# CS 101: A bird's eye view

Abhiram Ranade

# A computer can do many things

- Predict the weather
- Make railway bookings
- Play chess and beat human world champions
- Control machinery in large factories

How can a single machine do all this?

# A short answer

- Most problems that we want to solve can be formulated as numerical problems
- We can design electrical circuits that can perform numerical calculations.
- Computer = single universal circuit for all calculations.

# Outline

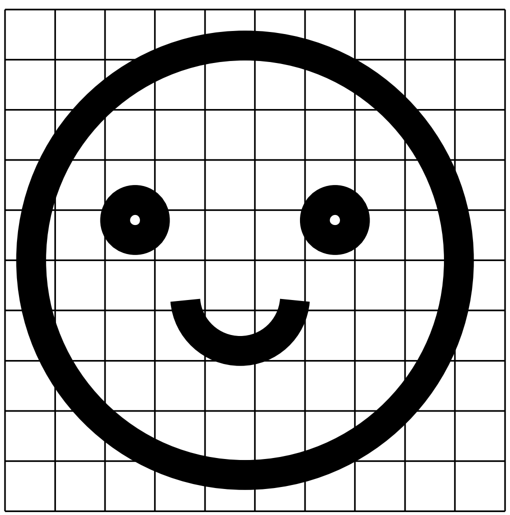
- How to represent real life problems as problems on numbers.
  - "What is in this picture?"
  - "Will it rain tomorrow?"
  - "Find information about Bt brinjal"
- Basics of processing numbers using circuits.
- Sketch of a computer as a huge circuit

# “What is in this picture?”

# How to represent black and white pictures

- Suppose picture is 10cm x 10cm.
- Break it up into 0.1 mm x 0.1 mm squares
- 1000 x 1000 squares.
- If square is mostly white, represent it as 0.
- If square is mostly black, represent it as 1.
- Picture = 1 million numbers!

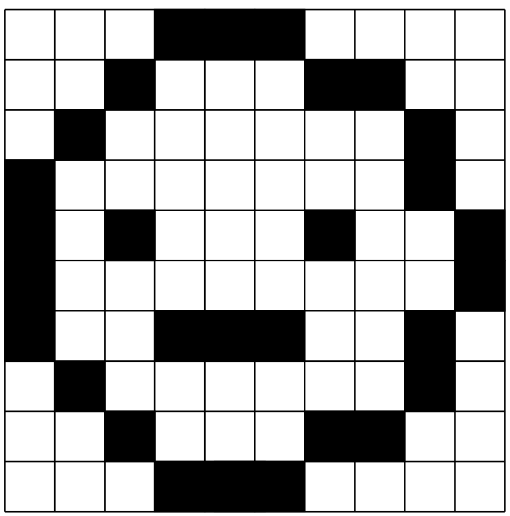
# Picture, Representation, Reconstruction



(a)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

(b)



(c)

# Remarks

- Better representation if picture divided into more cells.
- Pictures with different “gray levels”: sequence of numbers indicating degree of darkness
- Pictures with colours: picture = 3 sequences
  - sequence for red component,
  - sequence for blue component,
  - sequence for green component

# Is there a vertical line in this picture?

- Input: sequence of 1 million numbers (0 or 1) representing a 10cm x 10cm black and white picture
- What property does the sequence need to have if it is to contain a vertical line?

# Is there a vertical line in this picture?

- Input: sequence of 1 million numbers (0 or 1) representing a 10cm x 10cm black and white picture
- What property does the sequence need to have if it is to contain a vertical line?
- All 0s, except for 1s at positions

  i, i+1000, i+2000, i+3000, i+4000,…

Question about picture converted into question about numbers.

# Does the picture contain a chameleon?

- Question expressed as:
  - Does the sequence of numbers representing the picture contain a subsequence satisfying certain properties?
- “certain properties”: Enormous ingenuity needed to specify.
- Main concern of the deep subject “Computer Vision”

# Predicting weather

- Divide the surface of the earth into small cells, e.g. cut along integer latitude and integer longitude.
- For each cell have several numbers:
  - A number representing its temperature
  - A number representing its pressure
  - …
- Numbers we wrote down = representation of the current weather!

# Predicting the weather (contd.)

- Laws of physics can be used to determine numbers for the next step
- “Laws of physics”:
  - “Heat Equation”: how to calculate temperature for next time step given current temperature for a simple object
  - Laws are complex for land-sea-air system.
- Central concern of the deep subject “Meteorology”
- Representation better if cells are small

# “Tell me about Bt brinjal”

- Each character that we can type on the keyboard is represented by a specific number.
- American Standard Code for Information Interchange (ASCII)
  - ‘a’ = 97, ‘b’ = 98, …’z’ = 122.
  - ‘A’ = 65, ‘B’ = 66, …’Z’ = 90.
  - ‘brinjal’ = 98, 114, 105, 110, 106, 97, 108.
- Document = very long sequence of numbers.

