

CSE 141 – Computer Architecture
Spring 2005

Lectures 18
Course Review

Pramod V. Argade

June 1, 2005

CSE141: Introduction to Computer Architecture

Instructor: Pramod V. Argade (p2argade@cs.ucsd.edu)
Office Hour:
Mon. 5:00 - 6:00 PM (AP&M 5218)

TAs:

Baris Arslan: barslan@cs.ucsd.edu
Chris Roedel: croedel@cs.ucsd.edu
Raid Ayoub: rayoub@cs.ucsd.edu
Leo Porter: leporter@cs.ucsd.edu

Textbook: Computer Organization & Design
The Hardware Software Interface, 3rd Edition.
Authors: Patterson and Hennessy

Web-page: <http://www.cse.ucsd.edu/classes/sp05/cse141>

Announcements

- **Office Hour**

- Monday, June 6th: 6:30 - 8 PM, AP&M 5218
- Wednesday, June 8th: 6:30 - 8 PM, AP&M 5218

- **Final**

When: Friday, June 10th, 7 - 10 PM

Where: SOLIS 107

No calculators!

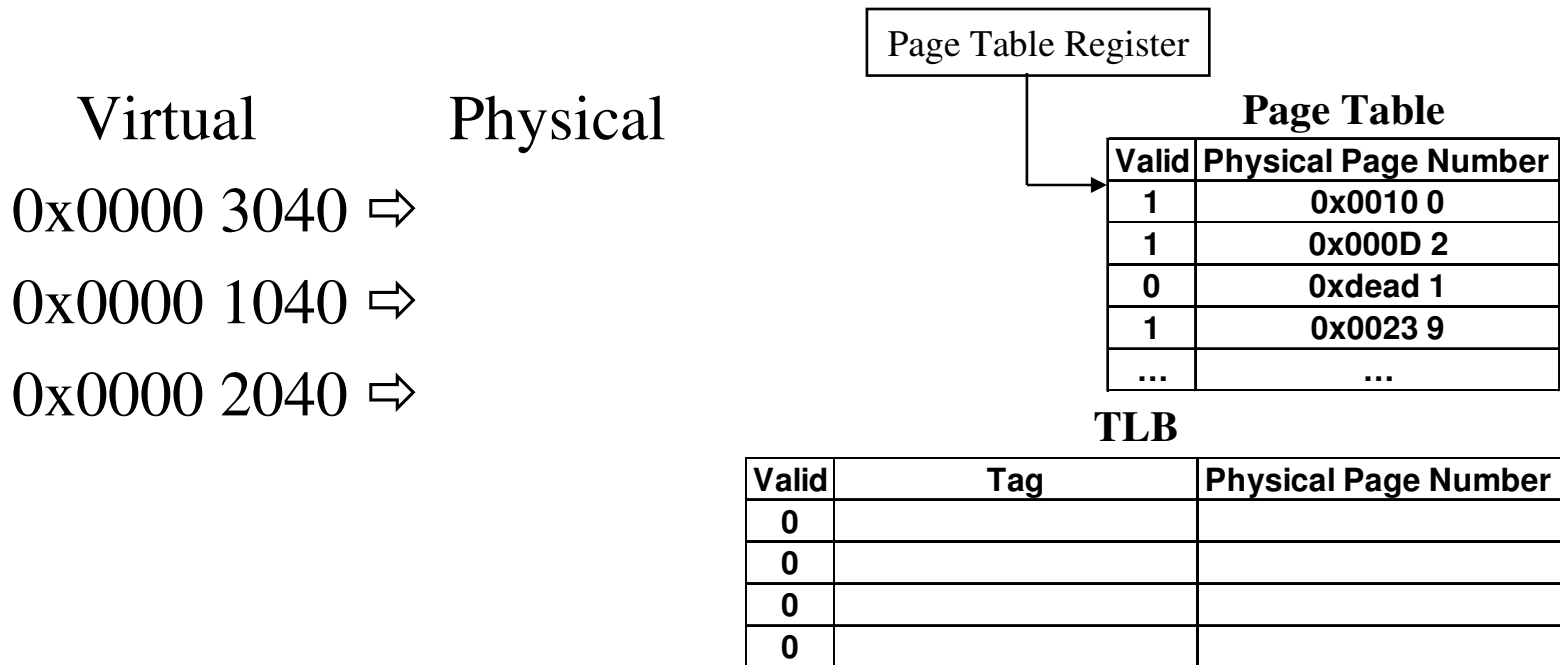
- **Conflict with final? See me or send email**

