32-BIT INSTRUCTION CODE IN THE INSTRUCTION TYPE FORMAT

li:

Format

auipc rd,imm

Description

Build pc-relative addresses and uses the U-type format. AUIPC forms a 32-bit offset from the 20-bit U-immediate, filling in the lowest 12 bits with zeros, adds this offset to the pc, then places the result in register rd.

Implementation

x[rd] = pc + sext(immediate[31:12] << 12)

Addi :

Format

addi rd,rs1,imm

Description

Adds the sign-extended 12-bit immediate to register rs1. Arithmetic overflow is ignored and the result is simply the low XLEN bits of the result. ADDI rd, rs1, 0 is used to implement the MV rd, rs1 assembler pseudo-instruction.

Implementation

x[rd] = x[rs1] + sext(immediate)

slti:

Format

slti rd,rs1,imm

Description

Place the value 1 in register rd if register rs1 is less than the signextended immediate when both are treated as signed numbers, else 0 is written to rd.

Implementation

x[rd] = x[rs1] <s sext(immediate)

xori:

Format

ori rd,rs1,imm

Description

Performs bitwise OR on register rs1 and the sign-extended 12-bit immediate and place the result in rd

Implementation

x[rd] = x[rs1] | sext(immediate)

andi:

Format

andi rd,rs1,imm

Description

Performs bitwise AND on register rs1 and the sign-extended 12-bit immediate and place the result in rd

Implementation

x[rd] = x[rs1] & sext(immediate)

srli:

Format

srli rd,rs1,shamt

Description

Performs logical right shift on the value in register rs1 by the shift amount held in the lower 5 bits of the immediate

In RV64, bit-25 is used to shamt[5].

Implementation

x[rd] = x[rs1] >>u shamt

sub:

Format

sub rd,rs1,rs2

Description

Subs the register rs2 from rs1 and stores the result in rd.

Arithmetic overflow is ignored and the result is simply the low XLEN bits of the result.

Implementation

x[rd] = x[rs1] - x[rs2]

sll:

Format

sll rd,rs1,rs2

Description

Performs logical left shift on the value in register rs1 by the shift amount held in the lower 5 bits of register rs2.

Implementation

x[rd] = x[rs1] << x[rs2]

jal:

Format

jal rd,offset

Description

Jump to address and place return address in rd.

Implementation

x[rd] = pc+4; pc += sext(offset)

bne:

Format

bne rs1,rs2,offset

Description

Take the branch if registers rs1 and rs2 are not equal.

Implementation

if (x[rs1] != x[rs2]) pc += sext(offset)

bge:

Format

bge rs1,rs2,offset

Description

Take the branch if registers rs1 is greater than or equal to rs2, using signed comparison.

Implementation

if (x[rs1] >=s x[rs2]) pc += sext(offset)

bltu:

Format

bltu rs1,rs2,offset

Description

Take the branch if registers rs1 is less than rs2, using unsigned comparison.

Implementation

if (x[rs1] <u x[rs2]) pc += sext(offset)

bgeu:

Format

bgeu rs1,rs2,offset

Description

Take the branch if registers rs1 is greater than or equal to rs2, using unsigned comparison.

Implementation

if (x[rs1] >=u x[rs2]) pc += sext(offset)

PreviousNext