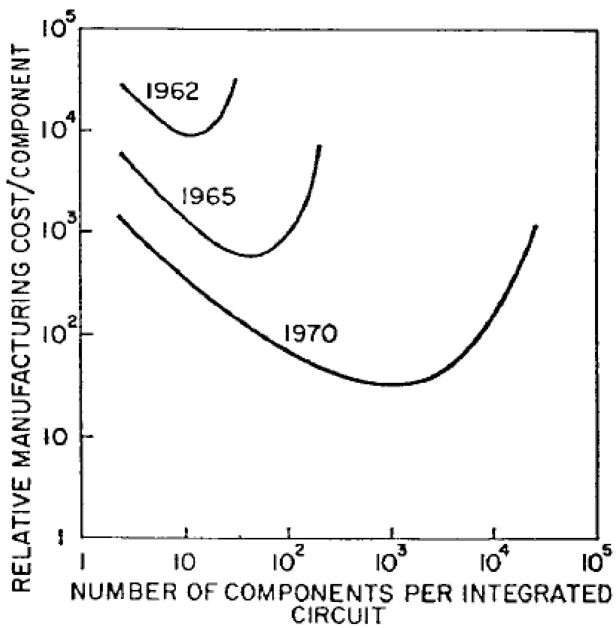
Lecture 2 – 1/21/11

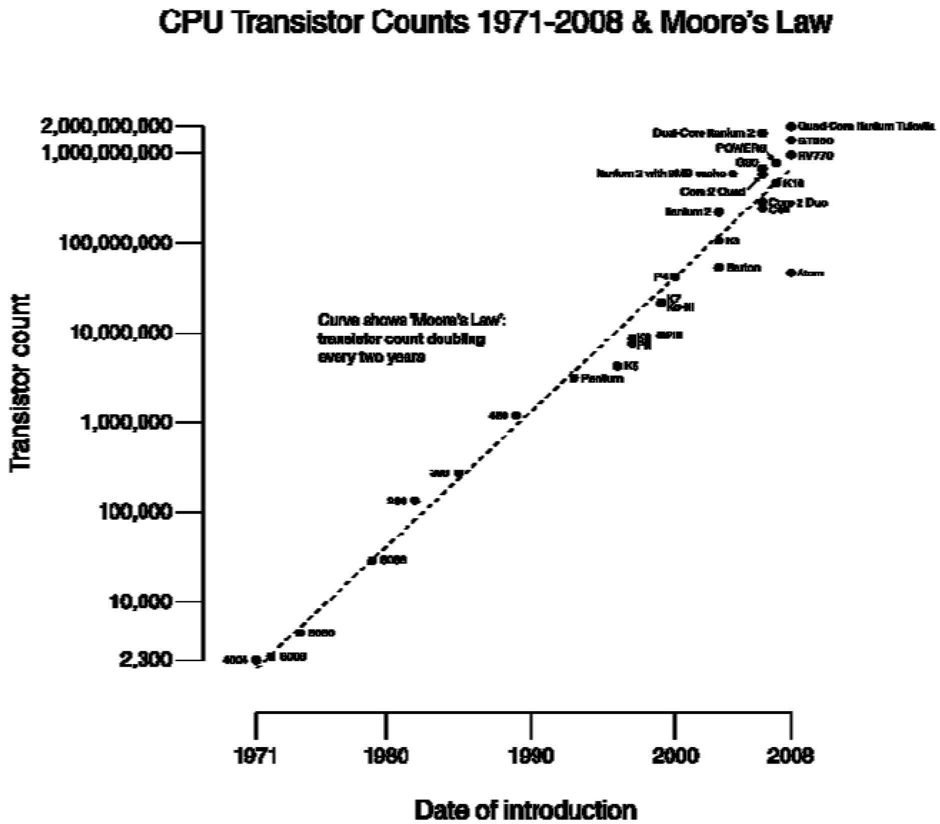
- Announcements
  - Let me know if you have not received Blackboard email notification
- Last Lecture
  - Grand tour of course
- Today's Lecture (P&P 1)
  - Moore's Law
  - Digital vs. analog
  - Realization
- Next lecture (P&P 3.1)
  - Transistor basics

Moore's Law (1965)

- Discrete components → Integrated circuits
  - Lower cost
  - Higher performance
  - Higher reliability
- Bathtub yield curve
- Density for min cost doubles every year
  - Projection to 1975
  - Later revised to every 2 years
- Industry roadmap



