IC150
Computation for Engineering

An Introduction to
Problem Solving using Computers

Timothy A. Gonsalves

Course Outline

- Introduction to Computing
- Problem solving using computers
- Exercises and examples are from the mathematical area of Numerical Methods
- Tools: C, Scilab, OpenOffice Calc, Linux

Evaluation

- Two Quizzes – 50
- End of Semester Exam – 50
- 5 marks extra for class 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
- Grace marks based on participation
  - Moodle discussion forum
- Professionalism

What is this IC150 about?

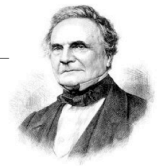
- Computer and its components
- Computing – personal, distributed, parallel
- Programming Languages
- Problem Solving and Limitations of a Computer

What are common uses of a computer?

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

What IS a computer?

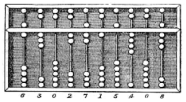


- A computer is a *machine*
- Something that operates *mechanically*
- But it is a *flexible* machine
- Its behaviour is controlled by a *program*
- A program is like a *spell* cast on a machine
- Programmers are like *wizards*
- Programs reside in the *memory* of the machine
  - *Charles Babbage (1791-1871):*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

Dromey pg. 1: Computer problem-solving can be summed up in one word – it is *demanding*! It is an intricate process requiring much thought, careful planning, logical precision, persistence, and attention to detail. At the same time it can be a challenging, exciting, and satisfying experience with considerable room for personal creativity and expression. If computer problem-solving is approached in this spirit then the chances of success are greatly amplified.

Early Computing Hardware

The Slide rule

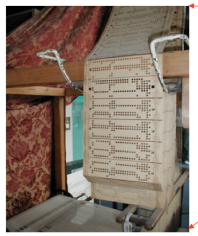


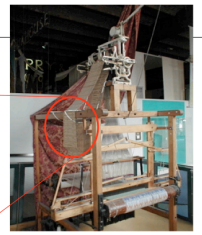
The Chinese Abacus

The gear replaced the beads in early mechanical calculators

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

Jaquard looms

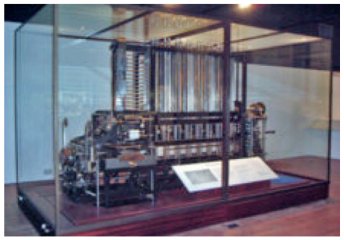




Used *punched cards* to weave *different patterns*

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.

He began in 1822 with what he called the difference engine, made to compute values of polynomial functions. Babbage's difference engine was created to calculate a series of values automatically. By using the method of finite differences, it was possible to avoid the need for multiplication and division.

The first difference engine was composed of around 25,000 parts, weighed fifteen tons (13,600 kg), and stood 8 ft (2.4 m) high.

The **analytical engine**, an important step in the history of computers, was the design of a mechanical general-purpose computer by English mathematician Charles Babbage. In its logical design the machine was essentially modern, anticipating the first completed general-purpose computers by about 100 years. It was first described in 1837. Babbage continued to refine the design until his death in 1871. Because of the complexity of the machine, the lack of project management science, the expense of its construction, and the difficulty of assessing its value by Parliament relative to other projects being lobbied for, the engine was never built.

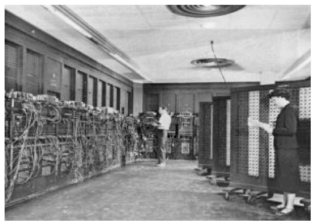
The First Programmer

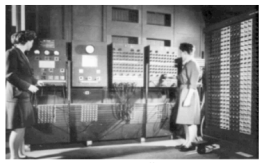
Augusta Ada King, Countess of Lovelace (December 10, 1815 – November 27, 1852), born **Augusta Ada Byron**, is mainly known for having written a description of Charles Babbage's early mechanical general-purpose computer, the analytical engine.

The programming language ADA is named after her

Lovelace was certainly one of the few people who fully understood Babbage's ideas and created a program for the Analytical Engine. Had the Analytical Engine ever actually been built, her program would have been able to calculate a sequence of Bernoulli numbers.

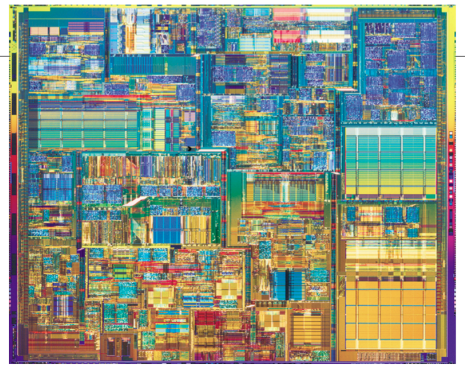
ENIAC – the first electronic computer





Physically, ENIAC was massive compared to modern PC standards. It contained 17,468 vacuum tubes, 7,200 crystal diodes, 1,500 relays, 70,000 resistors, 10,000 capacitors and around 5 million hand-soldered joints. It weighed 27 tons, was roughly 2.4 m by 0.9 m by 30 m, took up 167 m², and consumed 150 kW of power.

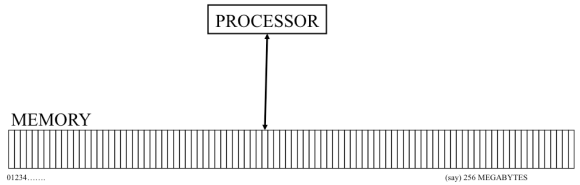
Construction started in 1943, completed in 1946, worked until 1955



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

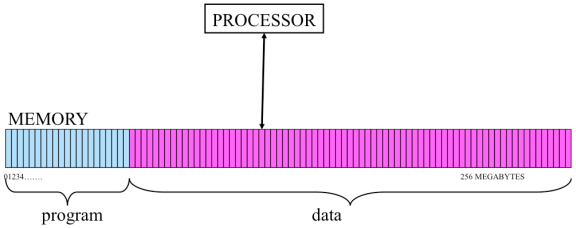
122 mm square die size

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 stored program *von Neumann* computer



- 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
- 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, 14, 25649), or characters (a, f, T, Z)
- 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 to 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*

Eg: find the average of 2 numbers a and b and store the result in av

Imperative: Tell the computer what do do; give orders to the computer

add a and b

store the sum in s

divide s by 2

store this in av

OO: List object and number objects. List object has "average" method. Insert number into list. Note: will work with any number of members in the list.

Logic: Make assertions –let av be average of a and b; let a be 5; let b be 7;. Any time a or b changes, av automatically changes

Functional: mathematical statements: divide(sum(a, b), 2);

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
Sign and decimal point are optional

example : 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 System designer. Why?

Why?

Consider 113. + -111. + 7.51

To us addition is associative

(a+b)+c yields 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
- So we must learn to use its speed
- And manage its limitations

Books

- 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*