

CSE141 and 141L: Introduction to Computer Architecture

◆ Outline

- Administrivia
- Background (expected knowledge)
- Course Overview

◆ Announcements

- **Book: Computer Organization and Design: The Hardware/software Interface (Patterson and Hennessy), 2nd Edition**
- **Reading assignment: P&H Chapter 1**
 - » Light reading, for perspective
- **Course Information, Syllabus, Lecture Slides, etc. all on the web at <http://www-csag.ucsd.edu/teaching/cse141-w00/cse141.htm>**
 - » will be linked from department pages and my home page

CS 141 Chien

1

Jan 11, 2000

Course Staff

◆ Professor Andrew Chien (CSE141)

- 4808 AP&M, x2-2458, achien@cs.ucsd.edu
- Office Hrs: after lectures (TuTh355pm, and by appointment)

◆ Professor Dennis Bauman (CSE141L)

◆ Course Secy: Lisa Bodecker, 4016 APM (bodecker@cs.ucsd.edu)

◆ Teaching Assistants

- Andre Barroso abarroso@cs.ucsd.edu
- Abdulaziz Al-Shammari, shammari@cs.ucsd.edu
- Bao Liu (CSE 141L), bliu@cs.ucsd.edu
- **Office Hours to be announced.**
- Electronic mail is a good way to get ahold of us. We'll try to set up a class newsgroup as well. Please read this regularly, particularly close to when assignments are due or exams.

CS 141 Chien

2

Jan 11, 2000

Course Meetings

- ◆ Lectures: TuTh355-515, HSS 1330
- ◆ Section
 - M1115-1205, Center 105
- ◆ All material in lectures, readings, sections, and homeworks.
- ◆ Overheads and course materials as possible will be made available electronically on web site.
 - Check often for new information
- ◆ or try TA's or Course Secretary

CS 141 Chien

3

Jan 11, 2000

Coursework Material, Grading

- ◆ Material & Grading
 - ~4 homeworks (20%)
 - Collaboration: discuss issues, but you're best served by solving the problems yourself.
 - Late homeworks? Not accepted.
 - We'll try to give good turnaround with graded homeworks.
- ◆ Exams:
 - ~Weekly Quizzes (20%) – every Thursday at the *START* of class
 - Midterm (25%)
 - Final exam (35%)
- ◆ Grade weights are approximate.

CS 141 Chien

4

Jan 11, 2000

Background

- ◆ Basic Boolean Algebra and Logic Design
- ◆ Combinational Circuits and Finite State Machines
- ◆ Computer Arithmetic and Basic Arithmetic Circuits
- ◆ C (or C++) programming knowledge
- ◆ => If any of these following terms make you uncomfortable, you should talk to me about it.

Boolean Algebra and Basic Logic Design

- ◆ Complete logic functions
- ◆ $A = (B + C) * D$
- ◆ DeMorgan's law
- ◆ Canonical sum-of-products, product of sums representations
- ◆ Karnaugh Map
- ◆ => these are the building blocks for the “computer architectures” we discuss in this class

Combinational Circuits and Finite State Machines

- ◆ Relation of Boolean algebra to circuits
- ◆ Implications of fan-in, fan-out
- ◆ Circuit complexity in terms of gates, delays, etc.
- ◆ State: what it is and how to implement it using gates; does the simple digital logic model apply here?
- ◆ Finite states, transitions, capabilities of such machines, and how to design one, given a problem.

CS 141 Chien

7

Jan 11, 2000

Example: FSM Design

- ◆ Your job is to design a streetlight controller (G, Y, R) for a basic two-street intersection
 - Show the FSM design
 - Show an implementation using gates and state elements (FlipFlops)
- ◆ Modify that FSM to support an intersection like that at Torrey Pines and La Jolla Village Drive
 - Asymmetric street traffic
 - Left turn arrows
 - Pedestrian crossing buttons

CS 141 Chien

8

Jan 11, 2000

Computer Arithmetic and Basic Arithmetic Circuits

- ◆ Basic number representations (integers -- twos complement, floating point numbers, concepts of limited precision)
- ◆ Adders (basic, ripple carry, faster methods and circuits, subtraction)
- ◆ More complex operations (multiplication, division?)
- ◆ Floating point arithmetic
- ◆ => building blocks for a computer, we'll focus mostly on the other pieces (stuff that controls how these operations are composed into a *computation*)

CS 141 Chien

9

Jan 11, 2000

Basic C (or C++) programming knowledge

- ◆ C expressions, types (int, float, char, etc.)
- ◆ C/C++ structs and classes
- ◆ Control structures (while, for, break, goto, etc.)
- ◆ Functions calls
- ◆ General comfort and familiarity with these things.
- ◆ => These language constructs are rather low level, and we'll show their close correspondence to assembly/machine instructions.
- ◆ => Important to know what they mean (function) so implementations and correspondence make sense.

