CSC: Computer Organization & Assembly Language

1 – An Introduction

Instructor: Mr. Farooq Faisal

ICS & IT, FMCS, The University of Agriculture Peshawar

Outline

- About this Course
- Basic Structure of Computer
- What is Computer Organization?
- About Assembly Language

What is this course about?

Course Objectives

- To understand organization of a computer system
  - To gain an insight knowledge about the internal architecture and working of microprocessors.
  - To understand working of memory devices, interrupt controllers and I/O devices.
- To learn Assembly Language
  - To understand how low level logic is employed for problem solving by using assembly language as a tool.

Course Contents

- Basic Structure & Components of a Computer System
- Difference in Computer Organization & Computer Architecture
- Computer Evolution
- Microprocessor & Microcontrollers
- Interconnection Structures
- Memory Organization
- Data Representation
- Instruction Set
- Processor Structure & Function
- Interrupts
- Processor Registers & FLAGS

Course Contents Contd..

- Assembly Language
  - Syntax
  - Basic Instructions
  - Flow Control Instructions
  - High Level Language Structures
  - Logic, Shift and Rotate Instructions
  - The Stack
  - Multiplication & Division Instructions
  - Array & Addressing Modes
  - String Instructions
  - Procedures & Macros
  - Translation of high level language into assembly language.

Basic Structure & Function – Computer System

Structure

- **Structure** is the way in which components relate to each other

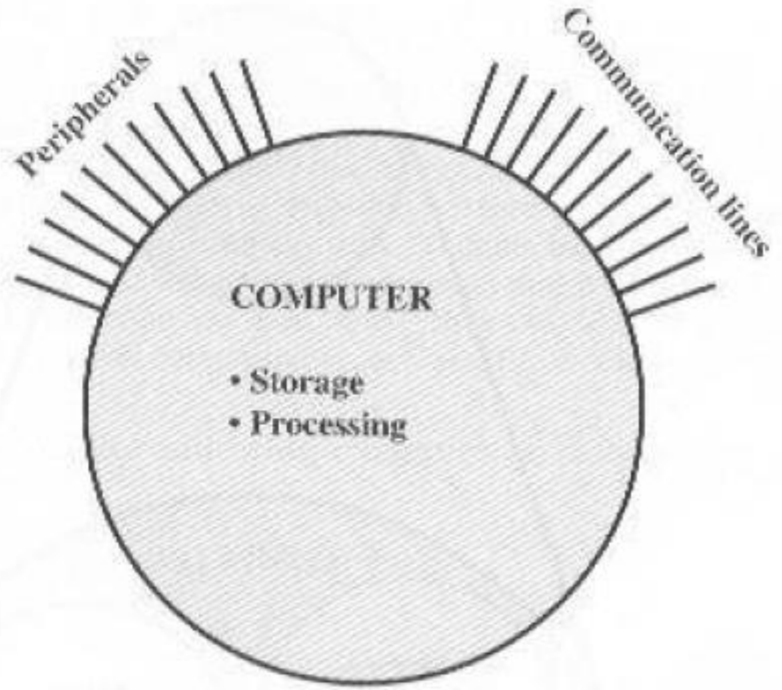
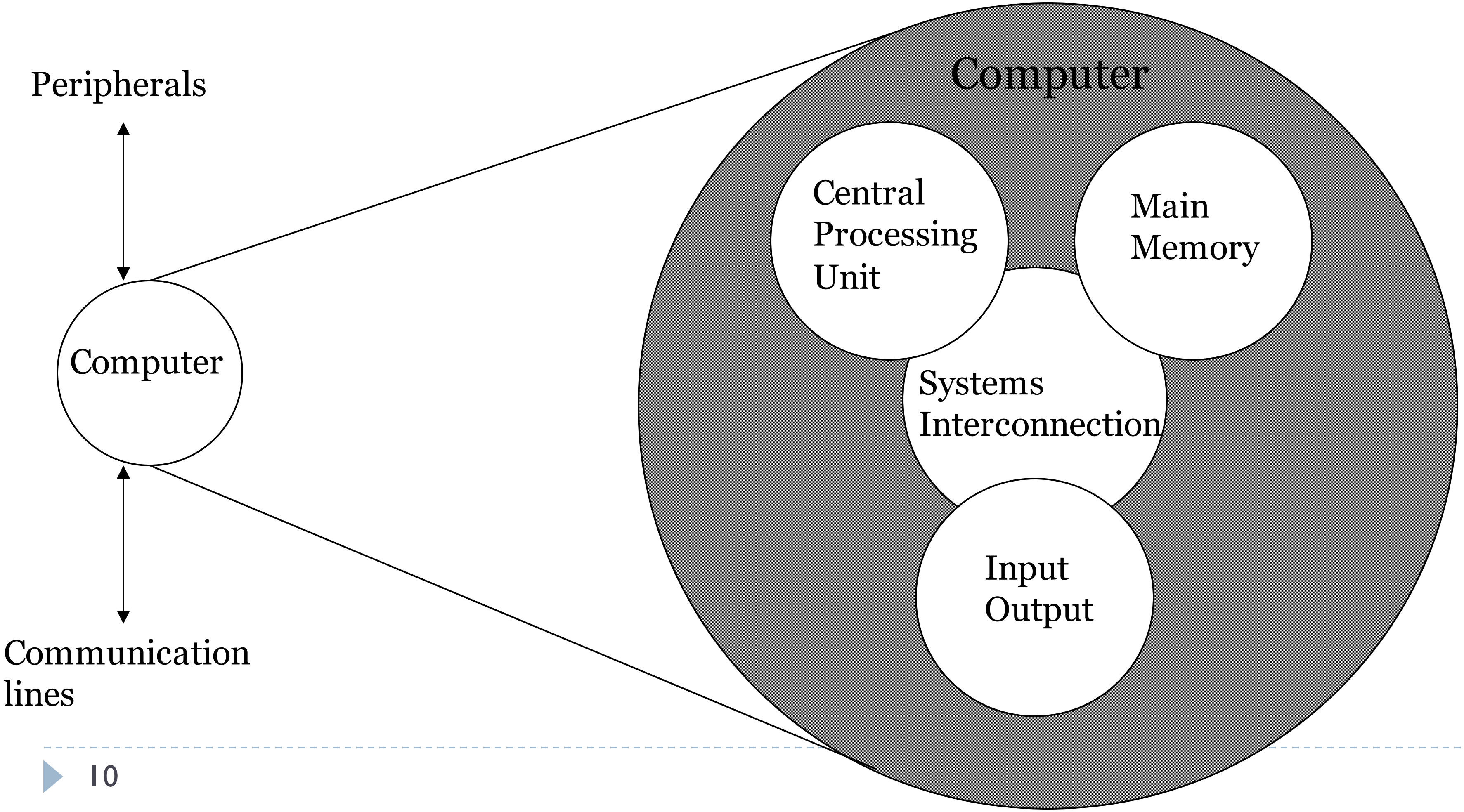


