

CS1100 – Introduction to Programming

CS1100 – Introduction to Programming

Instructor:

Prashanth L.A. (SSB 314, prashla@cse.iitm.ac.in)

CS1100 – Introduction to Programming

Instructor:

Prashanth L.A. (SSB 314, prashla@cse.iitm.ac.in)

Lectures : Three per week at CS 15 – F1 Slot

- Wed. 11:00am – 11:50am
- Thu. 09:00am – 09:50am
- Fri. 08:00am – 08:50am

CS1100 – Introduction to Programming

Instructor:

Prashanth L.A. (SSB 314, prashla@cse.iitm.ac.in)

Lectures : Three per week at CS 15 – F1 Slot

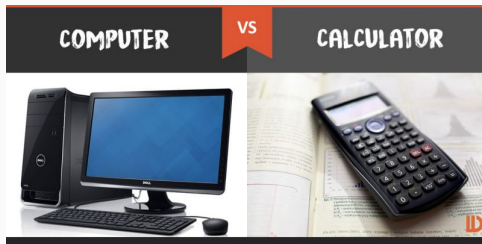
- Wed. 11:00am – 11:50am
- Thu. 09:00am – 09:50am
- Fri. 08:00am – 08:50am

Lab : One session per week at DCF (CSE Dept) – R1 Slot

Wed. 1400-1650

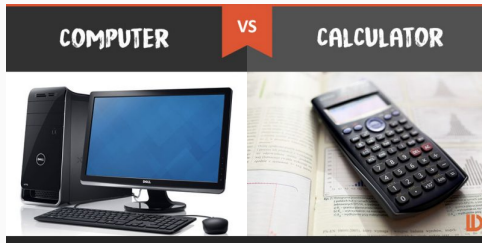
- Introduction to Computing and Computers.
- Programming (in C).
- Exercises and examples from various domains.
- Problem solving using computers.

A Calculator Calculates... and a Computer ...



- Calculators are single-purpose devices that perform mathematical operations input by the user.

A Calculator Calculates... and a Computer ...



- Calculators are single-purpose devices that perform mathematical operations input by the user.
- Computers are calculators that have vastly expanded capabilities, and are often called "general purpose computing devices".

Goal for today – have fun!

Observe the following patterns:

**

* *

* *

* *

Goal for today – have fun!

Observe the following patterns:

*****		*****
*****	**	* *
*****	*****	* *
*****	*****	* *
*****	*****	*****

- It is very easy to draw these patterns on paper.

Goal for today – have fun!

Observe the following patterns:

*****		*****
*****	**	* *
*****	****	* *
*****	*****	* *
*****	*****	*****

- It is very easy to draw these patterns on paper.
- How would you describe the same to a friend on the phone?

Describing a pattern

- How do you communicate?
- Use commonly understood commands.

Describing a pattern

- How do you communicate?
 - Use commonly understood commands.
 - draw a star.
 - go to new line.
 - repeat a set of commands k times.
-

Describing a pattern

- How do you communicate?
 - Use commonly understood commands.
 - draw a star.
 - go to new line.
 - repeat a set of commands k times.
-

- repeat 8 times
 - draw a star.
- go to new line.
- repeat 8 times
 - draw a star.

Can you describe all patterns in that list?

*****		*****
*****	**	* *
*****	****	* *
*****	*****	* *
*****	*****	*****

-
- draw a star.
 - go to new line.
 - repeat a set of commands k times.

Can you describe all patterns in that list?

[illegible]

- draw a star.
- go to new line.
- repeat a set of commands k times.
- **move right** (without drawing a star).

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
(and also inform the friend of its meaning).

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
(and also inform the friend of its meaning).
- Recall: A **Program** is a **precise** description of steps that we want to perform on the data.

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
(and also inform the friend of its meaning).
- Recall: A **Program** is a **precise** description of steps that we want to perform on the data.
- So, is the above a "program"?

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
(and also inform the friend of its meaning).
- Recall: A **Program** is a **precise** description of steps that we want to perform on the data.
- So, is the above a "program"? **Yes**. But the computer does not know the above language.

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
(and also inform the friend of its meaning).
- Recall: A **Program** is a **precise** description of steps that we want to perform on the data.
- So, is the above a "program"? **Yes**. But the computer does not know the above language.
- Is the above a "computer program"?

