Ceng 111 – Fall 2020
Week 2

Digital Computation

Credit: Some slides are from the "Invitation to Computer Science" book by G. M. Schneider, J. L. Gersting and some from the "Digital Design" book by M. M. Mano and M. D. Ciletti.

An example algorithm

Algorithm for Adding Two m-Digit Numbers

Given: $m \geq 1$ and two positive numbers each containing m digits, $a_{m-1}\ a_{m-2,} \ldots a_0$ and $b_{m-1}\ b_{m-2,} \ldots b_0$
Wanted: $c_m c_{m-1}\ c_{m-2} \ldots c_0$, where $c_m c_{m-1}\ c_{m-2} \ldots c_0 = (a_{m-1}\ a_{m-2} \ldots a_0) + (b_{m-1}\ b_{m-2} \ldots b_0)$

Algorithm:

Step 1 Set the value of *carry* to 0.
Step 2 Set the value of *i* to 0.
Step 3 While the value of *i* is less than or equal to $m - 1$, repeat the instructions in steps 4 through 6.
Step 4 Add the two digits a_i and b_i to the current value of *carry* to get c_i.
Step 5 If $c_i \geq 10$, then reset c_i to $(c_i - 10)$ and reset the value of *carry* to 1; otherwise, set the new value of *carry* to 0.
Step 6 Add 1 to *i*, effectively moving one column to the left.
Step 7 Set c_m to the value of *carry*.
Step 8 Print out the final answer, $c_m\ c_{m\text{-}1}\ c_{m\text{-}2} \ldots c_0$.
Step 9 Stop.

From “Invitation to Computer Science”

How to represent algorithms

- Pseudo-code

Algorithm for Adding Two *m*-Digit Numbers

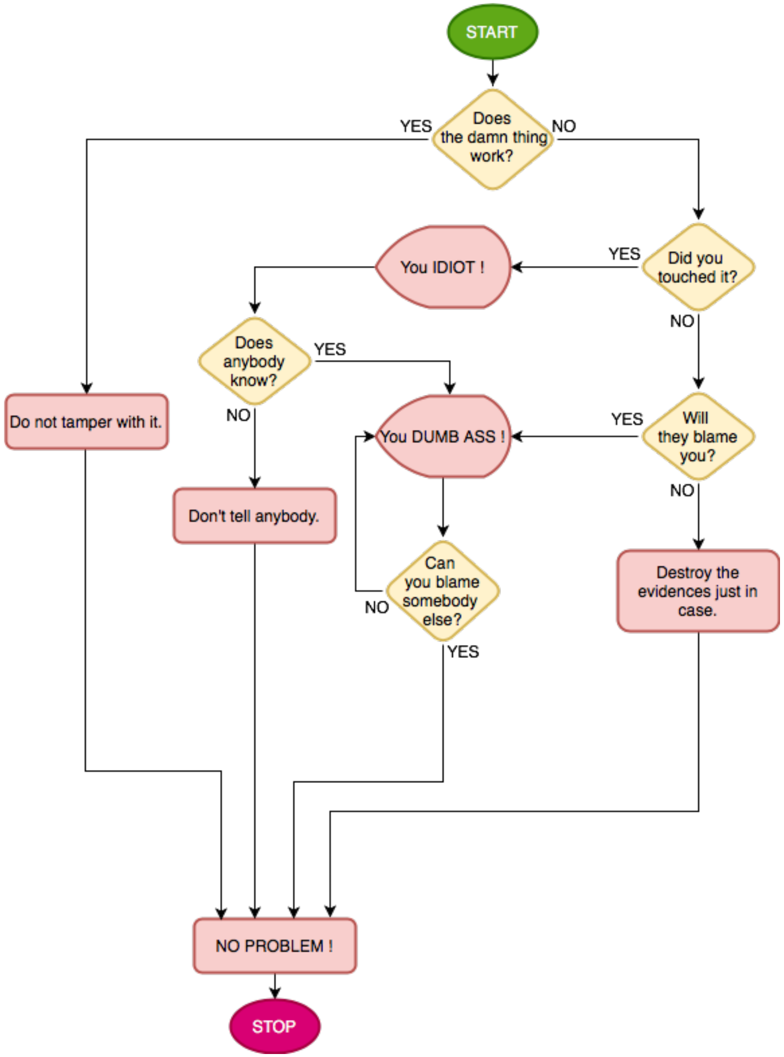
Given: $m \geq 1$ and two positive numbers each containing m digits, $a_{m-1}\ a_{m-2}, \ldots a_0$ and $b_{m-1}\ b_{m-2}, \ldots b_0$
Wanted: $c_m c_{m-1}\ c_{m-2} \ldots c_0$, where $c_m c_{m-1}\ c_{m-2} \ldots c_0 = (a_{m-1}\ a_{m-2} \ldots a_0) + (b_{m-1}\ b_{m-2} \ldots b_0)$

Algorithm:

Step 1 Set the value of *carry* to 0.
Step 2 Set the value of *i* to 0.
Step 3 While the value of *i* is less than or equal to $m - 1$, repeat the instructions in steps 4 through 6.
Step 4 Add the two digits a_i and b_i to the current value of *carry* to get c_i.
Step 5 If $c_i \geq 10$, then reset c_i to $(c_i - 10)$ and reset the value of *carry* to 1; otherwise, set the new value of *carry* to 0.
Step 6 Add 1 to *i*, effectively moving one column to the left.
Step 7 Set c_m to the value of *carry*.
Step 8 Print out the final answer, $c_m\ c_{m-1}\ c_{m-2} \ldots c_0$.
Step 9 Stop.

How to represent algorithms

- Flow-charts



Why are algorithms important?

- If we can specify an algorithm to solve a problem then we can automate its solution.
- No algorithm => No software => No automation!

From “Invitation to Computer Science”

Can we find algorithms to all problems?

NO!

- There are problems which have no generalized solutions – unsolvable or intractable
- Some with an algorithm would take so long to execute that the algorithm is useless
- Some problems we have not yet discovered an algorithm for

From "Invitation to Computer Science"

A formal definition of algorithm

- "Starting from an initial state and initial input (perhaps empty), the instructions describe a **computation** that, when executed, will proceed through a finite number of well-defined successive states, eventually producing "output" and terminating at a final ending state."

“Computation”

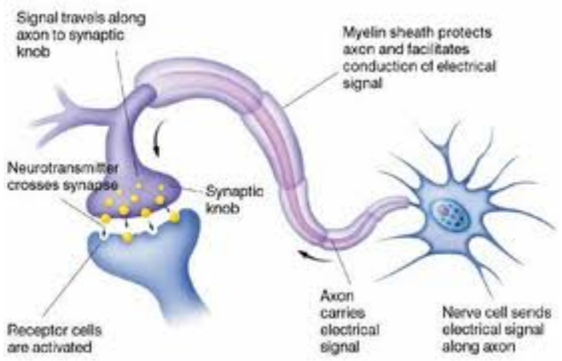
- Digital vs. analog computation
- Sequential vs. parallel computation
- Batch vs. interactive computation
- Evolutionary, molecular, quantum computation
- “Physical computation” / “Digital Physics”
  - ‘The whole universe is itself a computation’

“Computation” (cont.d)

- Problem: Find temperature of the water if A&B were mixed together.
- Any suggestions on how to solve it?

