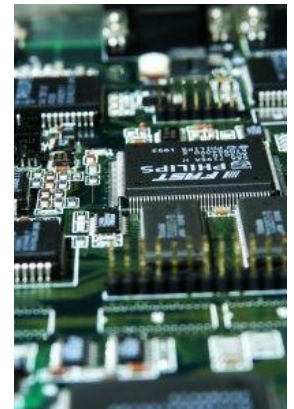# CSE 321a
Computer Organization (1)
تنظيم الحاسبات (1)

3rd year, Computer Engineering
Fall 2017
Lecture #2

Dr. Hazem Ibrahim Shehata
Dept. of Computer & Systems Engineering

Credits to Dr. Ahmed Abdul-Monem Ahmed for the slides

Administrivia

- Website:
  - http://hshehata.github.io/courses/zu/cse321a/
- Lecture Day/Time:
  - Tuesday 10:15am - 12:45am
- Office Hours:
  - TBA!
- Tutorials:
  - Started this week on Sunday at 2:55pm (just once!)
  - Regular day/time: TBA!
- Assignment #1:
  - To be released early next week.
  - Work in groups of two.

Ch. 3: A Top-Level View of Computer Function and Interconnection

Overview

- Computer Structure (Components)
- Computer Function
  - Instruction Fetch and Execute
  - Interrupts
- Interconnection Structures
- Bus Interconnection
  - Bus Structure
  - Multiple-Bus Hierarchies
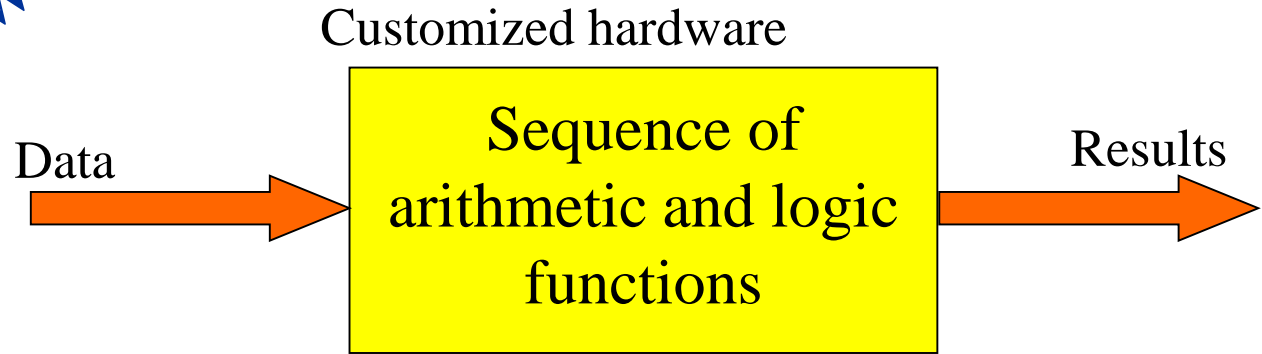 - Elements of Bus Design

Von Neumann (Princeton) Architecture

- Almost all contemporary computer designs are based on three basic concepts introduced by John von Neumann in 1945:

1. Data and instructions are stored in a single read-write memory.

2. The contents of this memory are addressable by location, without regard to the type of data contained there.

3. Execution occurs in a sequential fashion (unless modified) from one instruction to the next.
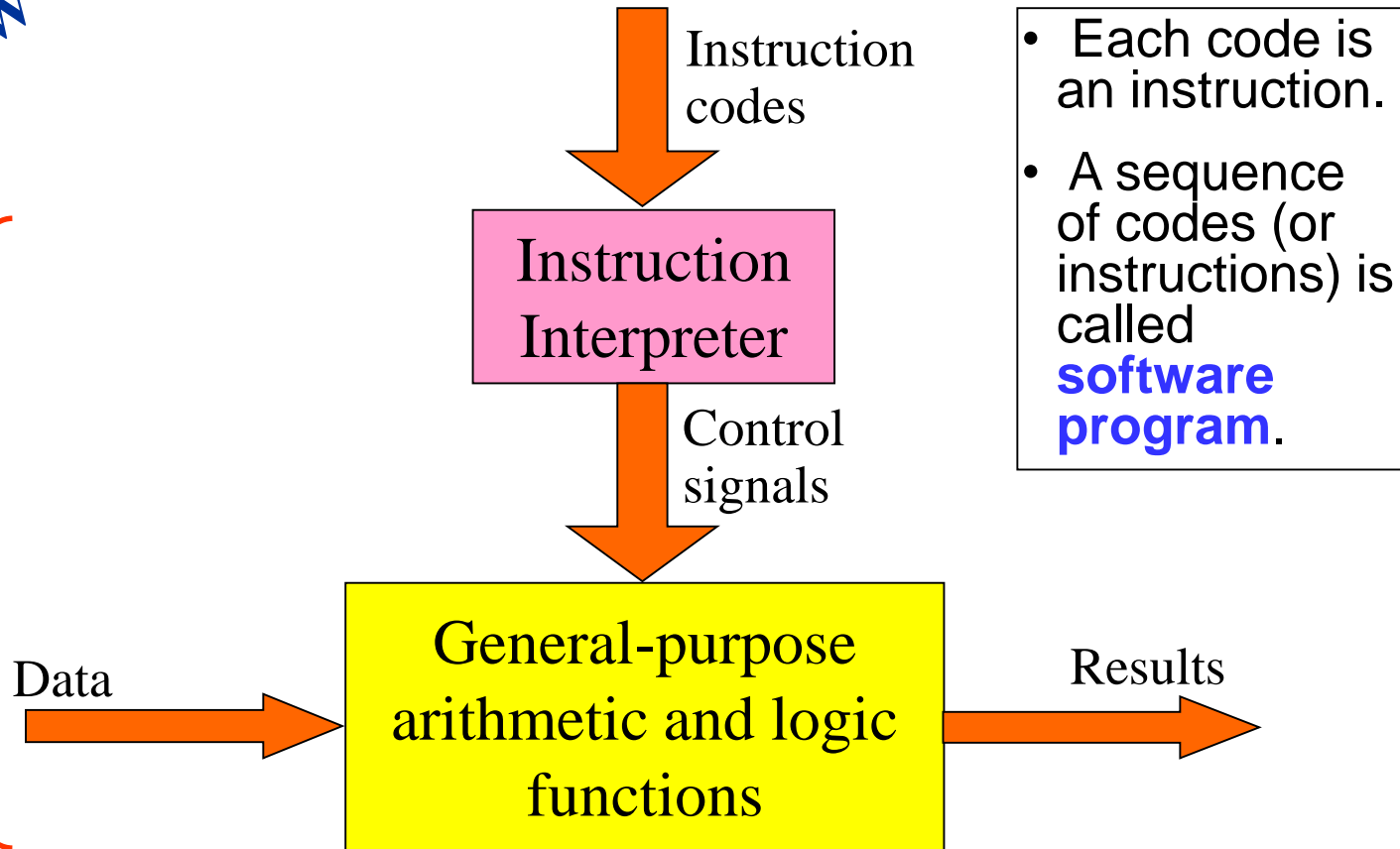
Program Concept

