ECE 571 – Advanced Microprocessor-Based Design Lecture 13

Vince Weaver

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

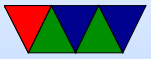
vincent.weaver@maine.edu

26 February 2013

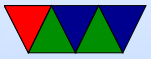
HW2 Comments

Aggregate counts

- Valgrind and gem5 do not measure events in the kernel
- ARM at disadvantage compared to other CPUs that can differentiate kernel and user space counts in hardware
- Aggregate counting doesn't use sampling.
  Sampling is used when you want a function profile; aggregate cannot provide that.



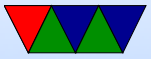
- perf can start/stop a new process; it supports setting up events and have them start once an `exec()` system call happens. Then it waits for the process to finish and can read the results.



HW2 Comments continued

Static binaries

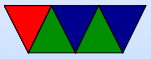
1. Ensures same libraries used when running on different machines
2. Helpful if running under simulator on different arch
3. Let you re-run later and have same libraries (even if machine has been upgraded)



HW2 Comments continued

PAPI

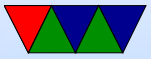
- Has overhead because it adds calls into your code
- Performance can actually end up *better*. It might shift cache lines not to alias, branch addresses to fit better in predictor, etc.
- This is what makes performance analysis difficult on modern architectures.



HW2 Comments continued

Sampled results

- Valgrind – `malloc_set_state()` is some sort of artifact, not sure why it shows up as the top function. Technically I'm a Valgrind dev. . .
- perf show result in kernel, which gprof and valgrind can't

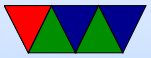


HW3 Comments

Typically when measuring memory benchmark results with varying size you can easily spot the memory hierarchy borders (L1/L2/L3).

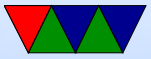
This is the theory. In practice it might be more difficult.

Why use log/log graph?



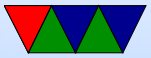
Cortex A9 Prefetch redux

- PLD – prefetch instruction
  has dedicated instruction unit
- Optional hardware prefetcher. **(Disabled on pandaboard!)**
- Can prefetch 8 data streams, detects ascending and descending with stride of up to 8 cache lines
- Keeps prefetching as long as causing hits



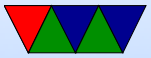
- Stops if: crosses a 4kB page boundary, changes context, a DSB (barrier) or a PLD instruction executes, or the program does not hit in the prefetched lines.

- PLD requests always take precedence



Virtual Memory

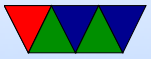
- Original purpose was to give the illusion of more main memory than available, with disk as backing store.
- Give each process own linear view of memory.
- Demand paging (no swapping out whole processes).
- Execution of processes only partly in memory, effectively a cache.



Memory Management Unit

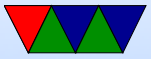
Can run without MMU. There's even MMU-less Linux.

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



Page Table

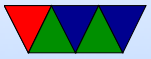
- Collection of Page Table Entries (PTE)
- Some common components: ID of owner, Virtual Page Number, valid bit, location of page (memory, disk, etc), protection info, page is dirty, age (how recent updated, for LRU)