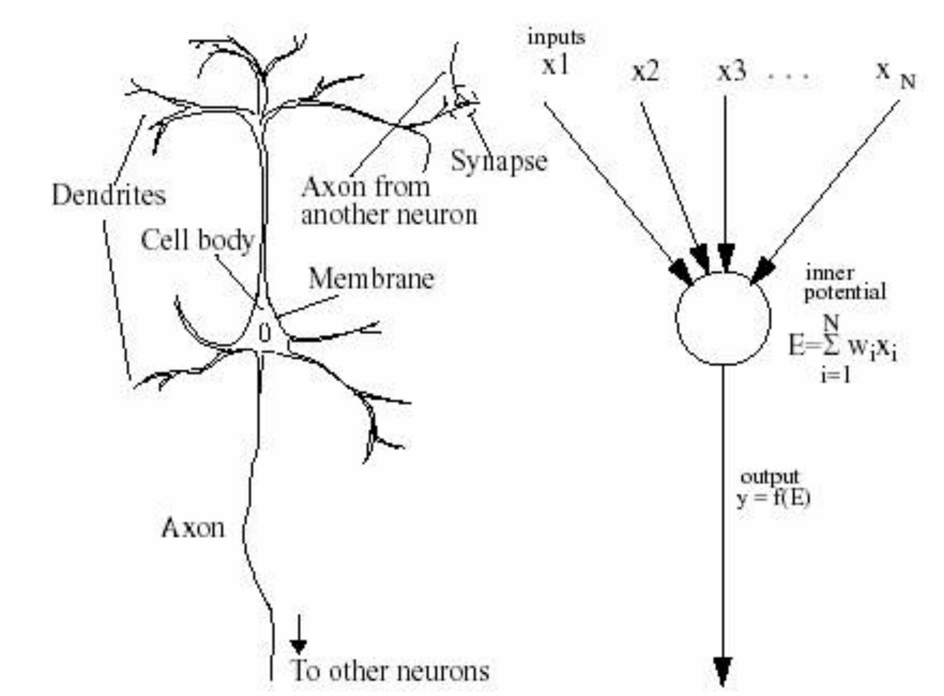
Computation in our brain

- Highly-connected network of neurons.
- How many neurons?
  - Approx. 10^{11} neurons and 10^{14} synapses.
- How do they transmit information?
  - Using nothing else than charged molecules.

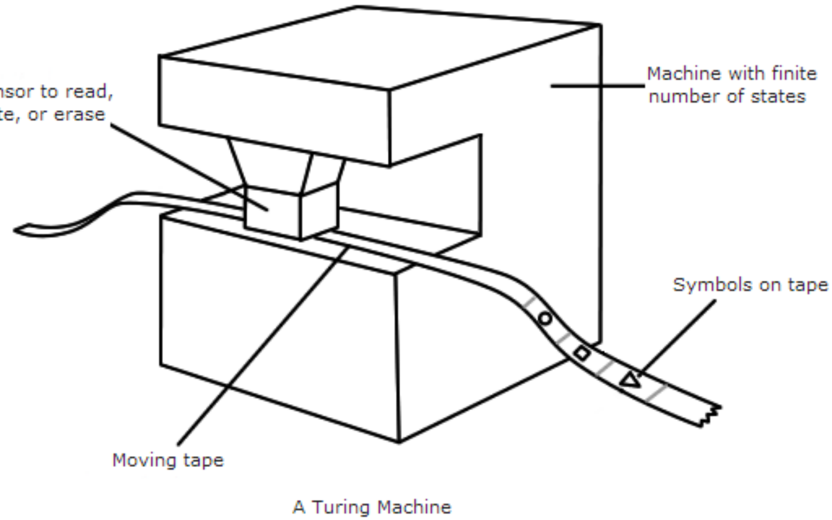


Computation in our brain (cont'd)

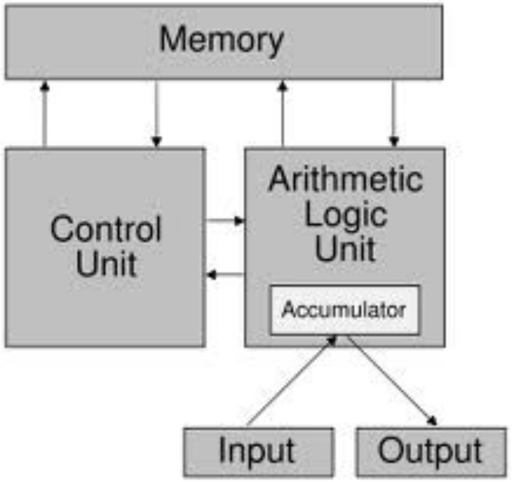
- Each neuron gets input and produces an output using an "activation function"



- Some of ours' is smaller but they have essentially the same computational mechanisms! ☺



Turing Machine



Von Neumann Architecture

DIGITAL COMPUTATION

METU Computer Engineering
METU Computer Engineering

BUT FIRST SOME HISTORICAL OVERVIEW

The Early Period: Up to 1940

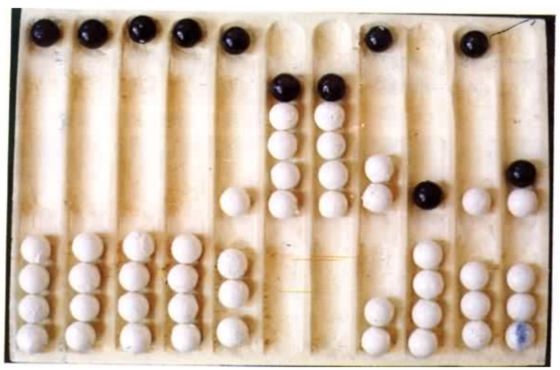
- 3,000 years ago: Mathematics, logic, and numerical computation
  - Important contributions made by the Greeks, Egyptians, Babylonians, Indians, Chinese, and Persians
  - Cuneiform
  - Stone "abacus"
- http://www.thocp.net/slideshow/0469.htm

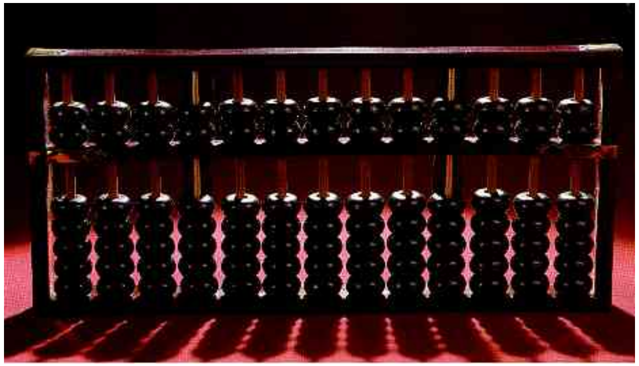
Slide from "Introduction to Computing"

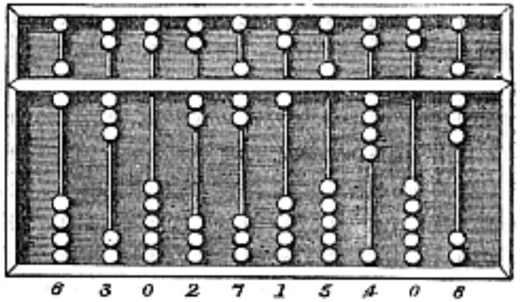
ABACUS

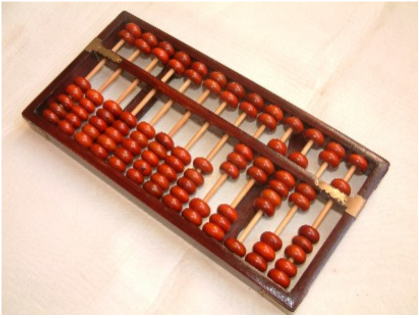
Early calculating devices

ABACUS – 2700 BC (Mesopotamia)







