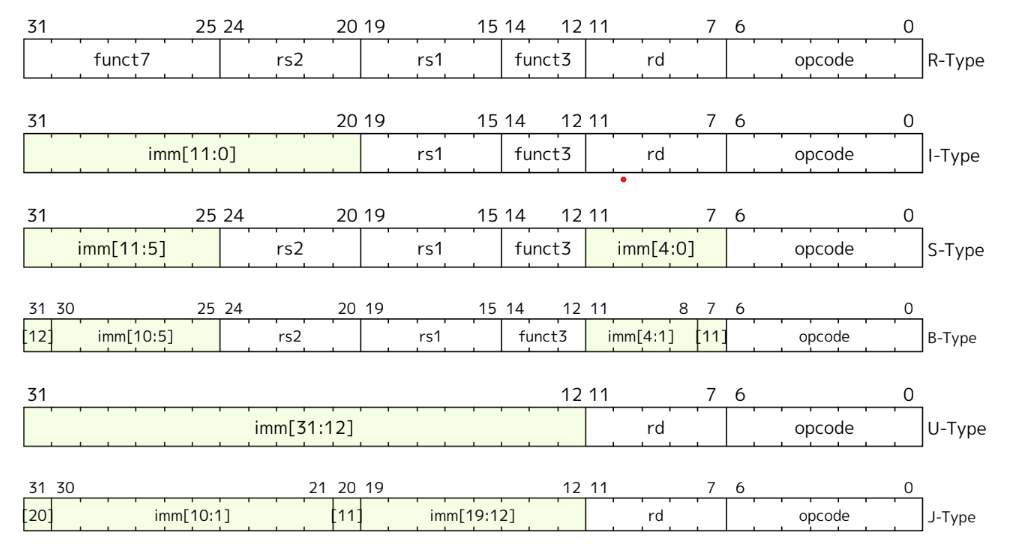
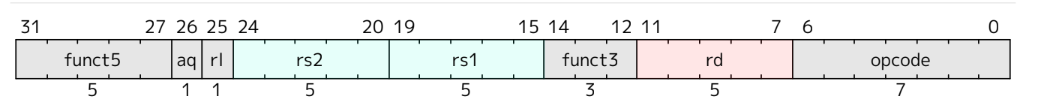
**RISC-V Instruction Types:**

****

From the RISC-V software documentation, the main instruction types are:

**R-type**: Register-to-register operations (e.g., add, sub, and, or, xor)



**1. opcode (7 bits)**

* **Purpose:** Identifies the type of instruction (e.g., add, sub, load, store, branch, jump).
* **Significance:** This is the most crucial field. It determines the operation that the processor needs to perform.

**2. rd (5 bits)**

* **Purpose:** Specifies the destination register for the result of the operation.
* **Significance:** This field indicates where the result of the instruction should be stored.

**3. funct3 (3 bits)**

* **Purpose:** Further specifies the operation within a particular opcode group.
* **Significance:** Some opcodes can represent multiple operations. This field helps differentiate between them. For example, opcodes 000 and 010 can both represent arithmetic operations, but funct3 distinguishes between add (000) and sub (010).

**4. rs1 (5 bits)**

* **Purpose:** Specifies the first source register for the operation.
* **Significance:** This field indicates the first operand for the operation.

**5. rs2 (5 bits)**

* **Purpose:** Specifies the second source register for the operation.
* **Significance:** This field indicates the second operand for the operation. This field is not present in all instruction types (e.g., I-type).

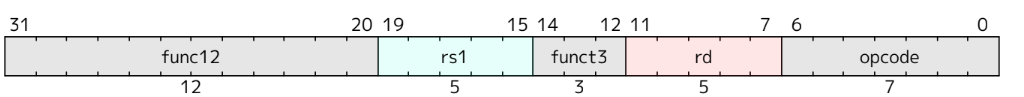
**6. funct5 (5 bits)**

* **Purpose:** Provides additional information for certain operations, especially R-type instructions.
* **Significance:** This field can further distinguish between different operations within the same opcode and funct3 combination.

**7. aq/rl (2 bits)**

* **Purpose:** Used for memory ordering and relaxation.
* **Significance:** These bits are used for advanced features like memory ordering and relaxation.

**I-type**: Immediate operations (e.g., addi, slli, slti) and load instructions (e.g., lb, lh, lw)



**1. opcode (7 bits)**

* **Purpose:** Identifies the type of instruction (e.g., add, sub, load, store, branch, jump).
* **Significance:** This is the most crucial field. It determines the operation that the processor needs to perform.

