CSCI -112
Introduction to computer Systems

Instructor: Santanu Banerjee

Courtesy: UMBC and JBLearning

Introduction to computer systems

Courtesy: UMBC and JBLearning

Topics

- Overview of computer systems
- Logical views and levels of abstractions
- Levels of machines
- Machine models: Von Neumann & System Bus
- Fetch-Execute Cycle

Overview of computer systems

Devices

Network

Hardware

Users

Software

Courtesy: UMBC and JBLearning

Hardware

....body....

- CPU
- Memory – Primary and Secondary
- Storage
- Interfaces
- Peripherals

Courtesy: UMBC and JBLearning

Devices

....tools....

- Keyboard
- Mouse
- Printer
- Scanner
- Camera

Courtesy: UMBC and JBLearning

Software

….mind….

- Programs
  - Operating systems
  - Device drivers
  - User Programs
  - Applications Programs
  - Scripts

Courtesy: UMBC and JBLearning

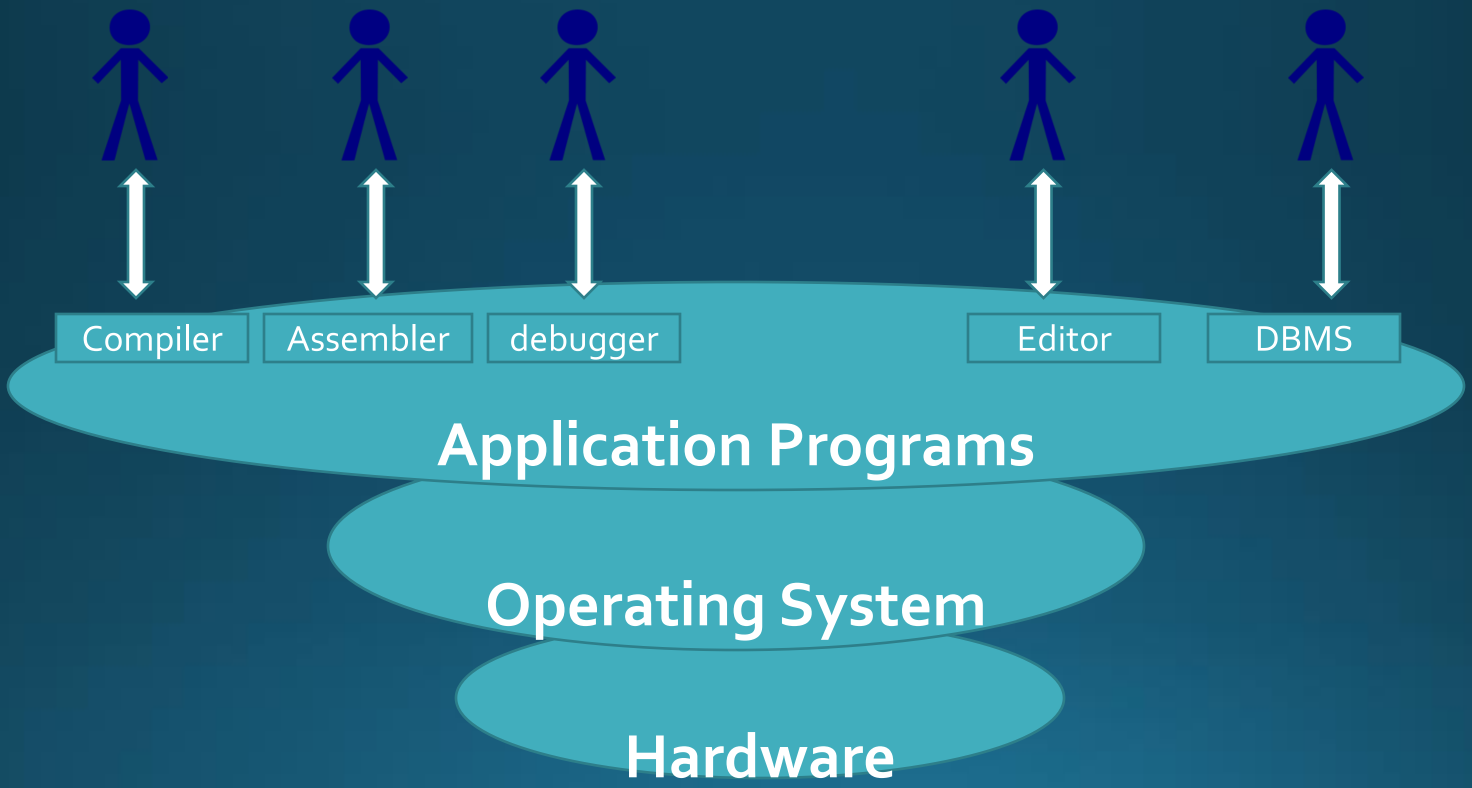
Network

….communication medium….

