



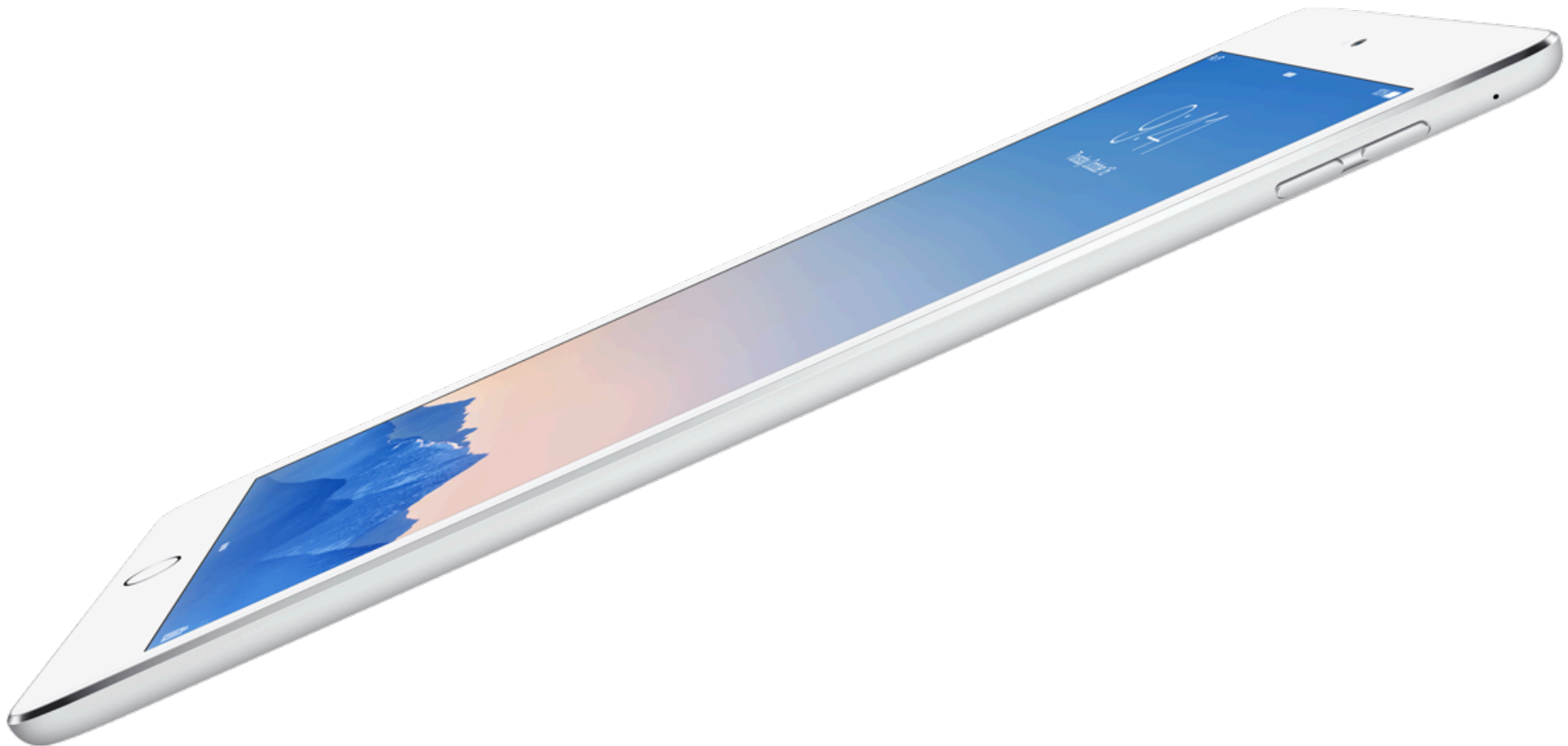
# Computer Systems and Networks

ECPE 170 – Jeff Shafer – University of the Pacific

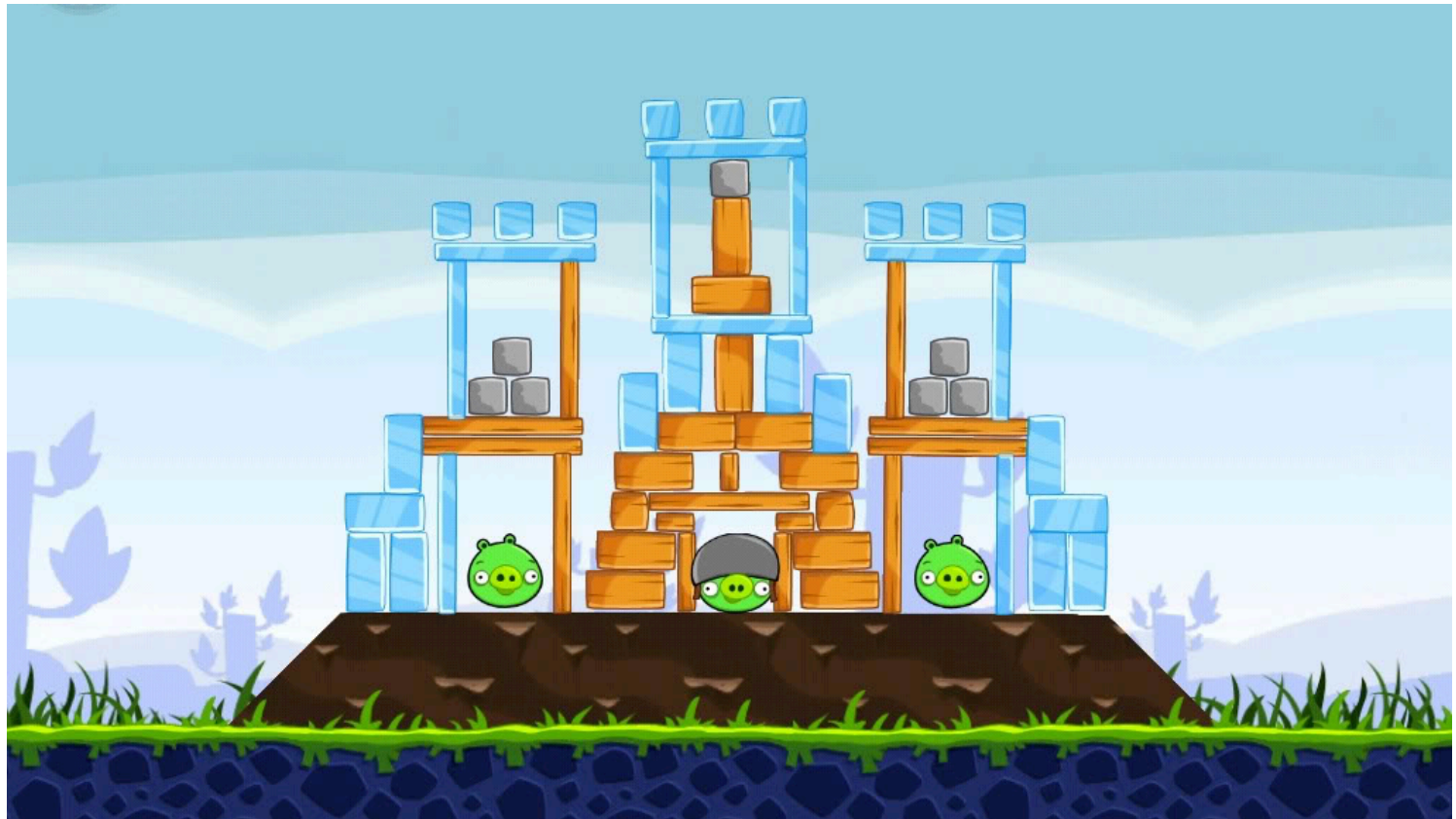
# Introduction

---

# A Modern Computer – iPad Air “2”



# Applications



# Application – Angry Birds

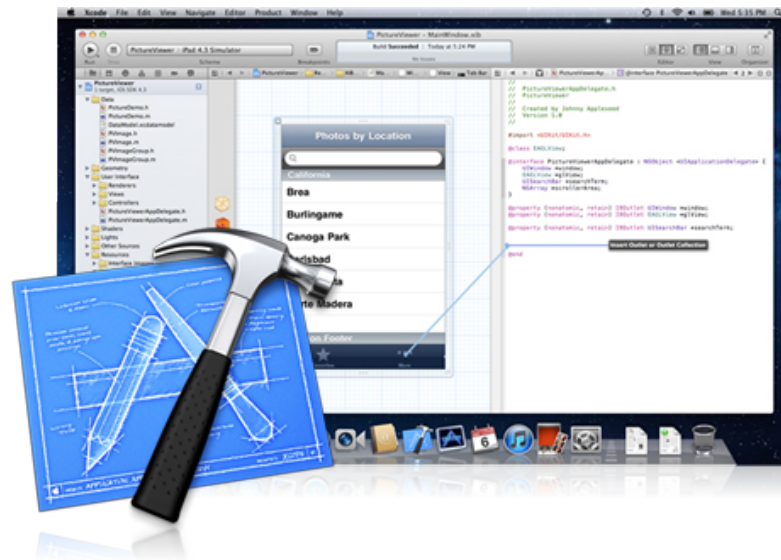
- Written in a high level language (Objective C)
- What **resources** does *Angry Birds* need to run? (i.e. what does the *Angry Birds* executable file need to execute?)
  - Hardware
    - Processor(s) – Run program, display graphics, ...
    - Memory – Store programs, store data
    - I/O – Touch screen, storage, network, 3-axis gyro, ...
  - Software - Operating system

