Computer Organization(CPSC 2500) Introduction

Today's Class

- Organizational meeting
  - Course organization & outline
  - Policies
  - Prerequisites & course sign-up
- Introduction to Computer Organization

Organizational Information

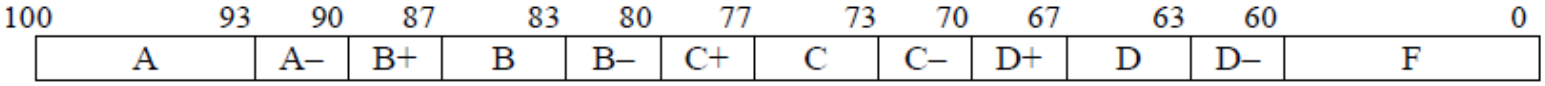
- Course page
  - https://seattleu.instructure.com/courses/1573222
- Contact info
  - mishraa@seattleu.edu
- Meeting times
  - Lectures: Mon, Wed, Fri 9:20 AM-10:45 AM
  - Location: PIGT 202

Prerequisites and Syllabus

- C or better in CPSC 1430 (formerly CPSC 152)
  - Logarithms and exponents
  - Programming in a high-level object-oriented language (C++, Java)
- Recommended Textbook: Structured Computer Organization, 6th edition. Andrew S. Tanenbaum and Todd Austin.
- Alternatives: 5th edition of the same book.
- Course requirements
  - 5 to 7 homework assignments (50%)
  - Midterm exam (20%) **[Tentatively scheduled for Nov 3 in class]**
  - Final exam (30%) **[December 5, 10 AM to 11:50 AM]**
- Attendance
  - Make an entry in the signup sheet at the beginning of every class. Attendance is not a direct component of your course grade, but may impact the grade if you are near a boundary.

Grading and Late Submission Policy

- Grade scale:



- The course grading scale may be curved but only to reduce the requirement to receive a particular grade
- Late submission:
  - Total of 3 late days for the homework assignments
    - Assuming academic calendar permits
    - Email me when you use late day(s), stating how many you used and the balance left
  - After late days are over, no late submissions accepted (except emergency cases)

Course Organization: Misc

- Office hours:
  - Monday and Friday 10:45 AM to 12:30 PM, Wednesday 10:45 AM to 12:45 PM
  - Or by appointment
- Office location: Engineering 507
- Email: mishraa@seattleu.edu

Honor Code

- Discuss the assignments with classmates
  - You may formulate the solutions together
  - But everyone should write/code their own solutions

- Violation of the honor code includes:
  - "Borrowing" write-up
  - "Borrowing" code from someone
  - Giving code to someone (even next year)
  - Copying code/answers from anyone (including the Internet)
  - Hiring someone to solve your assignments

Cell Phone and Laptop Policy

- Please put Cell phones on silent alert

- Please use Laptops for course use only
  - Taking notes
  - Refrain from email, browsing, Facebook, Twitter, etc.

Acknowledgments

- The lecture notes for this course contains material adapted from slide decks originally created by:
  - Linda Null and Julie Lobur (authors of The Essential of Computer Organization and Architecture)
  - Andrew S. Tanenbaum and Todd Austin (authors of Structured Computer Organization)
  - Umakishore Ramachandran and William D. Leahy Jr. (authors of Computer Systems: An Integrated Approach to Architecture and Operating Systems)
  - The instructors of EECS 370 at University of Michigan.
  - Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne (authors of Operating System Concepts Essentials)
  - Prof. Eric Larson, Computer Science faculty at the Seattle University
- A few definitions are derived from Wikipedia pages.
- Material from other sources will be cited.

Course Overview

- Data Representation, Digital Logic
- Assembly Language
  - Uses ANNA (a simple toy assembly language)
- Instruction Set Architecture, Microarchitecture
  - Uses ANNA
- Memory Organization, I/O
- Performance, Parallel Computer Architectures

Why Study Computer Organization?

