

Control Unit Operation

Computer Organization and Architecture

*SOUTH CHINA UNIVERSITY OF TECHNOLOGY
FALL 2016*

*DR. MAO Aihua
ahmao@scut.edu.cn*

Topics

- Micro-Operations
- Control of the Processor
- Hardwire Implementation

Specify the functions of processor

- Operations (opcodes)
 - Addressing modes
 - User visible registers
 - I/O module interface
 - Memory module interface
 - Interrupt processing structure
- } Instruction set
- } System bus
- OS

How these functions are performed and controlled?

3

Review: Micro-Operations

- A computer executes a program **consists of a sequence of instruction cycles**
- Each instruction cycle **has a number of subcycles**
 - see pipelining



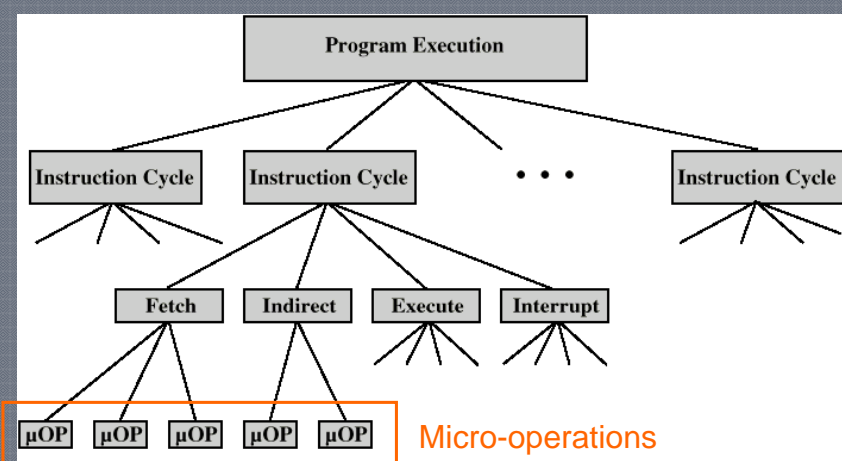
4

Review: Micro-Operations

- In the small subcycles of the instruction cycle, **Fetch/execute** cycles always occur
- Each of the small subcycles involves a series of steps, **called micro-operations**
- Each step does very little

5

Constituent Elements of Program Execution



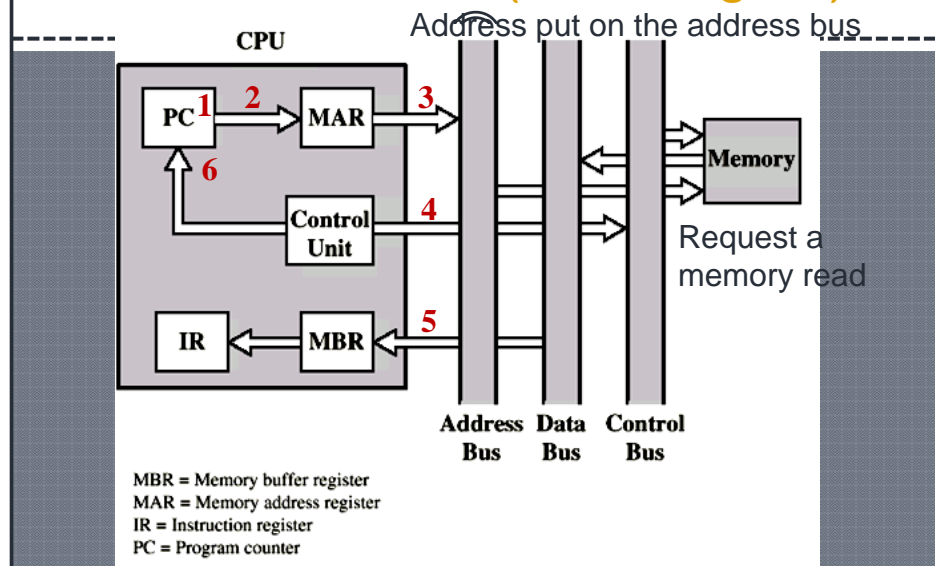
6

Fetch - 4 Registers

- Memory Address Register (MAR)
 - Connected to address bus
 - Specifies address for read or write op
- Memory Buffer Register (MBR)
 - Connected to data bus
 - Holds data to write or last data read
- Program Counter (PC)
 - Holds address of next instruction to be fetched
- Instruction Register (IR)
 - Holds last instruction fetched

7

Review: Data Flow (Fetch Diagram)



8

Fetch Sequence

- Address of next instruction is in **PC**
- **Address (MAR)** is placed on address bus
- Control unit issues **READ** command
- Result (data from memory) appears on **data bus**
- Data from data bus copied into **MBR**
- PC incremented by 1 (in parallel with data fetch from memory)
- Data (instruction) moved from **MBR to IR**
- MBR is now free for further data fetches

9

Fetch Sequence

MAR	
MBR	
PC	0000000001100100
IR	
AC	

(a) Beginning (before t_1)

MAR	0000000001100100
MBR	
PC	0000000001100100
IR	
AC	

(b) After first step

MAR	0000000001100100
MBR	0001000000100000
PC	0000000001100101
IR	
AC	

(c) After second step

MAR	0000000001100100
MBR	0001000000100000
PC	0000000001100101
IR	0001000000100000
AC	

(d) After third step

Figure 15.2 Sequence of Events, Fetch Cycle

10

Fetch Sequence (symbolic)

