ECE 471 – Embedded Systems
Lecture 4

Vince Weaver

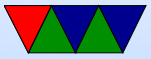
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

9 September 2020

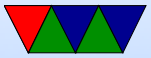
Announcements

- Any questions on HW#1?
  Will go over answers next class.
- HW#2 will be posted today
- Let me know if you don't have a Raspberry Pi yet



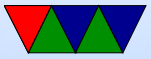
Current Events

- ARM released the Cortex-R82 processor
  64-bits, with MMU so can run Linux
- 5 times faster than previous Cortex-R8
- What would it be used for? Cortex-R widely used in storage devices. Can offload compute to storage?



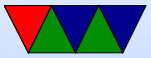
Raspberry Pi Models

- Model Names originally from BBC Micro
- All have more or less same SoC. VideoCore IV GPU runs show (VideoCore VI on pi4)
- First released in 2012

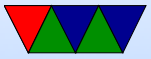


BCM2835/BCM2708 – ARM1176

- Model B – 700MHz ARM1176, 512MB RAM, SD, USB hub+USB Ethernet
- Model B+ – like B but micro-SD, composite video-out inside of audio jack, 4 USB ports, longer GPIO header, re-arranged outputs, more mounting holes, fewer LEDs, lower power
- Model A / Model A+ – less RAM (256MB/512MB), no Ethernet, no USB hub, cheaper, less power
- Zero – 1GHz, 512MB, smaller, cheaper, $5

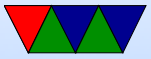


- Zero W – 1GHz, has wireless, $10
- Compute Node – like B but on SO-DIMM backplane, eMMC



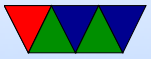
BCM2836/BCM2709 – ARM Cortex A7

- Model2 B – like B+ but with 1GB RAM, 900MHz Quad-core Cortex A7



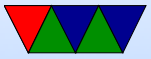
BCM2837/BCM2710 – ARM Cortex A53

- Model3 B – 64-bit, 1.2GHz Cortex A53, wireless Ethernet, bluetooth
- Model2 B (v1.2) – like Model 2 but with the Cortex A53
- Compute 3
- Model3 B+ – better thermal, faster Ethernet (1GB but maxes at 300MB), power over Ethernet header. Still only 1GB (cost?)
- Model3 A+



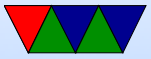
BCM2711 – ARM Cortex A72

- Videocore VI at 500MHz
- 1, 2, or 4GB RAM
- USB3
- PCIe if you de-solder USB chip
- Real gigabit Ethernet
- GPIO header has more i2c/spi etc options
- Model B



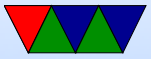
Homework #2 – Background

- It's mostly about getting everyone up to speed on the Pi as not everyone has used one before
  - Many ways to set up your Pi for use, everyone has a different preference
  - Be sure to change your password from default
- Also a small C coding assignment, and some short-answer questions.



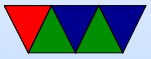
Homework #2 – Transferring Assignment

- It's a .tar.gz file. What is that?
- Sort of the Linux equivalent of a zip file
- tar = tape archive (ancient history) that runs lots of files together
  gz = gzip, which compresses it (makes it smaller)
- you may see other (Z, bz2, xz). What are the differences?
  Mostly in compressed size vs compress/uncompress resources
  gzip good enough for what we are doing.



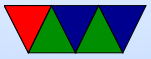
Homework #2 – Editing C

- Take some existing C code and modify it.
- Can use the editor of your choice. Many on Linux.
  - "nano" is easy
  - "vim" if you are serious about Linux.
  - "emacs" – I've known some emacs wizards
  - Also various graphical ones
  - if you want you can even code it up on your desktop/laptop, but you probably want to copy it over to test before submitting.



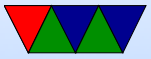
Why C?

- Portability (sort of) (i.e. how big is an `int`)
- Higher than assembly (barely)
  Pearce: "all the power of assembly with all the ease-of-use of assembly"
- Why over Java or C++?
  They can hide what is actually going on. C statements map more or less directly to assembly. With things like operator overload, and exceptions, garbage collection, extra layers are added. This can matter for both size,



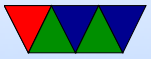
speed, determinism, and real time.
Might be restricted to a subset of C++

- Why over python?
  Mostly speed. (although you can JIT) Also if accessing low level hardware, in general you are calling libraries from python that are written in C anyway.
- What about Rust and Go? Don't overlook momentum of an old platform, sample code, libraries, etc.



Downsides of C?

- Undefined behavior. Compiler is allowed to do anything it wants (including dropping code) if it encounters something undefined by the standard. This can be something as simple as just overflowing an integer or shifting by more than 32.
- "Enough rope to shoot yourself in the foot". C gives a lot of power, especially with pointers. It assumes you know what you are doing though. With great power comes great responsibility.

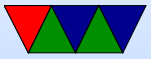


Downsides of C – Security

- Biggest issue is memory handling (lack thereof)
- Buffer overflows

```
int a[5];
a[0]=1;                 // fine
a[10000000]=1;          // obviously bad
a[5]=1;                 // subtly bad
```

- How can that go wrong? Crash? Corrupt Memory? Wrong results? Total system compromise?



- Overwriting stack can be bad, as return address there
- Especially if user input going into the variable