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
(and also inform the friend of its meaning).
- Recall: A **Program** is a **precise** description of steps that we want to perform on the data.
- So, is the above a "program"? **Yes**. But the computer does not know the above language.
- Is the above a "computer program"? **No**

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
(and also inform the friend of its meaning).
- Recall: A **Program** is a **precise** description of steps that we want to perform on the data.
- So, is the above a "program"? **Yes**. But the computer does not know the above language.
- Is the above a "computer program"? **No**
- Goal of the course: learn to program the computer to perform different tasks.

What Languages are Computers Taught with ...

Programming languages : C, C++, Java, Python ...

- the languages that the computers are apriori trained on.
(how? - for later !).
- means of communication with a computer.

What Languages are Computers Taught with ...

Programming languages : C, C++, Java, Python ...

- the languages that the computers are apriori trained on.
(how? - for later !).
- means of communication with a computer.
- In this course : [The C Programming Language](#)
 - Developed by Dennis Ritchie (1969 – 1973).
 - One of the most popular programming language.
 - Used to “write” large softwares, scientific computing etc.

What Languages are Computers Taught with ...

Programming languages : C, C++, Java, Python ...

- the languages that the computers are apriori trained on.
(how? - for later !).
- means of communication with a computer.
- In this course : **The C Programming Language**
 - Developed by Dennis Ritchie (1969 – 1973).
 - One of the most popular programming language.
 - Used to “write” large softwares, scientific computing etc.
- To be able to write programs in C, **we need to learn the language.**

What Languages are Computers Taught with ...

Programming languages : C, C++, Java, Python ...

- the languages that the computers are apriori trained on.
(how? - for later !).
- means of communication with a computer.
- In this course : **The C Programming Language**
 - Developed by Dennis Ritchie (1969 – 1973).
 - One of the most popular programming language.
 - Used to “write” large softwares, scientific computing etc.
- To be able to write programs in C, **we need to learn the language.**
- That is the goal of this course.

Summarizing ...

- Programming is fun.

Summarizing ...

- Programming is fun.
- Programming is useful - computational techniques to simulate, visualize and conclude without actually making the physical system.

Summarizing ...

- Programming is fun.
- Programming is useful - computational techniques to simulate, visualize and conclude without actually making the physical system.
- Programming is the Designer and the Programmer of a company. You need to know how to manage both!
- The Designer designs – MUST be accurate. The product must be relevant – so we need a CTO too.
- The programmer converts the design verbatim to a program in a language that the computer understands! S/he is responsible for efficient programming too.

Summarizing ...

- Programming is fun.
- Programming is useful - computational techniques to simulate, visualize and conclude without actually making the physical system.
- Programming is the Designer and the Programmer of a company. You need to know how to manage both!
- The Designer designs – MUST be accurate. The product must be relevant – so we need a CTO too.
- The programmer converts the design verbatim to a program in a language that the computer understands! S/he is responsible for efficient programming too.
- "Why this course", "What is in the course".

Books for the course

- Paul Deitel and Harvey Deitel. C: How to Program.
- V. Rajaraman: Computer Programming in C.
- R. G. Dromey: How to Solve It By Computer?
- Kernighan and Ritchie: The C Programming Language.

Evaluation for the course

- Two Quizzes – 24 marks: 12 for Q1 and 12 for Q2.
- Class participation – 6 marks.
- Lab tests – 30 marks.
- End of Semester Exam – 40 marks.
- Attendance – taken in the lab and in lectures.

- Strictly no cell phone usage in class
 - Keep your cell phone turned off
(Not even in silent mode)
 - No SMS, chat etc.
- Cell phones will be confiscated if there are violations
- Returned only after 2 weeks
- Repeat violations
Students will be sent to the Dean (Acad)

Acknowledgements

- Slides for the course are based on material prepared by faculty of CSE department IITM.
- Ideas will also be drawn from a book by Prof. Abhiram Ranade (IITB) (Introduction to programming using C++).
- All images – courtesy Google Images.
- This applies for all slides throughout the course.

- Read the remainder of the slides!!
- Learn to use a text editor: example: **GVim**,...

Rest of the presentation: Overview

- Turtle programs!!

Rest of the presentation: Overview

- Turtle programs!!
- A brief history about computers.

Rest of the presentation: Overview

- Turtle programs!!
- A brief history about computers.
- What is a computer made of?