# “Tell me about Bt brinjal”

- Find the sequence for ‘brinjal’ in all the different sequences representing different documents on the computer.
- Brinjal = also called Eggplant.  Must look in document for sequences for both words.
- “Is Bt Brinjal good for you?” : much more complicated searches..
- Subject of deep area of CS: “Information Retrieval”

# Summary

- Questions about pictures, weather, documents can be converted to questions about properties of number sequences.
- Finding answers requires solving interesting math problems.
- How will you represent Chess playing as a question on numbers?

# Representing numbers in circuits

- 0 : low voltage, say 0.0 volts on some wire or capacitor
- 1 : high voltage, say 0.7 volts
- Larger numbers:
  - Convert number to binary. Then use above representation for each bit. Bit = Binary digit
  - $25 = 2 \times 10^1 + 5 \times 10^0$
  - $25 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
  - 25 : 11001 : high, high, low, low, high

# Representing Numbers (contd.)

- Standard representations use fixed number of voltages, e.g. 32.

  25: LLLLLLLLLLLLLLLLLLLLLLLLLLLHHLLH

  Often we write 0 for L and 1 for H

  25: 00000000000000000000000000011001

- With 32 voltages (L/H) we can only represent numbers between 0 and $2^{32} - 1$ (“all 1s”)
- How to represent negative numbers and fractions: next.

# General principles of representation

- If we have 32 voltages, each taking value L/H, we can have a total of $2^{32}$ voltage patterns.
- We can decide what each pattern means.
- Previous representation:
  - each voltage pattern represents binary number obtained by setting Low = 0, High = 1.
  - But we can make other correspondences.
- Terminology: Bit = voltage taking value L/H

# Representing Positive/Negative Integers using 32 bits

- Represent magnitude using 31 bits/voltages.
- Represent sign using 1 bit/voltage:
  - L = +ve, H = -ve
- -25: 10000000000000000000000000011001
- Largest: $2^{31} - 1$, smallest: $-(2^{31} - 1)$
- Actual representation on real computers is slightly different.

# Representing real numbers

- Example: Avogadro's number $6.022 \times 10^{23}$
  - Convert to binary: $1.11111110001010111 \times 2^{1001110}$
  - Use 23 bits for magnitude of fraction, 1 bit for sign of fraction. Equivalent of 7-8 decimal digits.
  - Use 7 bits for magnitude of exponent, 1 bit for sign of exponent
  - 01111111100010101110000001001110
  - Decimal point is assumed after $2^{nd}$ bit.
  - Actual representation slightly different.

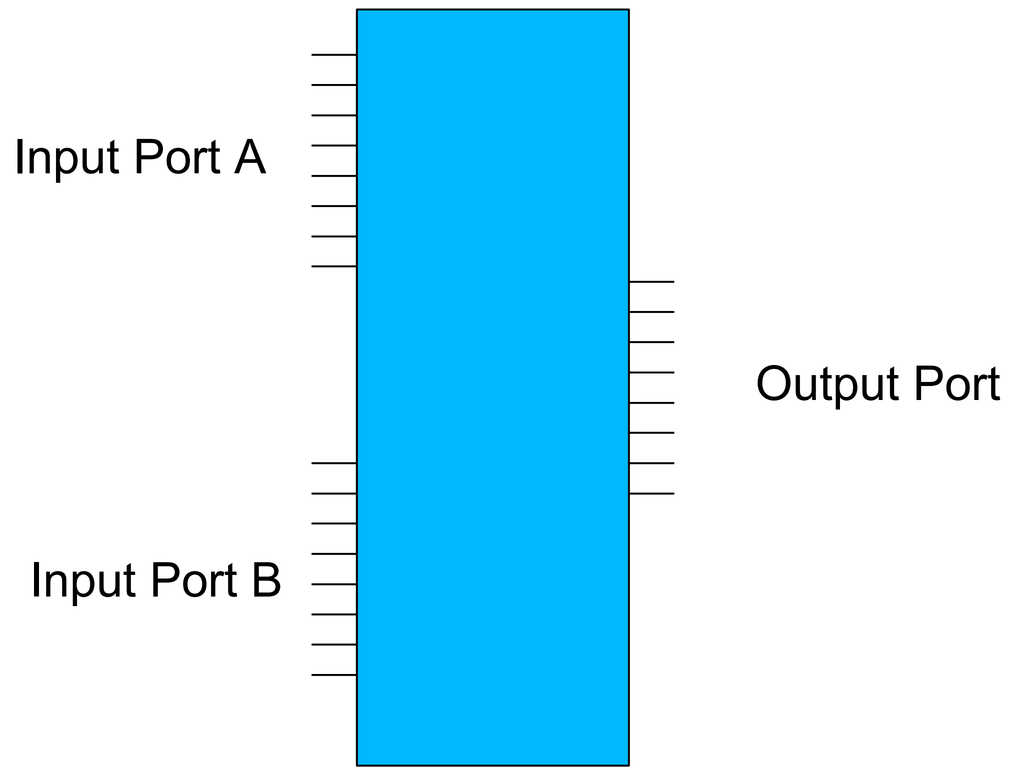
# Other representations

- Positive and negative integers: 16 bit, 64 bit
- Real numbers: 64 bits (“double precision”)
  - 53 bit fraction = 18 decimal digits
  - 11 bit exponent
- 96 or 128 bits also used.

