ECE 598 – Advanced Operating Systems
Lecture 13

Vince Weaver

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

vincent.weaver@maine.edu

1 March 2016

Announcements

- Homework #6
  Short. Due after midterm. Be sure to look at memory problem.
  Warnings on why its good to comment your code.

- Raspberry Pi 3 is out

HW#5 Review

- Shell to userspace
- Add a time system call

  Writing to a user-supplied pointer. Dangerous? copy_to_user()?
- nonblocking getchar
- Why run in userspace?
- Changing back to kernel mode
- What is an ABI

Midterm Review

- Closed book/notes/computer but can bring one piece of notebook paper (front only) with notes on it
- Questions will be similar to those from homeworks
- Topics
  - Benefits of an OS / Downsides of an OS
  - Serial communication: why are we using it? What does 9600 7E1 mean? How does hardware and software flow control work?
  - Boot process

- High level, how the GPIO interface works
- Interrupts: how they switch processor mode, why FIQ is different from IRQ mode. How to switch back from userspace.
- System calls
- ABI
- Memory allocation: first vs best fit

Advanced Memory Handling

Security/Safety

- Want a way to mark memory regions as user only, or read-only, or no-execute
- Some processors provide “segments” for this
- Some ARM processors have a “Memory Protection Unit” (MPU)
- Most modern processors have an MMU (memory management unit) to do full virtual memory

Using More Memory than Physically Available

- How can you have a program that accesses more RAM than available in physical memory?

- Swapping, as discussed before

- Can manually swap out small parts of a program, this technique is called overlays.

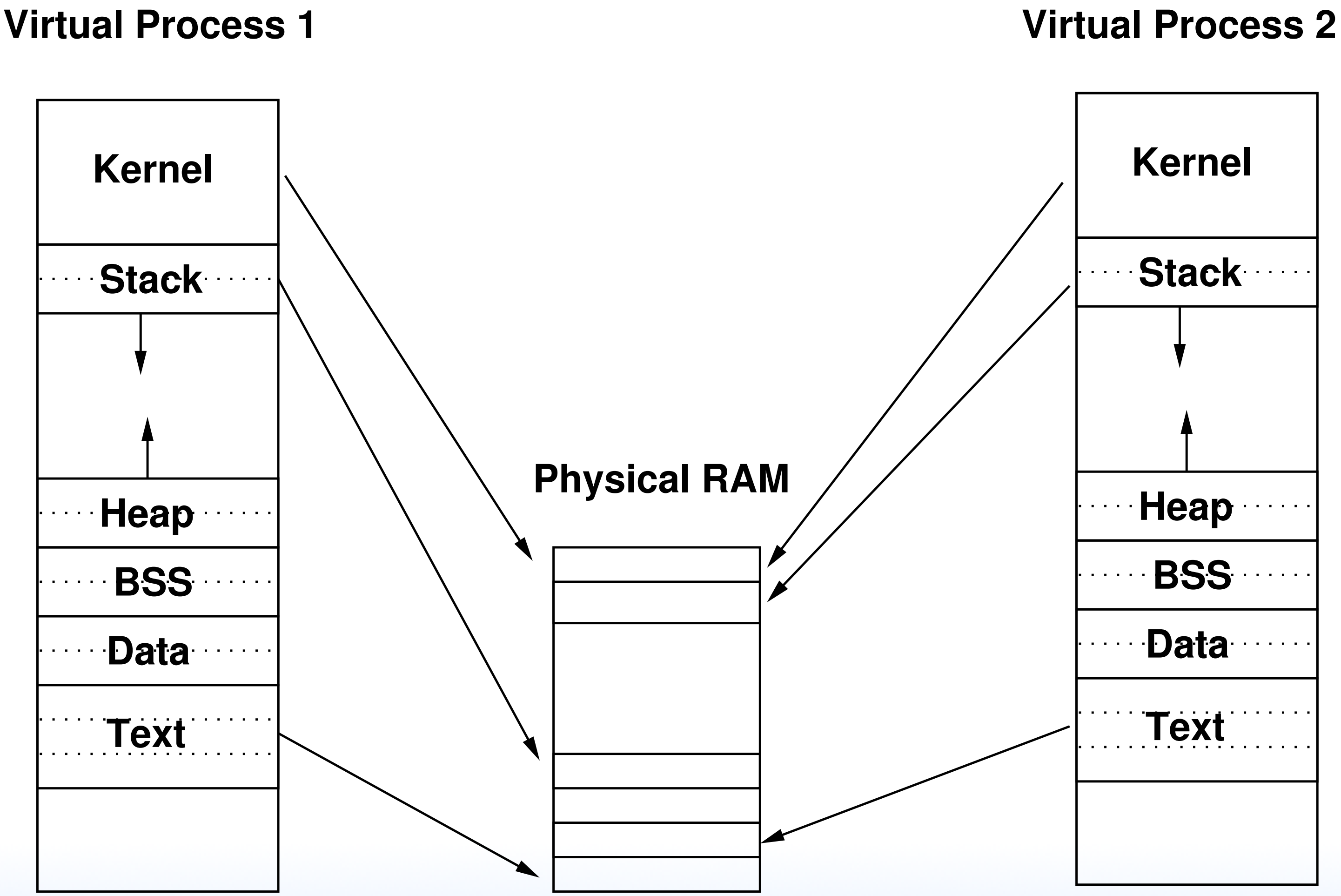
- Split program in parts. Only load the part currently running at any given time.