- Gain an understanding of the underlying implementation of code
  - pointers, memory usage, code constructs
- Design better programs
  - Better aware of the performance aspects
  - Better equipped to write system software such as compilers, operating systems, and device drivers.
- Learn important computer science concepts.
  - Caching, pipelining, and parallelism all have applicability throughout computer science.
- Understand time, space, and cost tradeoffs.
  - hardware or software decisions?

Outline

- Course Overview
- **Computers and Programs**
- Computer Level Hierarchy
- von Neumann Model

What is a Computer?

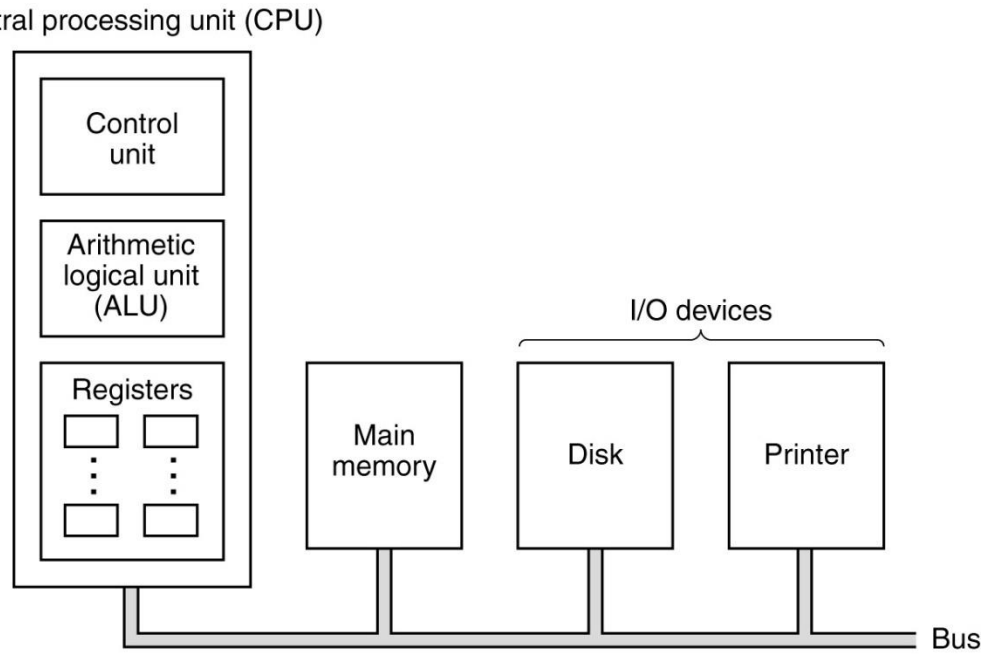
- A machine that can do work for people by carrying out instructions given to it.

- An electronic device for storing and processing data, typically in binary form, according to instructions given to it in a variable program.

Computer Components

At the most basic level, a **computer** consists of:

- A **processor** to interpret and execute programs.
- A **memory** to store both data and programs.
- An **input/output** mechanism for transferring data to and from the outside world.



