CpE 313: Microprocessor Systems Design
Fall 2004

Handout 19
TLBs and Virtual Caches

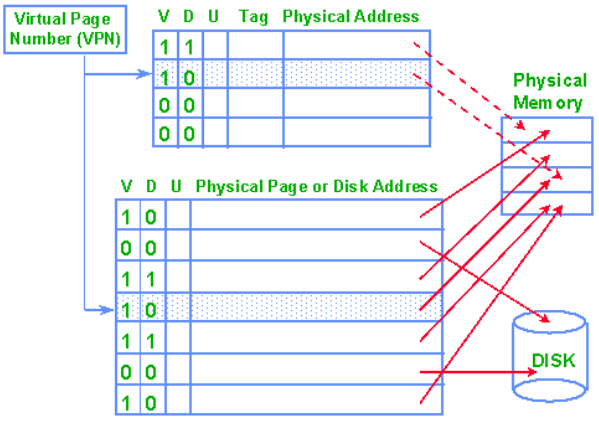
November 30, 2004
Shoukat Ali

shoukat@umr.edu

UMR UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.

TLB Review

- each TLB entry holds a virtual address (called "tag"), the corresponding physical address, and some associated page protection and usage information



1. CPU VPN is compared against ***ALL*** stored VPNs (or tags)

2. if an entry matches, and valid bit is set, physical address is taken out of TLB

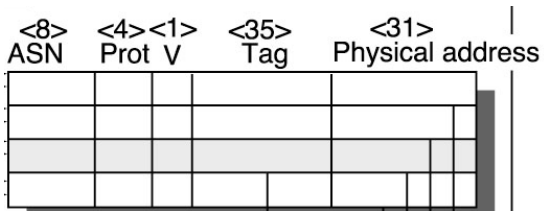
TLB Entries Must be Discarded At Process Switch

- a CPU routinely switches between programs (processes)
- process has its page table
- when a new process is "switched in," the new process's page table must be used
- there is TLB for all processes
- is it true that VPN→PPN mappings stored in TLB become meaningless at every context switch
  - yes
  -

3 3

PID-Augmented TLBs Require Fewer Flushings

- in some TLBs, each entry is augmented with a ASN field
  - ASN = address space identifier (aka PID = process ID)



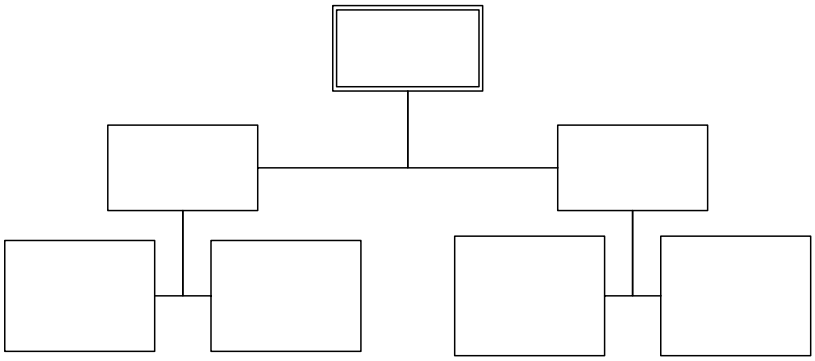
1. *ASN:* VPN from CPU is compared against ***ALL*** stored *ASN:* VPN values

2. if an entry matches, valid bit is set, *and no protection bits are violated*, then physical address is taken out of TLB

