VSDSQUADRON MINI INTERNSHIP JUNE 2024

By

VLSI SYSTEM DESIGN

Documented By

VIJAY N

R.M.K. ENGINEERING COLLEGE

ELECTRONICS AND COMMUNICATION ENGINEERING

1.RISC-V

- RISC-V (pronounced "risk-five") is a new instruction-set architecture (ISA) that was originally designed to support computer architecture research and education.
- RISC-V is an open-standard instruction set architecture (ISA) that is free to use for anyone.
- RISC-V is designed to be simple and modular, allowing for custom extensions and optimizations.
- Allows for easy implementation of custom extensions tailored to specific applications.
- Follows the principles of Reduced Instruction Set Computing (RISC), which simplifies the hardware and can improve performance.

2. RV32I Base Integer Instruction Set

The **RV32I Base Integer Instruction Set** is a fundamental subset of the RISC-V instruction set architecture (ISA) designed for 32-bit integer operations. It provides the essential instructions needed to perform basic computing tasks. Here's a detailed breakdown:

Overview of RV32I:

- **32-bit Architecture**: The "32" in RV32I refers to the 32-bit width of the instruction set, indicating that instructions and data are processed as 32-bit entities.
- **Base Integer Instructions**: The "I" stands for "Integer," meaning this set covers the basic operations for integer arithmetic, logic, and data manipulation.

Basic Operations:

- **Arithmetic**: Add, subtract, multiply, divide.
- **Logical**: AND, OR, XOR, shift operations.
- **Data Movement**: Load from memory, store to memory.
- **Control Flow**: Conditional branches, jumps, subroutine calls.

Instruction Formats:

- **R-Type**: For register-to-register operations (e.g., add, sub).
- **I-Type**: For immediate value operations and loads (e.g., addi, lw).
- **S-Type**: For store operations (e.g., sw).
- **B-Type**: For branch operations (e.g., beq).
- **U-Type**: For upper immediate operations (e.g., lui).
- **J-Type**: For jump operations (e.g., jal).

Instruction Set Overview:

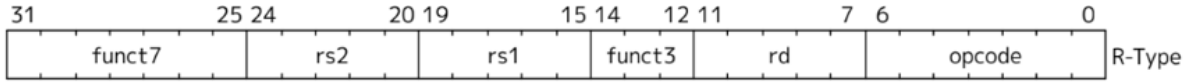
Instruction Type	Description	Examples
Arithmetic	Basic math operations	`ADD`, `SUB`, `MUL`, `DIV`
Logical	Bitwise operations	`AND`, `OR`, `XOR`, `SLL`
Memory	Data transfer between memory and registers	`LW`, `SW`
Branch	Conditional execution based on comparisons	`BEQ`, `BNE`
Jump	Unconditional changes in the execution flow	`JAL`, `JALR`
Immediate	Operations involving constants	`ADDI`, `ORI`

3.BASE INSTRUCTION FORMATS AND TYPES

- In RISC-V, Generally in base RV32I ISA, there are four core instruction formats (R/I/S/U)
- There are a further two variants of the instruction formats (B/J) based on the handling of immediates, these base instruction formats that define how instructions are encoded within a 32-bit word
- Finally, it can be classified as,
    - **R-type**: Register-Register operations
    - **I-type**: Immediate operations
    - **S-type**: Store operations
    - **B-type**: Branch operations
    - **U-type**: Upper immediate operations
    - **J-type**: Jump operations

R-TYPE: REGISTER-REGISTER OPERATIONS

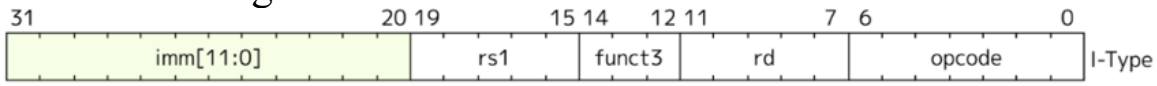
- **Purpose:** Performs operations using an immediate (constant) value and a register.