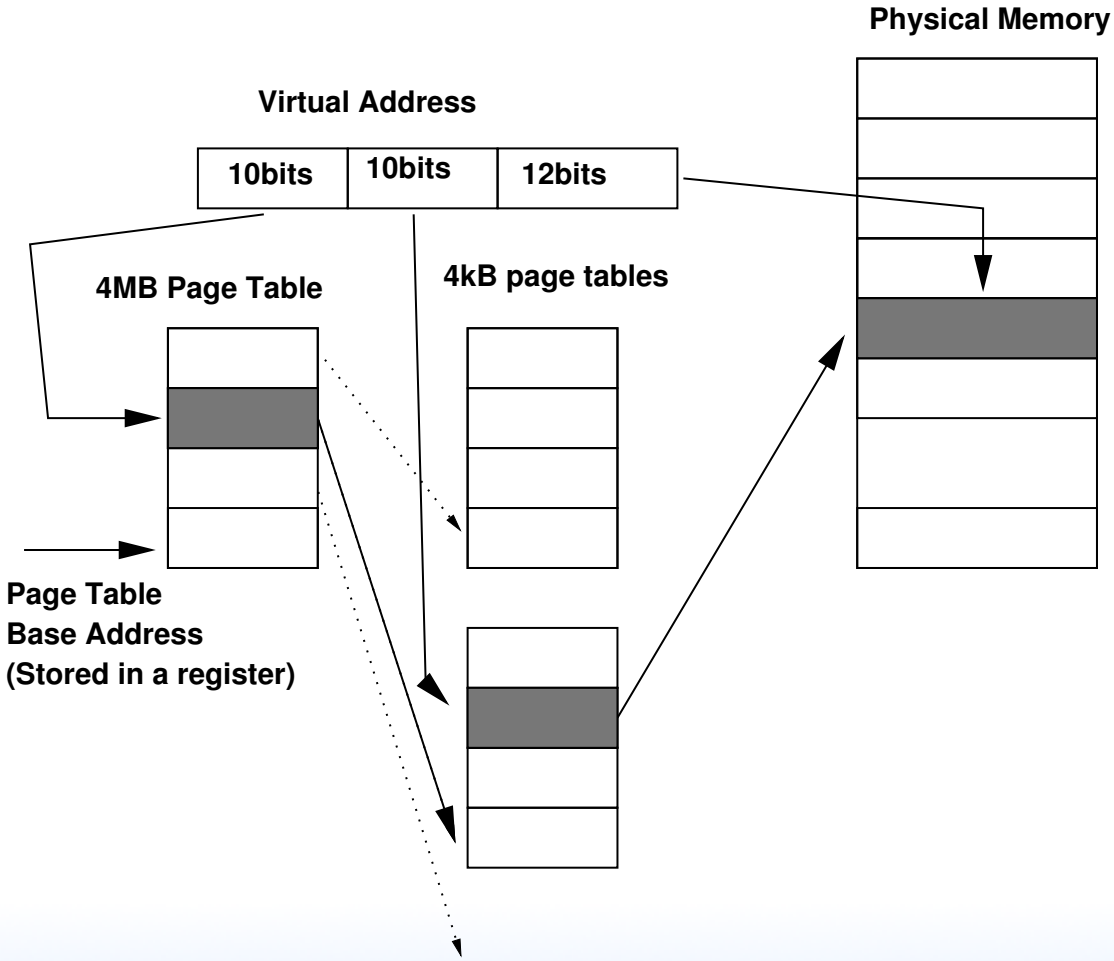
Hierarchical Page Tables

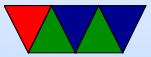
- With 4GB memory and 4kb pages, you have 1 Million pages per process. If each has 4-byte PTE then 4MB of page tables per-process. Too big.

- 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.



