Balamurugan Soundararaj

Postdoctoral Fellow - Data Analytics & Visualisation

@balaunsw s.bala@unsw.edu.au github.com/sbmkvp https://bala.sh

UNSW

Transport Planning

Retail Real Estate

Construction & Project Management Research

Geography and Urban informatics research

2005

2010

2015

2020

Bachelors in Urban Planning

Masters in Advanced Spatial Analysis & Visualisation

PhD in Geography

Problems

Little or **No** Computing Background

Rediscovering the methods on their own

Work on problems that are **already solved**

Introduction to Computer Science

Identifiers and primitive data types

Assignment, arithmetic, logical and relational operators

Expression and statements, Debugging

Flow of control: selection and repetition

Functions, parameters passing, call by value & reference

Object-oriented programming

1/2 dimensional arrays, strings and data structures

Pointers

Introduction to Computer Science

Είναι πλέον κοινά παραδεκτό ότι ένας
αναγνώστης αποσπάται από το
περιεχόμενο που διαβάζει, όταν εξετάζει τη διαμόρφωση
μίας σελίδας. Η ουσία
της χρήσης του Lorem Ipsum
είναι ότι έχει λίγο-πολύ
μία ομαλή κατανομή γραμμάτων,

A non computer scientist's

Introduction to Computer Science

A non computer scientist's

Introduction to Computer Science

Context - Where does all of this come from?

Utility - For what these things are used?

Relevance - How can I use these for my purposes

Resources - Where can I learn more?

