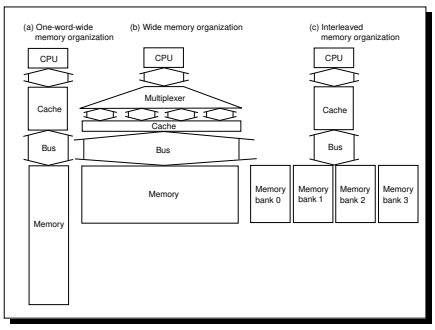
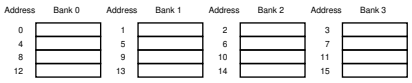
Lecture 19: Case Study— Virtual Memory, Alpha 21064 Memory Hierarchy and Performance

Professor Randy H. Katz
Computer Science 252
Spring 1996

Review: Main Memory Performance

- ***Simple***: CPU, Cache, Bus, Memory same width (32 bits)
- ***Wide***: CPU/Mux 1 word; Mux/Cache, Bus, Memory N words (Alpha: 64 bits & 256 bits)
- ***Interleaved***: CPU, Cache, Bus 1 word: Memory N Modules (4 Modules); example is ***word interleaved***

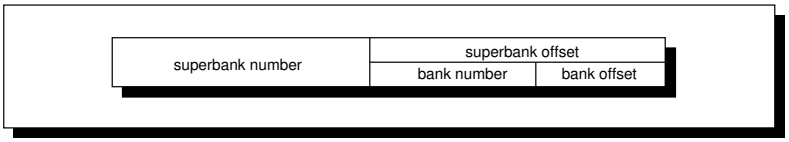




Timing model: 1 to send address, 6 access time, 1 to send data
Cache Block is 4 words
Simple M.P. **= 4 x (1+6+1) = 32**
Wide M.P. **= 1 + 6 + 1 = 8**
Interleaved M.P. **= 1 + 6 + 4x1 = 11**

Review: Independent Memory Banks

- Memory banks for independent accesses vs. faster sequential accesses
  - Multiprocessor
  - I/O
  - Miss under Miss, Non-blocking Cache
- *Superbank*: all memory active on one block transfer
- *Bank*: portion within a superbank that is word interleaved



Review: Independent Memory Banks

- How many banks?

  number banks ≥ number clocks to access word in bank
  - For sequential accesses, otherwise will return to original bank before it has next word ready
- Increasing DRAM => fewer chips => harder to have banks

Review: Fast Memory Systems—DRAM specific

- Multiple RAS accesses: several names (page mode)
  - 64 Mbit DRAM: cycle time = 100 ns, page mode = 20 ns
- New DRAMs to address gap; what will they cost, will they survive?
  - *Synchronous DRAM*: Provide a clock signal to DRAM, transfer synchronous to system clock
  - *RAMBUS*: startup company; reinvent DRAM interface
    - » Each Chip a module vs. slice of memory
    - » Short bus between CPU and chips
    - » does own refresh
    - » variable amount of data returned
    - » 1 byte / 2 ns (500 MB/s per chip)
- Niche memory or main memory?
  - e.g., Video RAM for frame buffers, DRAM + fast serial output

Main Memory Review

- Wider Memory
- Interleaved Memory: sequential or independent accesses
- Avoiding bank conflicts: SW & HW
- DRAM specific optimizations
  - Page mode & Specialty DRAM
- Next: Virtual Memory Organization

Virtual Memory

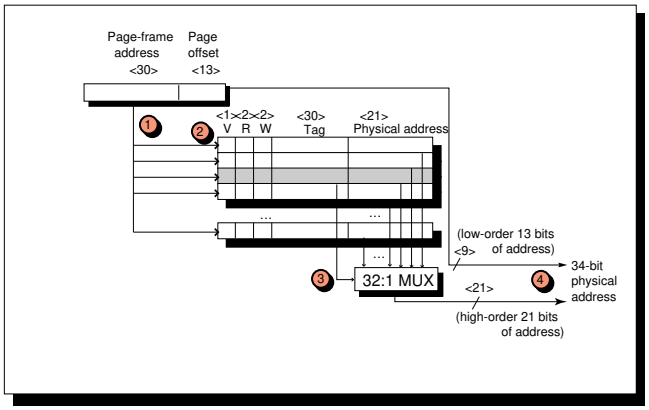
- Virtual address (2^{32}, 2^{64}) to Physical Address mapping (2^{28})
- Virtual memory terms of cache terms:
  - Cache block?
  - Cache Miss?
- How is virtual memory different from caches?
  - What Controls Replacement
  - Size (transfer unit, mapping mechanisms)
  - Lower level use

Virtual Memory

- 4Qs for VM?
  - Q1: Where can a block be placed in the upper level?
    Fully Associative, Set Associative, Direct Mapped
  - Q2: How is a block found if it is in the upper level?
    Tag/Block
  - Q3: Which block should be replaced on a miss?
    Random, LRU
  - Q4: What happens on a write?
    Write Back or Write Through (with Write Buffer)