# Software - Operating System

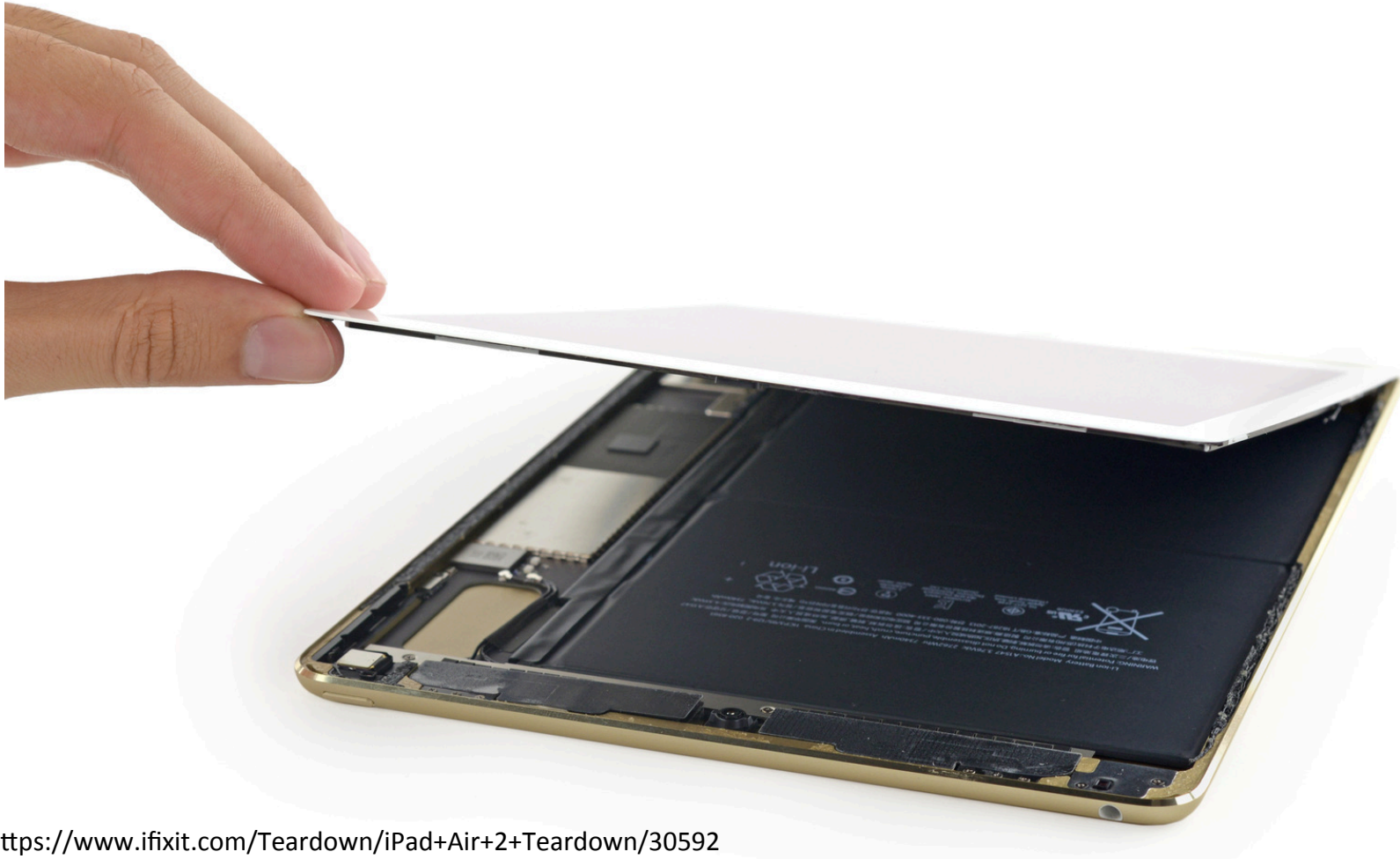
- Apple iOS – Used in iPads, iPhones, iPods, Apple TV
  - Variant of Mac OS X operating system used on traditional Macs
  
- **What are some jobs of this operating system?**
  - Manage hardware
  - Manage applications (multitasking)
  
- Written in high-level languages
  - C, C++, Objective C (varies by component)
  - **Can we run this code directly on the processor?**

# Software - Compilers / Interpreters

- These are programs that **build** other programs!
- Goal: Convert high-level languages into machine code that can be directly executed by hardware
- Examples
  - Apple Xcode
  - Microsoft Visual Studio
- **What's the difference between a compiler and interpreter?**