History of Computing

Computer

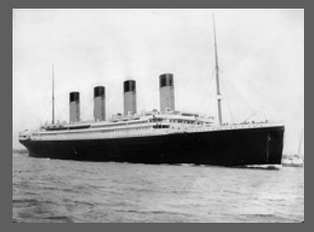
Computer

History of Computer Science

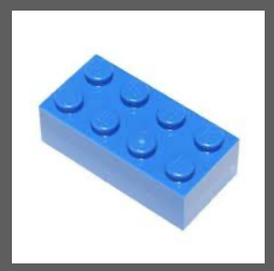
Technological Advancements






Layers of Abstraction



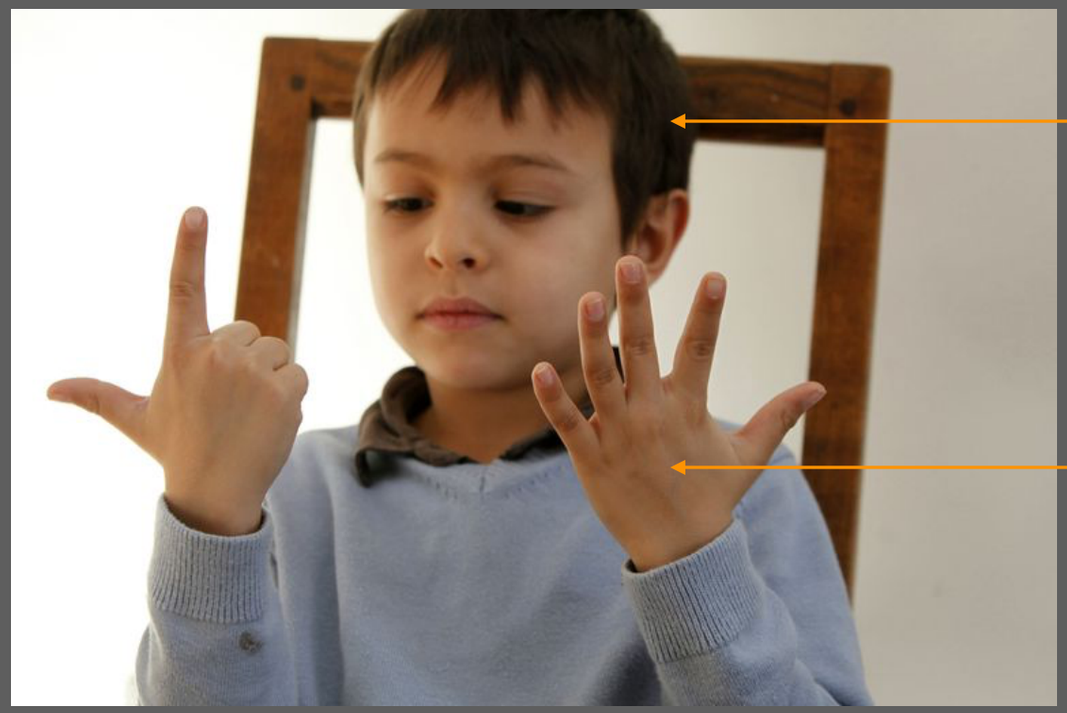




Introduction to Computer Science

Folding or extending fingers

Processing
Add or subtract



Memory
Registers A and B

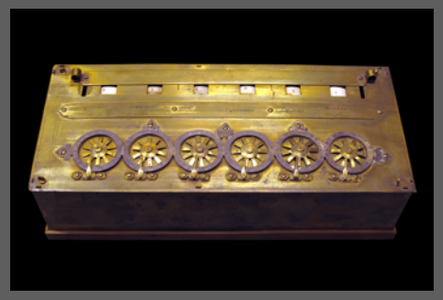
Even the most powerful complex computers
of the world are literally doing this at their core

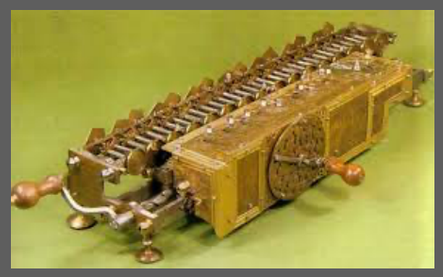
Numbers based computing

Hypothetical

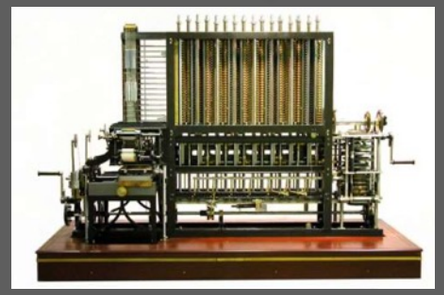
Marks (m)
Process of marking (p)

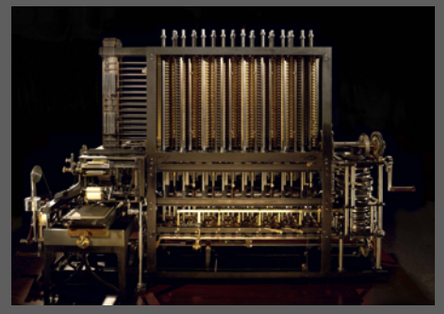
Position of Beads (m)
Moving the beads (p)



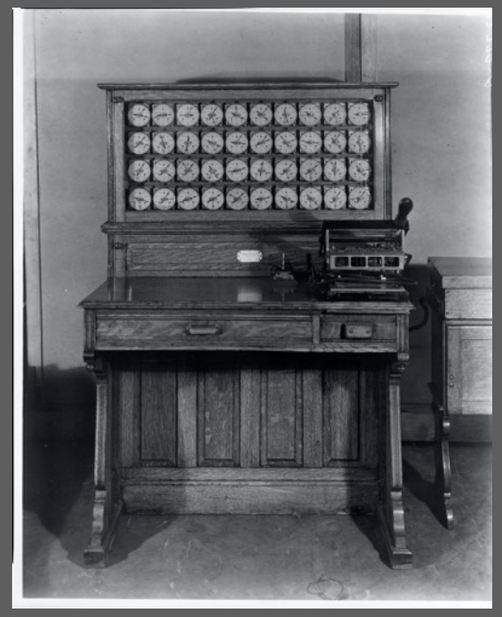


Position of wheels (m)
Rotating the wheels (p)





Position of wheels (m)
Rotating the wheels (p)



Punched Card(m)
Electromechanical wheels (p)

All of them count from 0-9 (decimal) system

They have difference in storage and processing

Boolean Arithmetic

Every number can be represented by true (1) and false (0)

Binary System!

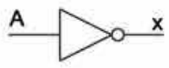
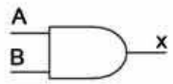
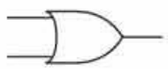
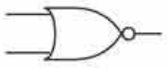
1	1
2	10
25	11001
1356	10101001100
65,535	1111111111111111