Rest of the presentation: Overview

- Turtle programs!!
- A brief history about computers.
- What is a computer made of?
 - Do we need to know **internals** of a computer to be able to program it?

Rest of the presentation: Overview

- Turtle programs!!
- A brief history about computers.
- What is a computer made of?
 - Do we need to know **internals** of a computer to be able to program it?
- How does a computer perform so many **diverse** tasks (number crunching, weather prediction, playing chess, ...)?

Rest of the presentation: Overview

- Turtle programs!!
- A brief history about computers.
- What is a computer made of?
 - Do we need to know **internals** of a computer to be able to program it?
- How does a computer perform so many **diverse** tasks (number crunching, weather prediction, playing chess, ...)?
 - Convert every task into a task on **numbers**.
 - How to represent numbers on computers? (**tomorrow !**)

More on the Turtle Language

Question : How will we draw a pentagon.

```
#include simplecpp
main_program
{
    turtleSim();
    forward(100); left(90);
    forward(100); left(90);
    forward(100); left(90);
    forward(100);
    wait(5);
}
```

```
#include simplecpp
main_program
{
    turtleSim();
    forward(100); left(72);
    forward(100); left(72);
    forward(100); left(72);
    forward(100); left(72);
    forward(100);
    wait(5);
}
```

Neater way to draw a Decagon

Turtle knows more ...

- `forward(n)`
- `right(d)`
- `left(d)`
- `wait(t)`
- `repeat(k) { commands }`
repeats the commands k times.

```
#include <simplecpp>
main_program
{
    turtleSim();
    repeat(10)
    {
        forward(100);
        left(36);
        wait(1);
    }
    wait(5);
}
```

More fun with Turtle ...

What will the following program draw?

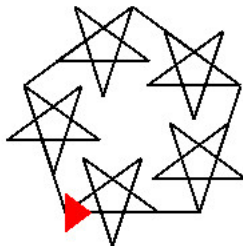
```
#include <simplecpp>
main_program
{
    turtleSim();
    left(72);
    repeat(5)
    {
        forward(200);
        wait(1);
        left(144);
    }
    wait(20);
}
```

More fun with Turtle ...

What will the following program draw?

```
#include <simplecpp>
main_program
{
    turtleSim();
    left(72);
    repeat(5)
    {
        forward(200);
        wait(1);
        left(144);
    }
    wait(20);
}
```

Make the turtle draw this !



Turtle knows more ...

- Turtle can print messages. `cout << "Hello World";`
- Turtle can wait for an input to be typed by you and use it for the drawing (computation). Command is : `cin >> n;` where `n` is a "variable".

Turtle knows more ...

- Turtle can print messages. `cout << "Hello World";`
- Turtle can wait for an input to be typed by you and use it for the drawing (computation). Command is : `cin >> n;` where `n` is a "variable".
- `penUp()`: Causes the pen to be raised.
- `penDown()`: Causes the pen to be lowered.
- `sqrt(x)` : square root of `x`.
- `sine(x)`, `cosine(x)`, `tangent(x)` : trigonometric functions, `x` is in degrees.

Text-only Turtle

Predict the output:

```
#include <simplecpp>
main_program
{
    cout << "a";
    repeat(5)
    {
        cout << "b";
        repeat(2){ cout << "c"; }
        cout << "d";
    }
}
```

Text-only Turtle

Predict the output:

```
#include <simplecpp>
main_program
{
    cout << "a";
    repeat(5)
    {
        cout << "b";
        repeat(2){ cout << "c"; }
        cout << "d";
    }
}
```

The program will print
abccdbccdbccdbccdbccd

Drawing a polygon with "given" number of sides

```
#include <simplecpp>
main_program
{
    turtleSim();
    cout << "How many sides?";
    int nsides;
    cin >> nsides;
    repeat(nsides){
        forward(100);
        right(360.0/nsides);
        wait(1);
    }
    wait(10);
}
```

Drawing a polygon with "given" number of sides

```
#include <simplecpp>
main_program
{
    turtleSim();
    cout << "How many sides?";
    int nsides;
    cin >> nsides;
    repeat(nsides){
        forward(100);
        right(360.0/nsides);
        wait(1);
    }
    wait(10);
}
```

cout << msg; : Print message msg on the screen.