# Hardware

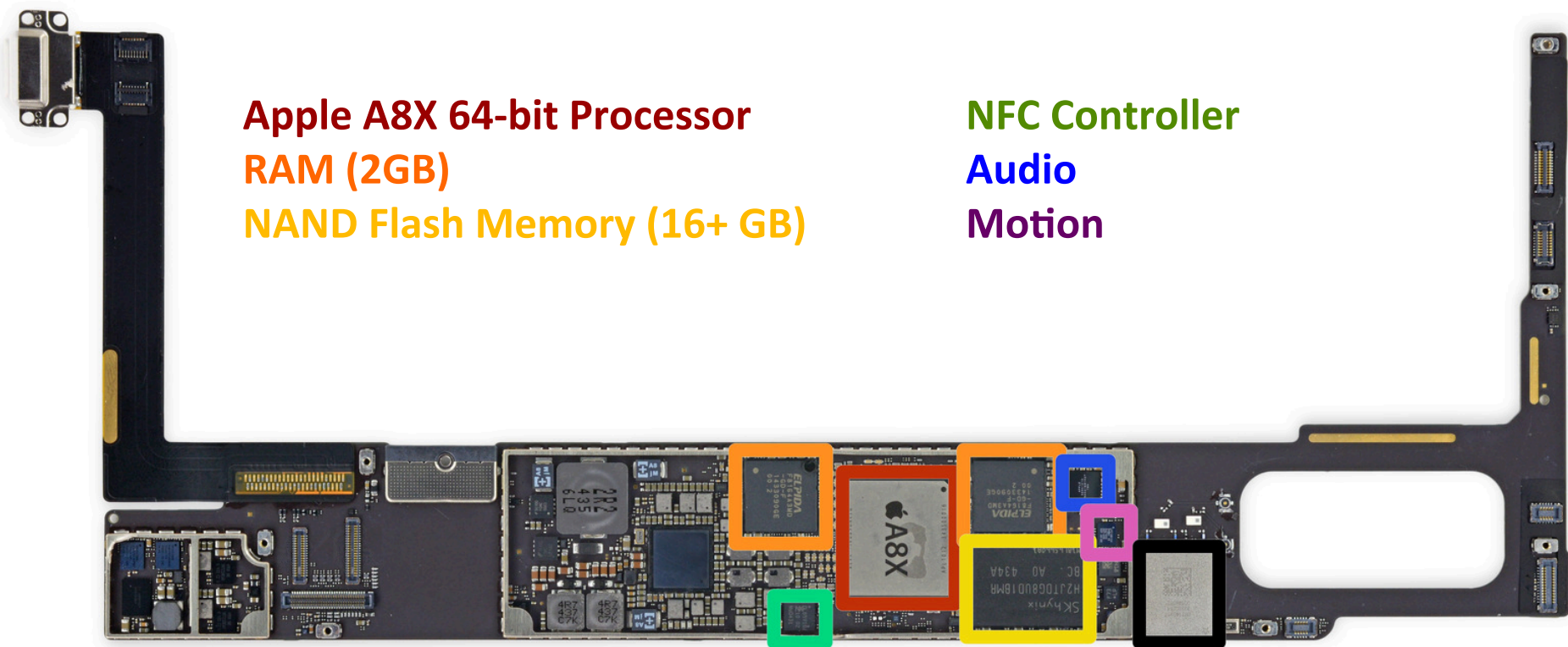


<https://www.ifixit.com/Teardown/iPad+Air+2+Teardown/30592>

# Hardware

**Apple A8X 64-bit Processor**  
**RAM (2GB)**  
**NAND Flash Memory (16+ GB)**

**NFC Controller**  
**Audio**  
**Motion**





# iPad Air “2” Processor

## ➤ Apple A8X Processor

➤ Clock speed – 1.5GHz

➤ 3 cores

➤ 2GB RAM

} What do these mean?

## ➤ **What does a processor do?**

➤ Executes machine language instructions

➤ **Machine language?**

➤ **How does the processor execute the instructions?**

# Microarchitecture



# How Does It Work?

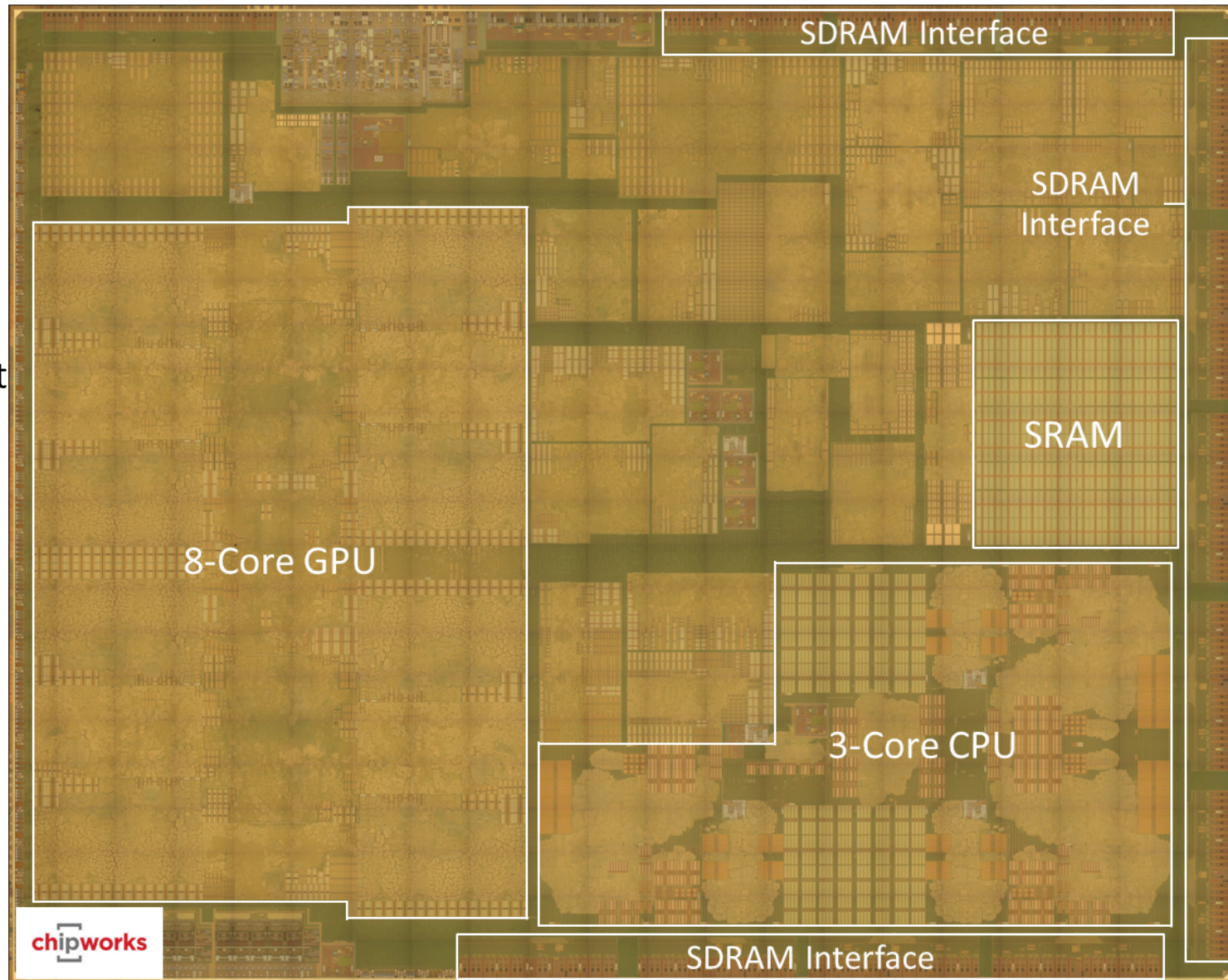
- Apple won't tell us – trade secret!
- Experts can dissolve (with acid), burn, or grind off outer protective layers of chip and then peer inside:
  - Need a *really good* microscope!
  - *Reverse Engineering in the Semiconductor Industry:*  
<http://www.scribd.com/doc/53742174/Reverse-Engineering>

