

| | | | | | | start=0 means first instruction in packet, end=1 means last instruction in packet | |
|---------------|---------------------------|---------------------------|---------|------------------|-------------------|---|---------------------------|
| R Type | 7'b opcode | 6'b rd | 6'b rs1 | 6'b rs2 | 5'b predication | 1'b packet start, 1'b packet end | |
| Functionality | | | R | Instruction Type | Description | | Notes |
| Mnemonic | Operation | Category | OpCode | | Assembly | Hardware | Notes |
| add | INT Addition | INT Arithmetic | 000 | 0000 | add rd, rs1, rs2 | $R[rd] = R[rs1] + R[rs2]$ | |
| sub | INT Subtraction | | 000 | 0001 | sub rd, rs1, rs2 | $R[rd] = R[rs1] - R[rs2]$ | |
| mul | INT Multiplication | | 000 | 0010 | mul rd, rs1, rs2 | $R[rd] = R[rs1] \times R[rs2]$ | |
| div | INT Division | | 000 | 0011 | div rd, rs1, rs2 | $R[rd] = R[rs1] / R[rs2]$ | |
| and | AND | Bitwise Logical Operators | 000 | 0100 | and rd, rs1, rs2 | $R[rd] = R[rs1] \text{ AND } R[rs2]$ | |
| or | OR | | 000 | 0101 | or rd, rs1, rs2 | $R[rd] = R[rs1] \text{ OR } R[rs2]$ | |
| xor | XOR | | 000 | 0110 | xor rd, rs1, rs2 | $R[rd] = R[rs1] \text{ XOR } R[rs2]$ | |
| slt | Set if Less Than | Comparison | 000 | 0111 | slt rd, rs1, rs2 | $R[rd] = (R[rs1] < R[rs2]) ? 1 : 0$ | Always assume signed |
| sltu | Set if Less Than Unsigned | | 000 | 1000 | sltu rd, rs1, rs2 | $R[rd] = (\text{unsig}\{R[rs1]\} < \text{unsig}\{R[rs2]\}) ? 1 : 0$ | |
| addf | FP Addition | FP Arithmetic | 000 | 1001 | addf rd, rs1, rs2 | $R[rd] = \text{fp}\{R[rs1]\} + \text{fp}\{R[rs2]\}$ | |
| subf | FP Subtraction | | 000 | 1010 | subf rd, rs1, rs2 | $R[rd] = \text{fp}\{R[rs1]\} - \text{fp}\{R[rs2]\}$ | |
| mulf | FP Multiplication | | 000 | 1011 | mulf rd, rs1, rs2 | $R[rd] = \text{fp}\{R[rs1]\} \times \text{fp}\{R[rs2]\}$ | |
| divf | FP Division | | 000 | 1100 | divf rd, rs1, rs2 | $R[rd] = \text{fp}\{R[rs1]\} / \text{fp}\{R[rs2]\}$ | |
| sll | Shift Left Logical | Bit Shifting | 000 | 1101 | sll rd, rs1, rs2 | $R[rd] = R[rs1] \ll R[rs2]$ | |
| srl | Shift Right Logical | | 000 | 1110 | srl rd, rs1, rs2 | $R[rd] = R[rs1] \gg R[rs2]$ | |
| sra | Shift Right Arithmetic | | 000 | 1111 | sra rd, rs1, rs2 | $R[rd] = R[rs1] \ggg R[rs2]$ | Sign extends R[rs1]'s MSB |
| | | | | | | | |
| I Type | 7'b opcode | 6'b rd | 6'b rs1 | 6'b imm | 5'b predication | 1'b packet start, 1'b packet end | |
| Functionality | | | I | Instruction Type | Description | | Notes |
| Mnemonic | Operation | Category | OpCode | | Assembly | Hardware | Notes |
| lw | Load Word | Memory Read | 001 | 0000 | lw rd, imm(rs1) | $R[rd] = \text{MEM}'32b[rs1 + \text{imm}]$ | |
| lh | Load Half-Word | | 001 | 0001 | lh rd, imm(rs1) | $R[rd] = \text{MEM}'16b[rs1 + \text{imm}]$ | |
| lb | Load Byte | | 001 | 0010 | lb rd, imm(rs1) | $R[rd] = \text{MEM}'8b[rs1 + \text{imm}]$ | |
| jalr | Jump and Link Register | Jump | 001 | 0011 | jalr rd, imm(rs1) | $R[rd] = \text{PC} + 4 ; \text{PC} = R[rs1] + \text{imm}$ | |
| isqrt | Inverse Square Root | Root | 001 | 0100 | isqrt rd, rs1 | $R[rd] = \sqrt{-1}(R[rs1])$ | imm unused |
| sin | Sine | Trig | 001 | 0101 | sin rd, rs1 | $R[rd] = \sin(R[rs1])$ | imm unused |
| cos | Cosine | | 001 | 0110 | cos rd, rs1 | $R[rd] = \cos(R[rs1])$ | imm unused |
| | | | 001 | 0111 | | | |
| | | | 001 | 1000 | | | |
| | | | 001 | 1001 | | | |
| | | | 001 | 1010 | | | |
| | | | 001 | 1011 | | | |
| | | | 001 | 1100 | | | |
| | | | 001 | 1101 | | | |