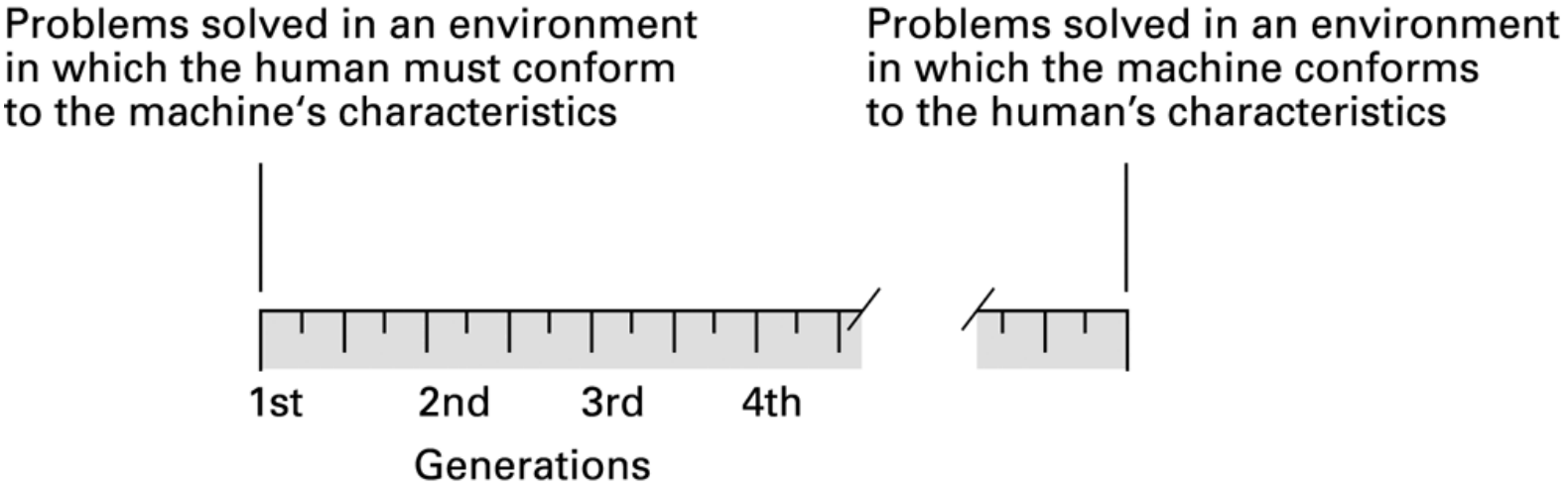
What is a Program?

- From the computer system point-of-view, a **program** is a list of ordered machine instructions.
- To execute the program, simply execute the instructions in order starting with the first one.
- What do these machine instructions do?
  - Perform some calculation (such as add)
  - Load / store values from memory
  - Get a value from an input device
  - Display a value on an output device
  - Jump to a different part of the program

Generations of Programming Languages

Problem:

- Computers like to speak in bits (binary)
- Humans like to speak in natural language (English, Spanish, etc.)



Generations of Programming Languages

- 1st generation: Machine Language
  - simply ones and zeros
  - understood by the computer
  - hard to understand by humans
- 2nd generation: Assembly Language
  - human-readable form of machine language
  - easy translation to machine language
  - very machine dependent
- 3rd generation: High-level Languages
  - common programming constructs are provided
  - machine independent (for the most part)
  - strict syntax and rules
  - compilers convert to assembly
  - Reduced software development time