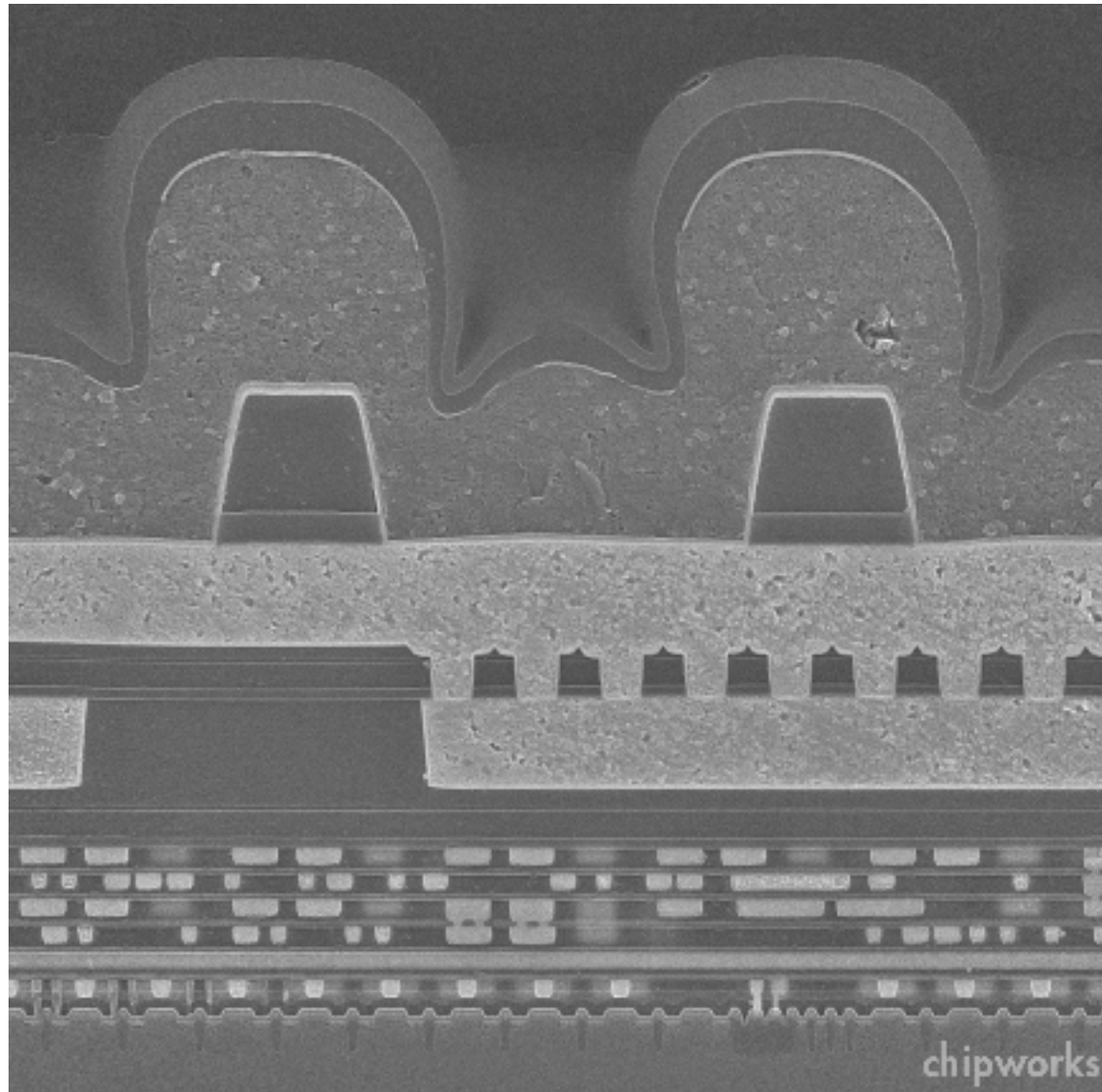


*Can see this level of detail with your own eyes...*

Divided into logic blocks with different functions:

- Processor
- Cache memory
- Memory Controller
- Video (GPU)