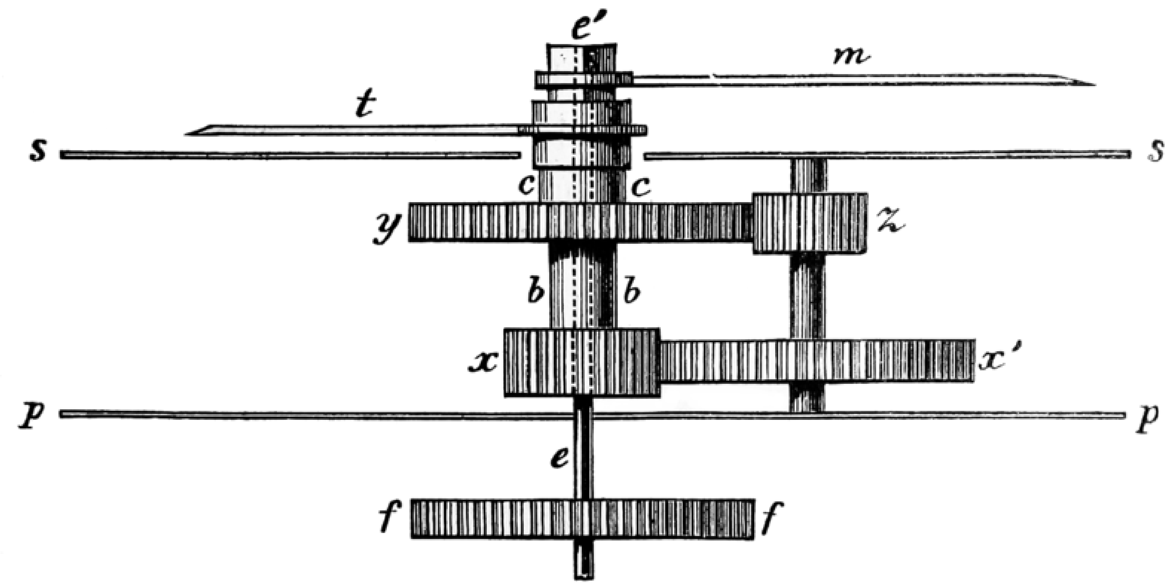
Moore's Law In Action



Digital vs. Analog Computing Systems

- Analog is a continuous measurement representing some physical quantity: weight, distance, voltage
  - Real world, difficult to be accurate, Specific to the task
  - Analog clock: angular measures of hands
- Digital systems encode the physical quantity using binary digits (or bits)
  - Applies to any physical quantity
  - Extensible accuracy
- Digital: many implementation technologies
  - Optical, acoustic, magnetic, electrical

Analog Clock Mechanism



Encodings

- Binary numbers (base 2 instead of base 10)
- Bit can hold "0" or "1"
- Byte = 8 bits strung together.
- Base-2 n-bit number can hold 2^n values
  - A byte can hold $2^8 = 256$ distinct values
- Examples (unsigned)
  - $1101_2 = 13_{10}$
  - $57_{10} = 111001_2$
- Octal encoding: Base 8 (0-7)
- Hexadecimal (hex) encoding: Base 16 (0-9, A-F)

Physical Realization 1

- Represent bits and operations
  - Storage (or memory)
  - Computation: add, compare, …
- Prevalent method today (and in the past several decades) uses current
  - Flow of electrons along a semiconductor
- A semiconductor is a material that can act as a conductor (allow flow) or a resistor (hinder flow)
  - Application of charge (voltage) changes this property
  - Silicon is the dominant material in use

Physical Realization 2

- Ohm's Law: V = I*R
  - Governs how much voltage ("electrical pressure") is needed to maintain a desired flow of electrons (current) given the resistance inherent in the material
- An individual TRANSfer resISTOR (transistor) acts as a "gate" controlling whether electrons flow through it
  - It acts like a wire when the gate is "closed"
- Electrons flowing through a (large) network of transistors is how we achieve computation
- 45nm manufacturing process means the transistor is $45*10^{-9}$m in each dimension
- Language for describing desired functionality and tools for translating the description into transistor networks

Universality of computing systems

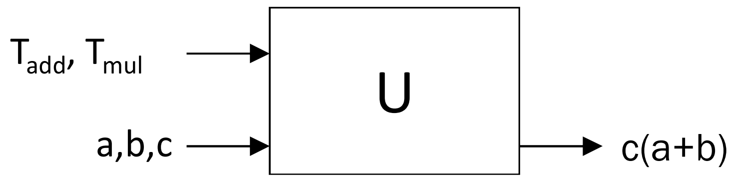
- Computer = Bits + Operations
- Bit: binary, digital
- Operation: add, branch on condition, …
- Program: a collection of bits
  - Encoding instructions
- An instruction:
  - Specifies what to do (operation)
  - On what quantities (operands)
  - What to do next
- Instructions distinguish computers from other computational devices (eg, adding machines)
  - Specify a sequence of operations
  - Vs. performing a given fixed task

