

# CSC 221: Computer Programming I

Fall 2001

- history of computing technology
  - hardware vs. software
  - technology generations
  - von Neumann architecture
- programming languages
  - machine vs. assembly vs. high-level languages
  - why C++ ?

## hardware vs. software

### basic terminology:

- hardware – the physical components of the computer
  - e.g., processor (Pentium 4, Celeron, PowerPC, Alpha)
  - memory (RAM, cache, hard drive, floppy drive)
  - input/output devices (keyboard, mouse, monitor, speaker)
- software – programs that run on the hardware
  - e.g., operating system (Windows, Mac OS, Linux)
  - applications (Word, Excel, Powerpoint, RealPlayer, Netscape, IE)
  - development tools (Visual C++, CodeWarrior, JDK)

*The easiest way to tell the difference between hardware and software is to kick it.  
If it hurts your toe, it's hardware.*

Carl Farrell

## History of computing technology

### DYK?

- When were "modern" computers invented?
- When were computers accessible/affordable to individuals?
- When was the Internet born?
- When was the Web invented?
- How did Bill Gates get so rich?

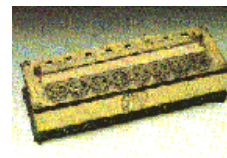
the history of computers can be divided into generations, with each generation defined by a technological breakthrough

- gears and relays
  - 1. vacuum tubes
  - 2. transistors
  - 3. integrated circuits
  - 4. very large scale integration
  - 5. parallel processing & networking

## Generation 0: Mechanical Computers (1642-1945)

### 1642 – Pascal built mechanical calculating machine

- mechanical gears, hand-crank, dials and knobs
- other similar machines followed

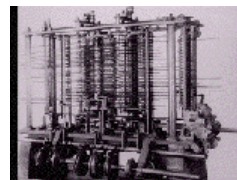


### 1805 – first programmable device, Jacquard loom

- wove tapestries with elaborate, programmable patterns
- pattern represented by metal punch-cards, fed into loom
- could mass-produce tapestries, reprogram with new cards

### mid 1800's – Babbage designed "analytical engine"

- expanded upon mechanical calculators, but programmable via punch-cards
- described general layout of modern computers
- never functional, beyond technology of the day



## Generation 0 (cont.)

### 1890 – Hollerith invented tabulating machine

- used for 1890 U.S. Census
- stored data on punch-cards, could sort and tabulate using electrical pins
- finished census in 6 weeks (vs. 7 years)
- Hollerith's company would become IBM



### 1930's – several engineers independently built "computers" using electromagnetic relays

- physical switch, open/close via electrical current
- Zuse (Nazi Germany) – destroyed in WWII
- Atanasoff (Iowa State) – built with grad student
- Stibitz (Bell Labs) – followed design of Babbage

## Generation 1: Vacuum Tubes (1945-1954)

### mid 1940's – vacuum tubes replaced relays

- glass tube w/ partial vacuum to speed electron flow
- faster than relays since no moving parts
- invented by de Forest in 1906



### 1940's – hybrid computers using vacuum tubes and relays were built

#### COLOSSUS (1943)

- built by British govt. (Alan Turing)
- used to decode Nazi communications

#### ENIAC (1946)

- built by Eckert & Mauchly at UPenn
- 18,000 vacuum tubes, 1,500 relays
- weighed 30 tons, consumed 140 kwatts

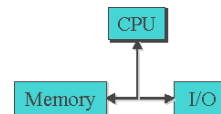
## Generation 1 (cont.)

COLOSSUS and ENIAC were not general purpose computers

- could enter input using dials & knobs, paper tape
- but to perform a different computation, needed to reconfigure

von Neumann popularized the idea of a "stored program" computer

- store both data and programs in Memory
- Central Processing Unit (CPU) executes by loading program instructions from memory and executing them in sequence
- interact with the user via Input/Output devices



virtually all modern machines follow this *von Neumann Architecture*

programming was still difficult and tedious

- each machine had its own *machine language*, 0's & 1's corresponding to the settings of physical components
- in 1950's, *assembly languages* replaced 0's & 1's with mnemonic names

## Generation 2: Transistors (1954-1963)

mid 1950's – transistors began to replace tubes

- piece of silicon whose conductivity can be turned on and off using an electric current
- smaller, faster, more reliable, cheaper to mass produce
- invented by Bardeen, Brattain, & Shockley in 1948 (won 1956 Nobel Prize in physics)



computers became commercial as cost dropped

high-level languages were designed to make programming more natural

- FORTRAN (1957, Backus at IBM)
- LISP (1959, McCarthy at MIT)
- BASIC (1959, Kemeny at Dartmouth)
- COBOL (1960, Murray-Hopper at DOD)

the computer industry grew as businesses could buy Eckert-Mauchly (1951), DEC (1957)  
IBM became market force in 1960's

## Generation 3: Integrated Circuits (1963-1973)

### integrated circuit (IC)

- as transistor size decreased, could package many transistors with circuitry on silicon chip
- mass production further reduced prices