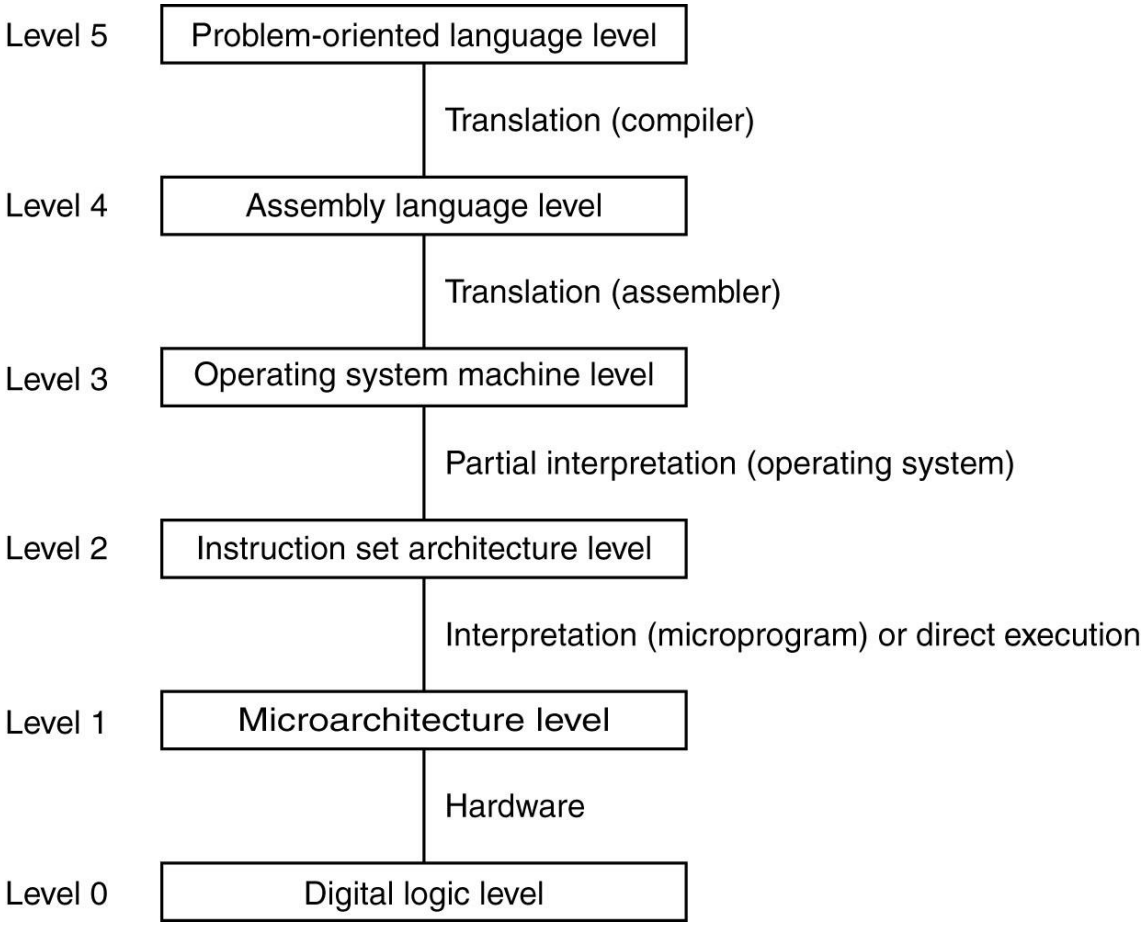
Outline

- Course Overview
- Computers and Programs
- **Computer Level Hierarchy**
- von Neumann Model

Computer Level Hierarchy

- A *hierarchical design* divides a computer system into manageable layers.
- Each layer can be implemented without intimate knowledge of the other layers.
- Each layer is an *abstraction* of the level below it.
- Each layer executes their own particular instructions, calling upon lower layers to perform tasks as required.
- Computer circuits ultimately carry out the work.

Computer Level Hierarchy



Computer Level Hierarchy

- Level 5: Problem-oriented language level
  - The level with which we write programs in languages such as C++, Python, and Java.

- Level 5 to Level 4: Assembly language level
  - A compiler converts source code into assembly code.
  - Even "interpreted" languages are compiled into assembly code.
    - Example: Java bytecodes

Computer Level Hierarchy

- Level 4: Assembly language level
  - Assembly language is a human readable form of machine language.
    - Difficult (time-consuming, error-prone) to program.
    - Most assembly language is produced by a compiler.

- Level 4 to Level 3: Operating system machine level
  - An assembler will convert the assembly language code into machine language.

Computer Level Hierarchy

- Level 3: Operating system machine level
  - OS controls executing processes, manages memory, and protects system resources.

- Level 3 to Level 2: Instruction set architecture level
  - Inserts necessary system library code.
  - Most assembly language instructions pass through Level 3 without modification.

Computer Level Hierarchy

- Level 2: Instruction set architecture level
  - Consists of machine instructions that are particular to the architecture of the machine.
  - Serves as the input to the microprocessor.

- Level 2 to Level 1: Microarchitectural level
  - Each instruction is translated into set of control signals that directs each component in the processor.
  - Control can also be implemented using a microprogram using a language internal to the processor.

Computer Level Hierarchy

- Level 1: Microarchitectural level
  - Consists of the various components inside a CPU:
    - Memory (called registers)
    - Arithmetic-logic unit (ALU)
    - Datapath: wires that connect the various components

- Level 1 to Level 0: Digital logic level
  - These CPU components are made of basic hardware components called gates.

Computer Level Hierarchy

- Level 0: Digital logic level
  - Consists of basic digital logic components such as AND, OR, and NOT gates.

- Level 0 to Level -1 and below?
  - These gates are actually made of transistors.
  - Transistors are constructed using semiconductors such as silicon.
  - Both of these topics are out-of-scope for this course.

