Introduction to Computer Systems
15-213/18-243 Spring 2009
March 23, 2009

Virtual Memory

Overview

- **News**
- **Shell Lab**
- **Virtual Memory**
  - Basics
  - Benefits / Costs
  - Address Translation
  - Examples
  - The TLB

News

- Perflab grades returned today
- Shell Lab due next Tuesday, 3/31
- Exam 2 on Apr 7

Shell Lab – Review of concepts

- How to send signals to a process?
- kill (pid_t pid, int signal);
  - Sends the specified signal to process with that pid
  - If pid is negative, will send the signal to ALL processes in that process group instead
  - Eg. kill (1024, SIGINT), sends SIGINT to process 1024
    - kill (-2048, SIGTSTP), sends SIGTSTP to all processes in process group 2048
  - Return value of 0 indicates success
  - See man pages for more details

Shell Lab – cont'd

- Creating new processes – child processes inherit the entire address space, including the blocked vectors and I/O redirects of their parents.
  - Unblock signals in the child if they were blocked in the parent

- Reaping child processes – remember that multiple child processes may terminate, but only one SIGCHLD is sent.
  - What can you do to account for this?

Shell Lab – cont'd

- Block signals before execution of any critical code
  - Critical code is code that would result in a race condition if interrupted
  - Code that manipulates the job list is susceptible to race conditions
  - Remember to unblock after execution!

- Check the return value of ALL system calls!
  - One approach is writing wrapper functions around system calls to perform the error checking
  - The wrapper function should handle the error appropriately (and not just exit the program)

Shell Lab

- Any questions?

Virtual Memory - Basics

- Virtual memory is a technique used to separate memory addressing in programs from actual physical memory access
- Each virtual memory address maps onto a corresponding space in physical storage (DRAM or disk)
- Memory Management Unit (MMU) translates virtual addresses into physical addresses by performing a lookup on a page table, which the processor uses to retrieve from physical storage

Benefits/Costs

- Benefits
  - Acts as a cache for disk accesses (DRAM is > 10000x faster than disk)
  - Memory management -> Allows each process to believe it has its own address space
  - Memory protection -> Each page table entry has several protection bits to control program access to particular regions of physical memory
- Costs
  - Additional lookup needed per physical memory access (mitigated somewhat by TLB)
  - Page faults may occur when referencing memory not stored in DRAM, incurring a huge penalty

Address Translation

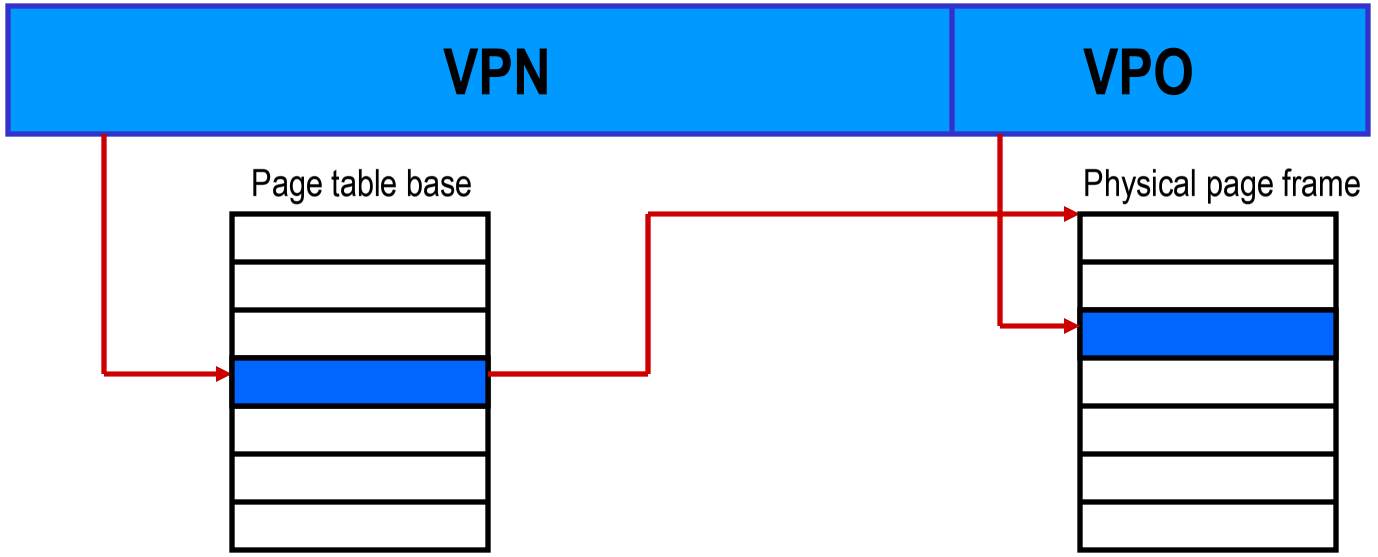
- Each virtual address encodes several things
  - Virtual Page Number(s) – Index into the page table/directory
  - Virtual Page Offset – Offset in terms of bytes from the start of the page frame. This is equal to the physical page offset