- LAN
- Internet
- USB
- P2P

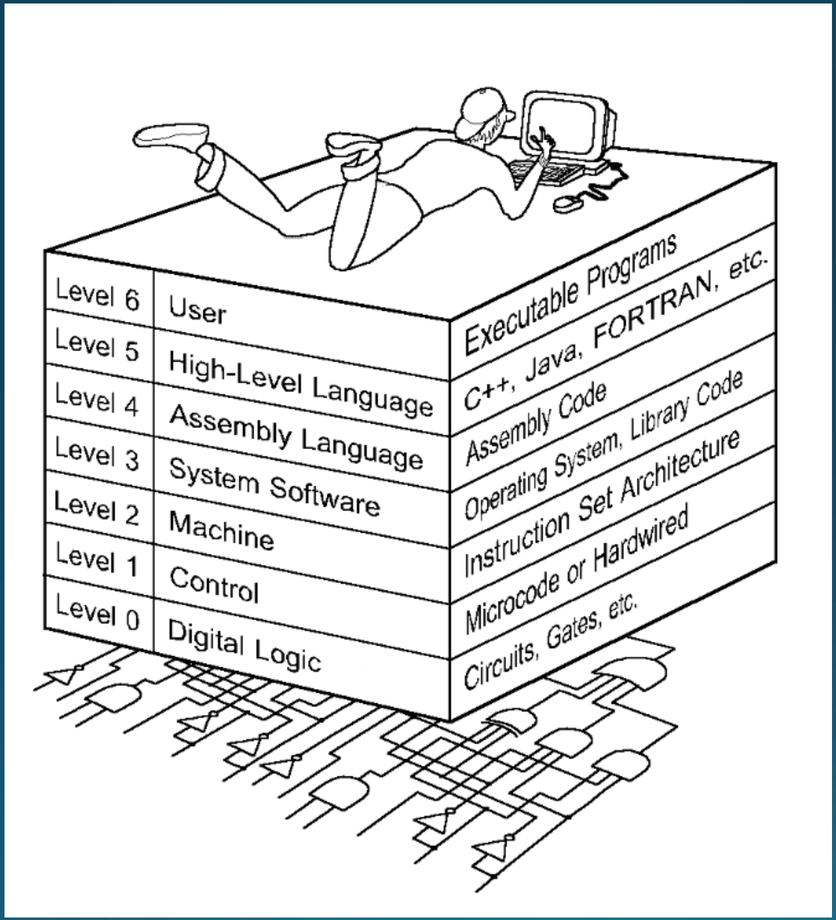
Courtesy: UMBC and JBLearning

Logical View – System



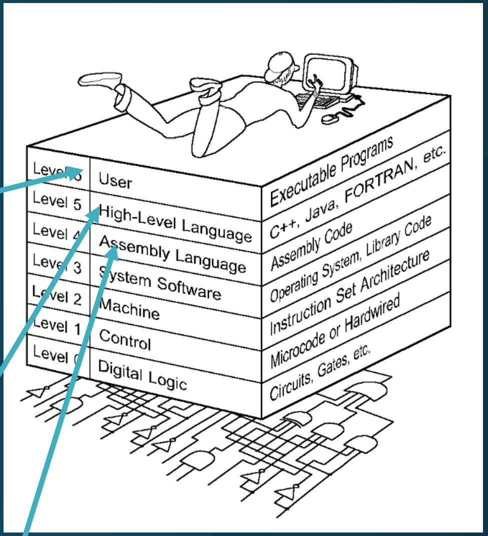
Courtesy: UMBC and JBLearning

Logical View – Hierarchy



Courtesy: UMBC and JBLearning

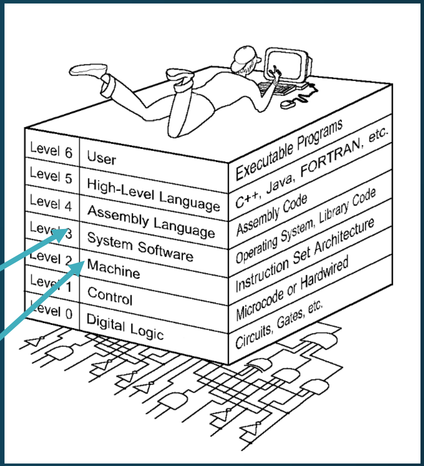
Levels of abstractions



- Level 6: The User Level
  - Program execution and user interface.
  - This is the living area for most.
- Level 5: High-Level Language Level
  - Programming (using languages such as C, Java).
- Level 4: Assembly Language Level
  - Acts upon assembly language produced from Level 5.
  - Assembly instructions programmed at this level.