Equivalence of Hardware and Software

- Anything that can be done with software can also be done with hardware.
  - Any operation can be built directly into hardware.
- Anything that can be done with hardware can also be done with software.
  - Any instruction executed by hardware can be simulated in software.
- Need to consider speed, cost, reliability, frequency tradeoffs.
  - Hardware is almost always faster and more expensive.

Outline

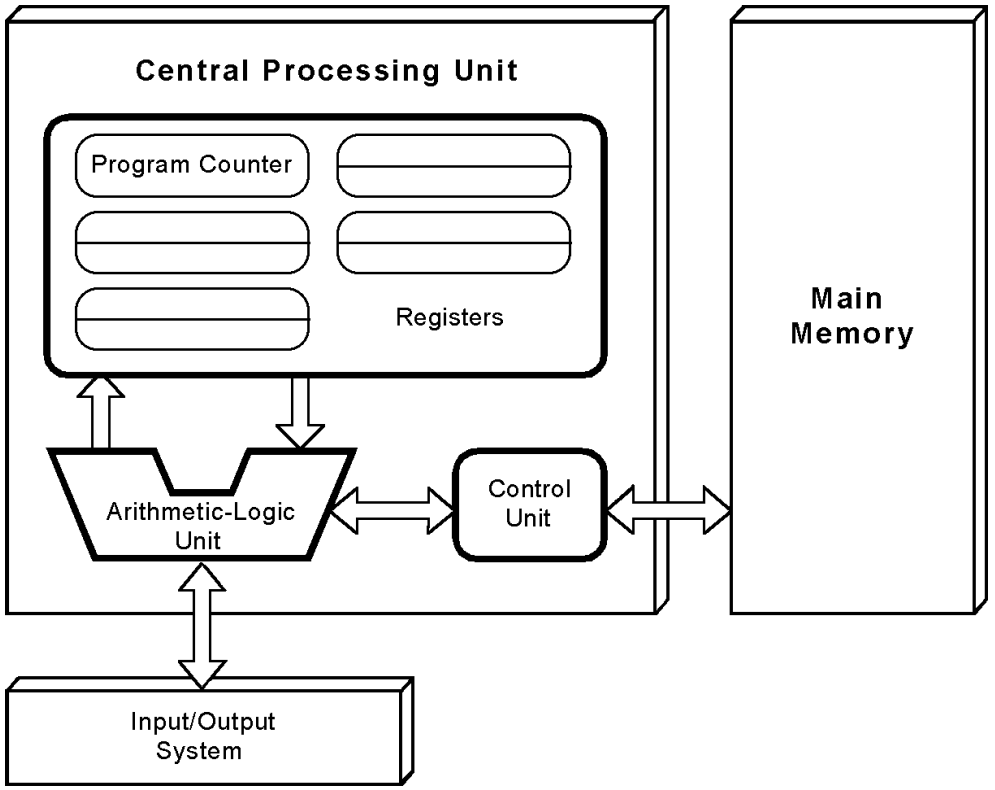
- Course Overview
- Computers and Programs
- Computer Level Hierarchy
- **von Neumann Model**

von Neumann Model

- Named after John von Neumann, a famous mathematician who was a pioneer in this model.
  - Model first used by EDSAC in 1949.
  - Still in use today.
- von Neumann computers have five basic components:
  - A central processing unit (CPU)
  - A main memory system
  - A control unit
  - Input equipment
  - Output equipment

Von Neumann Architectures

- *Registers:* Memory inside the chip used to hold results temporarily.
- *Program Counter:* A special register that keeps track of where you are in the program.
- *Main Memory:* Stores the program and the data used / generated by the program.
- *Control Unit:* Responsible for interfacing with memory and decoding instructions.
- *Arithmetic-Logic Unit:* Performs simple operations on data.

