

# Chapter 1

## Welcome Aboard

Based on slides © McGraw-Hill  
Additional material © 2004/2005 Lewis/Martin  
Edited by Diana Palsetia

CIT 593

1/17

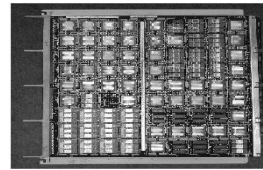
## Introduction to the World of Computing

### Computer: *Electronic genius?*

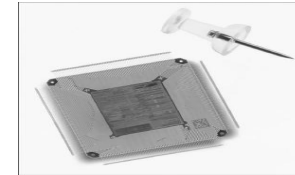
- NO! *Electronic idiot!*
- Does exactly what we tell it to, nothing more

### Goal

- Understand what's going on underneath



1980 - It took 10 of these boards to make CPU (Figure 1.1 Patt & Patel)



1998 - CPU - The Microprocessor (Figure 1.2 Patt & Patel)

CIT 593

2/17

## Recurring Themes

### Abstraction

- Allows us to manage seemingly insurmountable complexity

### Hardware v. Software

- Separation of hardware and software is artificial

CIT 593

3/17

## Recurring Theme #1: Abstraction

### Abstraction

- Productivity enhancer – don't need to worry about details...
  - Can drive a car without knowing how the internal combustion engine works.
- ...until something goes wrong!
  - Where's the dipstick? What's a spark plug?
- Important to understand the components and how they work together



### Bottom line

- Often best to operate at highest level of abstraction
- Dangerous to completely ignore lower levels of abstraction

CIT 593

4/17

## Recurring Theme #2: Hardware vs. Software

### Artificial Divide

- Hardware: physical computer and specifications associated w/ it
- Software: Operating System, application software

### Conventional Approach

- We really care about *computation*
- Ok to be expert in one and clueless in the other

### Greatness arises from blurring the HW/SW line

- A computing system works best when both capabilities and limitations are accounted.
- E.g.: Special video graphics processor

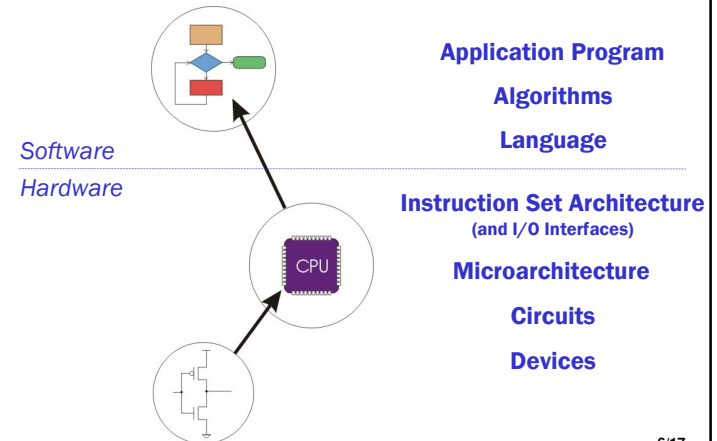
### Bottom Line

- Even if you specialize in one, you should understand capabilities and limitations of both.

CIT 593

5/17

## Recurring Themes



CIT 593

6/17

## Very Big Ideas

**These are core to what computing is all about!**

### Universality

- All computers can compute the same thing\*

### Layered Abstraction

- We can build very complex systems from simple components

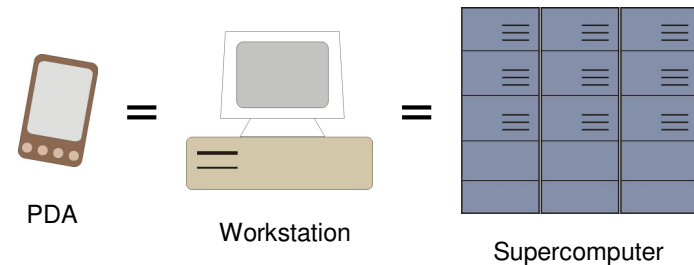
CIT 593

7/17

## Big Idea #1: Universal Computing Device

All computers *can* compute exactly the same things\*

\*given enough time and memory



CIT 593

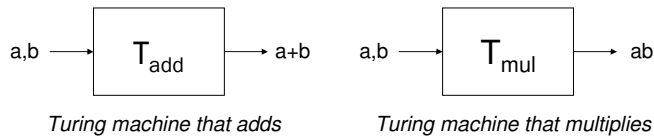
8/17

## 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*)



For more info about Turing machines, see [http://www.wikipedia.org/wiki/Turing\\_machine/](http://www.wikipedia.org/wiki/Turing_machine/)

