Introduction to Computer Science

Overview of Discussion

- What is computer science?
  - What is a computer?
  - What can computers do?
  - How do computers solve problems?
  - What is computer science?
- Who invented computers?
  - Conceptual computers
  - Computing devices

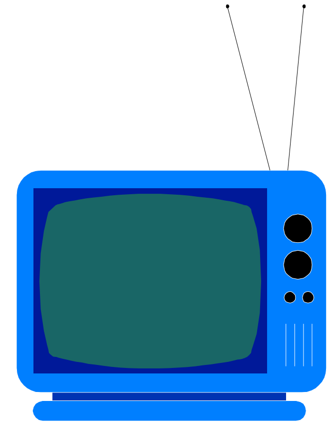
Learning Objectives

- Define and use terminology
  - Examples: computer, computer science, algorithm, specification, correctness, efficiency, von Neumann machine
- Distinguish between algorithms and non-algorithms
- Know something about the history of computers (up to 1950)

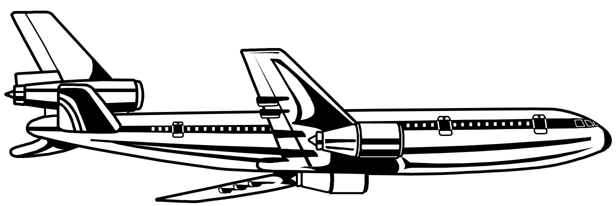
Which one is the computer?

Rock

Calculator



Television

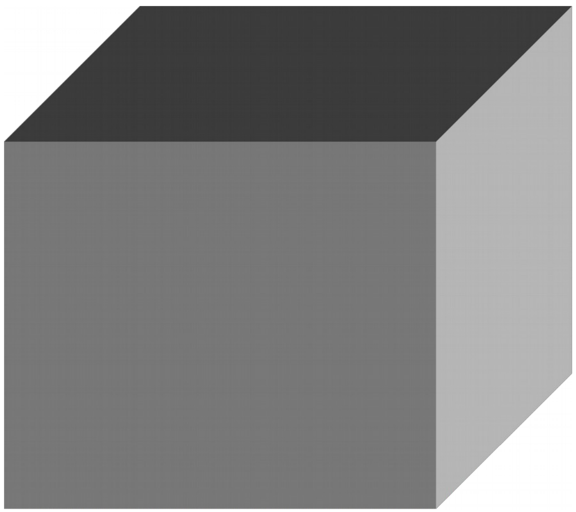


Modern Airplane

Washing Machine

Computer Workstation

Is it a Computer?



- What questions would you ask?
- What experiments would you run?

Is a rock a computer?

- Does not act or process
- Takes no input and produces no output

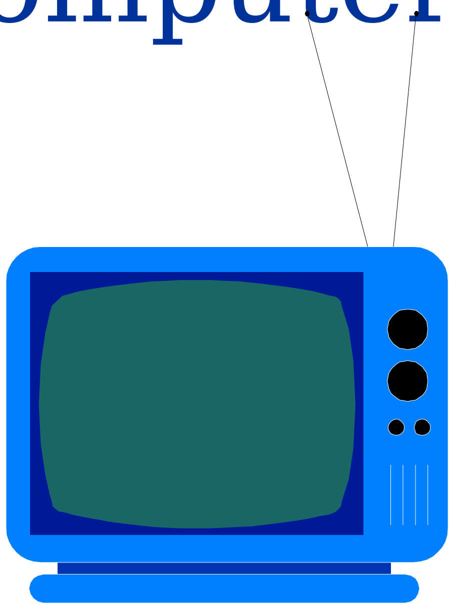
- Computers must be able to handle *input* and *output*

Is a washing machine a computer?

- Input: dirty clothes
- Output: clean clothes
- Does not handle information

- Computers input and output *information*

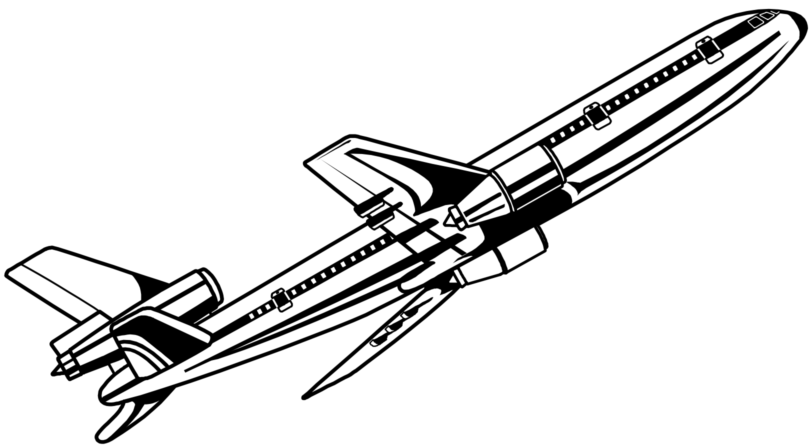
Is a **television set** a computer?