- t1: $MAR \leftarrow (PC)$
- t2: $MBR \leftarrow \text{memory}$
- $PC \leftarrow (PC) + I$
- t3: $IR \leftarrow (MBR)$
- Each micro-operation can be performed within a single time unit

11

Fetch Sequence

- The simple fetch cycle
 - Consists of **three steps**
 - **Four micro-operations**
 - ✦ Each involve data movement in or out of a register
 - If these movement do not interface with one another, several of them can take place during one time step, **saving time**

12

Fetch Sequence (symbolic)

- t1: $MAR \leftarrow (PC)$
- t2: $MBR \leftarrow \text{memory}$
- $PC \leftarrow (PC) + I$
- t3: $IR \leftarrow (MBR)$
- Each micro-operation can be performed within a single time unit

13

Fetch Sequence (symbolic)

- The third micro-operation could have grouped with the forth **without affecting the fetch operation**
- t1: $MAR \leftarrow (PC)$
- t2: $MBR \leftarrow \text{memory}$
- t3: $PC \leftarrow (PC) + I$
- $IR \leftarrow (MBR)$

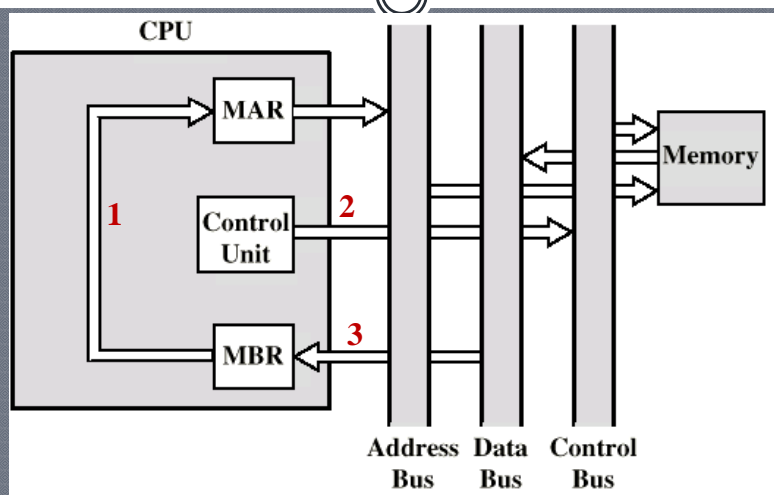
14

Rules for Clock Cycle Grouping

- Proper sequence must be followed
 - $MAR \leftarrow (PC)$ must precede $MBR \leftarrow (\text{memory})$
- Conflicts must be avoided
 - Must not read & write same register at same time
 - $MBR \leftarrow (\text{memory})$ & $IR \leftarrow (MBR)$ must not be in same cycle
- Also: $PC \leftarrow (PC) + 1$ involves addition
 - Use ALU to avoid duplication of circuitry

15

Review: Data Flow (Indirect Diagram)



16

Indirect Cycle (Fetch Operand)

