# ECE 571 – Advanced Microprocessor-Based Design Lecture 1

Vince Weaver

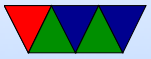
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

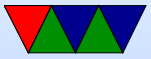
15 January 2013

# Introduction

- Distribute and go over syllabus
- Talk about the class

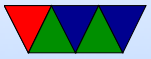


# Advanced Topics in Embedded Systems



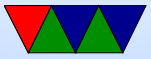
# What is an embedded system?

- Embedded. Traditionally fixed-purpose controller.
- Resource constrained. Small CPU, Memory, Display, Bandwidth
- Often real-time constraints.

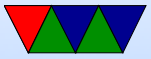


# What Size CPU/Memory?

- Anything from 8-bit/tiny RAM to 32-bit 1GHz 1GB
- Expanded widely over the years. ARM Cortex A9 in an iPad2 scores same on Linpack as an early Cray supercomputer



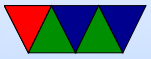
# Pushing the Limits



# What Processors Commonly Used?

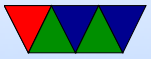
As reported by IDC at the SMART Technology conference in San Francisco for 2011

- ARM 71%
- MIPS 11%
- Other 9%
- x86 8% (at least Intel's desperately trying)
- Power 2%



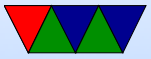
# We'll Use ARM

- Commonly used
- You'll see if it you move to industry
- Other classes in ECE are moving to it (271,471)



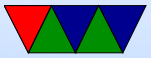
# We'll Use Linux

- Because I like it and understand it best
- Source code available
- Well-developed tools
- The ARM machine I have runs it



# Computer Architecture Review

- In-order Processors – Old 8-bits
- Super-scalar – multiple instructions "in-flight" at once. Original Pentium
- Out-of-order – Pentium Pro and Newer, Arm Cortex A15

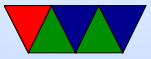


# RISC / CISC / VLIW

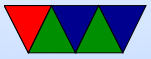
- RISC: Reduced Instruction Set Computer
  Small set of instructions to make processor design simpler. Usually fixed-length instructions, load/store

- CISC: Complex Instruction Set Computer
  Wide ranging complicated instructions; have complicated CPU decode circuitry. Often variable length instructions. Often allow operating on memory directly.

- VLIW: Very Long Instruction Word



Instructions come in long “bundles”, often 3 at a time. Cannot have dependencies; may have to fill with “nops”. Allows compiler to exploit inherit parallelism in code (most modern CPUs do this in hardware instead, VLIW puts this complexity in software).

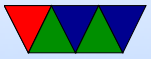


# CISC/RISC/VLIW Examples

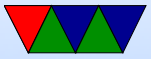
- MIPS is RISC: roughly only 40 integer instructions , (more if you include FP)

- x86 is CISC: hundreds of complicated instructions, including ones that access memory, auto-increment registers, have complex shift/add address modes

- Hybrid: ARM or Power started out RISC but have accumulated more complicated instructions over time

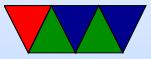


- x86, while CISC externally, internally decodes to a RISC-like code before executing



# How a Program is Loaded

- Kernel Boots
- `init` started
- `init` calls `fork()`
- child calls `exec()`
- Kernel checks if valid ELF. Passes to loader
- Loader loads it. Clears out BSS. Sets up stack. Jumps



to entry address (specified by executable)

- Program runs until complete.
- Parent process returned to if waiting. Otherwise, init.