

CSE 141 – Computer Architecture
Fall 2005

Lectures 17
Virtual Memory

Pramod V. Argade

November 23, 2005

Announcements

- **Final Review Discussion Section**

- Wed. Nov. 30, 6:30 - 7:50, Center 216

- **Final Exam**

When: Thursday., December 8, 7 - 9:59 PM

Where: Center 216

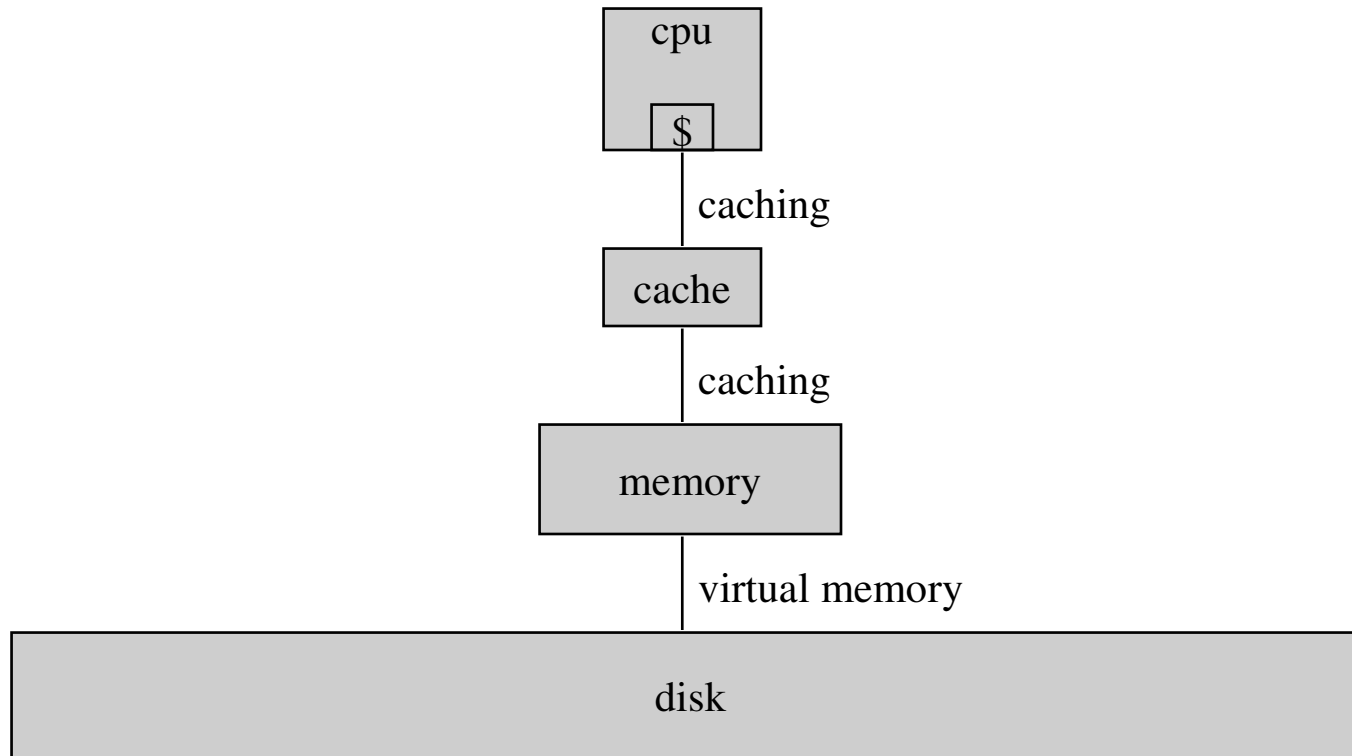
Course Schedule

Lecture #	Date	Day	Lecture Topic	Quiz Topic	Homework Due
1	9/26	Monday	Introduction, Ch. 1	-	-
2	9/28	Wednesday	ISA, Ch. 2	-	-
3	10/3	Monday	Arithmetic Part 1, Ch. 4	ISA	#1
4	10/5	Wednesday	Arithmetic Part 2, Ch. 4	-	-
5	10/10	Monday	Performance, Ch. 3	Arithmetic	#2
6	10/12	Wednesday	Single cycle CPU, Ch. 5	-	-
7	10/17	Monday	Single cycle CPU, Ch. 5	Performance	#3
8	10/19	Wednesday	Multi-cycle CPU, Ch. 5	-	-
9	10/24	Monday	Multi-cycle CPU, Ch. 5	Single Cycle CPU	#4
10	10/26	Wednesday	Review for the Midterm	-	-
	10/31	Monday	Mid-term Exam	-	-
11	11/2	Wednesday	Exceptions, Ch. 5 and Pipelining, Ch. 6	-	-
12	11/7	Monday	Pipelining, Ch. 6	-	-
13	11/9	Wednesday	Data and control hazards, Ch. 6	-	-
14	11/14	Monday	Data and control hazards, Ch. 6	Pipeline Hazards	#5
