EE321

Computer Architecture

Chap. 04: Control Unit

Dr. Abdelhakim Khouas

Email : akhouas@hotmail.com

ab.khouas@univ-boumerdes.dz

IGEE (ex. INELEC)

University M'hamed Bougara of Boumerdes

Course chapters

1. Review of Digital Design
2. Top Level of Computer
3. Central Processing Unit (CPU)
4. Control Unit
5. Memory
6. Instruction Set and Addressing Modes

Lecture Objectives

Understand function and design of the control unit

- Micro-operations
- Control unit function
- Control unit implementations
  - Hardwired implementation
  - Microprogrammed implementation

Lecture Outline

1. CPU Function Review
2. Micro-operations
3. Control Unit
4. Hardwired Implementation
5. Microprogrammed Control
    1. Basic Concepts
    2. Implementation
    3. Microinstructions

Readings

Textbook

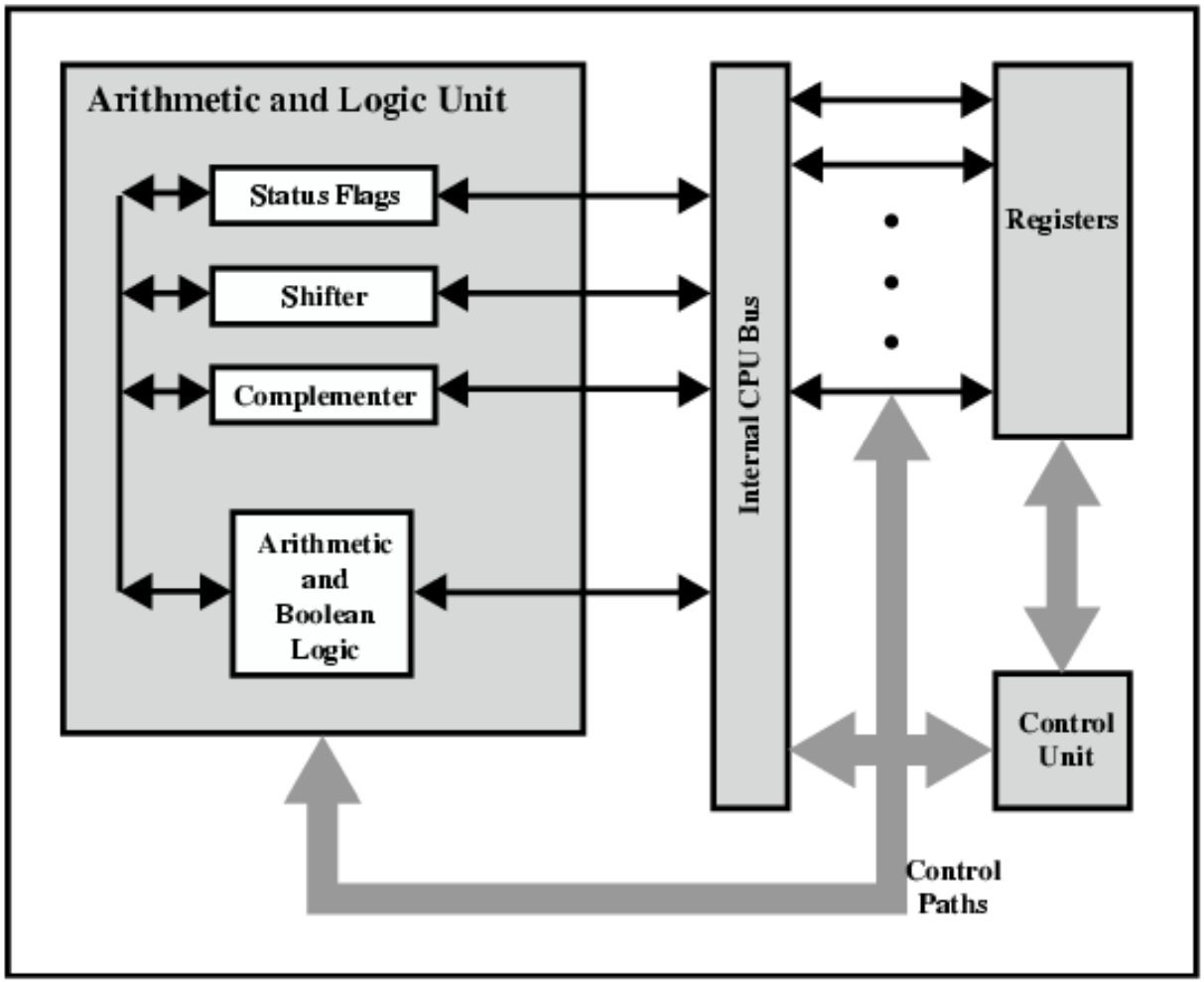
Computer Organization and Structure, Designing for Performance, By William Stallings, 8th edition

Sections

- Chapter 15, sections: 3.1 to 3.3
- Chapter 16, sections: 16.1

1. CPU Function Review

CPU internal structure



1. CPU Function Review

CPU Function

We have already seen that:

- The execution of a program consists of the sequential execution of instructions
- The execution of an instruction involves the execution of a sequence of cycles: fetch, decode, and execute
- Each cycle contains many smaller cycles (states) such as: read instruction, data operation, operand save, ... etc.

