Computer Architecture

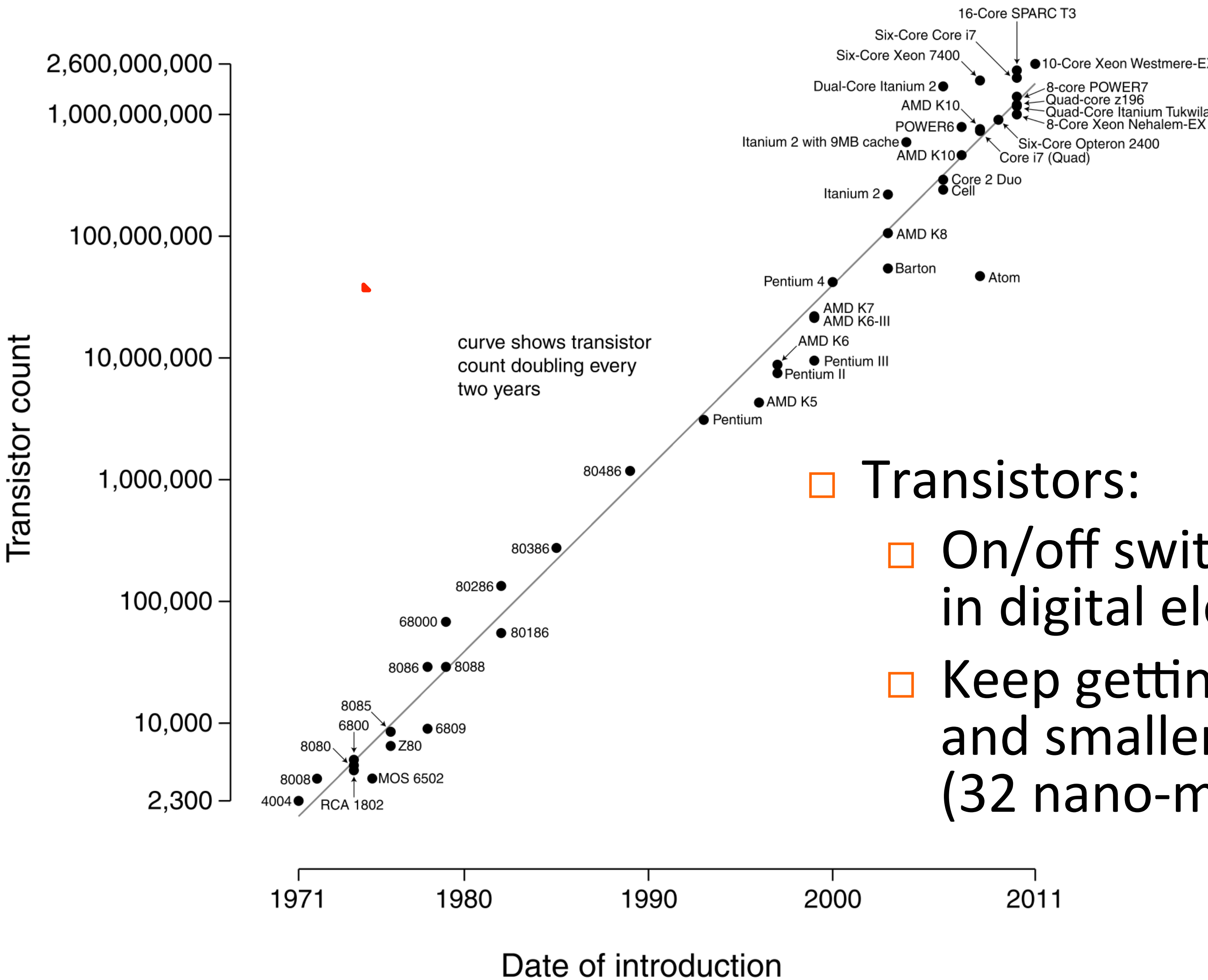
ICT 1019Y Week 1

Moore's Law, Computer Operation, and Number Systems

Historical Development

- **The Fourth Generation:** VLSI Computers (1980 - ????)
  - Very large scale integrated circuits (VLSI) have more than 10,000 components per chip
  - Build microprocessors on a single chip
    - 4-bit Intel 4004
    - 8-bit Intel 8008
    - 16-bit Intel 8086
    - 32-bit Intel 80386
    - …
- Transistors are getting smaller and smaller
  - **How far can this go?**

