Computer Organization

Introduction
CS301
Prof. Szajda
Fall 2017
Course Logistics

- **Prof Szajda**
  - Jepson 219
  - dszajda@richmond.edu
  - 287–6671

- **Meeting Times**
  - Lecture: TR 10:30–11:45 (in Jepson 231)
  - Lab: W 12:00–12:50 (in Jepson G30)
  - Office Hours:
    - TBD
    - and by appointment
Course Logistics

Grading
- Two Tests 30%
- Final Exam 25%
- Labs 20%
- Homework 10%
- Final Programming Project 15%

Important Dates
- Thurs., Oct. 5
  - Exam 1
- Thurs., Nov 16
  - Exam 2
- Thurs., Dec. 14
  - Final Exam (9am–noon)
• Computer Organization and Design: The Hardware/Software Interface, 5th Edition, by Patterson and Hennessey
Graded Work Policies

- Collaboration
  - You may discuss homework and other non-exam assignments with other students in this class
  - "Empty Hands" policy (see syllabus)
    - Must leave any discussion/communication without any written or otherwise recorded material
    - Must note who you worked with on assignment

- Late Work is absolutely not accepted!
  - It is difficult enough to keep up with grading in this course even when material is submitted on time!
Attendance Policy

• I expect you to attend class and to participate (meaning, don’t come if you’re going to sleep)
• DO NOT use your laptop/tablet/digital device during class for any function other than taking notes.
  ♦ Surf the web and read email on your own time
• If you miss 4 or more days of class (including labs), I can (and will) give you a grade of “V”
Reading

- Read over syllabus
- Read Chapter 1
What is this Course About?

• How do we design today’s computer systems?
  - Intel Core i7 at 3.3 GHz
  - 6 cores – 2 threads each
  - 15 MB Shared cache
  - 64 GB Main Memory

• Starting with something that can only represent a 0 or a 1?

  Transistor
Basics Behind Lots of Processors
High Level Info About Course

- Required for major or minor
- Prerequisite for many upper level courses
  - Material and skills you learn will be necessary for later courses

Of course, what you get out of this course depends on the effort you put into it!
Specific Topics

- **SW/HW Interface**
  - Assembly languages and instruction encoding
- **Processor Construction and Design**
  - How to build simple processor from simple circuits
- **Memory System Design**
  - How to construct a memory system that keeps processor fed
- **I/O Devices**
  - How processor interacts with disk, mouse, etc.
Skill Development

- Many of these are taught in CS240 and will be further developed via assignments in this course
  - Object oriented design
  - Systematic testing
  - Debugging with a debugger
  - Learning on your own
Computer Architecture Overview
What is a Computer?

Architecture

- Design transistor technology
- Optimize layout, circuits, etc
- Design processor
- Design assembly language
- Write compilers
- Program software

[Diagram showing the relationship between the steps and architecture]
Where do Logic Circuits Fit?

- Program software
- Write compilers
- Design assembly language
- Design processor
- Optimize layout, circuits, etc
- Design transistor technology

How do I put together registers, adders, SRAM, etc?
Where do Logic Circuits Fit?

- Program software
- Write compilers
- Design assembly language
- Design processor
- Optimize layout, circuits, etc
- Design transistor technology

How do I design register files, adders, etc. out of boolean gates?
Where do Logic Circuits Fit?

- Program software
- Write compilers
- Design assembly language
- Design processor
- Optimize layout, circuits, etc
- Design transistor technology

How do I design boolean gates out of transistors and put them on silicon?
What about Software?

- Program software
- Write compilers
- Design assembly language
- Design processor
- Optimize layout, circuits, etc
- Design transistor technology
Where do HW and SW Meet?