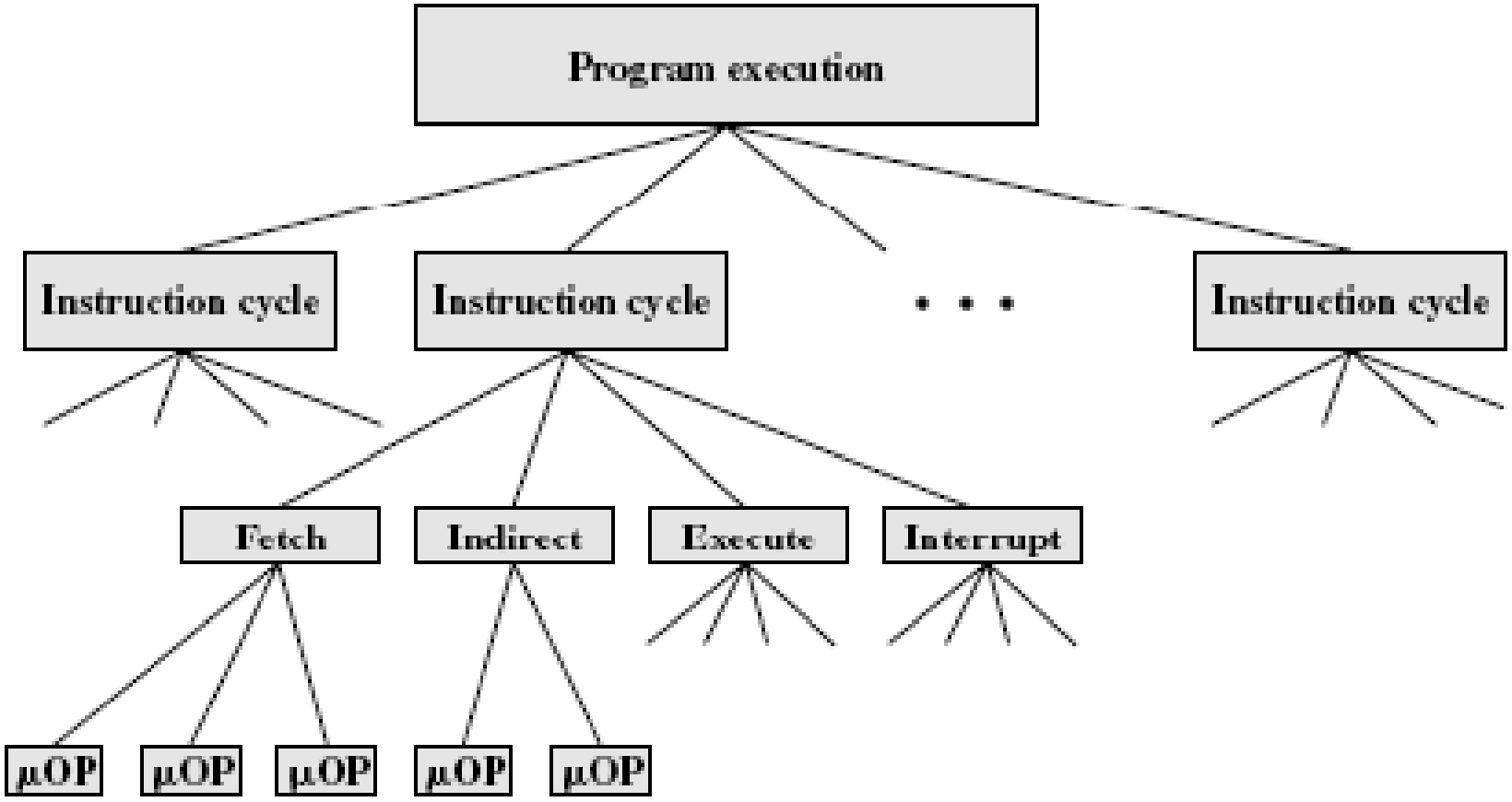
2. Micro-operation

Definition

Micro-operations (MO) are the atomic operations (smaller steps) of a processor that are executed in one clock cycle

Each instruction cycle (fetch, decode, execute, interrupt) is composed of a set of micro-operations that are performed in a precise time sequence

2. Micro-operation



Constituent Elements of a Program Execution

2. Micro-operation

Examples:

For all instructions

- Fetch Cycle: 3 steps and 4 micro-operations
  - t1: MAR <-- PC
  - t2: MBR <-- memory
  - PC <-- PC+I (I is the instruction length)
  - t3: IR <-- MBR

- t1, t2, and t3 are time unit (i.e. one or more clock cycles)

2. Micro-operation

Micro-operations grouping

To save time, some micro-operations that do not interfere can be executed during same time unit, following two simple rules:

- The proper sequence of events must be followed (e.g. set MAR register before writing or reading MBR)
- Conflicts must be avoided
  - should not attempt to read to and write from the same register in one time unit
  - If internal data bus, should not use the same data bus

2. Micro-operation

Micro-operations grouping

- For fetch cycle, the 3rd micro-operation could have been grouped with the 4th micro-operation
  - t1: MAR <-- PC
  - t2: MBR <-- memory
  - t3: IR <-- MBR
  - PC <-- PC+I (I is the instruction length)

2. Micro-operation

Examples: Direct add operation

ADD R1, (X) ; R1 <= R1 + [X]

- R1 : General purpose register used as accumulator
- X : Second operand (address)
- [X]: content of memory location X
- MBR is connected to ALU

- Execution Cycle:
  - t1: MAR <-- IR (X)
  - t2: MBR <-- memory
  - t3: R1 <-- R1 + MBR

2. Micro-operation

Examples: Immediate operation

ADI R1, X ; R1 <= R1 + X

- R1 : General purpose register used as accumulator
- X : Second operand (immediate value)
- MBR is connected to ALU

- Execution Cycle:
  - T1: R1 <-- R1 + MBR(X)

2. Micro-operation

Examples:

ISZ (X) ; Increment [X] and skip the next instruction if [X]=zero, X is the address of the operand

- Execution Cycle:
  - t1: MAR ← IR(X)
  - t2: MBR ← Memory
  - t3: MBR ← MBR + 1
  - t4: Memory ← MBR

    If (MBR = 0) then (PC ← PC + 1)

2. Micro-operation

Examples:

- Interrupt Cycle:
  - $t1_1$: MBR ← PC
  - $t2_2$: MAR ← Save_Address
    PC ← Routine_Address
  - $t3_3$: Memory ← MBR

