

# RISC-V REFERENCE

CS202/CS214

## Core Instruction Formats

|                       |    |    |    |     |    |     |    |        |    |             |   |        |   |        |
|-----------------------|----|----|----|-----|----|-----|----|--------|----|-------------|---|--------|---|--------|
| 31                    | 27 | 26 | 25 | 24  | 20 | 19  | 15 | 14     | 12 | 11          | 7 | 6      | 0 |        |
| funct7                |    |    |    | rs2 |    | rs1 |    | funct3 |    | rd          |   | opcode |   | R-type |
| imm[11:0]             |    |    |    |     |    | rs1 |    | funct3 |    | rd          |   | opcode |   | I-type |
| imm[11:5]             |    |    |    | rs2 |    | rs1 |    | funct3 |    | imm[4:0]    |   | opcode |   | S-type |
| imm[12:10:5]          |    |    |    | rs2 |    | rs1 |    | funct3 |    | imm[4:1:11] |   | opcode |   | B-type |
| imm[31:12]            |    |    |    |     |    |     |    |        |    | rd          |   | opcode |   | U-type |
| imm[20:10:1:11:19:12] |    |    |    |     |    |     |    |        |    | rd          |   | opcode |   | J-type |

## RV32I Base Integer Instructions

| Inst   | Name                    | FMT | Opcode  | funct3 | funct7         | Description (C)                            |
|--------|-------------------------|-----|---------|--------|----------------|--|
| 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 (Arith*)                    |
| 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                       |
| 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[11:5]=0x00 | rd = rs1 « imm[4:0]                        |
| srli   | Shift Right Logical Imm | I   | 0010011 | 0x5    | imm[11:5]=0x00 | rd = rs1 » imm[4:0]                        |
| srai   | Shift Right Arith Imm   | I   | 0010011 | 0x5    | imm[11:5]=0x20 | rd = rs1 » imm[4:0] (Arith*)               |
| 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                       |
| lb     | Load Byte               | I   | 0000011 | 0x0    |                | rd = {24'bM[rs1+imm][7],M[rs1+imm][7:0]}   |
| lh     | Load Half               | I   | 0000011 | 0x1    |                | rd = {16'bM[rs1+imm][15],M[rs1+imm][15:0]} |
| lw     | Load Word               | I   | 0000011 | 0x2    |                | rd = M[rs1+imm][31:0]                      |
| lbu    | Load Byte (U)           | I   | 0000011 | 0x4    |                | rd = {24'b0,M[rs1+imm][7:0]}               |
| lhu    | Load Half (U)           | I   | 0000011 | 0x5    |                | rd = {16'b0,M[rs1+imm][15:0]}              |
| sb     | Store Byte              | S   | 0100011 | 0x0    |                | M[rs1+imm][7:0] = rs2[7:0]                 |
| sh     | Store Half              | S   | 0100011 | 0x1    |                | M[rs1+imm][15:0] = rs2[15:0]               |
| sw     | Store Word              | S   | 0100011 | 0x2    |                | M[rs1+imm][31:0] = rs2[31:0]               |
| beq    | Branch ==               | B   | 1100011 | 0x0    |                | if(rs1 == rs2) PC = PC + {imm,1'b0}        |
| bne    | Branch !=               | B   | 1100011 | 0x1    |                | if(rs1 != rs2) PC = PC + {imm,1'b0}        |
| blt    | Branch <                | B   | 1100011 | 0x4    |                | if(rs1 < rs2) PC = PC + {imm,1'b0}         |
| bge    | Branch ≥                | B   | 1100011 | 0x5    |                | if(rs1 ≥ rs2) PC = PC + {imm,1'b0}         |
| bltu   | Branch < (U)            | B   | 1100011 | 0x6    |                | if(rs1 < rs2) PC = PC + {imm,1'b0}         |
| bgeu   | Branch ≥ (U)            | B   | 1100011 | 0x7    |                | if(rs1 ≥ rs2) PC = PC + {imm,1'b0}         |
| jal    | Jump And Link           | J   | 1101111 |        |                | rd = PC+4; PC = PC + {imm,1'b0}            |
| 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               |