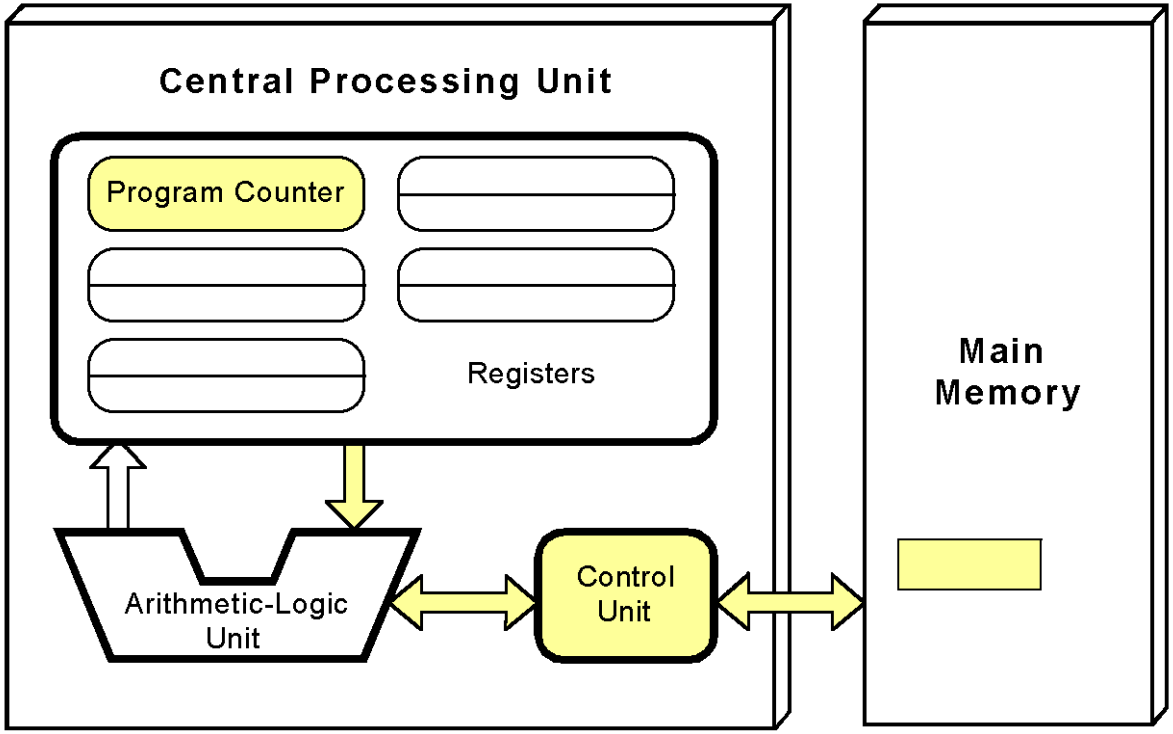


Instruction Execution

1. *Fetch* the next instruction from memory.
2. Update the program counter to point to the following instruction.
3. Determine the type (*decode*) of instruction just fetched.
4. If the instruction needs data from memory, determine where that data is and store it in a register.
5. *Execute* the instruction.
6. Store the results of the instruction in a register and/or memory.
7. Go to step 1 for the next instruction.