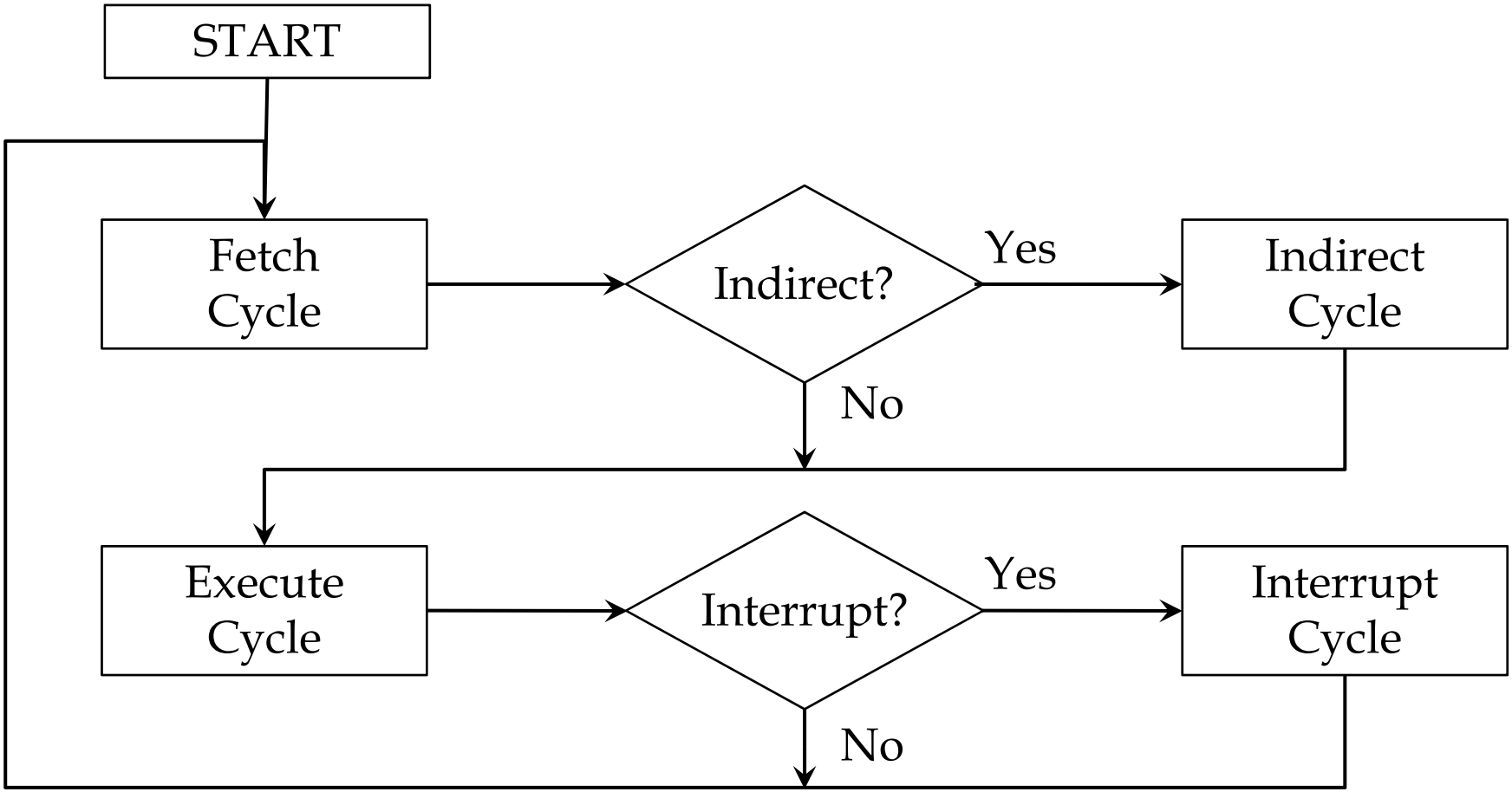
2. Micro-operation

Examples:

Indirect Operation: ADIN A, ((X)) ; A = A + [[X]]

- Indirect Cycle
  - t1: MAR ← IR(X)
  - t2: MBR ← memory ([X])
  - T3: IR(Address) ← MBR
- Execute Cycle
  - t3: MAR ← IR(X)
  - t4: MBR <-- memory ([[X]])
  - t5: A <-- A + MBR