Figure 1.3 The Computer

Difference in Peripherals & Communication Lines

- When data is received from or delivered by a device that is directly connected to the computer, process is called **Input-Output** (I/O).
- When data are moved over longer distance, to or from a remote device, the process is known as **Data Communication**.

Structure - Top Level

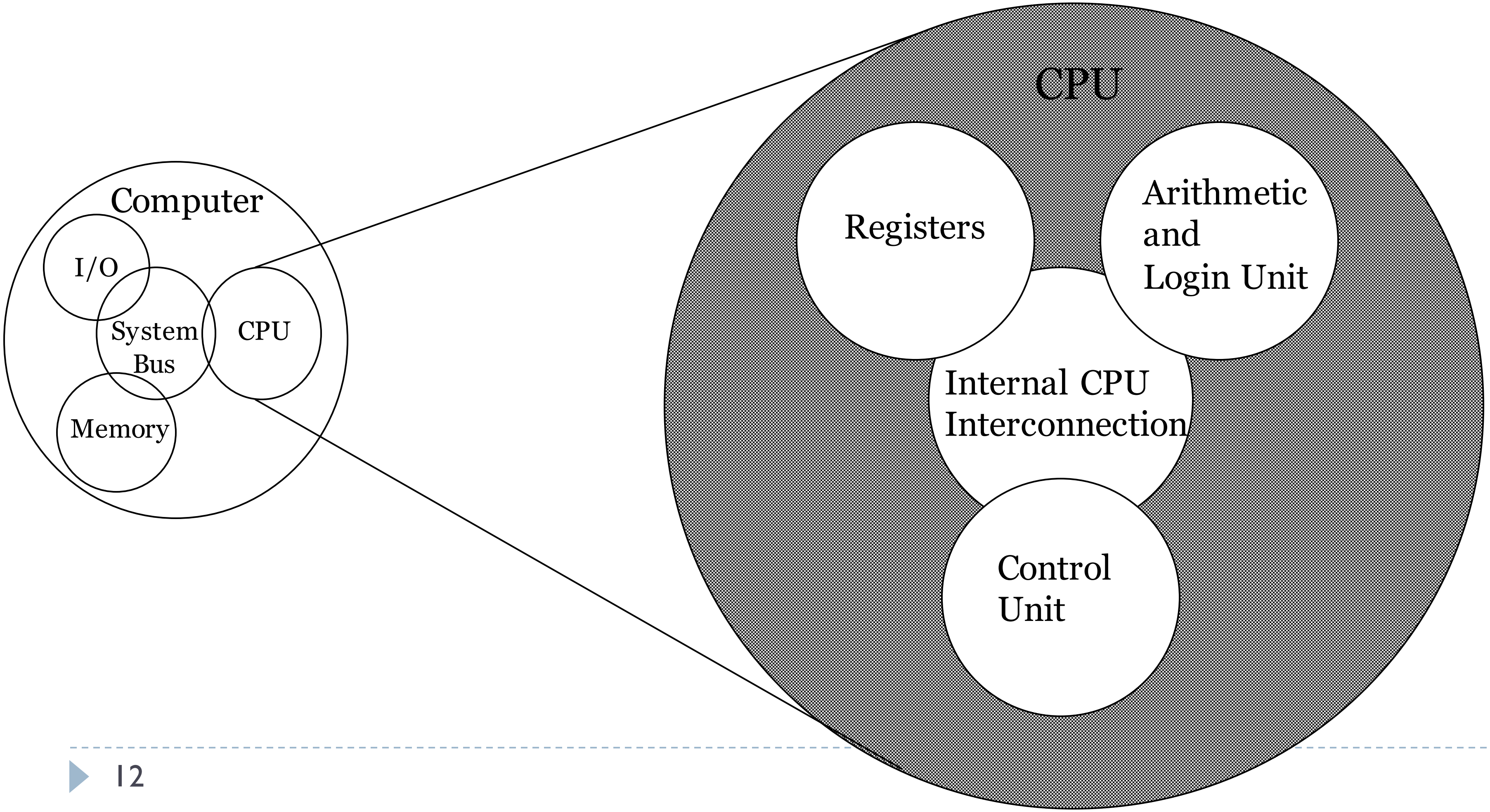


Structure – Top Level Contd..

- Four main structural components:

  - **CPU**: controls the operation of the computer and performs its data processing functions;often referred as processor.
  - **Main Memory**: stores data
  - **I/O**: moves data between the computer and its external environment.
  - **System Interconnections**: Mechanism for communication among CPU, memory, and I/O.

