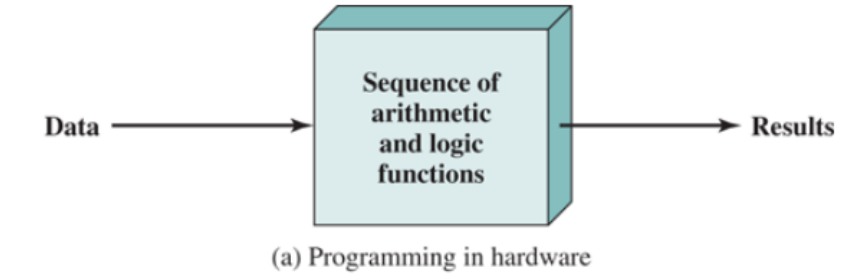
University of San Carlos | Department of
COMPUTER ENGINEERING

CpE 3202
Computer Organization & Architecture

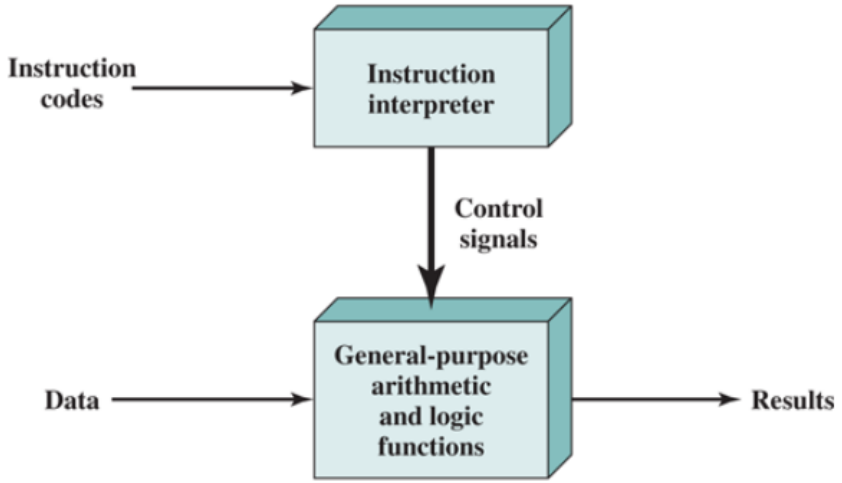
The Control Unit

Program Concept

- Hardwired systems are inflexible
- General purpose hardware can do different tasks, given correct control signals
- Instead of re-wiring, supply a new set of control signals



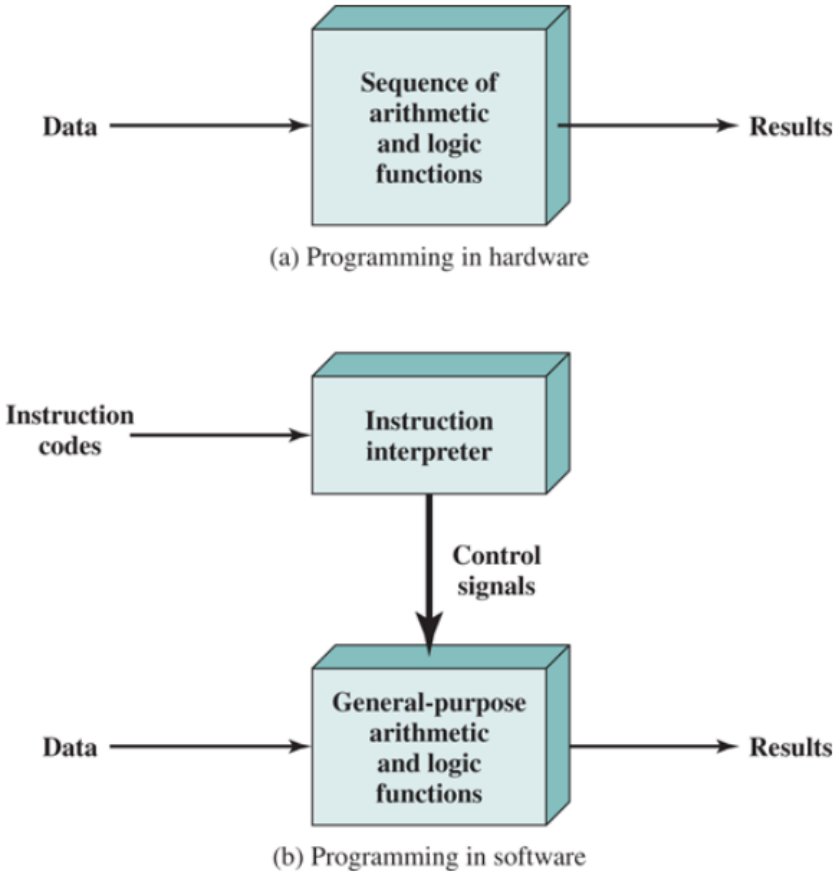
(a) Programming in hardware



(b) Programming in software

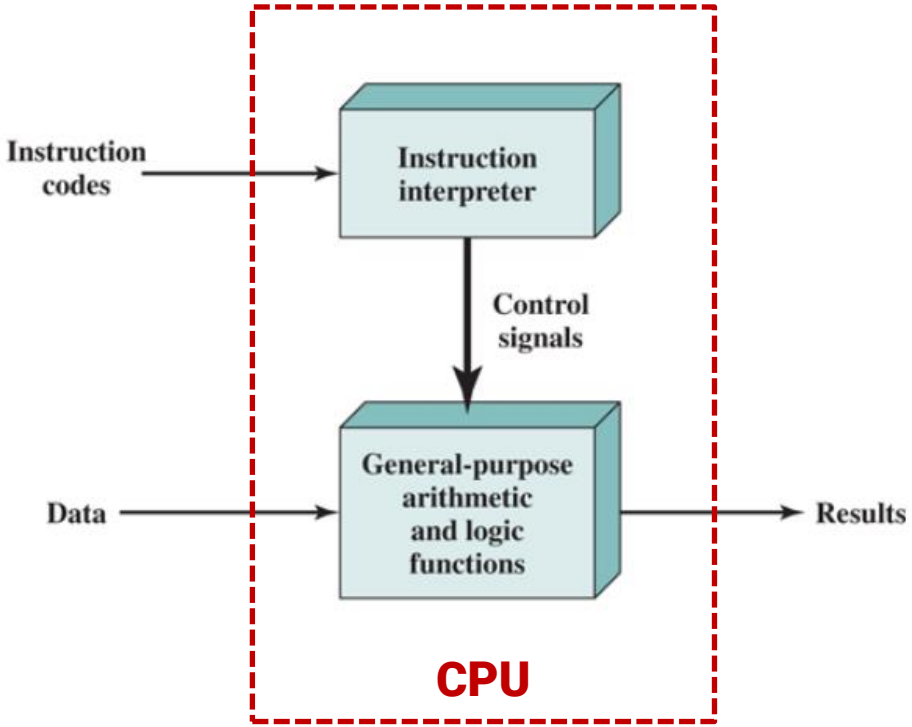
What is a Program?