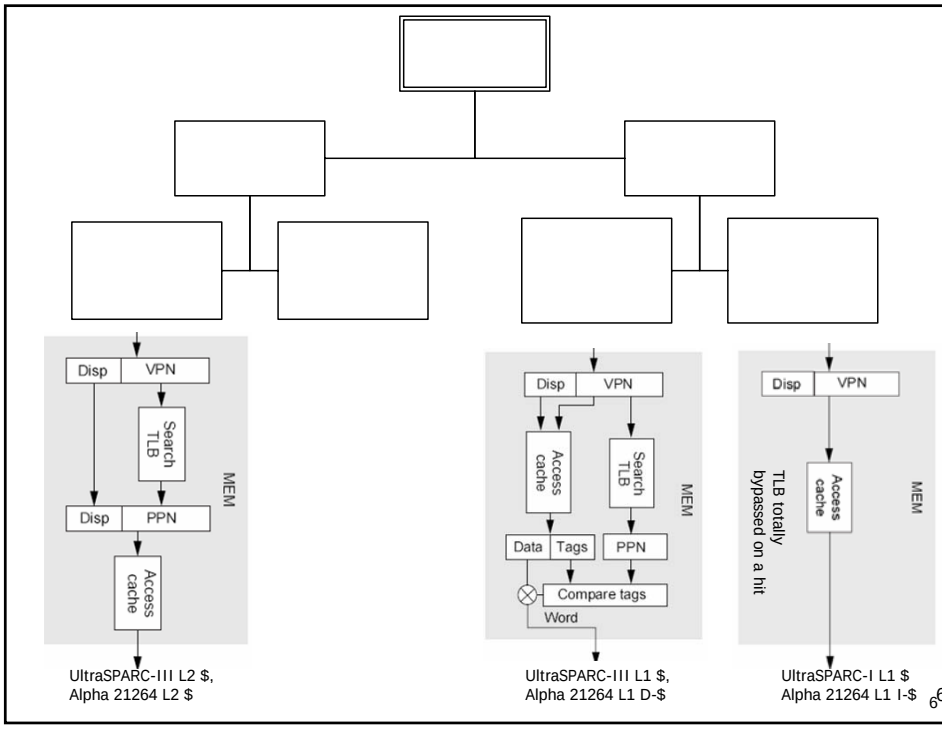
- such TLBs do not need to be flushed at every context switch
  - why?
- at what type of context switch, should this TLB be flushed?

4 4

Which Cache Organization Can Bypass TLB?



5 5



6 6

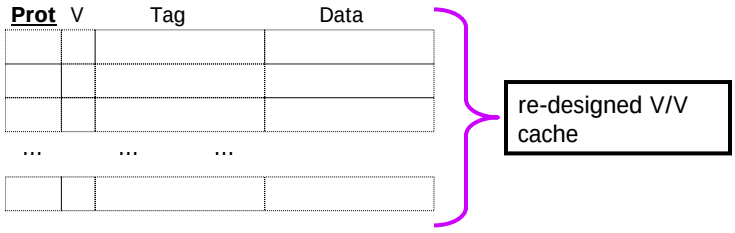
Physically Indexed,

Is Bypassing TLB Fine?

- page table and TLB (which is just cached page table) are like for memory accesses
  - contains protection information that can be checked against
  - bypassing TLB means a memory request can bypass the police checkpoint; NOT GOOD
- *for a virtually indexed, virtually tagged cache, stored in cache are virtual and are compared with the virtual tag*
  - *→ TLB is totally bypassed on*
  - *page table is accessed*
- so how can one re-design V/V caches so that protection information is checked for cache hits?

Enforcing Protection Checks in V/V Caches

- how can one re-design V/V caches so that protection information is checked for cache hits?
- solution:
  - store protection info in both TLB and the cache
  - check the protection information at each access to V/V cache



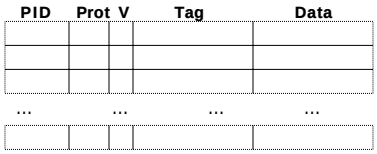
Should a V/V $ be Flushed At Process Switch?

- a CPU routinely switches between programs (processes)
- process has its page table
- when a new process is "switched in," the new process's virtual addresses
- there is cache for all processes
- is it true that (virtual tag→block) mappings stored in a V/V cache become meaningless at every context switch?
  -
  - a V/V cache must be
    incoming process fills up the cache as it proceeds