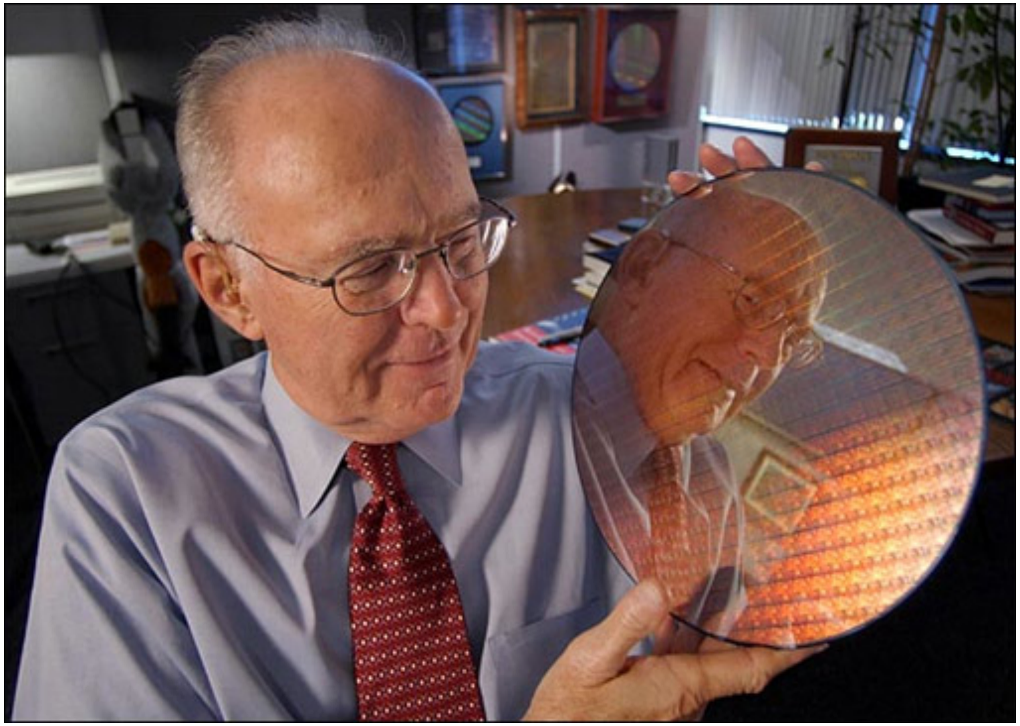
Microprocessor Transistor Counts 1971-2011 & Moore's Law



- Transistors:
  - On/off switches used in digital electronics
  - Keep getting smaller and smaller!
    (32 nano-meters)