- A sequence of steps
- For each step, an arithmetic or logical operation is done
- For each operation, a different set of control signals is needed



Function of Control Unit

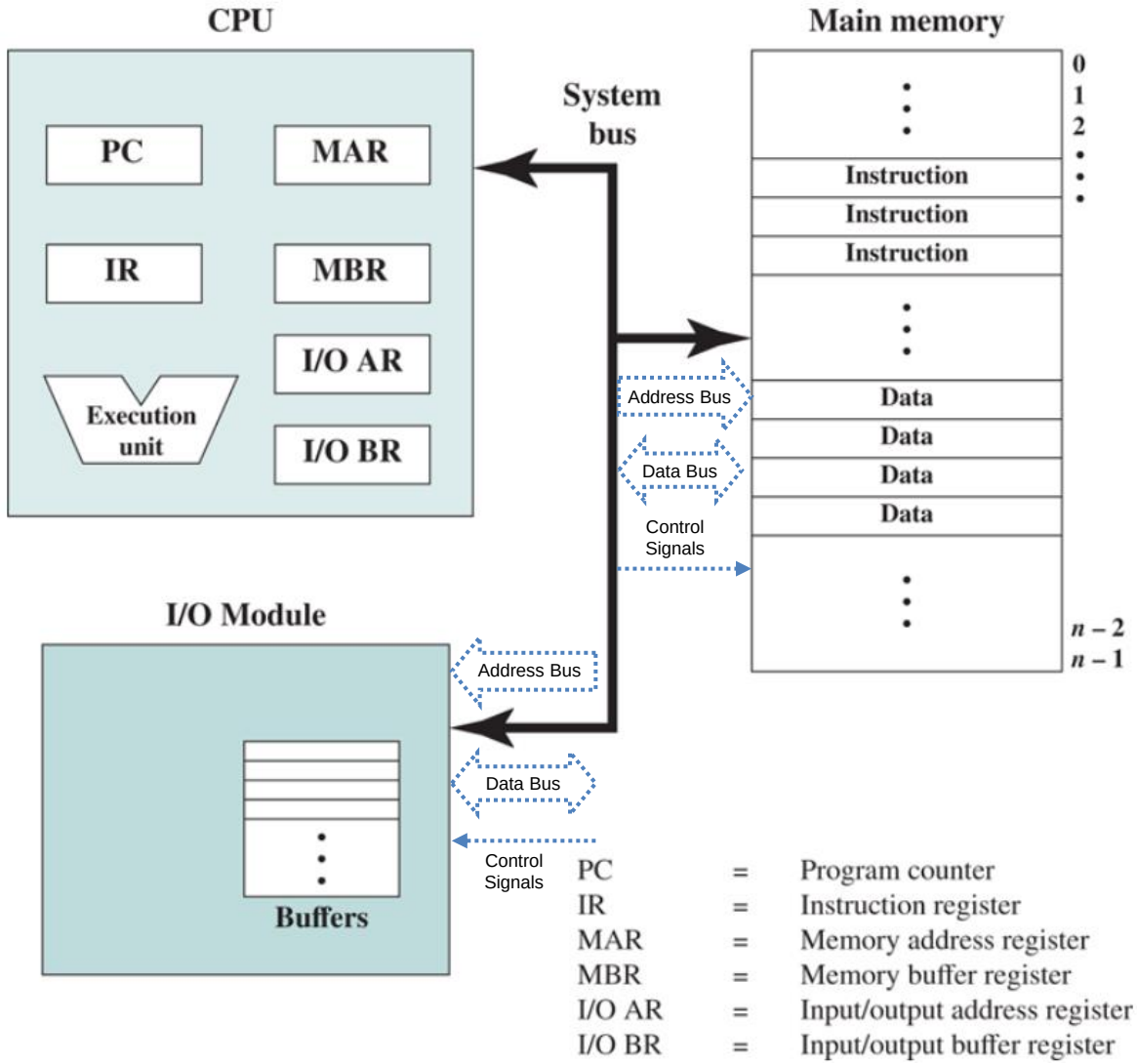
- For each operation, a **unique code** is provided
  - e.g. ADD, MOVE
- A hardware segment **accepts the code** and **issues the control signals**
- We have a computer!



Components

- The **Control Unit (CU)** and the **Arithmetic and Logic Unit (ALU)** constitute the **Central Processing Unit (CPU)**
- **Data** and **instructions** need to get into the system and results out
  - Input/output
- **Temporary storage of code and results** is needed
  - Main memory
  - For von Neumann architectures, same memory for both instructions and data

Computer Components: Top Level View



Buses

- **Address Bus** - *n*-bit parallel wires which connects to the address lines of data and I/O memory. It carries the address bits and is unidirectional (*read only*).

- **Data Bus** - 2^n-bit parallel wires which connects to the data lines of data and I/O memory. It carries the data bits and is bi-directional (*read/write*).

- **Control Signals** - *n*-bit signals which carries **control signals** from the control unit.

Registers (Address Pointers)

- Program Counter (PC) - contains a memory address that points to the next instruction to be fetched.
- Instruction Register (IR) - holds the instruction that is fetched.
- Memory Address Register (MAR) - contains a memory address that points to the data to be read/written from/to the data memory.
- I/O Address Register (IOAR) - contains a memory address that points to the data to be read/written from/to the I/O memory.

Registers (Data)

- Memory Buffer Register (MBR) - contains the data that is read or to be written from/to the data memory.

- I/O Buffer Register (IOAR) - contains the data that is read or to be written from/to the I/O memory.

Control Signals (External)

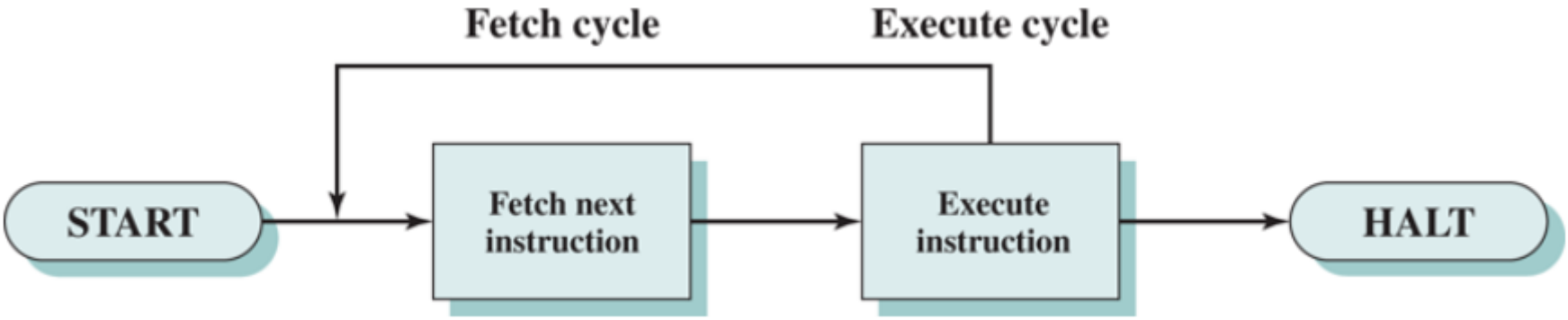
- $\overline{IOM}$ - a 1-bit signal that indicates which memory will be accessed data or I/O memory and is connected to both data and I/O memory.
- $\overline{RW}$ - a 1-bit signal that tells the data or I/O memory what operation to perform, read or write and is connected to both data and I/O memory.
- OE - enables the output of the data or I/O memory.
- Other signals which will be generated depending on the instruction to control other components like the bus controller, DMA controller, etc.

Control Signals (Internal)

- The control unit might generate also **“internal signals”** within the CPU.
  - These signals will be connected to the ALU and registers.
  - Example: **Instruction Code (Operation Code)**
    - An *n*-bit word
    - Tells the ALU and control unit which instruction to perform
    - This code is extracted from the instruction during the decode process.

Basic Instruction Cycle

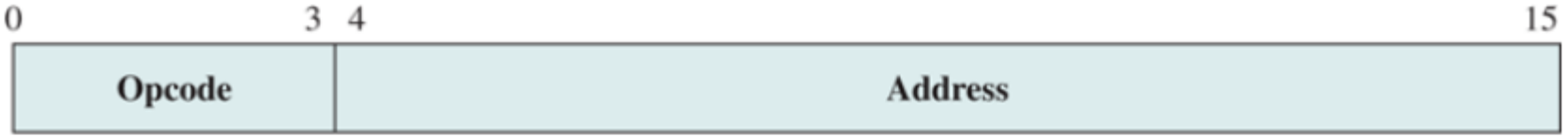
- Two steps:
  - Fetch
  - Execute



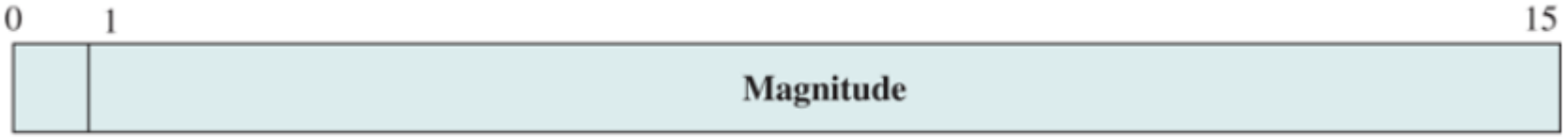
Fetch Cycle

- **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
- Fetched instruction is loaded into **Instruction Register (IR)**
- Processor interprets instruction and performs required actions