Universality of computing systems

- Alan Turing formalized the notion of a universal computation device in 1937
- Symbols and abstract operations on them
  - Implementation is encapsulated in a "black box" Turing machine
    - Adder, multiplier
- String a bunch of Turing machines together to achieve some desired computation
- Turing's thesis: Every computation can be carried out on a collection of Turing machines

Universality of computing systems

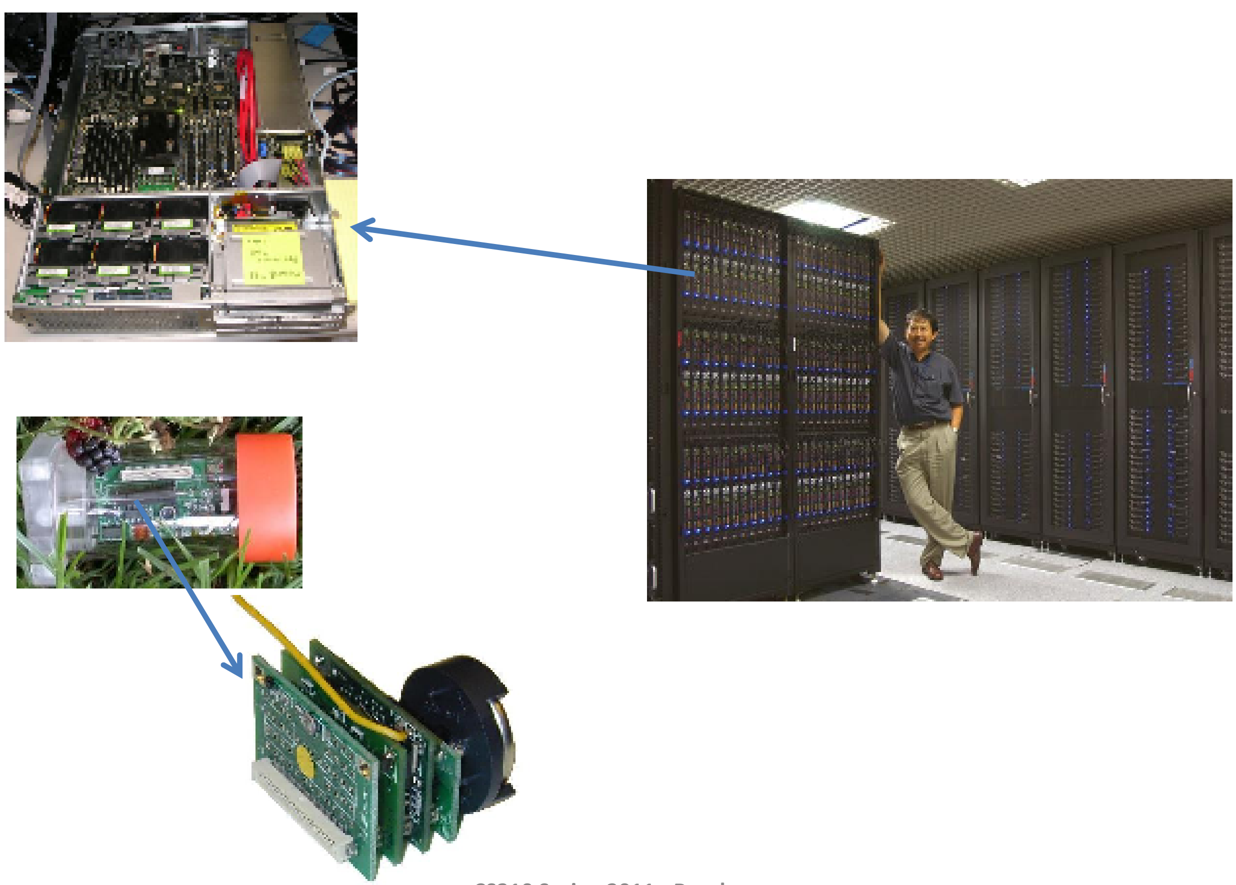
- Turing showed how a Universal Turing machine (U) can simulate any Turing machine
  - Effectively the first definition of what computers do!



Universal Turing Machine

- U is programmable
  - It can recognize input symbols (instructions and data)
  - It knows how to interpret instructions
  - It knows how to manipulate data

What Does This Mean for Us?



Abstraction as a Design Principle

- Computer systems use multiple levels of abstraction and interfaces
- (Ideally) Each level is only aware of the interfaces to the levels below it
- Example: Instruction Set Architecture (ISA)
- Example: I/O
- Example: Java
- Can lead to immense reduction of complexity
  (but you have to get the abstractions and interfaces just right)