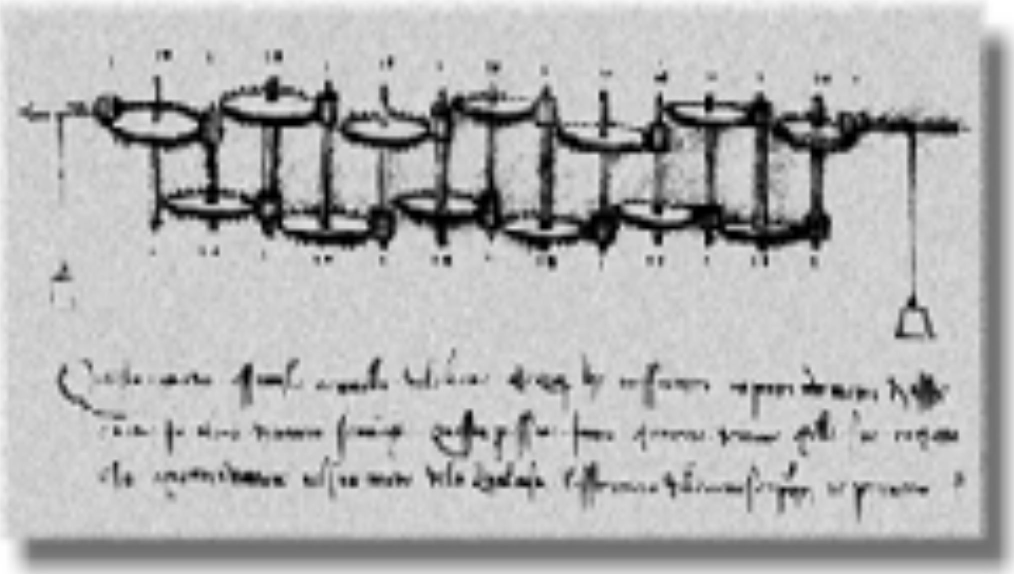


http://www.computersciencelab.com/ComputerHistory/History.htm

Slide from "Introduction to Computing"

DaVinci

- 1452-1519 Leonardo DaVinci sketched gear-driven calculating machines but none were ever built.



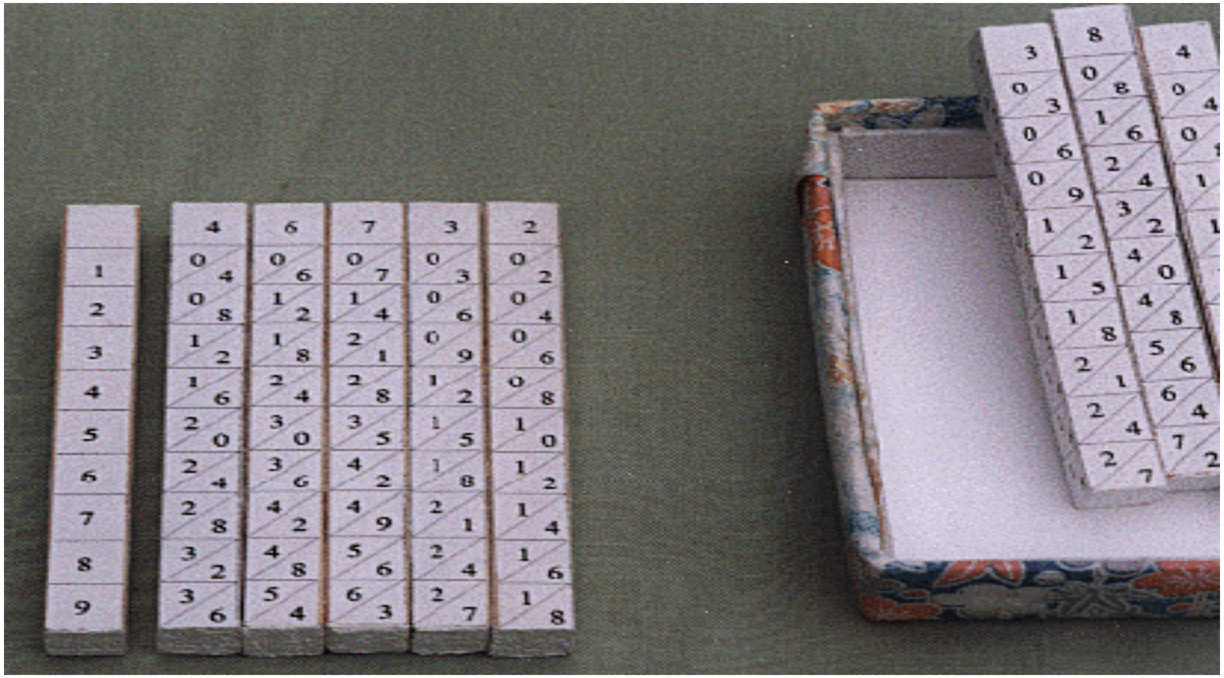
http://www.computersciencelab.com/ComputerHistory/History.htm

Slide from “Introduction to Computing”

Napier's Bones

- 1614: Logarithms
  - Invented by John Napier to simplify difficult mathematical computations

Napier's Bones:

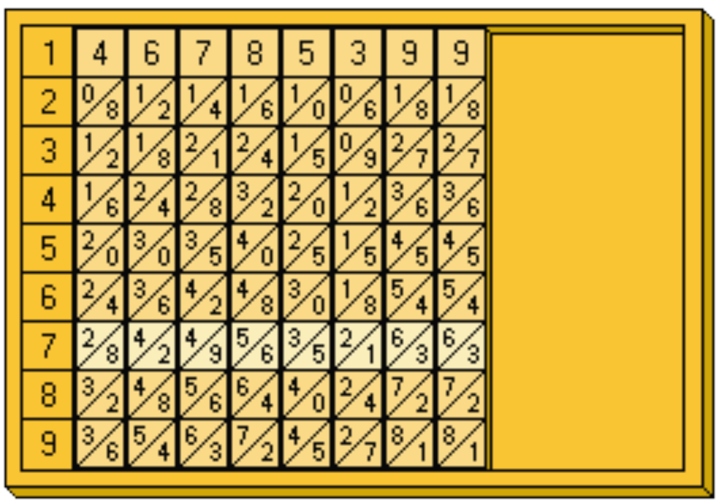


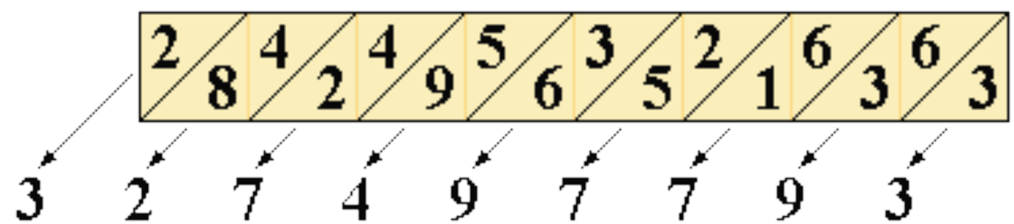
http://www.computersciencelab.com/ComputerHistory/History.htm

Slide from "Introduction to Computing"