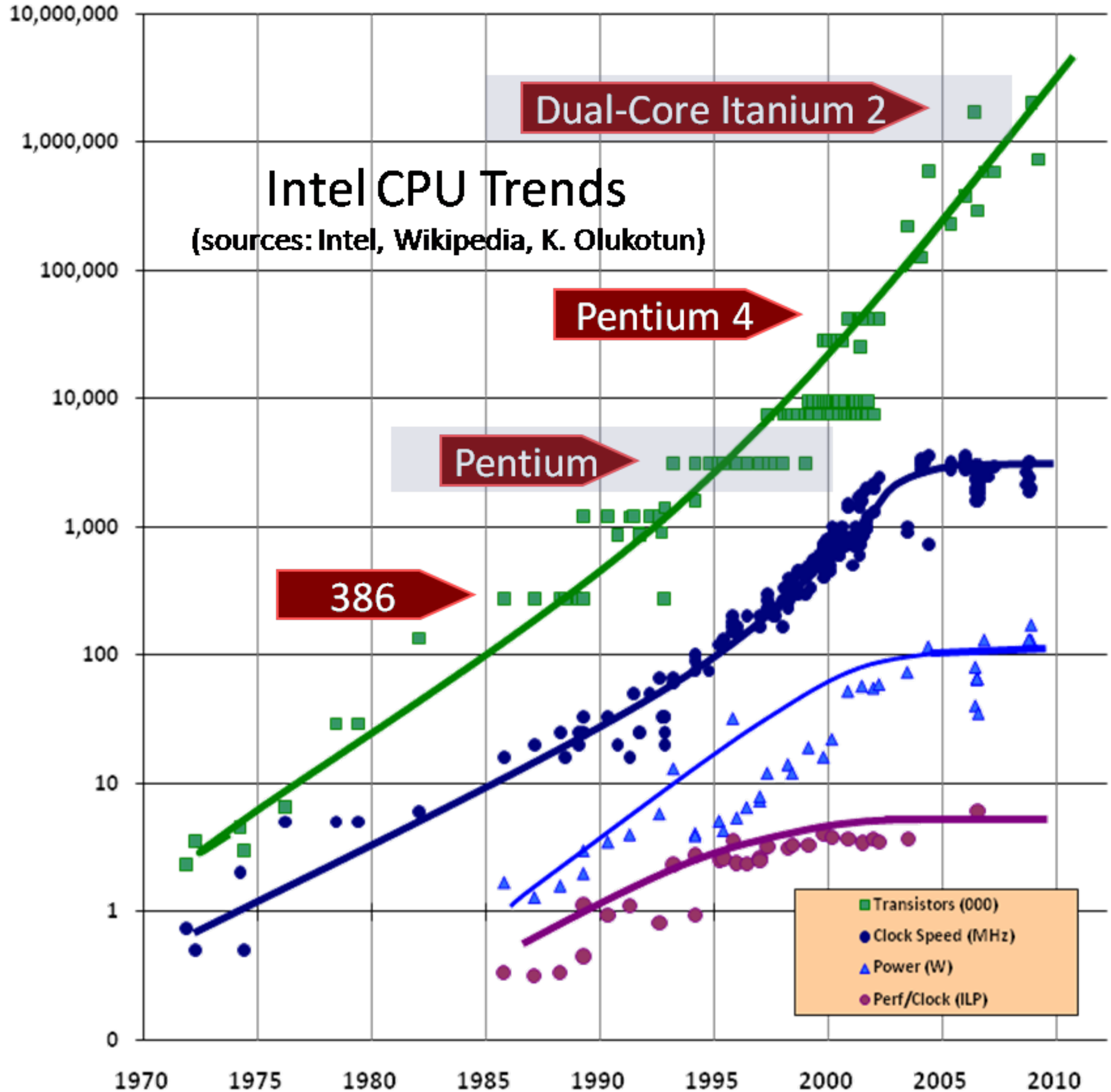
Moore's Law

- Gordon E. Moore
- Co-founder, Intel
- Proposed back in 1965
- Not a physical law!
  - An **observation of trends** in the semiconductor industry…



The "Law": The number of transistors available *at a given cost* doubles approximately every two years

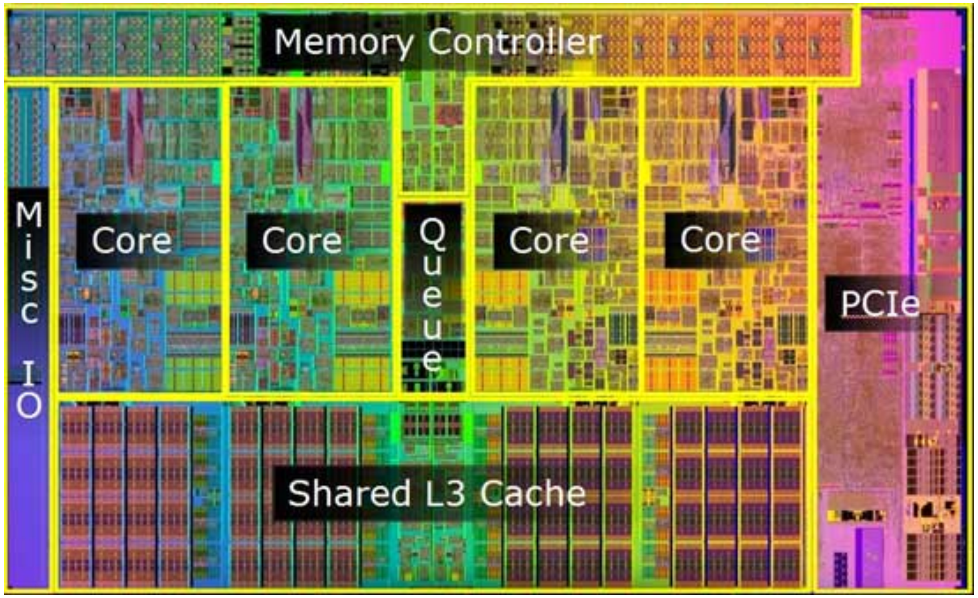
Where does Moore's Law end?



http://jai-on-asp.blogspot.com/2010/05/parallel-computing-and-net.html

What to do with a billion+ transistors?