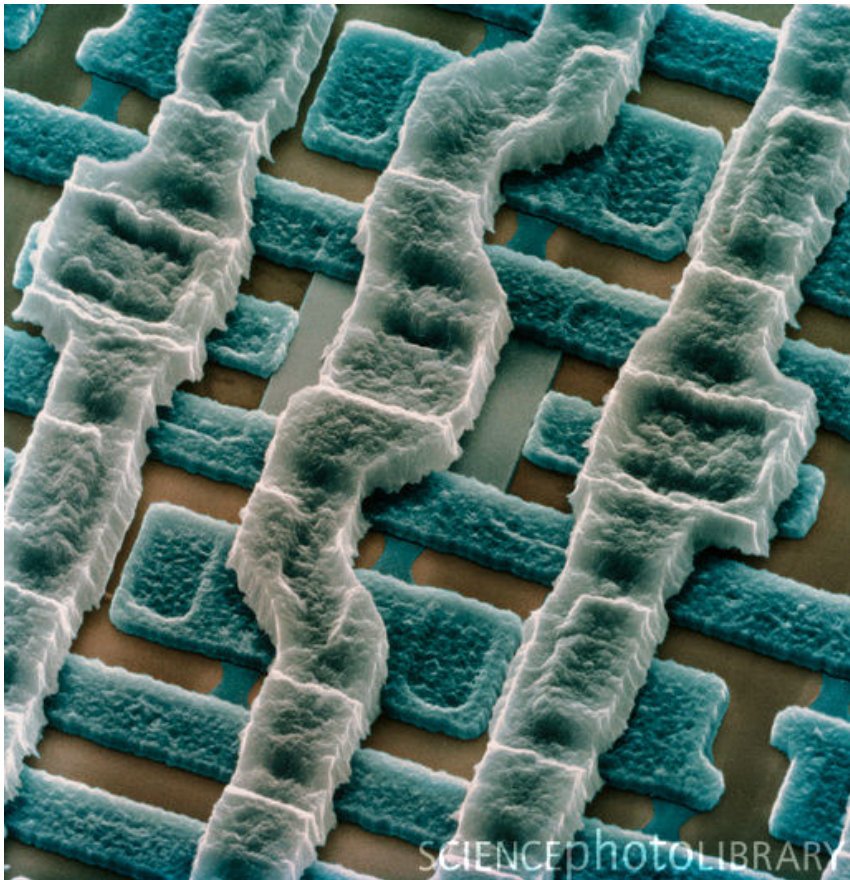




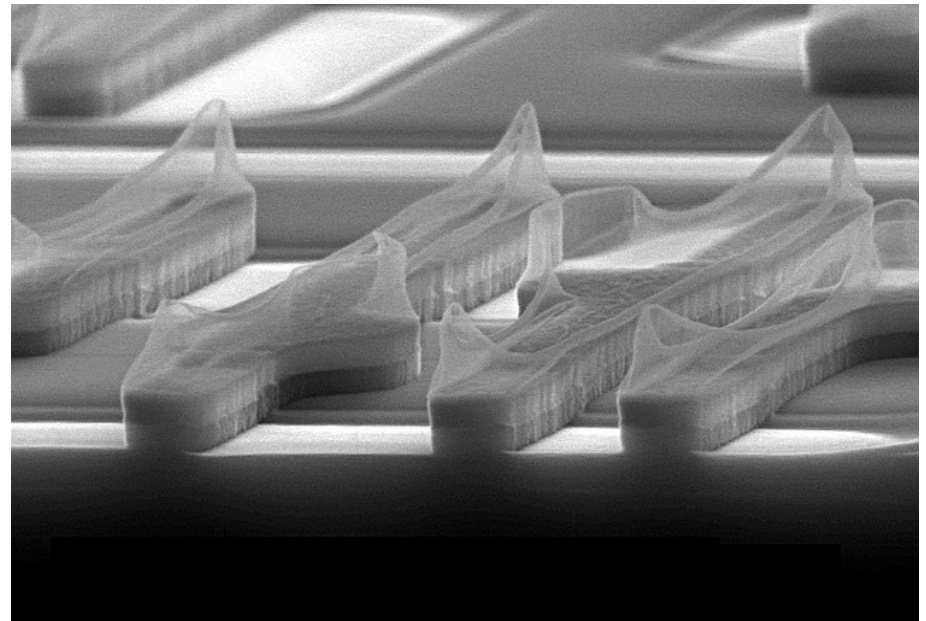
SEM Cross-Section of (older) Apple A5

# Digital Logic

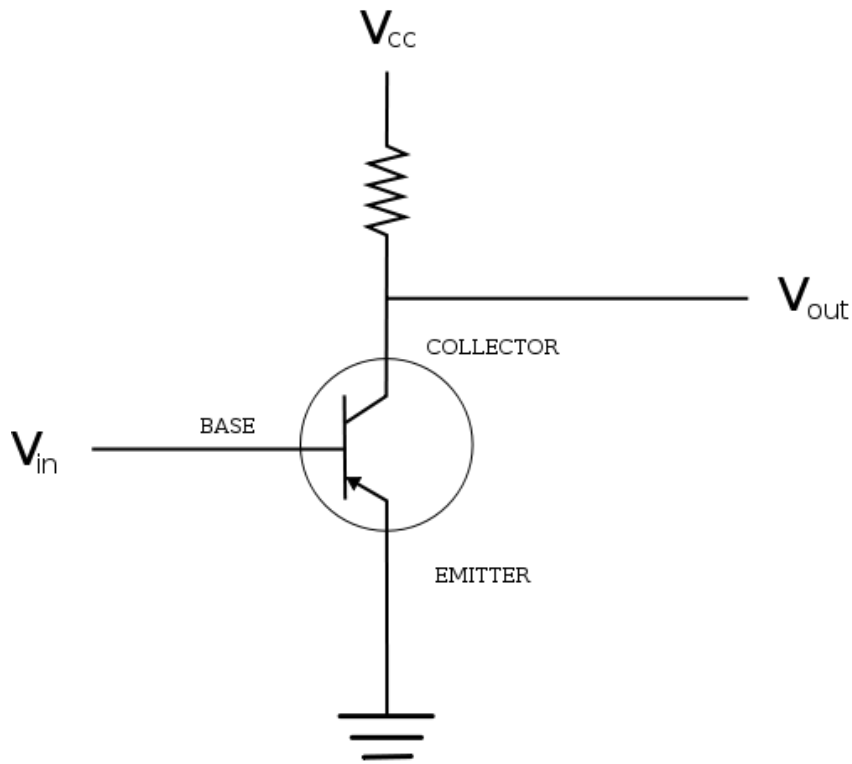
Memory cell



Transistor

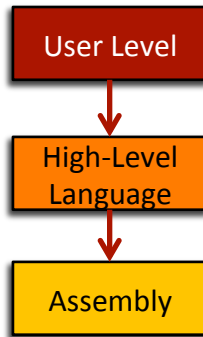


# Transistors



- You can still make assumptions at this level that the transistor is either “on” (1) or “off” (0)
- But below this are **analog circuits**

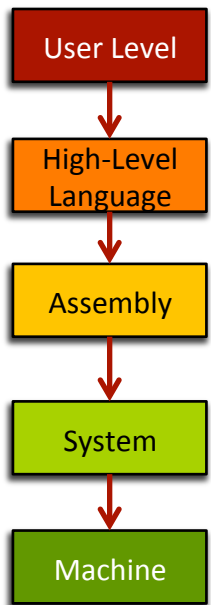
# The Computer Level Hierarchy



- Level 6: The **User Level** – “Angry Birds”
  - Program execution and **user interface** level
- Level 5: **High-Level Language Level** – “Objective C”
  - Programming languages like C++, Java, Python, ...
- Level 4: **Assembly Language Level** – “ARM Assembly”
  - Program directly at this level, or ...
  - **Use a compiler/interpreter** to process/convert high-level code



# The Computer Level Hierarchy



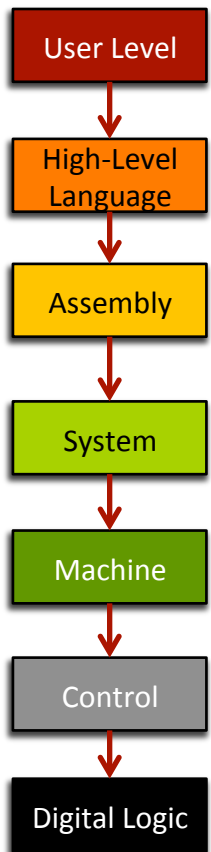
## ➤ Level 3: **System Software Level** - “iOS”

- Controls active programs and manages system resources
- Assembly language instructions often pass through Level 3 without modification

## ➤ Level 2: **Machine Level**

- Instruction Set Architecture (ISA) Level
- Instructions are particular to the architecture of the specific machine (i.e. Intel processors, ARM processors, IBM processors...)

# The Computer Level Hierarchy



*These levels are too hardware-oriented for ECPE 170...*

## ➤ Level 1: **Control Level**

- Decodes and executes instructions and moves data through the system
- **ECPE 173 – Computer Organization & Architecture**

## ➤ Level 0: **Digital Logic Level**

- Digital circuits, gates and wires implement the mathematical logic of all other levels
- **ECPE 71 – Digital Design**  
**ECPE 174 – Advanced Digital Design**

# Course Overview



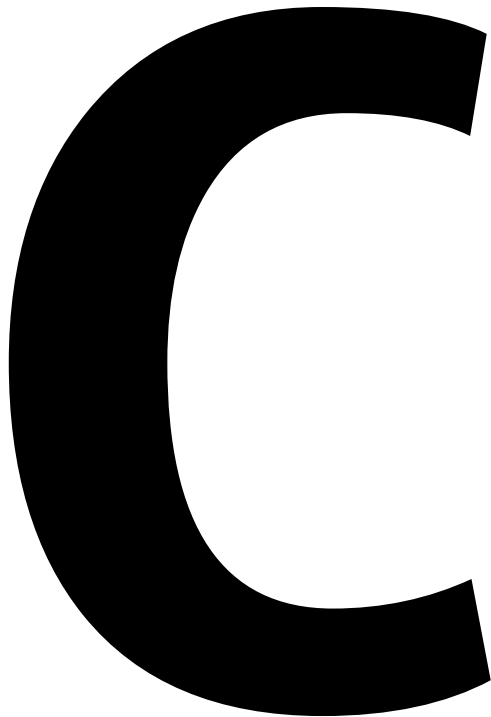
# Motivating Question

- **What do you, as a programmer, need to know about the underlying system (software *and* hardware) to write more efficient code?**
  - Role of the tools
    - Compiler, assembler, linker, profiler
  - Role of the operating system and its efficient usage
  - Assembly programming (using the CPU efficiently)
  - Memory hierarchy and its impact on performance