2. Micro-operation



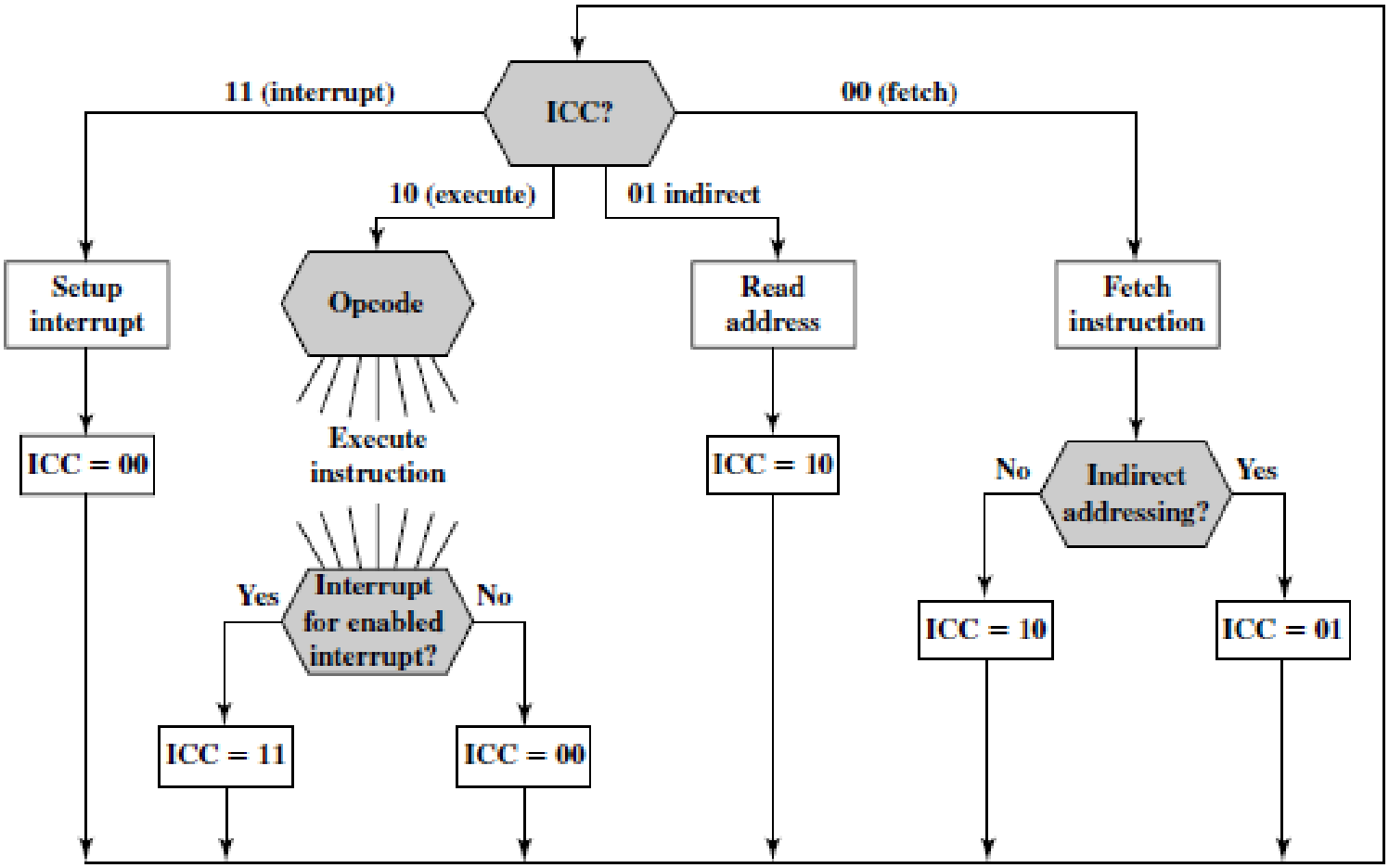
Flowchart for instruction cycle

2. Micro-operation

Instruction cycle code

- We have four cycles: fetch, indirect, execute, and interrupt
- We can use 2-bit register ICC (Instruction Cycle Code) to code the cycles
  - ICC = 00: Fetch
  - ICC = 01: Indirect
  - ICC = 10: Execute
  - ICC = 11: Interrupt
- At he end of each cycle, ICC is set appropriately

2. Micro-operation



Flowchart for instruction cycle using ICC

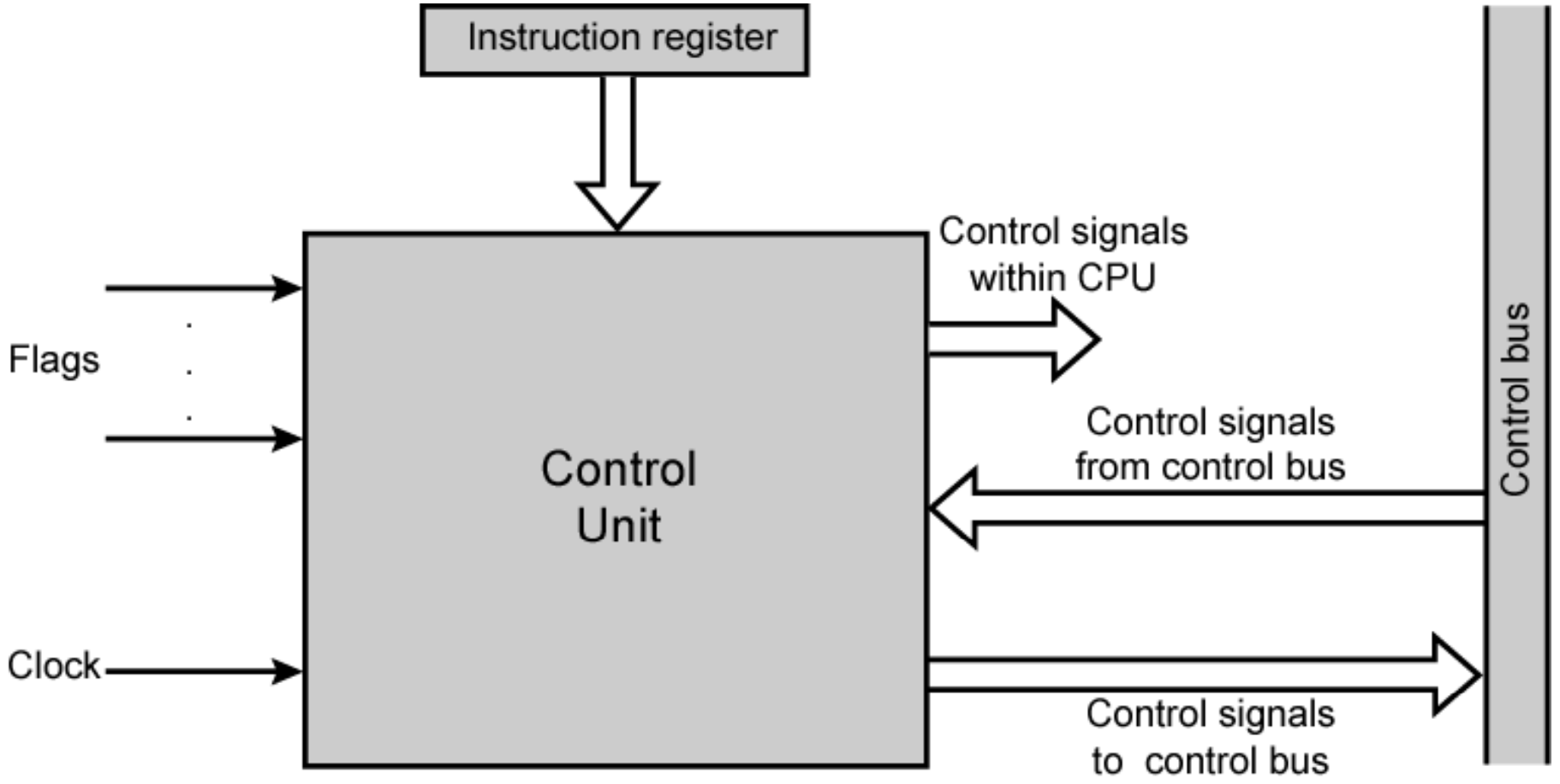
3. Control Unit

The control unit controls the execution of each step of the instruction cycle

The control unit of a CPU performs two tasks:

1. It causes the processor to step through a series of MO in the proper sequence
2. It generates the control signals that cause the execution of each MO

3. Control Unit



Control Unit Model

3. Control Unit

Inputs:

- Clock: The control unit causes one micro-operation (or a set of simultaneous micro-operations) to be performed for each clock pulse. Also known as the processor cycle time, or the clock cycle time
- IR: The code of the inst. is used to determine which micro-operations to perform during the execute cycle
- Flags: These are needed by the control unit to determine the status of the processor and the outcome of previous ALU operations
- Control signals from control bus

3. Control Unit

Outputs:

