DEPARTMENT OF INFORMATION SYSTEMS AND COMPUTER SCIENCE

CS152: Computer Organization

"From Sand to Supercomputers"

What is CS152?

- Computer Organization = how to build structures for computation
- Before (pre-1999)
  - CS142: Intro to Digital Electronics
  - CS50: Assembly Language
  - CS150: Computer Architecture
    - Now the MIS version of CS152.
- Today
  - CS 152: Integrated course based on an MIT course - 6.004: Computation Structures.

In this class, you will:

- ~~Experience pain and suffering.~~ Learn to love binary numbers, Boolean expressions, and math in general.
  - Expect lots and lots of exercises.
- Learn what computers are made of.
  - Mostly transistors, in case you didn't know.
- Build your own 32-bit RISC processor from simple logic gates (using a device logic simulator).
- Learn how to program in Assembly and Machine Language (using both the Beta RISC CPU simulator and an actual assembly language).

Lecture Time!

- Information: What is it?
- Machine-Readable Information: Not a Magical Transformation
- How to Measure Information: Simple Math
- Modular Design: Complicated Stuff is Made Out of Simple Stuff

From Sand to Supercomputers

- It's complicated... How do we manage this complexity?
- Answer: Modular Design (Abstraction & Composition)
  - Sand (SiO_2) + Electrons = MOSFET (much smaller than 1 micron, we're at 22 nanometers now!)
  - 2-8 Transistors = Logic Gate (AND, OR, etc.)
  - 2-16 Gates = Cell (as in memory cell)
  - 1K-10K Cells = Modules (RAM chip, anyone?)
  - 8-16 Modules = Integrated Circuit
  - 8-16 ICs + Wires = Printed Circuit Board
  - PCBs + Hardware + Software = Computer (PC)
    - Today's supercomputer = Tomorrow's computer
  - Multiple PCs = Parallel computing, Internet, etc.

In the beginning…

- What is Computation?
  - *Reliable processing of information*!
- A computer takes in “old” information...
- … and processes it to generate “new” information!
  - Load a file and turn it into a picture!
  - Generate a random number and see if your avatar manages to score a critical hit!
  - Take the coordinates and volumes of two objects and see if they collide!
  - And much, much more!

What is Information?

- Knowledge that reduces uncertainty. For example:
  - Uncertainty: What is Ateneo's phone #?
    - How much uncertainty? 10^7 possibilities.
      - How did we arrive at that number?
  - Information #1: "It starts with 426"
    - Reduces uncertainty to 10^4 possibilities.
      - Again, how did we arrive at that number?
  - Information #2: "It's 426-6001"
    - No more uncertainty: only 1 possibility!
- The more uncertainty reduced = the more information received! (and vice versa)

The Key to Computation

- Any information can be represented by a number.
    - Even this picture on the right! → →

- This fact simply says:
  You can map each possibility to a number!
  - Example: How to represent a 3-letter, uppercase only, password (e.g., ABC).
  - There are 26^3 = 17576 possibilities. So, to specify ONE combination, we can use a number from 0 to 17575.
    - AAA = 0, AAB = 1, … ZZZ = 17575
  - Note: Other encodings can also be used for partial info.
    - Example: Use a number from 0 to 25 for the first letter.

Why is this so important?

- If this fact was not true, computation is IMPOSSIBLE.
  - First, information can be represented by a number.
  - Next, a number can be represented in physical form.
    - e.g., voltage, magnetic charge, holes on paper, amount of water, light intensity, DNA molecule, etc.
  - Finally, now that the info is in physical form, it can be processed using a real machine!

Digits and Bits: How Much Information?

- Amount of Information = length of number needed to represent the info
- Using decimal numbers:

  $d = \log_{10}(N/M)$ *digits*

  – where: N = # of possibilities BEFORE info is received
  – M = # of possibilities AFTER info is received
  – (assuming all possibilities are equally likely)
- Using binary numbers:

  $b = \log_{2}(N/M)$ *bits*

  – Usually, we use binary (using 1's and 0's) instead of decimal, so info is measured in "bits" (binary digits)
- Note: The more uncertainty removed (the more specific the information) the longer the number.
- Note #2: Round it up! (Information amounting to a fraction of a bit/digit will still use a whole bit/digit.)

