# ECE 471 – Embedded Systems
# Lecture 15

Vince Weaver

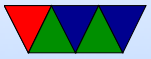
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

29 October 2013

# Announcements

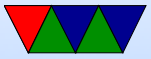
- Hopefully you are nearly done with HW#3
- HW#4 and Project will be posted soon



# Other Comments

- Static Power Consumption of i2c:
  i2c draws power through pull-up resistors when signal is low.
  Some people optimize power by having lots of 1s, use high addresses in EPROM?
  P=IV, V=IR, I=V/R, P=V*V/R, 5k and 5V means 5mW

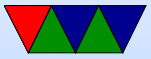
- Low-level Linux error-printing
  Calls like `open()` will just return -1 on error, not very



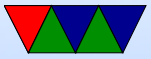
evocative.
Linux will also set the “errno” value. Include the errno.h and string.h header files, then after a failing system call you can do something like

```
printf("Error: %s\n",strerror(errno));
```

This will give you better error messages, but still limited to one of the 100 or so the kernel supports.



# Midterm Return

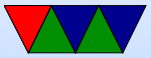


# Question 1 – Embedded Systems

- Three aspects of what makes an embedded system are:

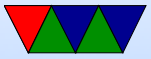
  1. Fixed Purpose
  2. Constrained Resources
  3. Real Time Response

  As long as a good argument was made I accepted most answers. The only one everyone agreed was embedded was the Traffic Light controller.



# Question 2 – Operating Systems

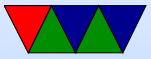
- a: Easiest answer was an O/S provides abstraction.
- b: Easiest answer is overhead (timing, resources, determinism)
- c: Most agreed this system was too low-powered to have or need an O/S
- d: Most agreed this system was powerful and complex enough that an O/S would reduce development time



# Question 3 – ARM Assembly Language

- a: The code reads the temperature from a sensor, compares the result to 100, sets a GPIO output based on the comparison, then loops.

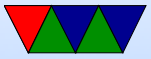
- b: The major problem with the code comments was being high-level enough. You need to say *why* something is being done, not just re-iterate what the assembly instruction does.
  The first two lines had the most problems; it's the file-descriptor being saved, not the address of the filename.



# Question 4 – Code Density

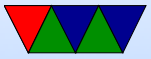
- This question was the one that tripped up the most people.

- a: The question asked for a change going from ARM32 to THUMB (not THUMB2) that improved density.
  A lot of people said conditional execution, but that was an original ARM32 feature dropped in THUMB, not vice-versa.
  Things that were changes: loss off access to top registers, no conditional execution, two (rather than 3) arguments



to opcode, smaller constants, fewer addressing modes.

- b: Things that can cause worse code density on THUMB. Conditional execution is a good argument. Code needing 3-arguments, complex addressing, or opcodes with 3 instructions are plausible too. Large constants is complex — arm32 is bad at large constants so THUMB is not always necessarily worse.

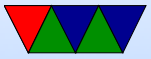


# Question 5 – Real-time Systems

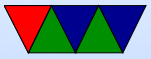
- a: Things that can cause overhead include (but are not limited to): branch predictor, caches, out-of-order execution, interrupts, operating system overhead.

- b: Most people agreed that lighting an indicator light on a car dashboard is probably soft-realtime. Realtime is about delay, and even a few seconds delay won't matter.

- c: Most people agreed that shutting down an overheating nuclear reactor is hard-realtime. Timing is critical and

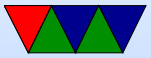


serious consequences will happen if deadlines are missed.

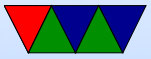


# USB Bus

- USB 1.0 – 1996 – 1.5Mbit/s (keyboard, etc), 12Mbit/s (disk)
- USB 1.1 –
- USB 2.0 – 2000 – 470MBit/s
- USB 3.0 – 2008 – 5GBit/s
- 3-5m



- 4 pins. 5V, GND, D+, D-. Differential signalling. More resistant to noise.
- Unit load, 100ma. Can negotiate up to 500ma (more USB 3.0)
- Up to 127 devices (by using hubs)
- Enumeration



# USB Protocol

- Each device has endpoint
- isochronous – guaranteed data rate but with some potential data loss (video)
- interrupt – low-latency, like keyboards
- bulk – disk access
- Device classes – HID, audio, etc