Drawing a polygon with "given" number of sides

```
#include <simplecpp>
main_program
{
    turtleSim();
    cout << "How many sides?";
    int nsides;
    cin >> nsides;
    repeat(nsides){
        forward(100);
        right(360.0/nsides);
        wait(1);
    }
    wait(10);
}
```

cout << msg; : Print message msg on the screen.

int nsides; : "Reserve a space in the "blackboard" in which I will store some integer value, and call that cell nsides".

Drawing a polygon with "given" number of sides

```
#include <simplecpp>
main_program
{
    turtleSim();
    cout << "How many sides?";
    int nsides;
    cin >> nsides;
    repeat(nsides){
        forward(100);
        right(360.0/nsides);
        wait(1);
    }
    wait(10);
}
```

`cout << msg; : Print message msg on the screen.`

`int nsides; : "Reserve a space in the "blackboard" in which I will store some integer value, and call that cell nsides".`

`cin >> nsides; : Read an integer value from the keyboard and put it in the cell nsides.`

Drawing a polygon with "given" number of sides

```
#include <simplecpp>
main_program
{
    turtleSim();
    cout << "How many sides?";
    int nsides;
    cin >> nsides;
    repeat(nsides){
        forward(100);
        right(360.0/nsides);
        wait(1);
    }
    wait(10);
}
```

cout << msg; : Print message msg on the screen.

int nsides; : "Reserve a space in the "blackboard" in which I will store some integer value, and call that cell nsides".

cin >> nsides; : Read an integer value from the keyboard and put it in the cell nsides.

360.0/nsides : represents the value obtained after dividing 360 by whatever is in nsides.

Drawing a polygon with "given" number of sides

```
#include <simplecpp>
main_program
{
    turtleSim();
    cout << "How many sides?";
    int nsides;
    cin >> nsides;
    repeat(nsides){
        forward(100);
        right(360.0/nsides);
        wait(1);
    }
    wait(10);
}
```

cout << msg; : Print message msg on the screen.

int nsides; : "Reserve a space in the "blackboard" in which I will store some integer value, and call that cell nsides".

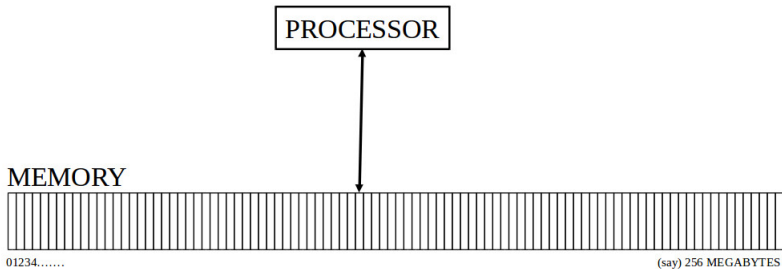
cin >> nsides; : Read an integer value from the keyboard and put it in the cell nsides.

360.0/nsides : represents the value obtained after dividing 360 by whatever is in nsides

A few general ideas ...

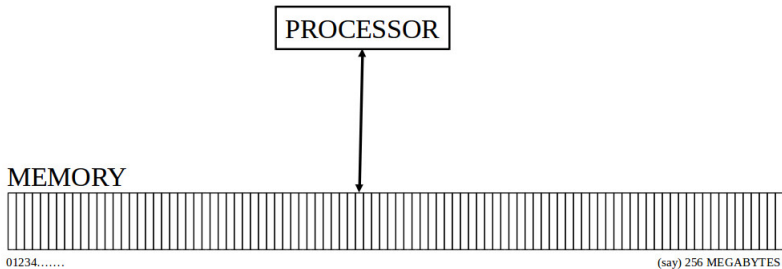
- *Control is at statement w*: Computer is currently executing statement w .
- *Control flow*: The order in which statements get executed. Execution starts at top and goes down. Retraced if there is a repeat statement.
- *Variable* used for storing data.
- *Computer memory*: blackboard
- *Variable* : Region on the board in which you can write a value.
- Variables have names, e.g. $nsides$. We can use the name to refer to the value written in the variable. Details later.