# An adder circuit

- Different circuit for each number representation.
- Input port A: 32 wires
- Input port B: 32 wires
- Output port: 32 wires
- First addend : Feed voltages representing the number on port A.
- Second addend: Feed on port B.
- After some delay: voltages representing sum available on output port.

# An Adder Circuit (8 bit input/output)



# Organization of a computer

- Memory unit
- Arithmetic and Logic Unit (ALU)
- Control Unit
- Keyboard, monitor screen, disk, …
- Wiring to connect these together

# Memory Unit

- Collection of capacitors.
  - Group of 8 capacitors storing 8 bits = 1 byte
  - Group of 32 capacitors storing 32 bits = 1 word
- Example:
  - Memory contains $2^{30}$ words, about $10^9$ words.
  - Imagine words are located on a long line. Distance along the line = address of the byte.
  - Addresses are between 0 and $2^{30} - 1$.
  - Addresses fit in 30 bits

# Memory Unit Example (contd.)

- Connection to the outside world:
  - Data port: 32 wires
  - Address port: 30 wires
  - Read control wire
  - Write control wire

# Example (contd.)

- Writing value V into word at address A:
  - Convert A to binary (30 bits)
  - Place corresponding voltages on Address port.
  - Place representation of V (32 bits) on Data port wires
  - Set write control wire to “high”.
  - Wait for circuit to do its work.
- Result: value V stored in word A of memory.
  - V will stay in address A even after address/data port values change.
- How: magic of circuit design!

# Example (contd.)

- Reading value from word starting at address A:
  - Convert A to binary (30 bits)
  - Place corresponding voltages on Address port.
  - Set read control wire to “high”.
  - Wait for circuit to do its work
- Result: value V stored in word A of memory appears on Data port.
- How: magic of circuit design!

# Arithmetic and Logic Unit (ALU) Example

- Addition already discussed. Circuits available for other arithmetic operations, e.g. subtraction, multiplication, division.

# Input/Output devices

- Keyboard: sends ASCII bit pattern of key pressed on connecting wire.
- Monitor: will display character whose value is sent on connecting wire.
- More complex devices/protocols possible.

# Control Unit

- Consists of Instruction Fetch Unit (IFU) + Decode and execute unit (DEU)
- DEU:
  - Connected to other units in the computer.
  - Decides what other units in the computer will do.
  - Has many preset “command sequences”

# Command Sequences

- Example: command sequence 0:
  - Send a value V to Address port of memory
  - Command memory to read
  - Ask for the data read to be sent to input port A of ALU.
- Example command sequence 1:
  - Same as above, except data goes to input port B.
- Example command sequence 10:
  - Command the ALU to add, assuming numbers are non-negative.

# Command sequence example (contd)

- Example: command sequence 2
  - Move value V to address port of memory
  - Move value at ALU outport to memory data port.
  - Command memory to write.
- What if DEU executes command sequences 0, 1, 10, 2 in succession?

# Control Unit functioning

- Which command sequence will be executed by DEU? What value V will it use?
  - IFU sends the sequence number, and value V.
- How does the IFU decide what to send?
  - IFU fetches the sequence number and V from memory!

# Sketch of IFU functioning

- IFU contains a register called program counter (PC).
- IFU sends PC to address port of memory, and reads a word. This will be sent to DEU as sequence number.
- IFU adds 1 to PC.
- Sends PC again to address port, reads another word. This will be the value V.
- IFU adds 1 to PC.
- Waits for DEU to do its work.
- Repeat

# Example

- PC = 100, Memory contains 0, 50, 1, 51, 10, 0, 2, 52 in locations 100 through 107.
- What happens when the computer executes?
  - IFU fetches 0, sends as sequence number to DEU
  - IFU fetches 50, sends as V to DEU
  - DEU executes: Data from address 50 goes to ALU port A.
  - IFU fetches 1, sends as sequence number to DEU
  - …

# Example (contd.)

- Effect of IFU fetching 0,50,1,51,10,0:
  - Content of memory address 50, 51 moved to ALU input ports, and added.
- Effect of 2,52:
  - Suppose command sequence 2 causes data in ALU output to be moved to address V, which has been specified as 52.
  - Sum of values in addresses 50, 51 stored in 52
- Sequence 0,50,1,51,10,0,2,52: "machine language program to add two numbers"

# Terminology

- Sequence number : operation code
- Value V: operand
- Sequence number + V : instruction
- Normally, IFU fetches instructions from consecutive addresses in memory.
- Some operation codes may cause DEU to modify PC register in IFU. This will cause IFU to fetch instructions from a new address.
  - Jump instruction

# What C++ compiler does

- Take a C++ program, generate an equivalent machine language program.
- Machine language program can be loaded into memory and run.

# Summary

- Numbers can be represented in many ways.
- Memory is organized as several words, each word has an address.
- What the computer does, what instruction it executes, are also stored in memory.
  - "Stored program computer"
- Compilation = translating C++ to machine language.