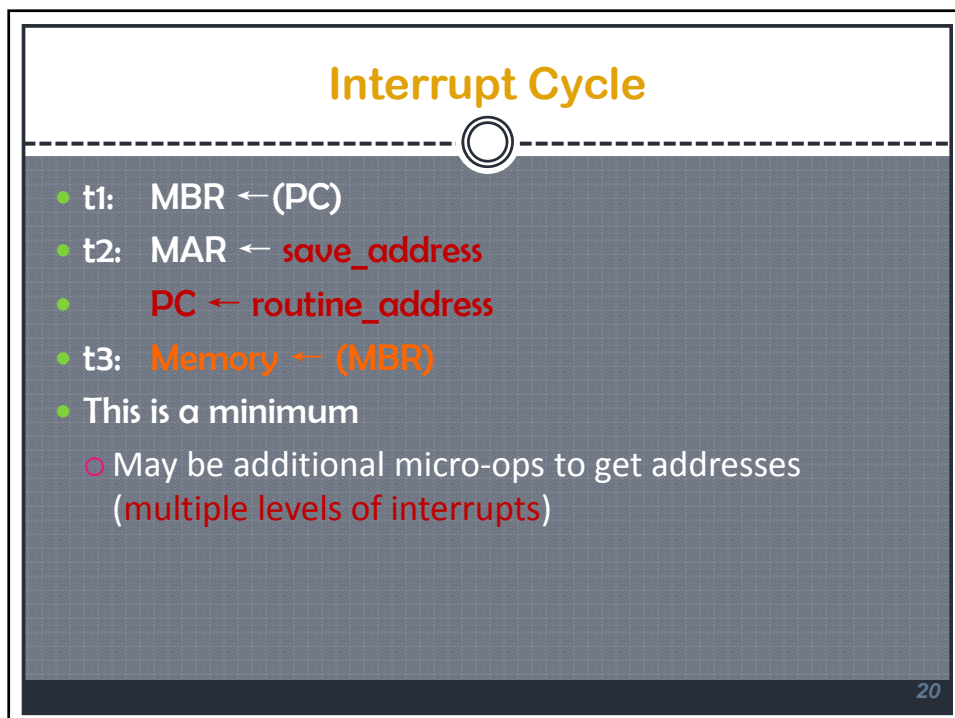
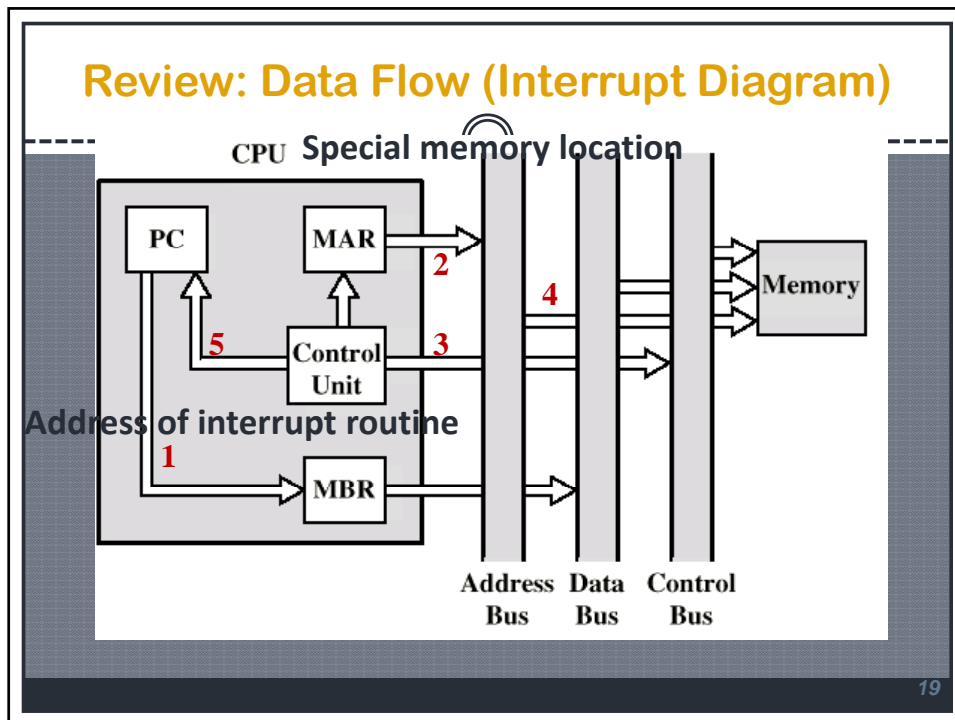
- If the instruction specifies an **indirect address**
- $MAR \leftarrow (IR(\text{address}))$ - **address portion of IR**
- $MBR \leftarrow \text{Memory}$
- $IR(\text{address}) \leftarrow (MBR(\text{address}))$
- Finally, IR is updated from MBR, and contains an **direct address**
- IR is now in same state as if direct addressing had been used

17

Indirect Cycle (Fetch Operand)

- t1: $MAR \leftarrow (IR(\text{address}))$
- t2: $MBR \leftarrow \text{Memory}$
- t3: $IR(\text{address}) \leftarrow (MBR(\text{address}))$

18



Execute Cycle (ADD)

- Different for each instruction, with N different opcodes, there are N different sequence of micro-operations
- e.g. **ADD R1,X** : add the contents of location X to Register 1, result in R1
- t1: $MAR \leftarrow (IR(\text{address}))$ (*the address of add instruction*)
- t2: $MBR \leftarrow \text{memory}$
- t3: $R1 \leftarrow R1 + (MBR)$
- Additional micro-operations may be required to **extract the register reference from IR**

21

Execute Cycle (ISZ)

- ISZ X - increment and skip if zero
 - t1: $MAR \leftarrow (IR(\text{address}))$ (address to save MBR)
 - t2: $MBR \leftarrow \text{memory}$
 - t3: $MBR \leftarrow (MBR) + 1$
 - t4: $\text{memory} \leftarrow (MBR)$ (store back the MBR to memory)
 - if $(MBR) == 0$ then $PC \leftarrow (PC) + 1$ (one micro-operation)