1 level page table

2 level page table

Address Translation

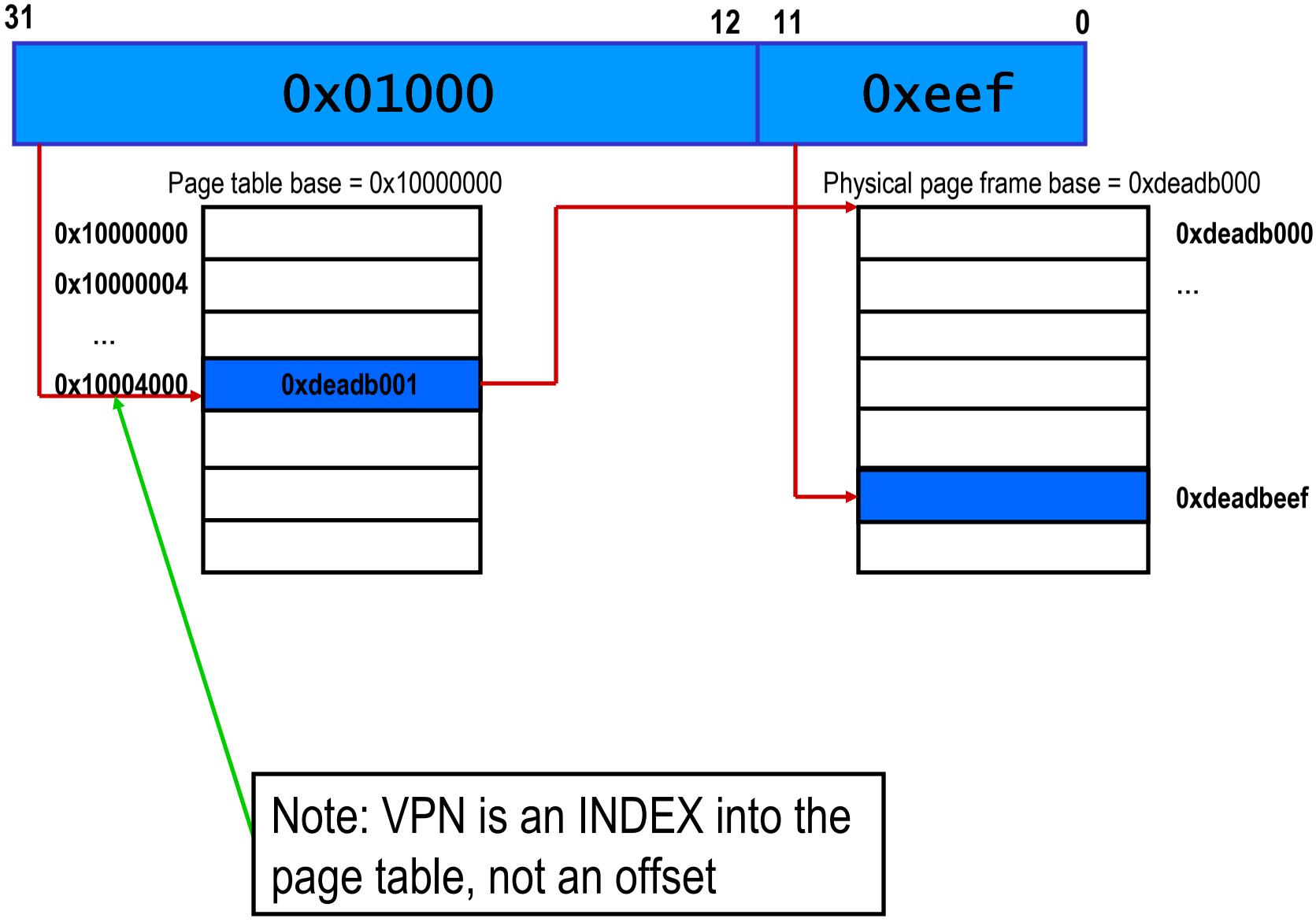
- When a memory access is requested, the processor will generate a virtual address and send it to the MMU
- The MMU will look up the page table entry by using the VPN to index into the page table, starting from the page table base (usually stored in a Page Table Base Register)



