Lecture 1 — Introduction to the Course

J. Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

March 14, 2015

Acknowledgments: W.D. Bishop

Course Objectives

The purpose of computing is insight, not numbers.

- R. W. Hamming

Goal: students gain insight into software design & construction.

You will have an opportunity to achieve the following three goals:

1. Learn to develop algorithms to solve programming problems
2. Acquire an understanding of a programming language
3. Establish a foundation for future courses

Course Syllabus

As our first order of business, we'll go over the course syllabus.

What is a Computer?

Computer: a programmable device that can store, retrieve, and process data.

Program: to work out a sequence of operations to be performed by a mechanism.

(Merriam-Webster Online Dictionary Definitions)

Definition of a Computer

The formal definition of a computer system encompasses devices other than just personal computers.

Embedded computer systems use computing devices to control other systems:

- Anti-lock braking systems
- MP3 players
- Cellular phones
- Televisions

Statistically speaking, personal computers (PCs) represent a very small fraction of the computer marketplace.

History of the Programmable Computer

Z3 (1941)
: Lacked support for conditional branch instructions

Colossus (1943)
: Targeted a specific application (code breaking)

Harvard Mark I (1944)
: Used separate storage units for instructions & data

ENIAC (1946)
: Required hard-wiring of the control paths.

Manchester Baby (1948)
: Implemented first working von Neumann computer architecture.

About the ENIAC

Built in 1944, the ENIAC had the following components:

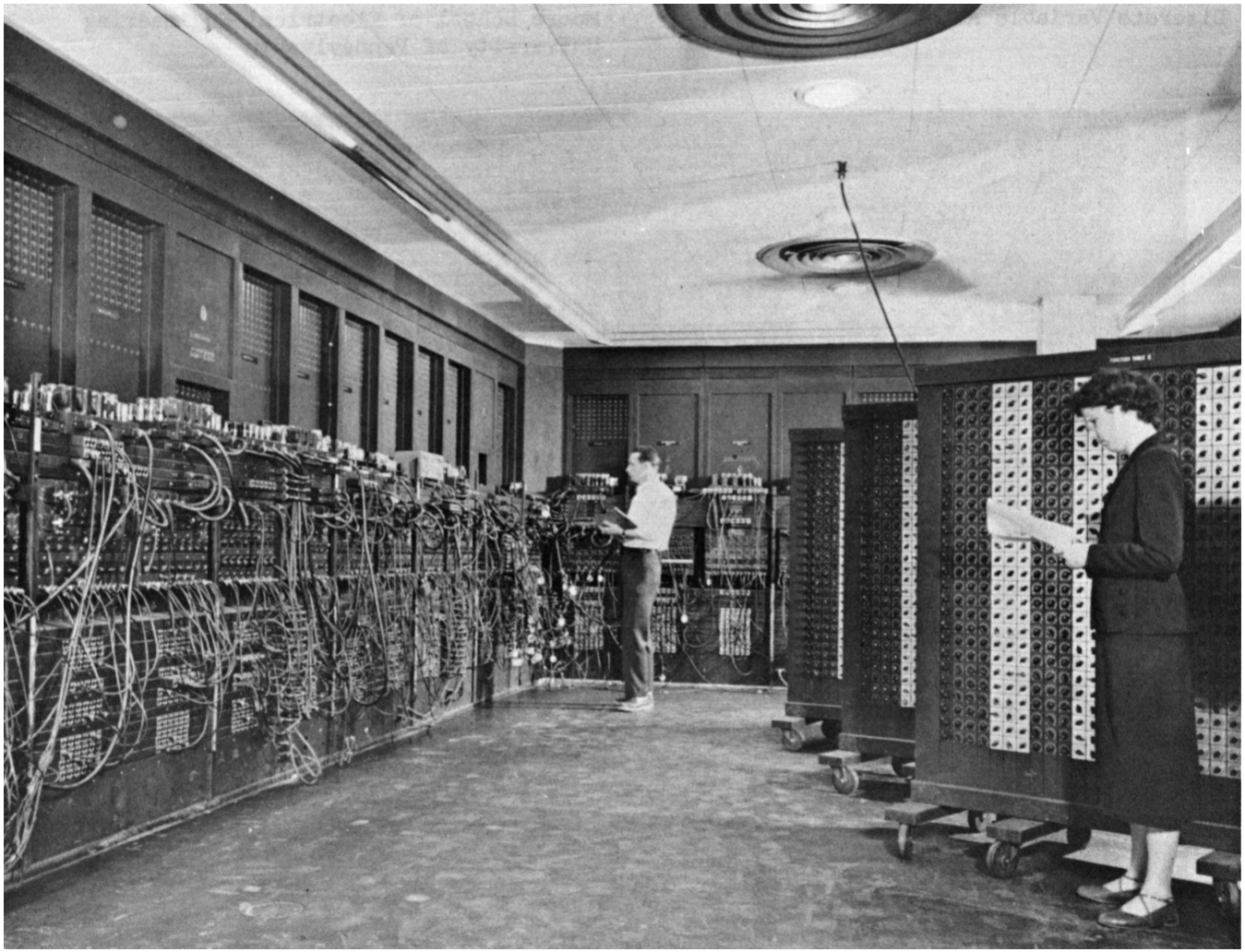
- 17 468 vacuum tubes
- 1 500 relays
- 7 200 crystal diodes
- 70 000 resistors
- 10 000 capacitors
- 5 000 000 hand-soldered joints