15	11/16	Wednesday	Memory & cache design, Ch. 7	-	-
16	11/21	Monday	Memory & cache design, Ch. 7	Cache	#6
17	11/23	Wednesday	Virtual Memory & cache design, Ch. 7	-	-
18	11/28	Monday	Course Review	-	-
	12/8	Thursday	Final Exam 7:00 - 9:59 PM Center 216		

Virtual Memory

Virtual Memory

- Virtual memory is the name of a technique that allows us to view main memory as a cache of a larger memory space (on disk).
 - Allows efficient and safe sharing of memory among programs
 - Creates an illusion of providing unlimited memory to programs



Virtual Memory (VM)

- Each program is compiled to run in its own address space
- A single program may exceed the size of primary memory
- Multiple programs may dynamically share portions of memory
- Main memory need contain only the active portions of the program
- VM and caching have different historical roots
- VM is like caching, but uses different terminology

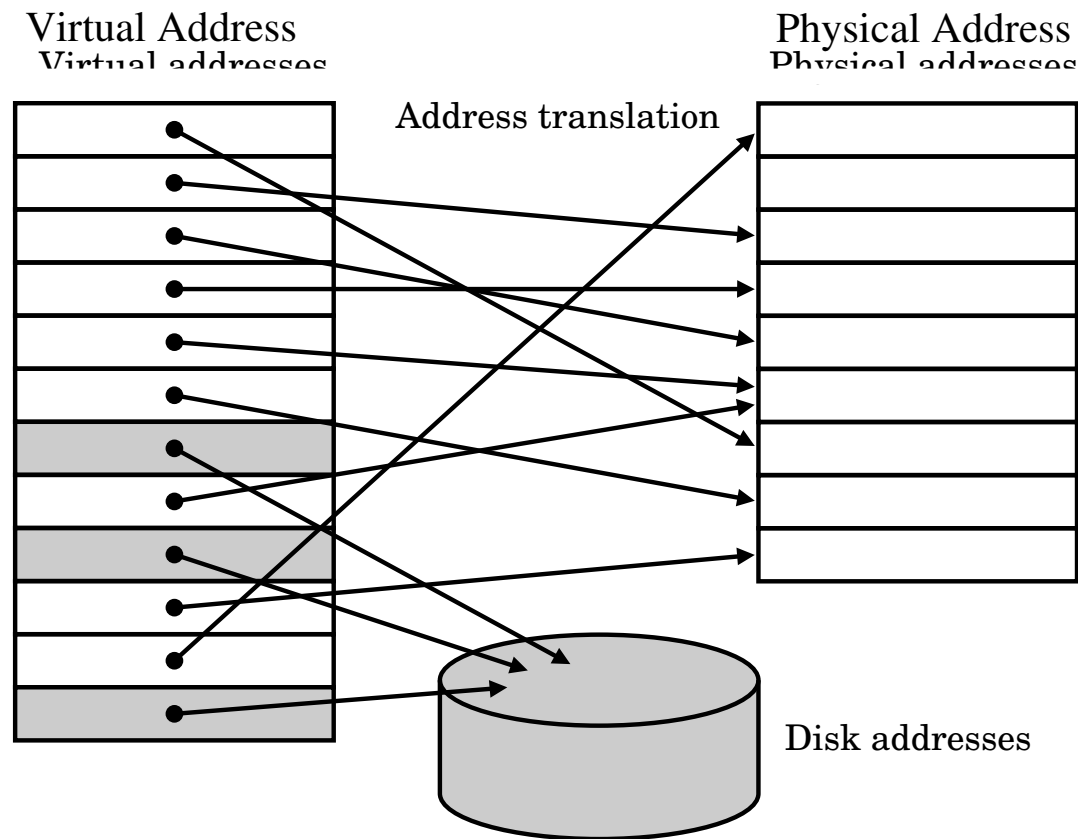
<u>Cache</u>	<u>VM</u>
block	page
cache miss	page fault
address	virtual address
index	physical address (sort of)