Courtesy: UMBC and JBLearning

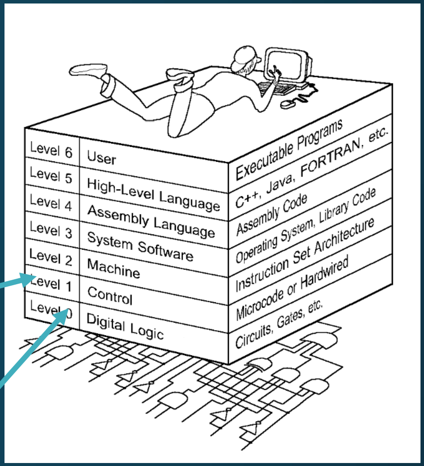
Levels of abstraction



- Level 3: System Software Level
  - Controls executing processes on the system.
  - Protects system resources.
  - Often a pass through for Assembly instructions.
- Level 2: Machine Level
  - Also known as the Instruction Set Architecture (ISA) Level.
  - Consists of instructions that are particular to the architecture of the machine.
  - Programs written in machine language need no compilers, interpreters, or assemblers.

Courtesy: UMBC and JBLearning

Levels of abstraction



- Level 1: Control Level
  - A *control unit decodes and executes instructions* and moves data through the system.
  - Control units can be *micro-programmed or hardwired.*
  - A micro-program is a program written in a low level language that is implemented by the hardware.
  - Hardwired control units consist of hardware that directly executes machine instructions.
- Level 0: Digital Logic
  - Core digital logic.
  - Implemented interconnected logic gates.

Courtesy: UMBC and JBLearning

CPU, Memory and drives

- CPU is the core where all the processing happens
  - Types
  - Speeds
  - Architecture
  - Multiple Cores
- Memory
  - Primary
  - Secondary
- Disk
  - IDE, SCSI, SATA…

Courtesy: UMBC and JBLearning

Storage -Bits and Bytes

- Storage
  - Memory
  - Hard Drive
  - Floppy Drive, Thumb Drive
  - Tape
- Bit
  - Smallest unit of info
  - True/False
  - Black/White
- Byte
  - Smallest addressable information storage unit
  - Consists of 8 bits

Courtesy: UMBC and JBLearning

Storage -Bits and Bytes

- Bytes – measurement units

File Storage Capacity by Powers of Two (Base 2)

	bit	byte	Kilobyte	Megabyte	Gigabyte	Terabyte	Petabyte	Exabyte	Zettabyte	Yottabyte
bit	2^0	2^3	2^13	2^23	2^33	2^43	2^53	2^63	2^73	2^83
byte	2^3	2^0	2^10	2^20	2^30	2^40	2^50	2^60	2^70	2^80
Kilobyte	2^13	2^10	2^0	2^10	2^20	2^30	2^40	2^50	2^60	2^70
Megabyte	2^23	2^20	2^10	2^0	2^10	2^20	2^30	2^40	2^50	2^60
Gigabyte	2^33	2^30	2^20	2^10	2^0	2^10	2^20	2^30	2^40	2^50
Terabyte	2^43	2^40	2^30	2^20	2^10	2^0	2^10	2^20	2^30	2^40
Petabyte	2^53	2^50	2^40	2^30	2^20	2^10	2^0	2^10	2^20	2^30
Exabyte	2^63	2^60	2^50	2^40	2^30	2^20	2^10	2^0	2^10	2^20
Zettabyte	2^73	2^70	2^60	2^50	2^40	2^30	2^20	2^10	2^0	2^10
Yottabyte	2^83	2^80	2^70	2^60	2^50	2^40	2^30	2^20	2^10	2^0

Addressing & architecture

- Addressing & architecture
  - 8 bit
  - 16 bit
  - 32 bit
  - 64 bit

Software – items

- Program - A set of instructions for a computer to follow.
- Data
- Input/output
- Network

Operating Systems

- Primary mind/soul of a computer system
- Closely related to hardware
- Provides a foundation/environment for all other software on the system
- Windows, Unix, MacOS

Courtesy: UMBC and JBLearning

Programs – ingredients

- Programming language
  - Native/High Level languages – C, C++
  - Assembly Language
  - Interpreted Languages & Scripts
  - Portable/Non native languages – Java/c#
- Compilers
- System and others libraries
- Linkers
- OS/Run time

Courtesy: UMBC and JBLearning

ENIAC

Electronic Numerical Integrator And Computer

- Was the first electronic general-purpose computer.
- Could solve "a large class of numerical problems" through reprogramming.

