IC150
Computation for Engineers

An Introduction to
Problem Solving using Computers

Timothy A. Gonsalves

Course Outline

- Introduction to Computing
- Problem solving using computers
- Exercises and examples are from engineering and sciences
- Tools: C, Scilab, OpenOffice Calc, Linux

Learning styles

- Deductive: from principles to examples
- Inductive: figure out principles by oneself

Learn how to learn

Professionalism

IIT BTech ➔ a proficient professional

Characteristics:

- See situations holistically
- See what is most important in a situation
- Perceive deviations from the normal pattern
- Quick decision-making
- Use maxims for guidance, whose meanings vary according to the situation

Requirements:

- Deep knowledge
- Broad and diverse knowledge
- Practical experience
- Punctuality
- Neatness
- Good English
- Respect for others
- Self-discipline

Evaluation

- Two Quizzes – 50 End Semester Exam – 50
- 5 marks extra for attendance
  - 0 missed = 5 marks
  - 1-2 missed = 4 marks
  - 3-4 missed = 3 marks
  - 5-6 missed = 2 marks
  - 7-8 missed = 1 mark

Minimum Attendance
>average in 2 quizzes => 50%
<average in one Quiz => 75%
<average in 2 Quizzes => 85%

- Grace marks for participation
  - Moodle discussion forum
- Professionalism

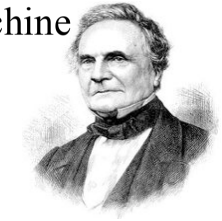
What is IC150 about?

- Computer and its components
- Computing – personal, distributed, parallel
- Problem Solving and Limitations of a Computer
- Programming Languages

What are common uses of a computer?

What IS a computer?

- A computer is a *machine*
- Something that operates *mechanically*
- But it is a *flexible* machine
- Its behaviour is controlled by a *program*
- A program is like a *spell* cast on a machine
- Programmers are like *wizards*
- Programs reside in the *memory* of the machine
  - *Charles Babbage (1791-1871):*
    - *"The stored program concept"*

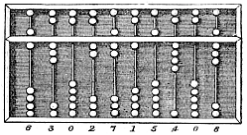


Common uses of a Computer

- As a tool for storing and retrieving information
  - Information regarding students entering IIT
- As a tool for providing services to customers
  - Billing, banking, reservation
- As a calculator capable of user-defined operations
  - Designing electrical circuit layouts
  - Designing structures
  - Non-destructive testing and simulation
- As an intelligent controller
  - Synchronised operation of traffic lights
  - Flying an aeroplane

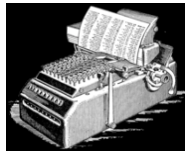
Early Computing Hardware

The Slide rule



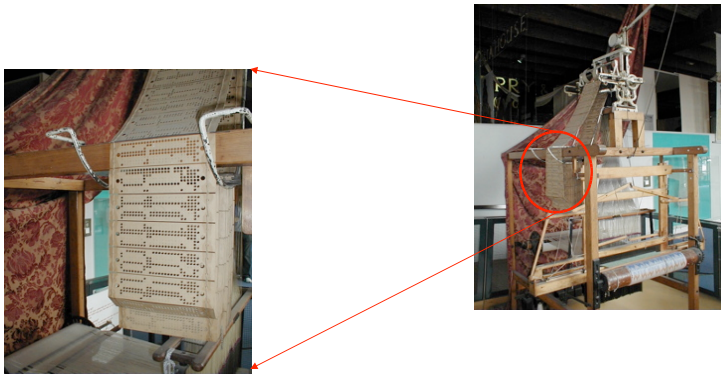
The Chinese Abacus

The gear replaced the beads in early mechanical calculators



"History of computing hardware"
From Wikipedia, the free encyclopedia

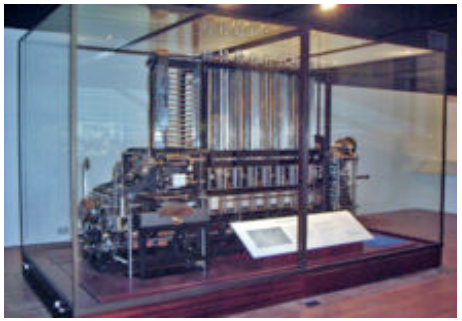
Jaquard looms



Used *punched cards* to weave *different patterns*

The Difference Engine

Part of Babbage's difference engine, assembled after his death by Babbage's son, using parts found in his laboratory.