METU Computer Engineering

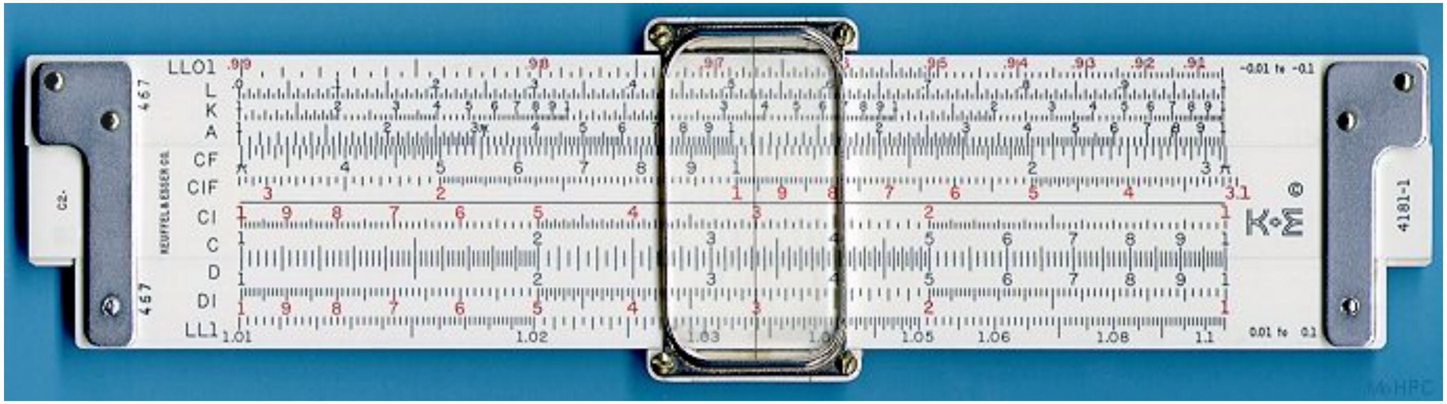
- If you want to multiply 7 by 46785499:





Slide Rule (slipstick) "a mechanical analog computer"

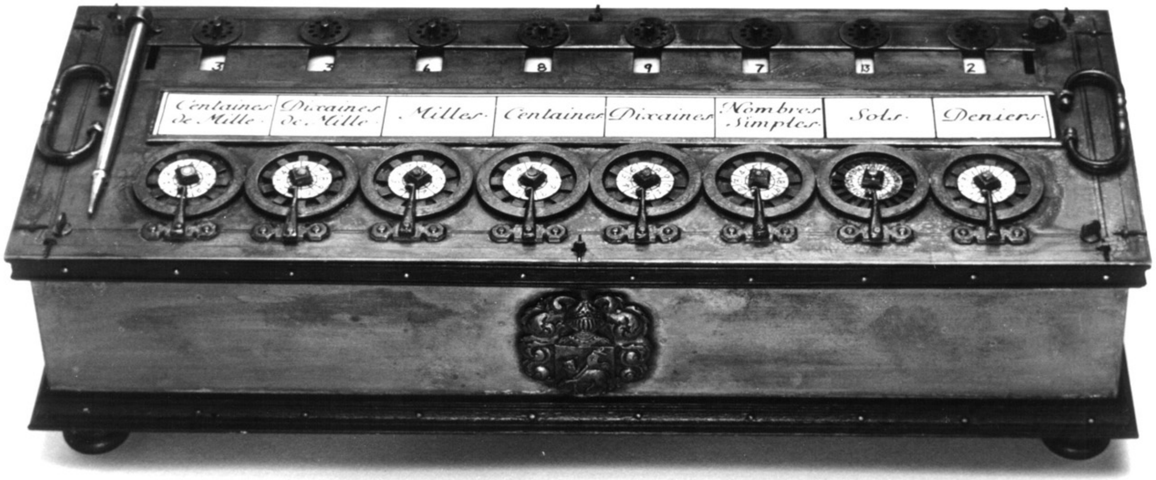
Around 1622: First slide rule created



http://www.computersciencelab.com/ComputerHistory/History.htm

Slide from "Introduction to Computing"

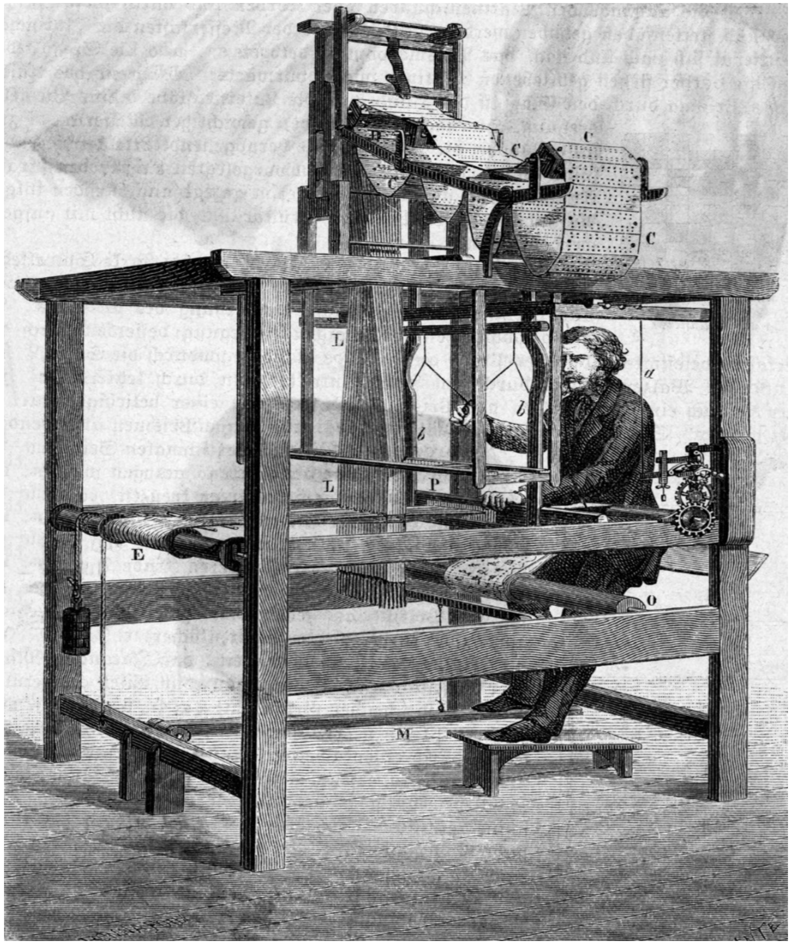
METU Computer Engineering



The Pascaline: One of the Earliest Mechanical Calculators

Slide from “Introduction to Computing”

