CSIS 429 Operating Systems

Lecture 10: Page Faults

October 12th 2020

Textbook chapters

Read “Advanced Page Tables” and “Swapping”

Intro	Virtualization		Concurrency	Persistence	Appendices
Preface	3 *Dialogue*	12 *Dialogue*	25 *Dialogue*	35 *Dialogue*	*Dialogue*
TOC	4 Processes	13 Address Spaces	26 Concurrency and Threads code	36 I/O Devices	Virtual Machines
1 *Dialogue*	5 Process API code	14 Memory API	27 Thread API	37 Hard Disk Drives	*Dialogue*
2 Introduction code	6 Direct Execution	15 Address Translation	28 Locks	38 Redundant Disk Arrays (RAID)	Monitors
	7 CPU Scheduling	16 Segmentation	29 Locked Data Structures	39 Files and Directories	*Dialogue*
	8 Multi-level Feedback	17 Free Space Management	30 Condition Variables	40 File System Implementation	Lab Tutorial
	9 Lottery Scheduling code	18 Introduction to Paging	31 Semaphores	41 Fast File System (FFS)	Systems Labs
	10 Multi-CPU Scheduling	19 Translation Lookaside Buffers	32 Concurrency Bugs	42 FSCK and Journaling	xv6 Labs
	11 *Summary*	20 Advanced Page Tables	33 Event-based Concurrency	43 Log-structured File System (LFS)	Flash-based SSDs
		21 Swapping: Mechanisms	34 *Summary*	44 Data Integrity and Protection	
		22 Swapping: Policies		45 *Summary*	
		23 Case Study: VAX/VMS		46 *Dialogue*	
		24 *Summary*		47 Distributed Systems	
				48 Network File System (NFS)	
				49 Andrew File System (AFS)	
				50 *Summary*	

Address Translation with Paging

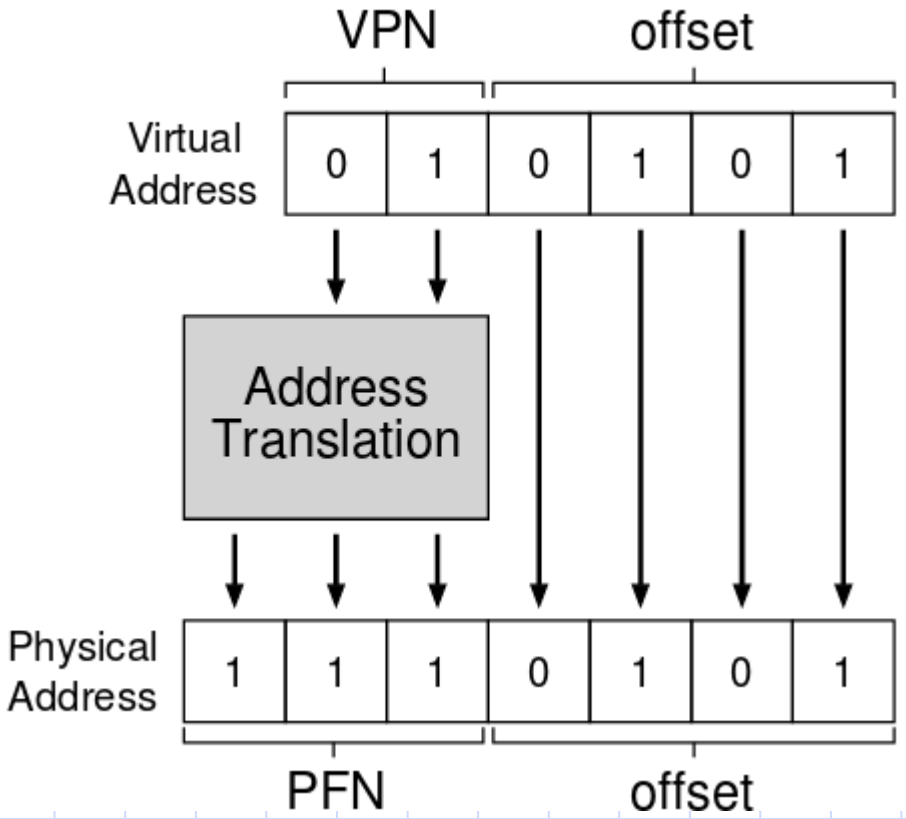
Example: Instruction to load data from memory to register

```
movl <virt-addr>, %eax
```

64 byte virtual address space

page size is 16 bytes

128 byte physical address space



Translation Lookaside Buffer

Every virtual memory reference will involve the TLB to speed address translations. A TLB miss is expensive → shows up as slow execution time.