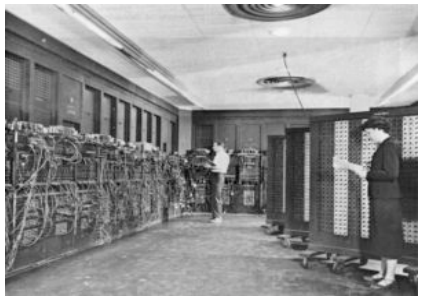
The London Science Museum's replica Difference Engine, built from Babbage's design.

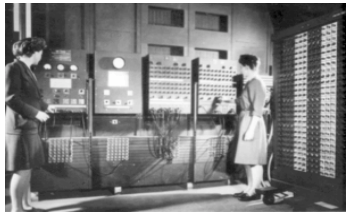
The First Programmer

Augusta Ada King, Countess of Lovelace (December 10, 1815 – November 27, 1852), born **Augusta Ada Byron**, is mainly known for having written a description of Charles Babbage's early mechanical general-purpose computer, the analytical engine.

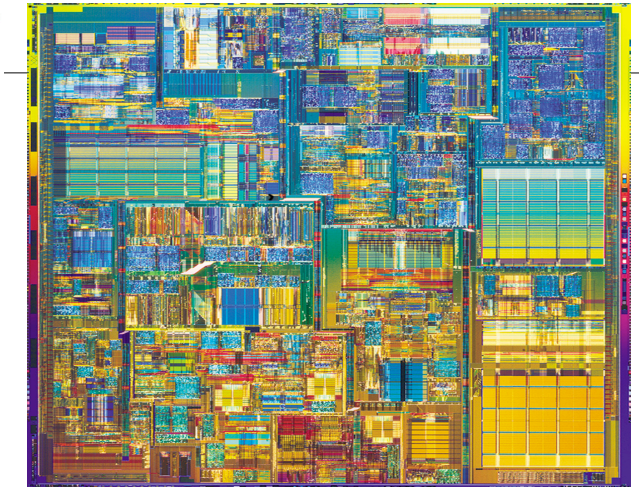
The programming language ADA is named after her

ENIAC – the first electronic computer



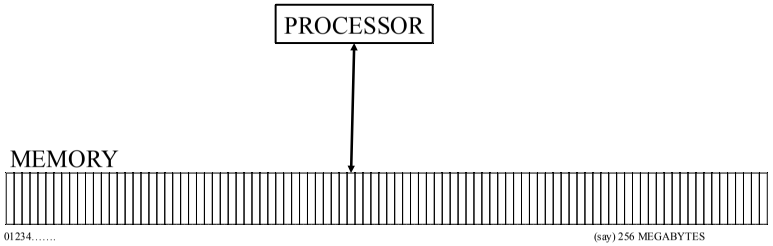


ENIAC was massive compared to modern PC standards:
17,468 vacuum tubes, 7,200 crystal diodes, 1,500 relays, 70,000 resistors, 10,000 capacitors, about 5 million hand-soldered joints.
Weighed 27 tons, 2.4 m by 0.9 m by 30 m, 167 m² floor space
150 kW of power