Courtesy: UMBC and JBLearning

The Von Neumann Model

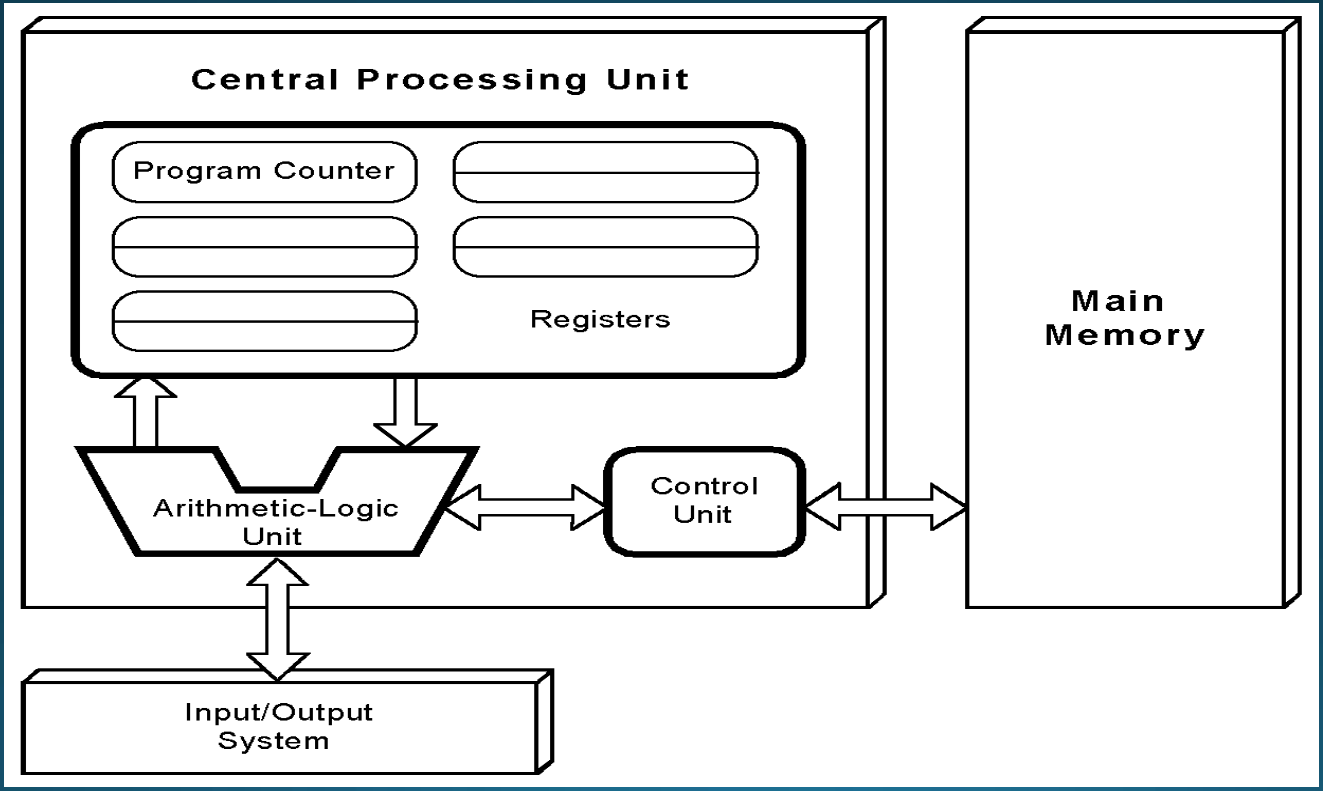
- On the ENIAC, all programming was done at the digital logic level.
- Programming the computer involved moving plugs and wires.
- A different hardware configuration was needed to solve every unique problem type.
- Inventors of the ENIAC, John Mauchley and J. Presper Eckert, conceived of a computer that could store instructions in memory.
- The invention of this idea has since been ascribed to a mathematician, John von Neumann, who was a contemporary of Mauchley and Eckert.
- Stored-program computers have become known as von Neumann Architecture systems.