The Computing Machine



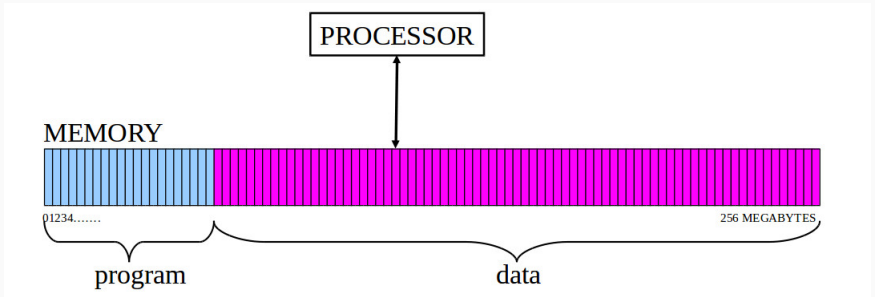
- The computer is made up of a processor and a memory.

The Computing Machine



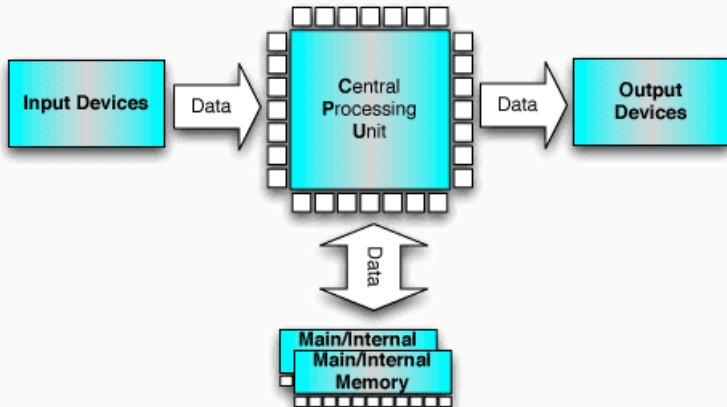
- The computer is made up of a processor and a memory.
- The memory can be thought of as a series of locations to store information.

The Computing Machine



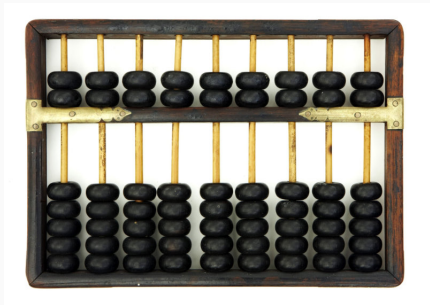
- A program is a sequence of instructions assembled for some given task.
- Most instructions operate on data.
- Some instructions control the flow of the operations.

The Computing Machine : von Neuman Architecture



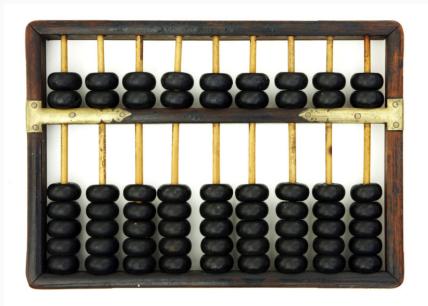
A brief look into the history...

From Abacus to Apple



- Counting frame.
- One of the earliest form of calculator.
- Still used by kids to do fast simple arithmetic.

From Abacus to Apple



- Counting frame.
 - One of the earliest form of calculator.
 - Still used by kids to do fast simple arithmetic.
-
- Followed by mechanical calculators by B. Pascal (1642), G. W. Leibniz (1671).
 - Used cogs / interlocking gears.
 - Performed $+$, $-$, $*$, $/$, $\sqrt{}$.
 - Leibniz is credited of creating the binary system.

Jaquard looms (1804)

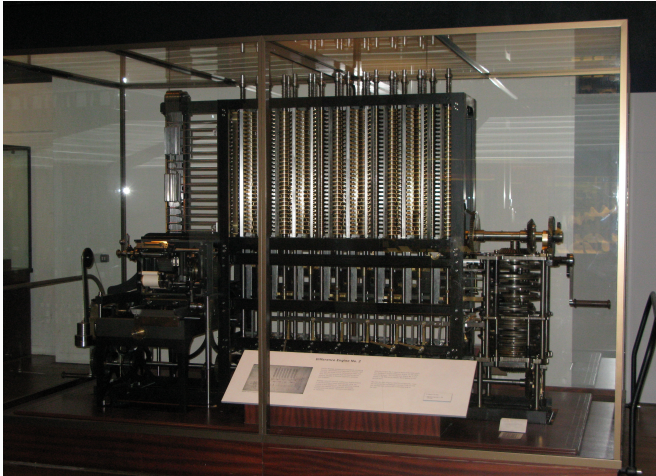


Charles Babbage (1791–1871)



- Regarded as the “Father of Computer”.
- Conceived of a machine that has all the parts of a modern computer, input, a memory, a processor, and an output (1850).