A Hypothetical Example



(a) Instruction format



(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

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 instruction

Data Flow (1)

- When the control unit **fetches the next instruction**:
  - ADDR* <= PC
  - IR <= data from memory
  - Control signals: $\overline{IOM}$ = 1, $\overline{RW}$ = 0, OE = 1

- When an instruction needs to perform a **branch operation**:
  - PC <= instruction address *(instruction operand)*
  - Control signals: $\overline{IOM}$ = x, $\overline{RW}$ = x, OE = x

- When an instruction needs to perform a **register-register operation**:
  - REG1 <= REG2 (Example: ACC <= MBR)
  - Control signals: $\overline{IOM}$ = x, $\overline{RW}$ = x, OE = x

* ADDR stands for the address bus

Data Flow (2)

- When an instruction **reads data from data memory**:
  - MAR <= memory address
  - BUS* <= data from memory
  - Control signals: $\overline{IOM}$ = 1, $\overline{RW}$ = 0, OE = 1

- When an instruction **writes data to data memory**:
  - MAR <= memory address
  - Data memory [address] <= BUS*
  - Control signals: $\overline{IOM}$ = 1, $\overline{RW}$ = 1, OE = 0

* BUS stands for the data bus

Data Flow (3)

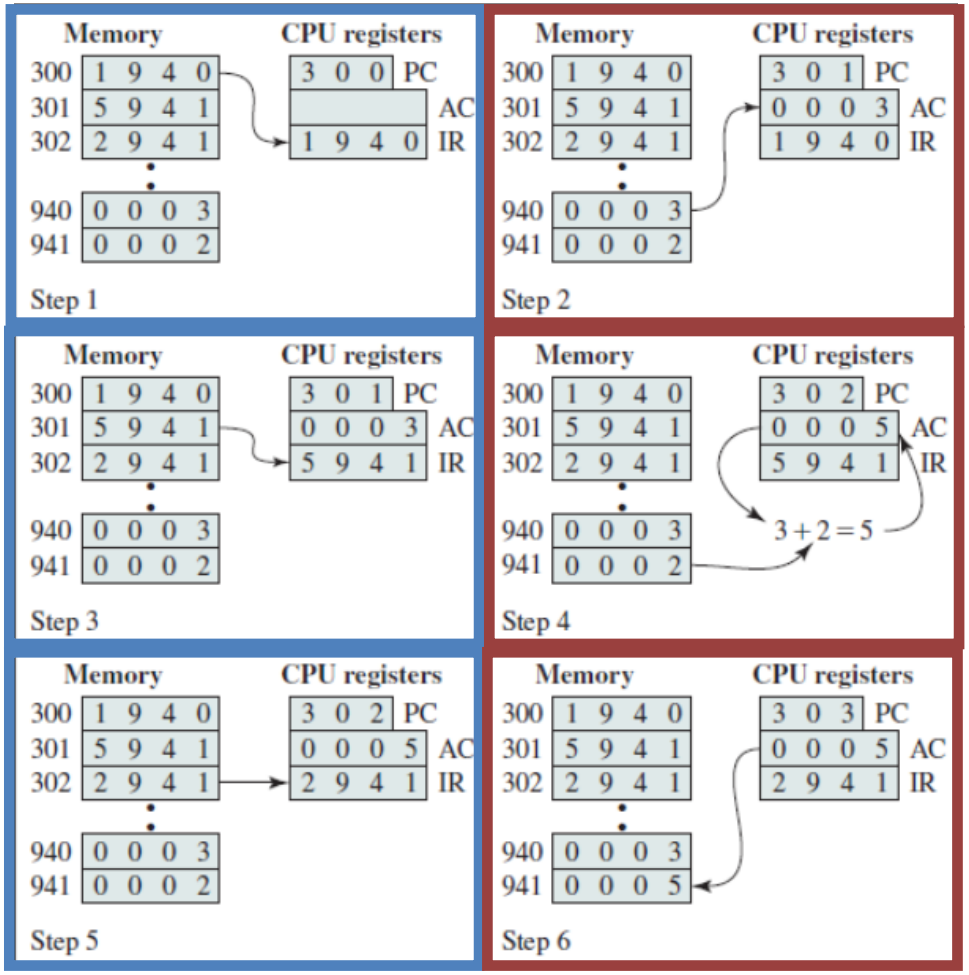
- When an instruction **reads data from I/O memory**:
  - IOAR <= memory address
  - BUS* <= data from memory
  - Control signals: $\overline{IOM}$ = 0, $\overline{RW}$ = 0, OE = 1

- When an instruction **writes data to I/O memory**:
  - IOAR <= memory address
  - I/O memory [address] <= BUS*
  - Control signals: $\overline{IOM}$ = 0, $\overline{RW}$ = 1, OE = 0

