Virtual Memory Wrap-Up
CSE 351 Winter 2018

Instructor:

Mark Wyse

Teaching Assistants:
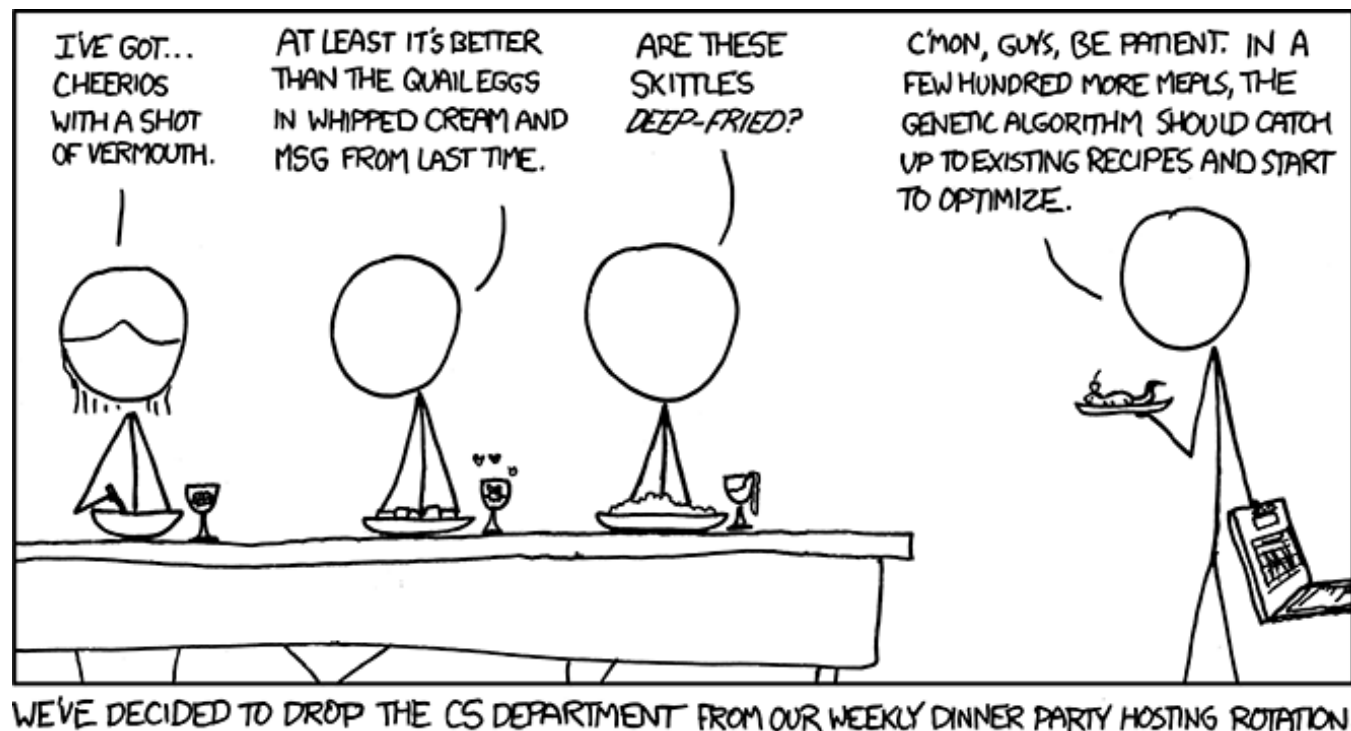
Kevin Bi

Parker DeWilde

Emily Furst

Sarah House

Waylon Huang

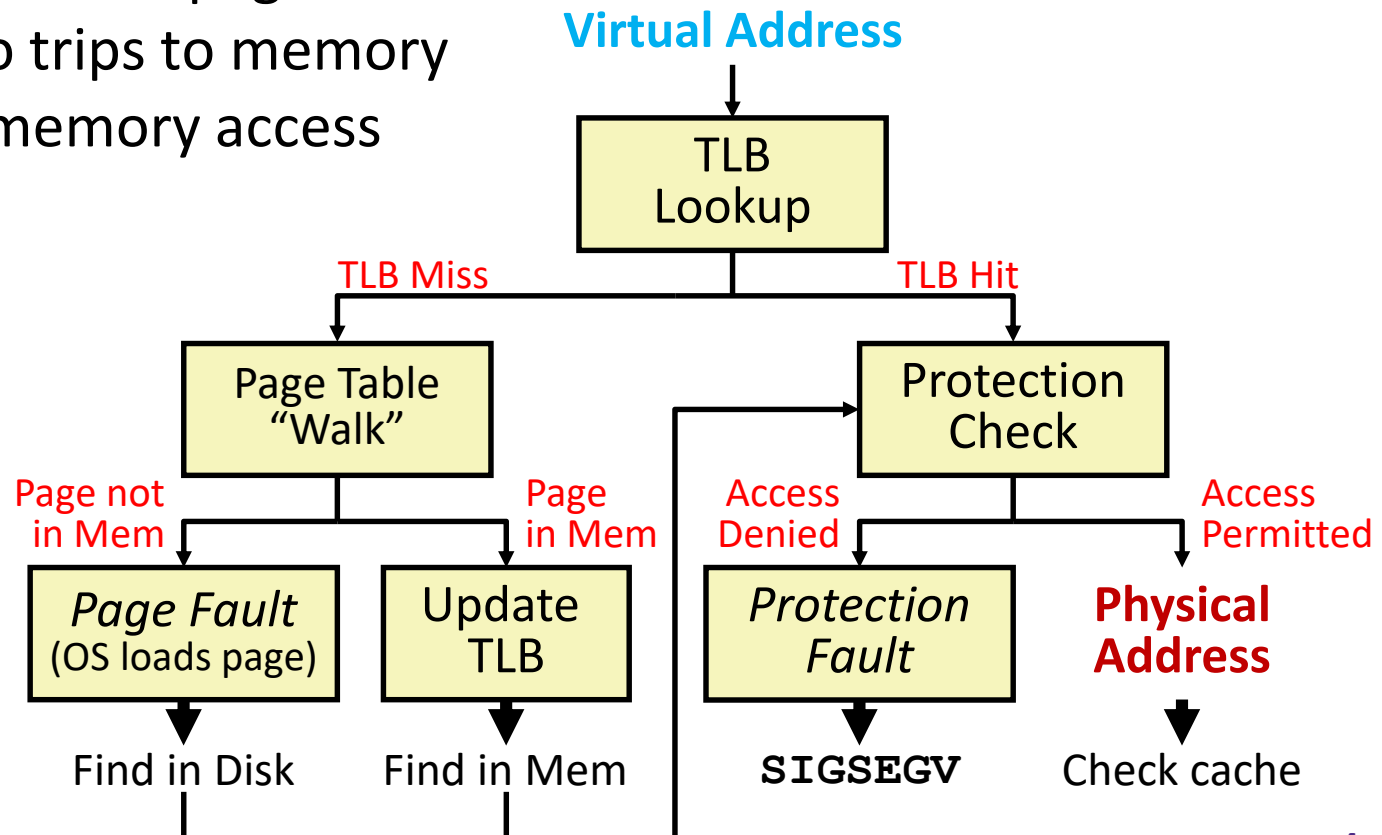Vinny Palaniappan

https://xkcd.com/720/

Administrative

❖ Lab 4 due tonight!

  ▪ By 11:59 pm!

❖ Homework 5 due next Wednesday (3/7)

❖ Lab 5 released tomorrow

  ▪ Due Saturday, March 10 @ 11:59 pm

❖ Final Exam

  ▪ Wednesday, March 14, 2:30 – 4:20 pm, KNE 110

  ▪ Topics: everything

Quick Review

❖ What do Page Tables map?

❖ Where are Page Tables located?

❖ How many Page Tables are there?

❖ Can your process tell if a page fault has occurred?