- Using a fixed set of logic components, there are two ways to build a system that performs a specific computation on some input data:

    1. Special-purpose (customized) hardware:
        – Logic components are connected and configured specifically to perform a particular computation.
        – Type of computation is specified by way in which components are connected/configured → hardwired programming.
        – To change computation, rewire components → inflexible!

    2. General-purpose hardware:
        – Logic components are connected to form a general-purpose hardware that gets configured using set of input control signals.
        – Type of computation (reflected by control signals values) is specified by sequence of input codes → software programming.
        – To change computation, change code → flexible!

Customized hardware

Data → **Sequence of arithmetic and logic functions** → Results

---

Instruction codes ↓

CPU {

Instruction Interpreter

Control signals ↓

Data → **General-purpose arithmetic and logic functions** → Results

- Each code is an instruction.

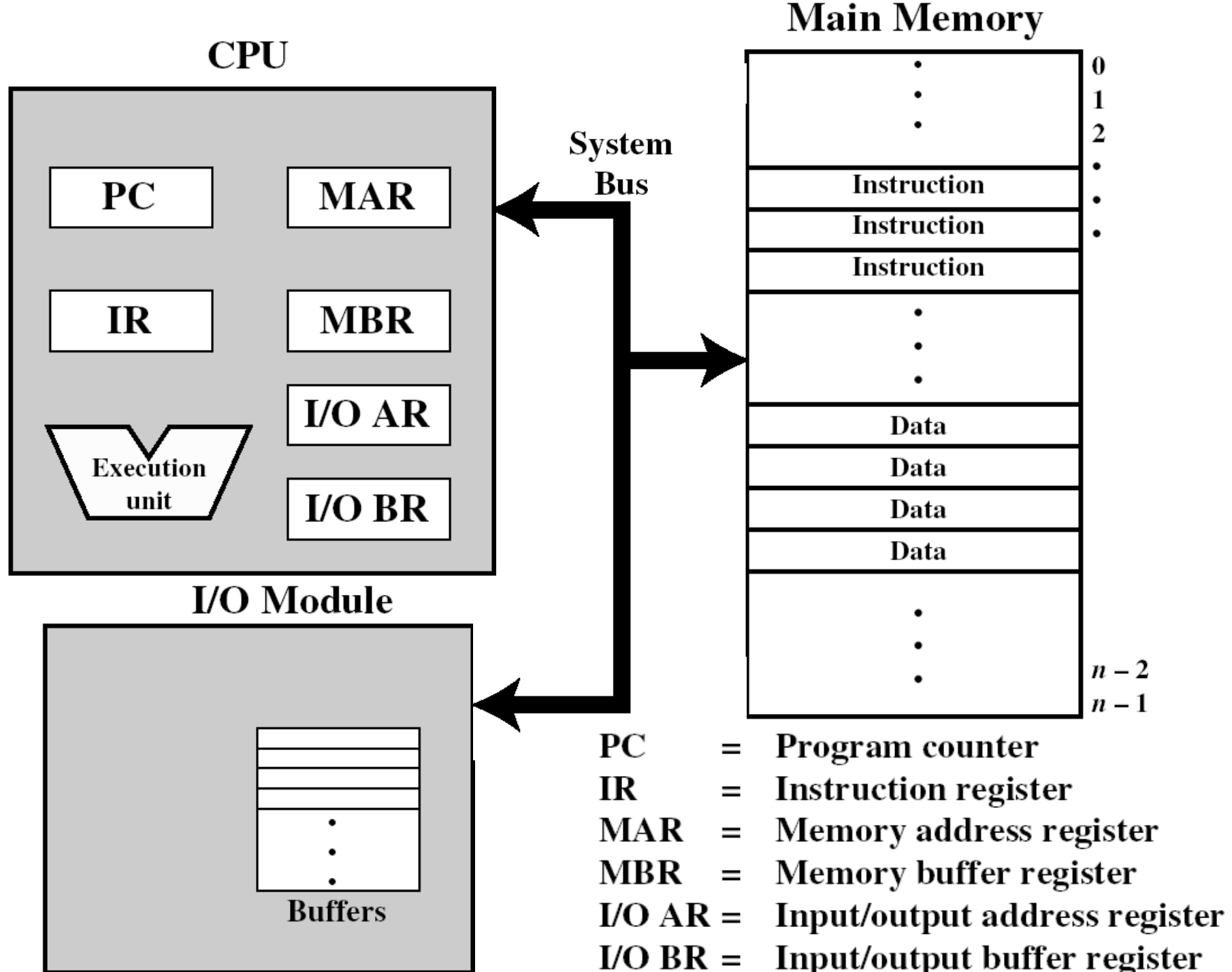- A sequence of codes (or instructions) is called **software program**.

What is a program?

- A sequence of steps.

- For each step, an arithmetic or logical operation could be done.

- For each operation, a different set of control signals are generated.
  - Each operation is assigned a unique code.
    - e.g., ADD ➔ 9D, MOVE ➔ E3, … etc.
  - A hardware segment accepts the code and issues the control signals.

- That's it. We have a computer!