* BUS stands for the data bus

Example of Program Execution

Fetch

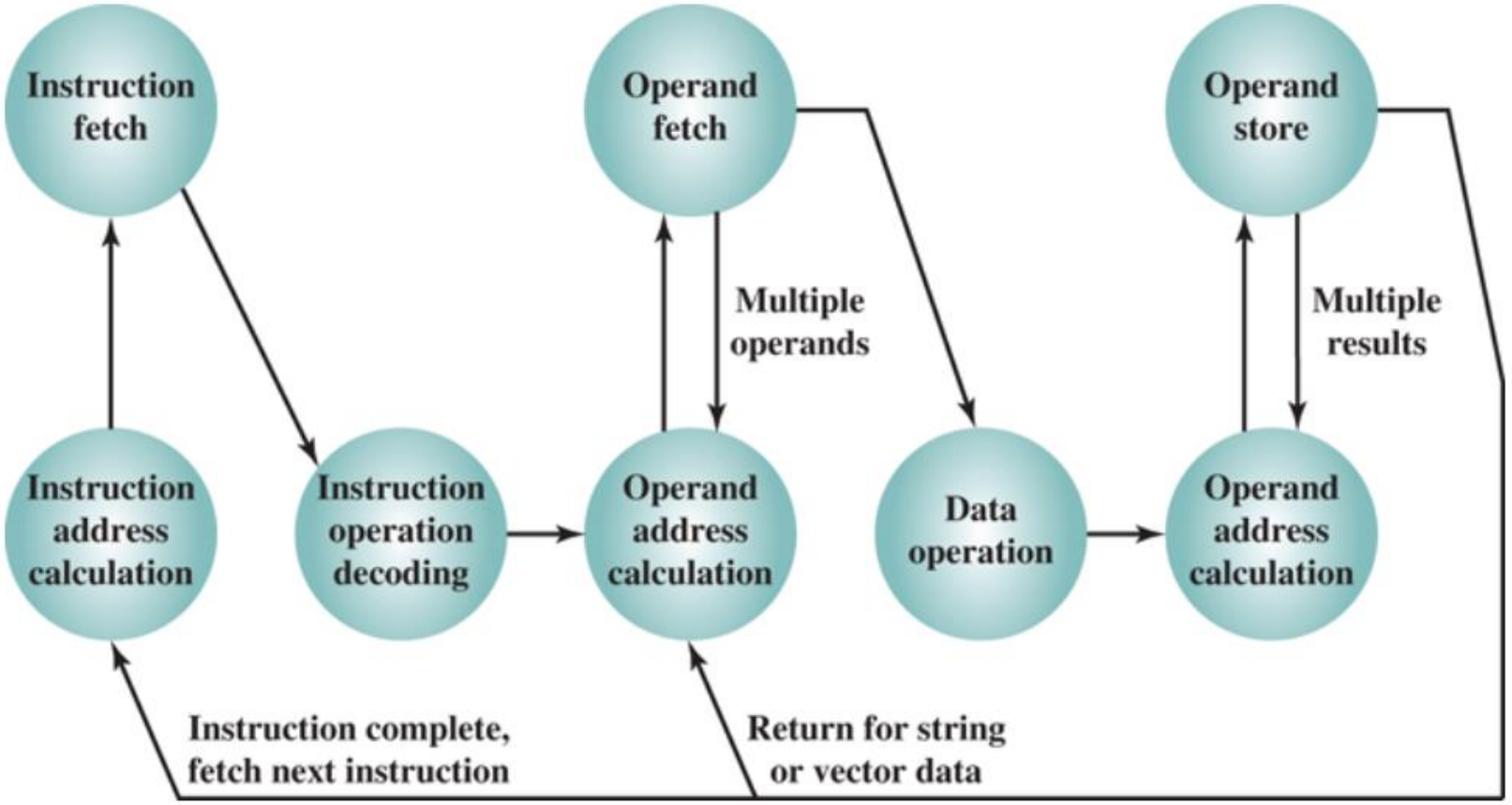


Decode-Execute

PC++

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

Instruction Cycle – State Diagram



Interrupts

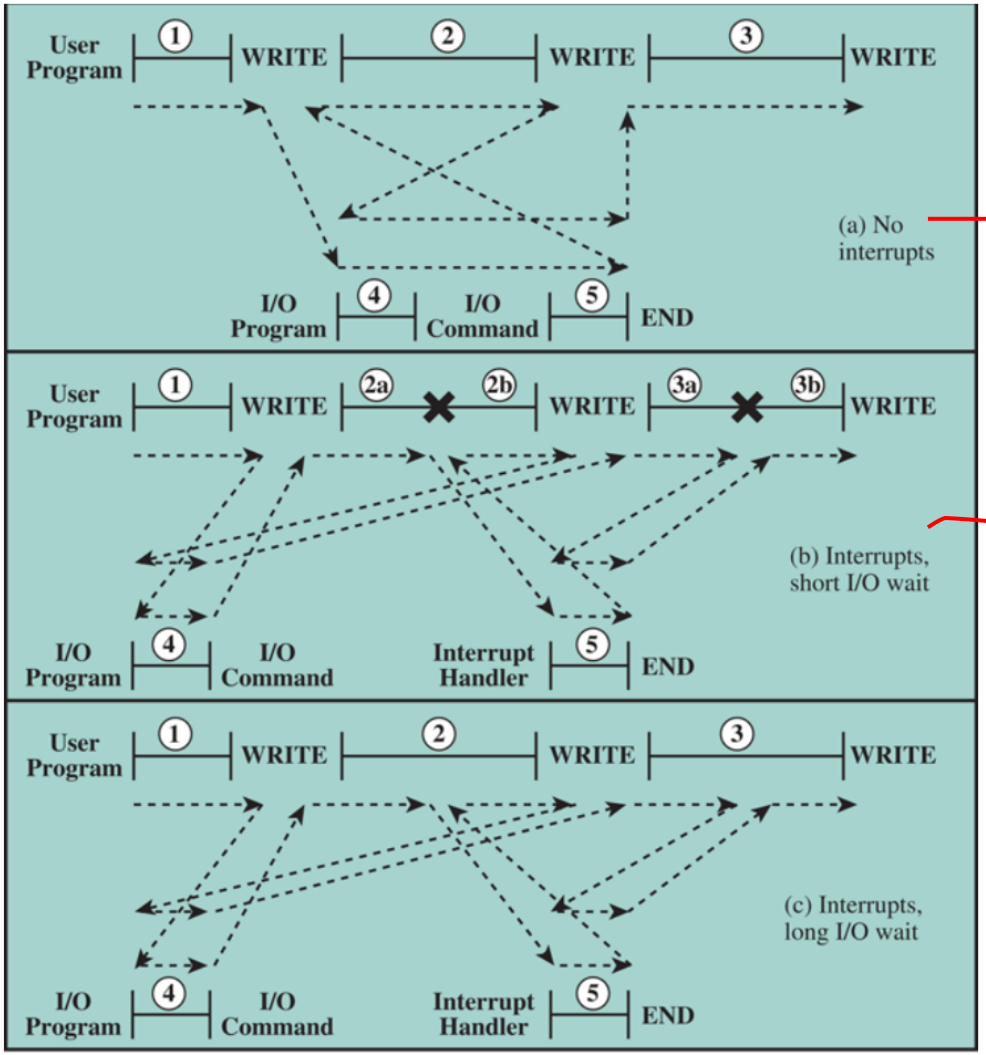
- Mechanism by which other modules (e.g. I/O) may **interrupt** normal sequence of processing

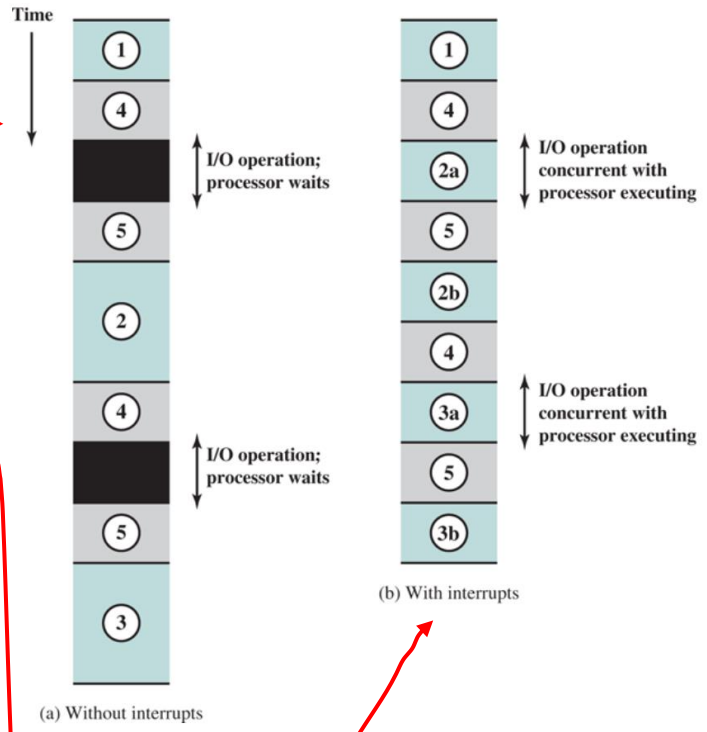
- Provided to improve processing efficiency

