Unit 3

Basic Computer Organization and Design

✓*3.1 Instruction Codes*
✓*3.2 Computer Registers*
✓*3.3 Computer Instructions*
✓*3.4 Timing and Control*
✓*3.5 Instruction Cycle*
✓*3.6 Input Output and Interrupt*

Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist

3.1 INSTRUCTION CODES

An **instruction code** is a binary pattern stored in memory that tells the computer **what operation to perform**.

Every instruction has two main fields:

IC = Opcode + Operand

MOV A,B

MVI A,21h

LDA 2001h

MOV A,B **MVI A,21h** **LDA 2001h**		
Opcode	**Operand**	
MOV A,B		1 Byte
MVI A	21	2 Byte
LDA	2001h	3 Byte

1. Opcode (Operation Code)

- Specifies the **operation** to be performed: ADD, LOAD, STORE, JUMP, etc. Example:

  Opcode 0001 → ADD

  Opcode 0100 → LOAD

2. Address / Operand Field

- Specifies **where the data is located** (register or memory address).

  Example:

  Instruction: 0001 0100

  Opcode = 0001 → ADD

  Address = 0100 → memory location 4

Instruction Code Example (Detailed)

Suppose the instruction is:

LOAD 12

Binary representation (example format):

0100 1100

Where:

- 0100 = opcode for LOAD
- 1100 = binary for 12

Meaning:
Load Memory[12] into the Accumulator (AC).

Execution:

AR ← 12

DR ← M[12]

AC ← DR

Types of Instruction Code Formats

1. Zero-Address Instructions

Used in stack computers.

ADD → pops top two stack values, adds them

2. One-Address Instructions

Use AC as implicit register.

ADD X → AC ← AC + M[X]

3. Two-Address Instructions

ADD R1, R2 → R1 ← R1 + R2

4. Three-Address Instructions

ADD R1, R2, R3 → R1 ← R2 + R3

Instruction Format	No. of Address Fields	General Structure	Description	Example Instruction	Binary Example	How It Executes
Zero-Address (Stack Instruction)	0	Opcode Only	Used in stack-based machines; operations use implicit stack top.	ADD	0001	Pops top two values, adds them, pushes result back.
One-Address	1	Opcode + Address	Uses **Accumulator (AC)** as implicit operand.	ADD 40	0001 101000	AC ← AC + Memory[40]
Two-Address	2	Opcode + R1 + R2	Result stored in one of the registers.	ADD R1, R2	0001 001 010	R1 ← R1 + R2

Instruction Format	No. of Address Fields	General Structure	Description	Example Instruction	Binary Example	How It Executes
Three-Address	3	Opcode + R1 + R2 + R3	Used in high-level CPUs; allows separate source & destination.	ADD R1, R2, R3	0001 001 010 011	R1 ← R2 + R3
Register Format	Registers only	Opcode + Regs	Operands and result are all registers (no memory).	AND R3, R4	0010 011 100	R3 ← R3 AND R4
Immediate Format	Opcode + Register + Immediate Value	Value is inside instruction	Fast execution; data encoded directly.	ADI R1, #5	0101 001 00000101	R1 ← R1 + 5

3.2 COMPUTER REGISTERS

Registers are **high-speed storage units** inside the CPU.

Main Registers in Basic Computer Design

---

1. Program Counter (PC)

- Holds address of **next instruction**.
- Automatically increments after each fetch.

 Example:
 PC = 100
 Instruction at 100 executed
 PC ← 101

---

2. Instruction Register (IR)

- Holds the **currently fetched instruction**.

 Example:

 IR = 1010 1101 1111 (binary instruction)

 The Control Unit decodes this.

---

3. Memory Address Register (AR / MAR)

- Holds the **memory address** that the CPU wants to read/write.

  Example:

  AR ← 230

  Memory[230] accessed

---

4. Memory Data Register (DR / MDR)

- Holds data **coming from memory** or **going to memory**.

  Example:

  DR ← Memory[230]

---

5. Accumulator (AC)

- Used for arithmetic and logic operations.

  Example:

  AC ← AC + DR

---

6. Temporary Register (TR)

Used for intermediate operations.

---

7. Input Register (INPR)

Stores input from keyboard.