1971 – Intel marketed first *microprocessor*, a chip w/ all the circuitry for a calculator



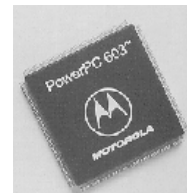
### 1960's saw the rise of Operating Systems

- an *operating system* is a collection of programs that manage peripheral devices and other resources
- allowed for *time-sharing*, where multiple users share a computer by swapping jobs in and out
- as computers became affordable to small businesses, specialized programming languages were developed
  - Pascal (1971, Wirth)
  - C (1972, Ritchie)

## Generation 4: VLSI (1973-1985)

### Very Large Scale Integration (VLSI)

- by mid 1970's, could fit hundreds of thousands of transistors w/ circuitry on a chip
- could mass produce powerful microprocessors and other useful IC's
- computers finally affordable to individuals



### late 1970's saw rise of personal computing

- Gates & Allen founded Microsoft in 1975
  - Gates wrote BASIC compiler for personal computer
  - would grow into software giant, Gates richest in world
  - <http://www.webho.com/WealthClock>
- Wozniak and Jobs founded Apple in 1977
  - went from garage to \$120 million in sales by 1980
- IBM introduced PC in 1980
  - Apple countered with Macintosh in 1984
- Stroustrup developed C++ in 1985
  - object-oriented extension of C language



## Evolution of programming: assembly language

mid 1950's: assembly languages replaced numeric codes with mnemonic names

- an *assembler* is a program that translates assembly code into machine code
  - input*: assembly language program
  - output*: machine language program
- still low-level & machine-specific, but easier to program

```
gcc2_compiled:
    .global _Q_qtod
    .section      ".rodata"
        .align 8
.LLC0: .asciz  "Hello world!"
    .section      ".text"
        .align 4
    .global main
    .type        main,#function
    .proc        04
main:    !#PROLOGUE# 0
        save $sp,-112,$sp
        !#PROLOGUE# 1
        sethi %hi(cout),%o1
        or %o1,%lo(cout),%o0
        sethi %hi(.LLC0),%o2
        or %o2,%lo(.LLC0),%o1
        call __ls__7ostreamPc,0
        nop
        mov %o0,%i0
        mov %i0,%o0
        sethi %hi(endl__FR7ostream),%o2
        or %o2,%lo(endl__FR7ostream),%o1
        call
        __ls__7ostreamPFR7ostream_R7ostream,0
        nop
        mov 0,%i0
        b .LL230
        nop
.LL230: ret
        restore
.LLfel: .size    main,.LLfel-main
        .ident  "GCC: (GNU) 2.7.2"
```

## Evolution of programming: high-level language

late 1950's – present:  
high-level languages allow the programmer to think at a higher-level of abstraction

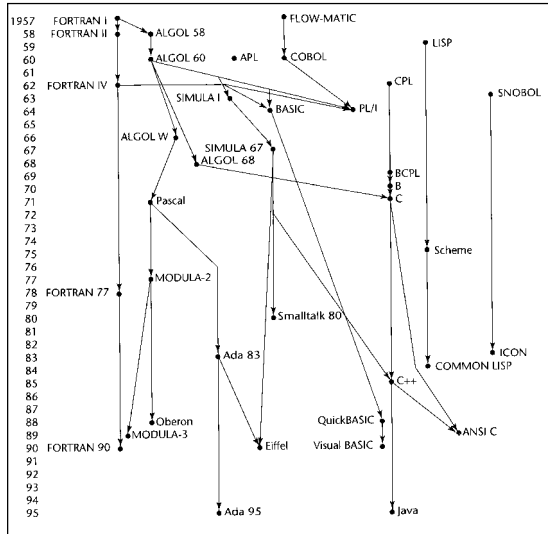
- a *compiler* is a program that translates high-level code into machine code
  - input*: C++ language program
  - output*: machine language program
- similar to assembler, but more complex
- in practice, often pays to divide programs into components (e.g., useful libraries). Need *linker/loader* program to combine separately compiled pieces of code into executable.

```
// Author: Dave Reed
// Date : 8/5/01
//
// This C++ program prints out "Hello world!"
///////////////////////////////////////////////////////////////////

#include <iostream>
using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

## Why C++ ?



C++ is a hybrid, general-purpose language

- derived from C, so can be used for low-level systems programming
- is a fully object-oriented extension, so can be used for large-scale software development

C++ and its descendant Java are the dominant languages in industry

## If you want to know more...

check out the following (purely optional) links

[Computers: History and Development](#)

[Personal Computers: History and Development](#)

[Computer Museum History Center](#)

[The History of Apple Computers](#)

[The History of Microsoft](#)

[Internet Pioneers: Tim Berners-Lee](#)

[Internet Pioneers: Marc Andreessen](#)

[Webopedia entry on Programming Languages](#)

[A short history of C++](#)