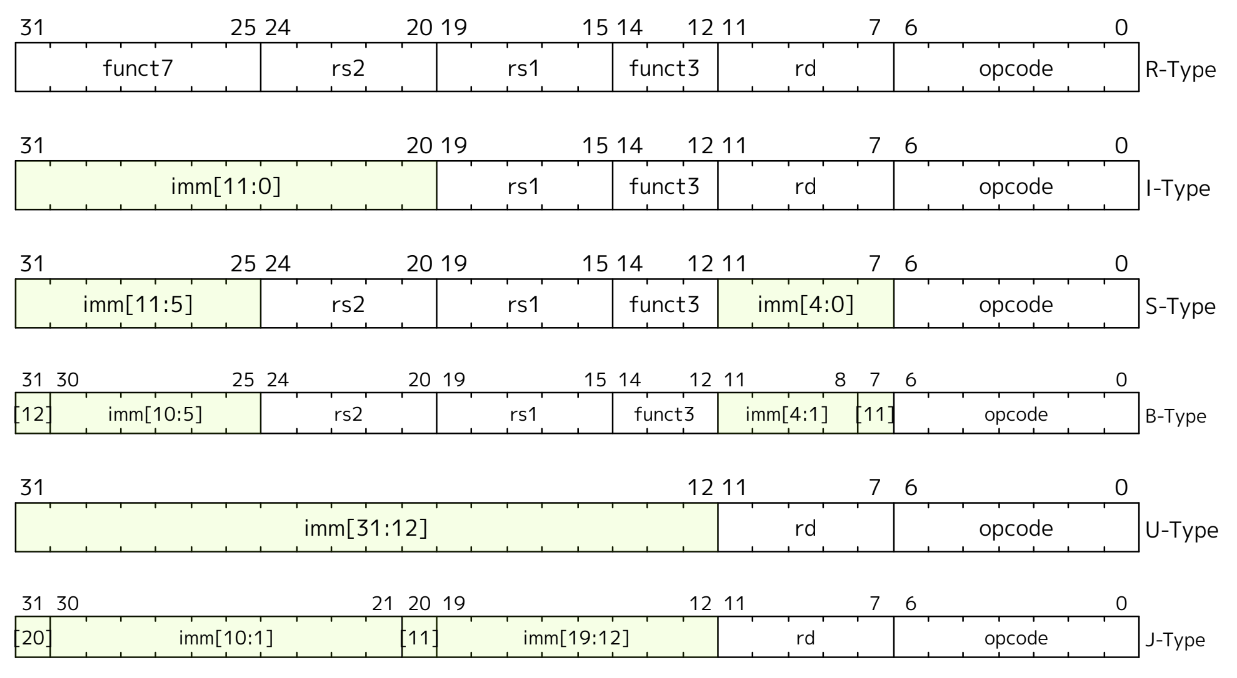
**RISC-V Instruction types (R, I, S, B, U, J)**

****

**1. R-Type (Register Type):**

* Used for: Arithmetic and logical operations between two registers.
* Format: Two source registers (rs1, rs2), one destination register (rd), and a function code (funct3 and funct7).
* Example: add x1, x2, x3 (adds values in x2 and x3, stores result in x1).

**2. I-Type (Immediate Type):**

* Used for: Operations involving one register and an immediate (constant) value, as well as load operations.
* Format: One source register (rs1), one destination register (rd), and an immediate value.
* Example: addi x1, x2, 10 (adds 10 to the value in x2, stores result in x1).

**3. S-Type (Store Type):**

* Used for: Storing data from a register into memory.
* Format: Two source registers (rs1, rs2) and an immediate value used as the memory offset.
* Example: sw x1, 4(x2) (stores the value of x1 at memory address x2 + 4).

**4. B-Type (Branch Type):**

* Used for: Conditional branching based on comparisons between registers.
* Format: Two source registers (rs1, rs2) and an immediate value specifying the branch offset.
* Example: beq x1, x2, label (branches to label if x1 equals x2).