For more about Alan Turing, see <http://www.turing.org.uk/turing/>

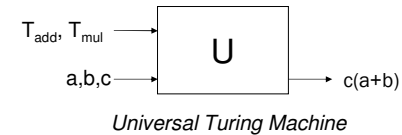
CIT 593

9/17

## Universal Turing Machine

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

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



**U is programmable – so is a computer!**

- Instructions are part of the input data (very important!)
- A computer can emulate a Universal Turing Machine, and vice versa

**Therefore, a computer is a universal computing device!**

CIT 593



10/17

## From Theory to Practice

### In theory

- Computers can compute anything that's possible to compute
- Given enough *memory and time*

### In practice

- Solving *real* problems requires computing under constraints
- Time
  - > E.g. Weather forecast 
- Cost
  - > E.g. Would you buy a \$1500 Cell phone? 
- Power
  - > E.g. 2 mins of talk time won't sell!

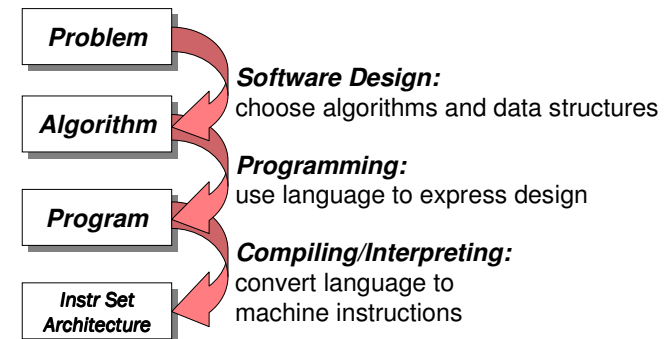
CIT 593

11/17

## Big Idea #2: Layered Abstraction

How do we solve a problem using a computer?

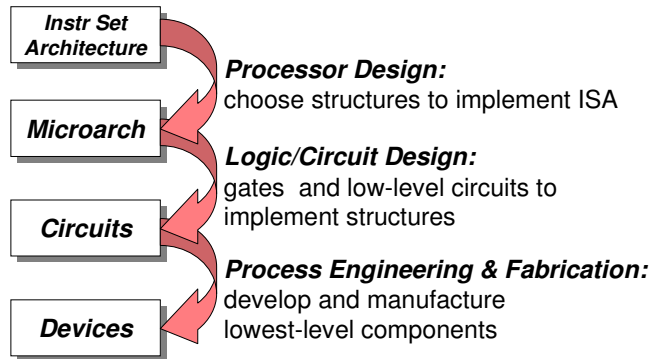
- Systematic sequence of transformations between layers of abstraction. . .



CIT 593

12/17

## Big Idea #2: Layered Abstraction (contd..)



CIT 593

13/17

## Descriptions of Each Level

### Problem Statement

- Stated using "natural language"

### Algorithm

- Step-by-step procedure, guaranteed to finish
- Definiteness, effective computability, finiteness

### Program

- Express the algorithm using a computer language
- High-level language (we will learn C language for the course)

### Instruction Set Architecture (ISA)

- Specifies the set of instructions the computer can perform
  - What kinds of operations ? (e.g. add or multiply)
  - What data is needed for each operation? (integer or floating point)
- We will learn LC3 Assembly (hypothetical) Language for the course

CIT 593

14/17

## 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

### CIT 595 - Digital System Organization & Design

- Covers the Microarchitecture, and Logic Circuits in detail
- And much more...

CIT 593

15/17

## Next Time

### Lecture

- Chapter 2: Bits and Bytes

### Reading

- History of Computers
- Chapter 1 - Welcome Aboard (Yale & Patt text)

Lookout for an homework assignment next week

CIT 593

16/17