Computer Components

- Each computer component takes some part in implementing the "s/w programming" approach:
  - —Control unit (CU) acts as instruction interpreter.
  - —Arithmetic and logic unit (ALU) implements general-purpose arithmetic and logic functions.
  - —Both CU and ALU constitute most of central processing unit (CPU).
  - —Data/code get into system, and results get out through input/output (I/O) units.
  - —Data, code, and results are stored temporarily while being processed in main memory (MM) unit.

CPU

PC	MAR
IR	MBR
Execution unit	I/O AR
	I/O BR

System Bus

Main Memory

	0
	1
	2
Instruction	
Instruction	
Instruction	
Data	
Data	
Data	
Data	
	$n-2$
	$n-1$

I/O Module

Buffers

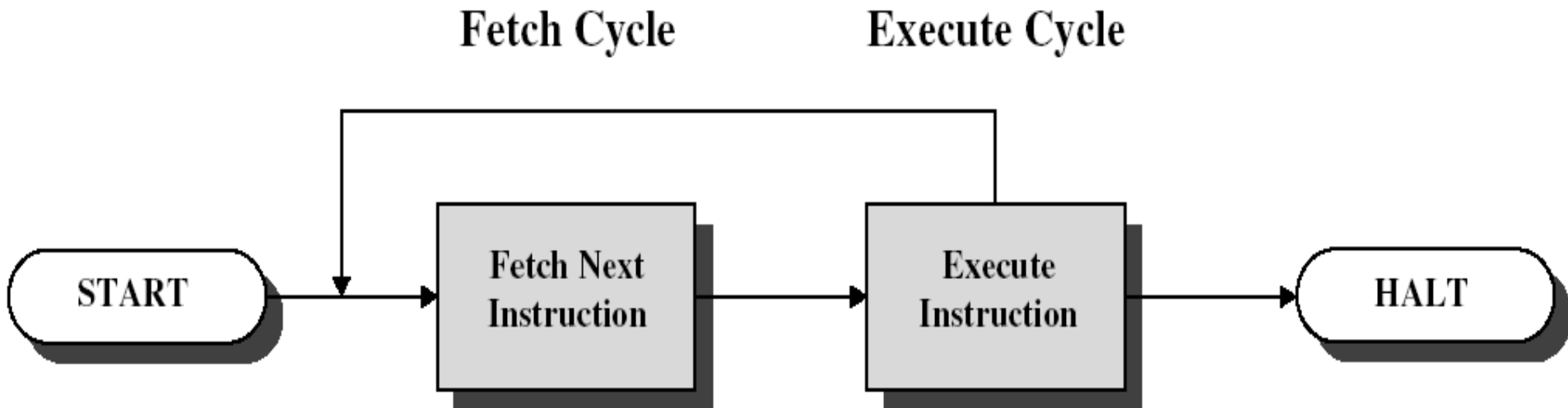| PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

Overview

- Computer Structure (Components)
- Computer Function
  - Instruction Fetch and Execute
  - Interrupts
- Interconnection Structures
- Bus Interconnection
  - Bus Structure
  - Multiple-Bus Hierarchies
  - Elements of Bus Design

Instruction Cycle

- Two steps:
  - —Fetch: CPU reads instructions from memory, one at a time.
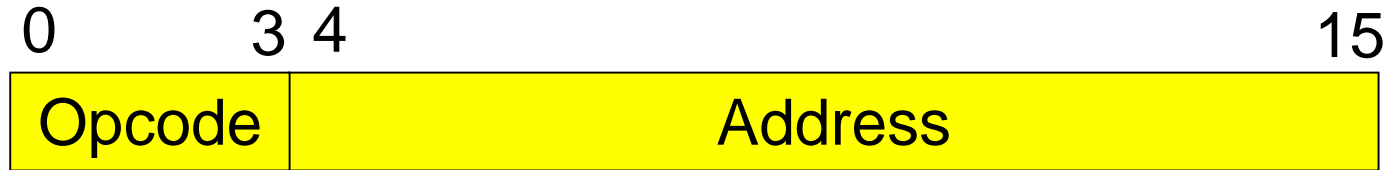 - —Execute: CPU executes instructions.

Fetch Cycle   Execute Cycle

START → Fetch Next Instruction → Execute Instruction → HALT

Fetch Cycle

- Program Counter (PC) holds address of next instruction to fetch.

- Processor fetches instruction from memory location pointed to by PC.

- Increment PC.
  - Unless told otherwise

- Instruction loaded into Instruction Register (IR).

- Processor interprets instruction and performs required actions in execute cycle.

Execute Cycle

- Processor-memory
  - Data transfer between CPU and main memory
- Processor-I/O
  - Data transfer between CPU and I/O module
- Data processing
  - Some arithmetic or logical operation on data
- Control
  - Alteration of sequence of operations
  - e.g., jump
- Combination of the above

Hypothetical Machine

0	3 4		15
Opcode		Address	

Instruction Format

0	1		15
S		Magnitude	

Integer Format

Program Counter (PC)

Instruction Register (IR)

Accumulator (AC)

0001 = Load AC from memory.

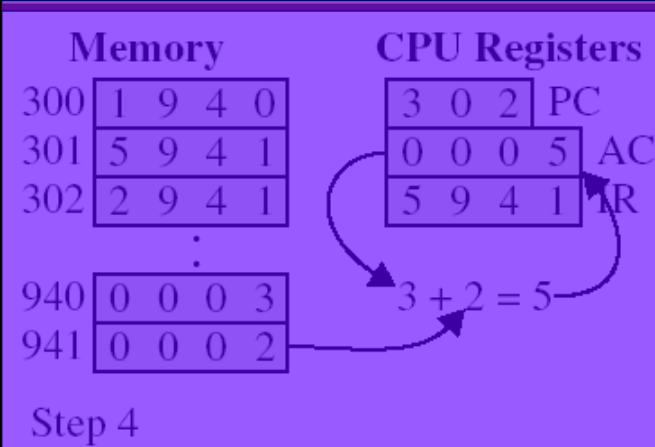**0010 = Store AC to memory.**
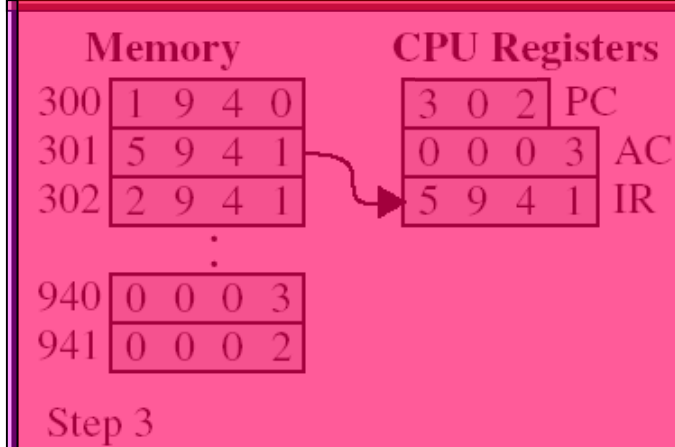
0101 = Add to AC from memory.

