Unit 4

Microprogrammed Control

4.1 Control Memory

4.2 Address Sequencing

4.3 Computer Configuration

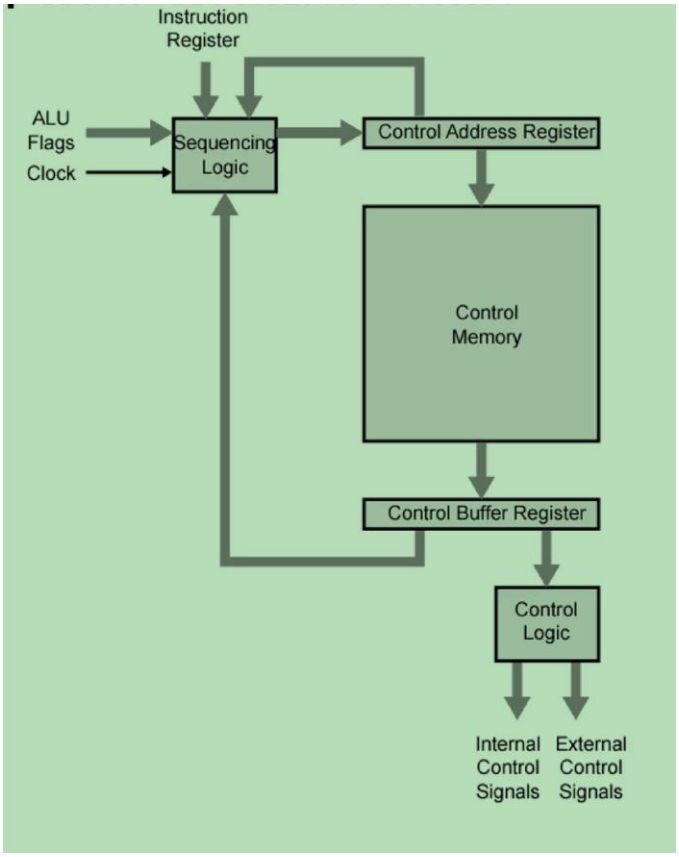
4.4 Microinstruction Format

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

MICROPROGRAMMED CONTROL

Microprogrammed control is a method of designing the **Control Unit (CU)** of a computer using **microinstructions stored in memory**, instead of using hardwired logic.

In this approach, control signals are generated by executing **microinstructions** stored in a special memory called **control memory**.



- Microprogrammed control uses microinstructions stored in control memory.
- Control memory stores microprograms for machine instructions.
- Address sequencing selects the next microinstruction.
- Computer configuration includes control memory, sequencer, registers, and ALU.
- Microinstruction format defines control signals and next-address logic.
- Widely used in complex CPU designs.

In 8085, one machine instruction (like ADD B) is executed using many internal micro-operations, which is the idea of microprogrammed control.

Unit Topic	**Meaning (Easy)**	**8085 Instruction**	**Internal Micro-operations (What happens inside CPU)**
4.1 Control Memory	Stores control steps (microinstructions)	`ADD B`	1. Fetch opcode from memory 2. Decode instruction 3. ALU adds A + B 4. Result stored in A
4.2 Address Sequencing	Decides next control step	`JZ 2050H`	If Zero flag = 1 → PC ← 2050H Else → PC ← PC + 1
4.3 Computer Configuration	Arrangement of registers, ALU, buses	`MOV A, C`	Data from register C → internal bus → register A
4.4 Microinstruction Format	Fields that control ALU, registers, next step	`INR A`	A ← A + 1 Flags updated Next instruction fetched