## RV32M Multiply Extension

| Inst  | Name             | Description (C)           |
|-------|------------------|---------------------------|
| mul   | MUL              | $rd = (rs1 * rs2)[31:0]$  |
| mulh  | MUL High         | $rd = (rs1 * rs2)[63:32]$ |
| mulsu | MUL High (S) (U) | $rd = (rs1 * rs2)[63:32]$ |
| mulu  | MUL High (U)     | $rd = (rs1 * rs2)[63:32]$ |
| div   | DIV              | $rd = rs1 / rs2$          |
| divu  | DIV (U)          | $rd = rs1 / rs2$          |
| rem   | Remainder        | $rd = rs1 \% rs2$         |
| remu  | Remainder (U)    | $rd = rs1 \% rs2$         |

## Registers

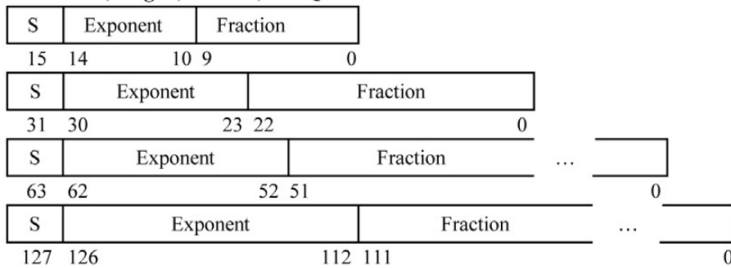
| Register | ABI Name | Description           | Saver  |
|----------|----------|-----------------------|--------|
| x0       | zero     | Zero constant         | —      |
| x1       | ra       | Return address        | Caller |
| x2       | sp       | Stack pointer         | Callee |
| x3       | gp       | Global pointer        | —      |
| x4       | tp       | Thread pointer        | —      |
| x5-x7    | t0-t2    | Temporaries           | Caller |
| x8       | s0 / fp  | Saved / frame pointer | Callee |
| x9       | s1       | Saved register        | Callee |
| x10-x11  | a0-a1    | Fn args/return values | Caller |
| x12-x17  | a2-a7    | Fn args               | Caller |
| x18-x27  | s2-s11   | Saved registers       | Callee |
| x28-x31  | t3-t6    | Temporaries           | Caller |
| f0-7     | ft0-7    | FP temporaries        | Caller |
| f8-9     | fs0-1    | FP saved registers    | Callee |
| f10-11   | fa0-1    | FP args/return values | Caller |
| f12-17   | fa2-7    | FP args               | Caller |
| f18-27   | fs2-11   | FP saved registers    | Callee |
| f28-31   | ft8-11   | FP temporaries        | Caller |

## IEEE 754 FLOATING-POINT STANDARD

$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

where Half-Precision Bias = 15, Single-Precision Bias = 127,  
Double-Precision Bias = 1023, Quad-Precision Bias = 16383