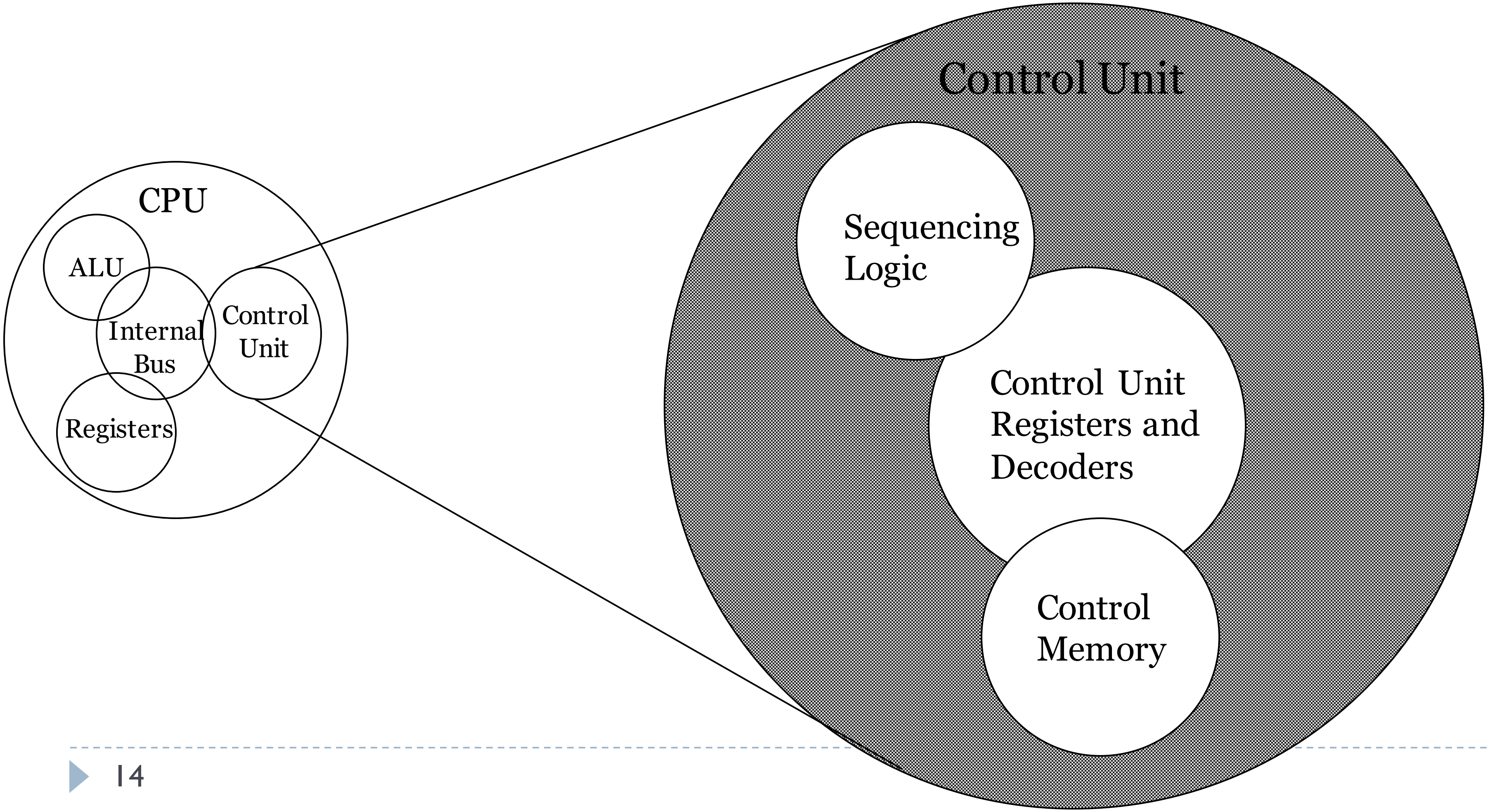
Structure - The CPU



Structure – The CPU

- **Control Unit**: controls the operation of CPU and hence the computer.
- **Arithmetic and logic unit**: performs the computer’s data processing functions.
- **Registers:** provides storage internal to CPU.
- **CPU interconnection**: Mechanism that provides for communication among the control unit,ALU, and registers.