- Can we have hardware do this automatically? This is part of the idea of virtual memory.

Virtual Memory

- Original purpose was to give the illusion of more main memory than available, with disk as backing store.
- Give each process own linear view of memory.
- Demand paging (no swapping out whole processes).
- Execution of processes only partly in memory, effectively a cache.
- Memory protection
- Reduces fragmentation

Diagram



Virtual Process 1
Virtual Process 2
Kernel
Stack
Heap
BSS
Data
Text
Physical RAM
Kernel
Stack
Heap
BSS
Data
Text

Memory Management Unit

Can run without MMU. There's even MMU-less Linux.

How do you keep processes separate? Very carefully...

Page Table

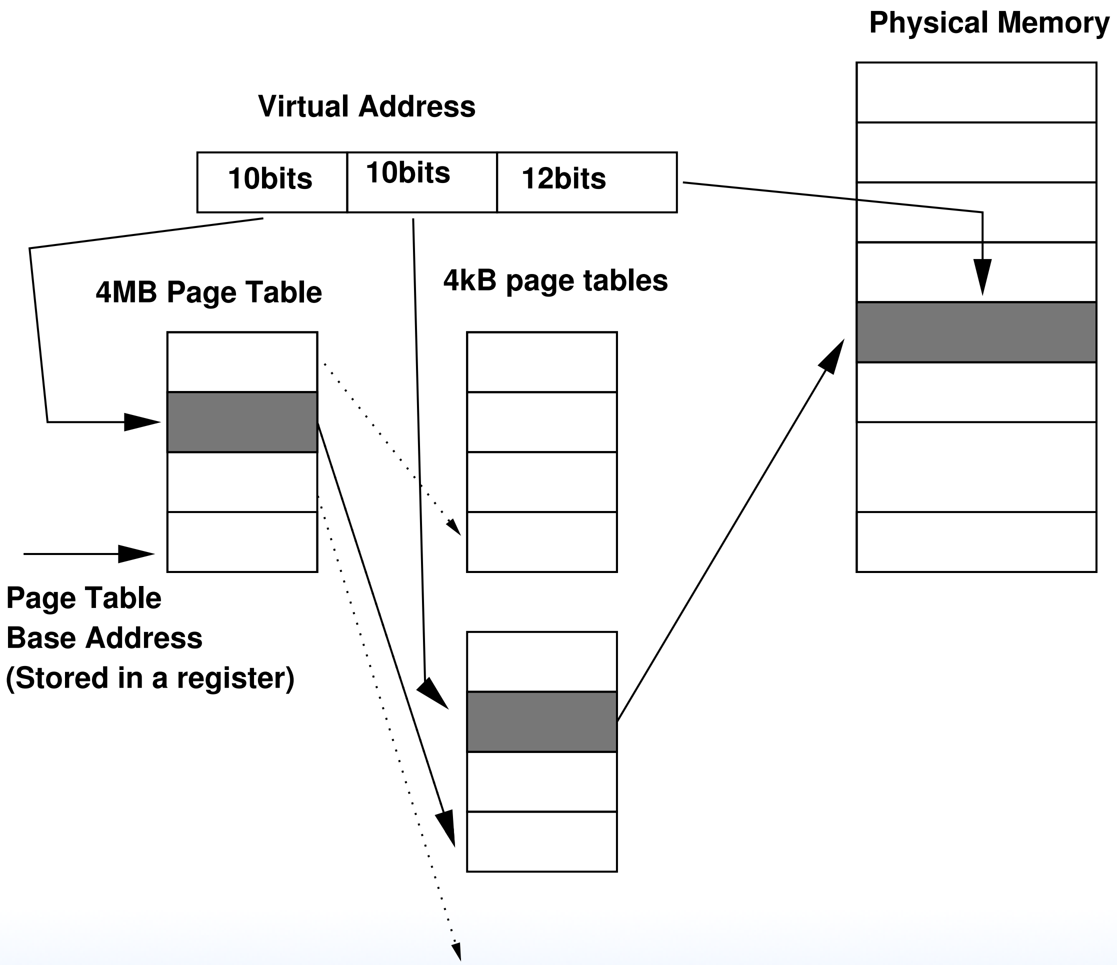
- Collection of Page Table Entries (PTE)
- Some common components: ID of owner, Virtual Page Number, valid bit, location of page (memory, disk, etc), protection info (read only, etc), page is dirty, age (how recent updated, for LRU)

Hierarchical Page Tables

- With 4GB memory and 4kb pages, you have 1 Million pages per process. If each has 4-byte PTE then 4MB of page tables per-process. Too big.

- It is likely each process does not use all 4GB at once. (sparse) So put page tables in swappable virtual memory themselves!
  4MB page table is 1024 pages which can be mapped in 1 4KB page.