| | | | | | | | | |
|-------|----------------------------------|-----------------------------------|-----|------|--------------------|--|-----------------------|--|
| | | | 001 | 1110 | | | | |
| | | | 001 | 1111 | | | | |
| addi | Immediate INT Addition | Immediate INT Arithmetic | 010 | 0000 | addi rd, rs1, imm | $R[rd] = R[rs1] + imm$ | | |
| subi | Immediate INT Subtraction | | 010 | 0001 | subi rd, rs1, imm | $R[rd] = R[rs1] - imm$ | | |
| itof | Integer to Float | Type Conversion | 010 | 0010 | itof rd, rs1, imm | $R[rd] = fl \text{ ' } R[rs1]$ | imm unused | |
| ftoi | Float to Integer | | 010 | 0011 | ftoi rd, rs1, imm | $R[rd] = int \text{ ' } R[rs1]$ | imm unused | |
| | | Immediate Logical Operators | 010 | 0100 | | | | |
| ori | Immediate OR | | 010 | 0101 | ori rd, rs1, imm | $R[rd] = R[rs1] OR imm$ | | |
| | | | 010 | 0110 | | | | |
| slti | Immediate Set Less Than | Immediate Comparison | 010 | 0111 | slti rd, rs1, imm | $R[rd] = (R[rs1] < imm) ? 1 : 0$ | | |
| sltiu | Immediate Set Less Than Unsigned | | 010 | 1000 | sltiu rd, rs1, imm | $R[rd] = (unsig\{R[rs1]\} < unsig\{imm\}) ? 1 : 0$ | | |
| | | | 010 | 1001 | | | | |
| | | | 010 | 1010 | | | | |
| | | | 010 | 1011 | | | | |
| | | | 010 | 1100 | | | | |
| | | | 010 | 1101 | | | | |
| srli | Immediate Shift Right Logical | Immediate Shifting | 010 | 1110 | srli rd, rs1, imm | $R[rd] = R[rs1] >> R[rs2]$ | | |
| srai | Immediate Shift Right Arithmetic | | 010 | 1111 | srai rd, rs1, imm | $R[rd] = R[rs1] >>> imm$ | sign extend rs1's msb | |