- Billions of transistors available today
  - More than we need to build a *single* fast and power-efficient ("cool running") processor
- Let's build **many processors** and put them on the same chip!
- How can we keep all the processors ("cores") busy doing productive work?

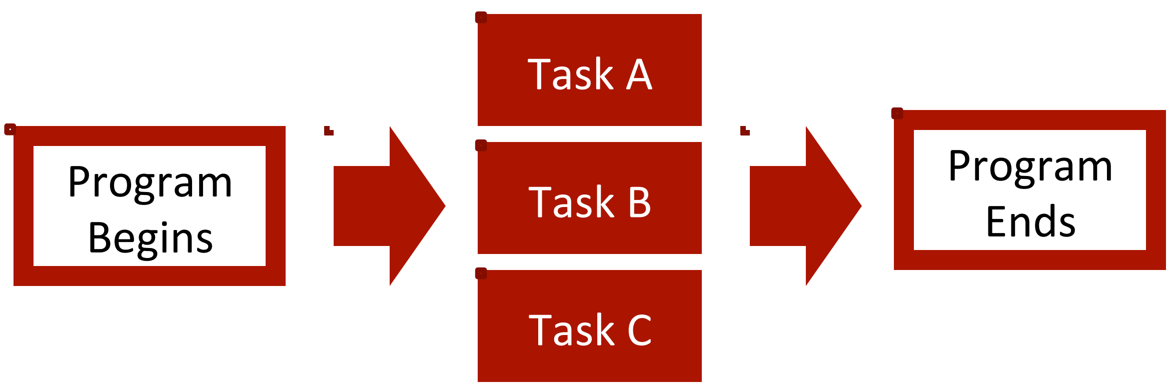


What is Parallel Programming?

- Writing code with multiple “threads” of execution.
  - These threads can be assigned to different cores

- *Sequential execution* (what we have been doing so far) means that each task is executed one after the other

- *Parallel execution* means that tasks are done at the same time



Computer Operation

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

Configuring the ENIAC to solve a "simple" problem required many days of work by skilled technicians