These Micro-operations can be done **during t4**

22

Instruction Cycle

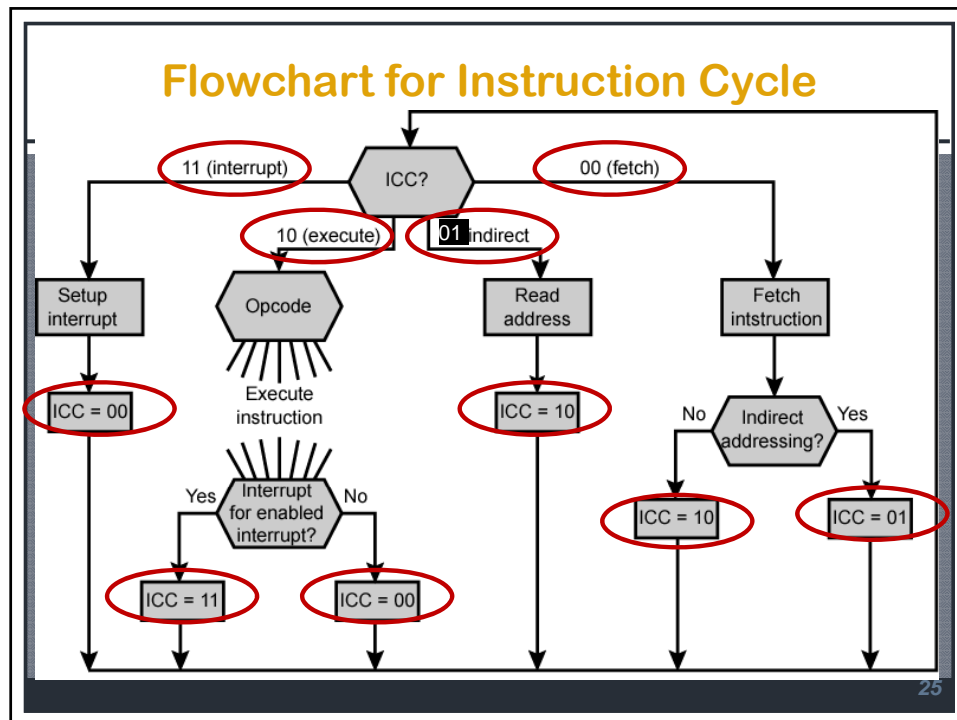
- **Fetch, indirect, and interrupt cycle**, each decomposed into a sequence of elementary micro-operations
- Execute cycle
 - **each opcode** consists of a sequence of micro-operations
- Need to **tie these sequences together**

23

Instruction Cycle


- Assume new 2-bit register, called **Instruction cycle code (ICC)**
 - designates which portion of cycle the processor is in
 - ✦ 00: Fetch
 - ✦ 01: Indirect
 - ✦ 10: Execute
 - ✦ 11: Interrupt

24



Functional Requirements

- With the decomposition, **the steps to characterize the control units:** those functions must perform
 - Define basic elements of processor
 - Describe micro-operations processor performs
 - Determine functions control unit must perform



26

Basic Elements of Processor

- **ALU**
 - functional essence of the computer
- **Registers**
 - Store data internal to the processor
- **Internal data paths**
 - Move data between registers and ALU
- **External data paths**
 - Link registers to memory and I/O modules
- **Control Unit**
 - Cause operations to happen within the processor

27

Types of Micro-operation

- All micro-operations involved in these operations fall into one of these categories:
 - *Transfer data between registers*
 - *Transfer data from register to external interface*
 - *Transfer data from external to register*
 - *Perform arithmetic or logical ops*

28

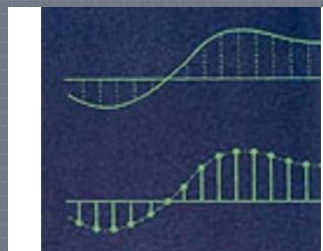
Functions of Control Unit

- The control unit performs *two basic tasks* :
- **Sequencing**
 - Causing the CPU to step through a series of micro-operations **in the proper sequence**, based on program execution
- **Execution**
 - Causing each micro-operation to be performed
- This is done using **Control Signals**

29

Control Signals

- It is the **engine** that runs the entire computer
- The control unit reads its input and emits **control signals** to cause micro-operations
- uses the **clock pulse** to stabilize the time sequence of events



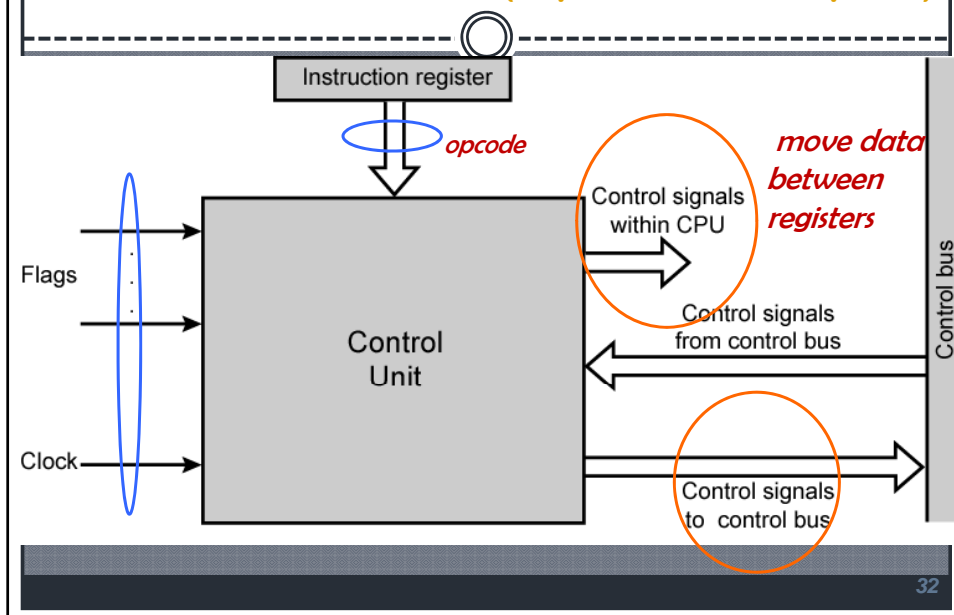
30

Control Signals

- For the control units
 - External specifications:
 - inputs that allow it to determine the state of the system
 - output that allow it to control the behaviours of the system
 - Internal logic
 - the control unit must have the logic to perform its sequencing and execution functions