| S Type | 7'b opcode | 6'b imm | 6'b rs1 | 6'b rs2 | 5'b predication | 1'b packet start, 1'b packet end | | |
|---------------|-----------------|--------------|---------|------------------|------------------|--|-------|--|
| Functionality | | | S | Instruction Type | Description | | Notes | |
| Mnemonic | Operation | Category | OpCode | | Assembly | Hardware | Notes | |
| sw | Store Word | Memory Write | 011 | 0000 | sw rs2, imm(rs1) | $MEM[R[rs1] + imm] = R \text{ ' } 32b [rs2]$ | | |
| sh | Store Half-Word | | 011 | 0001 | sh rs2, imm(rs1) | $MEM[R[rs1] + imm] = R \text{ ' } 16b [rs2]$ | | |
| sb | Store Byte | | 011 | 0010 | sb rs2, imm(rs1) | $MEM[R[rs1] + imm] = R \text{ ' } 8b [rs2]$ | | |
| | | | 011 | 0011 | | | | |
| | | | 011 | 0100 | | | | |
| | | | 011 | 0101 | | | | |
| | | | 011 | 0110 | | | | |
| | | | 011 | 0111 | | | | |
| | | | 011 | 1000 | | | | |
| | | | 011 | 1001 | | | | |
| | | | 011 | 1010 | | | | |
| | | | 011 | 1011 | | | | |
| | | | 011 | 1100 | | | | |
| | | | 011 | 1101 | | | | |
| | | | 011 | 1110 | | | | |
| | | | 011 | 1111 | | | | |

| B Type | 7'b opcode | 6'b Pred Dest | 6'b rs1 | 6'b rs2 | 5'b predication | 1'b packet start, 1'b packet end | | |
|---------------|--|---------------|---------|------------------|--------------------|--|-------|--|
| Functionality | | | B | Instruction Type | Description | | Notes | |
| Mnemonic | Operation | Category | OpCode | | Assembly | Hardware | Notes | |
| beq | Branch if Equal | Memory Write | 100 | 0000 | beq rs1, rs2, imm | if (R[rs1] == R[rs2]), R[PC] = R[PC] + imm | | |
| bne | Branch if Not Equal | | 100 | 0001 | bne rs1, rs2, imm | if (R[rs1] != R[rs2]), R[PC] = R[PC] + imm | | |
| bge | Branch if Greater or Equal Than | | 100 | 0010 | bge rs1, rs2, imm | if (R[rs1] >= R[rs2]), R[PC] = R[PC] + imm | | |
| bgeu | Branch if Greater or Equal Than Unsigned | | 100 | 0011 | bgeu rs1, rs2, imm | if (unsig{R[rs1]} >= unsig{R[rs2]}), R[PC] = R[PC] + imm | | |
| blt | Branch if Less Than | | 100 | 0100 | blt rs1, rs2, imm | if (R[rs1] < R[rs2]), R[PC] = R[PC] + imm | | |
| bltu | Branch if Less Than Unsigned | | 100 | 0101 | bltu rs1, rs2, imm | if (unsig{R[rs1]} < unsig{R[rs2]}), R[PC] = R[PC] + imm | | |
| | | | 100 | 0110 | | | | |
| | | | 100 | 0111 | | | | |
| | | | 100 | 1000 | | | | |
| | | | 100 | 1001 | | | | |
| | | | 100 | 1010 | | | | |
| | | | 100 | 1011 | | | | |
| | | | 100 | 1100 | | | | |
| | | | 100 | 1101 | | | | |
| | | | 100 | 1110 | | | | |
| | | | 100 | 1111 | | | | |