About the ENIAC

The ENIAC had the following properties:

- 30 tons in weight
- 167 m^3 in volume
- 160 kW of power consumption

ENIAC



Source: http://en.wikipedia.org/wiki/ENIAC

Computer Development Milestones

Generation	Technology	Examples	Comments
0^{th} (1642-1940)	Mechanical Systems	Pascalene, Analytical Engine	Simple machines performing addition and subtraction
1^{st} (1940-1955)	Vacuum Tubes	Z3, Colussus, ENIAC, EDVAC, Harvard Mark I, IBM 701, Manchester Baby, Univac I	Mostly special purpose computers for military applications
2^{nd} (1955-1965)	Transistors	DEC PDP-1, IBM 7030, CDC 6600	Computers with support for high level programming languages
3^{rd} (1965-1975)	Integrated Circuits	DEC PDP-8, DEC PDP-11, IBM System/360	Computers with faster memory and operating systems
4^{th} (1975-1990)	Very Large Scale Integration (VLSI)	DEC MicroVAX, IBM 3090, IBM PC, Cray X-MP, Osborne 1, Apple II, Apple Macintosh	First personal computers and the first portable computers
5^{th} (1990-)	Nanotechnology, Massive Parellelism	IBM BlueGene/L, IBM x345, OLPC, Apple Macbook, Sun Ultra II	Massive supercomputers, powerful workstations, portable computers

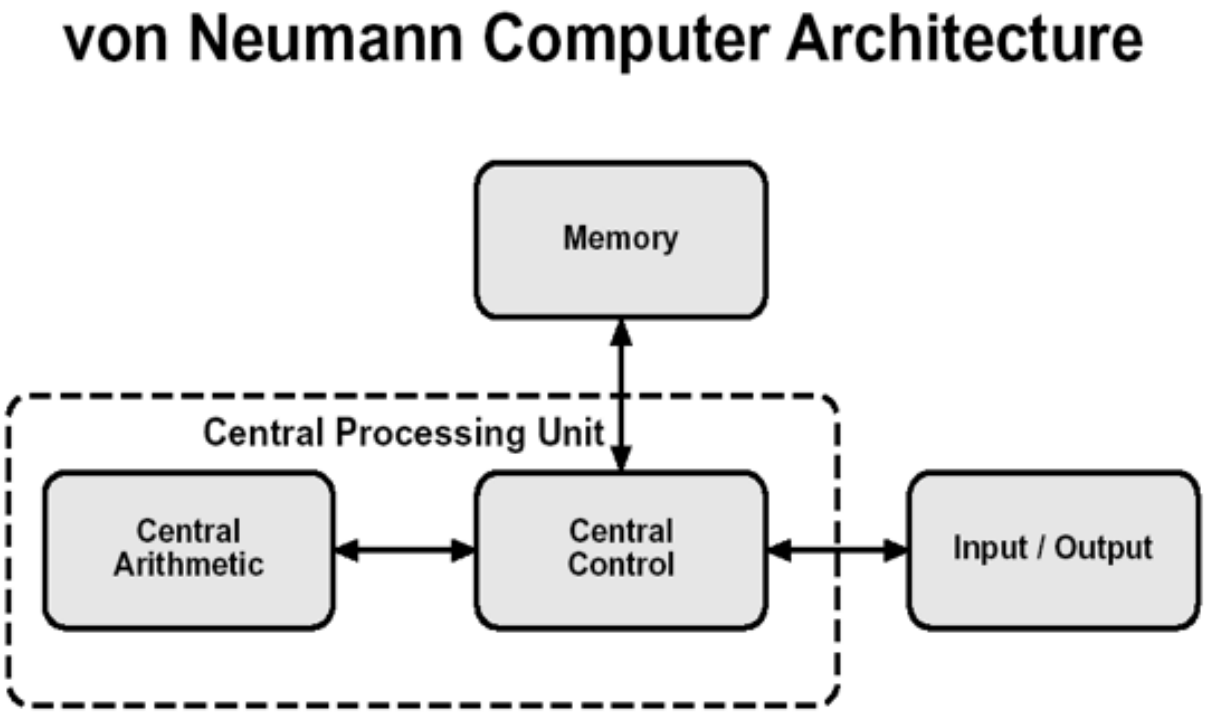
Source: IEEE Computer Society (http://www.computer.org/portal/cms_docs_computer/computer/timeline/timeline.pdf)