---

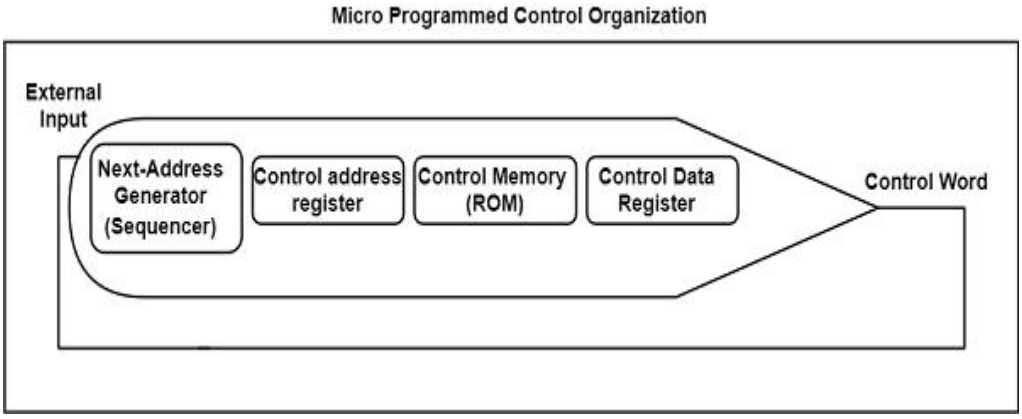
4.1 Control Memory

Control memory is a special type of memory used to store **microprograms**.
A **microprogram** is a sequence of microinstructions that control the execution of machine instructions.

Each microinstruction generates a set of control signals that:

- Transfer data between registers
- Select ALU operations

- Control memory read/write
- Control input/output



Key Points

- Control memory is usually implemented using **ROM** or **PLA**
- It is faster than main memory
- Each location stores **one microinstruction**

```
-----------------------------------
Control Memory (ADD B)
-----------------------------------
What CPU does internally
(micro-operations):
-----------------------------------
| Step | Micro-operation          |
| ---- | ------------------------ |
| µ1   | Fetch opcode from memory |
| µ2   | Decode ADD instruction   |
| µ3   | A ← A + B                |
| µ4   | Update flags             |
| µ5   | Go to next instruction   |
-----------------------------------
```

Example

Suppose the machine instruction is: **ADD X**

This single instruction is broken into microinstructions such as:

µ1: AR ← PC

µ2: IR ← M[AR]

µ3: PC ← PC + 1

µ4: AR ← IR(address)

µ5: DR ← M[AR]

µ6: AC ← AC + DR

All these microinstructions are stored in **control memory**.

4.2 Address Sequencing

Address sequencing is the process of **selecting the address of the next microinstruction** to be executed.

This task is handled by a component called the **Microprogram Sequencer**.

```
Address Sequencing (JUMP Instruction):  JMP 3000H
-------------------------------------------------
Internal steps:

Condition       Action
------------------------
Always          PC ← 3000H
-------------------------------------------------
Address sequencing decides → next microinstruction
                                address = 3000H.
-------------------------------------------------
```

Ways to Determine the Next Microinstruction Address

1. **Sequential Addressing**

- Next microinstruction is stored at the next memory location.
- Example: **µAddress ← µAddress + 1**

2. **Branching**
   - Microinstruction specifies the address of the next microinstruction.
   - Used for conditional execution.
   - Example: **If Z = 1 → go to µAddress 120**
3. **Mapping from Opcode**
   - Opcode of machine instruction is used to find the starting address of its microprogram.
   - Example: **Opcode ADD → microprogram starts at address 40**
4. **Conditional Branching**
   - Next address depends on condition flags (Z, C, etc.).

Example

For instruction JUMP:

If condition true → µAddress = branch address

Else → µAddress = µAddress + 1

4.3 Computer Configuration

Computer configuration refers to the **organization and interconnection of registers, ALU, buses, and control memory** required to support microprogrammed control.

Main Components

1. Control Memory
2. Control Address Register (CAR)
3. Microinstruction Register (MIR)
4. Microprogram Sequencer
5. ALU
6. Registers (AC, PC, IR, DR, AR, etc.)

```
Computer Configuration (MOV Instruction)
-----------------------------------------
MOV A, D
-----------------------------------------
Inside CPU:

Component       Action
-----------------------------------------
Register D      Places data on internal bus
Internal Bus    Transfers data
Register A      Receives data
-----------------------------------------
```

7. Internal Bus System

How It Works

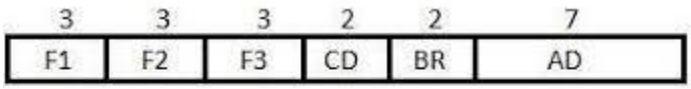
1. CAR holds the address of the current microinstruction
2. Control memory outputs the microinstruction
3. MIR stores the microinstruction
4. Control signals from MIR control registers, ALU, and buses
5. Sequencer decides next microinstruction address

Example

During execution of LOAD X:

- Control memory issues signals to load AR
- Next microinstruction causes memory read
- Another microinstruction loads AC

4.4 Microinstruction Format



F1, F2, F3: Microoperation fields
CD: Condition for branching
BR: Branch field
AD: Address field

What is a Microinstruction?