| U Type | 7'b opcode | 6'b Pred Dest | 12'b imm | | 5'b predication | 1'b packet start, 1'b packet end | | |
|---------------|---------------------------|---------------------|----------|------------------|-----------------|--|-----------------------|--|
| Functionality | | | U | Instruction Type | Description | | Notes | |
| Mnemonic | Operation | Category | OpCode | | Assembly | Hardware | Notes | |
| auipc | Add Upper Immediate to PC | Build PC | 101 | 0000 | auipc rd, imm | R[rd] = R[PC] + (imm << 12) | | |
| lli | Load Lower Immediate | Building Immediates | 101 | 0001 | lli rd, imm | R[rd] = {R[rd] [31:12], imm[11:0]} | Directly Assign Value | |
| lmi | Load Middle Immediate | | 101 | 0010 | lmi rd, imm | R[rd] = {R[rd] [31:24], imm[11:0], R[rd] [11:0]} | Directly Assign Value | |
| | | | 101 | 0011 | | | | |
| lui | Load Upper Immediate | | 101 | 0100 | lui rd, imm | R[rd] = {imm[7:0], R[rd] [23:0]} | Directly Assign Value | |
| | | | 101 | 0101 | | | | |
| | | | 101 | 0110 | | | | |
| | | | 101 | 0111 | | | | |

| C Type | 7'b opcode | 6'b rd | 10'b csr1 | 2'b X | 5'b predication | 1'b packet start, 1'b packet end | | |
|---------------|-------------------------------|-------------------------|-----------|------------------|-----------------|----------------------------------|-----------------------------|--|
| Functionality | | | C | Instruction Type | Description | | Notes | |
| Mnemonic | Operation | Category | OpCode | | Assembly | Hardware | Notes | |
| csrr | Control Status Register Read | Control Status Register | 101 | 1000 | csrr rd, csr1 | R[rd] = CSR[csr1] | csr1 = {rs2[3:0], rs1[5:0]} | |
| csrw | Control Status Register Write | | 101 | 1001 | csrw rd, csr1 | CSR[csr1] = R[rd] | csr1 = {rs2[3:0], rd[5:0]} | |

| | | | | | | | | |
|-----------------|--------------------|-----------------|---------------|------------------|-----------------|--|--------------|--|
| | | | 101 | 1010 | | | | |
| | | | 101 | 1011 | | | | |
| | | | 101 | 1100 | | | | |
| | | | 101 | 1101 | | | | |
| | | | 101 | 1110 | | | | |
| | | | 101 | 1111 | | | | |
| | | | | | | | | |
| J Type | 7'b opcode | 6'b Pred Dest | 12'b imm | | 5'b predication | 1'b packet start, 1'b packet end | | |
| Functionality | | | J | Instruction Type | Description | | Notes | |
| <u>Mnemonic</u> | <u>Operation</u> | <u>Category</u> | <u>OpCode</u> | | <u>Assembly</u> | <u>Hardware</u> | <u>Notes</u> | |
| jal | Jump And Link | Jump | 110 | 0000 | jal rd, imm | $R[rd] = R[PC] + 4$; $R[PC] = R[PC] + imm$ | | |
| | | | 110 | 0001 | | | | |
| | | | 110 | 0010 | | | | |
| | | | 110 | 0011 | | | | |
| | | | 110 | 0100 | | | | |
| | | | 110 | 0101 | | | | |
| | | | 110 | 0110 | | | | |
| | | | 110 | 0111 | | | | |
| | | | | | | | | |
| P Type | 7'b opcode | 6'b X | 6'b rs1 | 6'b rs2 | 5'b predication | 1'b packet start, 1'b packet end | | |
| Functionality | | | P | Instruction Type | Description | | Notes | |
| <u>Mnemonic</u> | <u>Operation</u> | <u>Category</u> | <u>OpCode</u> | | <u>Assembly</u> | <u>Hardware</u> | <u>Notes</u> | |
| jpnz | Jump Pred Not Zero | Jump Pred | 110 | 1000 | jpnz rs1, rs2 | if($PR[rs1] == 0$), $R[PC] = R[rs2]$; else, $R[PC] = R[PC] + 4$ | | |
| | | | 110 | 1001 | | | | |
| | | | 110 | 1010 | | | | |
| | | | 110 | 1011 | | | | |
| | | | 110 | 1100 | | | | |
| | | | 110 | 1101 | | | | |
| | | | 110 | 1110 | | | | |
| | | | 110 | 1111 | | | | |