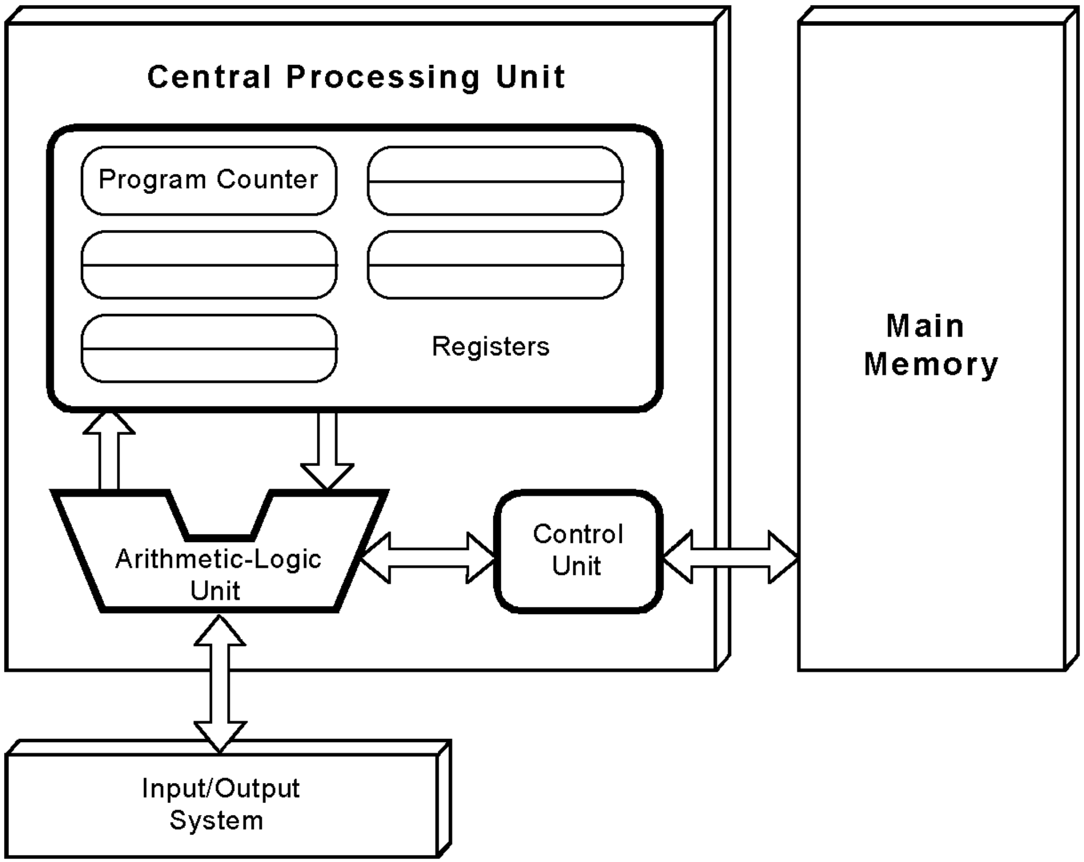
The von Neumann Model

- Inventors of the ENIAC (Mauchley and Eckert) conceived of a computer that could store instructions in memory
  - No need to re-wire the machine each time!
- First to publish this idea: John von Neumann
  - Contemporary of Mauchley and Eckert, who had to keep their ideas top secret (military)
- Stored-program computers have become known as *von Neumann Architecture* systems

The von Neumann Model

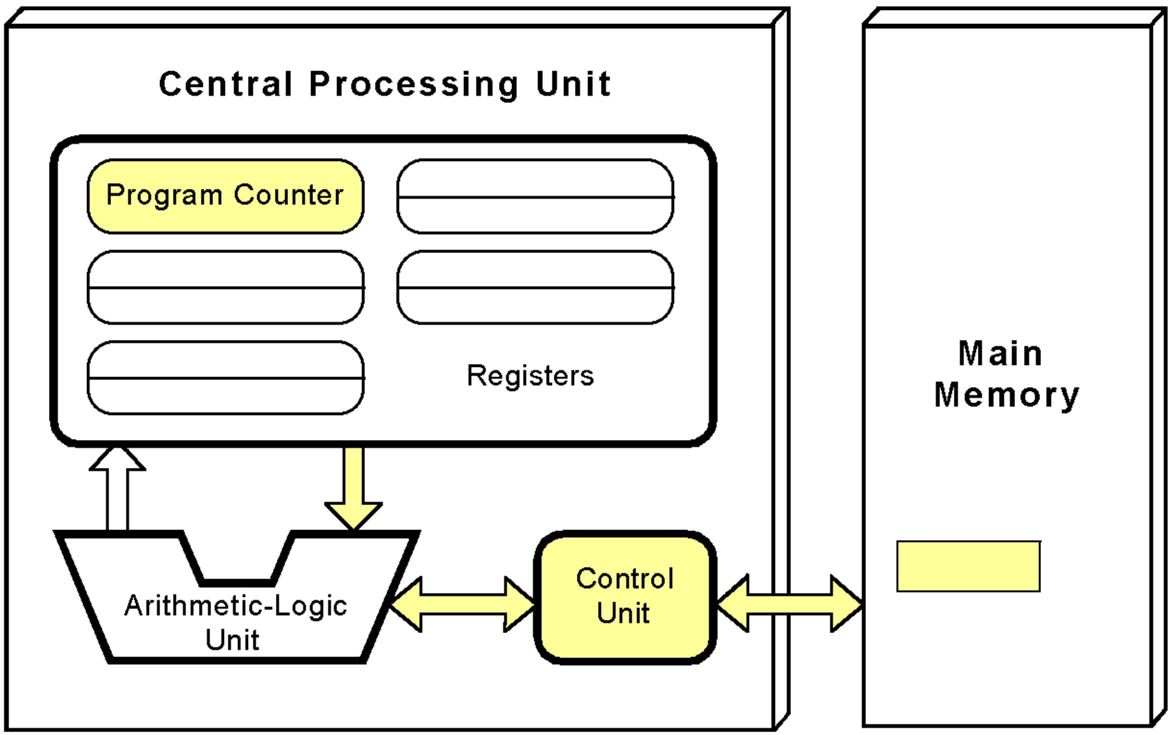
- Today's stored-program computers have the following characteristics:
  - Three hardware systems:
    - A **central processing unit** (CPU) to interpret programs
    - A **main memory** system to store programs & data
    - An **I/O** system to transfer data to/from the outside world
  - The capacity 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