We can see speed-ups due to TLB hit rate in our programs – mainly in executing loops.

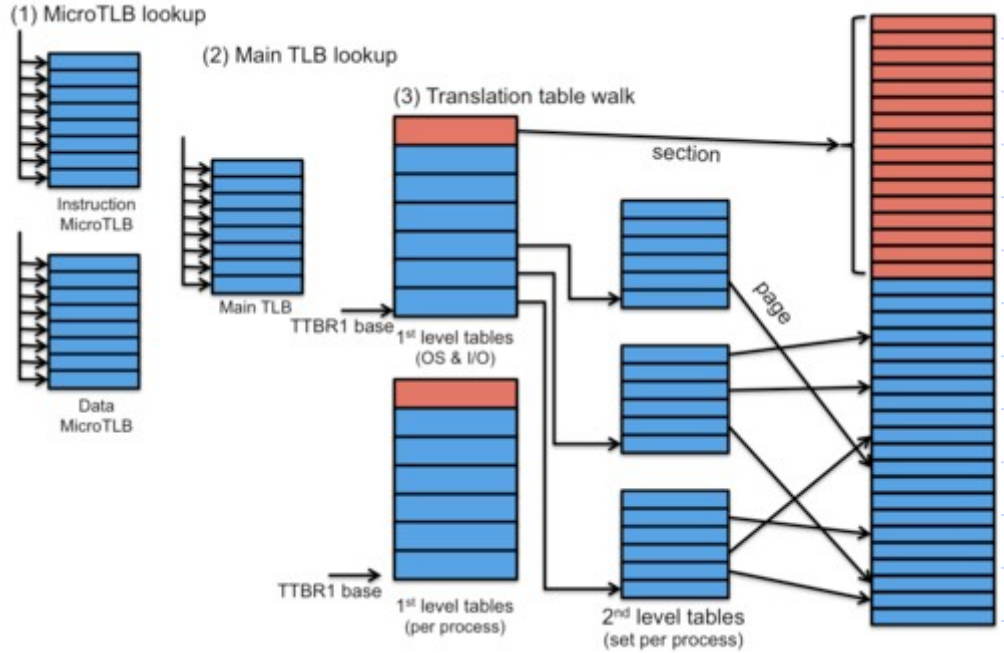
Why loops? Because repeatedly accessing the same or nearby data can increase the TLB hit rate if we do it right.

Modern TLBs

ARM – RISC CPU with 64-bit address space

Two levels of TLBs:

- Fast 32-entry fully-associative Micro TLBs for Data and Instruction
- Slower back-up Main TLB with 8 fully associative plus 64 set-associative (variable # of clock cycles for search)



Multi-level Paging

- 2- and 3-level paging work

Needed when we have large address spaces and we want to fit each page table in a page

Each level of paging does make a TLB miss more expensive.