- Program software
- Write compilers
- Design assembly language
- Design processor
- Optimize layout, circuits, etc
- Design transistor technology

Hardware / Software Interface
System Components

- Processor
  - Datapath
  - Control
- Memory
- Input and output (I/O)
Anatomy of a Computer

Output device

Network cable

Input device

Input device
Mice

- Optical mouse
  - LED illuminates desktop
  - Small low-res camera
  - Basic image processor
    - Looks for x, y movement
  - Buttons & wheel
- Supersedes roller-ball mechanical mouse
Display

- LCD screen: picture elements (pixels)
  - Mirrors content of frame buffer memory
DIMM
Nonvolatile Storage

- **Volatile main memory**
  - Loses instructions and data when power off
- **Non-volatile secondary memory**
  - Magnetic disk
  - Flash memory
  - Optical disk (CDROM, DVD)
Networks

- Communication and resource sharing
- Local area network (LAN): Ethernet
  - Within a building
- Wide area network (WAN): the Internet
- Wireless network: WiFi, Bluetooth
Software Terminology

Instruction Set Architecture (ISA)

Operating System vs. User program

System Software: includes OS and compiler
Software Terminology

- Binary or executable
  - Compiler
  - Assembler
Why binary?

In order to build circuits that implement logic, we need voltage-controlled switches:
- Control input = 1 \implies \text{Switch is closed}
- Control input = 0 \implies \text{Switch is open}

This can be accomplished with electro-mechanical relays:
- Large, clunky, power-hungry
- Transistors are a better way:
  - Tiny, efficient, fast

Three slides from [http://oa-003.spu.edu/bolding/EE1210/070-NMOS-CMOS.ppt](http://oa-003.spu.edu/bolding/EE1210/070-NMOS-CMOS.ppt)
MOS Semiconductor

MOS: “Metal Oxide Semiconductor” this is nMOS (source/drain n-type)

**P-type silicon**: Excess positive charges (electron holes)

**N-type silicon**: Excess negative charges (electrons)

**Oxide**: Insulator

**Gate**: Metal pad

In this state, current (electrons) **cannot flow** between source and drain – **switch is OPEN**
Place a **positive** charge on the gate wire (gate = +5V)

The gate’s **positive charge** attracts **negatively-charged electrons**

This **row of electrons** forms a **channel** connecting the Source and Drain – **Current can flow** – **Switch is CLOSED**
Transistors

- Transistors
  - Store 0 or 1 when on or off
  - Can connect transistors in series or parallel to create larger building blocks called **gates**

![CMOS Inverter created from two transistors](image)

**CMOS**: Complementary Metal Oxide Semiconductor
Technology: Microprocessor Logic Density

[Graph showing the increase in logic density from 1976 to 2008, with key points at 18K, 64K, 258K, 1M, 4M, 18M, 64M, 128M, 256M, 512M, and 1G.]
Technology: Microprocessor Logic Density

Source: http://www.nature.com/nature/journal/v479/n7373/full/nature10676.html
Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)
Billion Transistor Chips

Intel Core i7 Extreme Edition - 2.27 billion transistors, 435 mm$^2$ area
Growing Silicon

- Silicon is a crystal grown in a vat.
- It comes out the shape of a cylinder.
- This is called an ingot.
Creating Chips

• Sliced into thin discs called **wafers**
• Etch grooves and pour metal, etc
  - 20 - 40 steps
• Cut the wafer into **dies** or **chips**
• A flaw is called a **defect**
• The percentage of good ones is **yield**

Yield: 8/10 = 80% (not realistic)
Manufacturers secretive about yields, but can be as low as 30%
Cost

- **Cost per die**
  - \( \frac{\text{CostPerWafer}}{((\text{DiesPerWafer})\times\text{Yield})} \)
- **Dies per wafer**
  - \( \frac{\text{Wafer area}}{\text{Die area}} \) – wasted edge space
- **Yield**
  - \( \frac{1}{1 + (\text{DefectPerArea} \times \text{DieArea}/2))^2} \)
    - 2 in denominator is “alpha” which is determined by number of masking levels used (a measure of manufacturing complexity)
- **Cheapest when yield is high** and **dies per wafer are high**
Current Chip Trends

- Shrinking Technology
  - Reported in **microns** (width of wire)
  - Each generation allows more to fit in same space
  - **Defect rate** gradually falls in time with same technology

- Increasing Area
  - Yield – Increases chance of a defect on die
  - Dies/wafer – fewer dies, more wasted space