ECE 531 – Advanced Operating Systems
Lecture 22

Vince Weaver

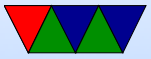
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

29 October 2025

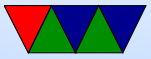
Announcements

- Don't forget HW#6



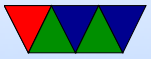
Returned and went over Midterm

Average was 75%

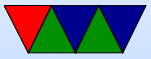


HW Notes: Debugging

- Debugging an embedded system is difficult
- printk()? Flashing the ACT LED?
- Can you do JTAG on pi?
- In theory it does. You have to activate the proper pins (in config.txt?) and make a custom cable
- Useful link `https://sysprogs.com/VisualKernel/tutorials/raspberry/jtagsetup/`
- Might make an interesting project

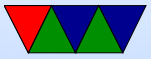


Virtual Memory Wrapup

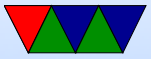


Aside: what if you have unused bits in address?

- Virtual address is so huge, in some cases architectures never use the top bits
- Can these be used by the user? Things like smart pointers or to stash a few bits?
- People do use them, causes problems later. See M68k/MacOS (24/32), IBM 390 (31/32), ARM1 (26/32)
- AMD64 canonical addresses to avoid this (top bits have

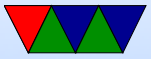


to be all zeros or all ones)



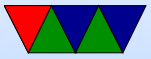
Recent Architectures have added official support to this

- By using the official interface and explicitly using the top bits in theory later processors will be aware of this and work around it
  - ARM64 Memory Tagging Extension (MTE), Top Byte Ignore (TBI)
  - AMD64 Upper Address Ignore (UAI)
  - Intel Linear Address Masking (LAM)



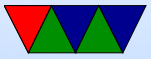
Large Pages

- Another way to avoid problems with 64-bit address space
- Larger page size (64kB? 1MB? 2MB? 2GB?)
- Upsides: Fewer TLB entries needed to map large data structures
- Downsides: Less granularity. Potentially waste space



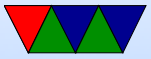
Large Pages by Architecture

- x86: 4k (2MB/4MB if PAE)
- x86_64: 4k/2MB/1GB
- ARM64: 4k, 16k, 64k, 2MB, 32MB, 512MB, 1GB
  big push to use 16k by default by ARM
- ARM1176: 4k, 64k, 1MB, 16MB



Transparent Huge Pages

- Compromise: multiple page sizes.
  Complicate O/S and hardware. OS have to find free blocks of contiguous memory when allocating large page.
- Transparent usage? Transparent Huge Pages? Alternative to making people using special interfaces to allocate.



Having Larger Physical than Virtual Address Space

- 32-bit processors cannot address more than 4GB
  x86 hit this problem a while ago, ARM more recently
- Real solution is to move to 64-bit
- As a hack, can include extra bits in page tables, address more memory (though still limited to 4GB per-process)
- Linus Torvalds hates this.
- Hit an upper limit around 16-32GB because entire low 4GB of kernel addressable memory fills with page tables