31

Model of Control Unit (Inputs and Outputs)



32

Control Signals - Inputs

- **Clock: how the control unit keeps time**
 - One micro-instruction (or set of parallel micro-instructions) per clock cycle
- **Instruction register**
 - Op-code for current instruction, determines which micro-instructions are performed
- **Flags**
 - State of CPU
 - Results of previous ALU operations
- **Control signal from control bus**
 - Interrupts and acknowledgements

33

Control Signals - Output

- **Control signal within CPU**
 - Cause data movement between registers
 - Activate specific ALU functions
- **Control signal to control bus**
 - To memory
 - To I/O modules
- All of these signals are ultimately applied directly as **binary inputs** to logic gates

34

A Control Signals Example

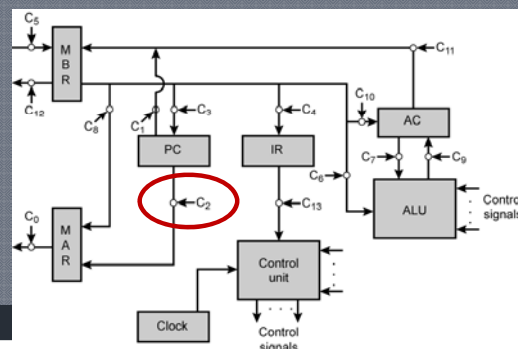
- A simple processor with a single accumulator
- The terminations of control signals are **label Ci**
- The control unit receives inputs from **the clock, the instruction register and flag**
- It emits a set of control signals, which go to three separate destinations:
 - Data path: control the internal flow of data through opening the gate
 - ALU: control the operation of the ALU
 - System bus: control signals onto the system bus

35

Example Control Signal Sequence

Fetch cycle: see how the control unit maintains the control

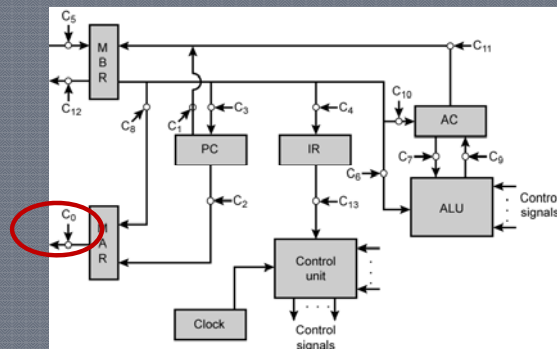
- First step: **MAR ← (PC)**
 - Control unit activates signal to **open gates between PC and MAR**



36

Example Control Signal Sequence

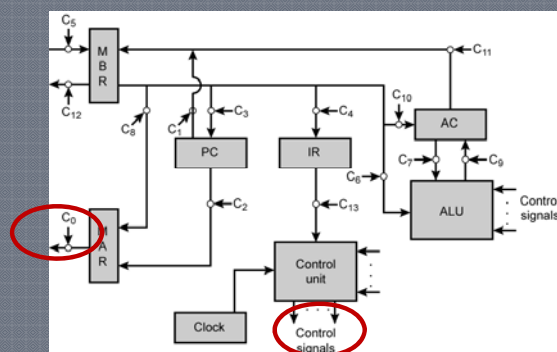
- Next step: **MBR - memory & increment the PC**
- Open gates allowing the content of MAR onto the address bus



37

Example Control Signal Sequence

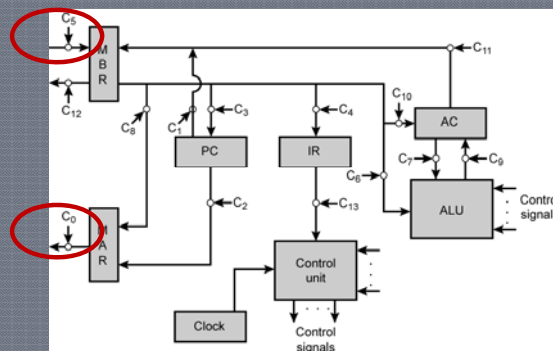
- Next step: **MBR - memory & increment the PC**
- Memory read control signal on the control bus



38

Example Control Signal Sequence

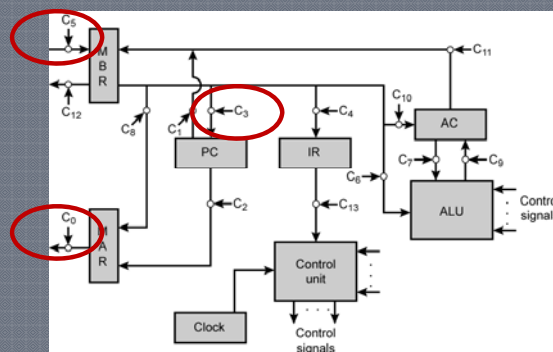
- Next step: **MBR - memory & increment the PC**
- Open gates allowing the content of the data bus to be stored in the MBR



