



Introduction to Computer Engineering

ECE/CS 252, Fall 2010

Prof. Mikko Lipasti

Department of Electrical and Computer Engineering
University of Wisconsin – Madison

Chapter 1

Welcome Aboard

Slides based on set prepared by
Gregory T. Byrd, North Carolina State University

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Computer System: Layers of Abstraction

Application Program

Algorithms

Language

Software

Hardware

Instruction Set Architecture
(and I/O Interfaces)

Microarchitecture

Circuits

Devices

1-3

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Big Idea #1: Universal Computing Device

All computers, given enough time and memory, are capable of computing exactly the same things.

Smartphone = Desktop PC = Supercomputer

1-4

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Turing Machine

Mathematical model of a device that can perform any computation – Alan Turing (1937)

- ability to read/write symbols on an infinite “tape”
- state transitions, based on current state and symbol

Every computation can be performed by some Turing machine. (*Turing’s thesis*)

$a, b \rightarrow T_{add} \rightarrow a+b$

Turing machine that adds

$a, b \rightarrow T_{mul} \rightarrow ab$

Turing machine that multiplies

1-5

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Universal Turing Machine

Turing described a Turing machine that could implement all other Turing machines.

- inputs: data, plus a description of computation (Turing machine)

$T_{add}, T_{mul} \rightarrow U$

$a, b, c \rightarrow U$

$U \rightarrow c(a+b)$

Universal Turing Machine

U is programmable – so is a computer!

- instructions are part of the input data
- a computer can emulate a Universal Turing Machine, and vice versa

Therefore, a computer is a universal computing device!

1-6

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

From Theory to Practice

In theory, computer can *compute* anything that's possible to compute

- given enough *memory* and *time*

In practice, *solving problems* involves computing under constraints.

- time
 - weather forecast, next frame of animation, ...
- cost
 - \$25 cell phone, automotive engine controller, ...
- power
 - battery-operated laptop, handheld video game, ...

1-7

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Big Idea #2: Transformations Between Layers

How do we solve a problem using a computer?
A systematic sequence of transformations between layers of abstraction.

Problem → **Software Design:** choose algorithms and data structures

Algorithm → **Programming:** use language to express design

Program → **Compiling/Interpreting:** convert language to machine instructions

Instr Set Architecture

1-8

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Deeper and Deeper...

Instr Set Architecture → **Processor Design:** choose structures to implement ISA

Microarch → **Logic/Circuit Design:** gates and low-level circuits to implement components

Circuits → **Process Engineering & Fabrication:** develop and manufacture lowest-level components

Devices

1-9

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Descriptions of Each Level

Problem Statement

- stated using "natural language"
- may be ambiguous, imprecise

Algorithm

- step-by-step procedure, guaranteed to finish
- definiteness, effective computability, finiteness

Program

- express the algorithm using a computer language
- high-level language, low-level language

Instruction Set Architecture (ISA)

- specifies the set of instructions the computer can perform
- data types, addressing mode

1-10

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Descriptions of Each Level (cont.)

Microarchitecture

- detailed organization of a processor implementation
- different implementations of a single ISA

Logic Circuits

- combine basic operations to realize microarchitecture
- many different ways to implement a single function (e.g., addition)

Devices

- properties of materials, manufacturability

1-11

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Many Choices at Each Level

```

graph TD
    A[Solve a system of equations] --> B[Red-black SOR]
    A --> C[Gaussian elimination]
    A --> D[Jacobi iteration]
    A --> E[Multigrid]
    C --> F[FORTRAN]
    C --> G[C]
    C --> H[C++]
    C --> I[Java]
    G --> J[Sun SPARC]
    G --> K[Intel x86]
    G --> L[IBM PowerPC]
    K --> M[Pentium 4]
    K --> N[Core 2 Duo]
    K --> O[AMD Athlon X2]
    M --> P[Ripple-carry adder]
    N --> P
    N --> Q[Carry-lookahead adder]
    O --> P
    O --> Q
    P --> R[Static CMOS]
    P --> S[Dynamic CMOS]
    Q --> R
    Q --> S
    Q --> T[Nanomechanical]
  
```

Tradeoffs: cost, performance, power (etc.)

1-12

What's Next

Bits and Bytes

- How do we represent information using electrical signals?

Digital Logic

- How do we build circuits to process information?

Processor and Instruction Set

- How do we build a processor out of logic elements?
- What operations (instructions) will we implement?

Assembly Language Programming

- How do we use processor instructions to implement algorithms?
- How do we write modular, reusable code? (subroutines)

I/O, Traps, and Interrupts

- How does processor communicate with outside world?

1-13