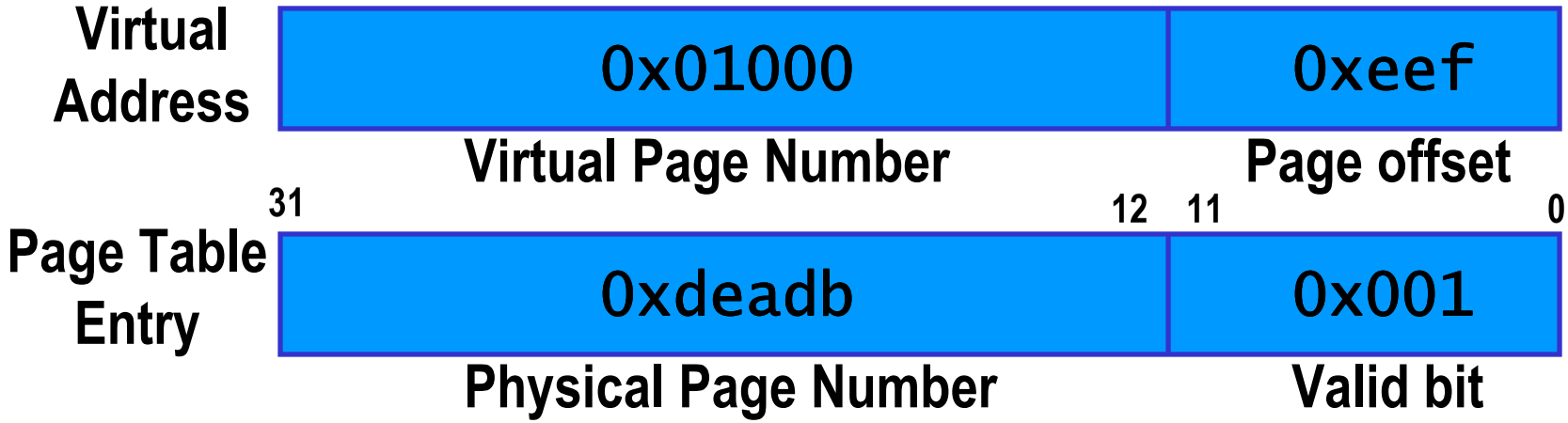
Page Table Entry

- Each page table entry holds the starting address of the corresponding page frame in physical memory.
- The lowest few bits of each page table entry also encodes the valid and permission bits:
  - Valid bit: indicates whether page exists in physical memory (DRAM), if not a page fault will occur upon access
  - Supervisor bit: indicates whether processes in user mode may access this region of memory
  - Read/Write bit: indicates if reads and/or writes are allowed to this region of memory
- Encoding of valid and permission bits vary from system to system