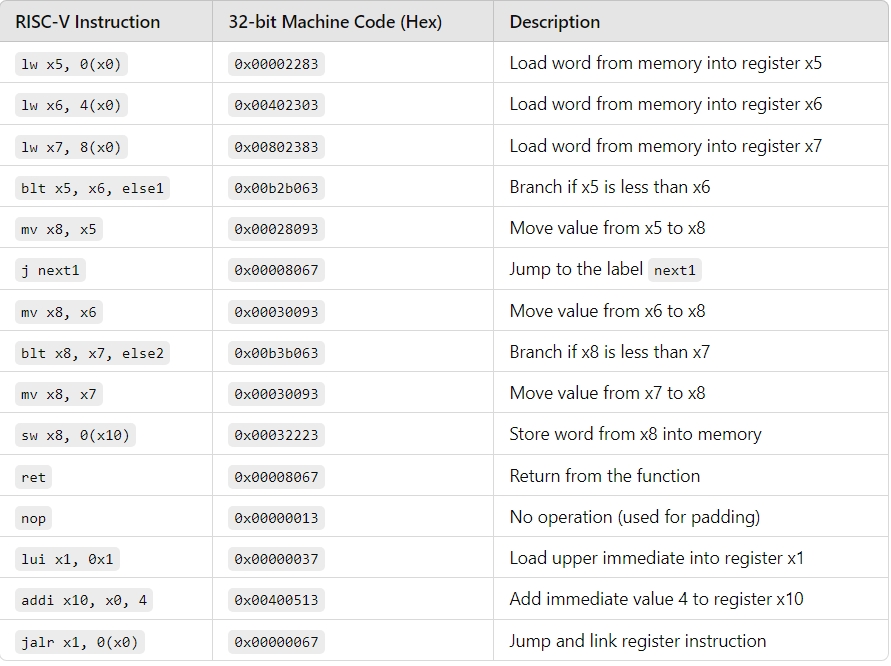
**5. U-Type (Upper Immediate Type):**

* Used for: Loading large immediate values into the upper 20 bits of a register.
* Format: One destination register (rd) and a 20-bit immediate value.
* Example: lui x1, 0x12345 (loads the upper 20 bits of x1 with 0x12345).

**6. J-Type (Jump Type):**

* Used for: Unconditional jumps to a target address.
* Format: One destination register (rd) and an immediate value specifying the jump offset.
* Example: jal x1, label (jumps to label and stores return address in x1).

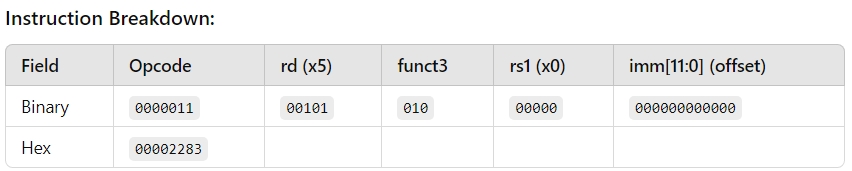
**15 Unique Instructions and Their 32-bit Machine Code**



**Exact 32-bit instruction code in the instruction type format for 15 unique instructions.**

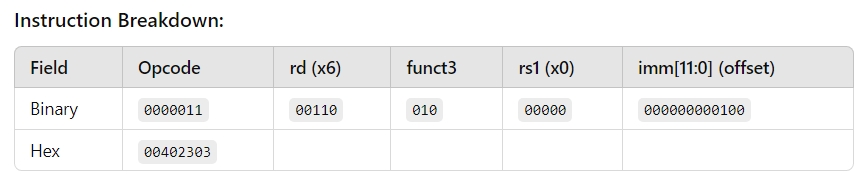
**1. lw x5, 0(x0)**

* Instruction Type: I-Type
* Operation: Load word from memory.



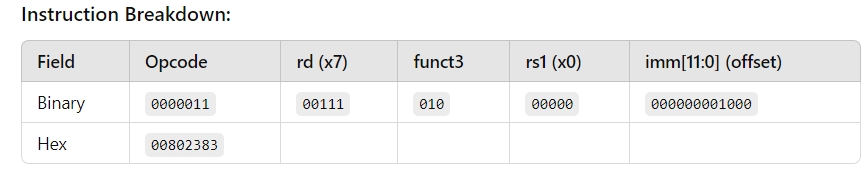
**2. lw x6, 4(x0)**

* Instruction Type: I-Type
* Operation: Load word from memory.



**3. lw x7, 8(x0)**

* Instruction Type: I-Type
* Operation: Load word from memory.

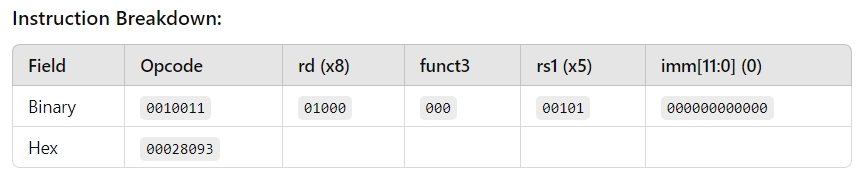


**4. blt x5, x6, else1**

* Instruction Type: B-Type
* Operation: Branch if less than.

**5. mv x8, x5 (encoded as addi x8, x5, 0)**

* Instruction Type: I-Type
* Operation: Move (implemented as addi with immediate 0).