The Early Period: Up to 1940

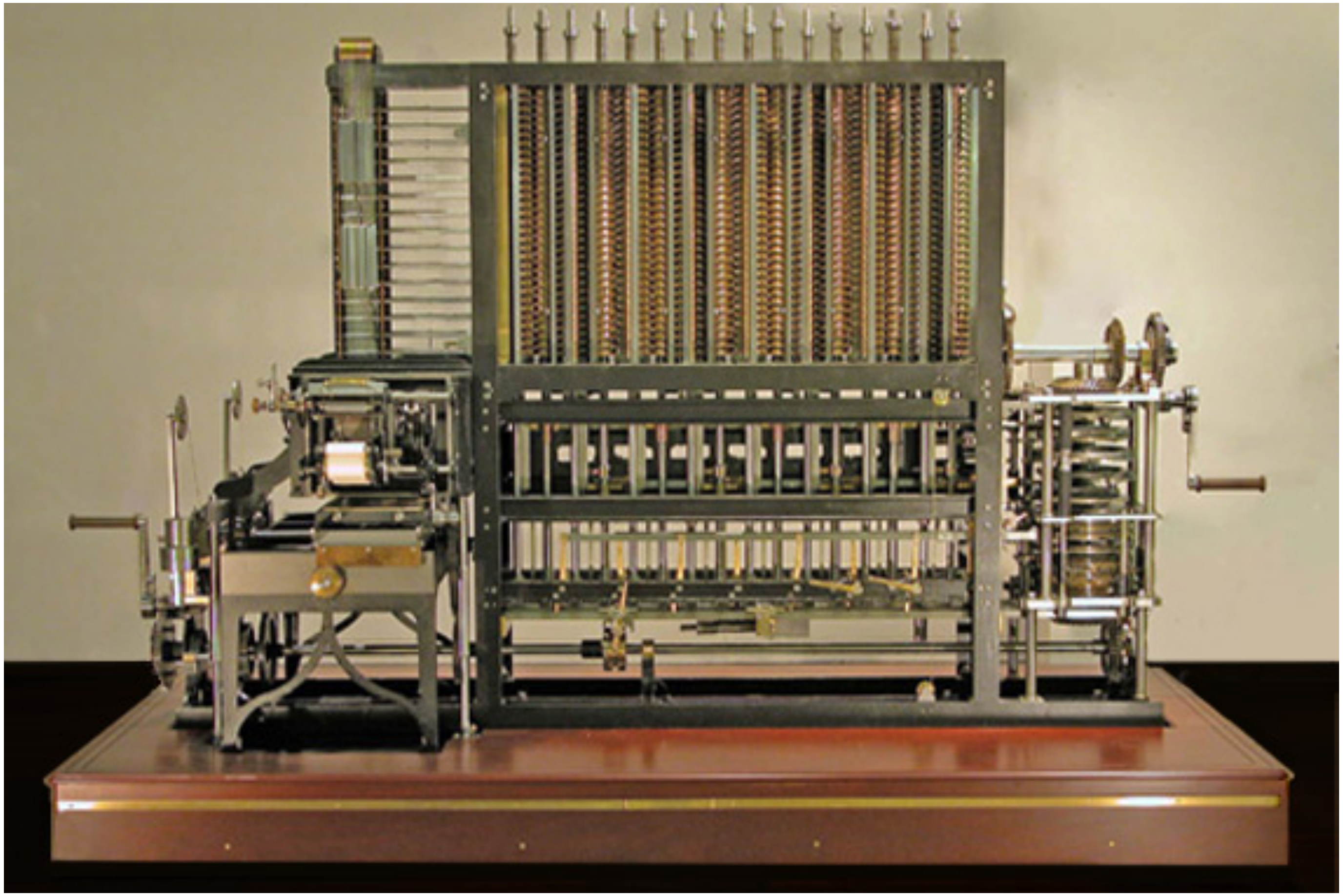


Jacquard's Loom

Also see http://www.computersciencelab.com/ComputerHistory/HistoryPt2.htm

Slide from "Introduction to Computing"

Difference engine



http://www.youtube.com/watch?v=0anIyVGeWOI

METU Computer Engineering

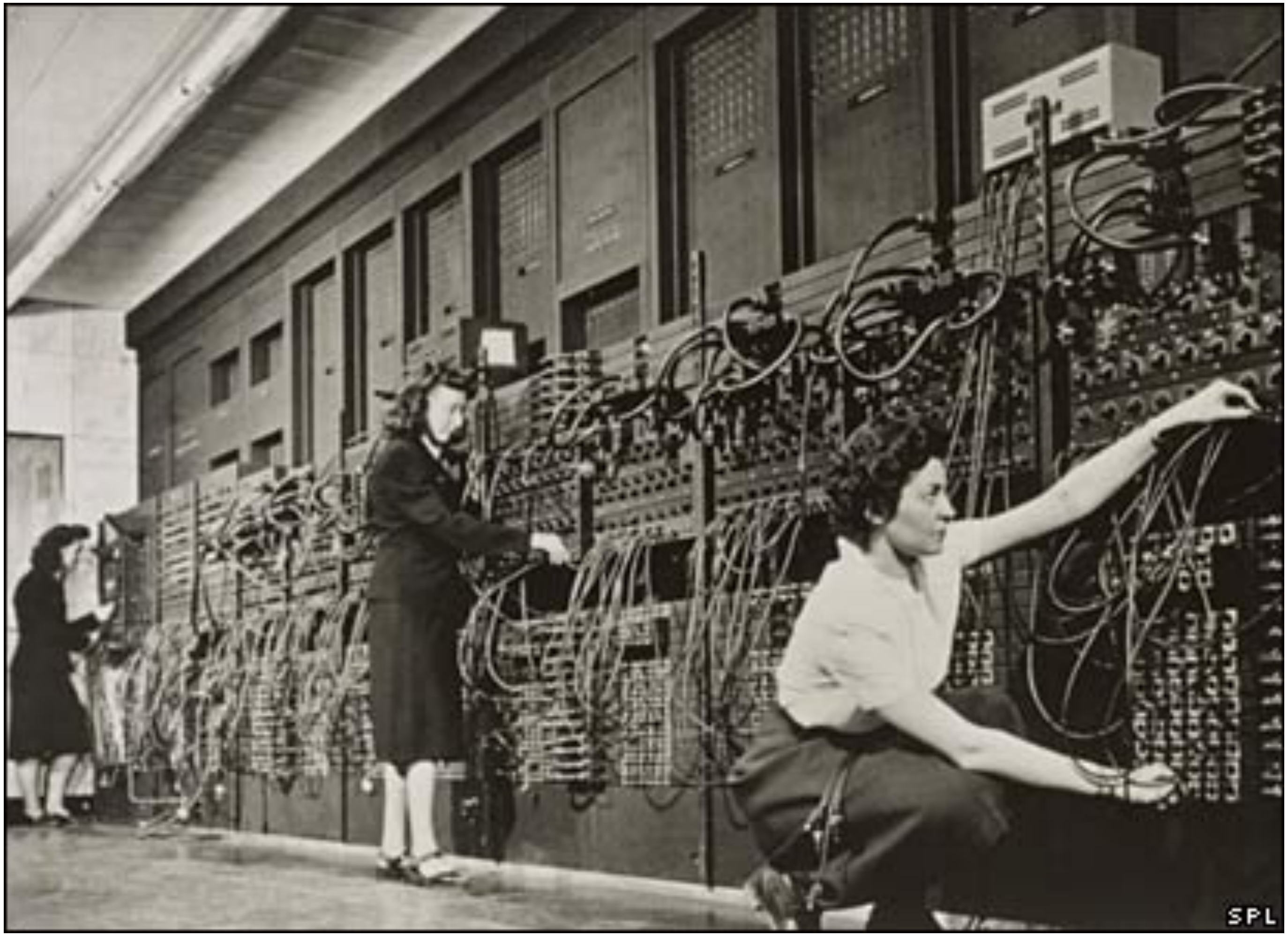
Slide from "Introduction to Computing"

The Harvard Mark-I

Grace M. Hopper working on the Harvard Mark-I, developed by IBM and Howard Aiken. The Mark-I remained in use at Harvard until 1959, even though other machines had surpassed it in performance, providing vital calculations for the navy in World War II.