---

8. Output Register (OUTR)

Holds data to display/print on screen.

Real-Life Example of Register Use

Opening a calculator app on your phone:

1. *You press 9 → goes to **INPR***
2. *CPU moves it to **AC***

3. *You press +*
4. *You press 5 → INPR → AC*
5. *CPU performs AC ← AC + DR*
6. *Result is placed in* ***OUTR***
7. *Display shows* ***14***

All uses **register-to-register** transfers.

3.3 COMPUTER INSTRUCTIONS

A **computer instruction** is a binary-coded command that tells the CPU **what operation to perform**. Instructions are stored in memory, fetched by the CPU, decoded by the Control Unit (CU), and executed by the ALU, registers, and memory.

Each instruction typically contains:

1. **Opcode (operation code)** → tells WHAT to do
2. **Operand(s)** → tells WHAT data to use
3. **Addressing information** → tells WHERE data is located

---

1. Structure of an Instruction

A general instruction format:

[Opcode | Address/Operand]

Example:

0001 010010

Opcode = 0001 → ADD

Address = 010010 → 18 (decimal)

Meaning:

ADD 18 → AC ← AC + Memory[18]

---

2. Types of Computer Instructions

Computer instructions are broadly divided into 4 categories:

⭐ (A) Data Transfer Instructions

Used to move data between registers, memory, and I/O.

Common Data Transfer Instructions

Instruction	Meaning
LOAD X	AC ← M[X]
STORE X	M[X] ← AC
MOVE R1, R2	R1 ← R2
IN	AC ← INPR
OUT	OUTR ← AC

✔ Example 1: LOAD

LOAD 40

If Memory[40] = 55:

AC ← 55

✔ Example 2: STORE

STORE 100

If AC = 25:

Memory[100] ← 25

✔ Example 3: MOVE

R1 = 8, R2 = 12

MOVE R1,R2 → R1 = 12

⭐ (B) Arithmetic and Logic Instructions

Executed by the ALU.

Arithmetic Instructions

Instruction	Meaning
ADD X	AC ← AC + M[X]
SUB X	AC ← AC − M[X]
INC	AC ← AC + 1
DEC	AC ← AC − 1

Logic Instructions

Instruction	Meaning
AND X	AC ← AC AND M[X]
OR X	AC ← AC OR M[X]
XOR X	AC ← AC XOR M[X]
NOT	AC ← NOT(AC)

✔ Example (Arithmetic)

Assume:

AC = 10

Memory[25] = 13

Instruction:

ADD 25

Execution:

AC ← 10 + 13 = 23

✔ Example (Logic)

Assume:

AC = 1010

Memory[30] = 1100

Instruction:

AND 30

Execution:

1010 AND 1100 = 1000

AC = 1000

⭐ (C) Control / Branching Instructions

Change the normal flow of program execution.

Branch / Control Instructions

Instruction	**Meaning**
JUMP X	$PC \leftarrow X$
BZ X	If $AC = 0 \rightarrow PC \leftarrow X$
BNZ X	If $AC \neq 0 \rightarrow PC \leftarrow X$
CALL X	Procedure call
RETURN	Return from procedure
HALT	Stop execution

✔ Example 1: JUMP

JUMP 200

Execution:

$PC \leftarrow 200$ (Next instruction fetched from 200)

✔ Example 2: Conditional Branch

Assume AC = 0:

BZ 150

Since $AC = 0 \rightarrow$ branch taken:

$PC \leftarrow 150$

If $AC \neq 0 \rightarrow$ no branch.

✔ Example 3: CALL and RETURN

CALL 300 (jump to subroutine at 300)

RETURN (go back)

Used in functions/procedures.

⭐ (D) Input / Output (I/O) Instructions

I/O devices cannot directly communicate with ALU; they use INPR and OUTR.

I/O Instructions

Instruction	Meaning
IN	AC ← INPR
OUT	OUTR ← AC
ION	Enable interrupts
IOF	Disable interrupts

✔ Example: Keyboard Input

If keyboard enters value F (binary 01000110):

IN

AC ← INPR

✔ Example: Send Data to Display

OUT

OUTR ← AC

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa

Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa

Sasa