CPU Registers

Partial List of Opcodes

Fetch cycle **Execute cycle**

Step 1

Memory
300	1	9	4	0
301	5	9	4	1
302	2	9	4	1
.
940	0	0	0	3
941	0	0	0	2

CPU Registers
3	0	1		PC
				AC
				IR

Step 2

Memory
300	1	9	4	0
301	5	9	4	1
302	2	9	4	1
.
940	0	0	0	3
941	0	0	0	2

CPU Registers
3	0	1		PC
0	0	0	3	AC
1	9	4	0	IR

First instruction

Step 3

Memory
300	1	9	4	0
301	5	9	4	1
302	2	9	4	1
.
940	0	0	0	3
941	0	0	0	2

CPU Registers
3	0	2		PC
0	0	0	3	AC
5	9	4	1	IR

Step 4

Memory
300	1	9	4	0
301	5	9	4	1
302	2	9	4	1
.
940	0	0	0	3
941	0	0	0	2

CPU Registers
3	0	2		PC
0	0	0	5	AC
5	9	4	1	IR

$3 + 2 = 5$

Second instruction

Step 5

Memory
300	1	9	4	0
301	5	9	4	1
302	2	9	4	1
.
940	0	0	0	3
941	0	0	0	2

CPU Registers
3	0	3		PC
0	0	0	5	AC
2	9	4	1	IR

Step 6

Memory
300	1	9	4	0
301	5	9	4	1
302	2	9	4	1
.
940	0	0	0	3
941	0	0	0	5

CPU Registers
3	0	3		PC
0	0	0	5	AC
2	9	4	1	IR

Third instruction

Instruction Cycle: State Diagram

CPU-memory or CPU-I/O operations
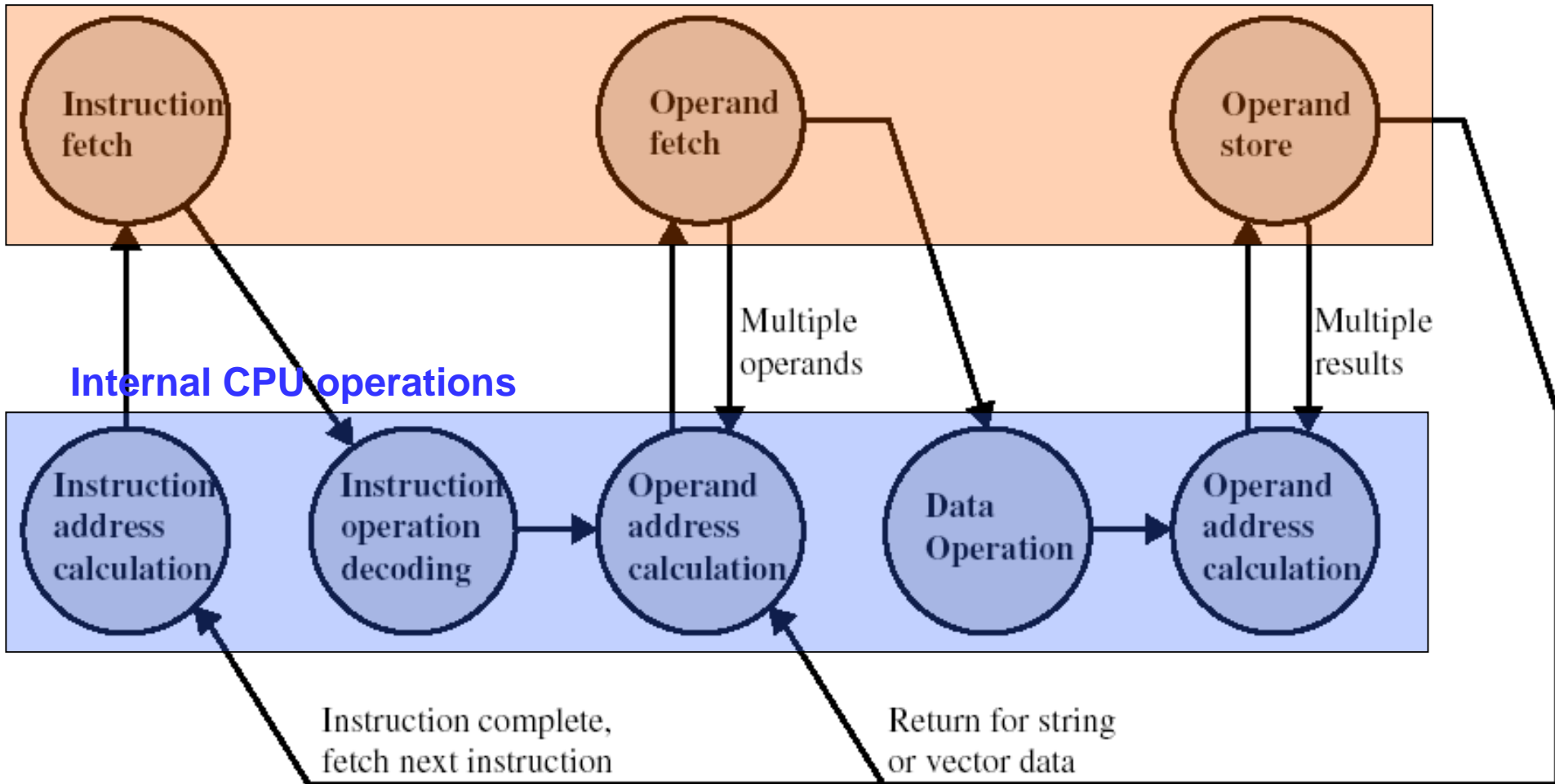
**Internal CPU operations**

Overview

- Computer Structure (Components)
- Computer Function
  - Instruction Fetch and Execute
  - Interrupts
- Interconnection Structures
- Bus Interconnection
  - Bus Structure
 - Multiple-Bus Hierarchies
  - Elements of Bus Design

Interrupts

- Interrupts are provided to improve processing efficiency.