Slide from “Introduction to Computing”

METU Computer Engineering

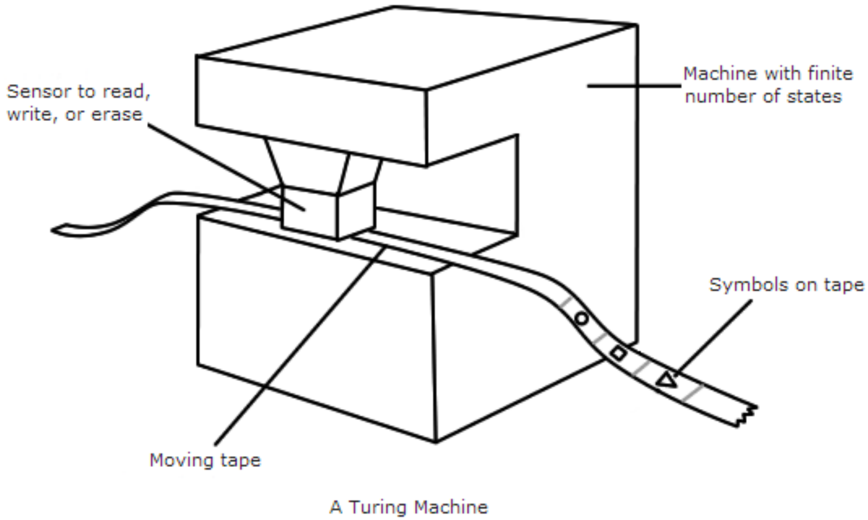


Programming the ENIAC

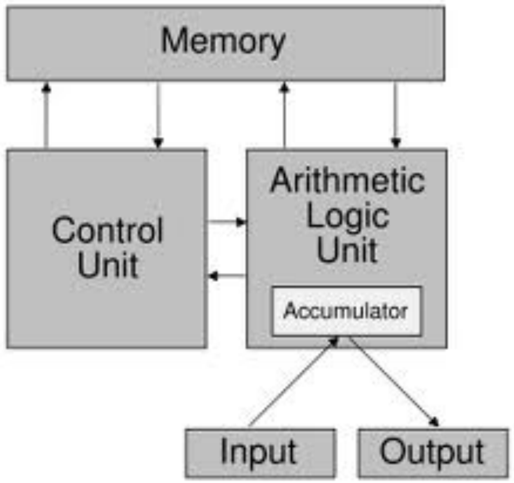
Slide from "Introduction to Computing"

History of Computation

- Read the reading material on this subject!
- And watch a video whose link we will post on cengclass.
  - Quiz from the reading material