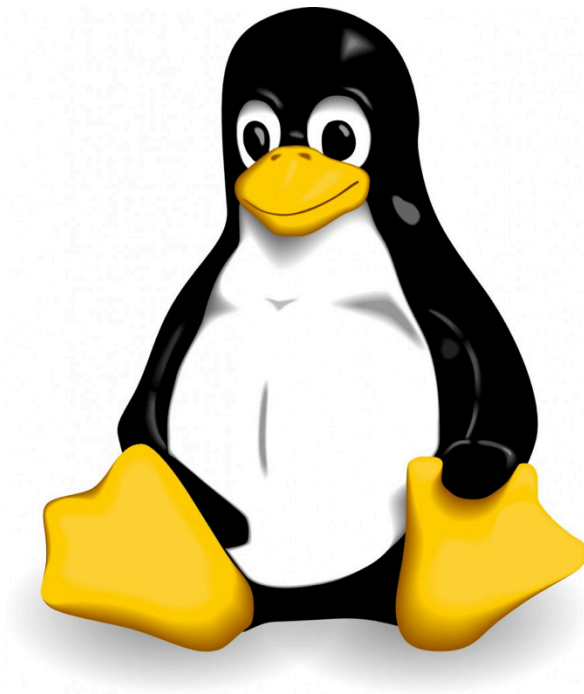
# Course Goals

- Present a complete view of how computer systems are constructed
  - From the CPU assembly programming level to the user application level
- Understand the relationship between computer software and hardware
- Lay the foundation for future courses
  - Advanced Digital design / VLSI
  - Operating systems
  - Computer networking
  - Application development

# C Programming Language

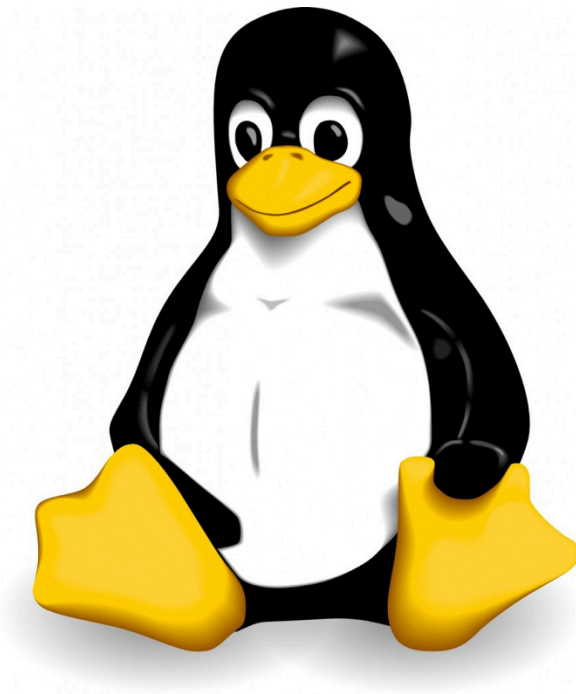


- **Why not Python, Java, Ruby, Perl, PHP, ...?**
- High-level languages (especially interpreted, managed code...) try to *hide* the underlying machine from you
- ECPE 170 wants to *reveal* the underlying machine to you!



- Course will be taught 100% in Linux
- *Did you have to choose Linux for ECPE 170?*
- No, not really, but...
  - Too many Pacific graduates were *escaping* without a working knowledge!
  - **Feedback from co-op employers and graduates: “More Linux/Unix skills please!”**

# Linux



- **Who here has used a Linux desktop/laptop/server before?**
- **Who here has used a Linux “device” before?**
  - *I’d be surprised if it isn’t everyone...*
  - Android runs a Linux kernel
  - Amazon Kindle runs a Linux kernel
  - TiVO runs a Linux kernel



# Discussion

- **What is open-source?**
- **What is an operating system *kernel*?**
  - **Is the kernel everything you need from an OS?**
- **What is Linux?**
- **What is Ubuntu Linux? (RedHat? Debian? ...)**
  - **→ Show family tree of distributions ←**

# Virtual Machine



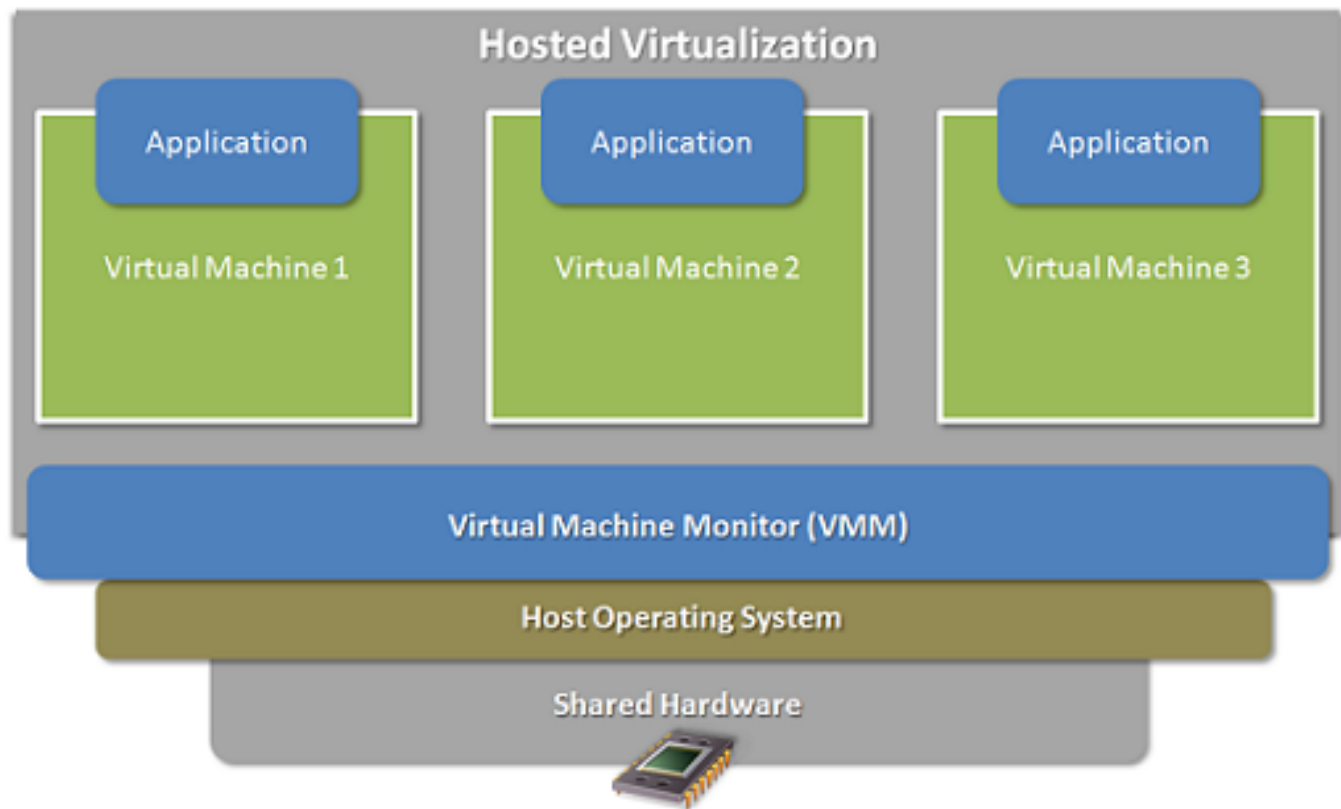
- **Course will be taught 100% from a virtual machine booting Linux that you install!**
- *Couldn't you just give us remote access to a server someplace that is already configured?*
- Yes, but...
  - By installing it yourself you will have the skills to use it again in the future
  - No mysterious "Professor Shafer" software configuration