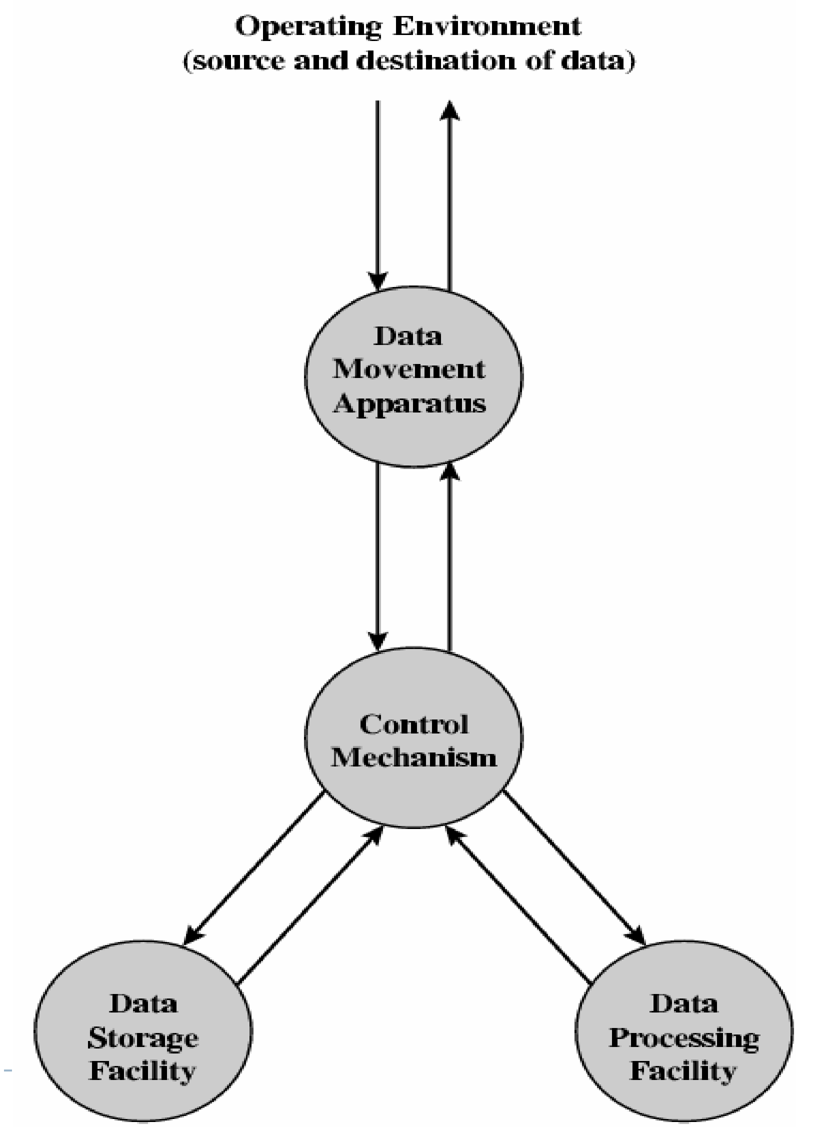
Structure - The Control Unit



Function

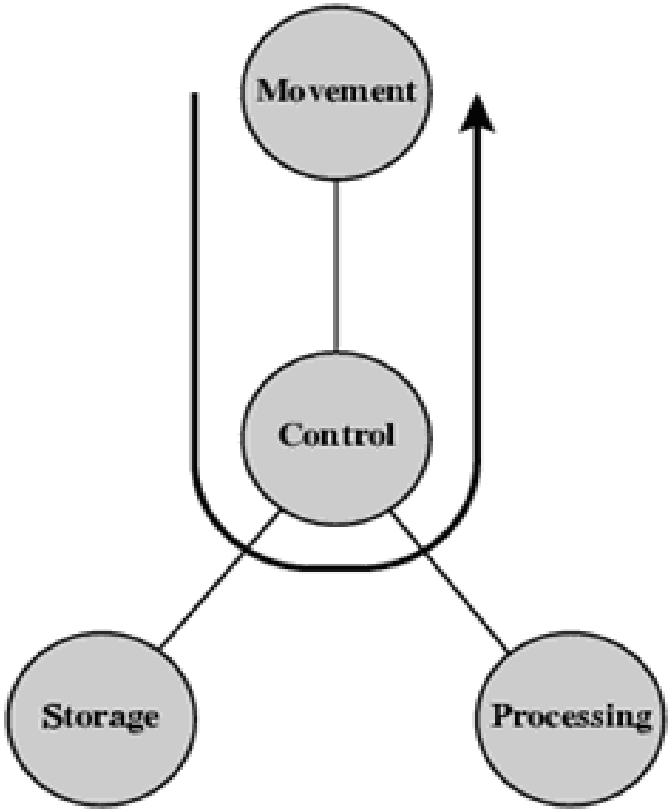
- **Function** is the operation of individual components as part of the structure.
- Main functions performed by a computer system are:
  - Process Data
  - Store Data
  - Move Data
  - Control the above three functions

Functional View of Computer



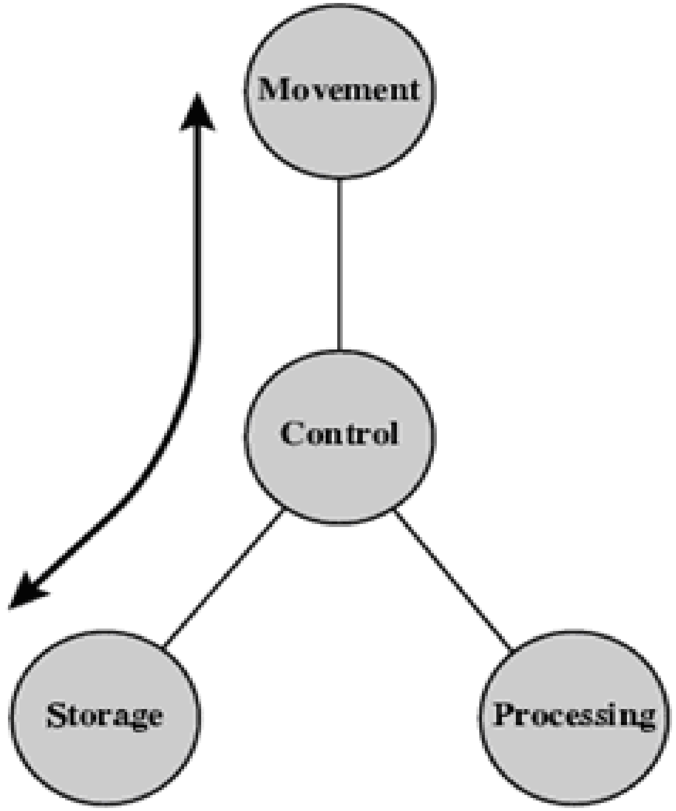
Possible Operations

Data movement device



(a)