Turing Machine

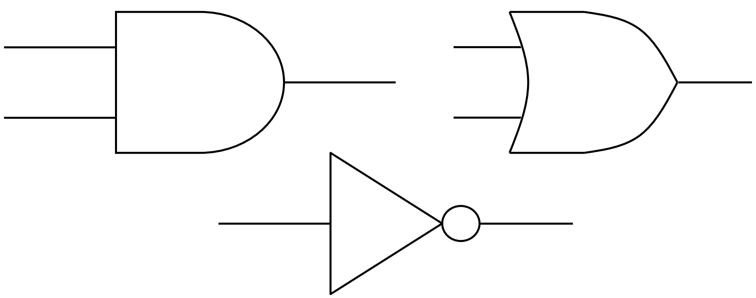


Von Neumann Architecture

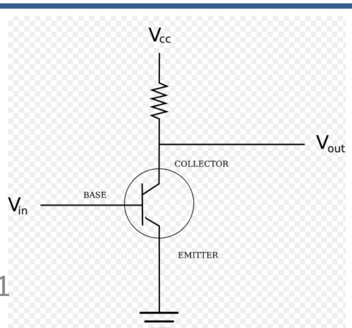
DIGITAL COMPUTATION

A computer

Devices



Gates

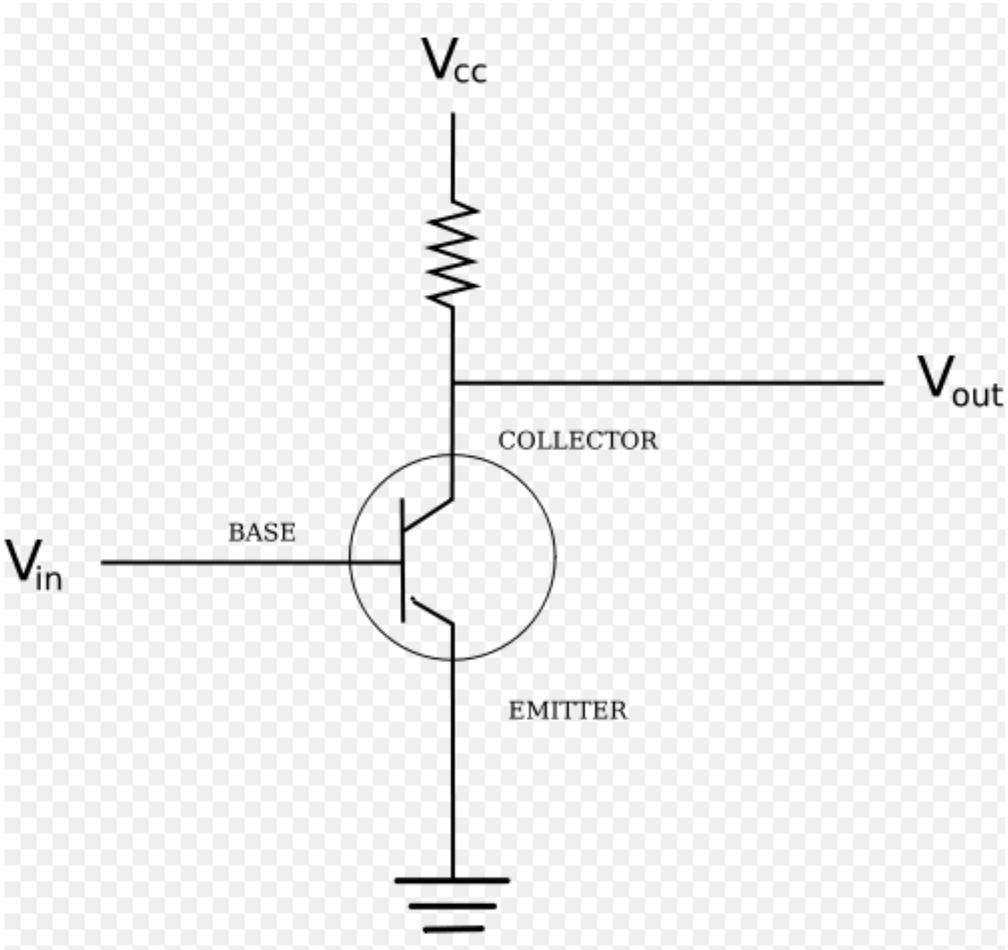


Transistors

Everything in a PC is Binary
… well, almost …

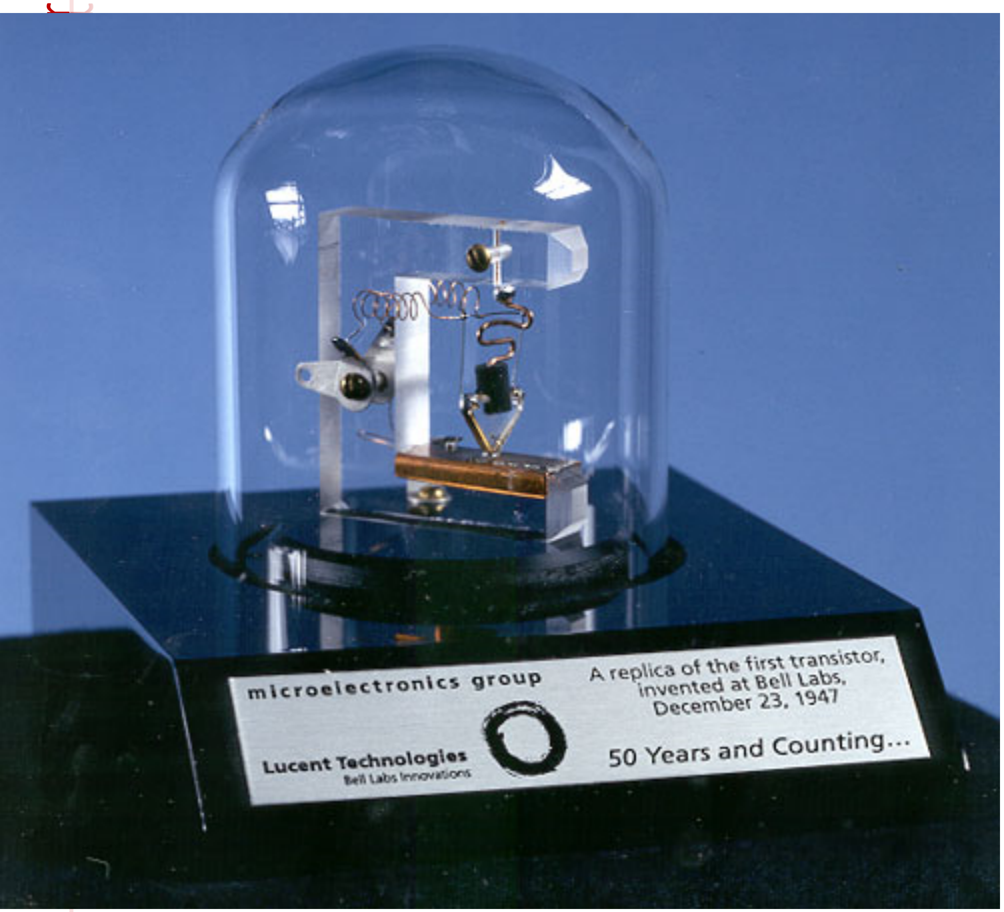
States of a Bit			
0	2+2=5 FALSE	OFF	LOW VOLTAGE
1	2+2=4 TRUE	ON	HIGH VOLTAGE

A transistor



This circuit functions as a switch. In other words, based on the *control* voltage, the circuit either passes Vin to output or not.

Examples of transistors



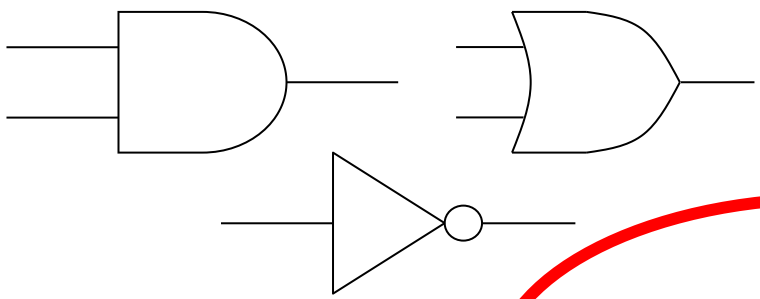
Replica of the first transistor

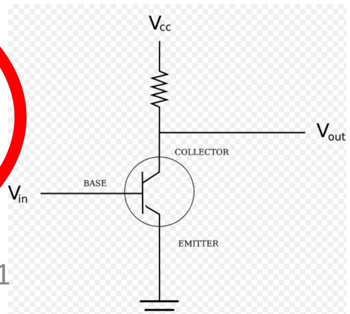
A set of transistors, depicting the fast change in technology.

