1] | | | | [11] | imm[19:12] | | rd | | opcode |
| | JAL | Jump And Link | R[rd] = PC + 4;<br>PC = PC + {imm, 1'b0} | [20] | imm[10:1] | | | | [11] | imm[19:12] | | rd | | 1101111 |