CS 141 Chien

10

Jan 11, 2000

Course Overview

- ◆ **The Idea of Computer Architecture**
- ◆ **Performance and Cost**
- ◆ **Machine Language: Elements and Utility**
- ◆ **The Relationship of Higher level languages and Machine Language**
- ◆ **Implementation of a Machine Language**
 - Basic Processor
 - Pipelined
 - Cache Memory
- ◆ **Computer Systems (outside the processor)**

CS 141 Chien

11

Jan 11, 2000

What should you get out of this course?

- ◆ **Appreciation of the basic elements of a computer system**
 - Levels of software (HL language, assembly, hardware)
 - Elements of the System (processor, cache, memory, I/O, peripherals)
- ◆ **Detailed understanding of how programming language features and the relationship of programming constructs to their execution cost**
- ◆ **Detailed understanding of instruction execution and some basic techniques for high performance**

CS 141 Chien

12

Jan 11, 2000

Elements of Computer Systems

Application Software: Word, Excel, Lotus, SimCity

Operating System: MSWindows, MacOS, DOS, Unix

Instruction Processor: Pentium, PowerPC, 486, 68040

Hardware Technology: Logic Gates, Transistors, etc.

◆ **Computer Systems are a series of interfaces**

- There's lots of complex activity going on inside your computer.
- Millions of transistors (one per bit of memory!)
- Hundreds of simultaneous activities
- All of this going on at speeds of up to 600 Megahertz!

CS 141 Chien

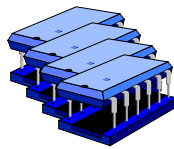
13

Jan 11, 2000

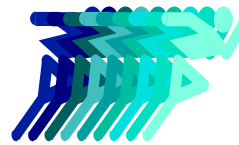
Hardware in Computer Systems



**Chips with Millions
of transistors**



**Tens of Chips
in a computer**



**Running at 600 Million
operations per second!**

- ◆ **Millions of transistors in 0.5"x0.5" square (1.25cm x 1.25cm) -- too small to see!**
- ◆ **Many chips connected together into a computer**
- ◆ **All of this operating at VERY high speeds**

CS 141 Chien

14

Jan 11, 2000

How fast are things happening?

- ◆ **Millions of transistors**

- Each much smaller than the cross-section of a human hair (100 microns versus 0.5 microns on a side)
- Each much smaller than the dots produced by a laser printer (typically 300 dpi or 600 dpi)

- ◆ **800 Megahertz = 800 million events / second**

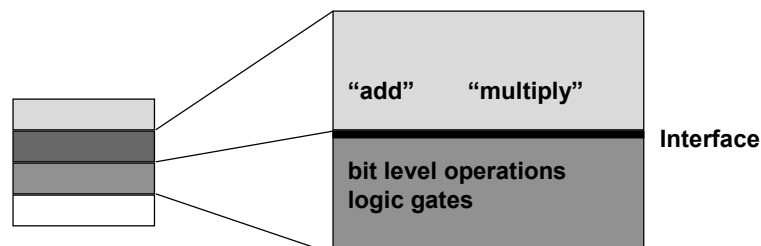
- TV screen - 30 or 60 frames per second
- Mach 1 (speed of sound) = ~ 600 mph = ~ 1000 feet per second
- Speed of light = 300,000,000 meters/sec = 900,000,000 ft/sec
- => light can only go ~1 foot in the time an operation occurs in a computer system!

CS 141 Chien

15

Jan 11, 2000

How can we track so many things?



- ◆ **Layers in the computer system assemble smaller activities into larger operations**
- ◆ **Higher levels need only keep track of larger operations**
- ◆ **The assembly of smaller operations into larger ones is critical.**
- ◆ **Set of large operations = an interface**

CS 141 Chien

16

Jan 11, 2000

Complexity Management

Large Operations:
Fill
Empty
Level?

Small Operations:
Open cover
Lift
Turn upside down
Put in dipstick
Read dipstick

- ◆ Idea is similar to object-oriented programming
- ◆ Large operations which form the interface are the external operations on an “object”
- ◆ Small operations are used to build the larger operations
- ◆ What does all this have to do with computer architecture?