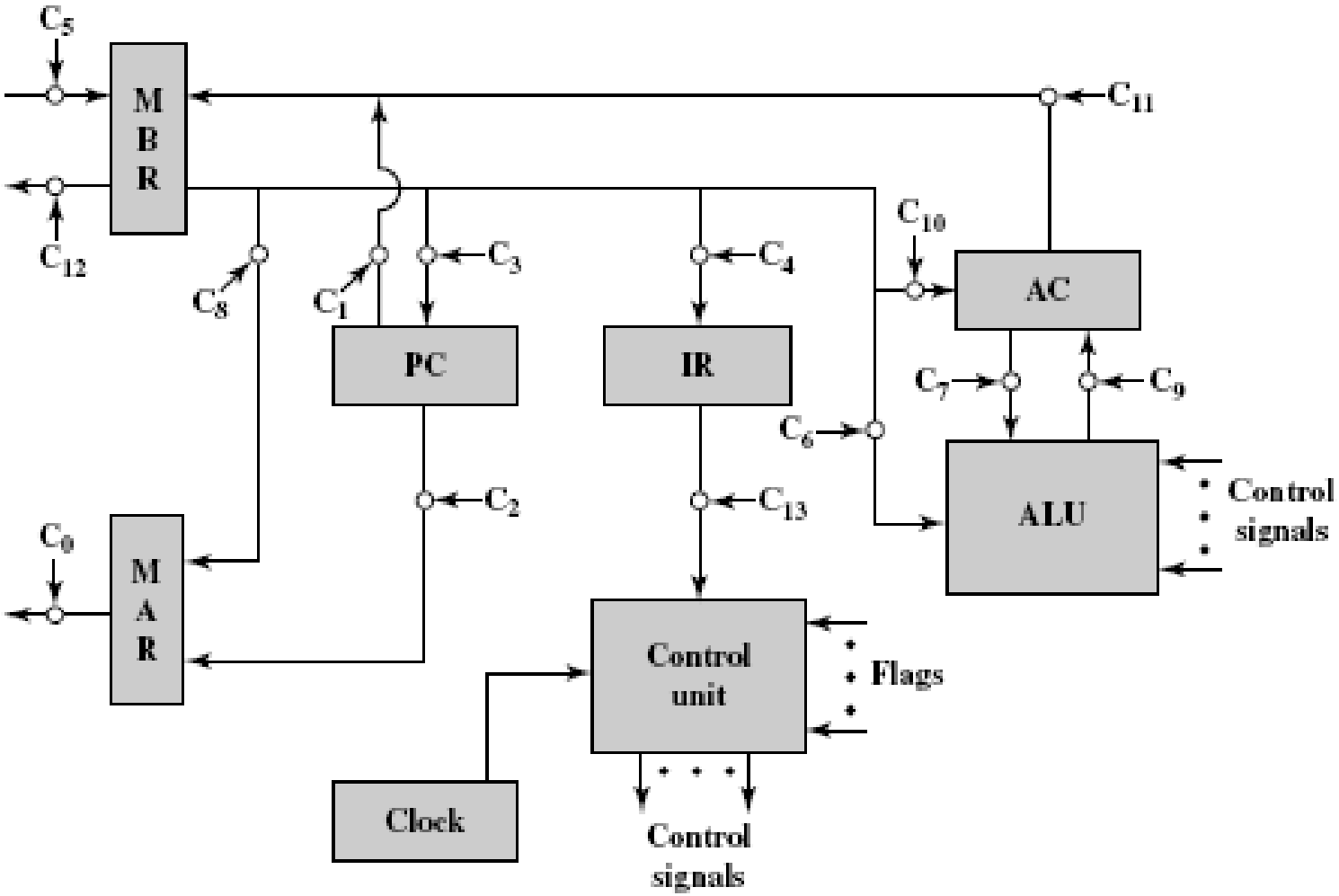
- Control signals within the processor: These are two types:
  - Signals that cause data movement (register to another)
  - Signals that activate specific ALU functions
- Control signals to control bus: These are also of two types:
  - control signals to memory (e.g. memory read)
  - control signals to the I/O modules

3. Control Unit

Example Control Signal Sequence:

- MAR <- (PC)
  - Control unit activates signal to open gates between PC and MAR
- MBR <- (memory)
  - Open gates between MAR and address bus
  - Memory read control signal
  - Open gates between data bus and MBR

3. Control Unit



Data Paths and Control signals: CPU with single accumulator

Source: Computer Organization and Architecture, by W. Stallings

3. Control Unit

Control Signals

- The terminations of the control signals are labeled Ci and indicated by a circle
- The control unit receives inputs from the clock, IR, and flags
- With each clock cycle, the control unit reads all of its inputs and emits a set of control signals Ci to open/close the gates between the different parts of the processor to control data transfer

Control Signals: examples

- Fetch Cycle:

Micro-operation	Active signals	Micro-inst.
t1: MAR <-- PC	C3	00...00100
t2: MBR <-- memory 10...10000	C5, RD	
PC <-- PC+1		
t3: IR <-- MBR	C4	00...01000

RD: Read control signal to bus

Control Signals: examples

- Interrupt Cycle:

Micro-operation	Active signals	Micro-inst.
$t1_1$: MBR ← PC	C1	00…00001
$t2_2$: MAR ← Save_Address		
PC ← Routine_Address		
$t3_3$: Memory ← MBR	C12, WR	
101…0000		

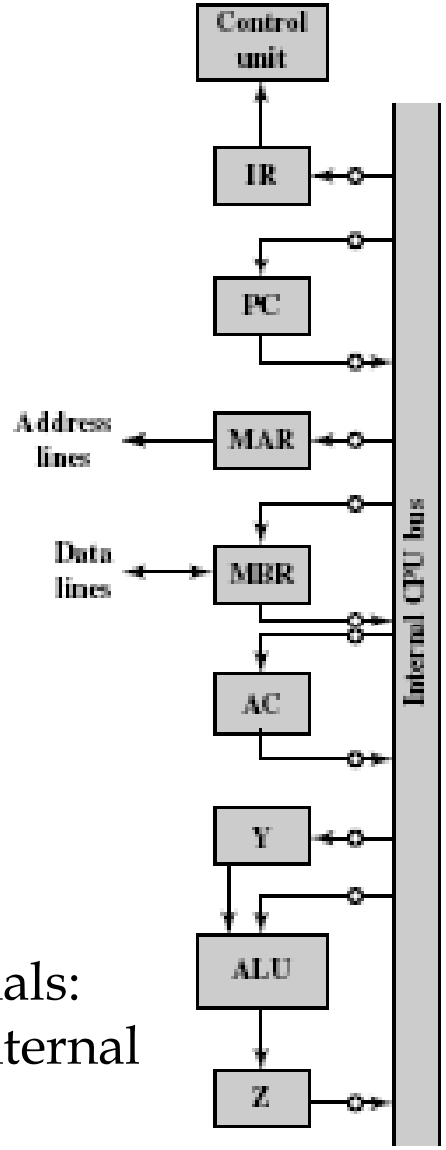
WR: Write control signal to bus

3. Control Unit

t_1: MAR ← (IR(address))
t_2: MBR ← Memory
t_3: Y ← (MBR)
t_4: Z ← (AC) + (Y)
t_5: AC ← (Z)

Micro-operations for **`ADD AC,X`** instruction: 5 clock cycles.