The Von Neumann Model

- Today's stored-program computers have the following characteristics:
  - Three hardware systems:
    - A central processing unit (CPU)
    - A main memory system
    - An I/O system
  - The capacity to carry out sequential instruction processing.
  - A single data path between the CPU and main memory.
    - This single path is known as the von Neumann bottleneck.

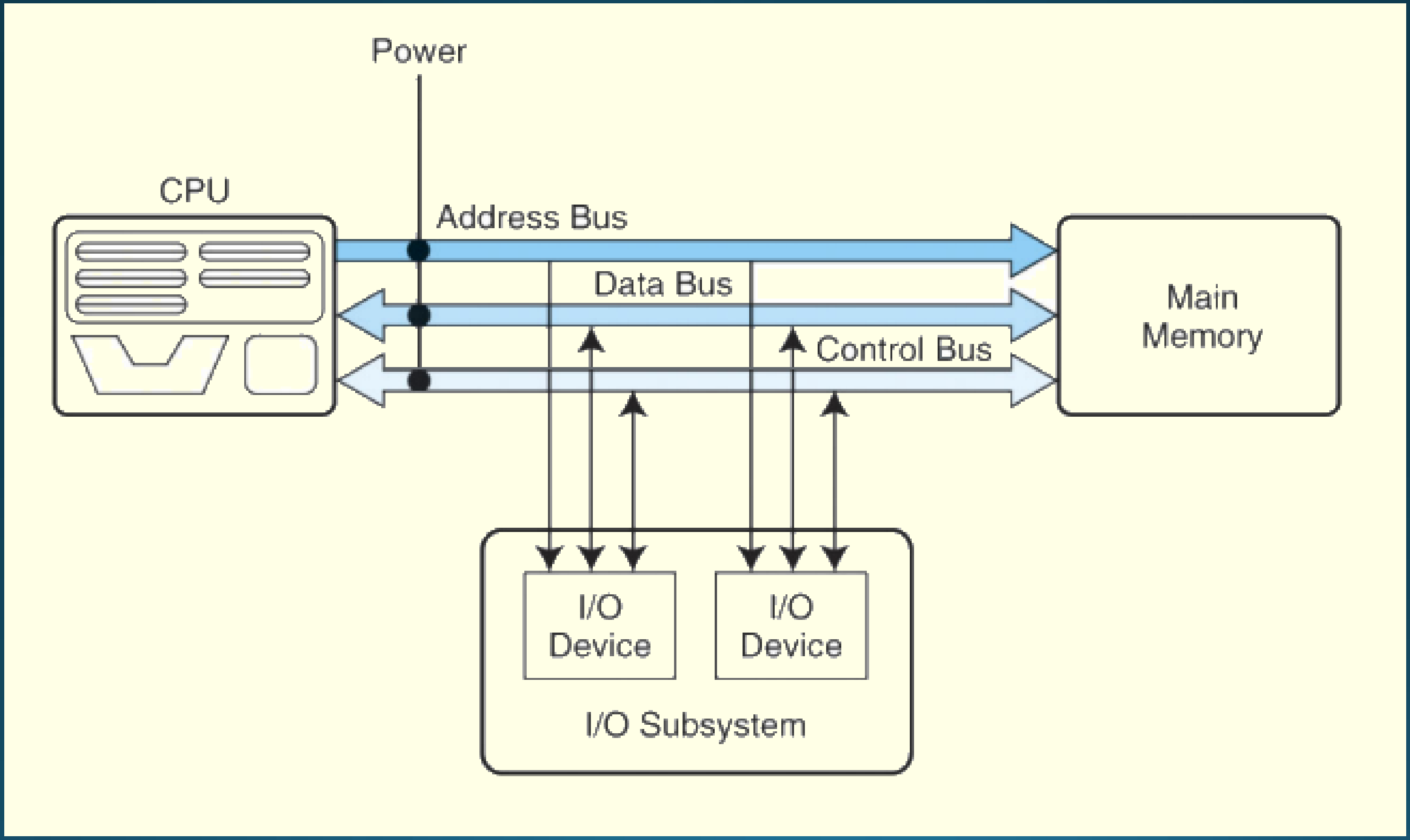
The Von Neumann Model

Fetch→ Decode →Execute →Fetch→…



Courtesy: UMBC and JBLearning

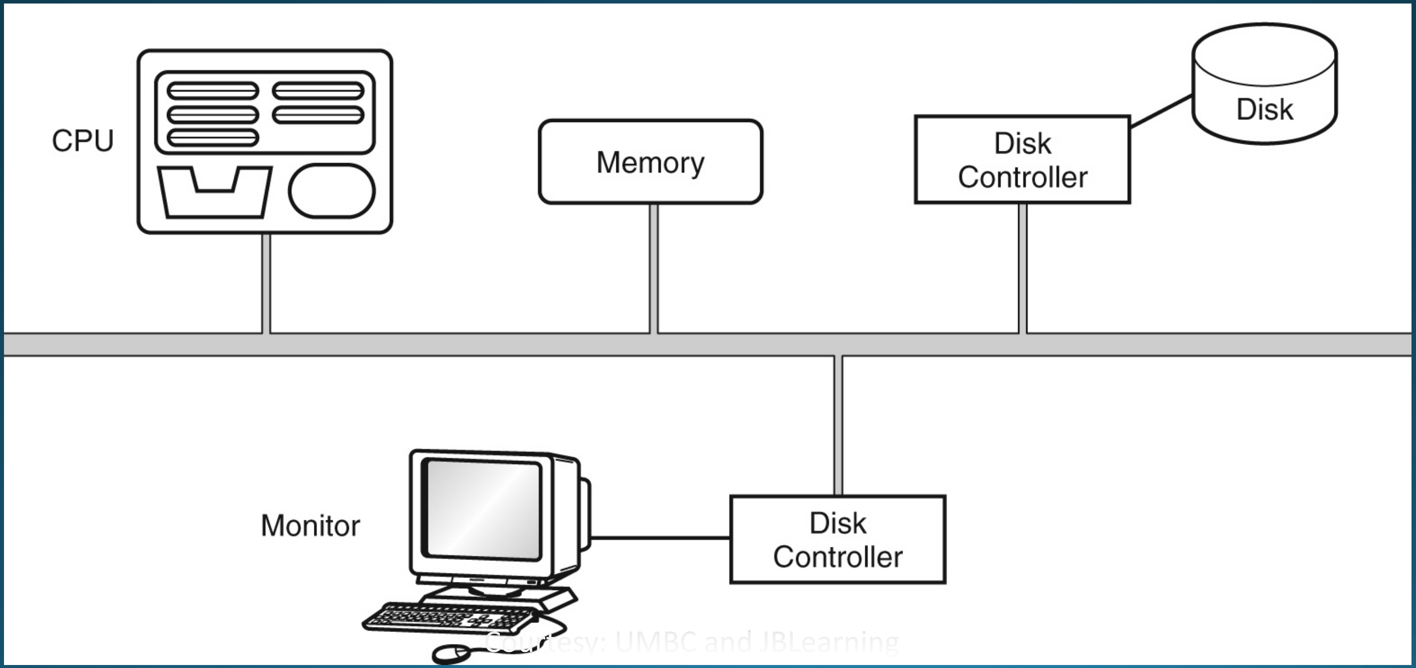
The Bus



Courtesy: UMBC and JBLearning

The Bus

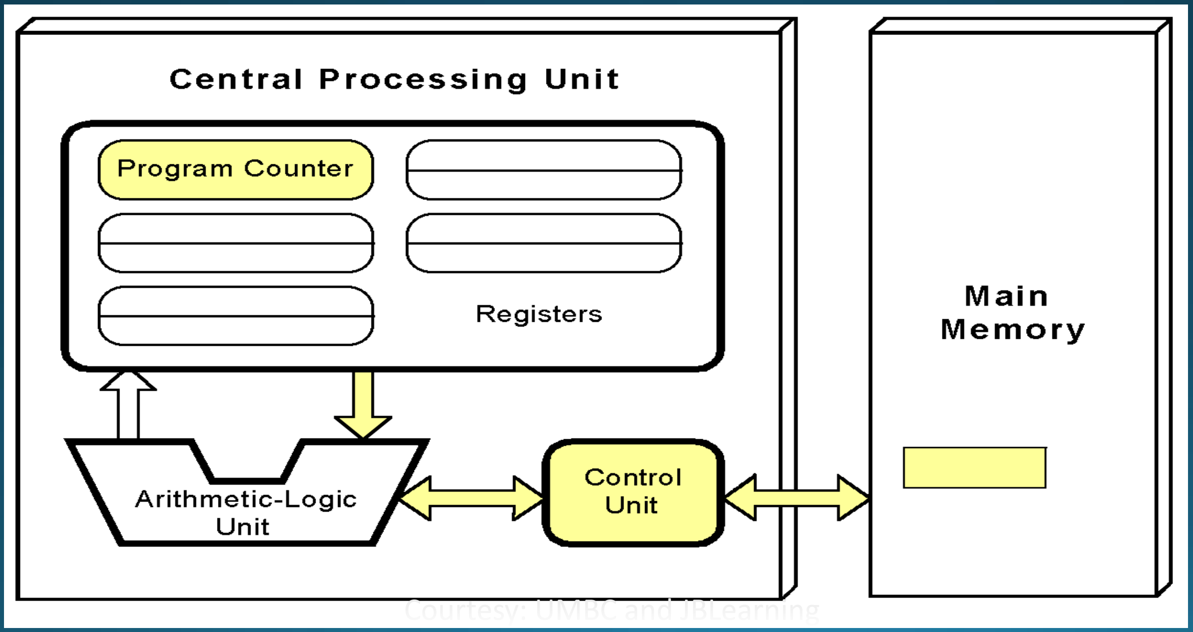
- A multipoint bus is shown below.
- Because a multipoint bus is a shared resource, access to it is controlled through protocols, which are built into the hardware.