39

Example Control Signal Sequence

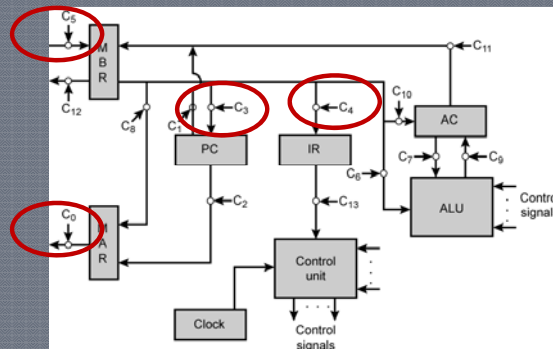
- Next step: **MBR - memory & increment the PC**
- Control signal to the logic incrementing the PC



40

Example Control Signal Sequence

- Next step: **MBR** - memory & increment the PC
- Following that, control unit *opens gates* between the MBR and IR

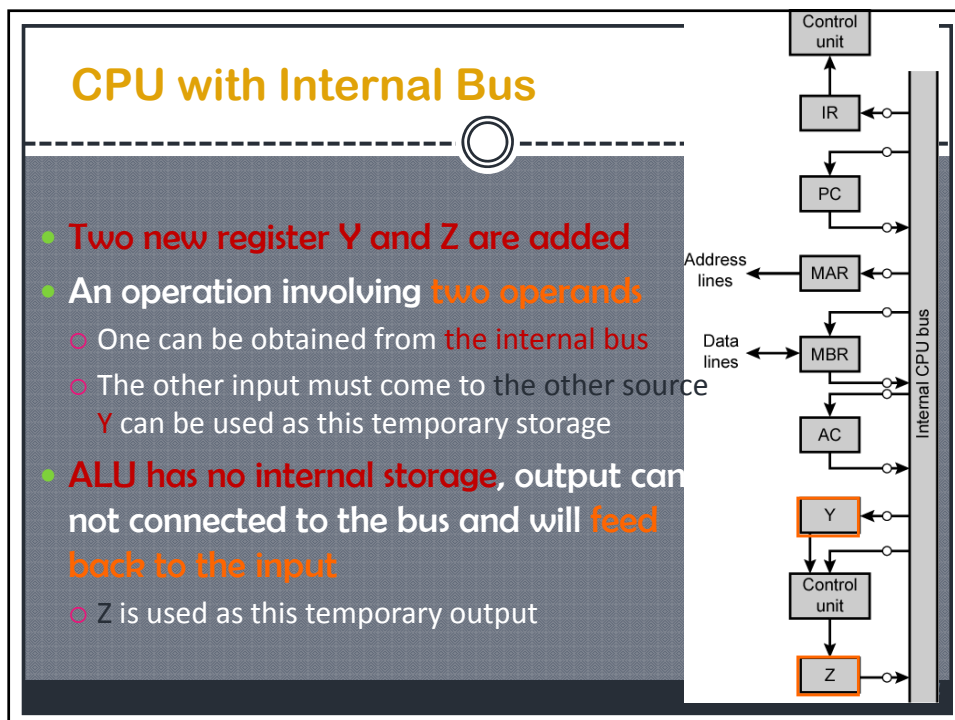
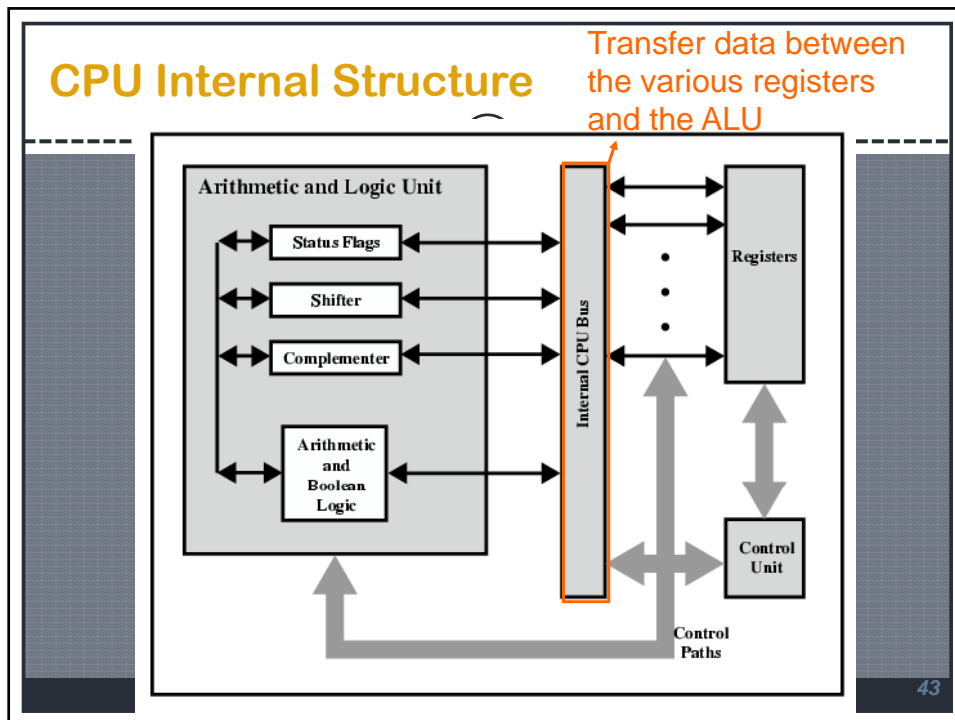