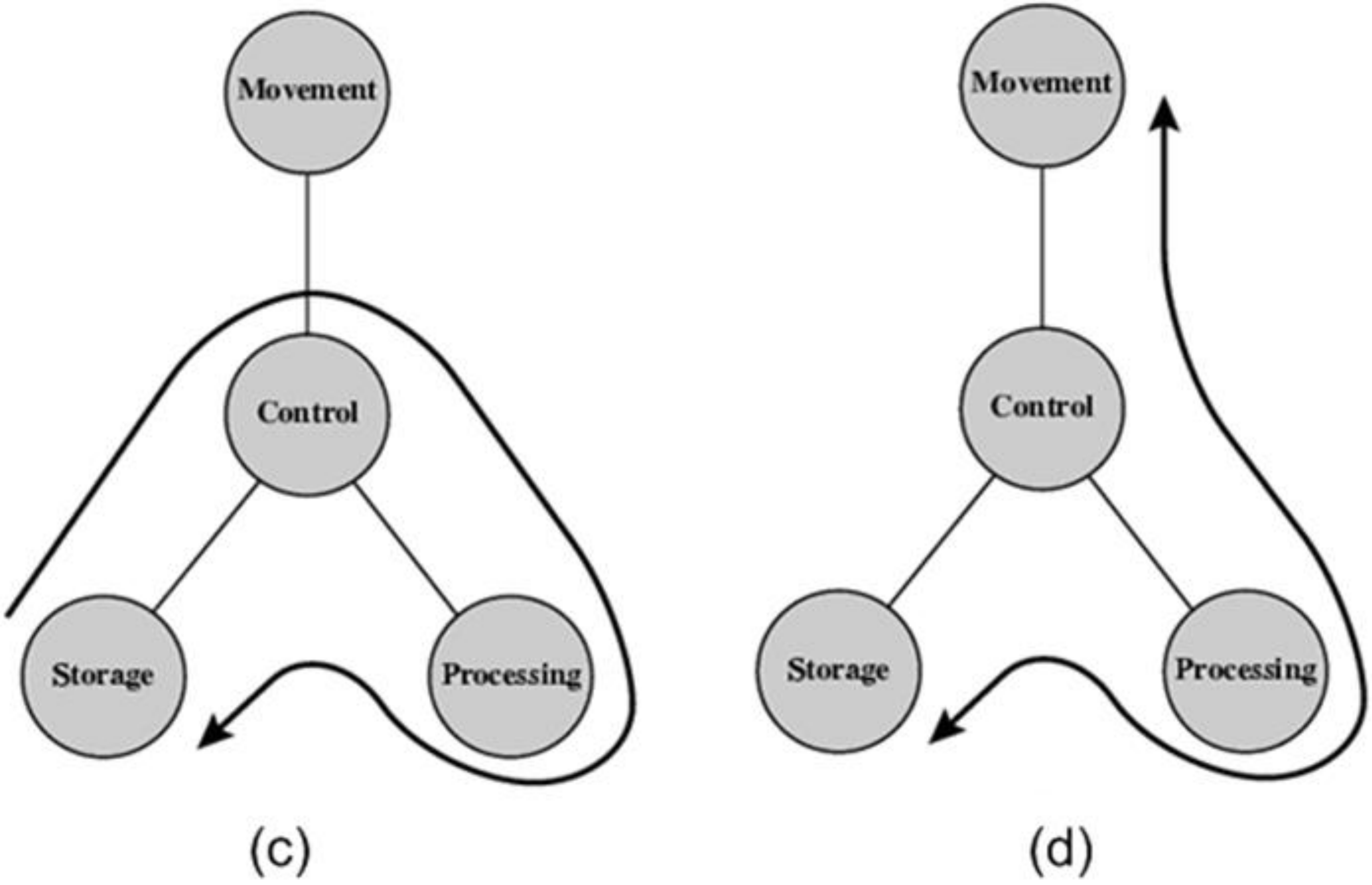
Data storage device (read/write)



(b)

Possible Operations Contd..

Processing on data stored in storage or in external environment



What is *C*omputer *O*rganization?

Computer Architecture

- Computer Architecture refers to those attributes of a system visible to a programmer
  - Those attributes that have direct impact on logical execution of a program.
- **Architectural attributes include**:
  - the instruction set,
  - no. of bits used to represent various data types (numbers, characters etc),
  - I/O mechanisms and technology for addressing memory.

- **Example**: Architectural design issue whether a computer will have multiply instruction or not.

What is Computer Organization?

- Organization is how features are implemented.

How does a Computer Work?

  - **For Example**: Is there a special hardware multiply unit for multiplication operation or is it done by repeated addition?
- *Computer Organization* refers to the operational units and their interconnections that realize the architectural specifications.
- **Organizational attributes**:
  - hardware details transparent to the programmer such as control signals,
  - interfaces between peripherals and the computer,
  - the memory technology used.

Computer Organization vs. Architecture

- Architecture:
  - Logical aspects of computer hardware that are visible to the programmer
    - What instruction a computer understands!
- Organization:
  - Physical aspects of computer hardware that are invisible to the programmer
    - How does the computer hardware carries out instructions!

Computer Organization vs. Architecture Contd..

- Computer Organization must be designed to implement a particular architectural specifications.

- It is possible to have same architecture but different organizations.
  - All computers in the Intel Pentium series have the same architecture.
  - Each version of the Pentium has a different organization or implementation.

Computer Organization vs. Architecture Contd..

- **Architectural Issues**:
  - Reduced Instruction Set Computing (RISC)
  - Complex Instruction Set Computing (CISC)
  - Pipeline etc

- **Organizational Issues**:
  - I/O, control unit, memory etc

Why Study Computer Organization?

- Understand how computer works!
  - Computer functional components, their characteristics, their performance, and their interactions.
- How to select a system?
  - Understand tradeoff among various components, such as memory size, CPU clock speed etc.