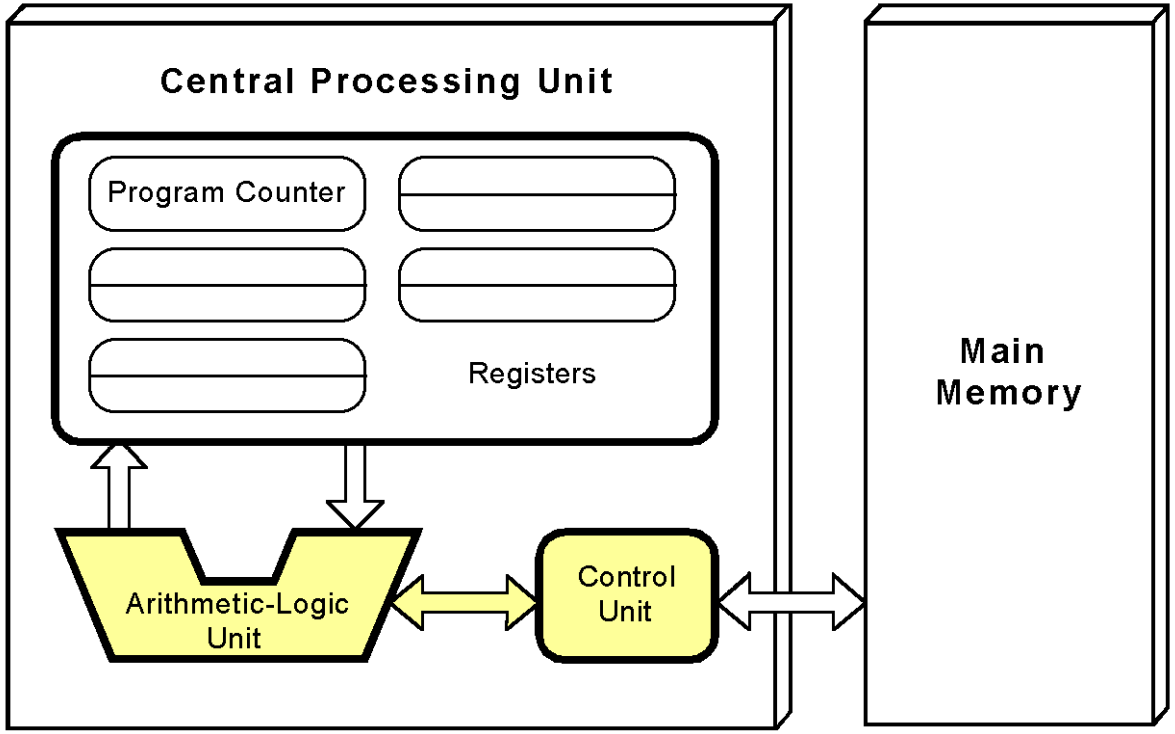
Instruction Execution: Fetch

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



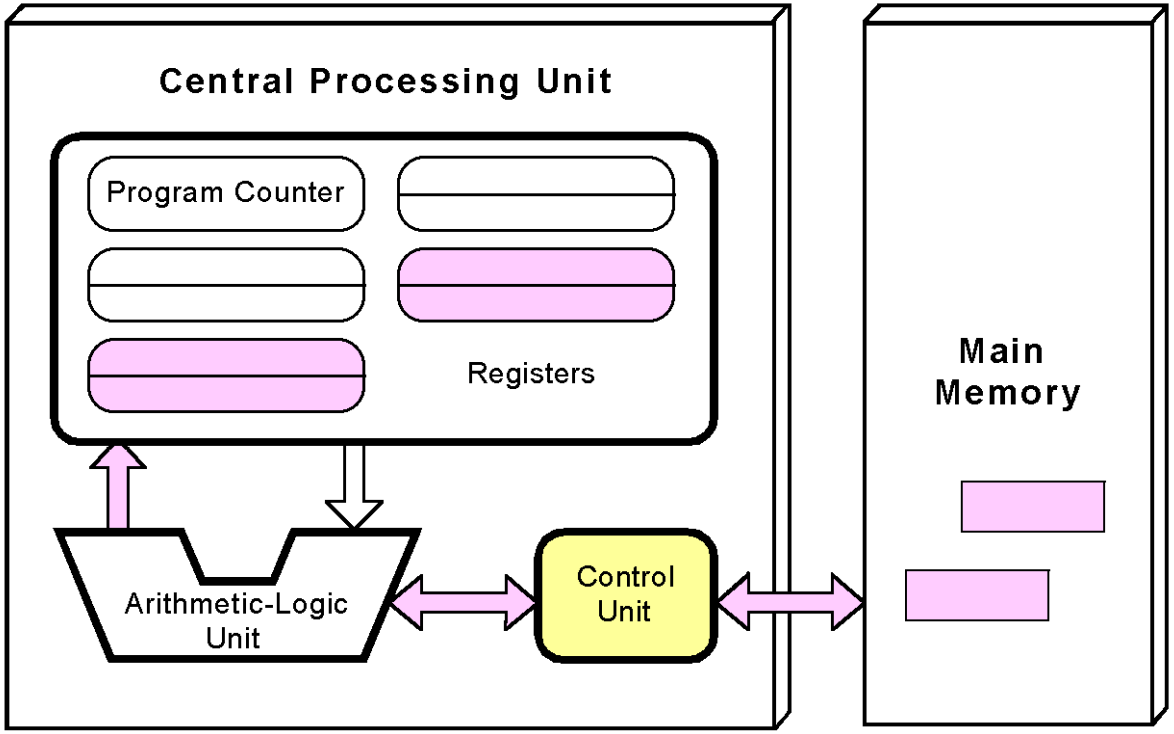
Instruction Execution: Decode

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



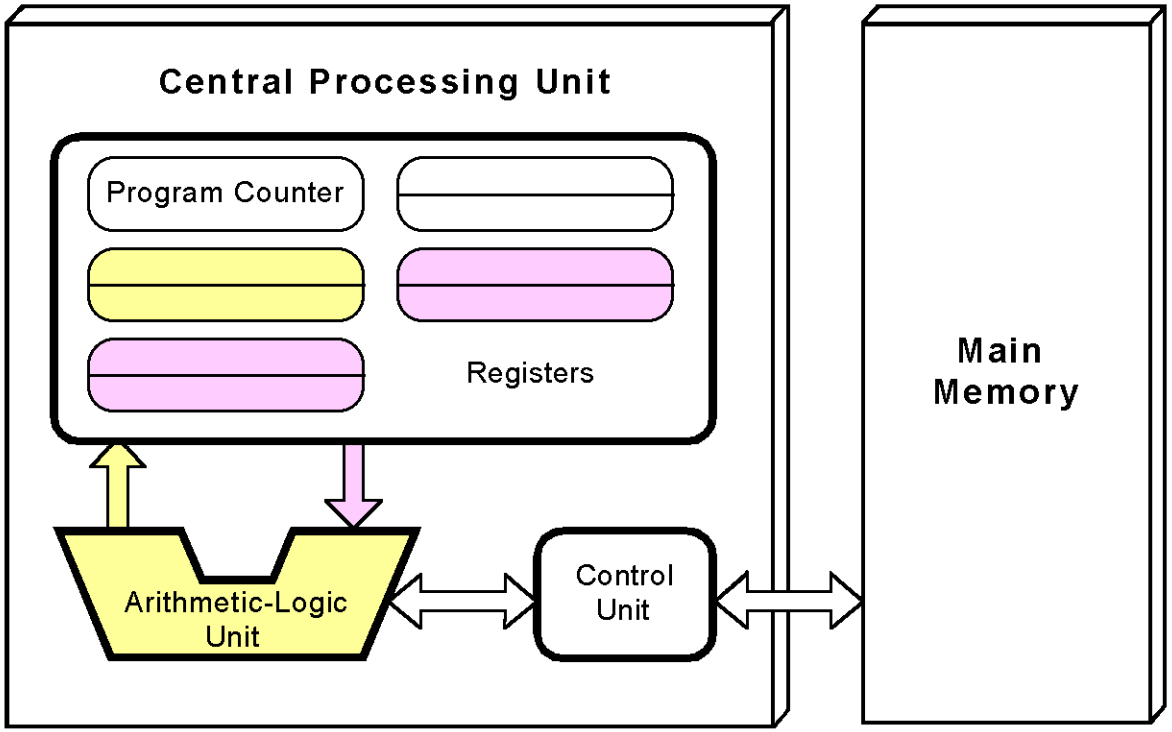
Instruction Execution: Gather Data

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



Instruction Execution: Execute

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



Multi Core Processors

- A single chip has multiple processing units
  - Dual core has 2 processing units
  - Quad core has 4 processing units
- Allows multiple programs to be executed at once.
- Programs can also take advantage of multiple processing units.
  - Software must be specifically designed to do this.
  - Software is hard to write
- Faster than having multiple processors (each on a separate chip)
  - Processing elements are “closer”
  - Communication off-chip is slow

Processors are Everywhere!

Type	Examples
Disposable computers	RFID
Microcontrollers	Watches, cars, appliances
Portable computers	Cell phones, tablets
Video game systems	Wii U, PS4, XBOX One
Personal computers	Desktop and laptop computers
Specialized processors	Graphics cards
Server	Network server
Collection of Workstations	Server farms, supercomputers
Mainframe	Batch processing

Thank You!