Examples

$d = \log_{10}(N/M)$ digits $b = \log_2(N/M)$ bits

It is in the nature of programmers to start from 0, not 1.
(0-9) not (1-10)
(0-1) not (1-2)

- What is Ateneo's phone #?
- Case 1: "426-6001"
  - reduces possibilities from 10^7 to 1, so, I've given you:
  - $d = \log_{10}(\ 10^7 / 1\ ) = \log_{10}(\ 10^7) = 7$ digits
  - $b = \log_2(\ 10^7 / 1\ ) = \log_2(\ 10^7) = 23.25$ bits $= 24$ bits
- Case 2: "it starts with 426"
  - reduces possibilities from 10^7 to 10^4,
  - $d = \log_{10}(\ 10^7 / 10^4\ ) = \log_{10}(\ 10^3) = 3$ digits
  - $b = \log_2(\ 10^7 / 10^4\ ) = \log_2(\ 10^3) = 9.97$ bits $= 10$ bits

Examples

$d = \log_{10}(N/M)$ digits $\quad b = \log_2(N/M)$ bits

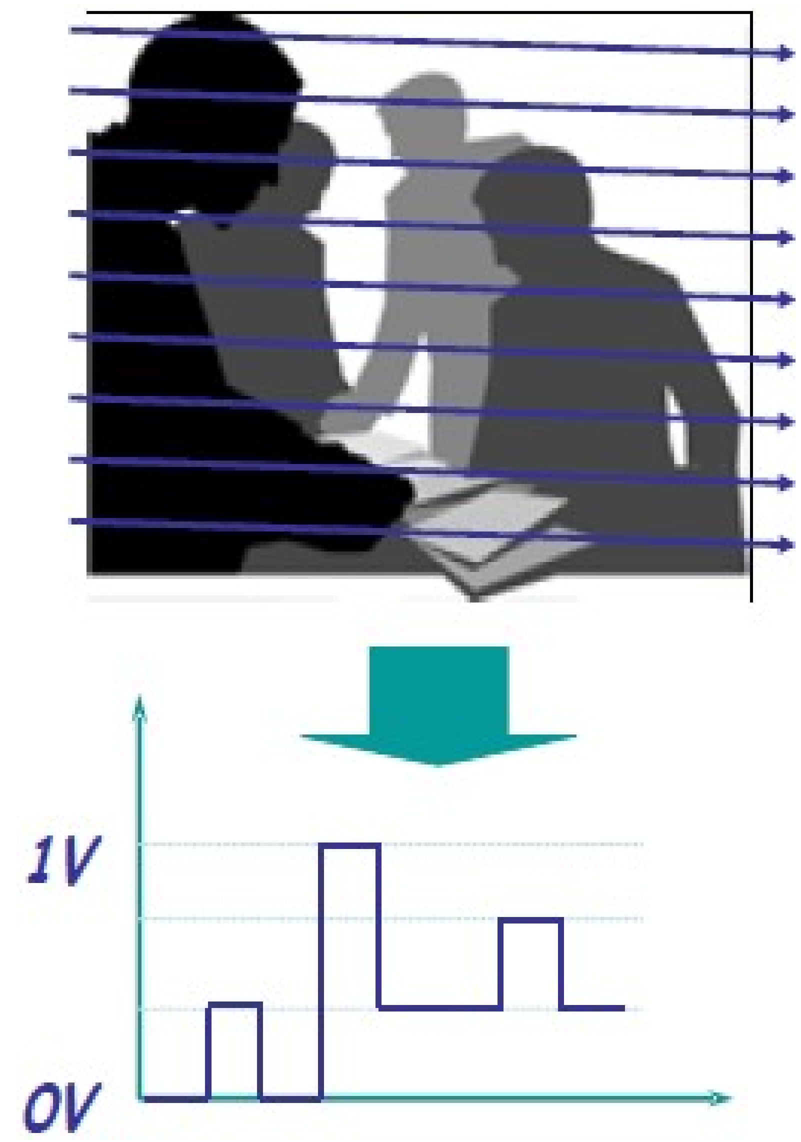
It is in the nature of programmers to start from 0, not 1.
(0-9) not (1-10)
(0-1) not (1-2)

- What is your 3-letter password?
- Case 1: "ABC"
  - reduces possibilities from 26^3 to 1,
  - $d = \log_{10}(26^3 / 1) = \log_{10}(17576)$
    $= 4.24$ digits $= 5$ digits
  - $b = \log_2(26^3 / 1) = \log_2(17576) = 14.10$ bits $= 15$ bits
- Case 2: "the first letter is A"
  - reduces possibilities from 26^3 to 26^2,
  - $d = \log_{10}(26^3 / 26^2) = \log_{10}(26) = 1.41$ digits
  - $b = \log_2(26^3 / 26^2) = \log_2(26) = 4.70$ bits