*A*ssembly *L*anguage

Computer Level Hierarchy

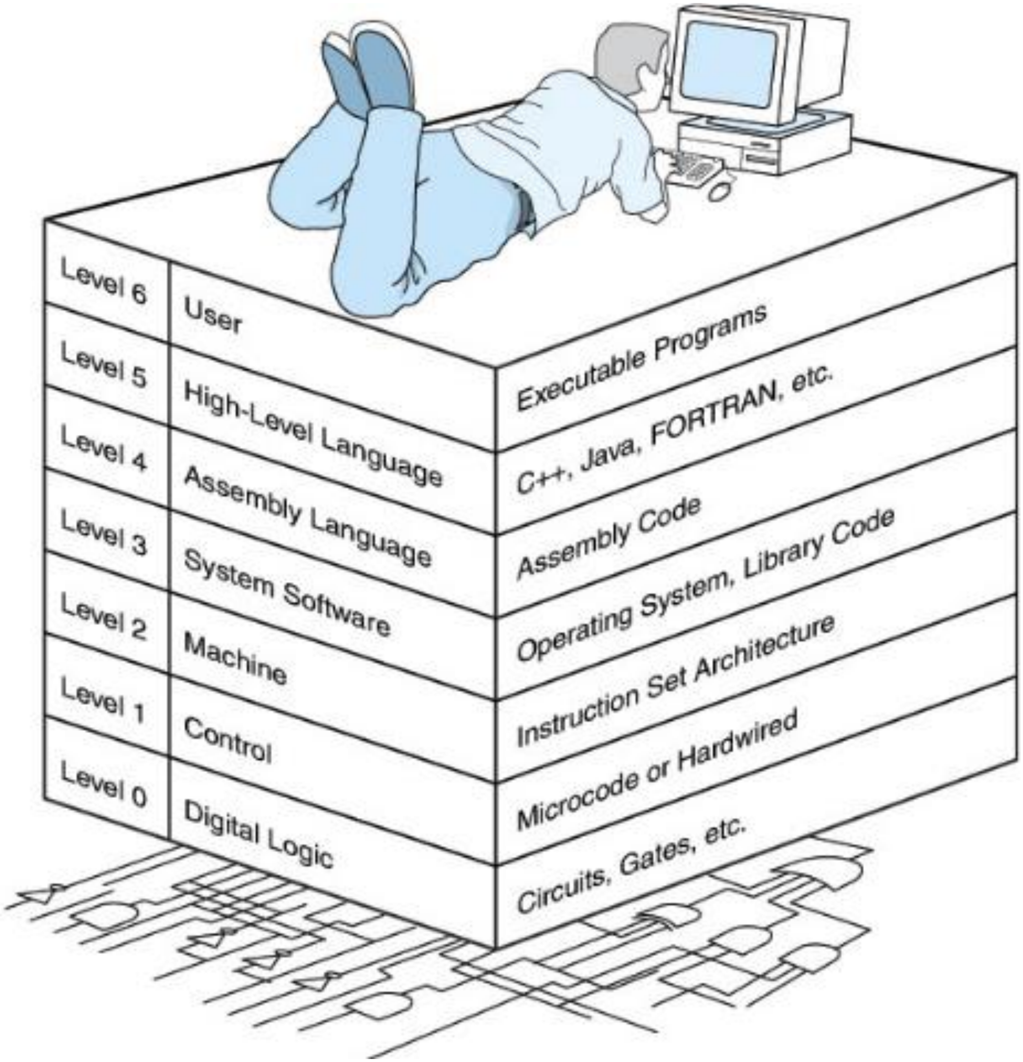


Figure Reference:
http://users.dickinson.edu/~braught/courses/cs251f09/topics/slides/intro.pdf

Programming Languages

- High-Level Languages (HLL)
- Assembly Language
- Machine Language

High-Level Language

- Allow programmers to write programs that look more like natural language.
- Examples: C++, Java, C#.NET etc
- A program called **Compiler** is needed to translate a high-level language program into machine code.
- Each statement usually translates into multiple machine language instructions.