# Discussion

- **What is a Virtual Machine?**
  - Is this the same thing as a *Java* virtual machine?
- **How is it different from dual booting?**
- **Which comes first, the virtual machine, or the OS?**
  - Answer: It depends!
  - Typical desktop install: hosted virtualization
  - Typical server install: bare-metal virtualization

Recommended  
technique for ECPE  
170

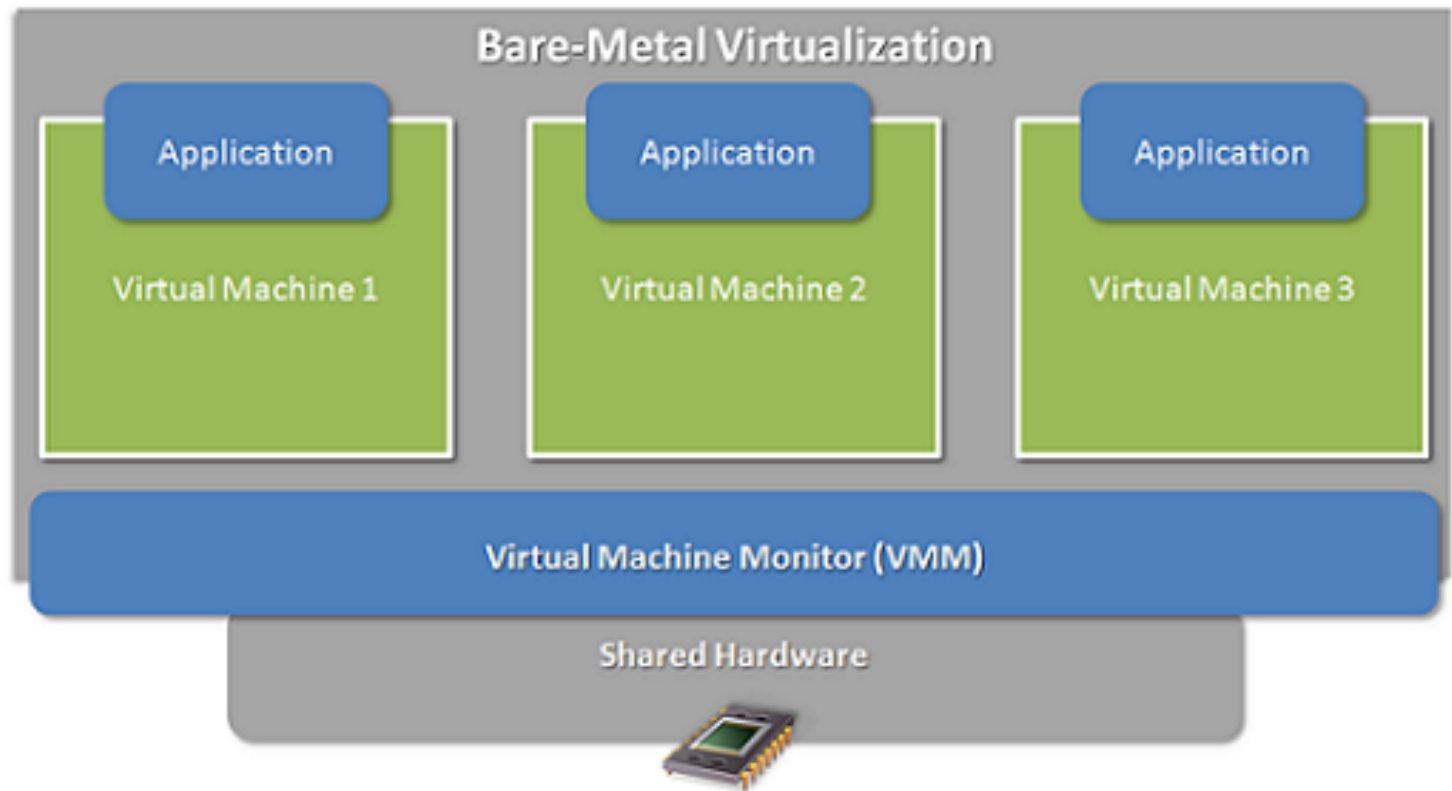
# Hosted Virtualization



# Bare-Metal Virtualization

More efficient, but not as easy to install.

The virtual machine monitor acts like an operating system itself!



# Version Control



## ➤ Course will use version control!

➤ Only way to get lab code or turn in assignments

➤ *Did you have to mandate VCS for ECPE 170?*

➤ No, not really, but...

➤ Too many Pacific graduates were *avoiding* learning this on their own!

➤ **Feedback from co-op employers and graduates: "Only n00bs work without version control!"**

➤ Used everywhere: Source code of all kinds! (C++, Python, Matlab, VHDL/Verilog, ...)

# Version Control



➤ **Who here has used a *version control system* before?**

- What system?
- Where at?
- What purpose?



**git**



# Questions?

➤ Questions?

➤ Concerns?



# Course Mechanics



# Websites

## Main website (syllabus, schedule)

- <http://ecs-network.serv.pacific.edu/ecpe-170>

## Canvas website (gradebook)

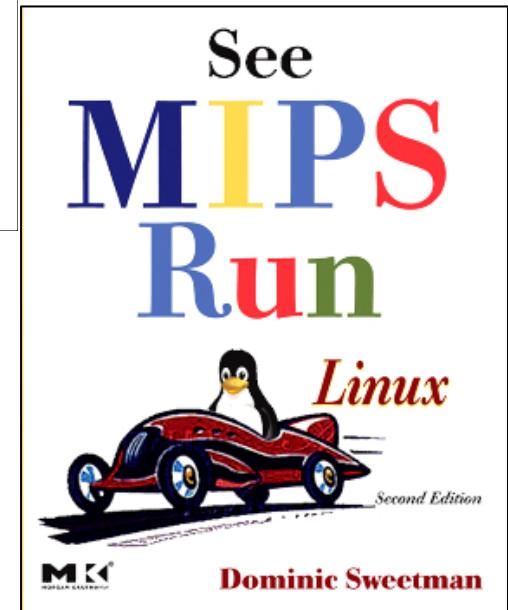
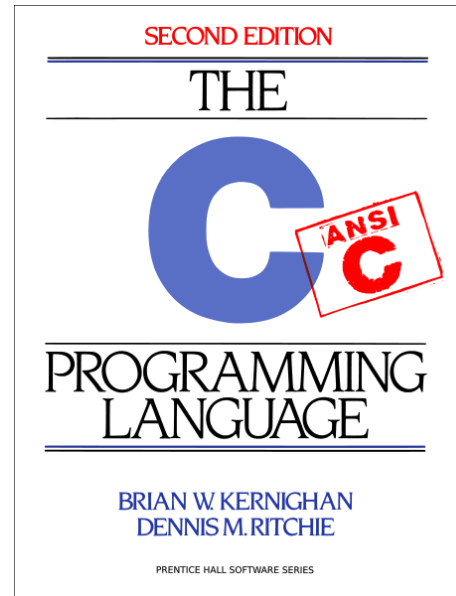
- <http://canvas.pacific.edu>