**2. rd (5 bits)**

* **Purpose:** Specifies the destination register for the result of the operation.
* **Significance:** This field indicates where the result of the instruction should be stored.

**3. funct3 (3 bits)**

* **Purpose:** Further specifies the operation within a particular opcode group.
* **Significance:** Some opcodes can represent multiple operations. This field helps differentiate between them. For example, opcodes 000 and 010 can both represent arithmetic operations, but funct3 distinguishes between add (000) and sub (010).

**4. rs1 (5 bits)**

* **Purpose:** Specifies the first source register for the operation.
* **Significance:** This field indicates the first operand for the operation.

**5. rs2 (5 bits)**

* **Purpose:** Specifies the second source register for the operation.
* **Significance:** This field indicates the second operand for the operation. This field is not present in all instruction types (e.g., I-type).

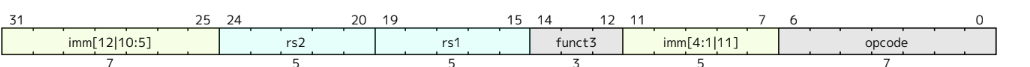
**6. funct5 (5 bits)**

* **Purpose:** Provides additional information for certain operations, especially R-type instructions.
* **Significance:** This field can further distinguish between different operations within the same opcode and funct3 combination.

**7. funct12 (12 bits)**

* **Purpose:** Provides additional information for certain operations, especially for some special instructions.
* **Significance:** This field is used for specialized instructions or extensions to the RISC-V instruction set.

**B-type:** Branch instructions (e.g., beq, bne, blt, bge)



**1. opcode (7 bits)**

* **Purpose:** Identifies the type of instruction (e.g., add, sub, load, store, branch, jump).
* **Significance:** This is the most crucial field. It determines the operation that the processor needs to perform.

**2. imm[4:1/11] (5 bits)**

* **Purpose:** Represents part of the immediate value used in the instruction.
* **Significance:** This field, along with other parts of the immediate value, is used for operations like memory addressing, branching, or other immediate-related calculations.

**3. rs1 (5 bits)**

* **Purpose:** Specifies the first source register for the operation.
* **Significance:** This field indicates the first operand for the operation.

**4. funct3 (3 bits)**

* **Purpose:** Further specifies the operation within a particular opcode group.
* **Significance:** Some opcodes can represent multiple operations. This field helps differentiate between them. For example, opcodes 000 and 010 can both represent arithmetic operations, but funct3 distinguishes between add (000) and sub (010).

**5. rs2 (5 bits)**

* **Purpose:** Specifies the second source register for the operation.
* **Significance:** This field indicates the second operand for the operation. This field is not present in all instruction types (e.g., I-type).

**6. imm[12/10:5] (7 bits)**

* **Purpose:** Represents part of the immediate value used in the instruction.
* **Significance:** This field, along with other parts of the immediate value, is used for operations like memory addressing, branching, or other immediate-related calculations.

**U-type**: Instructions with a 20-bit immediate (e.g., auipc, lui)

**J-type**: Unconditional jumps (e.g., jal, jalr)

**S-type**: Store instructions (e.g., sb, sh, sw)

**Unique RISC-V Instructions Identified :**

Based on the analysis of the assembly code, I identified the following unique RISC-V instructions:

1. addi
2. sd
3. li
4. jal
5. ld
6. ret
7. mv
8. beqz
9. jalr
10. jr
11. bltz
12. neg
13. sll
14. and
15. sub
16. lui
17. auipc
18. lut

**32-bit Instruction Codes (Example – addi: 1111 1111 0000 0001 0000 0001 0001 0011)**

**R-type Instruction Format:**

31 25 24 20 19 15 14 12 11 7 6 5 0

+-------+-------+-------+-------+-------+-------+

| func7 | rs2 | rs1 | funct3 | rd | opcode |

+-------+-------+-------+-------+-------+-------+

**Decoding the Given Sequence:**

* **opcode:** 0000000 (7 bits) - This is the opcode for R-type instructions.
* **rd:** 00011 (5 bits) - This corresponds to register x3.
* **rs1:** 00001 (5 bits) - This corresponds to register x1.
* **rs2:** 00010 (5 bits) - This corresponds to register x2.
* **funct3:** 001 (3 bits)
* **func7:** 1111111 (7 bits)