just like in a TLB

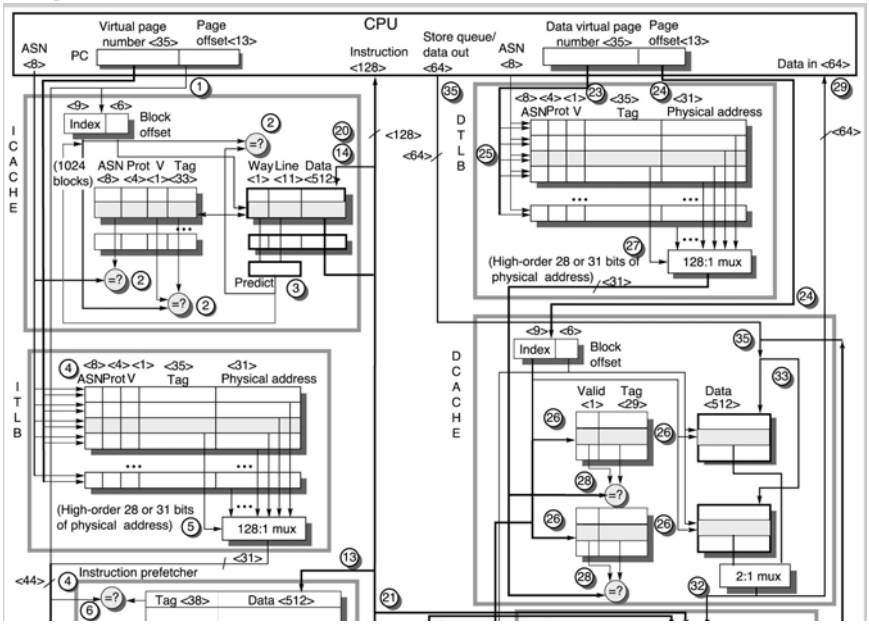
A Better Solution

- augment V/V cache line with a PID (process ID) field
- the CPU always sends PID and VA to the cache
  - in effect, PID:VA make the true virtual address
- the search process changes in the following manner:
  - PID:VA from the CPU is compared against the stored PID:Tag values



- what if a process dies, and its PID is re-cycled (i.e., assigned to a new born process)?
  - now the cache must be flushed

Alpha 21264 Test: Which Cache is a V/V Cache?



For Reference

Table 2. Cache organizations of some microprocessors.

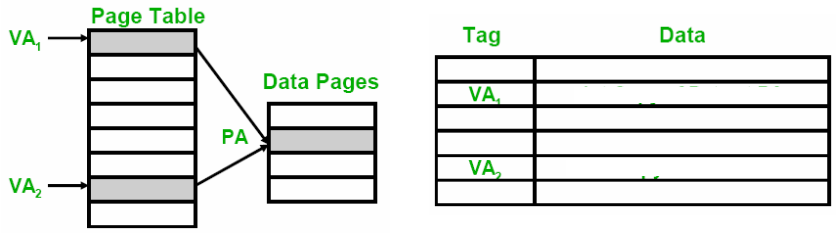
Microprocessor	First-level instruction cache	First-level data cache	Second-level cache
Intel Pentium	8 Kbytes; 2-way SA*; P/P	8 Kbytes; 2-way SA; P/P	No support
Intel PentiumPro	8 Kbytes; 2-way SA; P/P	8 Kbytes; 2-way SA; P/P	256 Kbytes; 4-way SA; P/P
DEC Alpha 21164	8 Kbytes; DM**;V/V	8 Kbytes; DM; P/P	96 Kbytes; 3-way SA; P/P
PowerPC 604	16 Kbytes; 4-way SA; P/P	16 Kbytes; 4-way SA; P/P	No support
PowerPC 620	32 Kbytes; 8-way SA; P/P	32 Kbytes; 8-way SA; P/P	1 to 128 Mbytes; DM; P/P (external)
H-P PA7100	Up to 1 Mbyte; DM; V/P (external)	Up to 2 Mbytes; DM; V/P (external)	No support
H-P PA 8000	Up to 4 Mbytes; DM; V/P (external)	Up to 4 Mbytes; DM; V/P (external)	No support
Mips R4400	64 Kbytes; DM; V/P	64 Kbytes; DM; V/P	128 Kbytes to 4 Mbytes; P/P (external)
Mips R8000	16 Kbytes; DM; V/V	16 Kbytes; DM; V/P	1 to 16 Mbytes; 4-way SA; P/P (external)
Mips R10000	32 Kbytes; 2-way SA; V/P	32 Kbytes; 2-way SA; V/P	1/2 to 16 Mbytes; 2-way SA; P/P (external)
Sun Ultrasparc I, II	16 Kbytes; 2-way SA; P/P	16 Kbytes; DM; V/P	512 Kbytes to 4 Mbytes; P/P (external)

*Set associative
**Direct mapped

Table taken from "VIRTUAL-ADDRESS CACHES, Part 2: Multiprocessor Issues," IEEE Micro, November/December 1997, pages 69—74.