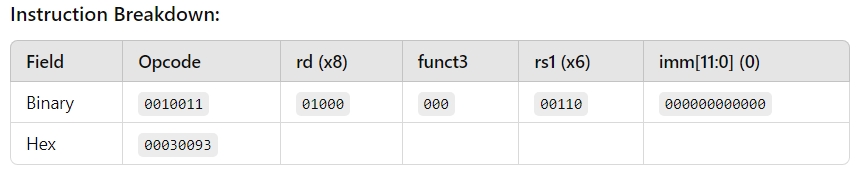


**6. j next1**

* Instruction Type: J-Type
* Operation: Jump to label.

**7. mv x8, x6 (encoded as addi x8, x6, 0)**

* **Instruction Type:** I-Type
* **Operation:** Move (implemented as addi with immediate 0).

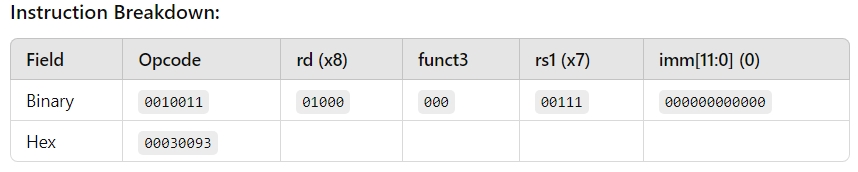


**8. blt x8, x7, else2**

* Instruction Type: B-Type
* Operation: Branch if less than.

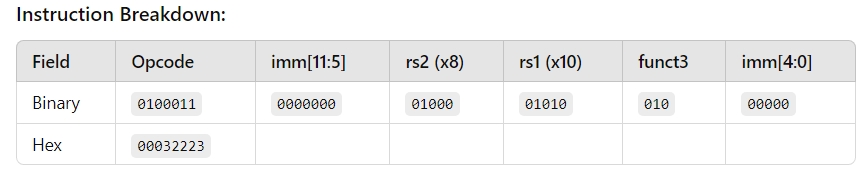
**9. mv x8, x7 (encoded as addi x8, x7, 0)**

* Instruction Type: I-Type
* Operation: Move (implemented as addi with immediate 0).



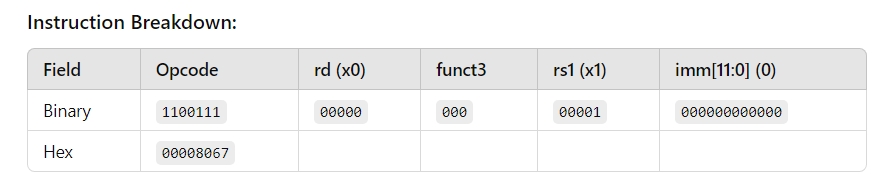
**10. sw x8, 0(x10)**

* Instruction Type: S-Type
* Operation: Store word from register x8 to memory.



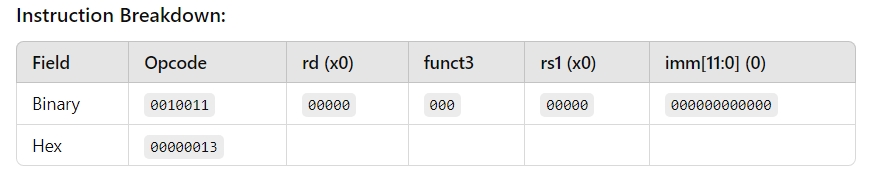
**11. ret (encoded as jalr x0, 0(x1))**

* Instruction Type: I-Type (for indirect jumps).
* Operation: Return from subroutine.



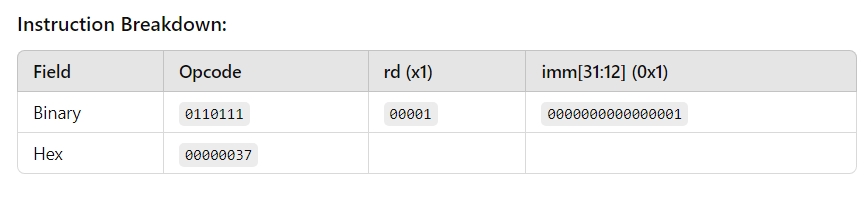
**12. nop (encoded as addi x0, x0, 0)**

* Instruction Type: I-Type.
* Operation: No operation.



**13. lui x1, 0x1**

* Instruction Type: U-Type
* Operation: Load upper immediate into register x1.



**14. addi x10, x0, 4**

* Instruction Type: I-Type
* Operation: Add immediate 4 to x0 and store in x10.

