Chapter 1:

1. What is response time and throughput?

Response Time: It is the total time taken to complete a single task, including execution and waiting time. It measures system performance from the user's perspective.

Throughput: It is the number of tasks a system completes in a given period. It measures system productivity, with higher throughput indicating better performance.

---

2. What is ISA? What is its impact?

ISA (Instruction Set Architecture): ISA is the interface between hardware and software, defining the instructions a processor can execute. It includes instruction formats, data types, addressing modes, and more.

Impact: A well-designed ISA improves performance, simplifies compiler design, and ensures compatibility across different implementations of the same architecture.

---

3. What does a multiprocessor do?

A multiprocessor system has multiple CPUs working together to execute tasks simultaneously, improving performance, reliability, and scalability. It is commonly used in servers and high-performance computing to handle parallel workloads efficiently.

Chapter 2:

1. R-Type (Register-Register)

opcode: Specifies the type of instruction (e.g., arithmetic, logical).

rd: Specifies the destination register where the result will be stored.

funct3 and funct7: Encodes specific operation details (e.g., addition, subtraction) within the instruction type.

rs1 and rs2: Specify the source registers containing operands for the operation.

---

2. I-Type (Immediate)

opcode: Identifies the instruction category, such as load or arithmetic with immediate.

rd: Indicates the destination register to store the result.

funct3: Encodes the operation subtype (e.g., load type, immediate operation type).

rs1: Specifies the source register that provides one operand.

immediate: Provides a constant value as the second operand or offset for memory access.

---

3. S-Type (Store)

opcode: Identifies the instruction as a store operation.

funct3: Specifies the size of the data being stored (e.g., byte, word).

rs1: Indicates the base register containing the memory address.

rs2: Specifies the register containing the data to be stored.

immediate: Provides the offset to calculate the effective memory address.

---

4. U-Type (Upper Immediate)

opcode: Identifies the instruction as a U-Type operation.

rd: Specifies the destination register where the upper immediate value will be stored or used.

immediate: Represents a 20-bit constant value, typically used for large address calculations or immediate constants.

---

5. SB-Type (Branch)

opcode: Specifies the instruction as a branch operation.

funct3: Encodes the condition type (e.g., equal, not equal).

rs1 and rs2: Provide the registers whose values are compared for the branch condition.

immediate: Specifies the signed offset for the branch target address, determining where to jump if the condition is met.

---

6. UJ-Type (Jump and Link)

opcode: Identifies the instruction as a jump operation.

rd: Specifies the destination register to store the return address for the jump.

immediate: Provides a 20-bit signed offset for the jump target address, enabling large jumps in the program.

Each field in these instruction types ensures efficient encoding and decoding for specific RISC-V operations.

# Roles of funct3 and funct7 in Load and Immediate Instructions

funct3 in Load Instructions: Specifies the size of data to be loaded and whether it is signed or unsigned.

Example: 000 for byte (LB), 001 for halfword (LH), 010 for word (LW), 100 for unsigned byte (LBU), and 101 for unsigned halfword (LHU).

funct7 in Load/Immediate Instructions: Not used in these instructions. These rely on opcode, funct3, and immediate fields for decoding.

---

2. Roles of funct3 and funct7 in Add and Sub Instructions (R-Type)

funct3: Specifies the operation type (e.g., addition, subtraction) within the arithmetic and logical instruction group.

Example: 000 for both addition (ADD) and subtraction (SUB).

funct7: Differentiates between ADD and SUB for the same funct3 value.

Example: 0000000 for ADD and 0100000 for SUB.

Thus, the combination of funct3 and funct7 uniquely identifies the specific operation.