Advantages of Virtual Memory

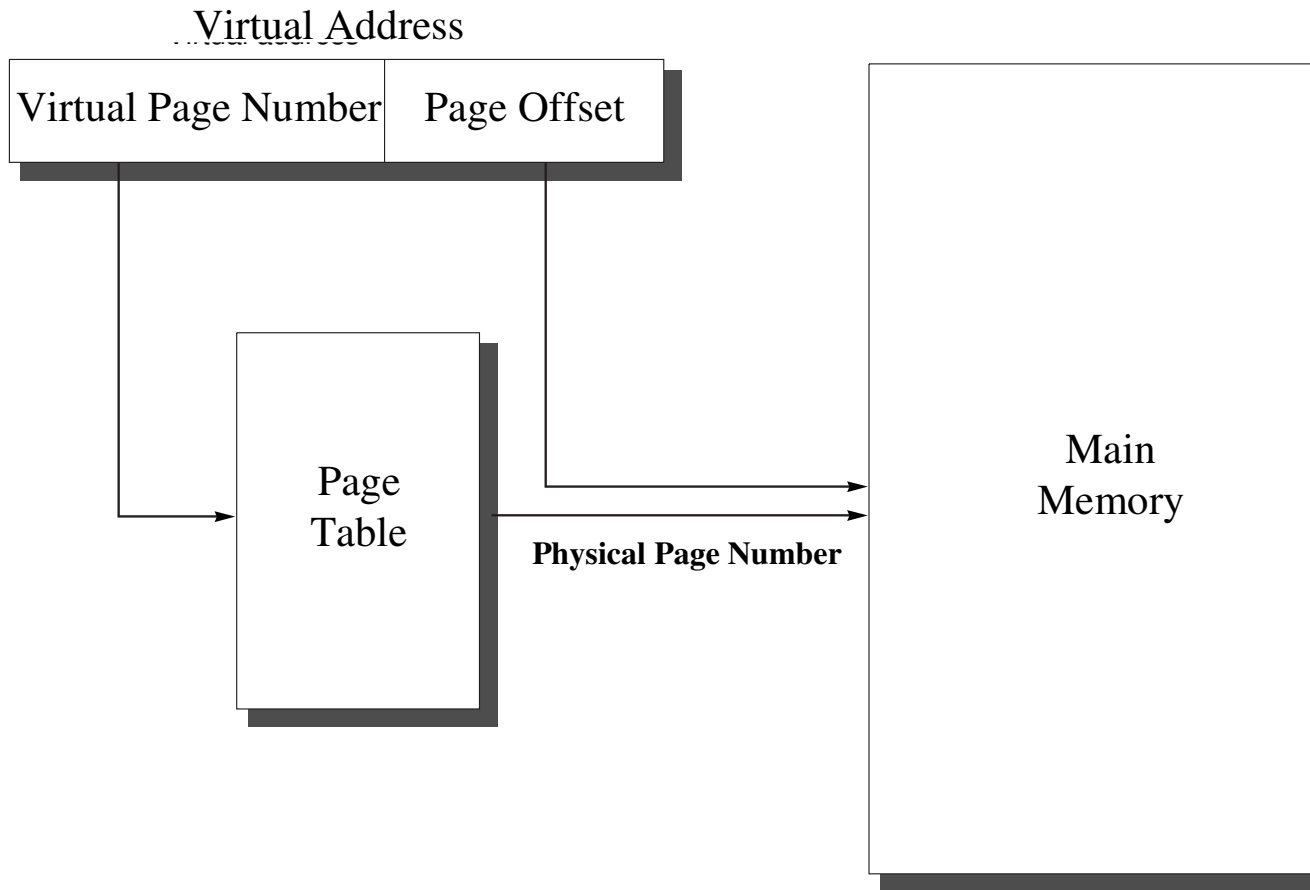
- Performance
 - Large amount of memory accessed efficiently
- Memory sharing among multiple programs
- Protection
 - Simultaneous (time-sharing) execution of multiple programs
 - Use of “kernel space” and “user space”
- Ease of programming/compilation
- Efficient use of memory

Memory Mapping/Address Translation

- Virtual to physical address mapping
- Page may be present or absent in main memory
- Page may be resident on the disk
- Two virtual pages may map to the same physical address