Hierarchical Page Table Diagram



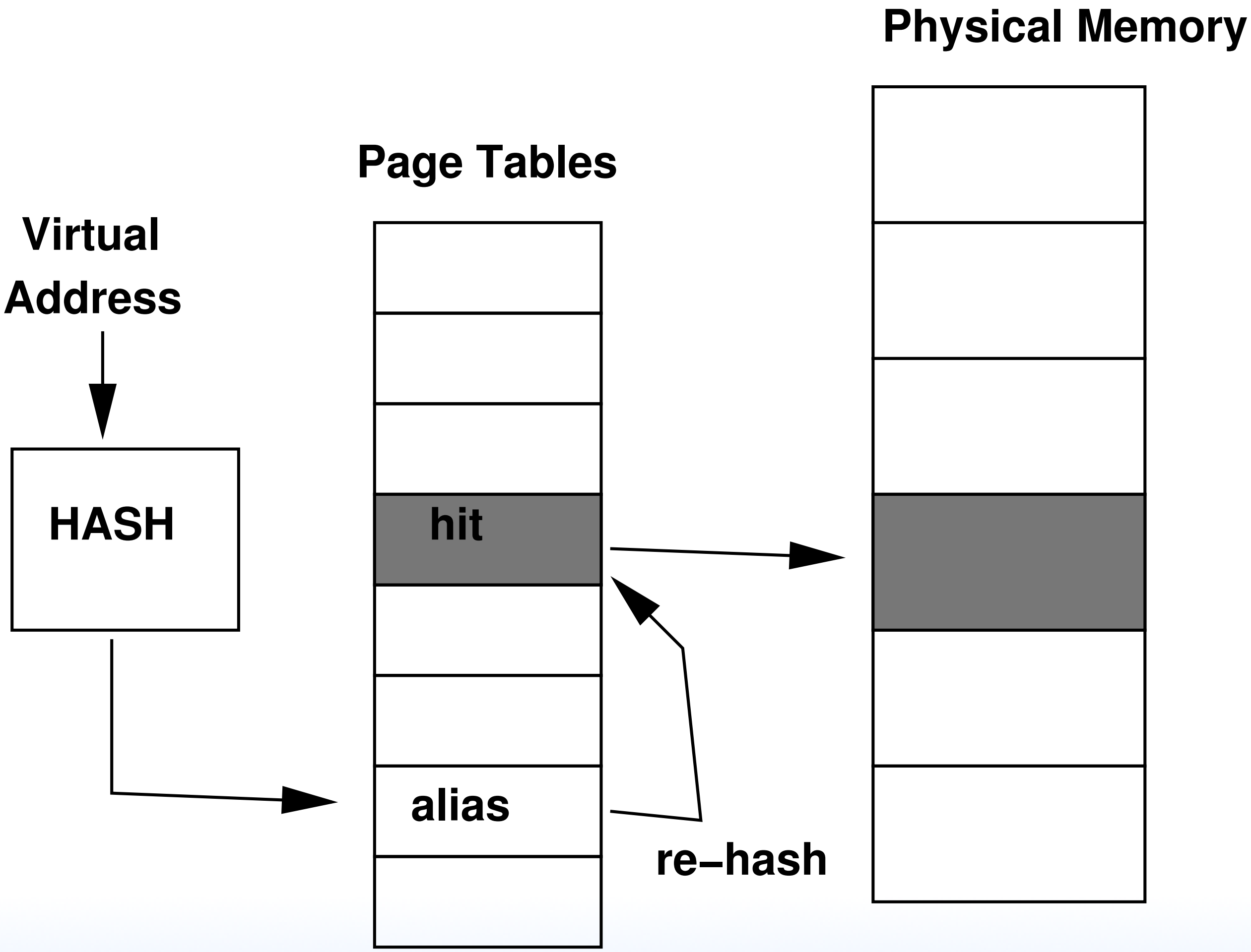
Hierarchical Page Table Diagram

- 32-bit x86 chips have hardware 2-level page tables
- ARM 2-level page tables

Inverted Page Table

- How to handle larger 64-bit address spaces?
- Can add more levels of page tables (4? 5?) but that becomes very slow
- Can use hash to find page. Better best case performance, can perform poorly if hash algorithm has lots of aliasing.

Inverted Page Table Diagram



Walking the Page Table

- Can be walked in Hardware or Software
- Hardware is more common
- Early RISC machines would do it in Software. Can be slow. Has complications: what if the page-walking code was swapped out?

TLB

- Translation Lookaside Buffer
  (Lookaside Buffer is an obsolete term meaning cache)

- Caches page tables

- Much faster than doing a page-table walk.

- Historically fully associative, recently multi-level multi-way

- TLB shootdown – when change a setting on a mapping and TLB invalidated on all other processors

Flushing the TLB

- May need to do this on context switch if doesn't store ASID or ASIDs run out.
- Sometimes called a "TLB Shootdown"
- Hurts performance as the TLB gradually refills
- Avoiding this is why the top part is mapped to kernel under Linux