AC: Accumulator
X : Address



Control signals: CPU with internal data bus

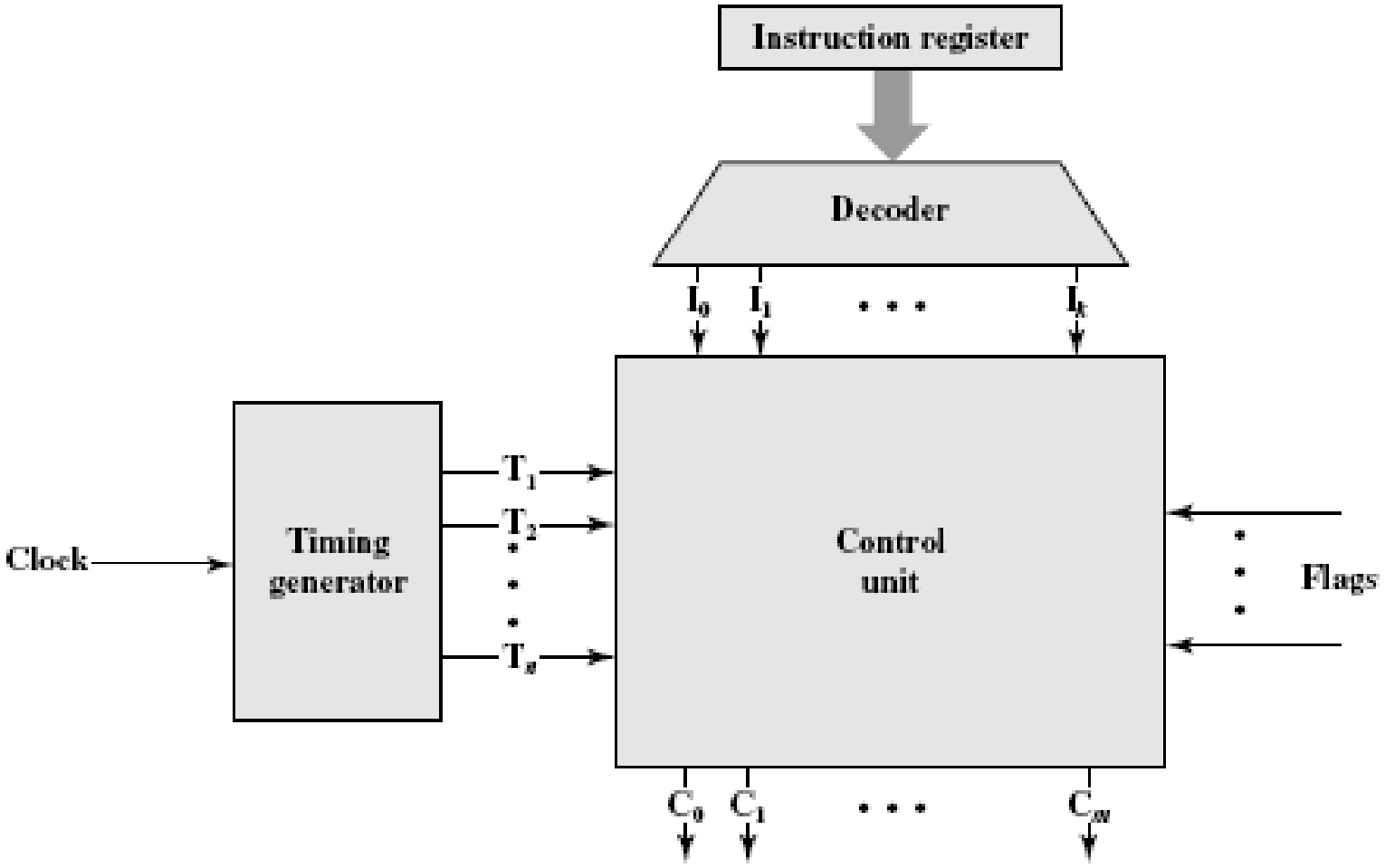
Source: Computer Organization and Architecture, by W. Stallings

3. Control Unit

We have discussed the control unit in terms of its inputs, output, and functions. For the implementation, a wide variety of techniques have been used. Most of these fall into one of two categories:

- Hardwired implementation
- Microprogrammed implementation

4. Hardwired Implementation



Control Unit with Decoded inputs

4. Hardwired Implementation

Control unit is viewed as a state machine circuit, and can be implemented using standard digital logic techniques

- For each control signal, we derive a Boolean expression as a function of the inputs and the current state
- Decoder is used to generate unique logic signal for each opcode

4. Hardwired Implementation

Example: Let us consider a control signal C5 that causes data to be read from the external data bus into the MBR (MBR<-- Memory)

- We assume two new control signal P and Q that have the following interpretation:
  - PQ = 00 Fetch Cycle
  - PQ = 01 Indirect Cycle
  - PQ = 10 Execute Cycle
  - PQ = 11 Interrupt Cycle

4. Hardwired Implementation

	Micro-operations	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory PC ← (PC) + 1	C_5, C_R
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address PC ← Routine-address	
	t_3: Memory ← (MBR)	C_{12}, C_W

C_R = Read control signal to system bus.

C_W = Write control signal to system bus.