FETCH DECODE EXECUTE

The von Neumann Model

- The control unit fetches the next instruction from memory using the program counter to determine where the instruction is located.

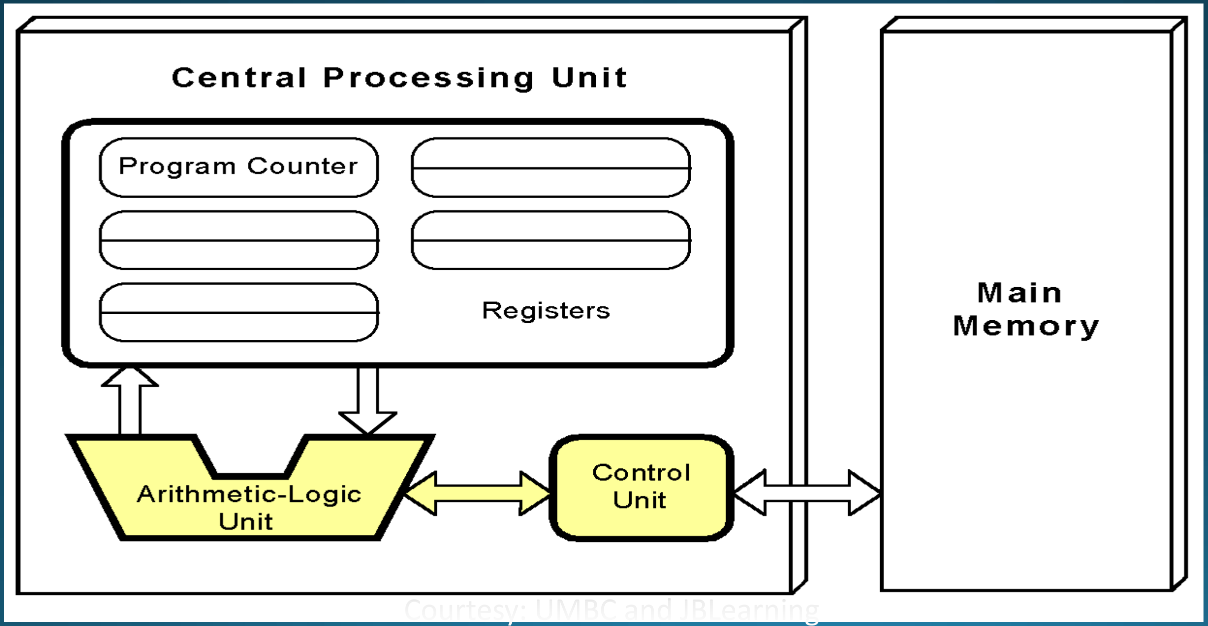


Courtesy: UMBC and JBLearning

FETCH DECODE EXECUTE

The von Neumann Model

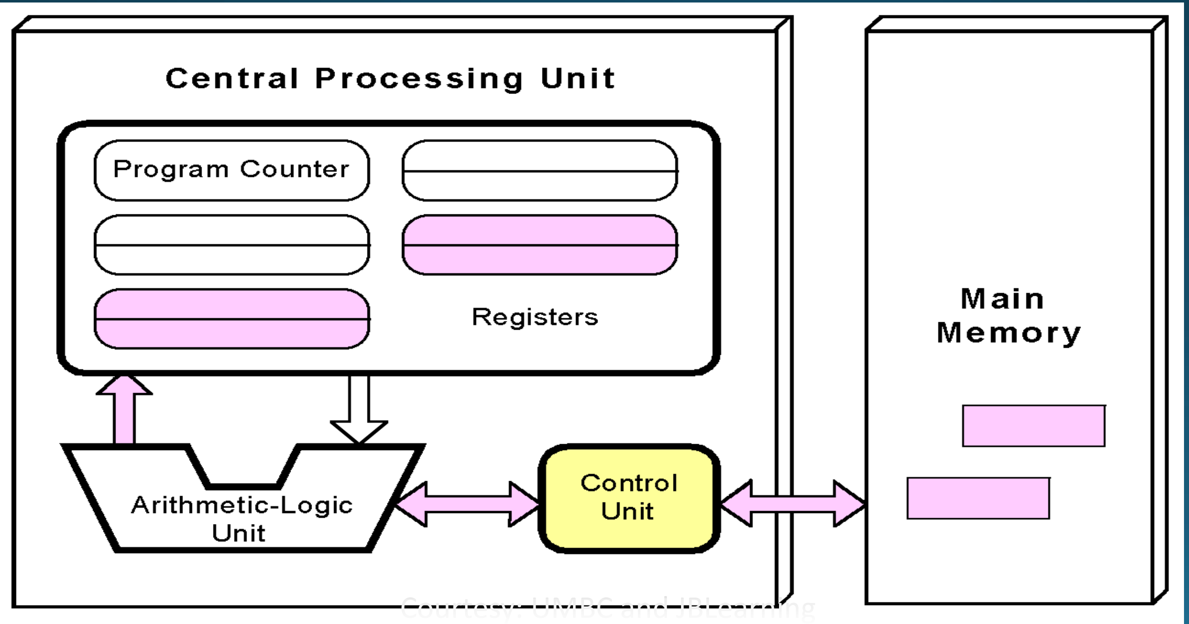
- The instruction is decoded into a language that the ALU can understand.



