# ECE 471 – Embedded Systems
# Lecture 18

Vince Weaver

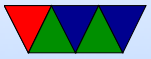
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

21 October 2019

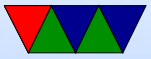
# Announcements

- HW#6 was posted, don't forget
- Info on project coming soon
- the NASA class (ECE598) for next semesters looks like it might be on again
- Midterms not quite graded yet



# HW#5 Review – Coding

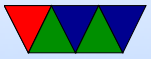
- Coding: Seemed to go OK



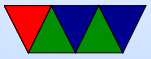
# HW#5 Review – Questions

- Raspberry Pi boot odd: GPU does it. Why? Originally the chip was designed to be mostly GPU.
- Fat32: gave lots of good reasons for Fat32, but the reason boot partitions often use it is it's simple enough to be read by firmware at extreme early boot. Q wasn't why FAT32 vs FAT16
- Program that loads kernel and jumps to it is called the bootloader

  Not start.elf. Not an init script. Not the firmware.



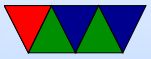
- Skip i2c – those addresses are reserved.
  For various things, not just “future purposes”
  what happens if you have a device living at addr0?



# HW#5 Review – Linux

- wc, diff, piping
- You may have seen this all before in ECE331
- diff – used when making patches, also git diff

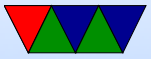
  Ask for wc -l which just shows lines. Can also show words, chars



# Real Time Constraints

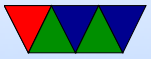
What are real time constraints?

- Time deadlines that hardware needs to respond in.
- Goal not performance, but response time



# Types of Real Time Constraints

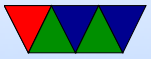
- Hard – miss deadline, total failure (people die?)
  Antilock brakes?
- Firm – result no longer useful after deadline missed
  lost frames in video, missed frames in video game
- Soft – results gradually less useful as deadline passes.
  Caps lock LED coming on?



# Constraints depend on the Application

Can almost always come up with a scenario where a soft constraint could become hard.

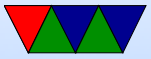
For example: Unlocking a car door taking an extra second? Not hard real-time, except maybe if your car is about to crash and you need to escape quickly.



# What can cause problems with real-time?

Sources of "Jitter"

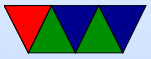
- Interrupts. Taking too long to run; being disabled (cli)
- Unpredictable nature of modern CPUs. Caches, branch-predictors, etc.
- Operating system. Scheduler. Context-switching.
- Dynamic memory allocation, garbage collection.
- Slow/unpredictable hardware (hard disks, network access)



# Uses of Real Time

Who uses realtime?

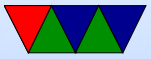
- Timing critical situations. Cars, medical equipment, space probes, etc.
- Industrial automation. SCADA. Stuxnet.
- Musicians, important to have low-latency when recording
- High-speed trading



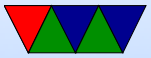
# Do a video game keyboard latency example

See Dan Luu's Paper "Computer Latency: 1977-2017"
`https://danluu.com/input-lag/`

- 1977 computers can have less latency to getting keypress on screen than fastest 2010s computers
- Having a fast processor only helps so much
- Slow hardware (keyboards, LCD displays), layers of abstraction in the way



- Apple II (1977) 30ms, modern machines 60-100+ms

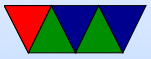


# Worst Case Behavior – Hardware

- Easier on older and simple hardware
- Old chips like 6502 – fixed clock, each instruction takes an exact number of cycles. Deterministic. With interrupts disabled you can perfectly predict how long code will take.

  Steve Wozniak famously wrote disk firmware on 6502 that more or less cycle-accurate bit-banged stepper motors.

  Also video games, racing the beam.



- Modern hardware more complex:
  - Memory accesses unpredictable with caches – may take 2 cycles or 1000 cycles
  - Interrupts can take unknown amount of time
  - Branch prediction
  - Power-save may change clock frequency
  - Even in manuals instructions can take a range of cycles