Classes of Interrupts

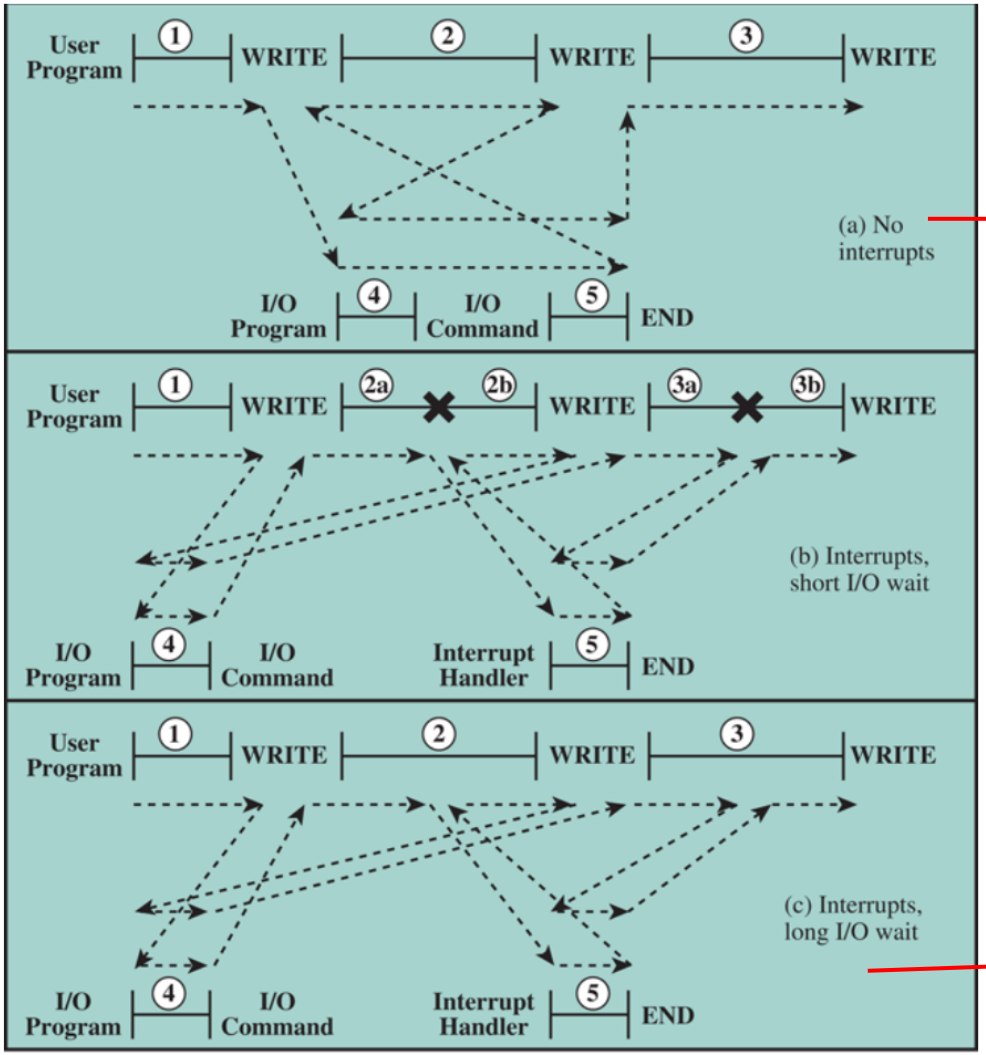
- Program
  - Result of an instruction execution (e.g. overflow, division by zero)
- Timer
  - Generated by internal processor timer
  - Used in pre-emptive multi-tasking
- I/O
  - From an I/O controller (e.g. signal completion of operation, request service, signal error conditions)
- Hardware failure
  - e.g. memory parity error, power failure

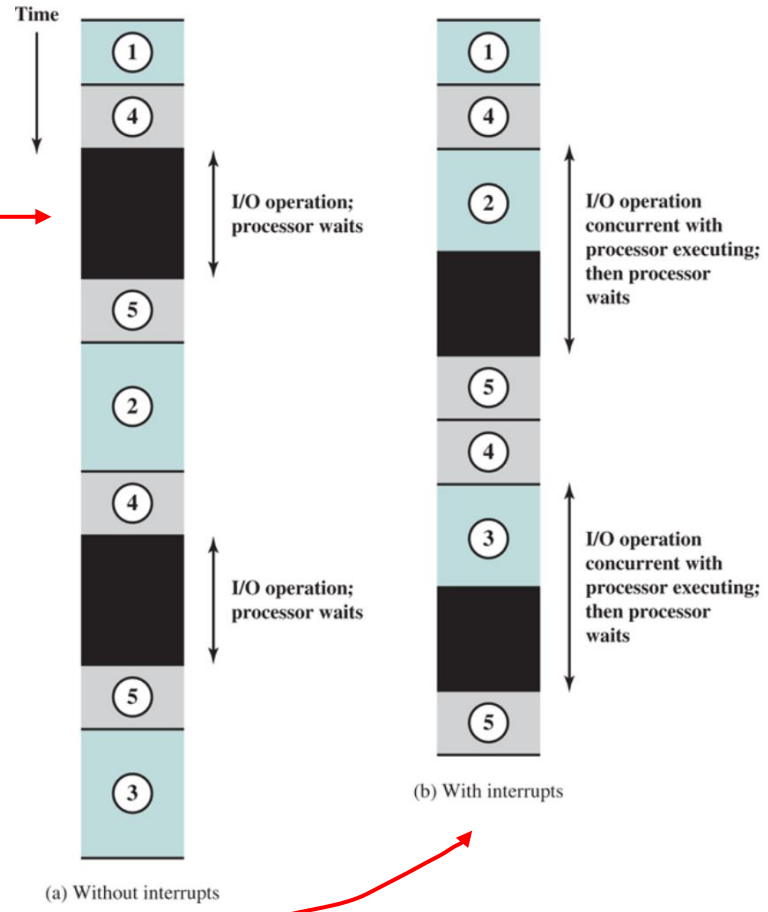
Program Flow Control (1)





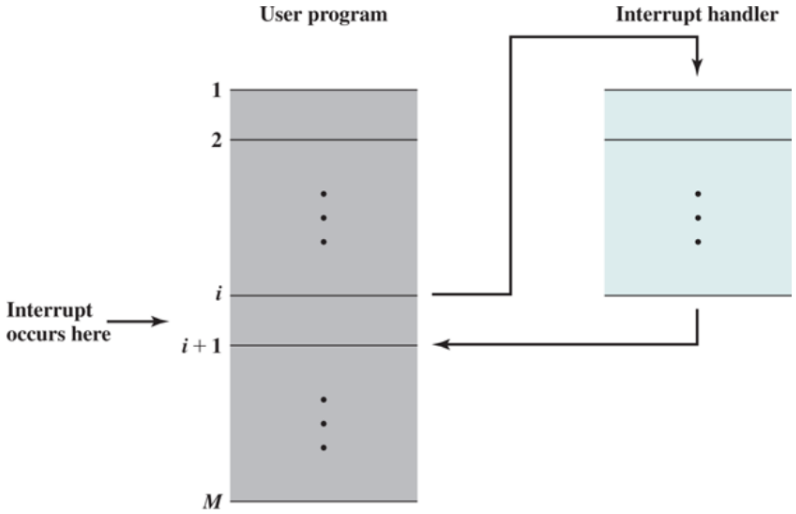
Program Flow Control (2)





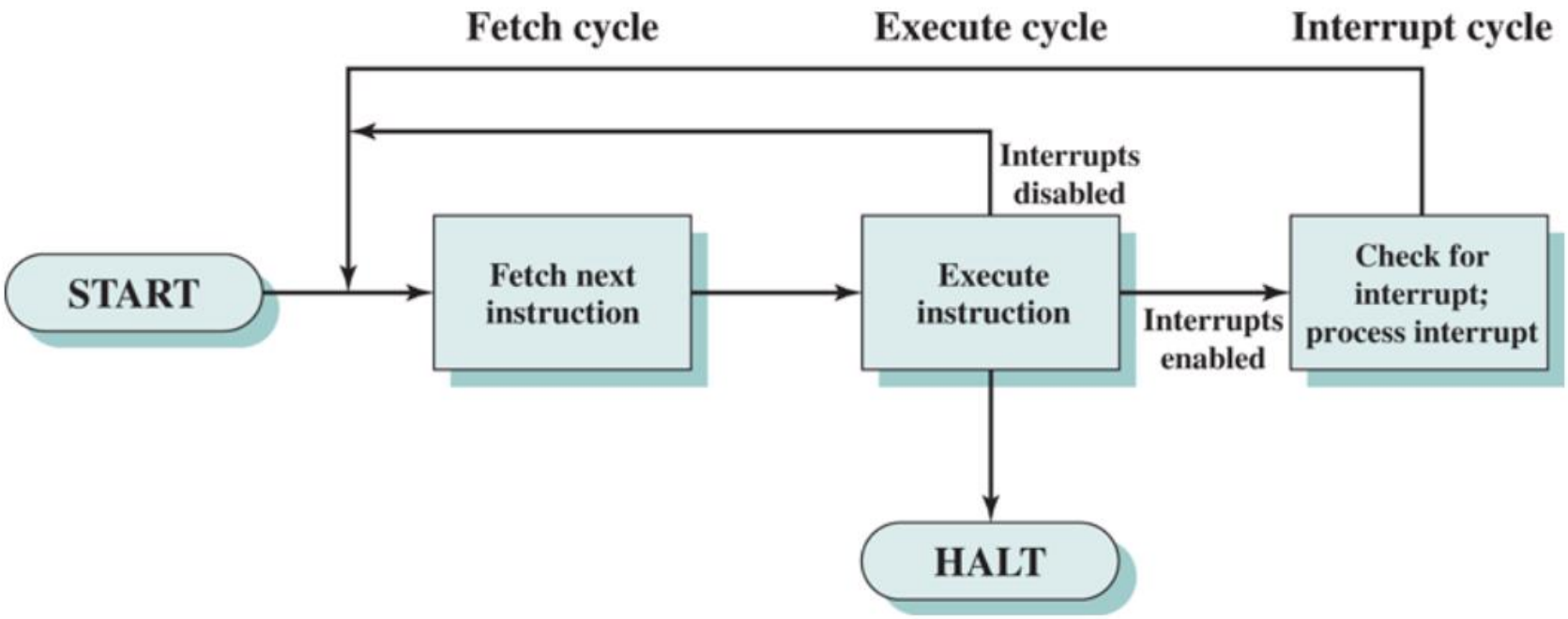
✖ = interrupt occurs during course of execution of user program

