1 | Low Level Programming

2 | Basic Computer Architecture

2.1 | Core Architecture

Before computers existed, research was put into designing methods of automating calculations: notable examples being Lambda calculus, the Turing machine, and the von Neumann architecture. The von Neumann architecture won out due to its ease of programmability and robustness.

It consists of a CPU connected to a memory bank, where a Arithmetic Logic Unit (ALU) in the CPU performs calculations and a control unit fetches things from memory.

Programs are a series of instructions fetched sequentially (except when JMP instructions are involved). Assembly language is a human-readable version (or mnemonics) for the binary instruction.

2.2 | Evolution

There are some problems with the above design model:

- Manual memory editing is required for interacting with computer
- Multitasking is impossible so computer is just stuck for periods of time
- No protection for instructions so user applications can wreak havoc
- Hardware bottlenecks become a problem when people start building parts separately

The Intel 64 architecture addresses these problems with extensions to von Neumann architecture.

- Registers, or tiny pieces of memory located on the CPU die itself
- The hardware stack, for storing local variables and implementing scope/functions
- Interrupts, which can affect/suspend/stop program execution from outside the program. Examples of when interrupts would be called are things like zero division, invalid instructions, etc.
- Protection rings, which restrict certain instructions based on what ”ring” you are in
- Virtual memory, which allows memory to be distributed across programs?

2.3 | Registers

2.4 | Protection Rings

2.5 | Hardware Stack

2.6 | Summary

3 | Assembly Language