Digital Computers

A Digital Computer uses electronic circuits to implement a programmable computer.

All information within a digital computer is represented using a sequence of `0`s and 1s.

Typically built using chips known as ASICs (Application-Specific Integrated Circuits) made of millions of transistors.



Source: J. von Neumann, First Draft of a Report on the EDVAC. Technical Report W-6700RD-492, Moore School of Electrical Engineering, University of Pennsylvania, June 1945.

Before learning to program, let's examine why people program...

Computers are best suited to solving problems that involve repetitive processing or large amounts of data.

Computers do not (yet) exhibit intelligence like humans.

They do exactly what they are instructed to do.

Why Create Software?

Computers are well-suited to reliably performing repetitive tasks.

Humans get distracted and this leads to mistakes.

Computers don't get bored!

Also, computers are often much faster...

Why Create Software?

Imagine your task is to calculate the sum of integers from 1 to 10 000.

You can do this with a calculator, but: slow, boring, error-prone.

Summing Integers: Analytical Approach

A smart student might observe that half the values will be 5,000 or less and half the values will be 5,001 or more.

Estimate the result to be close to 50,000,000.

The exact answer: multiplying the average (5,000.5) by total number of values (10,000) to produce: 50,005,000.

Summing Integers: Computer Approach

Write software to do it for you!

Even a novice programmer would need only a few minutes.

A computer calculates the exact sum very quickly.

Summing Integers: Repeat

Performing only once? the analytical approach might be best.

Suppose you need to do this task for different ranges of integers.

Programming makes a good solution to automate this boring task.

When Does it Make Sense to Write Programs?

A few examples of useful computer programs:

1. Databases (store, process, & retrieve information)
2. Spreadsheets (manipulates tables of numbers; produces graphs)
3. Word processors (edit letters, reports, and books)
4. Social networking (exchange photographs and messages)
5. Audio/video players (record and playback songs and videos)
6. Videogames (entertainment)

Goals of Programming

Arguably, the most important goal is correctness.

A calculator gets the answer right 99% of the time. Would you buy it?

Computers must provide correct answers if they are to be trusted and used effectively.

Goal 1: Correctness

Developing a program that produces correct output for all possible inputs is very challenging (if not impossible).

Functional correctness examines whether the outputs are correct for the inputs provided.

The program must also be predictable; the program is not correct if it produces the right answer only some of the time.

The program must also provide a logical user interface.

If users don't understand it, they'll use it incorrectly.

Secondary Goals of Software

Other potential goals during the design & construction of software:

- maintainability
- efficiency
- readability
- usability
- security
- reliability
- robustness
- simplicity
- portability

About Computer Programming

Computer programming can be challenging, exciting, frustrating, rewarding, fun, interesting...

Think of it like another kind of literacy.

You learned to read & write, to count, and now... to program.

Programming is not always easy, but it's a skill well worth learning.