FETCH DECODE EXECUTE

The Von Neumann Model

- Any data operands required to execute the instruction are fetched from memory and placed into registers within the CPU.

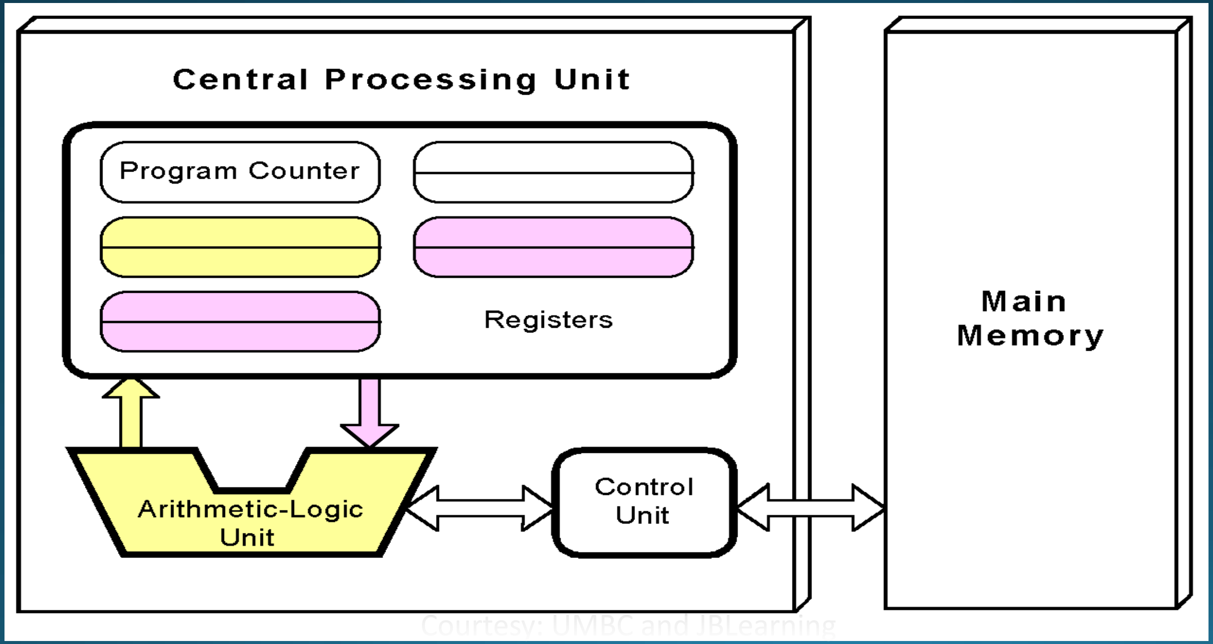


Courtesy: UMBC and JBLearning

FETCH DECODE EXECUTE

The von Neumann Model

- The ALU executes the instruction and places results in registers or memory.



Courtesy: UMBC and JBLearning

Q & A

?

Courtesy: UMBC and JBLearning