- Input: information from cables or radio waves
- Output: information as sound and picture
- Does not process information

- Computers *process* information by computing new results and answering queries

Is a **modern airplane** a computer?



- Input: information from radio waves
- Output: manipulations to the airplane
- Can only handle specific information necessary for flight control

- Computers are *general purpose* because they can perform many different tasks

Is an **ordinary calculator** a computer?

- Input: numbers and mathematical operations
- Output: answer
- Handles any numeric task
- Cannot remember which buttons are pressed

- Computers are *programmable* so they can remember sequences of operations

Definition of a Computer

- a general purpose,
- programmable,
- information processor
- with input and output

How do computers solve problems?

- Humans deconstruct problems into small operations that a computer can carry out
  - Writing an *algorithm*
- Solve a problem by computer requires
  - State the problem clearly in a *problem statement*
  - Solve the problem with an *algorithm* that gives clear instructions
  - Use a *computing agent* to carry out the instructions

Solving the problem using an Algorithm

- Algorithm – a clear sequence of instructions for performing a task
  - a well-ordered sequence
  - of well-defined,
  - feasible operations
  - that takes finite time to carry out

Almost Algorithms

- To shampoo your hair
  1. Rinse
  2. Lather
  3. Repeat
- To set the time on the VCR
  1. Open the front panel
  2. Push the button
  3. Set the hours, then the minutes
- To write the Great American Novel
  1. Get paper and pencil
  2. Sit down
  3. Write word on paper
  4. If novel is great, quit. Otherwise, go back to step 3.

Necessity of artificial languages

- Problems with natural languages (like English)
  - Flexible
  - Often ambiguous
- Computers use artificial languages with precise meanings
  - mathematical equations, music notation, programming languages
- Programming languages define primitive operations computing agents understand

Who invented computers?

- Computer science has roots in two fields
  - Mathematics
    - Alan Turing and the Turing machine (1930s)
    - Developed theories with paper and pencil about how to perform computations by hand
  - Engineering
    - John von Neumann and the von Neumann machine (1940s)
    - Showed how to build physical computers out of electronic circuitry

Mathematical Roots

- Leibniz's Dream (1600s)
  - Can we find a universal language for mathematical algorithms that will let us describe and solve any problem?
    - Reduce all reasoning to a fixed set of basic rules
    - Determine truth or falsity of sentences by fixed rules for manipulating sentences
- George Boole (1800s)
  - Introduces binary notation of calculation
    - Computers use binary system for logic and arithmetic

More on Theory

- David Hilbert (1928)
  - Challenges the mathematical community to find an infallible, mechanical method for constructing and checking truth of mathematical statements
    - Interested in an algorithm
- Alonzo Church, Alan Turing, and Kurt Gödel construct arguments that there is no solution to Hilbert's Challenge
  - Turing builds a conceptual computer for his argument

The Turing Machine and the Church-Turing Thesis

- Turing Machine
  - Machine with a finite set of rules and an infinite amount of “scratch paper” for computation
    - No one has designed a physical computer that can do more than a Turing machine
  - Machine could not solve Hilbert’s problem
- Church-Turning Thesis
  - The Turing Machine captures what we mean by computational systems
  - Is as powerful an any other mechanical computing agent

Engineering Roots

- First step development of calculators
  - Abacus – developed 5000 years ago in the Middle East
  - Pascaline – first mechanical calculator using gears for calculation (1642)
  - Charles Babbage's Difference Engine – conceptual design that used hundreds of gears to compute mathematical functions (1820s)