- Mechanism by which other modules (e.g. I/O) may interrupt normal sequence of processing

- Most common classes of interrupts:
  - Program
    - e.g. overflow, division by zero
  - Timer
    - Generated by internal processor timer
    - Used in pre-emptive multi-tasking (i.e., CPU time sharing).
  - I/O
    - From I/O controller
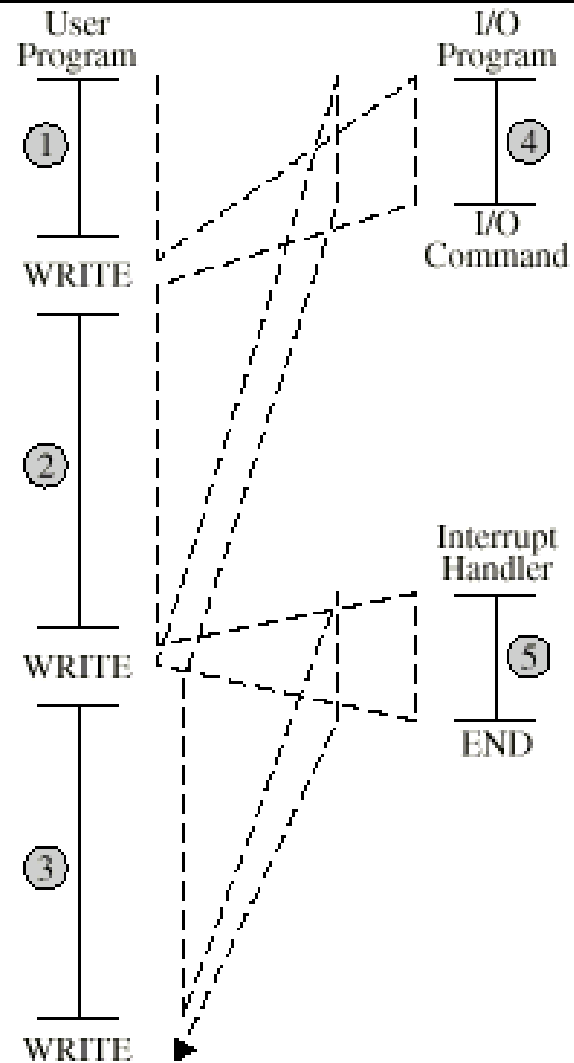 - Hardware failure
    - e.g. memory parity error