CS 141 Chien

17

Jan 11, 2000

Computer Systems Architecture is the Design of Interfaces

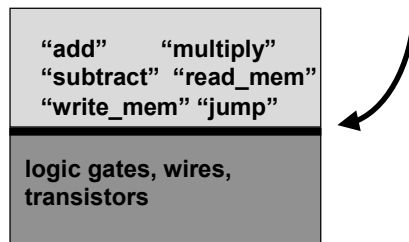
- ◆ Interfaces = abstraction and set of operations
- ◆ Computer systems include several levels of interfaces
- ◆ Computer Architects put pieces together to make “Computer Systems”
- ◆ Examples
 - User interface to MS Word, Lotus 1-2-3, SimCity
 - Application to Operating System interface, Graphics Toolboxes
 - Software to hardware interface -- computer instruction sets (we’ll focus here)
 - Traditionally, computer architecture = instruction set architecture, large operations are basic software instructions, small operations are built directly in hardware

CS 141 Chien

18

Jan 11, 2000

Computer Architecture: Instruction Set Architecture



- ◆ Software/Hardware interface, vertical interfaces
- ◆ ISA is critical, as a basis for software portability across machines
- ◆ ISA’s form the notion of processor families: 386--486—Pentium—PII--PIII, Sparc, MIPS, PowerPC—G4, 68000--68020--68040, IBM 360, etc.

CS 141 Chien

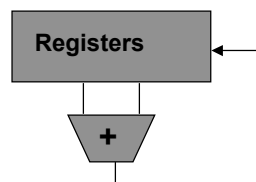
19

Jan 11, 2000

Interfaces are critical for Performance

add r2, r3, r4
mul r3,r4,r5

Operations



Abstraction

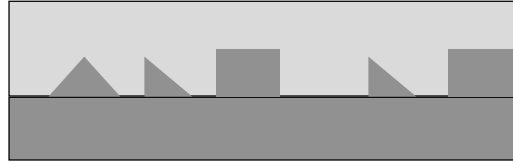
- ◆ Instruction sets <--> hardware organization
- ◆ Simple Instruction Sets (simple operations)
 - Very fast operation rate
- ◆ Complicated instruction Sets
 - Slower operation rate

CS 141 Chien

20

Jan 11, 2000

Interfaces are not Flat



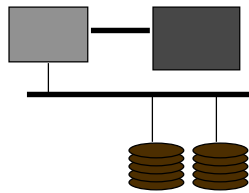
- ◆ Jagged interfaces reflect the operations exposed
 - ◆ Layers on top must “fill the gaps” themselves
 - ◆ Clean high-level interfaces make building atop easy, but may make it hard to get performance.
 - ◆ Lower level interfaces, make it hard to make things to work, and may make it hard to optimize performance.
- Performance is accessible.

CS 141 Chien

21

Jan 11, 2000

Computer SYSTEM Architecture: Horizontal Interfaces



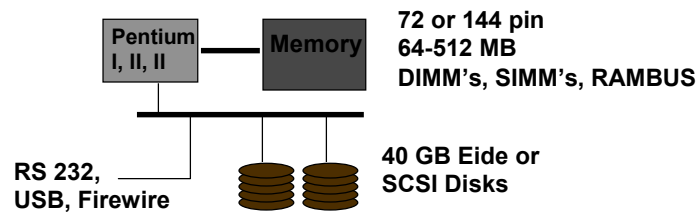
- ◆ Interfacing parts of the computer system: logical operations (protocols) and electrical signals (signalling)
- ◆ Design of interfaces, parts, which shape the SYSTEMs, influencing the performance, cost, and flexibility

CS 141 Chien

22

Jan 11, 2000

System Issues = Integration Issues



- ◆ Interfaces become standards, enabling interoperability -- must be general and cost effective (examples)
- ◆ Design of interfaces is critical, or they rapidly become performance bottlenecks
- ◆ Most useful interfaces are continually redesigned/extended

Scope of Computer Architecture

- ◆ Vertical (layering) and Horizontal (integration) interfaces
- ◆ How systems are organized, and the issues that drive that organization
- ◆ Interfaces have a major impact on performance
- ◆ => Architecture is key to delivering system performance!

Summary

- ◆ **Central issue in computer architecture is the design of interfaces to manage complexity while delivering performance.**
- ◆ **Computer architects design both the vertical interfaces and horizontal interfaces that are needed.**
- ◆ **We'll be exploring these issues in the coming quarter...**