Interrupt Cycle

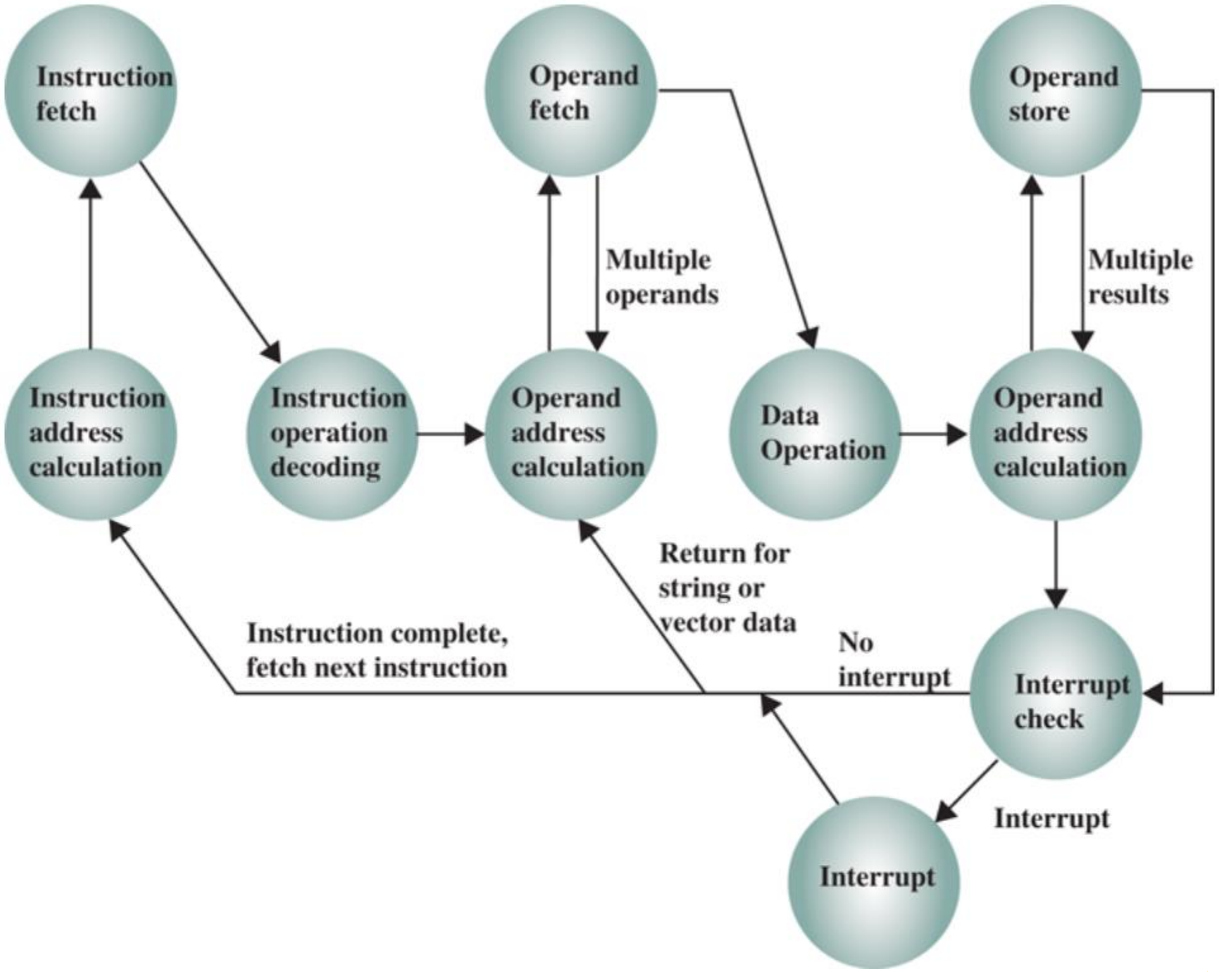


- Added to instruction cycle
- Processor checks for interrupt
  - Indicated by an interrupt signal
- If no interrupt, fetch next instruction
- If interrupt pending:
  - Suspend execution of current program
  - Save context
  - Set PC to start address of interrupt handler routine
  - Process interrupt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

Instruction Cycle with Interrupts



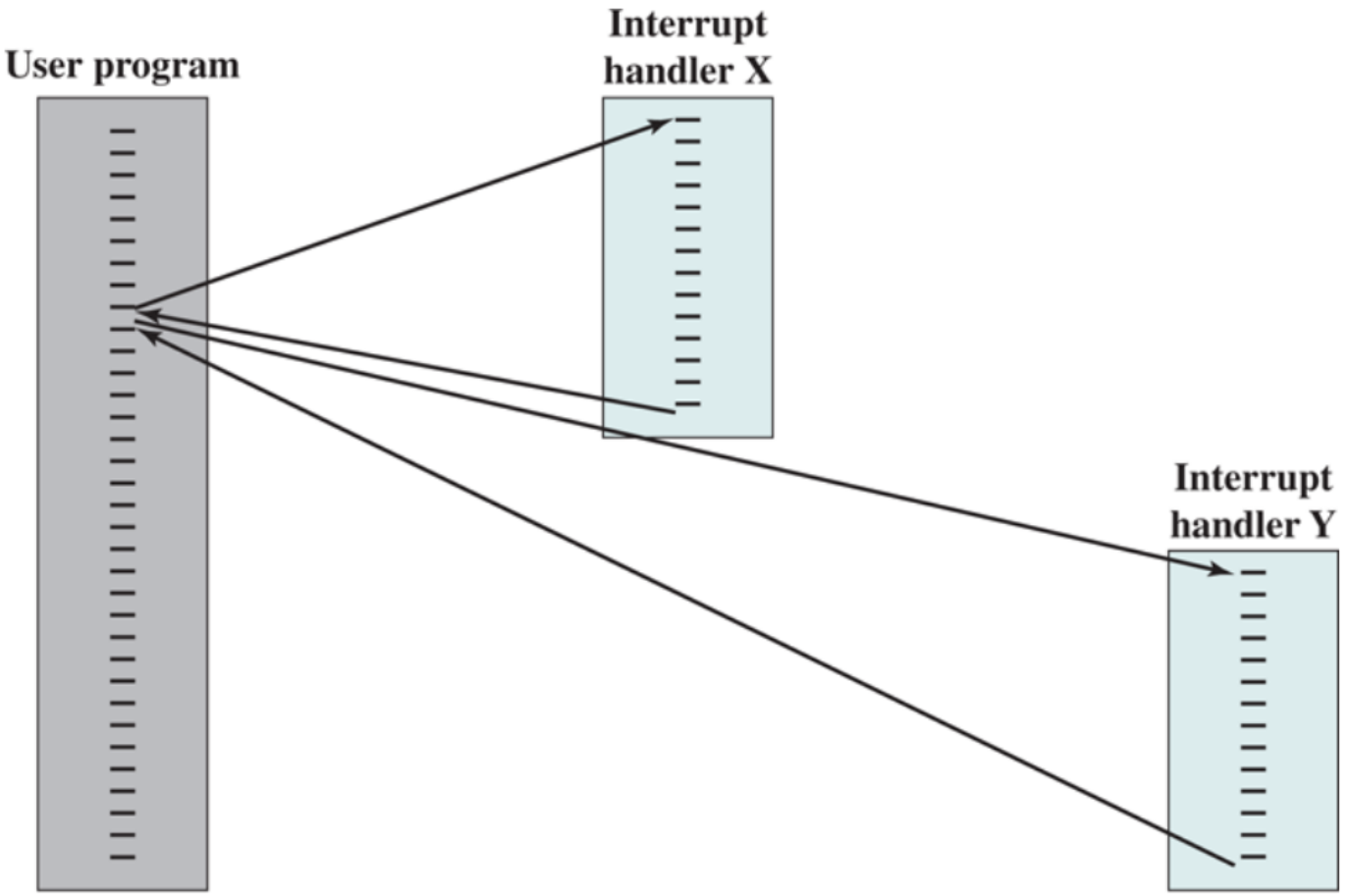
Instruction Cycle with Interrupts - State Diagram