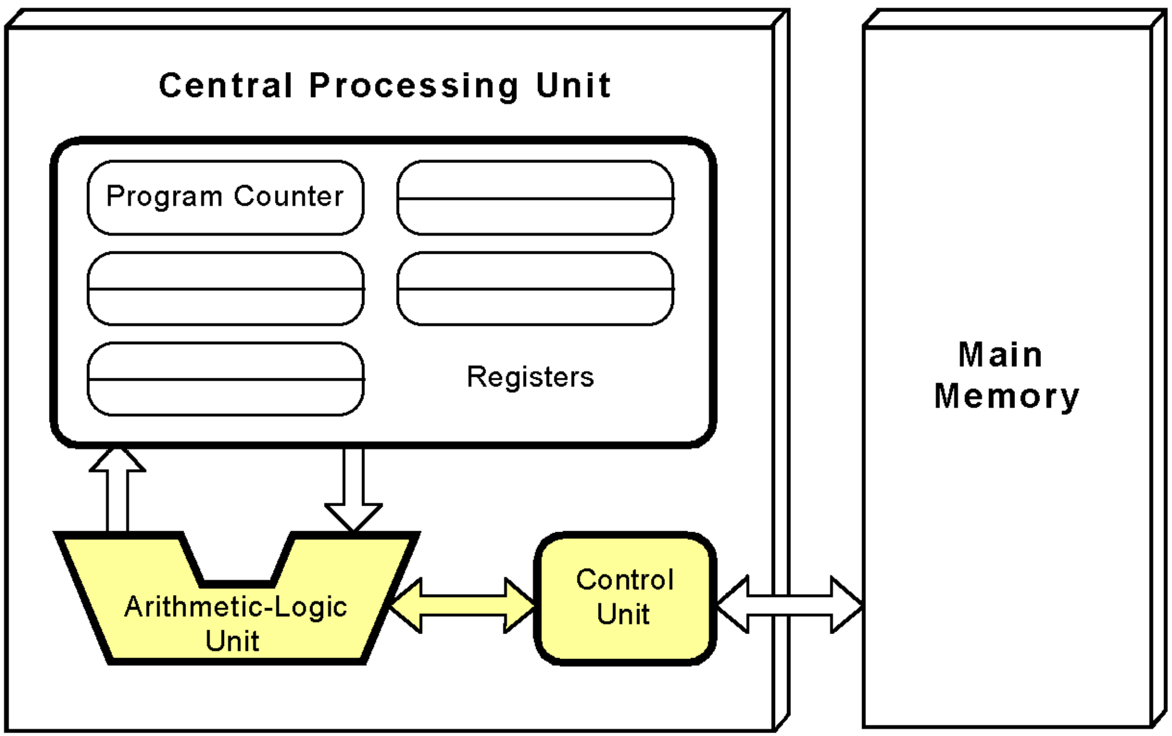
- This is a general depiction of a von Neumann system
- These computers employ a **fetch-decode-execute** cycle to run programs as follows . . .

The von Neumann Model



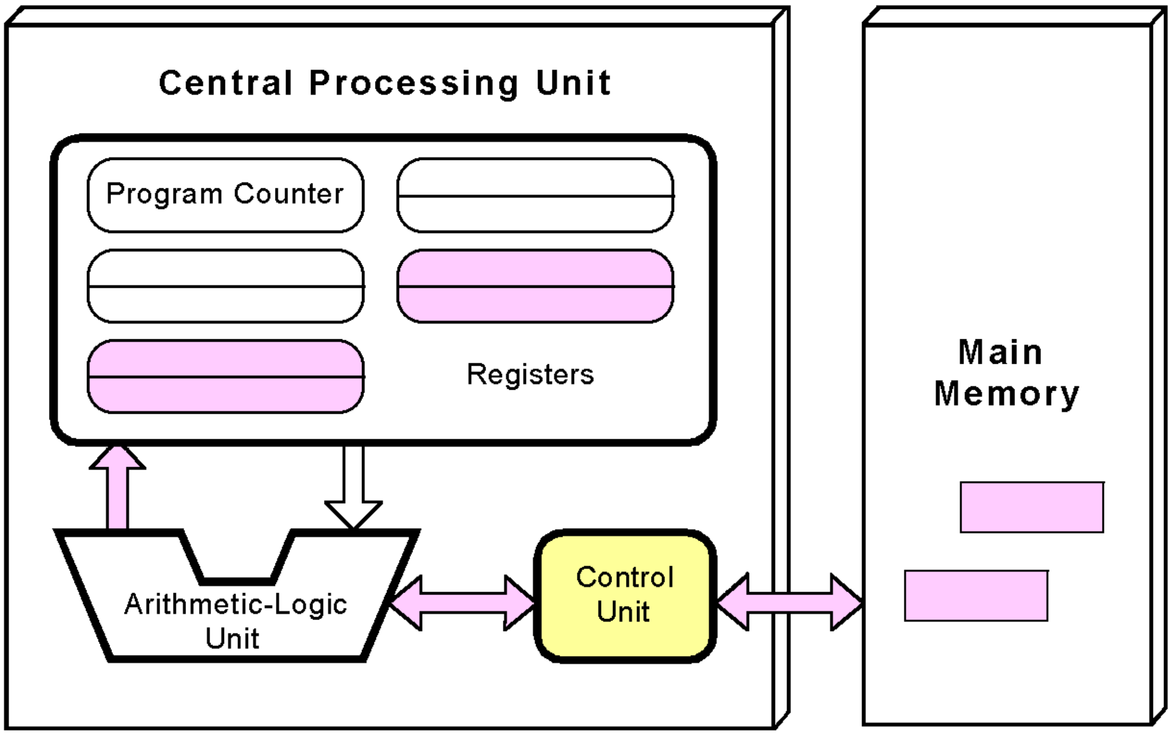
- The control unit **fetches** the next instruction from memory
- Which instruction?
  - Use the **program counter**