Example: 1 level page table, 4KB page frame



Example – cont'd

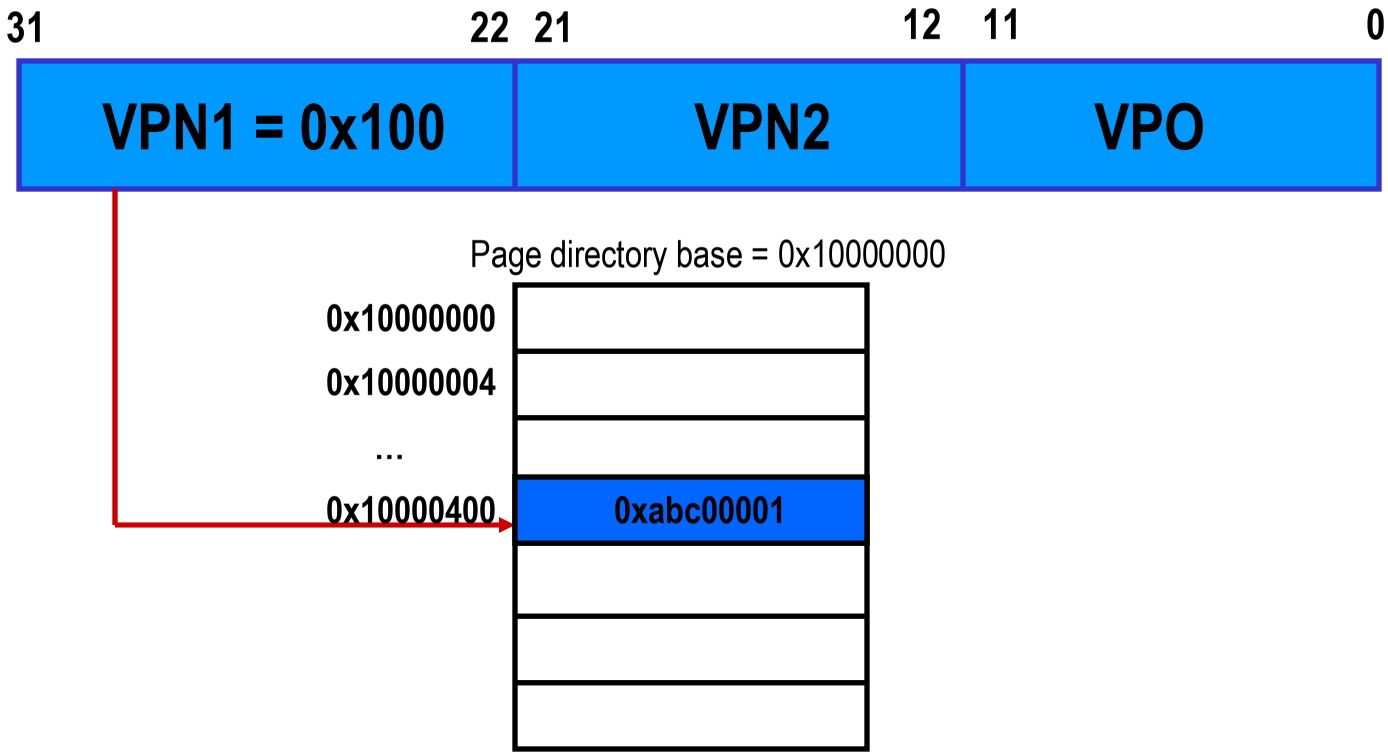


This tells us that virtual page 0x01000 is stored at the physical address starting at 0xdeadb000. The valid bit is set, so the page exists in physical memory. Add the page offset to obtain the actual physical location 0xdeadbeef

Question

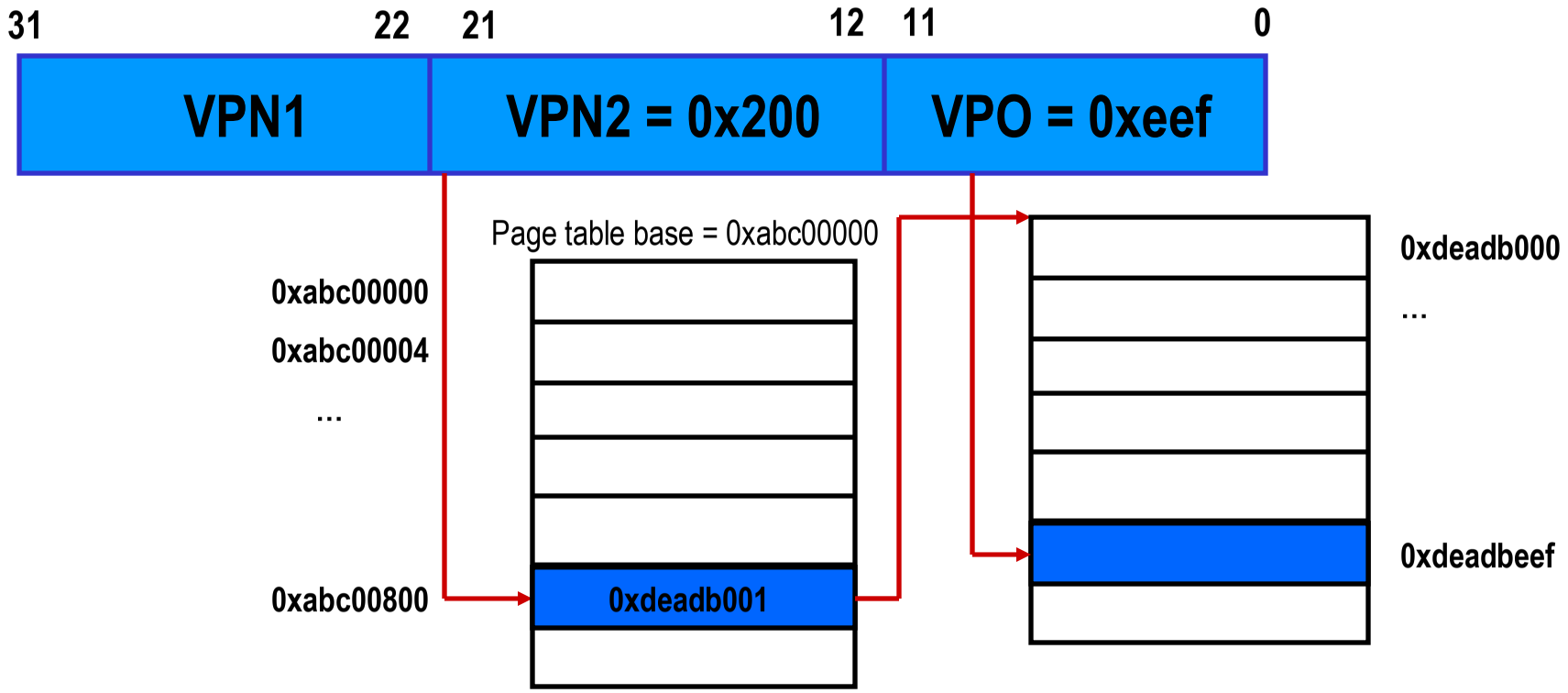
- How big must the page table be, if the VPN consists of 20 bits? (Each page table entry is the size of a memory address, 32 bits in this case)
- What if only a small amount of memory needs to be managed by VM?
  - Solution: Use a two level page table to manage sparse allocations of physical memory
- In a two level page table, the first level is called a page directory. Each page directory entry contains the address of a page table and a valid bit.
  - More efficient usage of memory if not all page tables need to be allocated