Course Schedule

Lecture #	Date	Day	Lecture Topic	Quiz Topic	Homework Due
1	3/28	Monday	Introduction, Ch. 1	-	-
2	3/30	Wednesday	Performance, Ch. 4	-	-
3	4/4	Monday	ISA, Ch. 2	Performance	#1
4	4/6	Wednesday	Arithmetic, Ch. 3	-	-
5	4/11	Monday	Arithmetic, Ch. 3	ISA	#2
6	4/13	Wednesday	Single cycle CPU, Ch. 5	-	-
7	4/18	Monday	Single cycle CPU, Ch. 5	Arithmetic	#3
8	4/20	Wednesday	Multi-cycle CPU, Ch. 5	-	-
9	4/25	Monday	Multi-cycle CPU, Ch. 5	Single Cycle CPU	#4
10	4/27	Wednesday	Review for the Midterm	-	-
	5/2	Monday	Mid-term Exam Center 101	-	-
11	5/4	Wednesday	Exceptions, Ch. 5 and Pipelining, Ch. 6	-	-
12	5/9	Monday	Pipelining, Ch. 6	-	-
13	5/11	Wednesday	Data and control hazards, Ch. 6	-	-
14	5/16	Monday	Data and control hazards, Ch. 6	Pipeline Hazards	#5
15	5/18	Wednesday	Memory & cache design, Ch. 7	-	-
16	5/23	Monday	Memory & cache design, Ch. 7	-	-
17	5/25	Wednesday	Virtual Memory & cache design, Ch. 7	Cache	#6
No Class	5/30	Monday	Memorial Day Holiday	-	-
18	6/1	Wednesday	Course Review	-	-
	6/10	Friday	7 - 10 PM, SOLIS 107: Final Exam		

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 Problems

- TLB Miss
 - Entry does not exist in the TLB
 - A mechanism must be provided to bring a page table entry into the TLB
- Page fault
 - The valid bit is not set for the page table entry
- Cache miss
 - Tag mismatch or valid bit not set
 - Cache line is brought in (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 returns from the exception
- Program re-executes the same instruction
- Page translation finds valid bit on 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 state
- Process is an instance of a program executing on a CPU

Implementing Protection with VM

- Protection is essential for:
 - Allowing to share a single main memory among multiple processes
 - Prevent once 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 mode: 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 usr/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.

Example

Assuming $M \Rightarrow M$ Forwarding for $LW \Rightarrow SW$

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
ADD R1, R2, R3															
SW R1, 1000(R2)															
LW R7, 2000(R2)															
ADD R5, R7, R1															
LW R8, 2004(R2)															
SW R7, 2008(R8)															
ADD R8, R8, R2															
LW R9, 1012(R8)															
SW R9, 1016(R8)															

Example: Cache Organization

If you knew that the address 0x99B1 mapped to set 77 (base 10), what could you identify about the cache? You can assume that the cache is two-way set associative and that we are using 16-bit addresses and data.

1. How many sets are in the cache?
2. What is the block size?
3. What is the total size of the cache, including the valid bit and the tag bits?
4. What is the block offset for the address specified above?
5. What is the byte offset for the address specified above?
6. What is the tag for the address specified above?

Example: Cache Hit/Miss

Consider a two-way set associative cache with 4 sets. Assume 1 word (4 byte) blocks and LRU replacement scheme. Also assume that all valid bits are zero in the cache. For the byte address trace presented below:

0, 4, 8, 11, 12, 20, 8, 33, 15, 27, 29, 50, 33, 10, 21, 2

1. Update the entries in the cache and show the final state of the cache, including valid and tag bits. In the data portion, show which bytes are stored.
2. How many hits are there in the cache?
3. How many misses are there in the cache?
4. If you were allowed to double the size of the cache, what change would you make to decrease the number of misses the most?

Example: Virtual Memory and TLB

A 32-bit processor has a virtual memory with page size of 4 Kbytes. The physical memory space is 256 Mbytes. There is a 4 entry fully associative TLB with LRU replacement policy. All the entries in the TLB are initially invalid. Following four accesses show virtual addresses and corresponding physical addresses.

Virtual Address	Physical Address
0x0000 210a	0x000 010a
0x0000 0004	0x000 4004
0x0001 0010	0x000 1010
0x0030 2244	0x000 2244

1. In the TLB, how many bits are required for the tag?
2. In the TLB, how many physical page address bits are required?
3. Show the entries in the TLB after the access to the above four addresses are complete.
4. Explain why a “write bit” is needed in a TLB as well as in the page tables

Announcements

- **Office Hour**

- Monday, June 6th: 6:30 - 8 PM, AP&M 5218
- Wednesday, June 8th: 6:30 - 8 PM, AP&M 5218

- **Final**

When: Friday, June 10th, 7 - 10 PM

Where: SOLIS 107

No calculators!

- **Conflict with final? See me or send email**