The von Neumann Model



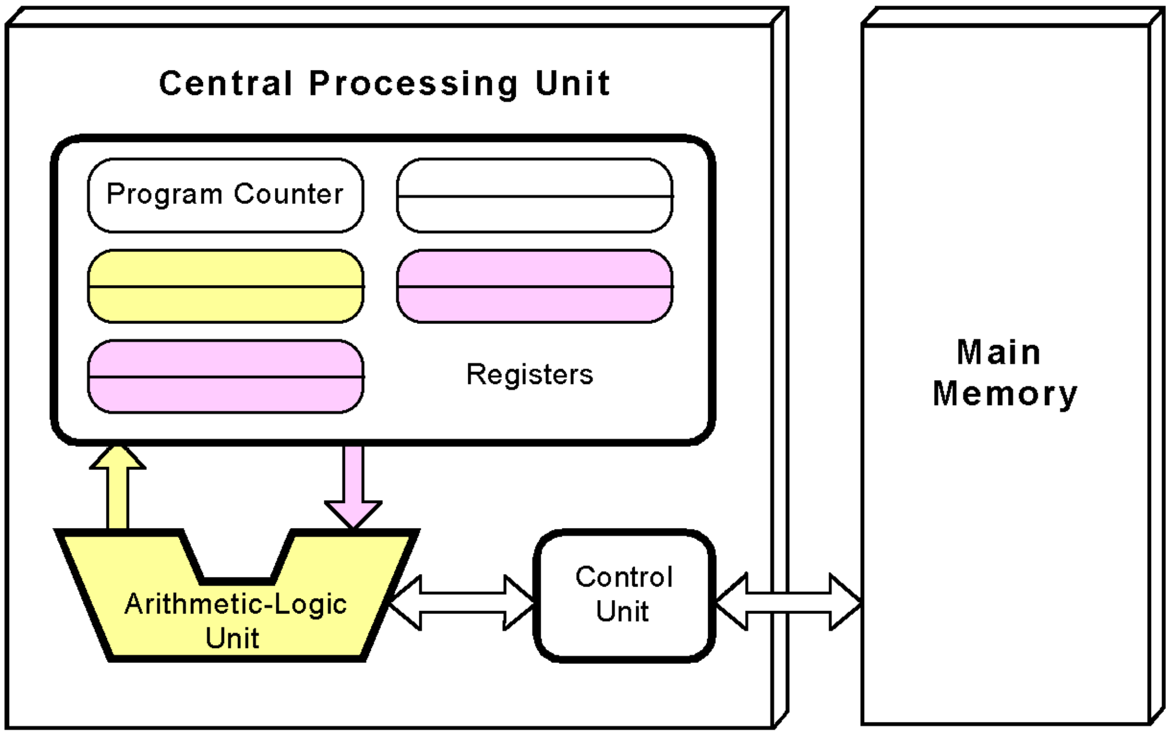
- The instruction is **decoded** into a language that the ALU can understand
  - Add?
  - Subtract?
  - Multiply?
  - Compare?
  - etc…

The von Neumann Model



- Any **data operands** required to execute the instruction are fetched from memory and placed into registers within the CPU
- Operands?
  - X = 3+5
  - *3 and 5 are operands…*

The von Neumann Model



- The ALU **executes** the instruction
- Results are placed back in memory (or temporary spots called **registers**)

Non-von Neumann Models

- Conventional stored-program computers have undergone many incremental improvements over the years
  - Specialized buses
  - Floating-point units
  - Cache memories
  - …
- Further improvements in computational power requires departure from the classic von Neumann architecture
  - One approach: Adding processors

Multi-Processor is an Old Idea!