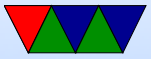
Hierarchical Page Table Diagram



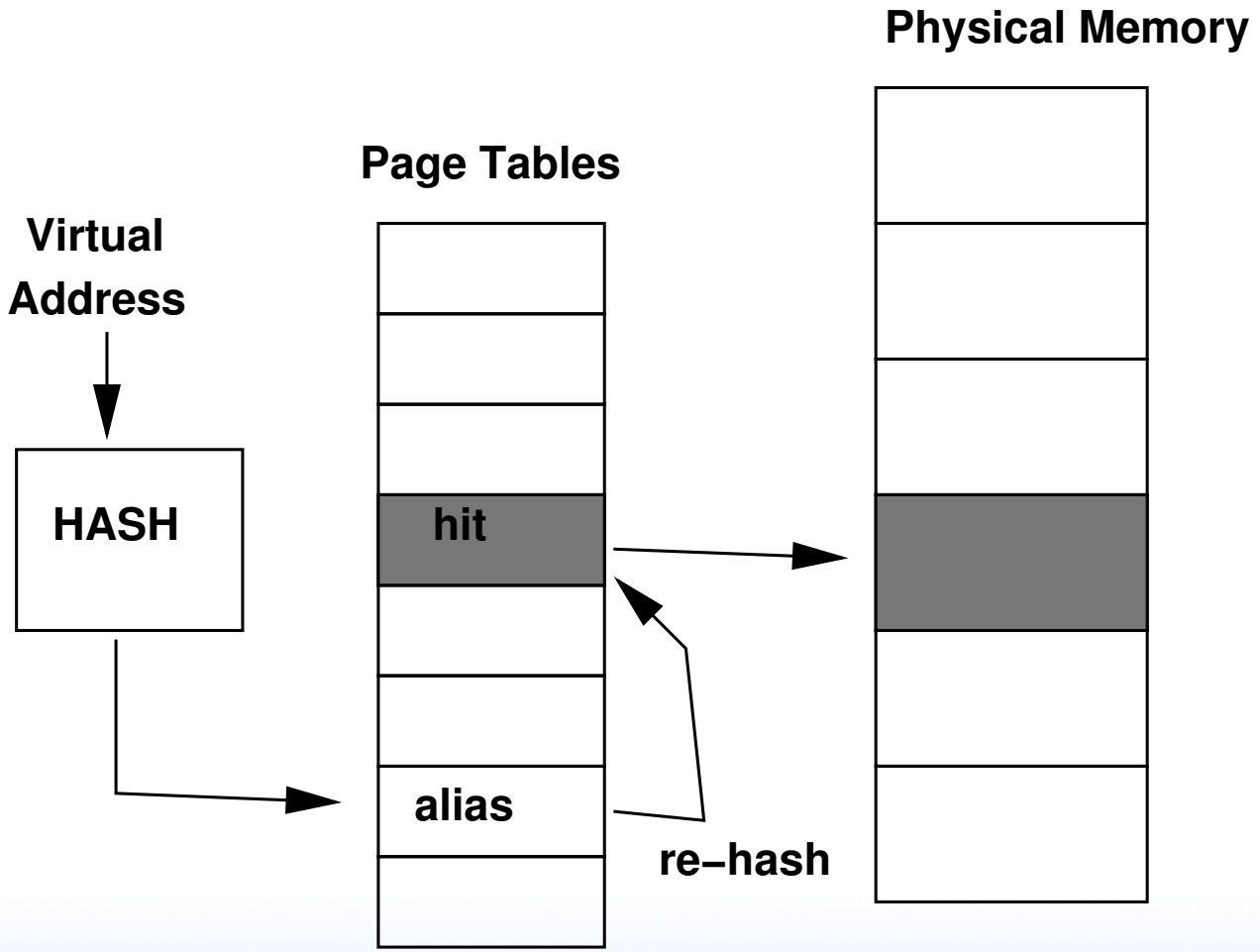


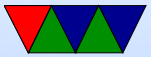
Inverted Page Table

- 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.



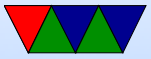
Inverted Page Table Diagram





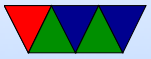
Walking the Page Table

- 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?



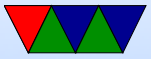
TLB

- 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



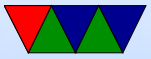
Flushing the TLB

- May need to do this on context switch if doesn't store ASID or ASIDs run out.
- Also called a "TLB Shootdown"
- Hurts performance as the TLB gradually refills

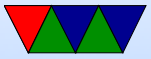


Cache Issues

- Page table Entries are cached too
- What happens if more memory can fit in the cache than can be covered by the TLB?
- If you have 128 TLB entries * 4kB you can cover 512kB
- If your cache is larger (say 1MB) then a simple walk through the cache will run out of TLB entries, so page lookups will happen (bringing page table data into cache) and so you do not get maximal usefulness from the cache

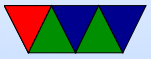


- This has happened in various chips over the years



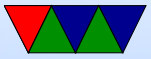
Virtual Caches

- Location in cache based on virtual address
- Faster, as no need to do TLB lookup before access
- Can have aliasing issues when processes use same virtual addresses.
  Flush cache on context switch?
- Operating system has to do more work



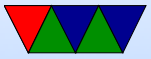
Physical Caches

- Location in cache based on physical address
- Can be slower, as need TLB lookup before accessing cache



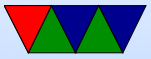
Large Pages

- Another way to avoid problems with 64-bit address space
- Larger page size (64kB? 1MB? 2MB?)
- Less granularity. Potentially waste space
- Fewer TLB entries needed to map large data structures
- Compromise: multiple page sizes.
  Complicate O/S and hardware. OS have to find free blocks of contiguous memory when allocating large page.