DISCS

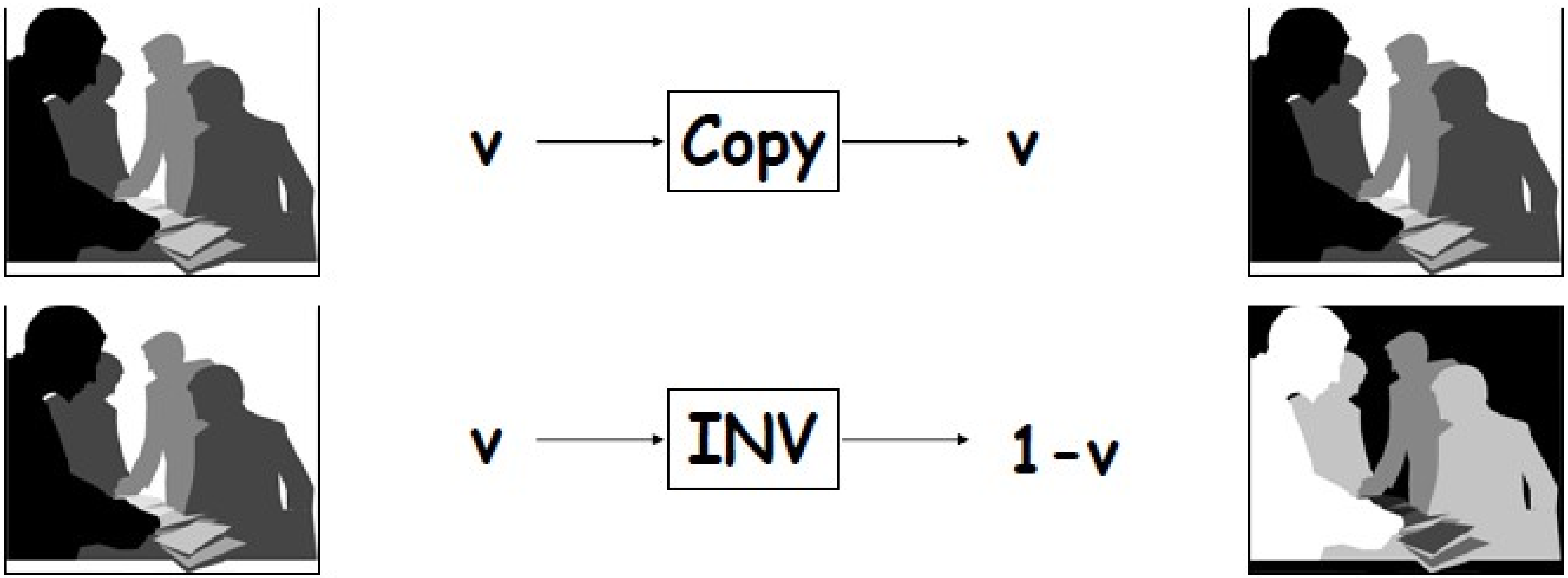
Representing Information

- Info → numerical value → physical form
- Example: Representing Images
  - Represent color of each point as number
    - e.g., BLACK = 0, WHITE = 1,
      28% gray = 0.28
  - Represent number as a voltage
    - e.g., 0 = 0 volts, 1 = 1 volt,
      0.28 = 0.28 volts
  - Scan points in order and generate
    a voltage waveform V(x,y) or V(t)
    - Voltage waveform can now be transmitted
      via radio waves, stored to tape, processed, etc.



Processing Information

- Computation = reliable *processing* of information
- Computers "do" things with information!
    - Some very simple "computers":



    - Others: add, rotate, resize, move, etc.

Processing Blocks

- First step in Modular Design is *Abstraction*:
  - Understanding BEHAVIOR without knowing IMPLEMENTATION.
- "Design-by-contract"
  - Determine what the thing is SUPPOSED to do.
    - Inputs, Outputs, and Function
  - But leave it to implementer to do it.
    - Don't care HOW they do it, as long as they FULFILL contract!
- Advantages
  - Saves designer trouble of worrying about how.
  - Allows implementer to implement in different ways or different technologies.
- Second step is *Composition*:
  - As long as contracts are followed, then we can chain or combine processing blocks together!

How We Build Computers

- Perform complex computations by composing simple processing blocks together.
  - Think of LEGO or Tinkertoys.

You’ll be doing this a lot in CS152, especially in CS152b!