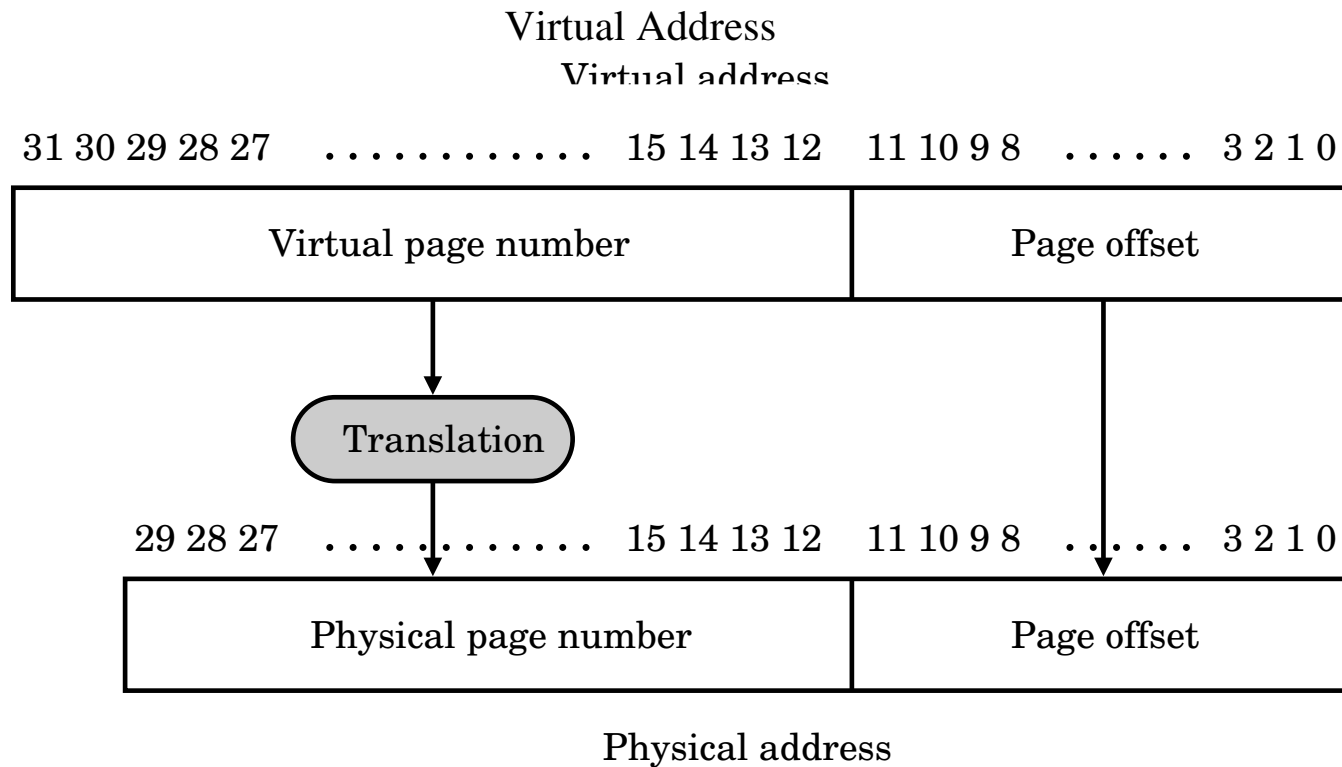
Mapping Virtual to Physical Address



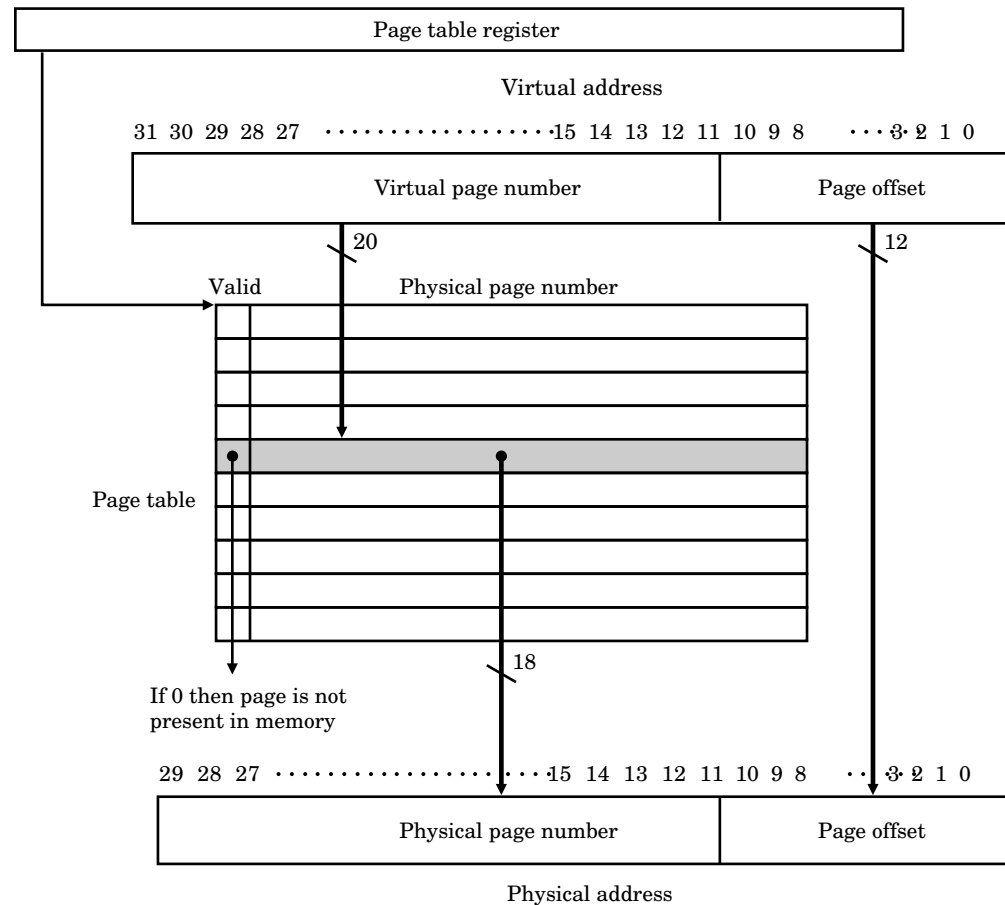
Mapping from a Virtual to a Physical Address

Example:

- Virtual address space 4 Gbytes
- Physical address space 1 Gbytes
- Page size 4 Kbytes



Address Translation via the Page Table



Notes:

- The page table contains mapping for every possible virtual page
- Valid bit indicates whether the page is present in the main memory
- Extra bits in the page table are used for protection information

Cache vs Virtual Memory Access

- Access time
time between when a read is requested and when the desired word arrives
- Transfer time
time it takes to transfer the whole request

Parameter	First-level Cache	Virtual memory
Block (Page) size	16 - 128 bytes	4096 - 65,536 bytes
Hit time	1 - 2 clock cycles	40 - 100 clock cycles
Miss Penalty	8 - 100 clock cycles	700,000 - 6,000,000 clock cycles
(Access time)	(1 - 60 clock cycles)	(500,000 - 4,000,000 clock cycles)
(Transfer time)	(2 - 40 clock cycles)	(200,000 - 2,000,000 clock cycles)
Miss Rate	0.5 - 10%	0.00001 - 0.001%
Data memory size	0.016 - 1 MB	16 - 8192 MB