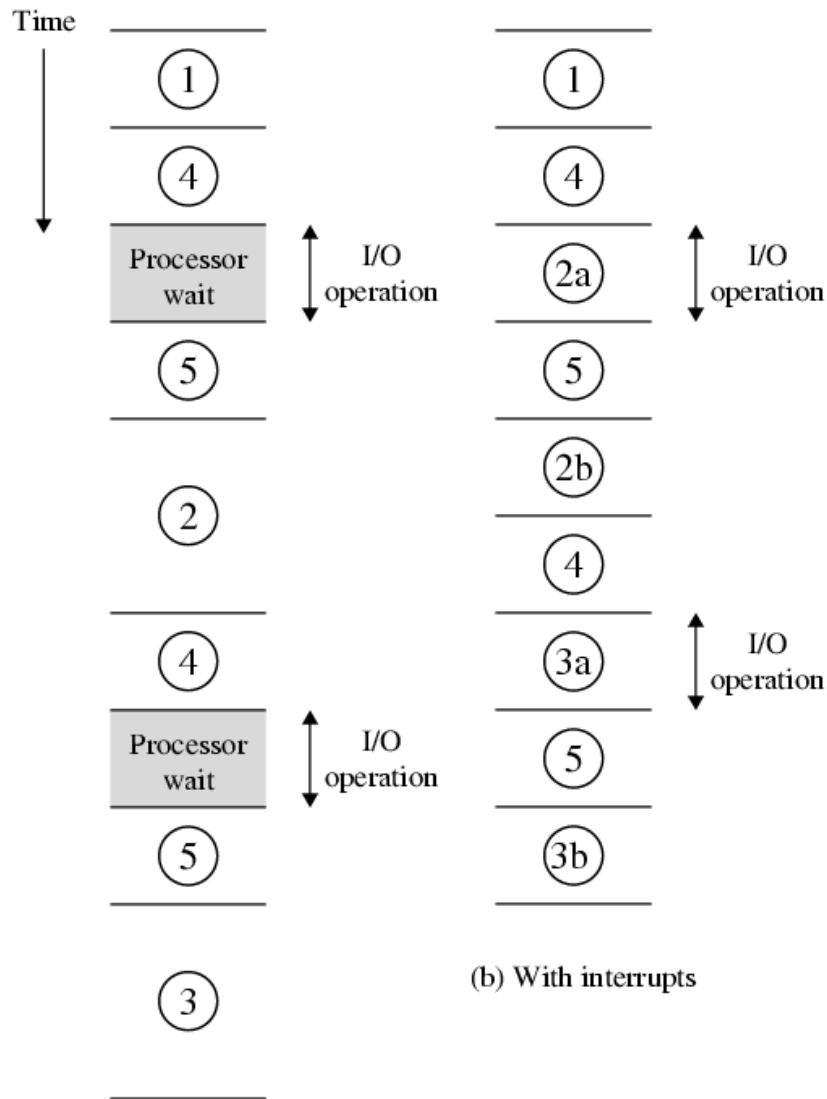# Transfer of Control via Interrupts

User Program

Interrupt Handler

1
2

i
Interrupt occurs here
$i + 1$

M