Fast Translation: Translation Buffer

- Cache of translated addresses
- Alpha 21064 TLB: 32 entry fully associative



RHK.S96 9

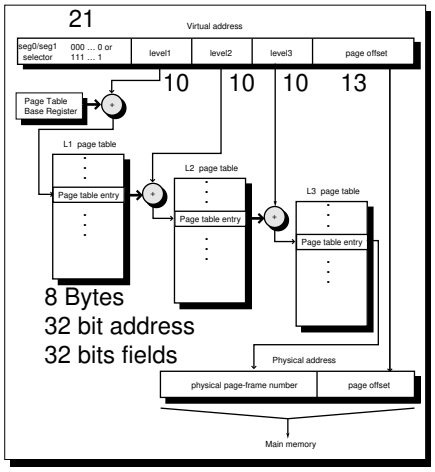
Selecting a Page Size

- Reasons for larger page size
  - Page table size is inversely proportional to the page size; therefore memory saved
  - Fast cache hit time easy when cache ≤ page size (VA caches); bigger page makes it feasible as cache size grows
  - Transferring larger pages to or from secondary storage, possibly over a network, is more efficient
  - Number of TLB entries are restricted by clock cycle time, so a larger page size maps more memory, thereby reducing TLB misses
- Reasons for a smaller page size
  - Fragmentation: don't waste storage; data must be contiguous within page
  - Quicker process start for small processes(??)
- Hybrid solution: multiple page sizes
  - Alpha: 8KB, 16KB, 32 KB, 64 KB pages (43, 47, 51, 55 virt addr bits)

RHK.S96 10

Alpha VM Mapping

- "64-bit" address divided into 3 segments
  - seg0 (bit 63=0) user code/heap
  - seg1 (bit 63 = 1, 62 = 1) user stack
  - kseg (bit 63 = 1, 62 = 0) kernel segment for OS
- Three level page table, each one page
  - Alpha only 43 unique bits of VA
  - (future min page size up to 64KB => 55 bits of VA)
- PTE bits; valid, kernel & user read & write enable (No reference, use, or dirty bit)
  - What do you do?



RHK.S96 11

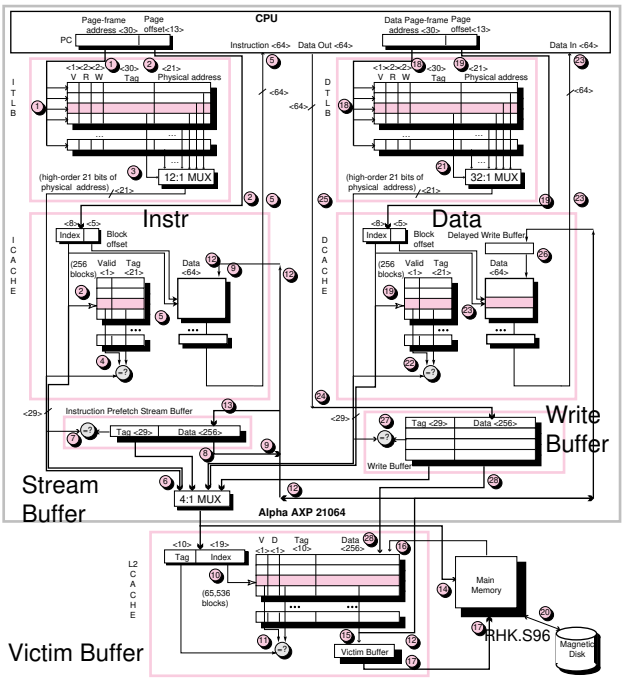
Cross Cutting Issues

- Superscalar CPU & Number Cache Ports
- Speculative Execution and non-faulting option on memory
- Parallel Execution vs. Cache locality
  - Want far separation to find independent operations vs. want reuse of data accesses to avoid misses
- I/O and consistency of data between cache and memory
  - Caches => multiple copies of data
  - Consistency by HW or by SW?
  - Where connect I/O to computer?

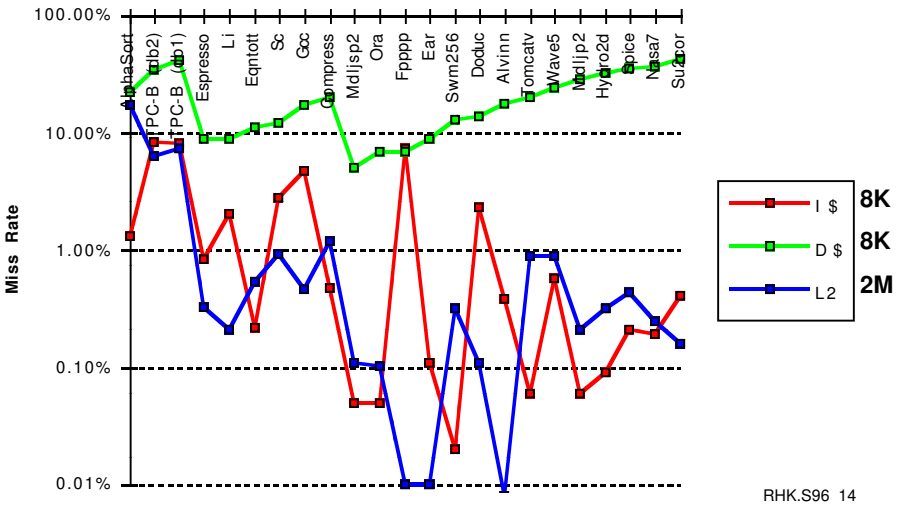
RHK.S96 12

Alpha 21064

- **Separate Instr & Data TLB & Caches**
- **TLBs fully associative**
- **TLB updates in SW (“Priv Arch Libr”)**
- **Caches 8KB direct mapped**
- **Critical 8 bytes first**
- **Prefetch instr. stream buffer**
- **2 MB L2 cache, direct mapped (off-chip)**
- **256 bit path to main memory, 4 x 64-bit modules**