Example: 2 level page table



Use the first VPN to index into the page directory. This gives the address of the start of the page table.

2-level page table – cont'd



Use the second VPN to index into the page table. This gives the address of the start of the page frame. Add the offset to obtain the location in physical memory.

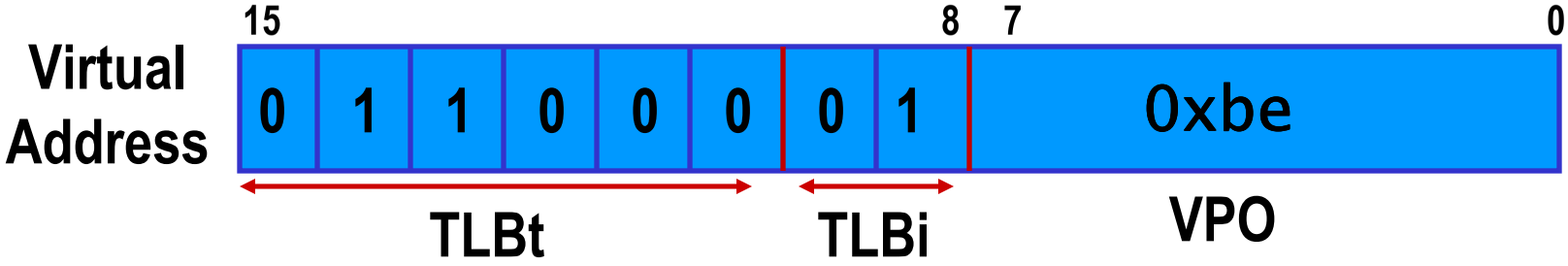
Translation Lookaside Buffer (TLB)

- Every physical memory access requires two memory accesses: page table lookup, and actual memory access. A high cost for the benefits of virtual memory
- The TLB is an on-chip cache that holds page table entries
- The MMU checks the TLB for a cached PTE, and if there is a TLB hit, looks up the cached page table entry without needing to access main memory
- If there is a TLB miss, the MMU will proceed to fetch the PTE from the L1 or L2 cache, or at worst main memory. The newly fetched PTE will be stored in the TLB.
- Programs with good spatial and temporal locality can better exploit the TLB

TLB Example

Assume a 16-bit system with pages of size 256 bytes.

The TLB has 4 sets and is 4-way set associative. The VPN is used to address the TLB.



Note: If the TLB has 2^n sets, then there are n TLBi bits

To check for TLB hit, the set corresponding to the TLBi (TLB index) is checked, in this case, set 1. All cached PTEs in the set are checked to see if the tag bits match the TLBt in the original Virtual Address (in this case, tag = 0x18).

TLB Example – cont'd

Set	Tag	PTE	Tag	PTE	Tag	PTE	Tag	PTE
00	0x02	0x8a00	0x20	0x4201	0x05	0x4800	0x3a	0x3301
01	0x05	0xa001	0x00	0xb001	0x1f	0xab00	0x18	0xba01
10	…							
11								

Note: the last bit in the PTE is the valid bit

In the above case, there is a TLB hit and the MMU does not have to fetch the PTE from main memory. The PPN obtained is 0xba00, which gives the physical address of 0xbabe.

Questions?