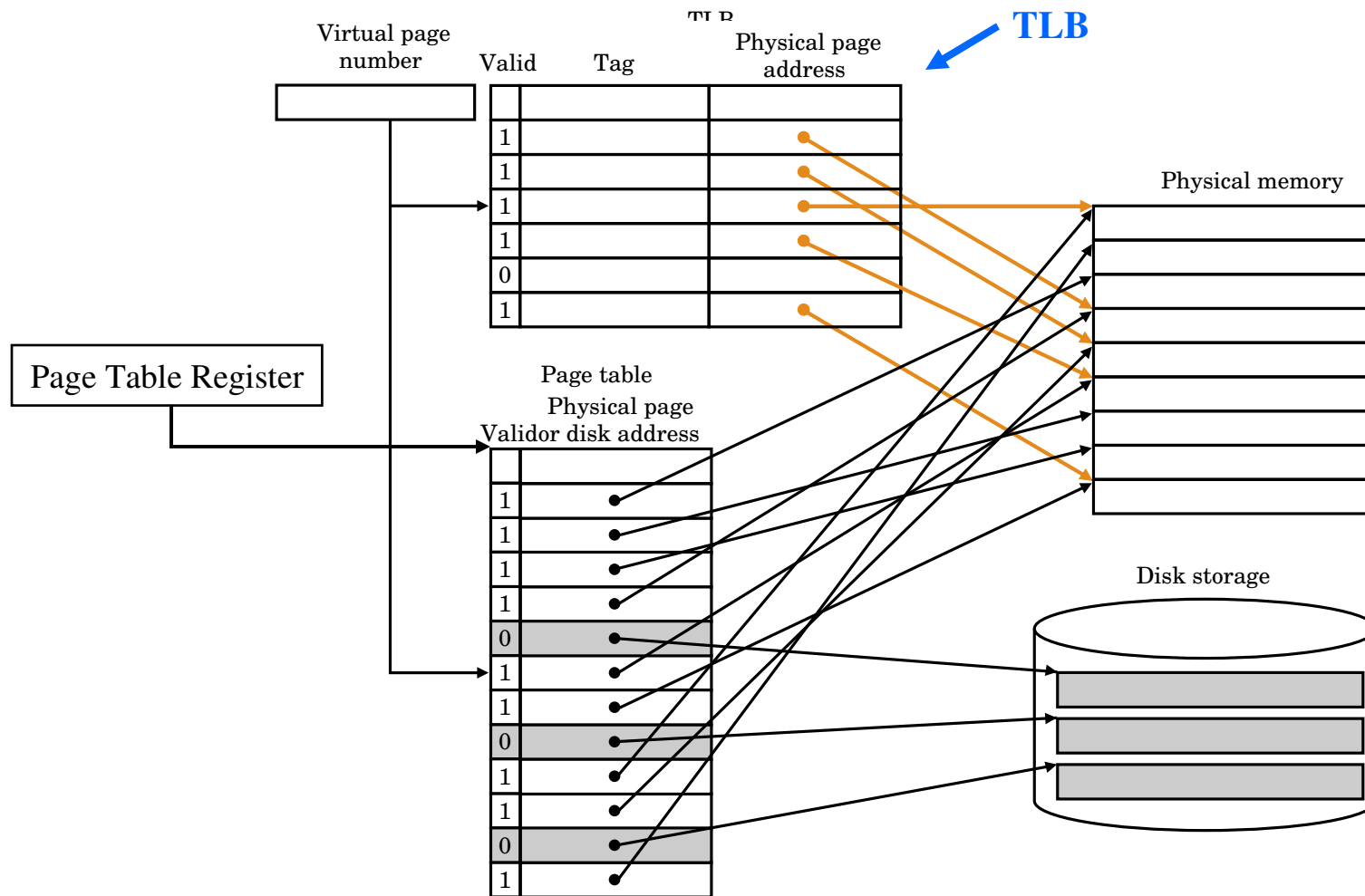
- VM has very high miss penalty
 - large pages (4 KB to MBs)
 - associative mapping of pages (typically fully associative)
 - software handling of misses (but not hits!!)
 - *write-through* not an option, only write-back