The Issue of Synonyms in VI Caches

- issue: programs sometimes use two or more different virtual addresses for the same physical address
  - these duplicate VA are called synonyms or aliases
  - a VI cache may keep two or more copies of one physical location
  - small problem: *cache pollution*



  - big problem: what if two synonyms are in the cache, and one is modified by a program?
    -

Solution to Synonym Problem

- *synonym problem: the problem of ensuring that a block must never co-exist in the cache under different addresses*
- solution**:** before bringing in a new block in cache, *search the VI cache* for any synonyms if any are found
- how do we search a VI cache for synonyms of v1
  - V/P cache: for each cache block being examined, retrieve the stored tag, and see if it matches tag obtained by TLB translation of v1
  - V/V cache: for each cache block being examined, retrieve the stored tag, and see if it matches tag obtained by TLB translation of v1

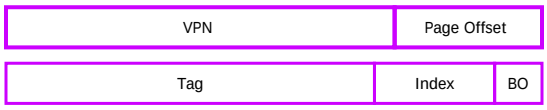
Two Implementations of the Solution

- how do we implement the above solution?
  - UltraSPARC – uses a DMC along with OS-implemented page coloring
  - Alpha 21264 – implemented by HW using antialiasing

15

Key Observation About Synonyms - 1

- by definition, synonyms v1 and v2 are virtual addresses that translate to one physical address p
  1. above implies they have exact same value
  2. can only differ



  3. item 2 implies that indices for synonyms may differ only in superset bits (if there are any superset bits)

- grand conclusion: all synonyms of v1 have to be formed by v1's index

16

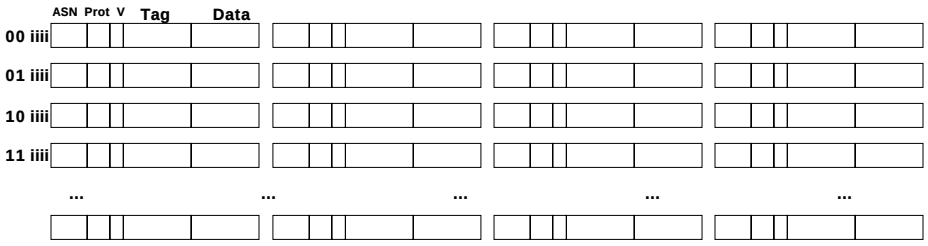
Locating Synonyms in a Set Associative Cache

- assume memory system wants to locate any synonyms of v1
  - i.e., in which cache lines, synonyms of v1, if any, are residing?
- based on grand conclusion (slide 16), determine if the memory system should search the
  - entire cache
  - entire set indexed by v1
  - entire superset formed by v1's index
- to locate v1's synonyms,

17

Example: Locating Synonyms in a Set Associative Cache

- assume a 4-way set associative cache, an index of 6 bits, and a page offset of 4 bits equal to iiii
- to locate v1's synonyms, system needs to search the superset formed by v1's index



18

Alpha 21264's Solution to Synonym Problem

- *synonym problem: the problem of ensuring that multiple copies of a block must never co-exist in the cache under different addresses*

- assume
  - v1 and v2 are two synonyms
  - CPU sends a request for v2
- hardware searches the entire superset to see if tag for any block matches physical address of v2
  - if yes, (which will happen if v1 is in cache)
    -
    - bring the block from main memory and store it in cache in v2's set
  - if no, (which will happen if v1 is not in cache)
    - bring the block from main memory and store it in cache in v2's set

19

Synonym Alignment

- accessing entire superset in Alpha's case is easy because there are only eight blocks in the superset
- superset access gets ve the number of superset bits increases

- is there any way that HW does not have to search the entire superset?

- yes:
  - OS places the restriction that all synonyms must have the same superset bits
  - OS places the restriction that superset bits of the physical page are same as that of virtual page (aka: both pages have the same "color")
  - called "synonym alignment"
  - done in SunOS as part of a scheme called "page coloring"

20

UltraSPARC-III's Solution to Synonym Problem

- with synonym alignment: "*HW searches only one set to see if tag for any block matches physical address of v2*"
- with DMC: a cache set contains only one block

- with synonym alignment + DMC,
  - CPU sends out a request for v2
  - v2 is brought in
  - if v1 is in cache, it is in the same block frame where v2 will be housed
    - v1 is kicked out to make space for v1
      - no need to search

- UltraSPARC-III uses a DMC along with synonym alignment to solve the synonym problem

21

Cost of Aligning Synonyms

- page coloring imposes constraints on memory allocation

- when a new physical page is allocated on a page fault, the memory management algorithm must pick a page with the same color as the virtual color from the free list

- memory is not fully associative any more
  - page fault rate will likely increase

22