Boolean Arithmetic

Even arithmetic can be reduced to the basic logic of

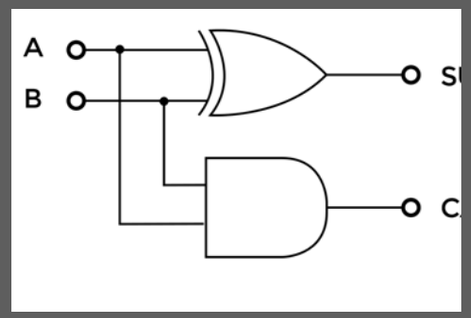
TRUE FALSE AND OR

Logic gates

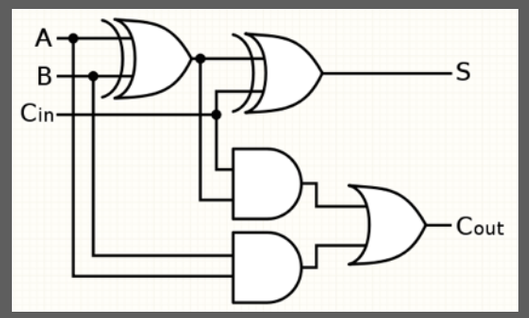
Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							
Truth Table	A X 0 1 1 0	B A X 0 0 0 0 1 0 1 0 0 1 1 1	B A X 0 0 1 0 1 1 1 0 1 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 1	B A X 0 0 1 0 1 0 1 0 0 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 0	B A X 0 0 1 0 1 0 1 0 0 1 1 1