## Bitbucket.org (version control)

- <http://bitbucket.org>

# Textbook

- **No official textbook**
- Optional reference books (useful for this class and beyond)
  - The C Programming Language, 2<sup>nd</sup> Edition
  - See MIPS Run, 2<sup>nd</sup> Edition
- **Please suggest useful online or print references throughout the semester**



# Grading

## ➤ 30% - Exams

- 15% - Mid-term exam
- 15% - Final exam

## ➤ 70% - Labs

- Points assigned to each lab will vary based on complexity
- Each lab *begins* as an in-class activity
  - Unfinished work becomes homework/project
  - **Labs are large – assume “the usual” amount of homework/projects for a 4-credit class**
- **Tip: The best students last semester *started* the labs outside of class, and finished them as an in-class activity**

# Honor Code

- All assignments are submitted individually
- Encouraged Activities
  - Collaborating with your classmates (asking questions, solving problems together)
  - Searching for solutions online
    - Provided code copied does not exceed 25% of total assignment length
    - Provided you clearly **document this copy** in your source code and lab report
      - What did you copy? Where did it come from?

# Honor Code

## ➤ **Risky Activities**

- Having your classmates type on your computer or assignment file

## ➤ **Forbidden Activities**

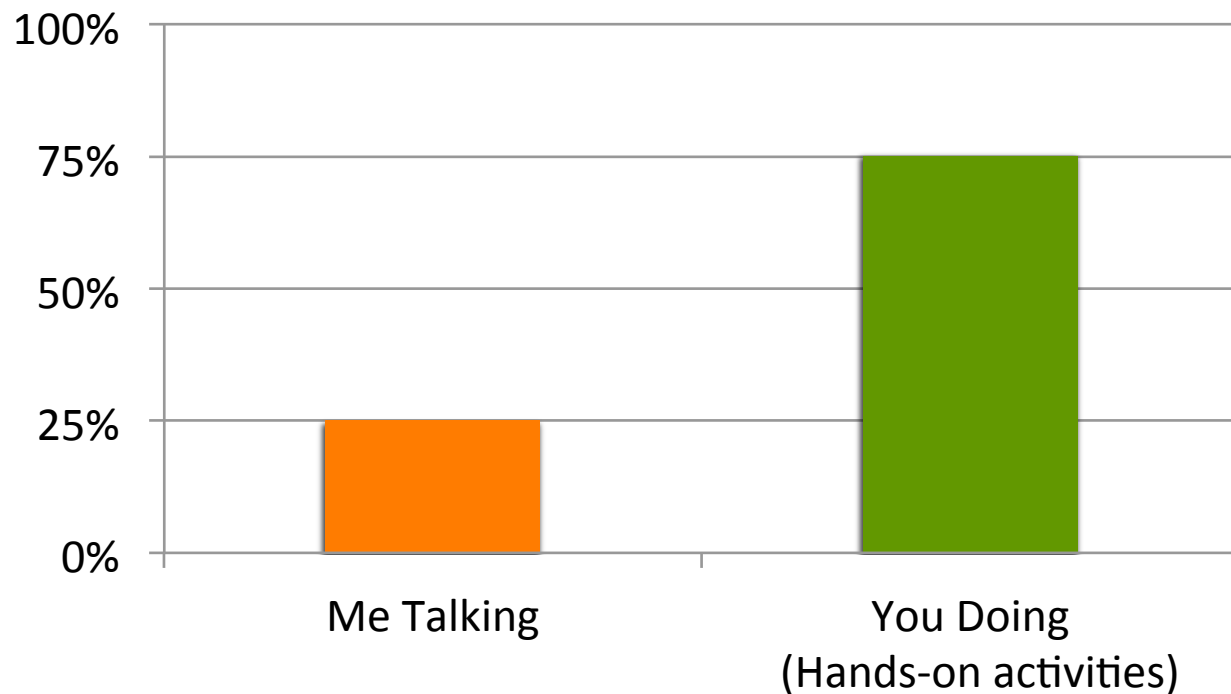
- Copying someone's work verbatim (classmate or otherwise)
- Copying someone's work and obfuscating its source

# Lab Topics

1. Linux
2. Version Control
3. C Programming
4. C Programming Project
5. Performance Measurement
6. Performance Optimization (compiler and programmer techniques)
7. Performance Optimization (Memory systems)
8. Network Programming 1 (Python)
9. Network Programming 2
10. Assembly Programming 1 (MIPS)
11. Assembly Programming 2
12. Assembly Programming 3

# Class Time

➤ The goal\* in designing this course:



\* Actual time in any specific class may vary



# Lab 1 - Linux



# Homework

## ➤ **Before the next class**

1. **Skim “Virtual Machine Setup” tutorial instructions on website**
  - [http://ecs-network.serv.pacific.edu/ecpe-170/tutorials/vm\\_setup](http://ecs-network.serv.pacific.edu/ecpe-170/tutorials/vm_setup)
2. **Decide on what computer system you want to use for this class**
3. **Download all software**
  - Virtual machine installer (VMWare Player)
  - Linux .iso image (installer) – 64-bit version

# Next Class - Linux Installfest

- Tutorial Day
- Objectives
  - Follow the “Virtual Machine Setup” tutorial from website to install Linux
  - Debug individual problems if needed
  - Verify OS works
  - **Email me screenshot as proof of success**

# Next Class - Linux Installfest

- I want you to be comfortable as professionals working independently to solve problems
- If you complete the “Virtual Machine Setup” tutorial independently (and email me a screenshot by Thursday morning), you don’t need to attend Thursday’s class. Sleep in! *(Or come help out)*
- I will still be here to answer all questions and solve problems

# Next Class - Linux Installfest

- ➔ **Warning: Don't skip class Thursday, and then tell me next Tuesday at Lab #1 that your OS doesn't work!**

# Lab 1 - Linux

- **The first lab is next Tuesday**
  - Topic: Linux
  - Crash course in command-line usage
  
- **Lab 1: Pre-Lab**
  - Show me the working command prompt in your Linux install. Hopefully you will have this done by end-of-class Thursday
  - **Pre-Labs are always due at the start of the lab**

# Bring Laptop!

**Every class – bring your laptop**



# Bring Laptop!

**Every class – bring your laptop!**





# Bring Laptop!

**Every class – bring your laptop!!**



(\* ) Maybe not this one, but you get the idea...

# Bring Laptop!

**Every class – bring your laptop!!**

**Just assume we'll do at least *some* lab activity in class unless it's been made crystal clear in advance that a day will be all lecture/discussion instead...**

# Bring Laptop!

- *No laptop? Let's try installing Linux to a USB stick and dual boot the classroom computers.*
- *See me after class to sign-out hardware...*

# Questions?

➤ Questions?

➤ Concerns?