Machine Language

- The "native" language of the computer
- Numeric instructions and operands that can be stored in memory and are directly executed by computer system.
- Each ML instruction contains an *op code* (operation code) and zero or more operands.
- Examples:

Opcode	**Operand**	**Meaning**
40		**increment the AX register**
05	**0005**	**add 0005 to AX**

Assembly Language

- Use instruction mnemonics that have one-to-one correspondence with machine language.
- An ***instruction*** is a symbolic representation of a single machine instruction
- Consists of:

label	always optional
mnemonic	always required
operand(s)	required by some instructions
comment	always optional

Sample Program

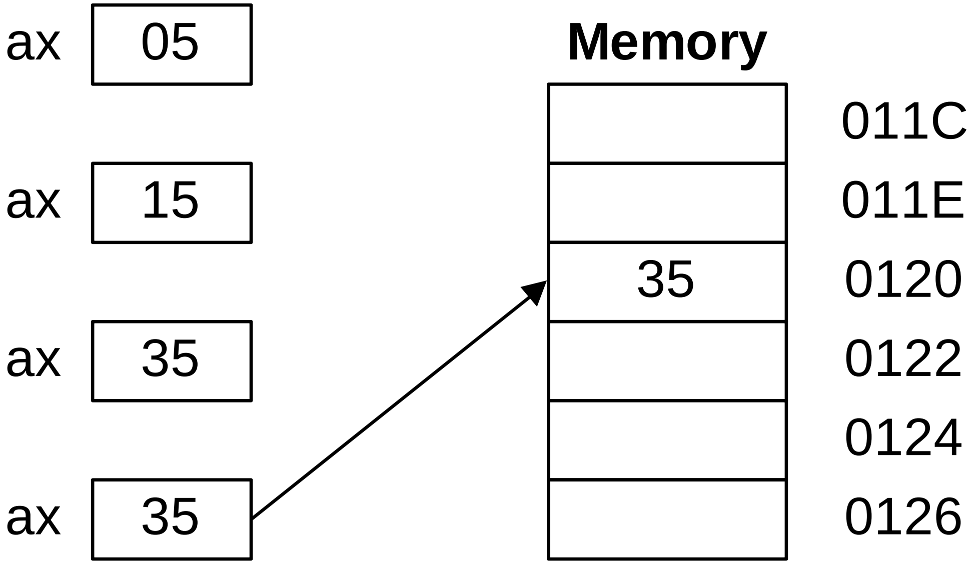
1. mov ax, 5

2. add ax, 10

3. add ax, 20

4. mov [0120], ax

5. int 20



ax 05

ax 15

ax 35

ax 35

Memory

011C

011E

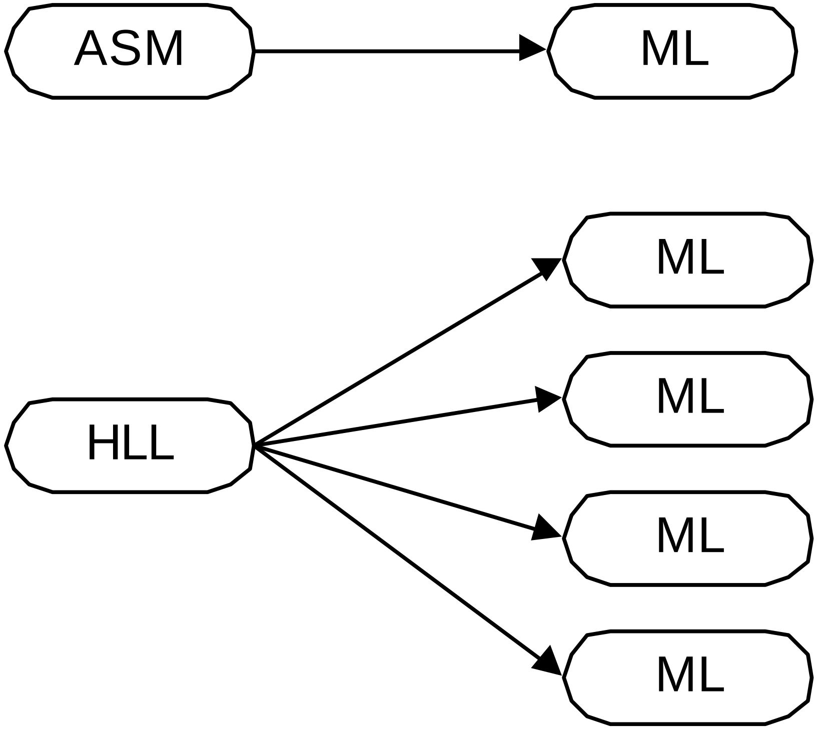
35 0120

0122

0124

0126

Figure: Machine Language Generation by ASM and HLL programs.



Essential Tools

- ***Assembler*** is a program that converts source-code programs into a machine language (*object fil*e).
- ***Linker*** joins together two or more object files and produces a single executable file.
- ***Debugger*** loads an executable program, displays the source code, and lets the programmer step through the program one instruction at a time, and display and modify memory.
- ***Emulator*** allows you to load and run assembly language programs, examine and change contents of registers. Example: EMU8086

Why Learn Assembly Language?

- Learn how a processor works
  - Explore the internal representation of data and instructions
  - How to structure a program so it runs more efficiently.

- Compilers/Device Drivers/ OS codes
- Games/Embedded System