Micro-operations and control signals

4. Hardwired Implementation

Example:

- The control signal C5 will be asserted during the second time unit of the fetch and interrupt cycles, so the Boolean expression of control signal C5 is:

$$C5 = \overline{P}\,\overline{Q}T_2 + \overline{P}QT_2$$

- If we assume C5 is also needed during the execute cycle of three instructions that read from memory: LDA, ADD, and AND, then C5 is given by:

$$C5 = \overline{P}\,\overline{Q}T_2 + \overline{P}QT_2 + P\overline{Q}T_2(LDD + ADD + AND)$$

4. Hardwired Implementation

Example: Signal C4 (IR <-- MBR)

- The control signal C4 will be asserted during the third time unit of the fetch and indirect cycles, so the Boolean expression of control signal C4 is:

$$C4 = \overline{P}\,\overline{Q}T_3 + PQT_3$$

4. Hardwired Implementation

Main advantages

- High(er) speed operation
- Smaller implementations (component counts)

Main disadvantages

- Inflexible design
  - Modifications to the design can be hard to do
- Difficult to add new instructions
- Complex sequencing & micro-operation logic
- Difficult to design and test

5. Microprogrammed Control

An alternative to a hardwired control unit is a microprogrammed control unit

- Control unit is specified by a microprogram
- Microprogram (firmware) consists of a sequence of microinstructions (MI)
- Each MI specify a micro-operation and is very simple

5. Microprogrammed Control

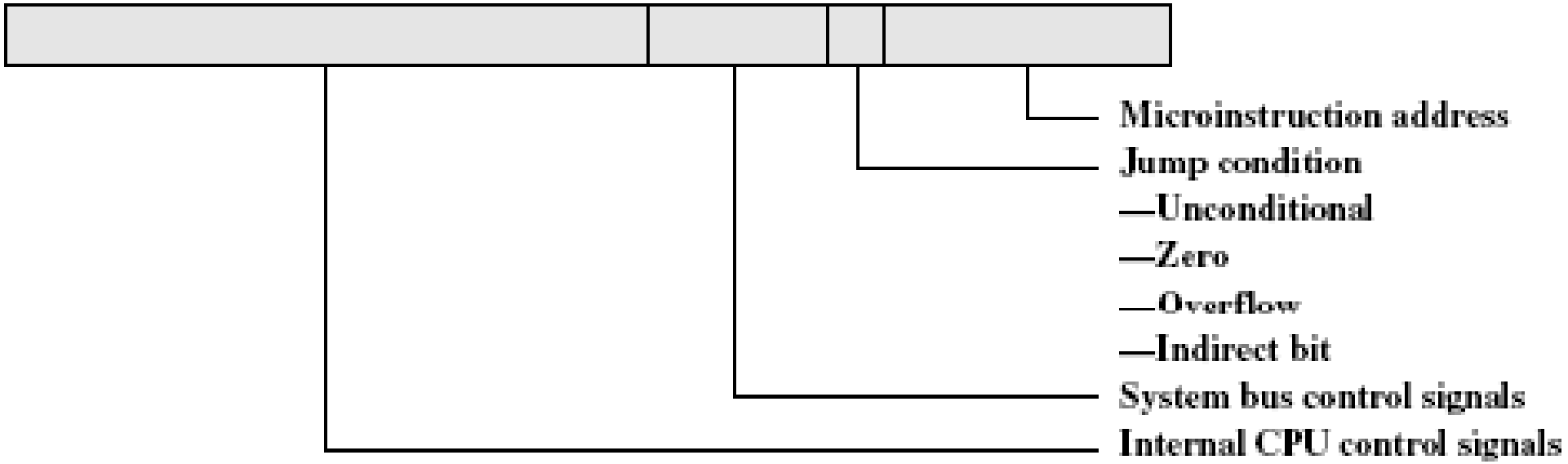
Implementation

The control unit only generate a set of control signals, each control signal is on or off

- In horizontal MI, each bit represents one control signal
- We put each MI in a memory with unique address
- Add to the MI an address to specify the next micro-instruction, depending on conditions

5. Microprogrammed Control

Horizontal MI format



Source: Computer Organization and Architecture, by W. Stallings

5. Microprogrammed Control

Horizontal MI format

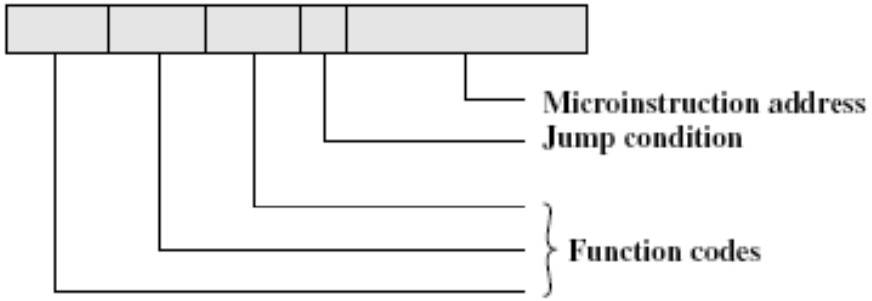
- A code is used for each action to be performed
- Condition branch field
- Address of the next MI if branch
- No need for microinstruction decoder
- Bigger (more bits) microinstruction and control memory

Source: Computer Organization and Architecture, by W. Stallings

5. Microprogrammed Control

Vertical MI format

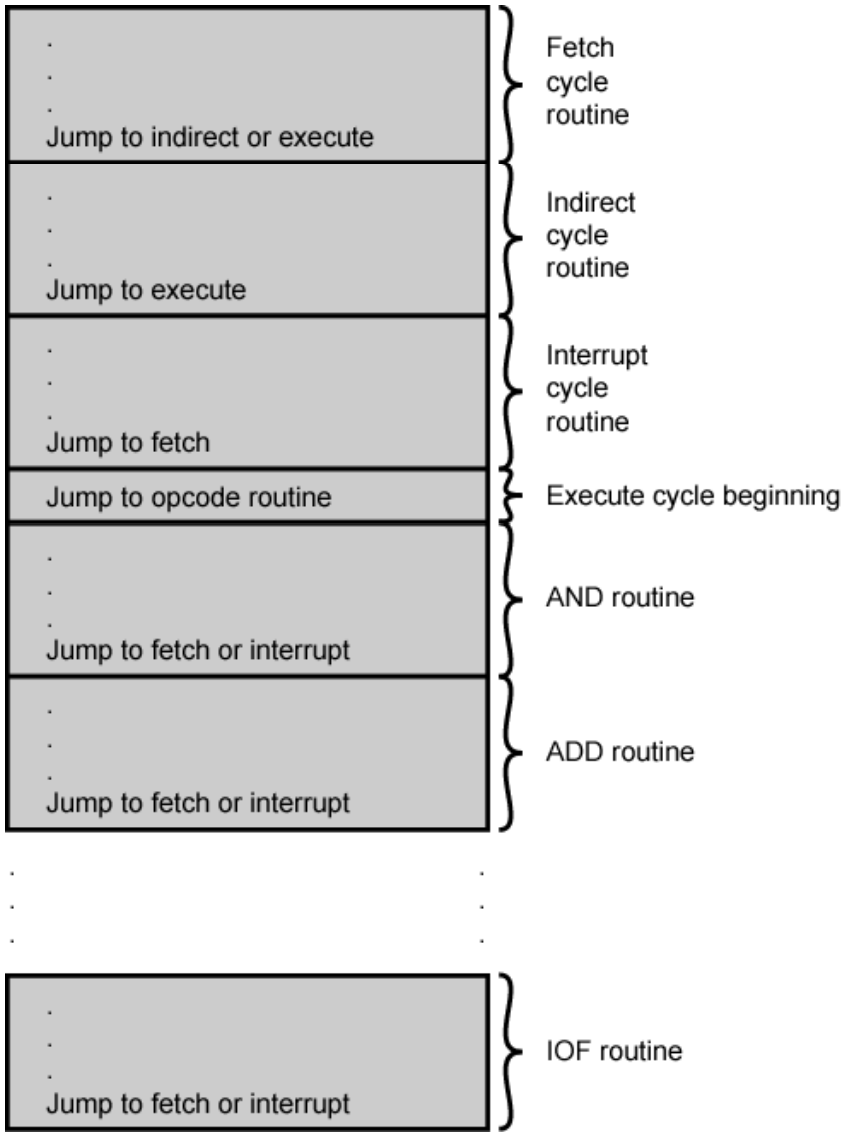
- Condition branch field
- Address of the next MI if branch
- Need for microinstruction decoder to generate control signals
- Smaller control memory



