**1. Register Transfer Language (RTL)**

**Definition:**

Register Transfer Language (RTL) is a way to describe how data moves between **registers** inside a computer and what **operations** are performed.

**Points (7 Marks):**

1. RTL uses **symbols** to describe data movement. o *Example:* R1 ← R2 means "copy data from R2 to R1".
2. RTL is used in designing the **CPU and control unit**.
3. It shows **what operation** happens and **which register** is involved.
4. Can also include **arithmetic or logic operations**. o *Example:* R1 ← R2 + R3 adds R2 and R3, stores result in R1.
5. RTL helps in **designing micro-operations**.
6. Each step of instruction execution is described in RTL.
7. RTL is like a **blueprint** for processor design.

**2. Register Transfer**

**Definition:**

Register transfer means moving data from one register to another within the CPU.

**Points (7 Marks):**

1. A register stores small amounts of data (like a mini memory box).
2. Register transfer = copy data from one register to another.

o *Example:* A ← B means move data from Register B to A.

1. Transfer is controlled by a **control signal**.
2. Happens in **one clock cycle**.
3. Used in performing CPU operations.
4. Very fast and done inside the CPU.
5. Basis of all CPU instructions.

**3. Bus and Memory Transfers**

**Definition:**

Bus is a common path used to move data between CPU, memory, and other devices. Memory transfer means reading or writing data to/from memory.

**Points (7 Marks):**

1. **Bus** carries data, address, and control signals.
2. There are **3 types** of buses: Data Bus, Address Bus, Control Bus.
3. **Memory read**: CPU takes data from memory.
4. **Memory write**: CPU sends data to memory.
5. Bus helps different parts of computer talk to each other.
6. Saves hardware by using one path instead of separate lines.
7. *Example:* Like a shared road for cars (data) to move between houses (memory and CPU).

**4. Arithmetic Microoperations**

**Definition:**

These are simple math operations like addition, subtraction, increment, or decrement done inside the CPU.

**Points (7 Marks):**

1. Performed using the **Arithmetic Logic Unit (ALU)**.
2. Common operations: Add, Subtract, Increment, Decrement.
3. Done between values stored in **registers**.
4. *Example:* R1 ← R2 + R3 adds R2 and R3.
5. Used in processing calculations.
6. Affects flags like Carry, Zero, Overflow.
7. Fast and done in one clock cycle.

**5. Logic Microoperations**

**Definition:**

These operations perform **bit-level logic** like AND, OR, NOT, XOR between data in registers.

**Points (7 Marks):**

1. Also done inside the ALU.
2. Operates on **binary bits** (0s and 1s).
3. Common logic ops: AND, OR, XOR, NOT.
4. *Example:* R1 ← R2 AND R3
5. Used in decision making and bit masking.
6. Doesn’t affect carry or overflow flags.
7. Very useful in hardware-level programming.

**6. Shift Microoperations**

**Definition:**

These operations move bits in a register left or right, either to multiply, divide or adjust data.

**Points (7 Marks):**

1. Shifts bits in registers.
2. **Left Shift**: multiplies by 2 **Right Shift**: divides by 2
3. Types: Logical Shift, Arithmetic Shift, Circular Shift.
4. *Example:* R1 ← R1 << 1 (Shift left)
5. Useful in fast math operations.
6. Helps in encoding, encryption, compression.
7. Affects bit positions but not actual values.

**7. Arithmetic Logic Shift Unit (ALSU)**

**Definition:**

This unit in the CPU performs **arithmetic, logic, and shift operations** together.

**Points (7 Marks):**

1. Combines functions of ALU and shifter.
2. Can do add, subtract, AND, OR, shift, etc.
3. Controlled by **control signals**.
4. Operates on data from registers.
5. *Example:* Like a smart calculator that can add, compare, and shift in one device.
6. Improves CPU efficiency.
7. Central part of data processing in CPU.