Translation Look-aside Buffer

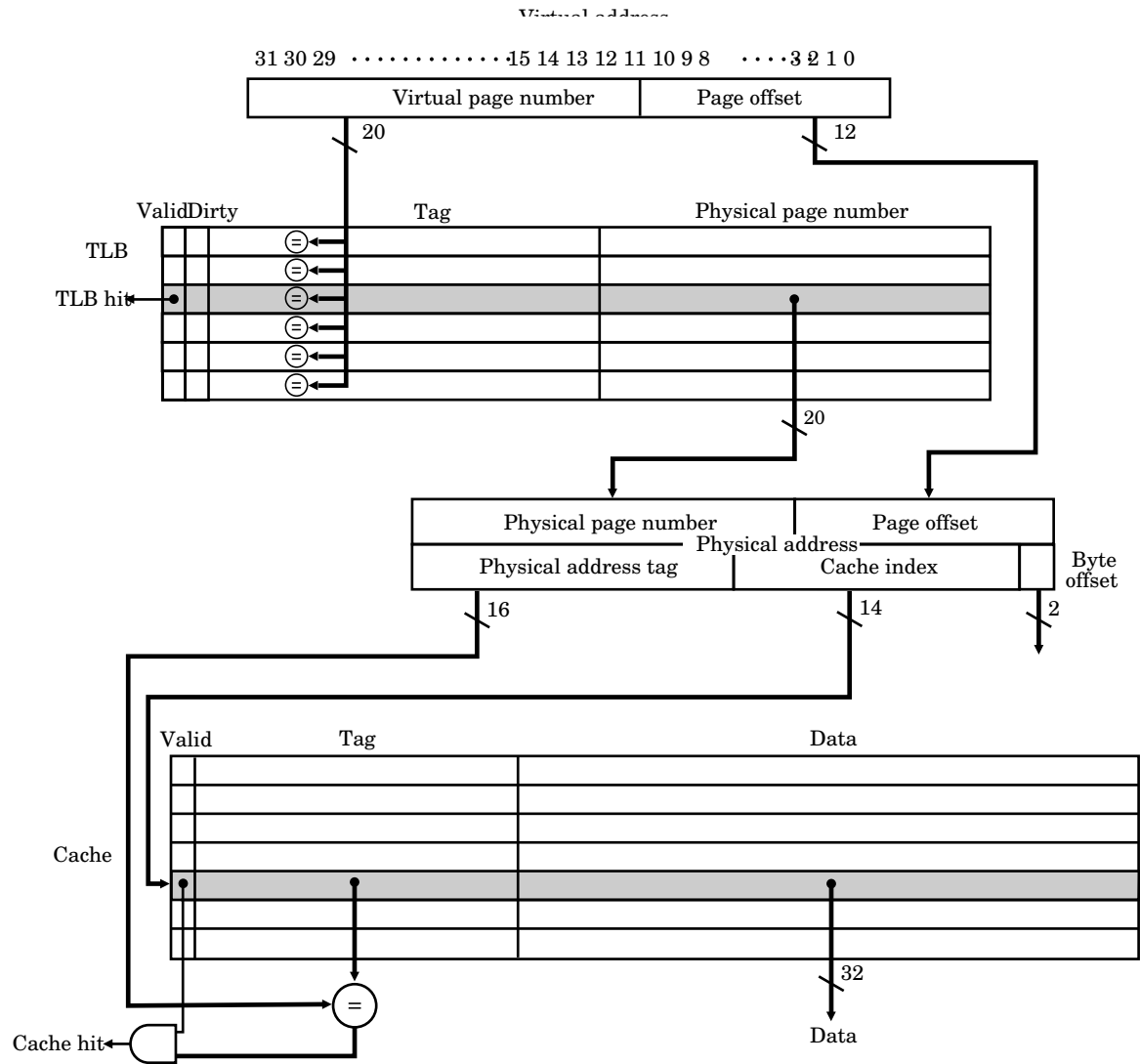
- Address translation could be expensive to perform for every memory access
 - page tables are stored in main memory
 - need to access page table before accessing data location
- Solution is to remember the last address translation so the mapping lookup can be skipped
 - use a translation buffer to hold the last N translations