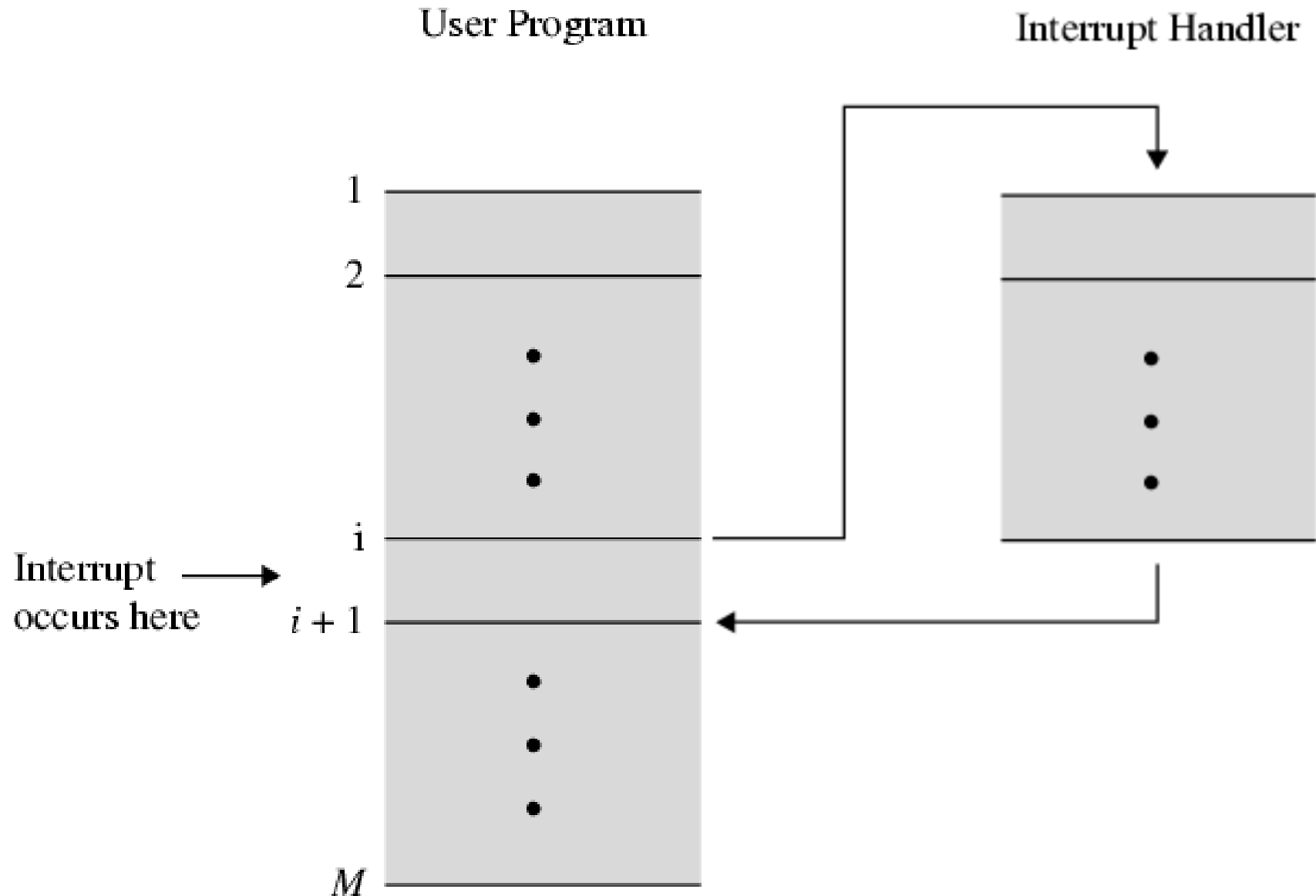
Program Flow Control
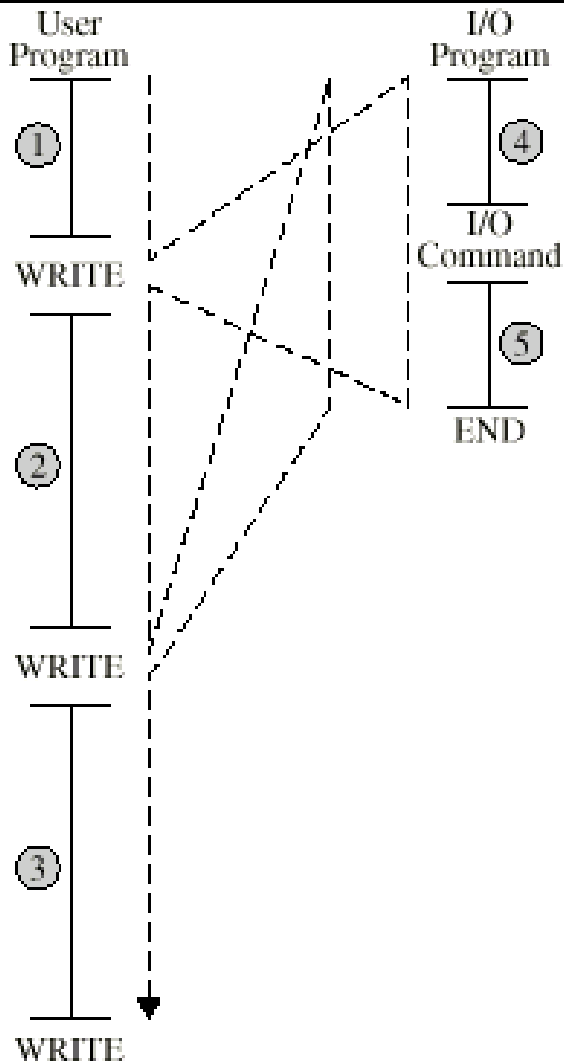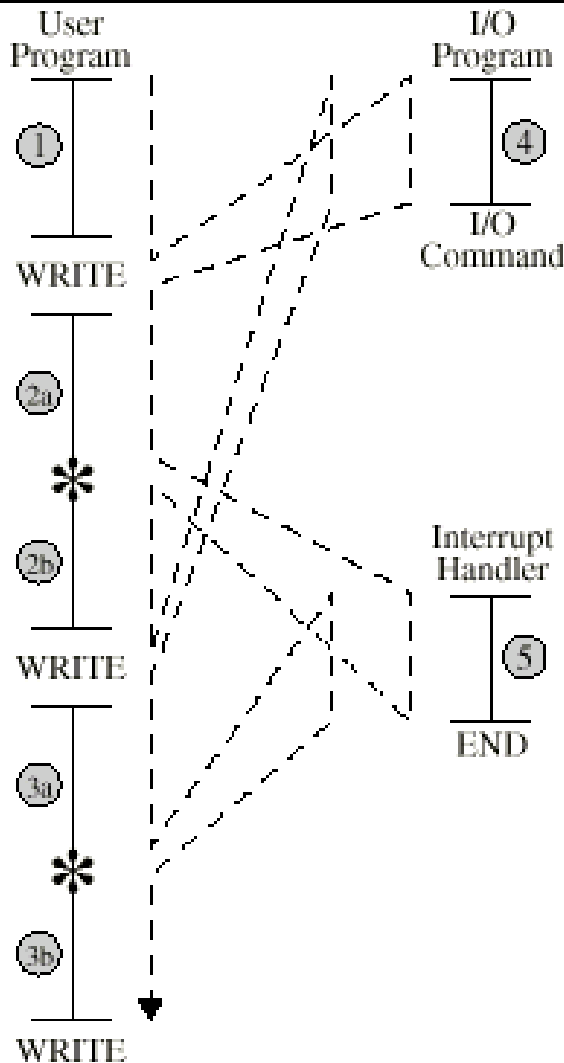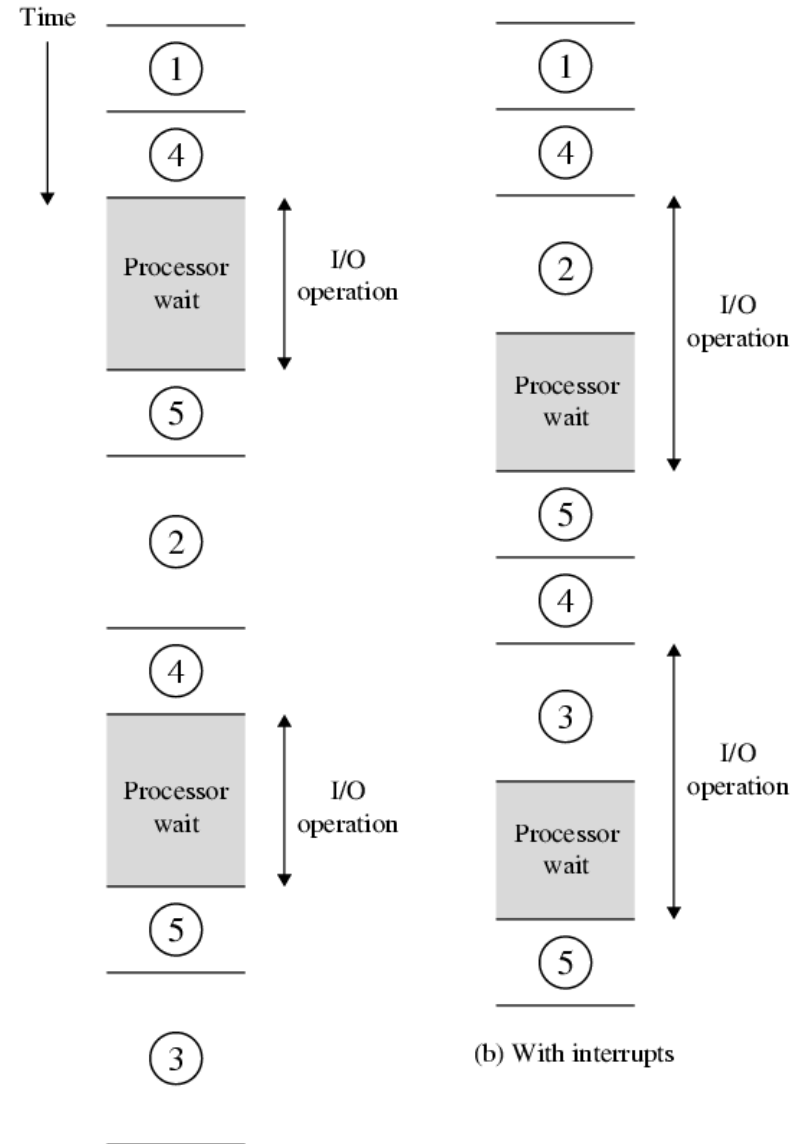