Arithmetic

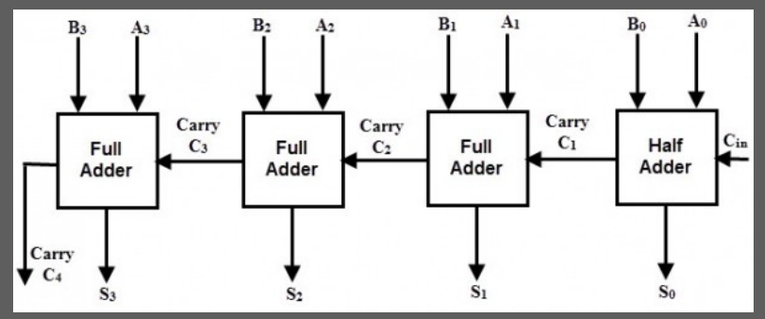
Half Adder



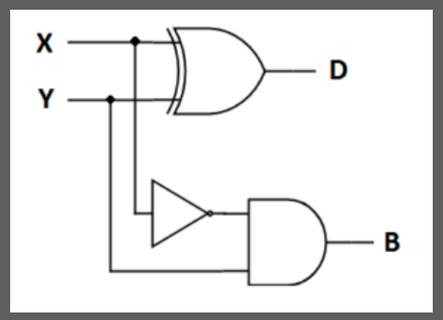
Full Adder



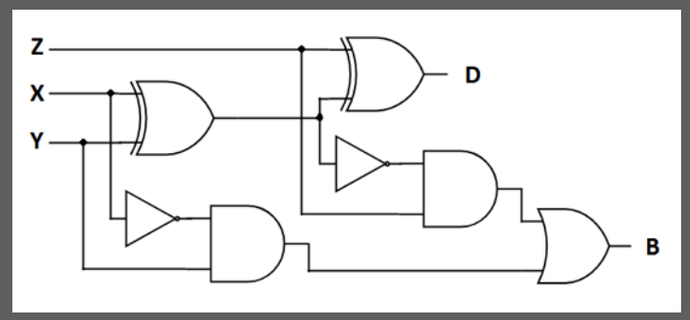
4 bit Adder



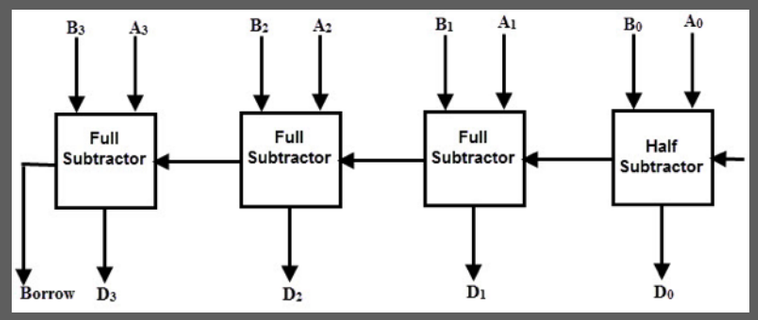
Half Subtractor

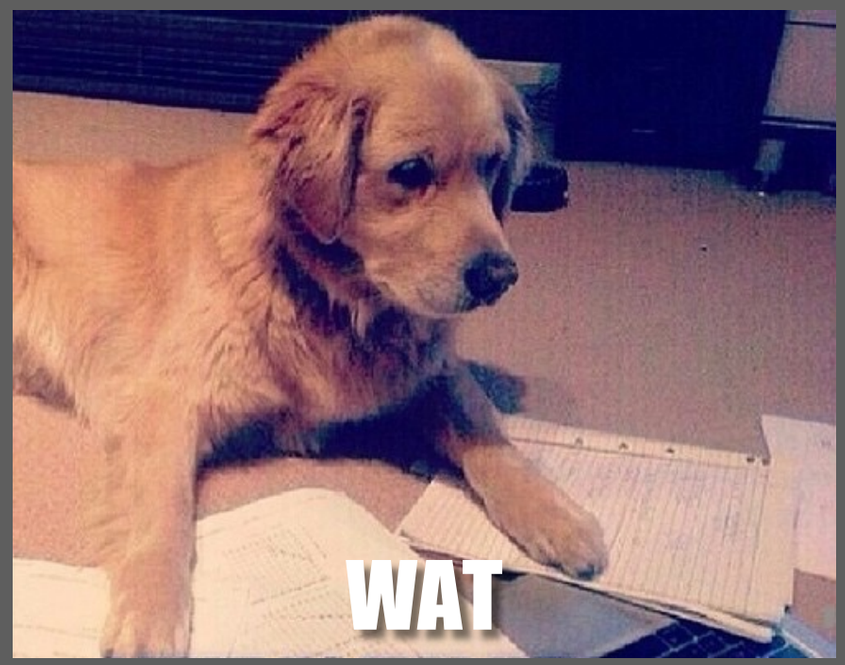


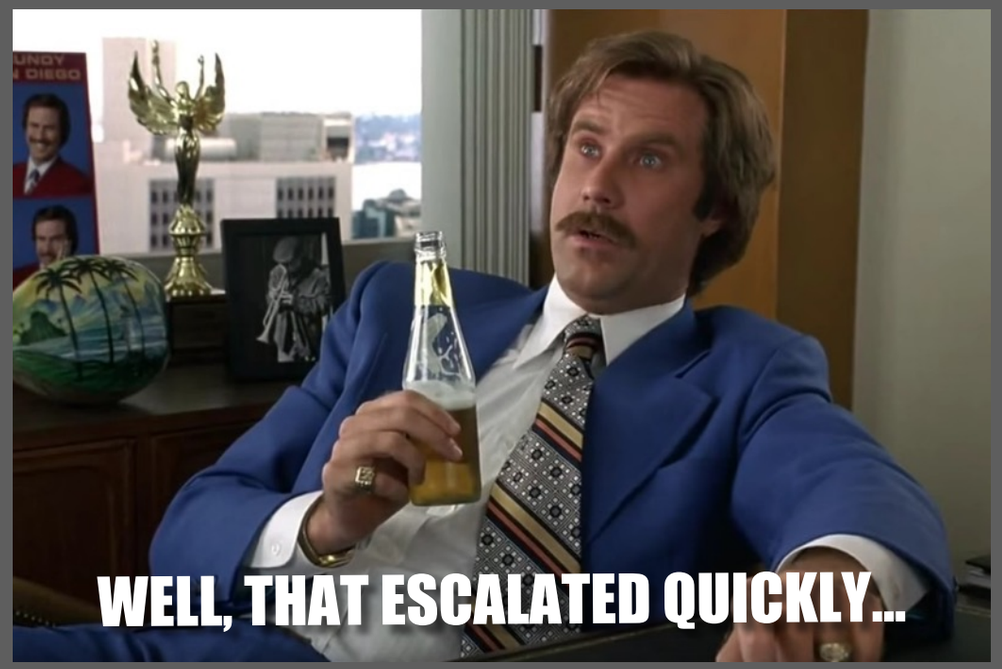
Full Subtractor



4 bit Subtractor







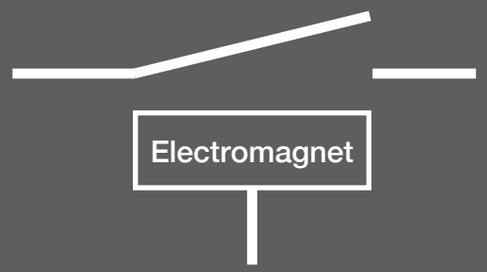
Don’t worry. Nobody gets all the details…

Basic units are true, false and some logic

Numbers can be represented by combining true or false.

Arithmetic can be represented by combining logic.

