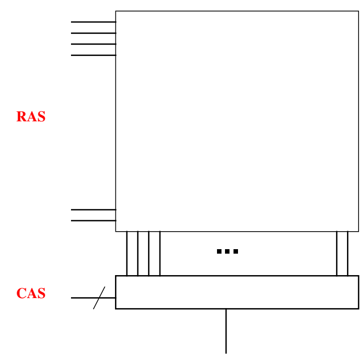
DRAMs



Memory (DRAM)

Increasing memory bandwidth

- Wider memory
  - wider bus, MUX to CPU
- Interleaved memory
  - 1 access, multiple data
  - requires large memory size or smaller capacity chips
  - good for sequential accesses
- Independent memory banks
  - separate address lines
  - good with nonblocking caches
- Avoiding bank conflicts
  - loop interchange
  - memory bank conflict resolution
  - prime number of banks

Virtual Memory (Review)

Virtual memory

Page frame

Page

Page fault

Segment

Address translation

Page table

Page table entry

LRU replacement page policy

FA page placement

TLB

Translation Look-aside Buffer

- cache for most recent virtual-to-physical mappings
- fully associative or high set-associative

Page Size Today

Multiple page sizes (A S X)

- rationale
  - more memory mapped by a TLB entry
    - fewer TLB misses
    - fewer TLB entries
  - larger virtually indexed cache
  - more efficient transfer from disk
  - smaller page table
  - good match with newer applications (multimedia & object-oriented)
  - memories are large enough that internal fragmentation is not a big issue
- uses in V9
  - 8KB & 64KB for programs
  - 512KB & 4MB for mapped files, mapped frame buffers, etc.
- software-enforced restrictions on page allocation in V9
  - page alignment wrt page size
  - pages of different sizes cannot overlap

V9 Pages & Page Frames

63 … 13	12 … 0
8KB virtual page #	page offset

39 … 13	12 … 0
8KB physical page frame #	PF offset

63 … 16	15 … 0
64KB virtual page #	page offset

39 … 16	15 … 0
64KB physical PF #	PF offset

63 … 19	18 … 0
512KB virtual page #	page offset

39 … 19	18 … 0
512KB physical PF #	page frame offset

63 … 22	21 … 0
4MB virtual page #	page offset

39 … 22	21 … 0
4MB physical PF #	page frame offset

V9 Memory Management

Translation Look-aside Buffer (HW) ← **Translation Storage Buffer (SW)** ← **Software Translation Table (PTs)**

TLB miss handled in software

Software “caching”:

- refill from small, fast translation storage buffer
- if not there, translate from page tables (software translation table)

V9 TLB

TLB

- 64 entries, fully associative, separate data & instruction
- 16B TLB entry
  - TLB tag handled in hardware
  - TLB data handled in software
  - encoded size field in TLB entry
  - global bit: context ignored in TLB hit/miss detection to enable different contexts to share

TLB Tag

63 13	12 0
virtual page number<63:13>	context #

TLB Data

63	61 60	40 39	13 12	6						0
V	Size	Diagnostics	Physical PN<42:13>	Soft	L	C_p	C_v	E	P	W	G

V9 TSB

Translation Storage Buffer (TSB)

- 512 - 64K entry software cache of TLB entries
- provides fast TLB refill for 8KB & 64KB pages
- direct mapped
  - potential entry can be in 2 locations, depending on page size
  - on TLB miss MMU HW generates 2 entry pointers

8KB Page Pointer

63 13	12 4	3 0
TSB base<63:13>	TSB index VA<21:13>	0000

64KB Page Pointer

63 13	12 4	3 0
TSB base<63:13>	TSB index VA<24:16>	0000

V9 Memory Management

TLB miss

- handled in software with hardware support
  - a long procedure; hardware would be too complex
  - can implement various paging structures & replacement algorithms
  - occurs relatively infrequently so does not have to be fast
- procedure
  - hardware puts VA & context id of miss address in a special register (TLB Tag Access Register)
  - hardware calculates TSB entry addresses & put in 2 TSB page pointers
  - load a TSB entry from memory
  - compare tags (VA & context id)
  - if hit, select replacement TLB entry
    - software has determined which one & set a hardware pointer to it
    - HW updates TLB tag from TLB Tag Access Register
    - TLB data loaded by software in one instruction
  - data TLB miss serviced before I TLB miss
- the other 2 pages could also be stored in the TSB
  - store 512KB (4MB) page as 8 (64) 64KB pages
  - use 64KB TSB pointer
    - e.g., for 512KB page, ignore low 3 TSB index bits & check 8 entries

V9 Memory Management

TSB miss

- trap & consult page tables (software translation table)

V9 Memory Management

Issues for page tables

- 64b address space
  - **huge page tables**
  - 6-8 memory accesses if multi-level
- multiple-sized pages
  - **how to represent them**

V9 Memory Management

Solution to both: **forward-mapped page tables**

- store PTEs in n-ary trees
  - leaf nodes store PTEs
  - intermediate nodes store pointers to the next level
  - can have different branching factors at different nodes
- **sparse page tables**
  - **page table pointer (PTP) = 0 ⇒ no levels beneath**
- **representing multiple-sized pages**
  - **PTP = PTE ⇒ larger page size**

Machine Comparison

	21164	Ultra	P6
TLB D	64 entries 8, 16, 32, 64KB FA dual-ported PAL code miss handling	64 entries 8, 64, 512KB, 4MB FA software miss handling	64 entries 4KB, 4MB FA
TLB I	48 entries 8, 16, 32, 64KB FA PAL code miss handling	64 entries 8, 64, 512KB, 4MB FA software miss handling	32 entries 4KB, 4MB FA

Improving Memory Performance

Victim cache (reduce)

Hardware or compiler-based prefetching (hide)

Cache-conscious compiler optimizations (op)

Write buffer with write-through memory update policy (hide)

Handle read miss before block replacement with write-back memory update policy (reduce)

Subblock placement (reduce)

Non-blocking caches (hide)

Requested word first/early restart (reduce)

Cache hierarchy (reduce)

Virtual cache (reduce hit time)

Pipeline writes (hide)

Wider bus (band)

Banked memory/interleaved memory (band)

reduce= reduce miss penalty	hide = hide memory latency
op = eliminate memory op	band = increase memory bandwidth