**Lecture Notes: Fall 2018 Week 1**

**History of Computers - Hardware**

*{Search for image of Danny Hillis tinkertoy computer}*

*-*Yes, it is possible to build a computer from sticks and stones (or just pieces of wood)

**(*all dates below are only for historical context, I don’t expect you to remember them!)***

- 1804 Punch card input (Jacquard loom)

- 1822 Charles Babbage ‘Difference Engine’ to compute log + trig tables (only a small test version was built)

1842-1870 Babbage ‘Analytical Engine’ design began (never built)

Used punch card input ala the Jacquard loom

1842 [Ada Lovelace](https://en.wikipedia.org/wiki/Ada_Lovelace) wrote software for the Analytical Engine

and is now recognized as the world’s first computer programmer.

1946 – ENIAC christened. Vacuum tube digital, binary computer.

1947 – First ‘Bug’ was an actual moth stuck in a relay of the MARK II computer, found by [Grace Hopper](https://en.wikipedia.org/wiki/Grace_Hopper).

‘Bug’ now refers mostly to software problems, not hardware.

1950s – Transistors replace vacuum tubes

1960s – Integrated Circuits (ICs) replace individual transistors

1970s – VLSI (Very Large Scale Integrated circuits) replace Integrated Circuits

1970s - Personal Computers

1980s – Networking

1990s – WorldWideWeb. Java.

2000s – Social Media

2010s – Smartphones and…VR?...AI?

**Basic Computer Architecture**

* Input devices
* Output devices
* Long term storage (Hard drive, tape backups, CD/DVD, USB stick…)
* Short term storage (RAM)
* Communication devices (MODEM = acronym of ‘**MO**dulator **DEM**odulator’).
* Central Processing Unit (CPU)
  + - Control circuitry
    - Registers (Used for temporary storage of calculations. there are very few registers in the CPU, but they have the fastest possible storage and retrieval of data.)
    - Arithmetic and Logical Unit (ALU)

**Other Computer Architectures (a few examples)**

* + - [Analog](https://en.wikipedia.org/wiki/Analog_computer)
    - [Non-binary, but digital](https://en.wikipedia.org/wiki/Ternary_computer)
    - [Dataflow](https://en.wikipedia.org/wiki/Dataflow_architecture)
    - [Quantum](https://en.wikipedia.org/wiki/Quantum_computing)

**History of Computers – Software**

* ENIAC: was first programmed by plugging in cables
* ENIAC: [John Von Neumann](https://en.wikipedia.org/wiki/John_von_Neumann) – store programs just like you store data

“See- Stored Program Architecture”

(check out this part of the Von Neumann Wikipedia page:

[https://en.wikipedia.org/wiki/John\_von\_Neumann#Cognitive\_abilities](https://en.wikipedia.org/wiki/John_von_Neumann%23Cognitive_abilities)

Whenever I feel like I’ve been amazingly clever, I re-read this. It smacks a person right back down to earth.)

**Binary and Hexadecimal counting, addition, and conversion between bases. You should know how these work, or be able to do the following tasks by hand, for exam #1.**

* Place value in base-10, base-2, base 16 systems
* Counting in binary
* Adding in binary, with carry.
* Hexadecimal digits
* Convert binary to hexadecimal and back again
* Convert binary to decimal and back again.
* Convert between decimal and hex using binary as an intermediate step. (less difficult)
* Convert directly from decimal to hex and back. (more difficult)
* RGB color scheme as 6 hex digits

Number base conversion tutorials can be found on the web. Here is one web page:

<http://www.purplemath.com/modules/numbbase.htm>

--------------------------------------------------------------------------------------------

* Machine Language is the binary language that executes directly on the hardware, but is very different for each different CPU model.

* Assembly Language (somewhat more readable machine language)
* **High level Languages**
  + - FORTRAN (optimized for scientific/math applications)
    - Cobol (for business applications)
    - C (general purpose, very popular, very powerful, ancestor of many modern languages)
    - ***Java!*** (object oriented, familiar syntax based on the C language <https://en.wikipedia.org/wiki/List_of_C-family_programming_languages> , currently the most popular programming language in the world.)

Programming Paradigms (ways of thinking)

* Monolithic (code is not modular, re-usable, or easily maintainable. It is possible to create poor code in any language.)
* Structured Programming
  + Procedural Programming
    - Procedures (Modula II, Pascal)
    - Subroutines (Fortran)
    - Functions (Python, Fortran)
    - Methods (Java)
* Functional Programming – treat computation as evaluation of mathematical functions
* Object Oriented Programming
* Object Oriented Languages
  + - Simula
    - Smalltalk
    - Modula -2
    - C++
    - Python
    - Ruby
    - Java!

Problems with older languages–

**Fortran** : Implicit typing, strips blanks, computed GOTO results in Spaghetti code.

These problems have been ‘cured’ with work-arounds. Fortran is still the best choice for very efficient mathematical programming.

**C:** Very powerful and efficient, but allowing the use of ‘naked’ addresses (pointers) can result in hard-to-find bugs.

Still very widely used, especially for low-level programming (device drivers, etc.)

Java’s Object Oriented nature encourages “Information Hiding”

Each portion of the software is only given access to as much of the data as it needs to accomplish its task.