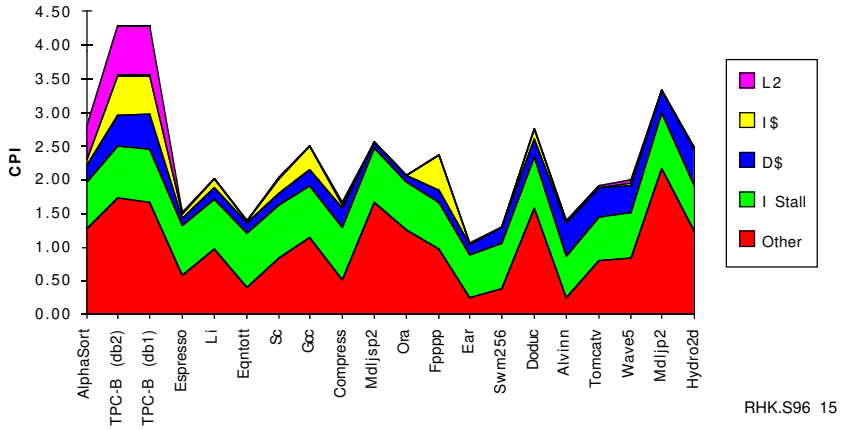
Alpha Memory Performance: Miss Rates



RHK.S96 14

Alpha CPI Components

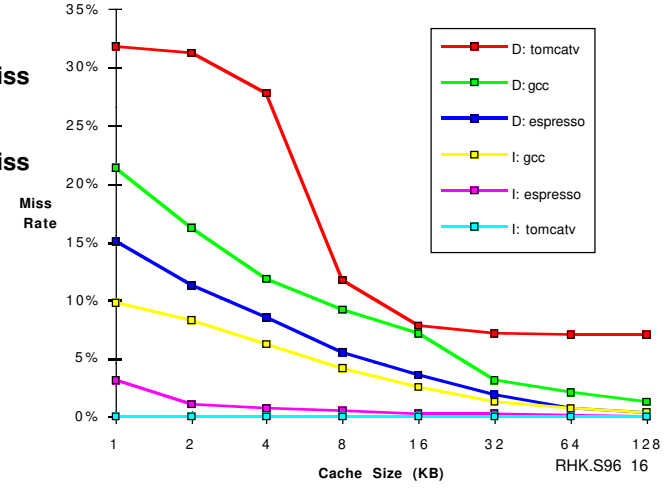
- **Instruction stall: branch mispredict;**
  Other: compute + reg conflicts, structural conflicts



RHK.S96 15

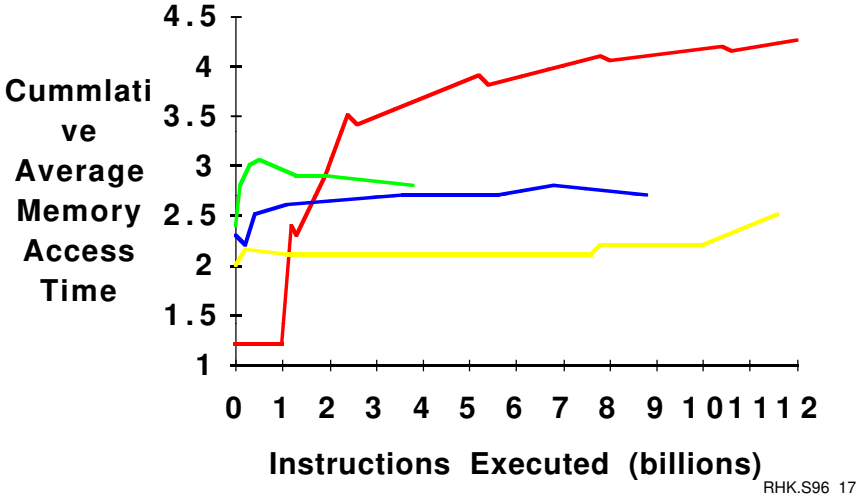
Pitfall: Predicting Cache Performance from Different Prgrm (ISA, compiler, ...)

- **4KB Data cache miss rate 8%,12%, or 28%?**
- **1KB Instr cache miss rate 0%,3%, or 10%?**
- **Alpha vs. MIPS for 8KB Data: 17% vs. 10%**



RHK.S96 16

Pitfall: Simulating Too Small an Address Trace



RHK.S96 17