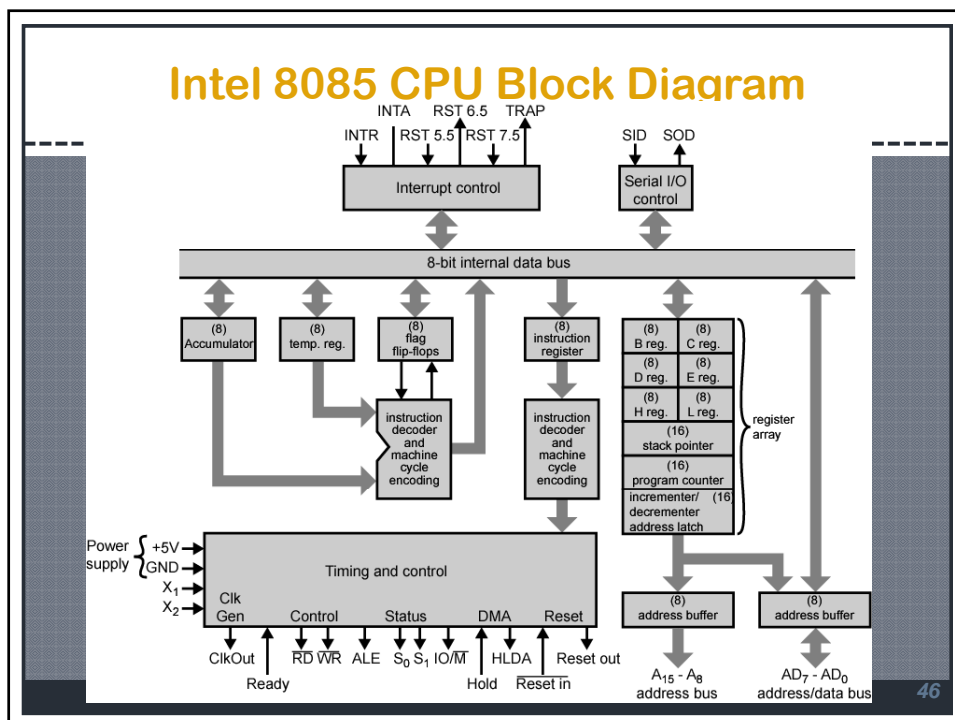
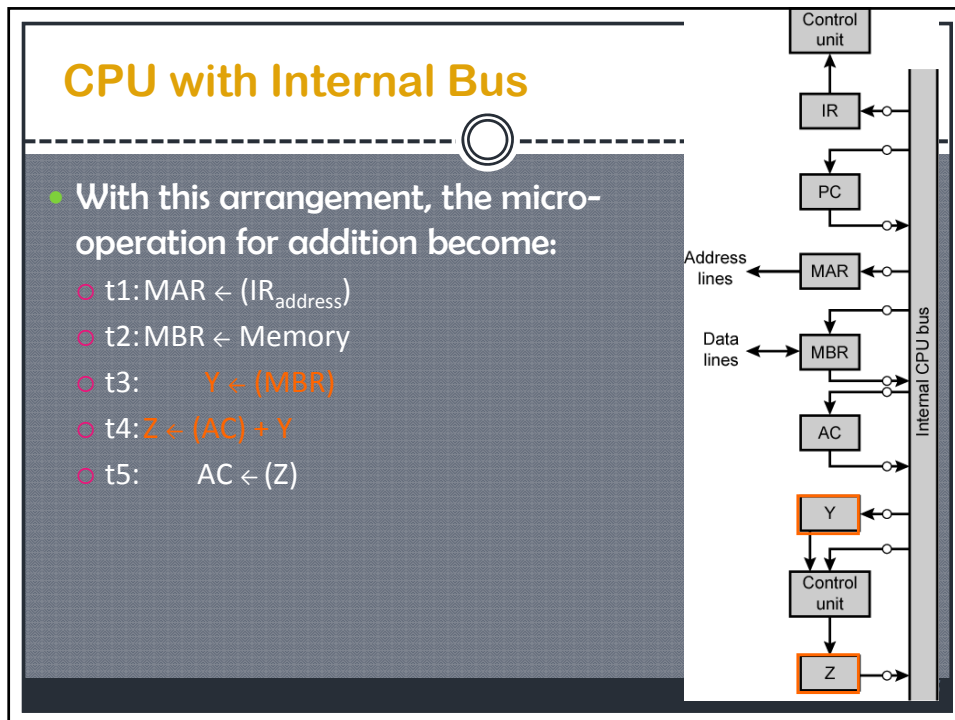
41

Internal Organization

- Usually, CPU uses a single **internal bus**
- **Gates** provided control the movement of data onto and off the bus
- **Control signals** control data transfer to and from external systems bus
- **Temporary registers** needed for proper operation of ALU

42





Hardwired Implementation

- Control unit implementation
- A wide variety of techniques have been used, and can be categorized:
 - Hardwired implementation
 - Microprogrammed implementation

47

Hardwired Implementation

- Control unit **inputs**:
 - Flags, **each bit has some meaning**, such as overflow
 - **Instruction register (key input)**
 - ✦ Op-code causes different control signals for each different instruction
 - ✦ For simplification, unique logic for each op-code
 - ✦ Decoder takes encoded input and produces single output
 - ✦ Has n binary inputs and 2^n outputs

48

Hardwired Implementation

o Clock

- ✦ Issue repetitive sequence of pulses
- ✦ Useful for measuring duration of micro-ops
- ✦ Must be **long enough** to allow signal propagation along data path
- ✦ Different control signals at different times within instruction cycle
- ✦ Need a counter with different control signals for t1, t2 etc.