TLB: Making Address Translation Fast

Translation Lookaside Buffer: A cache for address translations



TLB and Cache

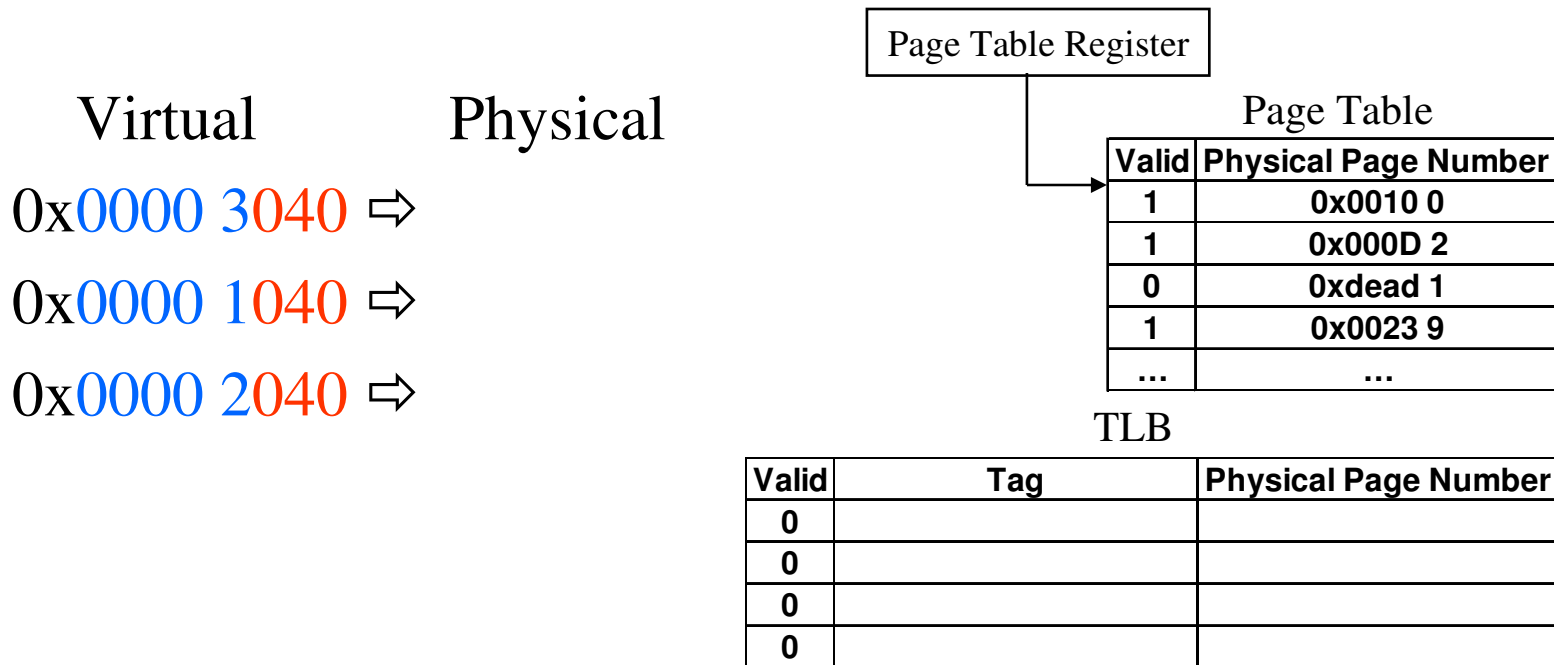


Example: Page Table Size

- What is the total page table size if:
 - Virtual address is 32 bits
 - 8 Kbytes page size
 - Assume that each page table entry is 4 bytes
- Page Table Size computation

Example: Address translation

- Using the page table shown, translate following 32-bit virtual addresses into physical addresses. Make entries in the TLB assuming LRU replacement.
 - The page size is 4 Kbytes
 - Addresses: 0x0000 3040, 0x0000 1040, 0x0000 2040



Memory Access Events

- TLB Miss
 - Entry does not exist in the TLB
 - A page table entry is brought into the TLB
- Page fault
 - The valid bit is not set for the page table entry
 - A page of data from disk is brought in to the main memory
- Cache miss
 - Tag mismatch or valid bit not set
 - Cache line is brought from next level of memory hierarchy (depending on the policy for a write)

Processing a Page Fault

- If the valid bit for a virtual page is off, page fault occurs
- CPU generates an exception
- OS takes control
- OS finds the page in the next level of hierarchy (disk)
- OS decides where to place the requested page in memory
- OS copies the page from next level of hierarchy to memory
- OS sets valid bit in the page table entry for the virtual address
- OS returns from the exception
- Program re-executes the same instruction
- Page translation finds valid bit set for the virtual page
- Data access succeeds

What is a Process?

- Program state consists of:
 - Page tables, PC and the registers
- This state is referred to as a process
- Process is an instance of a program executing on a CPU

Implementing Protection with VM

- Protection is essential for:
 - Allowing single main memory to be shared among multiple processes
 - Prevent one process from writing into the memory space of another
 - Prevent a user process from modifying its own page tables
 - Controlling raw access to peripheral devices
- Hardware capabilities needed for protection
 - Two operating modes: user mode and kernel mode of execution
 - A portion of the CPU state that a user process can read, but not write
 - This is the user/kernel mode bit
 - A mechanism to switch between user mode and kernel mode
 - Accomplished by a system call

Additional bits in the Page Table

- User or Kernel bit
 - This bit restricts access to some pages to kernel only
- Write bit
 - This bit restricts read-only or read/write access to a page
- Referenced bit
 - OS periodically sets this bit to zero
 - It is set by CPU hardware when the page is referenced
 - Used by OS for replacing the page with other memory pages
- Dirty bit
 - If a process writes to a page, the dirty bit is set
 - It is used by OS to write the page to secondary storage before replacing it

Virtual Memory Key Points

- How does virtual memory provide:
 - illusion of large main memory?
 - sharing?
 - performance?
 - protection?
- Virtual Memory requires twice as many memory accesses, so we cache page table entries in the TLB.
- Three things can go wrong on a memory access: cache miss, TLB miss, page fault.