## IEEE Half-, Single-, Double-, and Quad-Precision Formats:



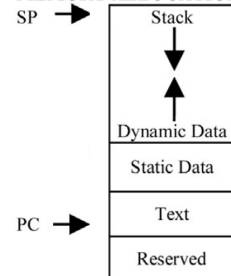
## RV32F / D Floating-Point Extensions

| Inst      | Name                   | Description (C)                           |
|-----------|------------------------|---|
| flw       | Flt Load Word          | $rd = M[rs1 + imm]$                       |
| fsw       | Flt Store Word         | $M[rs1 + imm] = rs2$                      |
| fmadd.s   | Flt Fused Mul-Add      | $rd = rs1 * rs2 + rs3$                    |
| fmsub.s   | Flt Fused Mul-Sub      | $rd = rs1 * rs2 - rs3$                    |
| fnmadd.s  | Flt Neg Fused Mul-Add  | $rd = -rs1 * rs2 + rs3$                   |
| fnmsub.s  | Flt Neg Fused Mul-Sub  | $rd = -rs1 * rs2 - rs3$                   |
| fadd.s    | Flt Add                | $rd = rs1 + rs2$                          |
| fsub.s    | Flt Sub                | $rd = rs1 - rs2$                          |
| fmul.s    | Flt Mul                | $rd = rs1 * rs2$                          |
| fdiv.s    | Flt Div                | $rd = rs1 / rs2$                          |
| fsqrt.s   | Flt Square Root        | $rd = \sqrt{rs1}$                         |
| fsgnj.s   | Flt Sign Injection     | $rd = \text{abs}(rs1) * \text{sgn}(rs2)$  |
| fsgnjn.s  | Flt Sign Neg Injection | $rd = \text{abs}(rs1) * -\text{sgn}(rs2)$ |
| fsgnjx.s  | Flt Sign Xor Injection | $rd = rs1 * \text{sgn}(rs2)$              |
| fmin.s    | Flt Minimum            | $rd = \min(rs1, rs2)$                     |
| fmax.s    | Flt Maximum            | $rd = \max(rs1, rs2)$                     |
| fcvt.s.w  | Flt Conv from Sign Int | $rd = (\text{float}) rs1$                 |
| fcvt.s.wu | Flt Conv from Uns Int  | $rd = (\text{float}) rs1$                 |
| fcvt.w.s  | Flt Convert to Int     | $rd = (\text{int32\_t}) rs1$              |
| fcvt.wu.s | Flt Convert to Int     | $rd = (\text{uint32\_t}) rs1$             |
| fmv.x.w   | Move Float to Int      | $rd = *((\text{int}*) \&rs1)$             |
| fmv.w.x   | Move Int to Float      | $rd = *((\text{float}*) \&rs1)$           |
| feq.s     | Float Equality         | $rd = (rs1 == rs2) ? 1 : 0$               |
| flt.s     | Float Less Than        | $rd = (rs1 < rs2) ? 1 : 0$                |
| fle.s     | Float Less / Equal     | $rd = (rs1 \leq rs2) ? 1 : 0$             |
| fclass.s  | Float Classify         | $rd = 0..9$                               |

## Pseudo Instructions

| Mnemonic       | Name            | Description                                | Uses  |
|----------------|-----------------|--|-------|
| beqz           | Branch = zero   | if( $rs1 == 0$ ) $PC = PC + \{imm, 1b'0\}$ | beq   |
| bnez           | Branch zero     | if( $rs1 != 0$ ) $PC = PC + \{imm, 1b'0\}$ | bne   |
| fabs.s, fabs.d | Absolute Value  | $rd = (rs1 < 0) ? -rs1 : rs1$              | fsgnx |
| fmv.s, fmv.d   | FP Move         | $rd = rs1$                                 | fsgnj |
| fneg.s, fneg.d | FP negate       | $rd = -rs1$                                | fsgnj |
| j              | Jump            | $PC = \{imm, 1b'0\}$                       | jal   |
| jr             | Jump register   | $PC = rs1$                                 | jalr  |
| la             | Load address    | $rd = \text{address}$                      | auipc |
| li             | Load imm        | $rd = imm$                                 | addi  |
| mv             | Move            | $rd = rs1$                                 | addi  |
| neg            | Negate          | $rd = -rs1$                                | sub   |
| nop            | No operation    | zero = zero                                | addi  |
| not            | Not             | $rd = rs1$                                 | xori  |
| ret            | Return          | $PC = ra$                                  | jalr  |
| segz           | Set = zero      | $rd = (rs1 == 0) ? 1 : 0$                  | sltiu |
| snez           | Set $\neq$ zero | $rd = (rs1 != 0) ? 1 : 0$                  | sltiu |

## MEMORY ALLOCATION



## STACK FRAME