49

Hardwired Implementation

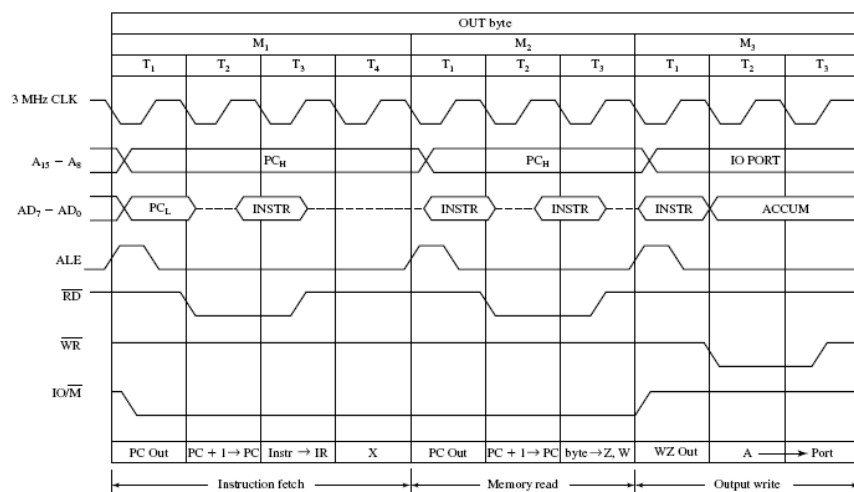
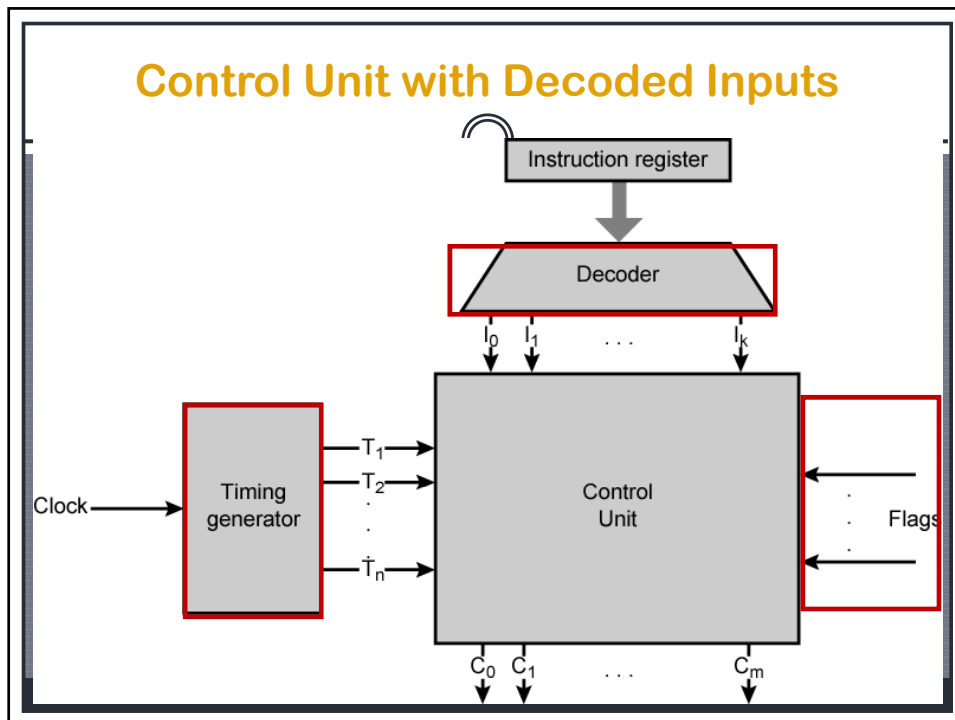


Figure 15.9 Timing Diagram for Intel 8085 OUT Instruction

50



Problems With Hard Wired Designs

- **Complex** sequencing & micro-operation logic
- **Difficult** to design and test
- **Inflexible** design
- **Difficult** to add new instructions

Homework

- Review Questions: 16.6
- Problems 16.2
- Hand in 27th May

53

In-Class Exercise

Describe the micro-operations of the instruction

- Mov **R1,X** ; move the contents of location X to Register 1 , result in R1
- Imp **X** ; *jump to the address stored in X*

54

In-Class Exercise

Describe the micro-operations of the instruction

- Mov **R1,X** ; move the contents of location X to Register 1 , result in R1
- t1: MAR \leftarrow (IR(address))
- t2: MBR \leftarrow (memory)
- t3: R1 \leftarrow (MBR)

55

In-Class Exercise

Describe the micro-operations of the instruction

- Jmp **X** ; *jump to the address stored in X*
- t1: MAR \leftarrow (IR(address))
- t2: MBR \leftarrow memory
- t3: PC \leftarrow (MBR)

56