Difference Engine (1850)



Difference engine built from Babbage's design
(London Science Museum).

Ada Lovlace (1815–1852)



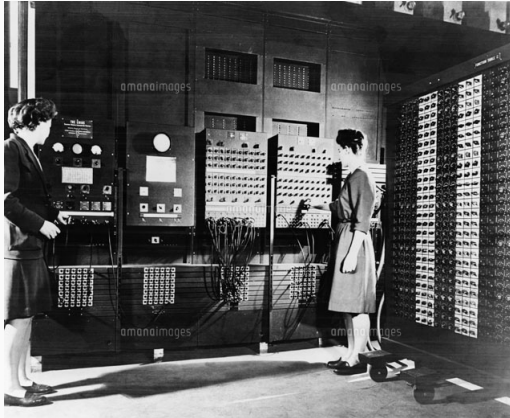
- “Wrote” the description of the mechanical computer of Babbage.
- Regarded as the first programmer ever.
- The programming language ADA is named after her.

Alan Turing (1912 – 1954)



- Father of Theoretical Computer Science (TCS) and Artificial Intelligence (AI).
- Turing machine – a model for a general purpose computer.
- Turing test – how intelligent is a machine?

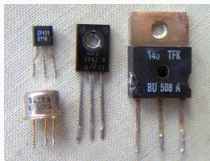
First Electronic Computer : ENIAC 1946



Electronic Numerical Integrator
And Calculator.

- 50,000 vacuum tubes, diodes, relays, resistors, capacitors.
- 5 million hand-soldered joints.
- Weighed 27 tons.
- Covered $167m^2$ area.
- Consumed 150 kW of power.

1946 – 1976

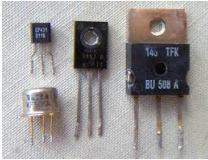


Transistors



Integrated Circuits

1946 – 1976



Transistors

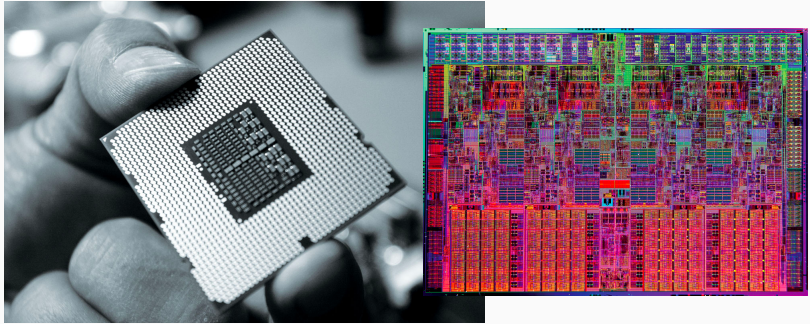


Integrated Circuits



Apple Macintosh

Today's World : Core i7 Processor



2008-15: Intel Core i7 Processor

Clock speed: > 2.5 GHz

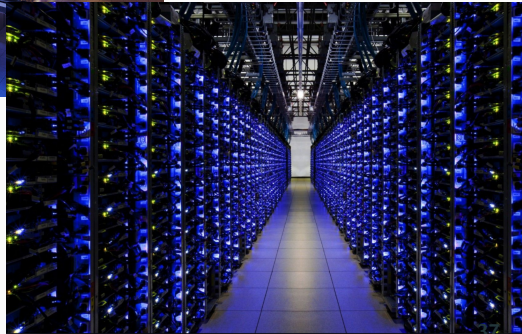
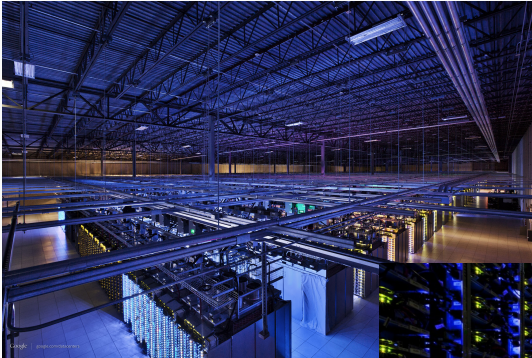
No. of Transistors: $0.731 - 1.3B$

Technology: $45 - 22nm$ CMOS Area: $263 - 181mm^2$.