Multiple Interrupts

- Disable interrupts
  - Processor will ignore further interrupts while processing one interrupt
  - Interrupts remain pending and are checked after first interrupt has been processed
  - Interrupts handled in sequence as they occur

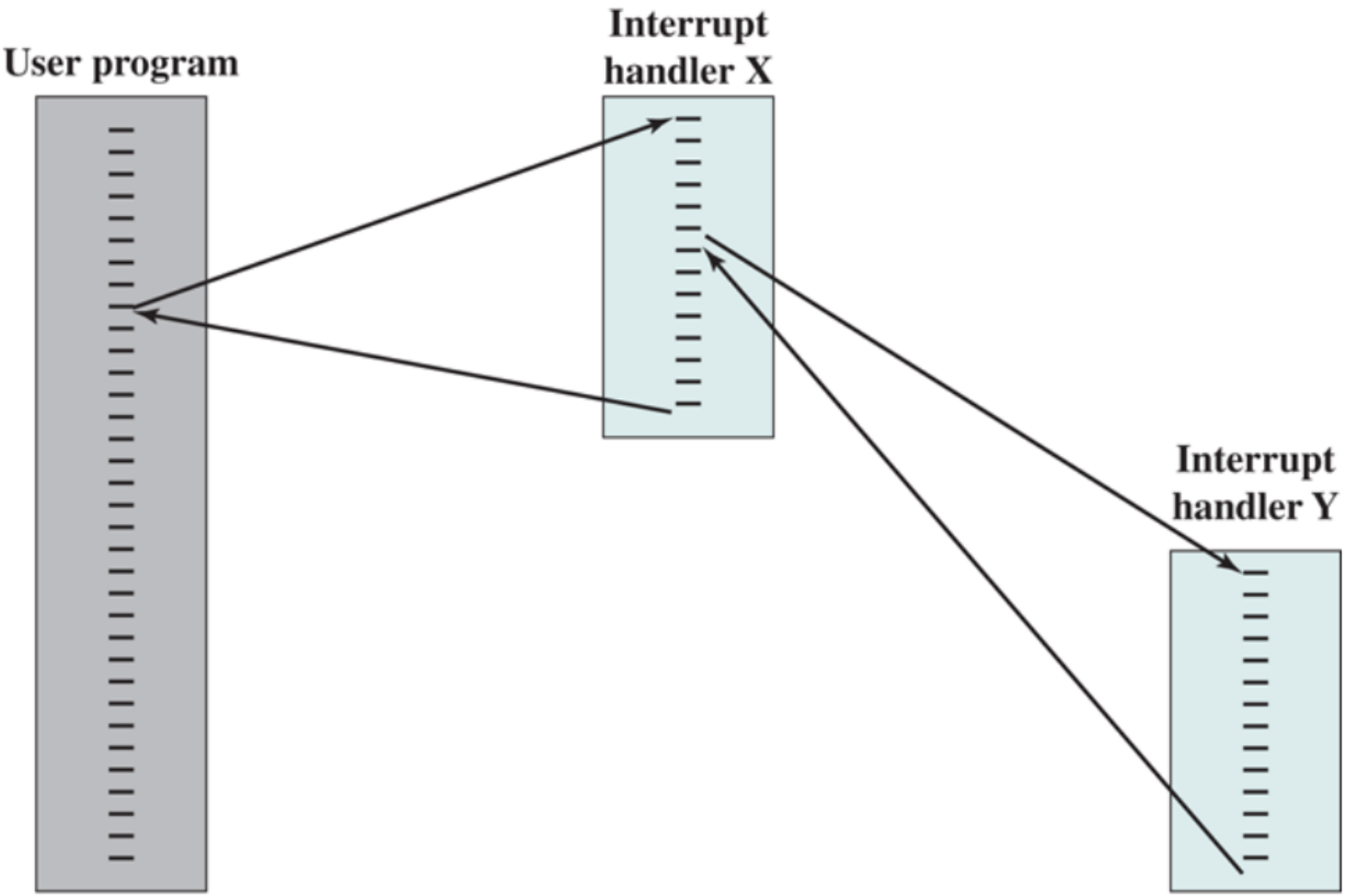
Multiple Interrupts - Sequential



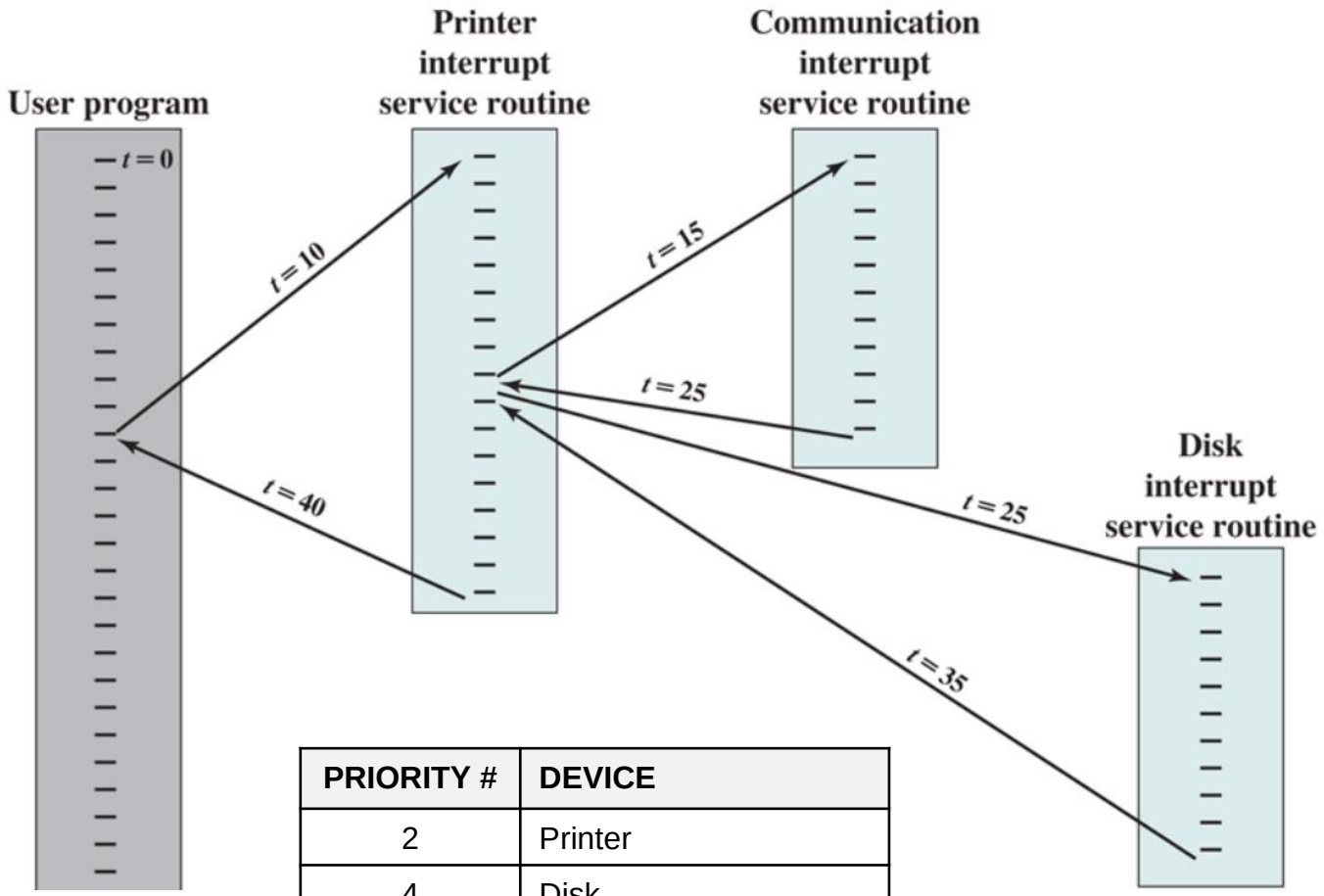
Multiple Interrupts

- Define priorities
  - Low priority interrupts can be interrupted by higher priority interrupts
  - When higher priority interrupt has been processed, processor returns to previous interrupt