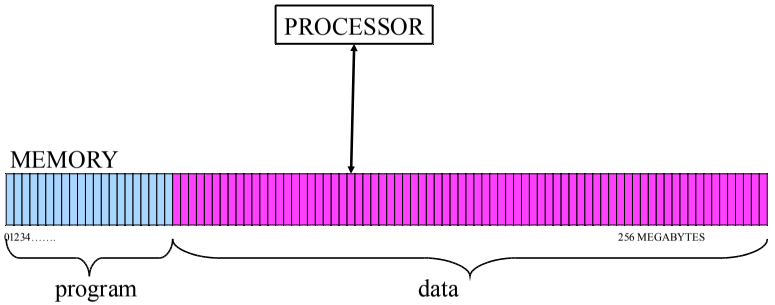
2000: Intel Pentium 4 Processor
Clock speed: 1.5 GHz
Transistors: 42 million
Technology: 0.18µm CMOS
Size: 1.22 cm square

The computing machine



The computer has a *processor* and a *memory*. The memory is a series of *locations* to store information.

The stored program *von Neumann* computer



- A *program* is a sequence of *instructions* for some task
- Most instructions operate on *data*
- Some instructions *control* the sequence of the instructions
- It is even possible to treat programs as data. By doing so a program could create another program, *or even modify itself*

Variables

- Each memory location is given a name
- The name is the *variable* that refers to the data stored in that location
  - Eg: rollNo, classSize
- Each variable has a *type* that defines the interpretation of data
  - e.g. integers (1, 175, 25649), or characters (‘a’, ‘M’, ‘n’, ‘i’, ‘d’)
- All data is stored as binary strings. That is, a sequence of 0’s and 1’s (bits), in a word of a predetermined size. A *byte* is made of *8 bits*.

Instructions

- Instructions take data stored in variables as arguments.
- Some instructions do some operation on the data and store it back in some variable.
- The instruction "X←X+1" on integer type says that "Take the integer stored in location X, add 1 to it, and store it back in X"
- Other instructions tell the processor to do other things. For example, "jump" to a particular instruction next, or terminate the program

Programs

- A program is a sequence of instructions.
- Normally the processor works as follows,
  - Step A: pick next instruction in the sequence
  - Step B: get data for the instruction to operate upon
  - Step C: execute instruction on data (or "jump")
  - Step D: store results in designated location (variable)
  - Step E: go to Step A
- Such programs are known as *imperative programs.*

Programming paradigms

- *Imperative programs* are sequences of instructions. They are abstractions of how the *von Neumann machine* operates.
  - Pascal, C, Fortran
- *Object Oriented Programming Systems (OOPS)* model the domain into objects and interactions between them.
  - Simula, CLOS, C++, Java
- *Logic programs* use logical inference as the basis of computation.
  - Prolog
- *Functional programs* take a mathematical approach of functions.
  - *LISP, ML, Haskell*

A Limitation – Computer Arithmetic

- Number of digits that can be stored is limited
- Causes serious problems

Consider a computer that can store:

Sign, 3 digits and a decimal point

Eg: 212, -212, -21.2, -2.12, -.212

More Examples

- 113. + -111. = 2.00
- 2.00 + 7.51 = 9.51
- -111. + 7.51 = -103.49 (exact arithmetic)

But our computer can store only 3 digits.
So it rounds –103.49 to –103

This is a very important thing to know as a programmer. Why?

Why?

Consider 113. + -111. + 7.51
To us addition is associative
(a+b)+c yields the same as a+(b+c)
For our 3-digit computer:
(113. + -111.) + 7.51 = 2.00 + 7.51 = 9.51
113. + (-111. + 7.51) = 113. – (-111. +8.)
= 113. – 103. = 10.0

The order of evaluation can affect the results

Conclusion

- Computer is fast, but only for well-defined purposes
- Needs a precise sequence of instructions
- We must learn to use its speed
- And manage its limitations

Books

- A. Bradley: *Programming for Engineers*
- V. Rajaraman: *Computer Programming in C*
- R. G. Dromey: *How to Solve It By Computer*
- Kernighan and Ritchie: *The C Programming Language*
- Kernighan and Pike: *The Unix Programming Environment*