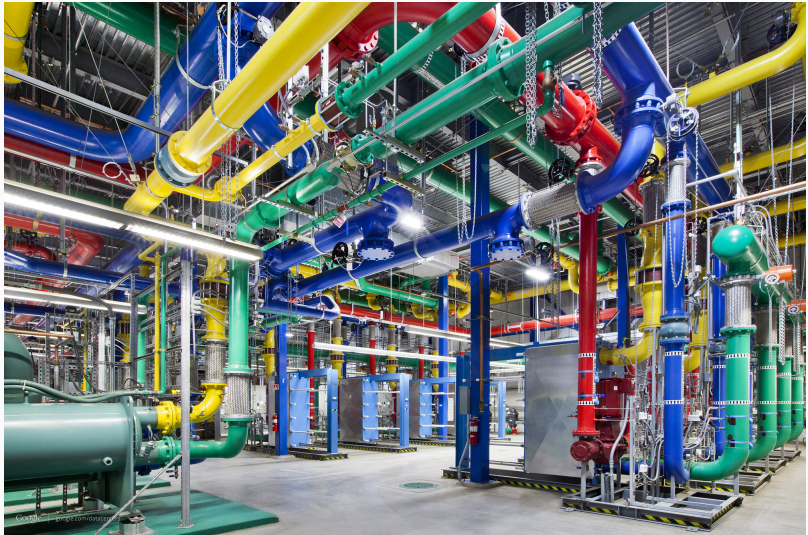
Modern computing devices



Data Centers: Processing/Storing Huge volume of data



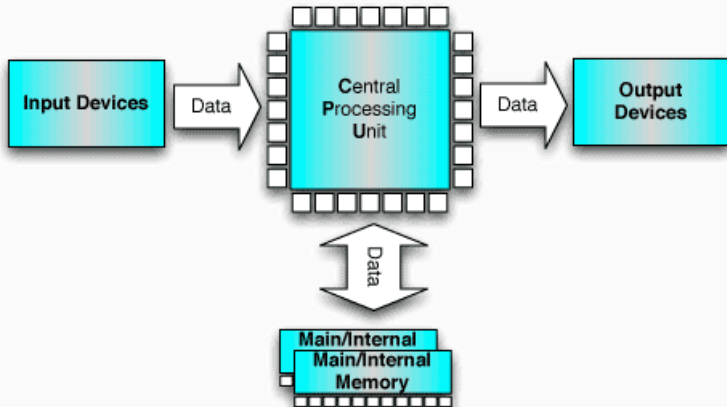
Even Cooling them is a big deal ...



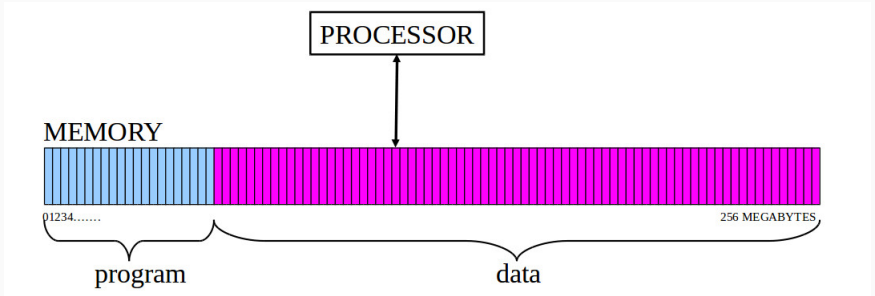
What do we learn from History?

- Untiring efforts by several pioneers (Babbage, Turing, Von Neuman, George Boole, Steve Jobs and many unnamed computer scientists and innovators) have given us the technology that we have today.
- Several failed attempts and criticisms for path breaking ideas.
- Most of the advances were made by necessity and driven by passion and immense hard work.

The Computing Machine : von Neuman Architecture



The Computing Machine



- A program is a sequence of instructions assembled for some given task.
- Most instructions operate on data.
- Some instructions control the flow of the operations.

Coming up...

- How does the computer execute a program?
- How does the computer represent data/programs?
- Introduction to C programming language.