Going Beyond Physical Memory

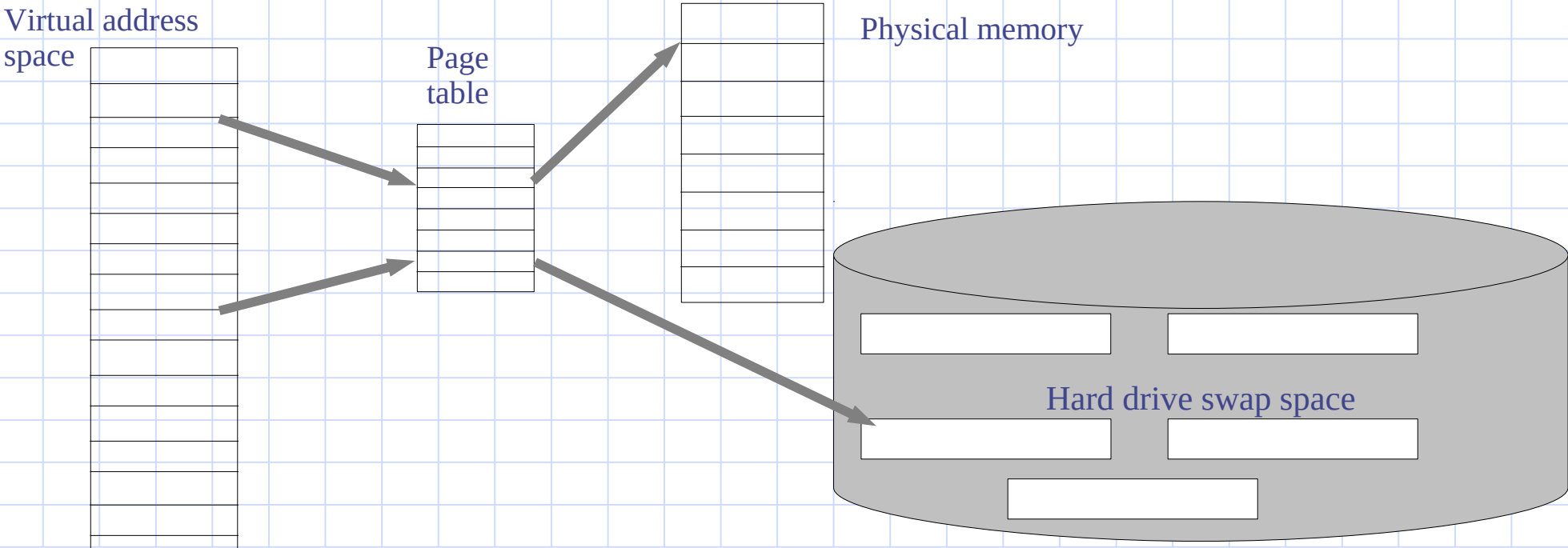
What if we need large address spaces?

What if the virtual address space is larger than physical memory?

How can the OS make use of a larger, slower device to transparently provide the illusion of a large virtual address space?

Swap space!

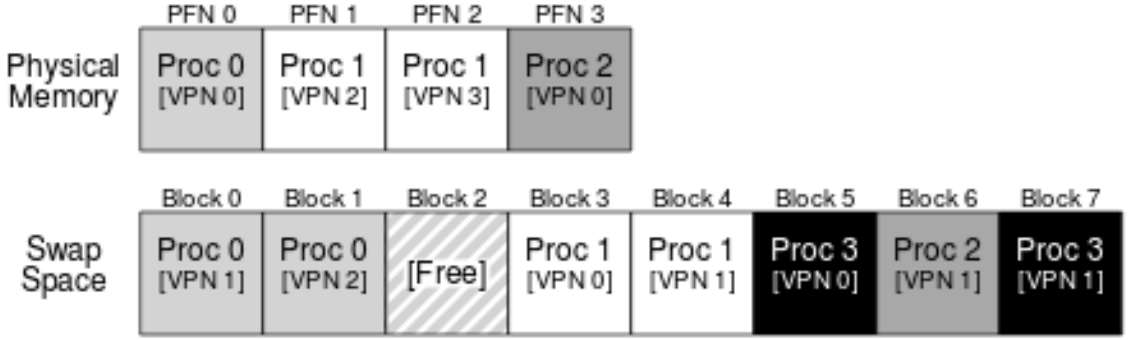
- Swap space is secondary (disk) memory used to store page frames when physical memory is used completely.



Swap space

Swap page frame **out** of physical memory into disk when we need a free page frame.

Swap page frame **in**to memory when data in that page frame is needed.



The Present bit

- Recall Memory Accesses and Translation Lookaside Buffer:

  TLB hit → we have the address translation step, fetch data from physical memory

  TLB miss → get Page Table data → if “Present bit” set, fetch data from memory

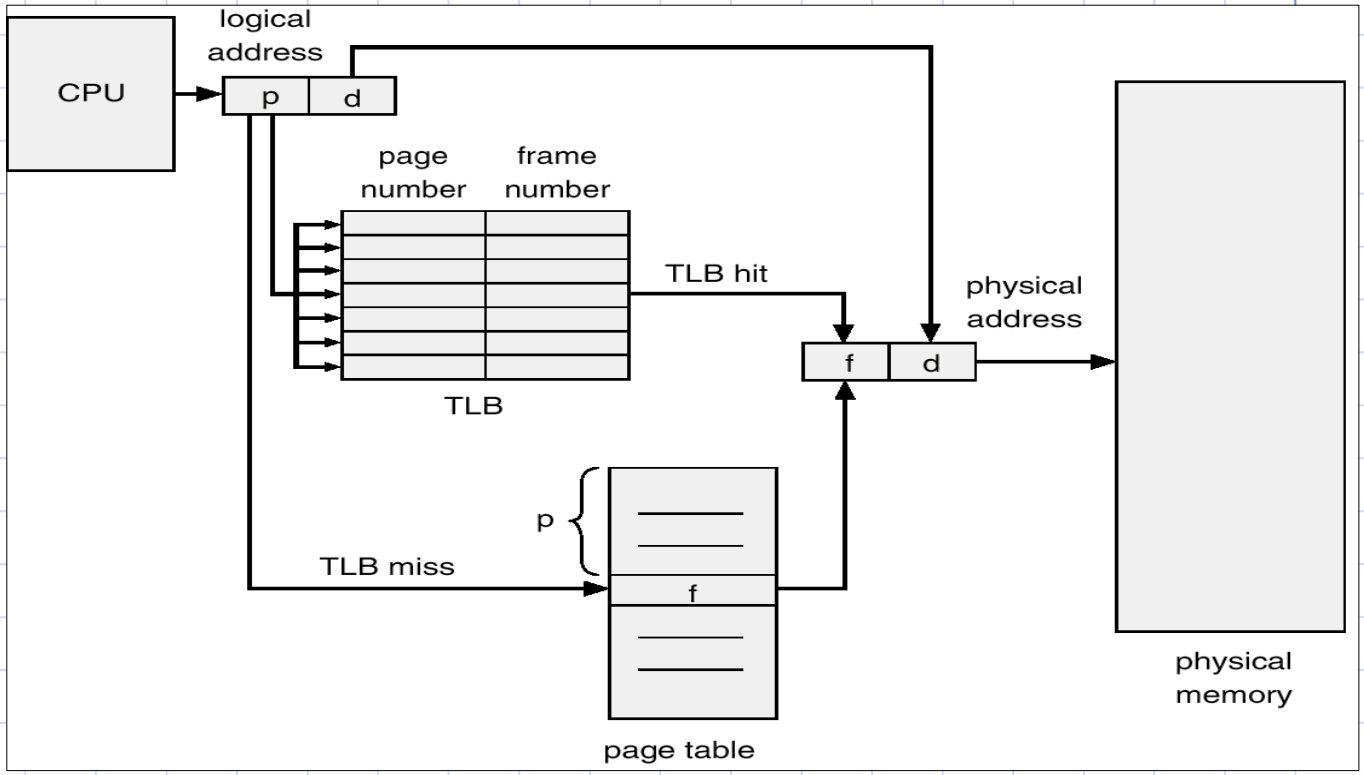
  If the Present bit is not set, need to load page frame from disk into memory → Page Fault

The Present bit

TLB hit → we have the address translation step, fetch data from physical memory

TLB miss → get Page Table data → if “Present bit” set, fetch data from memory

If the Present bit is not set, need to load page frame from disk into memory → Page Fault



Page Fault

A Page Fault is an OS event that signals that a frame is not in main memory and needs to be swapped in.

When a page fault happens:

- the process is blocked (why?)
- the page-fault-handler runs to figure out how to swap the frame in
- If main memory is full, need to swap out a frame first.

Page Faults in Windows

- 32-bit Windows system use these interrupts numbers

Interrupt Number	Exception
0	Divide Error
1	Debug (Single Step)
2	Non-Maskable Interrupt (NMI)
3	Breakpoint
4	Overflow
5	Bounds Check
6	Invalid Opcode
7	NPX Not Available
8	Double Fault
9	NPX Segment Overrun
10	Invalid Task State Segment (TSS)
11	Segment Not Present
12	Stack Fault
13	General Protection
14	Page Fault
15	Intel Reserved
16	Floating Point
17	Alignment Check
18	Machine Check
19	SIMD Floating Point

Page Replacement Policy

If main memory is full and a frame has to be swapped in, we need to swap out a frame first.

To do this, we need to chose the frame in physical memory that is least likely to be needed soon

→ page replacement policy

A bad policy can result in excessive swapping → programs run at the speed of the disk rather than physical memory

Page Replacement Policy

- In order to avoid big delays, the OS does not wait for memory to be completely full before swapping frames out

- The OS may pre-emptively swap out least recently used frames to keep memory free.

- The OS may also cluster or group a number of frames and swap them out en masse to reduce disc seek times.