(a) No interrupts   (b) Interrupts; short I/O wait   (c) Interrupts; long I/O wait

Program Timing

(a) Without interrupts

(b) With interrupts

Interrupt Cycle

- Added to instruction cycle.
- Processor checks for interrupt
  - Indicated by an interrupt signal
- If no interrupt:
  - fetch next instruction
- If interrupt pending:
  - Suspend execution of current program.
  - Save context (PC & relevant data).
  - Set PC to start address of interrupt handler routine.
  - Process interrupt.
  - Restore context and continue interrupted program.

Instruction Cycle (with interrupts)
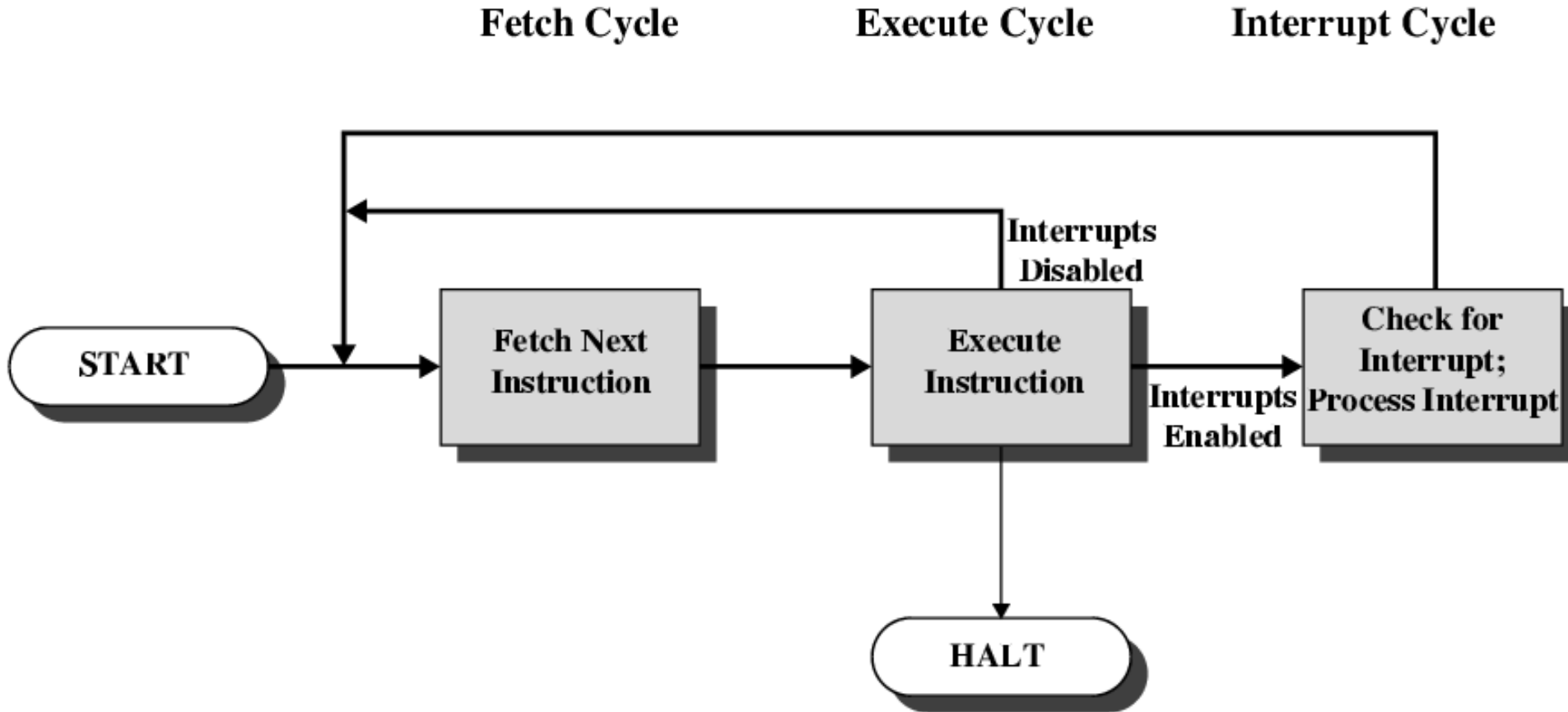
Instruction Cycle: State Diagram (with interrupts)
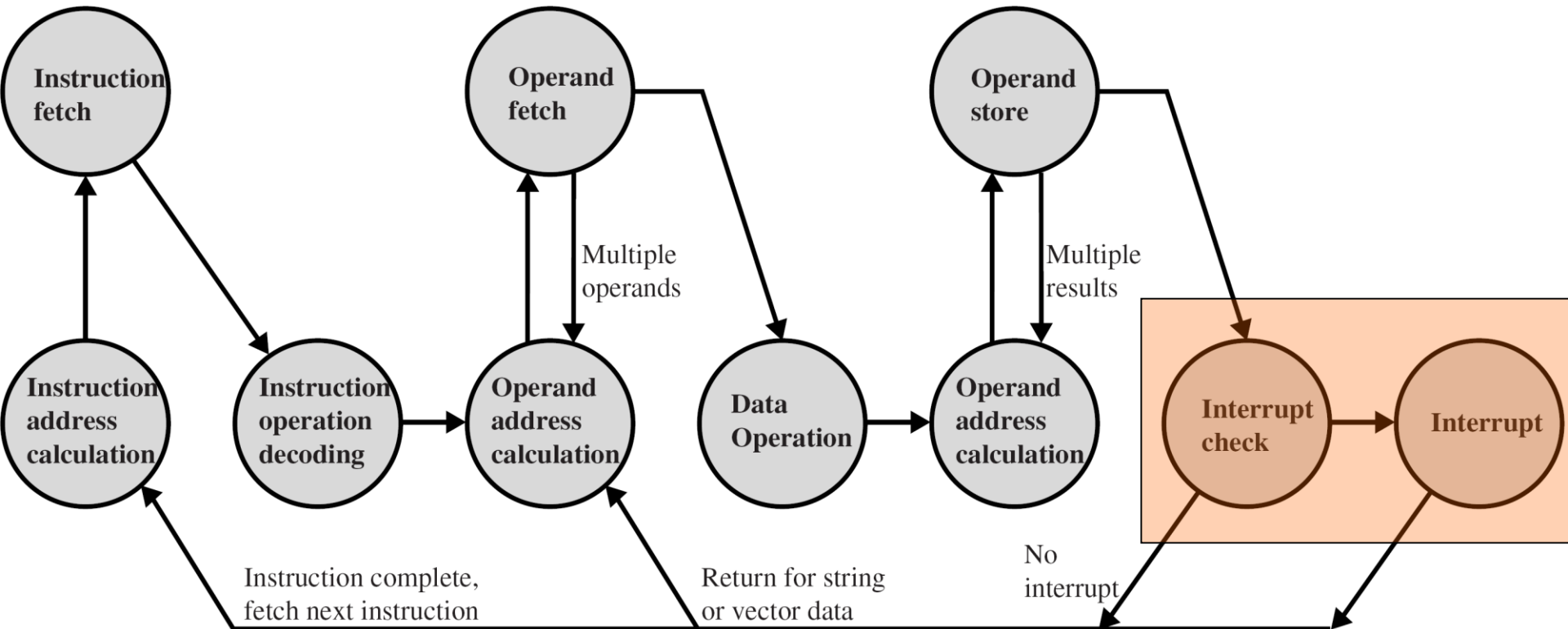
Reading Material

- Stallings, Chapter 3:
  - Pages 66 – 80