- **Fields:**
  - **opcode:** Specifies the type of instruction.
  - **rd:** Destination register.
  - **funct3:** Further specifies the operation.
  - **rs1:** Source register.
  - **imm[11:0]:** 12-bit immediate value.

I-TYPE: IMMEDIATE OPERATIONS

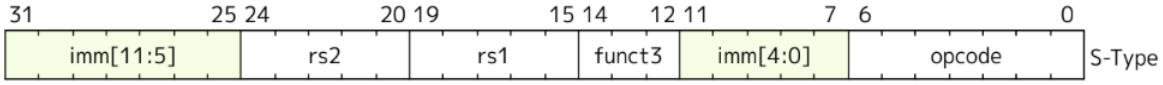
- **Purpose:** Performs operations using an immediate (constant) value and a register.



- **Fields:**
  - **opcode:** Specifies the type of instruction.
  - **rd:** Destination register.
  - **funct3:** Further specifies the operation.
  - **rs1:** Source register.
  - **imm[11:0]:** 12-bit immediate value.

S-TYPE: STORE OPERATIONS

- **Purpose:** Stores data from a register to memory.

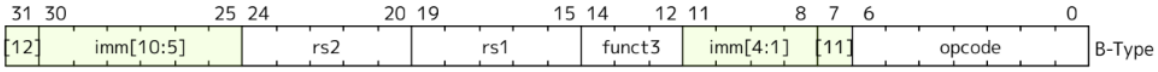


- **Fields:**
  - **opcode:** Specifies the type of instruction.
  - **imm[11:5]:** Higher bits of the 12-bit immediate value.
  - **funct3:** Further specifies the operation.
  - **rs1:** Base address register.

- **rs2:** Source register.
- **imm[4:0]:** Lower bits of the 12-bit immediate value.

B-TYPE: BRANCH OPERATIONS

➢ **Purpose:** Performs conditional branches based on the comparison of two registers.

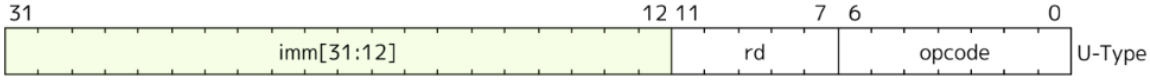


➢ **Fields:**

- **opcode:** Specifies the type of instruction.
- **imm[12]:** Sign bit of the immediate value.
- **imm[10:5]:** Higher bits of the immediate value.
- **funct3:** Further specifies the operation.
- **rs1:** First source register.
- **rs2:** Second source register.
- **imm[4:1]:** Lower bits of the immediate value.
- **imm[11]:** Middle bit of the immediate value.

U-TYPE: UPPER IMMEDIATE OPERATIONS

➢ **Purpose:** Loads a 20-bit immediate value into the upper 20 bits of a register.

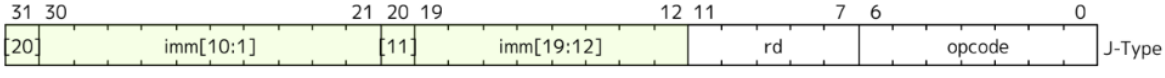


➢ **Fields:**

- **opcode:** Specifies the type of instruction.
- **rd:** Destination register.
- **imm[31:12]:** 20-bit immediate value.

J-TYPE: JUMP OPERATIONS

➢ **Purpose:** Performs jumps to a target address specified by an immediate value



- **Fields:**
    - **opcode:** Specifies the type of instruction.
    - **rd:** Destination register.
    - **imm[20]:** Sign bit of the immediate value.
    - **imm[10:1]:** Lower bits of the immediate value.
    - **imm[11]:** Middle bit of the immediate value.
    - **imm[19:12]:** Higher bits of the immediate value.