A microinstruction is a **low-level instruction** that specifies one or more micro-operations to be executed in a single clock cycle.

General Microinstruction Format

A microinstruction typically contains:

1. **Control Field**
2. **Condition Field**
3. **Next Address Field**

```
Microinstruction Format (INR A)
---------------------------------
INR A
---------------------------------
Microinstruction Fields (conceptual):
---------------------------------
Field                   Meaning
---------------------------------
ALU Control             Increment
Register Select         A
Flag Control            Update flags
Next Address            PC + 1
---------------------------------
👉 This shows how a single microinstruction
 controls multiple actions.
---------------------------------
```

Types of Microinstruction Formats

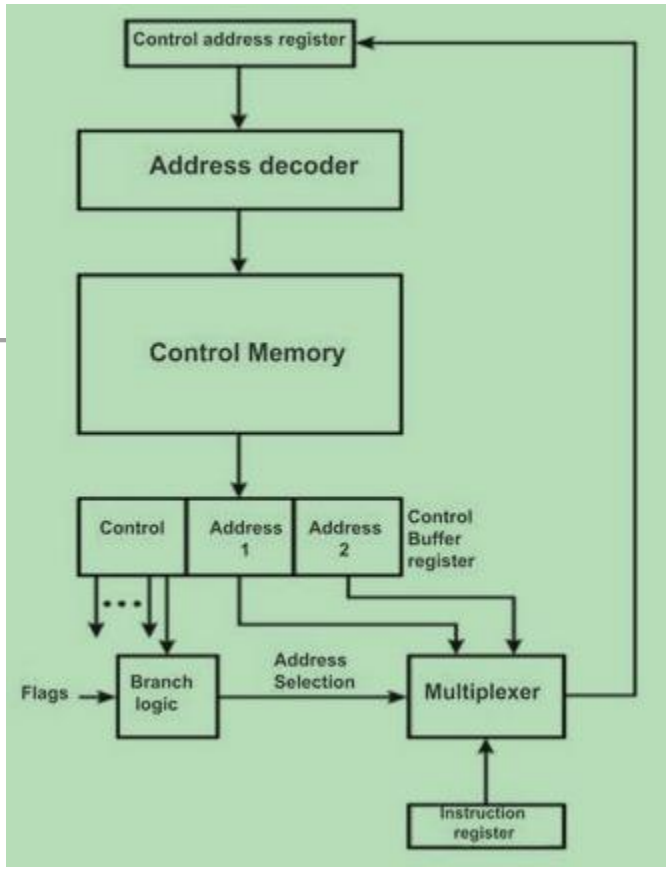
1. Horizontal Microinstruction Format

- Many control bits
- One bit per control signal

- Fast execution
- Requires large control memory

Format Example **| Control Signals | Next Address |**

Example **AR ← PC, DR ← M[AR], ALU = ADD**



---

2. Vertical Microinstruction Format

- Encoded control signals
- Fewer bits
- Slower than horizontal
- Smaller control memory

Format Example **| Opcode | Register Select | ALU Function | Next Address |**

Example of Microinstruction Format

Field	Purpose
F1	ALU operation (ADD, SUB, AND)
F2	Register transfer
F3	Memory operation
CD	Condition select
BR	Branch control
AD	Next microinstruction address

Example Microinstruction

F1 = ADD

F2 = AC ← AC + DR

F3 = None

CD = Z

BR = Conditional

AD = 120

Meaning:
If Zero flag is set, go to microinstruction at address 120.

Example: Execution of ADD Instructi on (Microprogram Level)

μ1: AR ← PC

μ2: IR ← M[AR]

μ3: PC ← PC + 1

μ4: AR ← IR(address)

μ5: DR ← M[AR]

μ6: AC ← AC + DR

Each step corresponds to **one microinstruction stored in control memory**.

Advantages of Microprogrammed Control

- Easy to design and modify
- Instruction set can be changed by updating microprogram
- Less complex than hardwired control
- Suitable for complex instruction sets (CISC)

Disadvantages

- Slower than hardwired control

- Control memory cost
- Extra memory access time