- Late 1960s, dual processors used to increase computational throughput
- 1970s – Supercomputer systems introduced with 32 processors
- 1980s – Supercomputer systems built with 1,000 processors
- 1999 – IBM Blue Gene system with 1 million+ processors
- What is “new” is multiple processors in *your* PC

Measures of Capacity and Speed

- Kilo- (K) = 1 thousand = 10^3 and 2^{10}
- Mega- (M) = 1 million = 10^6 and 2^{20}
- Giga- (G) = 1 billion = 10^9 and 2^{30}
- Tera- (T) = 1 trillion = 10^{12} and 2^{40}
- Peta- (P) = 1 quadrillion = 10^{15} and 2^{50}
- Exa- (E) = 1 quintillion = 10^{18} and 2^{60}
- Zetta- (Z) = 1 sextillion = 10^{21} and 2^{70}
- Yotta- (Y) = 1 septillion = 10^{24} and 2^{80}

Whether a metric refers to a power of ten or a power of two *typically* depends upon what is being measured.

Measures of Capacity and Speed

- Hertz = clock cycles per second (frequency)
  - 1MHz = 1,000,000Hz
  - Processor speeds are measured in MHz or GHz.
- Byte = a unit of storage
  - 1KB = 2^{10} = 1024 Bytes
  - 1MB = 2^{20} = 1,048,576 Bytes
  - 1GB = 2^{30} = 1,099,511,627,776 Bytes
  - Main memory (RAM) is measured in GB
  - Disk storage is measured in GB for small systems, TB (2^{40}) for large systems.

Objectives

- Digital computers
  - How do we represent numbers and characters?
  - How do we convert between human and computer representations?
    - i.e. convert between base 10 and 2
  - Why do errors occur in computation?
    - Overflow?
    - Truncation?
  - How do we detect and correct errors?

Basics

- A **bit** is the most basic unit of information in a computer
  - It is a state of “on” or “off” in a digital circuit
  - Sometimes these states are “high” or “low” voltage instead of “on” or “off”

0

1

Basics

- A **byte** is a group of **eight bits**
  - A byte is the **smallest possible addressable unit** of computer storage
  - **Addressable?**
    - A particular byte can be retrieved according to its location in memory

01101001

Basics

- A **word** is a contiguous **group of bytes**
  - Words can be any number of bits or bytes
  - Word sizes of 16, 32, or 64 bits are most common
  - In a word-addressable system, a word is the smallest addressable unit of storage

01101001 11001010 01110001 01000111

Positional Numbering Systems

- Binary (base 2) numbers
  - Each position represents a power of 2
  - Two digits: 0, 1
- Decimal (base 10) numbers
  - Each position represents a power of 10
  - Ten digits: 0 - 9
- Hexadecimal (base 16) numbers
  - Each position represents a power of 16
  - Sixteen digits: 0-9 and A-F

Positional Numbering Systems

- The decimal number 947 in powers of 10 is:

$$9 \times 10^2 + 4 \times 10^1 + 7 \times 10^0$$

- The decimal number 5836.47 in powers of 10 is:

$$5 \times 10^3 + 8 \times 10^2 + 3 \times 10^1 + 6 \times 10^0 + 4 \times 10^{-1} + 7 \times 10^{-2}$$

Positional Numbering Systems

- The binary number 11001 in powers of 2 is:

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 16 + 8 + 0 + 0 + 1 = 25$$

- When the radix of a number is something other than 10, the **base is denoted by a subscript**.
  - Sometimes, the subscript 10 is added for emphasis:
  - $11001_2 = 25_{10}$

Positional Numbering Systems

- This system works for any base (aka *radix*) you want
  - Base 3, Base 19, etc…
- Any **integer** quantity can be represented **exactly** using any base
- **Why do computers use base 2?**
- **Why do (modern) humans use base 10?**
  - *Babylonians used base 60*
  - *Mayans used base 20*

Positional Numbering Systems

- Where do we use binary numbers beyond homework problems?
- Understanding operation of computer components
  - How big is the memory system?
  - How does the processor do arithmetic?
- Designing new processors
  - Instruction set architecture – the language of the machine
- Assembly programming
  - Particularly if you convert from assembly code to the binary executable by hand