ECE M116C

Contents

Part I

First 5 Weeks

Chapter 1

Preface

1.1 Abstraction

Definition: Abstraction

Abstraction is the concept of providing a (relatively) simple interface to higher level programs, hiding unnecessary complexity.

We define a computer as a black box with multiple layers of *abstraction*. When focusing on a particular layer, we abstract away the irrelevant layers. There are three main abstraction layers:

(i) Application Layer: Here, we translate from algorithms to code. We usually write these in high level languages (e.g. C/++, Java, etc.).

(ii) Systems Layer: Here, we have the compiler that translate HLL[1] code to machine code, and the operating system, which deals with everything you learned in CS111.

(iii) Hardware Layer: The hardware layer is the physical hardware (who would've thought) that makes all of this possible (e.g. CPU, RAM, etc.)!

Similar to computers, program code also has three layers of abstraction!

(i) High-Level Language: This layer hosts all of our favorite languages (e.g. C/++, JS, etc.). This level of abstraction provides good productivity and portability[2].

(ii) Assembly Language: A textual representation of hardware instructions. Assembly is architecture dependent!

(iii) Hardware Representation: Here we have the actual binary (1's and 0's) that encode instructions and data.

1.2 Instruction Set Architecture

Definition: Instruction Set Architecture

The **Instruction Set Architecture (ISA)** is the set of instructions supported by a computer. There are multiple (all incompatible) ISA's and they usually come in families. ISA's usually come with privilaged and standard instruction sets.

The ISA is an *interface* between hardware and software, and as such, allows them to develop and evolve *independently*.

The software sees a *functional* description of the hardware: *(i)* Storage locations (e.g. memory) and *(ii)* Operations (e.g. `add`).

The hardware sees a list of instructions and their *order*.

[1] HLL: High Level Language.

[2] Most languages are hardware/architecture agnostic.

1.3 Efficiency

The main objective when architecting a computer is to make it *efficient*. Here, we define "efficient" to be:

(i) Performance[3]: Fast as fuck.

(ii) Power Consumption: Low.

(iii) Cost: Low.

(iv) Reliable and Secure.

1.4 History

Look at the slides for the full history. TLDR: we've come a long way. The main takeaway of this section is that we've been able to make these improvements for two major reasons: *new technologies* and *innovative techniques*.

1.5 Laws

1.6 Moore's Law

Definition: Moore's Law

Moore's Law states: "The number of transistors in an IC[a] doubles every two years."

[a] IC: Integrated Circuit.

This Moore guy is pretty smart since we've been roughly on track with his prediction (up until about 2005).

Definition: Dennard's Scaling Law

Dennard's Scaling Law states: Moore's law $\implies$ each transistor's area is reduced by 50% (or every dimension by 30%).

Naturally, it follows that:

(i) Voltage is reduced by 30% to keep the electric field constant (remember $V = EL$? Me neither.).

(ii) L is reduced $\implies$ delays are reduced by 30% ($x = Vt$).

(iii) Frequency is increased by 40% ($\textit{frequency} = \frac{1}{time}$).

(iv) Capacitance is reduced by 30% ($C = \frac{kA}{L}$).

(v) Since $P = CV^2f$ (apparently), power consumption per transistor is reduced by 50%. As such, the power consumption of the entire chip stays the same.

Definition: Amdahl's Law

Amdahl's Law states: "The performance improvement (*speed up*) is limited by the part you cannot improve (*sequential part*)." That is,

$$\textit{speed up} = \frac{1}{(1-p) + \frac{p}{s}}$$

where p is the part that can be improved and s is the factor of improvement.

[3] Performance is *usually* the most important metric.

1.6.1 The Power Wall

Up until 2005, we've been able to make transistors smaller (ergo faster) while keeping power consumption the same. Unfortunately, since 2005, due to *tiny* transistor sizes, the static power leakage has become so dominant that we couldn't keep the power consumption the same.

1.6.2 Multi-Core Era

Guess what's better than one CPU core? Multiple! Unfortunately, Amdahl is a party pooper and his law suggests we're hitting peak performance.