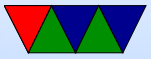


Cortex A9 MMU

- Virtual Memory System Architecture version 7 (VMSAv7)
- page table entries that support 4KB, 64KB, 1MB, and 16MB
- global and address space ID (no more TLB flush on context switch)
- instruction micro-TLB (32 or 64 fully associative)

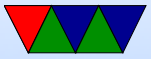


- data micro-TLB (32 fully associative)
- Unified main TLB, 2-way, 2x64 (128 total) on pandaboard
- 4 lockable entries (why want to do that?)
- Supports hardware page table walks



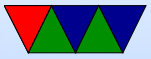
Cortex A9 MMU

- Virtual Memory System Architecture version 7 (VMSAv7)
- Addresses can be 40bits virt / 32 physical
- First check FCSE – linear translation of bottom 32MB to arbitrary block in physical memory (optional with VMSAv7)

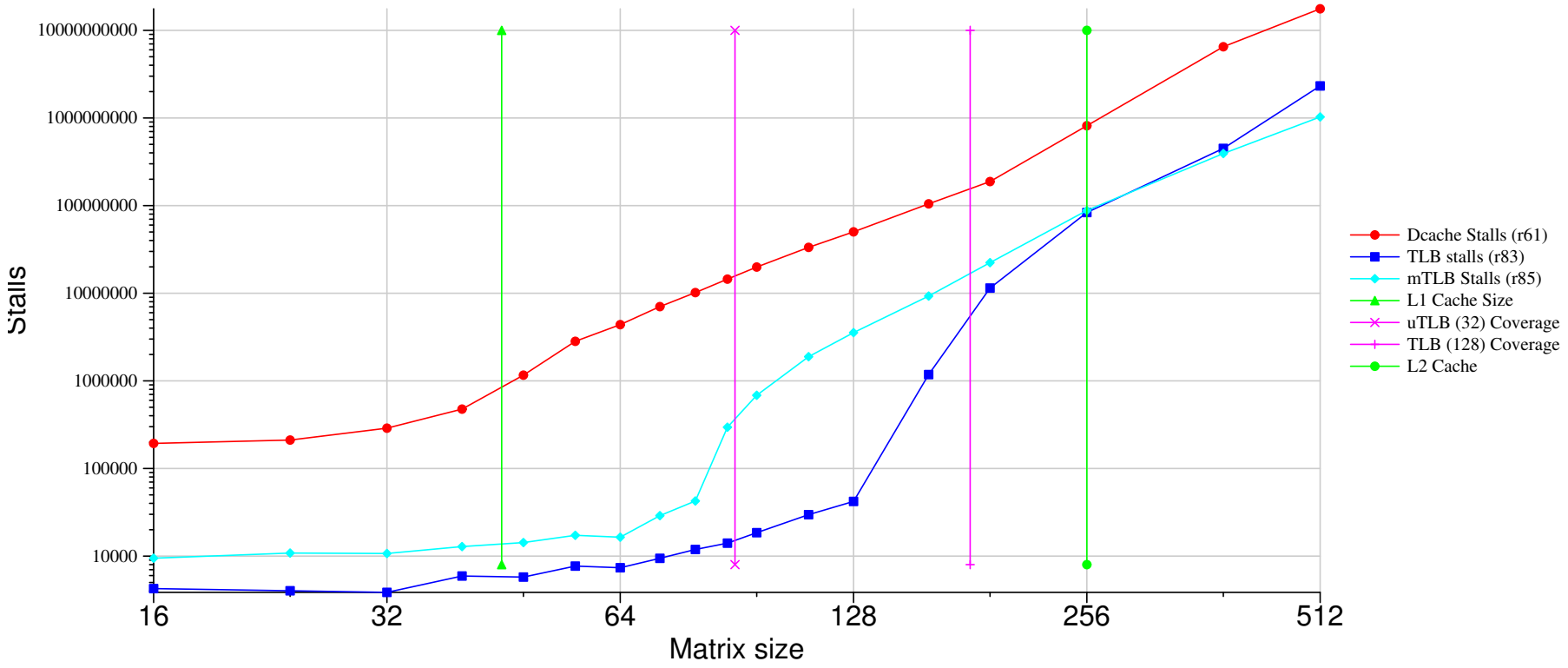


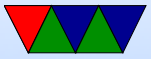
Cortex A9 TLB

- micro-TLB. 1 cycle access. needs to be flushed if ASID changes
- fully-associative lockable 4 elements plus 2-way larger. varying cycles access



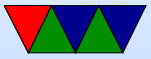
Cortex A9 TLB Measurement



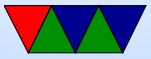


Having Larger Physical than Virtual Address Space

- 32-bit processors cannot address more than 4GB
  x86 hit this problem a while ago, ARM just now
- 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



Midterm Review