Multiple Interrupts - Nested



Time Sequence of Multiple Interrupts



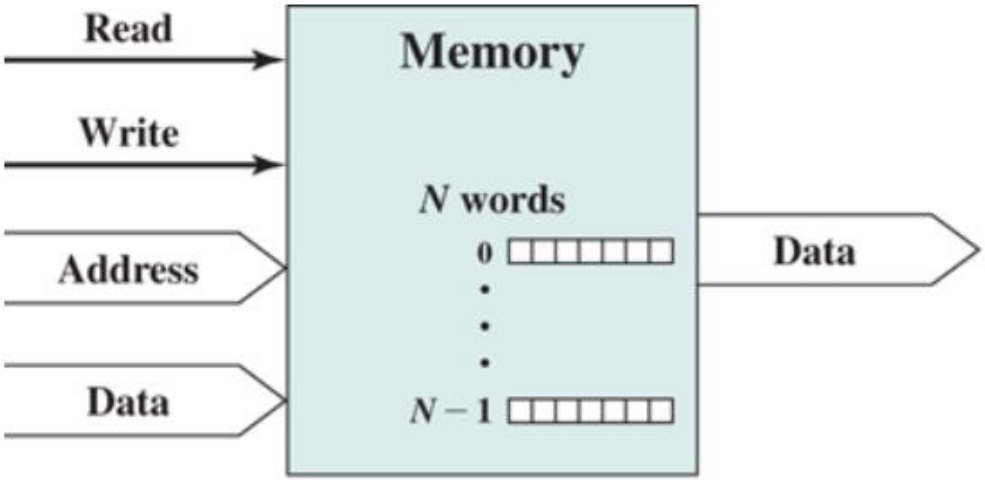
PRIORITY #	DEVICE
2	Printer
4	Disk
5	Communications Line

Interconnection Structure

- All the units must be connected
- Different type of connection for different type of unit
  - Memory
  - Input/Output
  - CPU

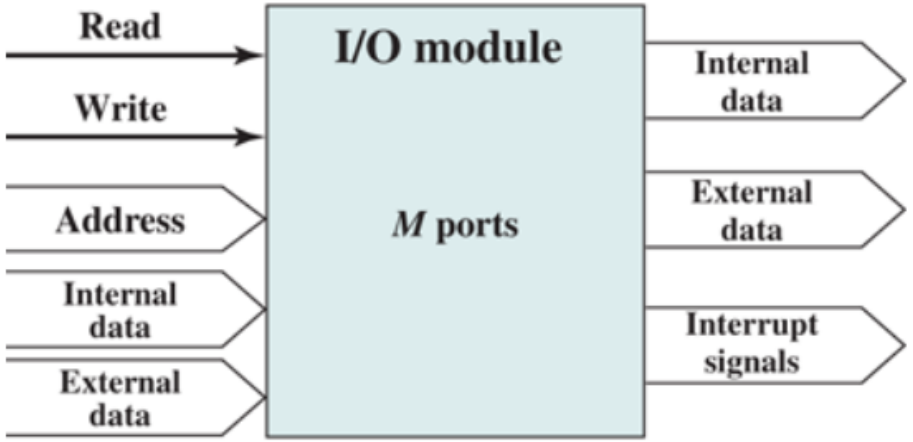
Memory Connection

- Receives and sends data
- Receives addresses (of locations)
- Receives control signals
  - Read
  - Write
  - Timing



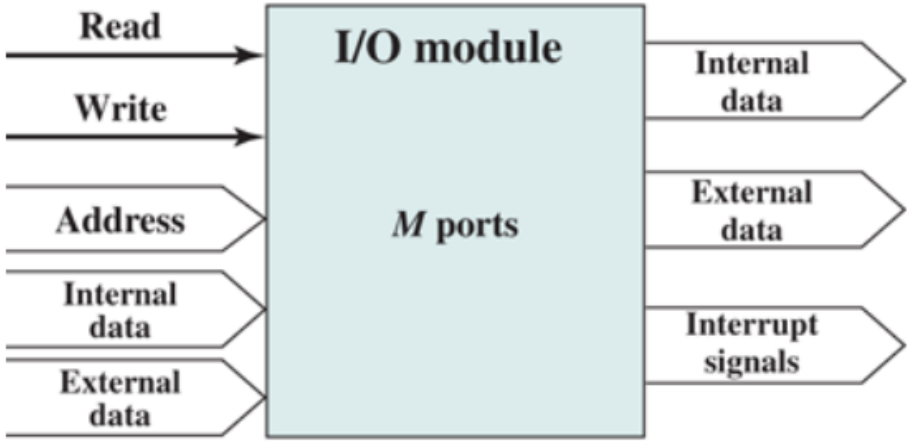
Input/Output Connection (1)

- Similar to memory from computer's viewpoint
- Output
  - Receive data from computer
  - Send data to peripheral
- Input
  - Receive data from peripheral
  - Send data to computer



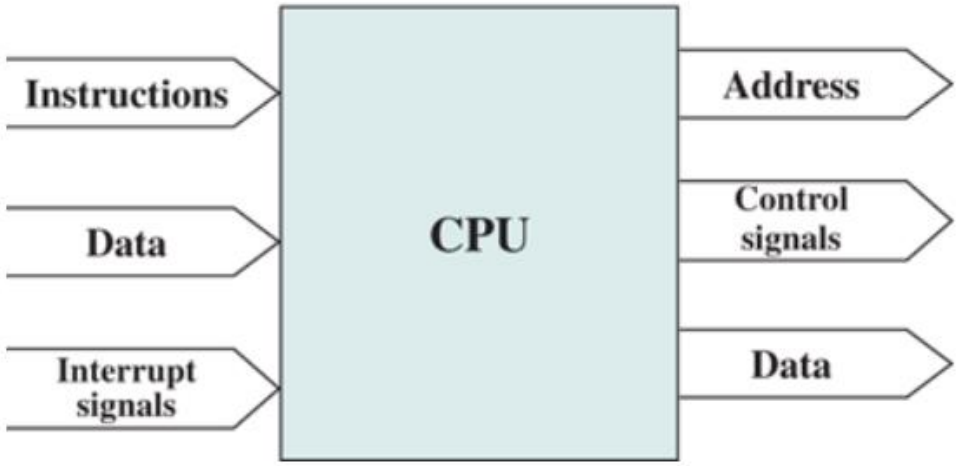
Input/Output Connection (2)

- Receive control signals from computer
- Send control signals to peripherals
  - e.g. spin disk
- Receive addresses from computer
  - e.g. port number to identify peripheral
- Send interrupt signals



CPU Connection

- Reads instruction and data
- Writes out data (after processing)
- Sends control signals to other units
- Receives (& acts on) interrupts



University of San Carlos | Department of
COMPUTER ENGINEERING

CpE 3202
Computer Organization & Architecture

End of Lecture

References:

- Brey, Barry B. *The Intel microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro processor, Pentium II, Pentium III, Pentium 4, and Core2 with 64-bit extensions: architecture, programming, and interfacing* / Barry B. Brey—8th ed.
- Stallings, W. Computer Organization and Architecture, 6th edition, Pearson Education, Inc. (2003).
- Stallings, W. Computer Organization and Architecture, 11th edition, Pearson Education, Inc. (2019).