Vertical microinstruction format

Source: Computer Organization and Architecture, by W. Stallings

5. Microprogrammed Control



Organization of Control Memory

5. Microprogrammed Control

MI is interpreted as follows:

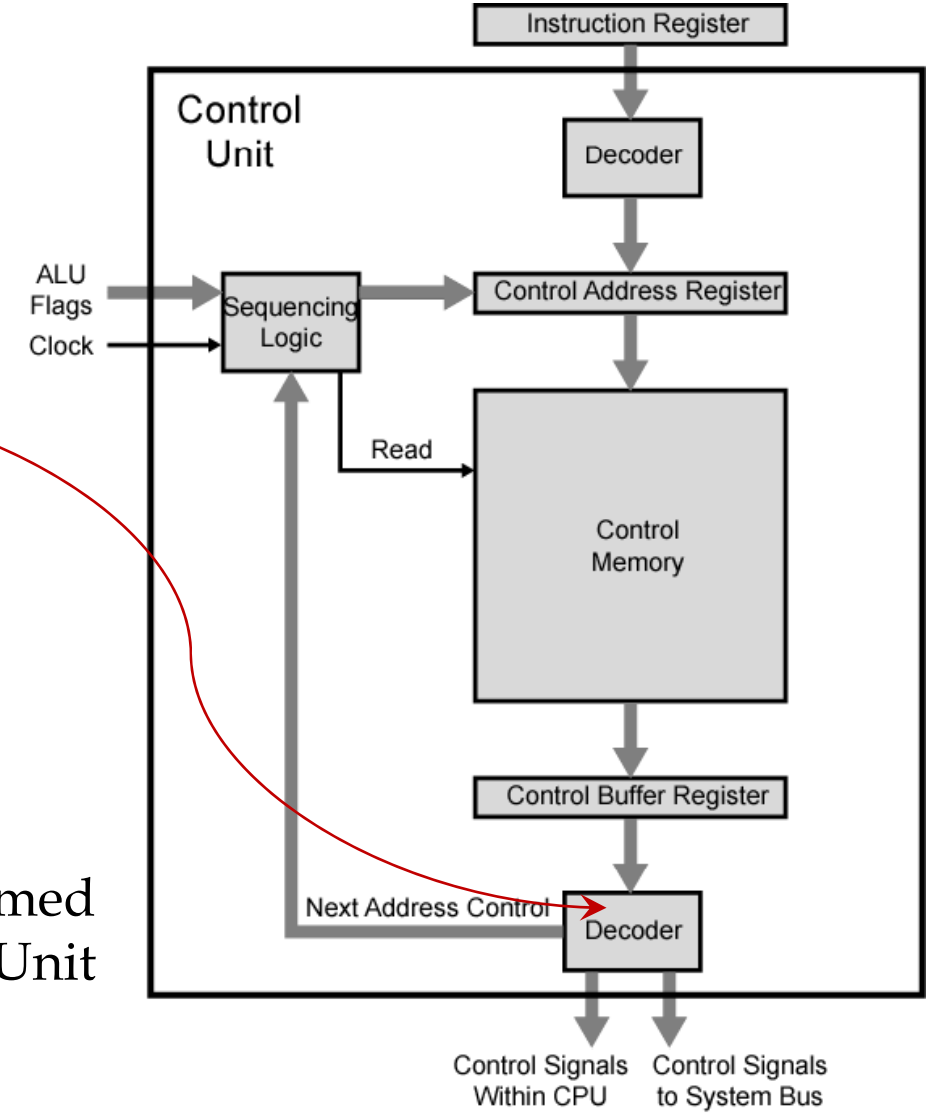
i. To execute this MI, turn on/off all the control signals, the resulting control signals will cause one or more micro-operations to be performed

ii. If the condition indicated by the condition bits is: false, execute the next microinstruction in sequence

iii. If it is true, the next microinstruction to be executed is indicated in the address field

5. Microprogrammed Control

This decoder is only used for vertical microinstruction



Functioning of Microprogrammed Control Unit

5. Microprogrammed Control

The control unit functions as follow:

- To execute an instruction, the sequencing logic unit issues a READ command to the control memory
- The control word is read into the control buffer register
- The content of the control buffer register generates control signals and next address information for the sequencing logic unit
- The sequencing logic unit loads a new address into the control address register based on the next-address information and the ALU flags

5. Microprogrammed Control

Next address decision

Depending on ALU flags and control buffer register, one of the three following decision is made:

1. Add 1 to control address register
   - Add 1 to control address register
2. Jump to new routine based on jump microinstruction
   - Load control address register with address field of control buffer register
3. Jump to machine instruction routine
   - Load control address register based on opcode in IR

Summary

Micro-operation

Control Unit Function

- Inputs
- Outputs
- Control signals

Control Unit Implementation

- Hardwired implementation
- Microprogrammed Control