Electronic Circuits

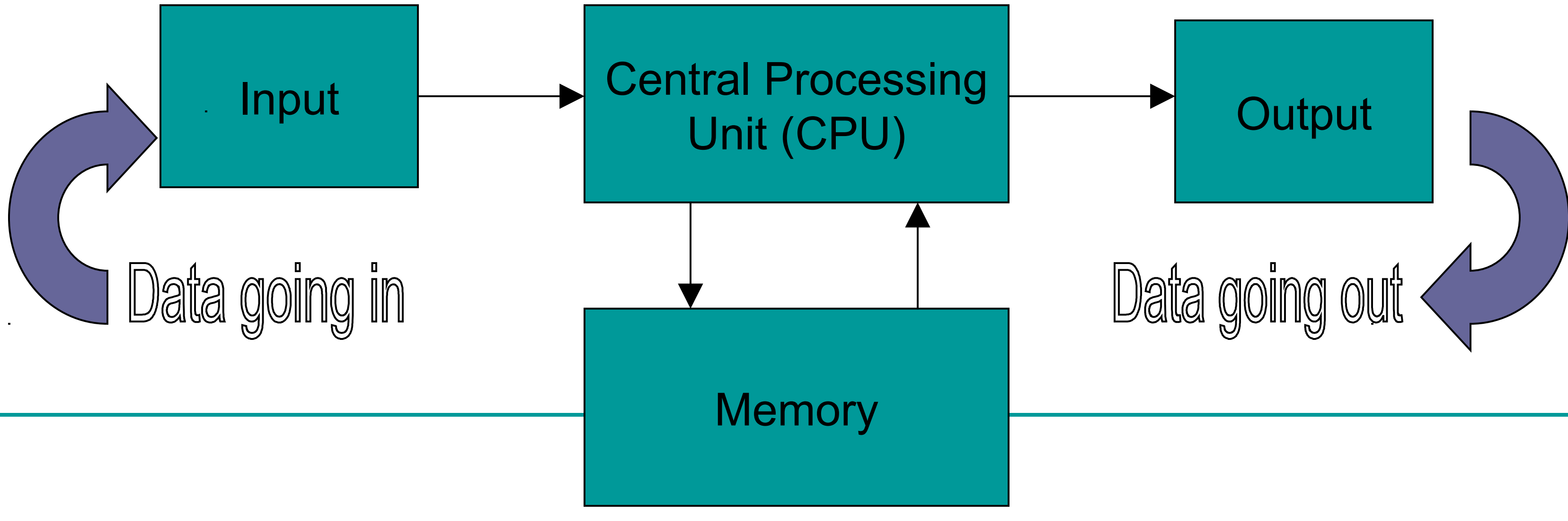
- Telegraph – uses electricity to convey letters and transmit information quickly (1844)
- Hollerith Tabulating Machine – Uses electricity and punch cards to calculate the US census (1890)
- Z2 – used circuitry to compute arithmetic operations (1930s)

Programmed Devices

- Jacquard Loom – weaves cloth using a pattern specified using punch cards (1801)
- The Analytic Engine – conceptual design for a machine consisting of a Mill, Store, Printer, and Readers
  - Led Ada Lovelace to define programming concepts such as the subroutine
- ENIAC – one of the first programmable electronic computers (1945)
  - Programmed by routing cables and flipping switches

von Neumann Machine

- Store programs in electronic memory along side the data (1943)
  - Move and manipulate a program like data
  - Enabled high-level programming languages



Machine Languages

- Only language computers directly understand
- “Natural language” of computer
- Defined by hardware design
  - Machine-dependent
- Generally consist of strings of numbers
  - Ultimately 0s and 1s
- Instruct computers to perform elementary operations
  - One at a time
- Cumbersome for humans
- Example:

```
+1300042774
+1400593419
+1200274027
```

Assembly Languages

- English-like abbreviations representing elementary computer operations
- Clearer to humans
- Incomprehensible to computers
  - Translator programs (assemblers)
    - Convert to machine language
- Example:

```
LOAD      BASEPAY
ADD       OVERPAY
STORE     GROSSPAY
```

High-level Languages

- Similar to everyday English, use common mathematical notations
- Single statements accomplish substantial tasks
  - Assembly language requires many instructions to accomplish simple tasks
- Translator programs (compilers)
  - Convert to assembly language
- Interpreter programs
  - Directly execute high-level language programs
- Example:

```
grossPay = basePay + overTimePay
```

Programming Approaches

- Structured programming (1960s)
  - Disciplined approach to writing programs
  - Clear, easy to test and debug, and easy to modify
  - Focus on what the program does
- Object Oriented programming
  - Object is an entity characterized by a *state* and a *behavior*
    - state is encoded in the computer program as *data*
    - behavior is encoded as methods

Objects

- Reusable software components that model real world items
- Meaningful software units
  - Date objects, time objects, paycheck objects, invoice objects, audio objects, video objects, file objects, record objects, etc.
  - Any noun can be represented as an object
- More understandable, better organized and easier to maintain than structured programming
- Favor modularity
  - Software reuse
    - Libraries