Register Transfer Language & Micro-operations

Course Module: Computer Organization & Architecture

---

1. Introduction

Modern digital computers execute operations using registers and micro-operations. To describe the movement of data and the operations performed on data stored in registers, we use a symbolic language called **Register Transfer Language (RTL)**. Understanding RTL is essential for studying CPU design and instruction execution.

---

2. Register Transfer Language (RTL)

Definition

Register Transfer Language is a symbolic notation used to specify:

- Transfer of data between registers
- Operations performed on data inside registers
- Control conditions for transfers

Basic Notation

- **R1 ← R2**
    - Means: Contents of register **R2** are transferred to **R1**
- **P: R1 ← R2**

- Means: If **control signal P = 1**, then transfer occurs

Components

Symbol	Meaning
R1, R2	Registers
←	Data transfer
+, -, AND, OR	Arithmetic / Logic operations
P	Control signal (Boolean expression)

Example

```
If (T = 1) then MAR ← PC
```

Meaning: When timing signal T becomes active, copy PC into MAR.

---

3. Register Transfer

A **register transfer** refers to moving binary data from one register to another.

Basic Transfer Example

```
R1 ← R2
```

Conditional Transfer Example

```
If (P = 1) then R3 ← R4
```

Parallel transfer

```
( R1, R2, R3 ) ← ( PC, MAR, MDR )
```

Multiple registers updated simultaneously.

---

4. Bus & Memory Transfer

Bus Transfer

Computers use a **bus** to connect multiple registers and memory.

- A **common bus system** allows many registers to share the same data path.

Bus Representation

```
R1 → Bus → R2
```

Memory Transfer

```
Read: MDR ← Memory[MAR]
```

```
Write: Memory[MAR] ← MDR
```

Signal	Operation

Read	Data transferred from memory to register
Write	Data moved from register to memory

---

5. Types of Micro-operations

Micro-operations are **elementary operations performed on data stored in registers** during a clock cycle.

Categories

Type	Example
Arithmetic Micro-operations	Add, Subtract, Increment
Logic Micro-operations	AND, OR, XOR
Shift Micro-operations	Left shift, Right shift

---

6. Arithmetic Micro-operations

Performed on integer binary values.

Common Arithmetic Operations

Operation	RTL Example	Description

Addition	R1 ← R1 + R2	Add registers
Subtraction	R1 ← R1 - R2	Subtract
Increment	R1 ← R1 + 1	Add 1
Decrement	R1 ← R1 - 1	Reduce by 1
Add with Carry	R1 ← R1 + R2 + Cin	ALU with carry

---

7. Logic Micro-operations

Bitwise operations on binary data.

Common Logic Operations

Operation	RTL Example	Meaning
AND	R1 ← R1 ∧ R2	Bitwise AND
OR	R1 ← R1 ∨ R2	Bitwise OR
XOR	R1 ← R1 ⊕ R2	Exclusive OR
NOT	R1 ← ¬R1	Complement

---

8. Shift Micro-operations

Used to shift data bits left or right in a register.

Types of Shifts

Operation	Example	Description
Logical Shift Left	R1 ← R1 << 1	Insert 0 on right
Logical Shift Right	R1 ← R1 >> 1	Insert 0 on left
Arithmetic Shift Left	Same as logical	
Arithmetic Shift Right	Preserve sign bit	
Rotate Left	R1 ← R1 rol 1	Circular shift
Rotate Right	R1 ← R1 ror 1	Circular shift

Example

```
R1 ← shl R1     (Left shift)

R1 ← shr R1     (Right shift)
```

---

9. Summary

Topic	Key Points
RTL	Language to specify register operations
Register Transfer	Moves data between registers
Bus Transfer	Data path for shared register communication
Memory Transfer	Read/write from/to memory
Arithmetic Micro-operations	Add, subtract, increment
Logic Micro-operations	AND, OR, XOR, NOT
Shift Micro-operations	Logical, arithmetic, rotate

---

10. Classroom Discussion Questions

1. What is difference between instruction and micro-operation?
2. Why do we need a bus system instead of direct register connections?
3. Explain arithmetic vs logic micro-operations with examples.
4. What is the difference between logical and arithmetic shift?