A computer

Devices

Gates

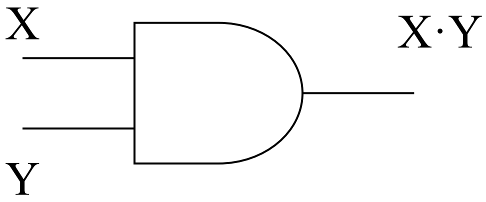




Transistors

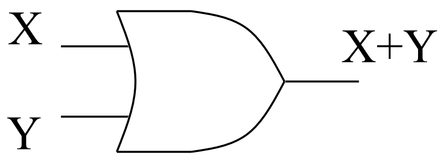
AND gate

X	Y	X·Y
0	0	0
0	1	0
1	0	0
1	1	1



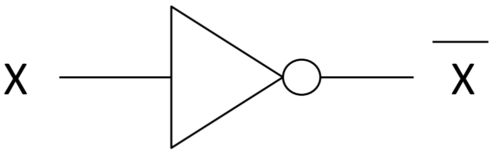
OR Gate

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1



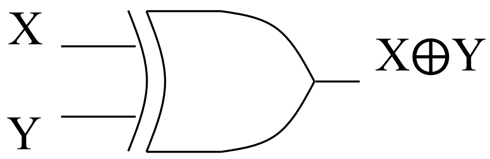
NOT Gate

X	$\overline{X}$
0	1
1	0



XOR Gate

X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0



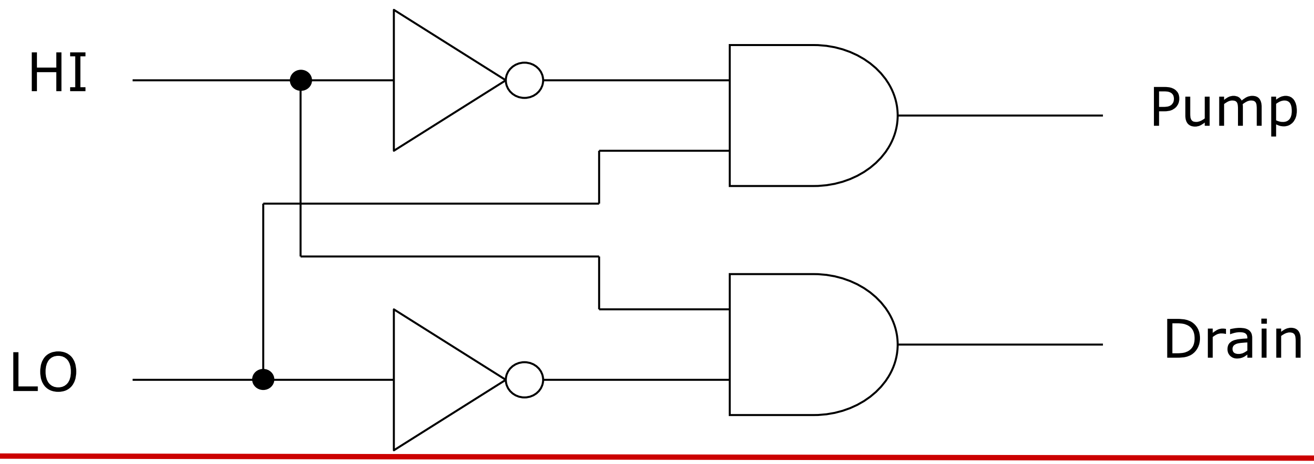
METU Computer Engineering

An example problem: Water Tank

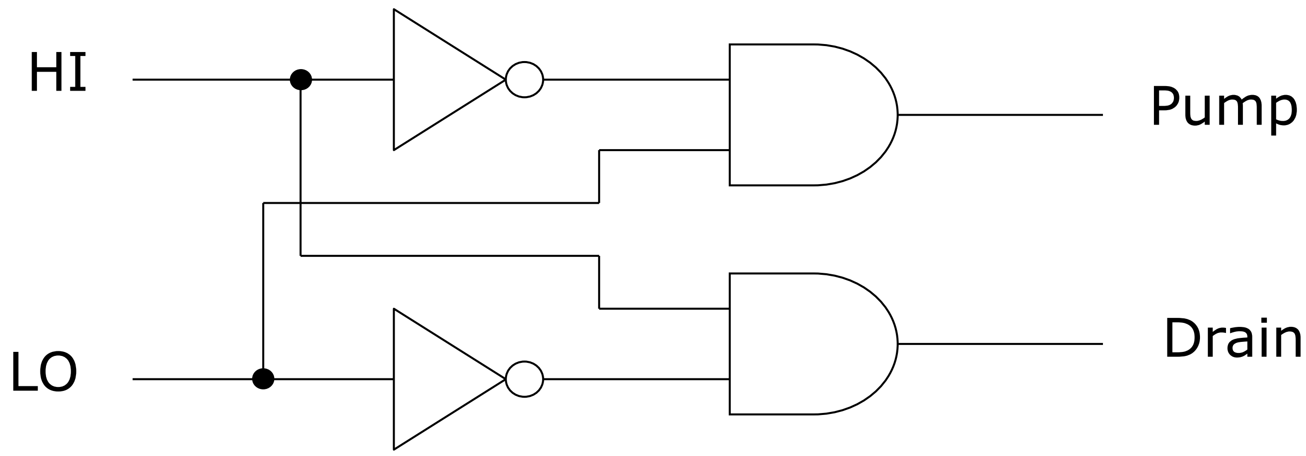
Truth Table Representation

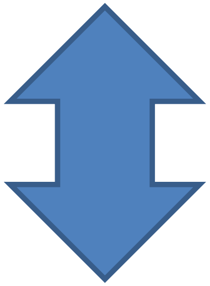
HI	LO	Pump	Drain	
0	0	0	0	→ Tank level is OK
0	1	**1**	0	→ Low level, pump more in
1	0	0	**1**	→ High level, drain some out
1	1	x	x	→ Inputs cannot occur

Schematic Representation



Boolean Logic/Algebra





Pump = HI'.LO
Drain = HI.LO'

Boolean formula describing the circuit.

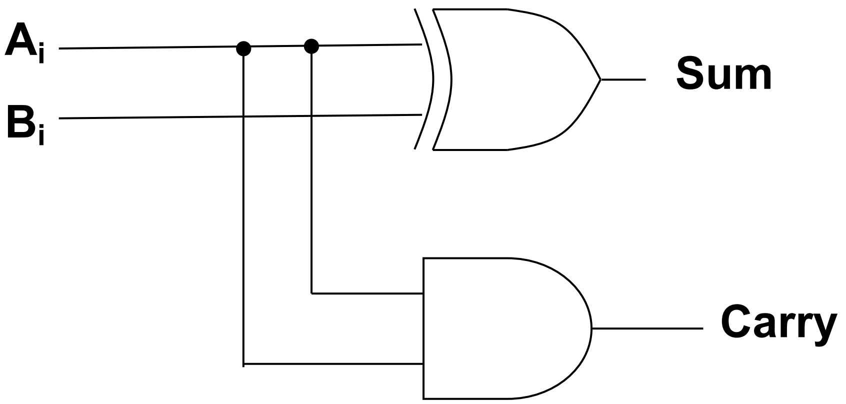
The binary addition

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,1 \\ \hline 10 \end{array}$$

Question (Binary notation) : 111010 + 11011 = ?

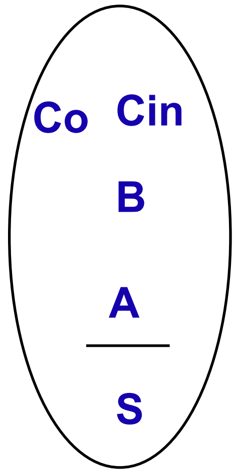
1-bit Half-adder

A_i	B_i	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



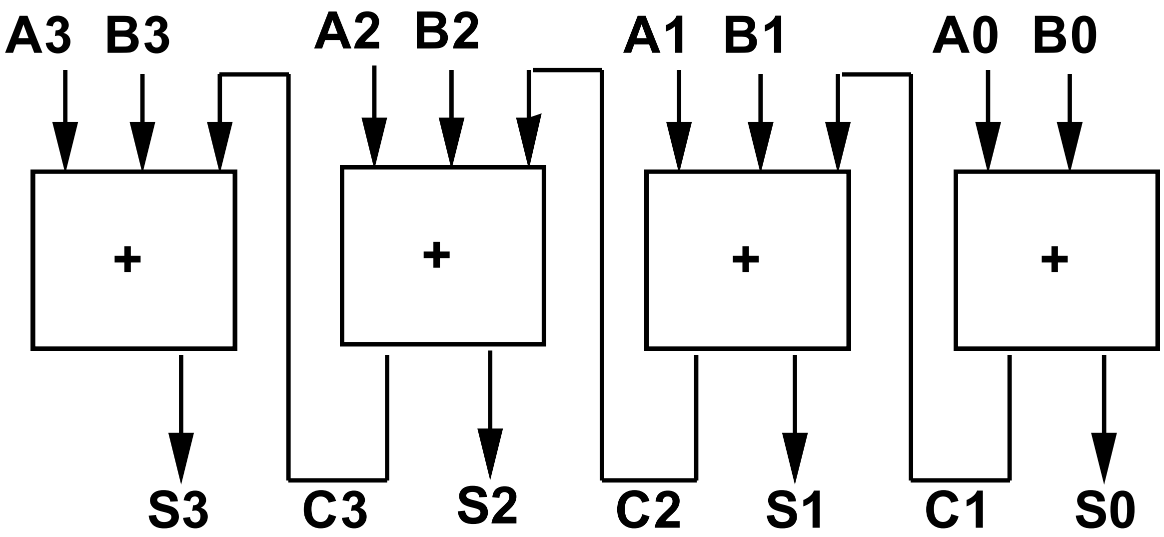
1-bit full-adder

```
      1
    0 0 1 1
+   0 0 1 0
  ---------
    0 1 0 1
```



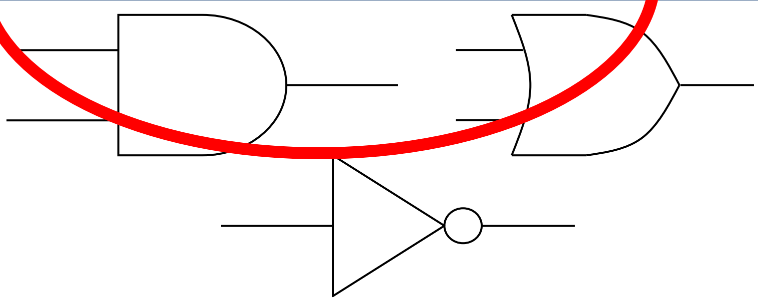
A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

N-bit Adder

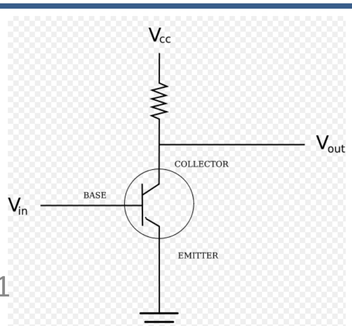


A computer

Devices



Gates



Transistors