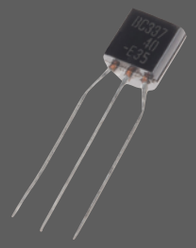
Implementing all the above



Mechanical Relay

Thermionic Valve

Vacuum Tube



Transistors

All of them are essentially AND gates

They can be combined for constructing Processing and Memory

CPU RAM

A very very Simplified

Anatomy of a CPU

Registers

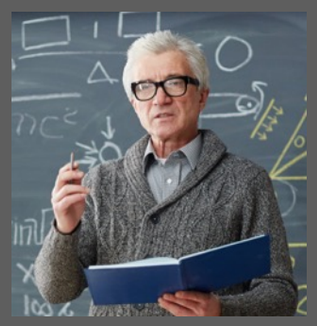
- Register A
- Register B
- Register C
- Register D

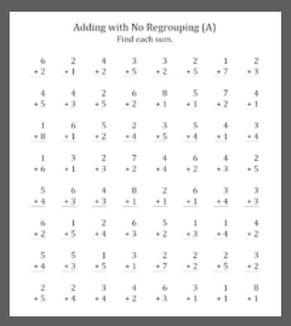
Arithmetic Logic Unit

Control Unit

Memory (RAM)

1000 0111
1100 1000
0011 1010
1000 1001
1100 1001
0011 1011
0101 0001
1101 0010
0000 0000
0000 0000





Computer Program!!

```
1000 0111
1100 1000
0011 1010
1000 1001
1100 1001
0011 1011
0101 0001
1101 0010
0000 0000
0000 0000
```

Every program ever written is read by the CPU in this format

MACHINE CODE!

Layers of Abstraction

```
1000 0111
1100 1000
0011 1010
1000 1001
1100 1001
0011 1011
0101 0001
1101 0010
0000 0000
0000 0000
```

Machine Code

ASSEMBLER

```
global  _start
        section .text
_start: mov rax, 1
        mov rdi, 1
        mov rsi, message
        mov rdx, 13
        syscall
        mov rax, 60
        xor rdi, rdi
        syscall
        section .data
message:db "Hello, World", 10
```

Assembly

COMPILER

```
# include <stdio.h>

int main()
{
   printf("Hello, World!”);
   return 0;
}
```

Higher-level Language

Layers of Abstraction (Processing)

Machine Code
Strictly numerical, understood by the processor and is very fast

Assembly
Much more readable format, converted into machine code by the assembler

C, cobol, fortran
Higher level language with loads of features, compiler converts this into assembly

C++, Objective C
Very similar to C language but with object oriented programming concepts.

Java
Object oriented language with a C like syntax. Compiles into byte code which runs in a Virtual machine.

Python, Perl
General purpose scripting languages, Used with operating systems for general tasks

R, Javascript, SQL
Specialised scripting languages, Usually used for a specific purpose and works within a framework

HTML, XML, Ps
Markup languages used for displaying things in a certain way.

Which is the **best** language?

readability, efficiency, data structure, available ecosystem, base use case, paradigm

A concise Guide to

Programming Languages

Language	Most Probable Scenario
Machine code	You are a robot
Assembly	Either you are developing embedded systems or writing a high performace software on a browser.
C	Developing an operating system or writing software for embedded systems/ IoT hardware or writing a package for python or R
C++	You are developing a desktop application
C#	You are developing a windows desktop application
Objective C	You are developing an iOS app
Java	You are developing an Android app or your company has already paid a tonne of money to Oracle
R	You are data scientist with statistics background, You deal with maps and geo data and you cannot replace ggplot.

Language	Most Probable Scenario
Python	You are a data scientist with computer science background, working fields related to machine learning and AI
SQL	You are using a relational database
Perl	You are working with NLP or a old fashioned sysadmin
Bash	You are working on a linux/unix system/ server or you are copying instructions from web
Javascript	You are developing websites and other interactive stuff.
Scala	You are dealing with a LOT of data.
Haskell	You want to do proper functional programming and you like math a lot
COBOL	You are maintaining a very old mainframe or you are a bored retiree with loads of money :D
Fortran	You are a physicist doing simulations in old software or developing packages for R/Python.

Layers of Abstraction (Storage)

Raw data

Series of 0s and 1s on
a storage medium

File System

The disk is partitioned and in
Each partition data is written in a
Certain method

Databases

Adds a layer of strict structure to
the filesystem. Improves the efficiency of the
Storage and retrieval but sacrifices flexibility

Distributed File System

These are filesystems which reside
In multiple systems but behaves as one.

Which is the **best** way to store data?