❖ True / False: Virtual Addresses that are contiguous will always be contiguous in physical memory
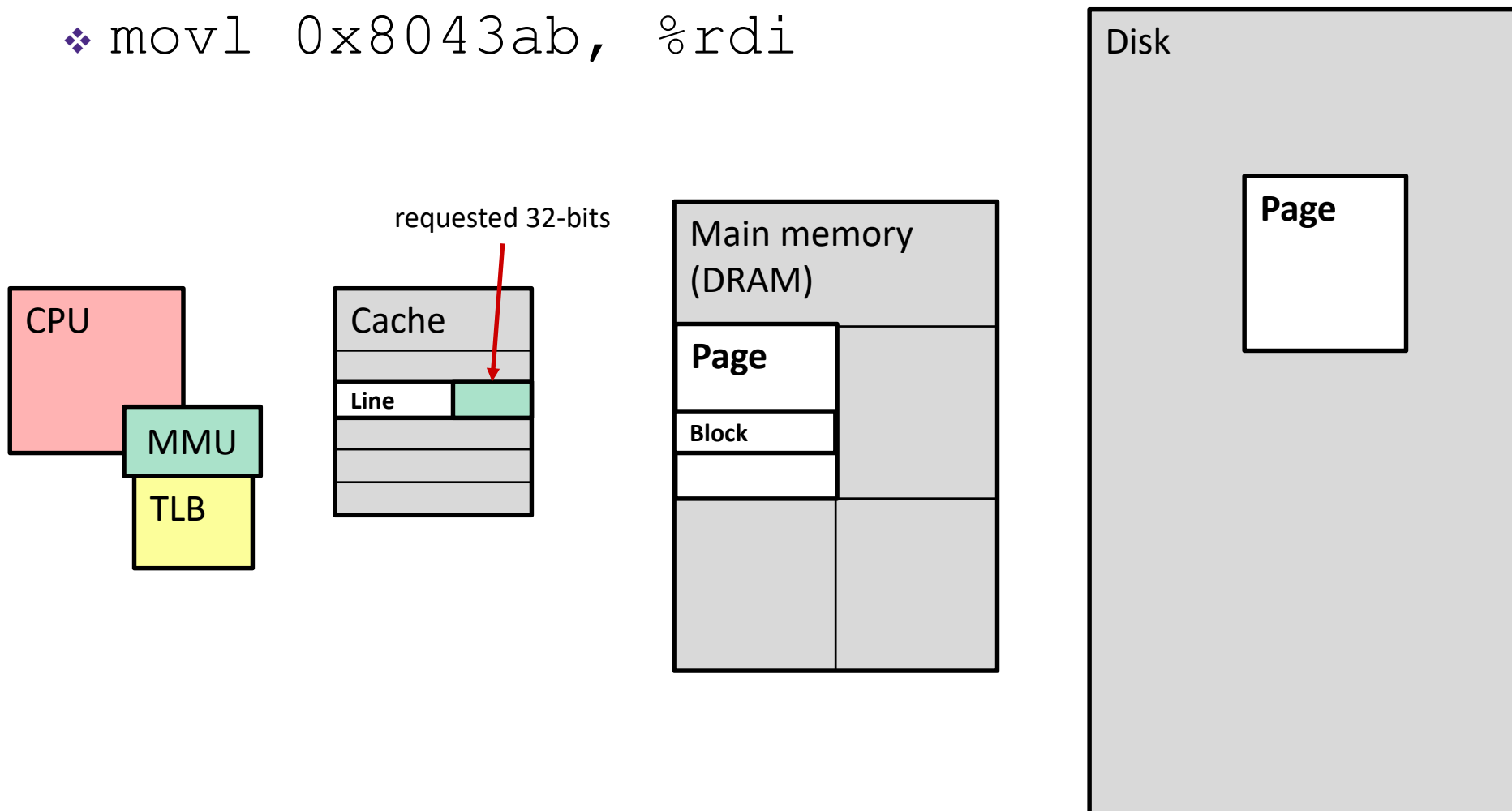
❖ TLB stands for _____ and stores _____

Address Translation

❖ VM is complicated, but also elegant and effective
  - Level of indirection to provide isolated memory & caching
  - TLB as a cache of page tables avoids two trips to memory for every memory access

Virtual Address

| TLB Lookup |

TLB Miss   TLB Hit

| Page Table "Walk" | | Protection Check |

Page not in Mem   Page in Mem   Access Denied   Access Permitted

| *Page Fault* (OS loads page) | | Update TLB | | *Protection Fault* | | **Physical Address** |

Find in Disk   Find in Mem   `SIGSEGV`   Check cache

Memory Overview

❖ `movl 0x8043ab, %rdi`

requested 32-bits

CPU

MMU

TLB

Cache

Line

Main memory (DRAM)

Page

Block

Disk

Page

Context Switching Revisited

❖ What needs to happen when the CPU switches processes?

- Registers:
  - Save state of old process, load state of new process
  - Including the Page Table Base Register (PTBR)

- Memory:
  - Nothing to do! Pages for processes already exist in memory/disk and protected from each other

- TLB:
  - *Invalidate* all entries in TLB – mapping is for old process' VAs

- Cache:
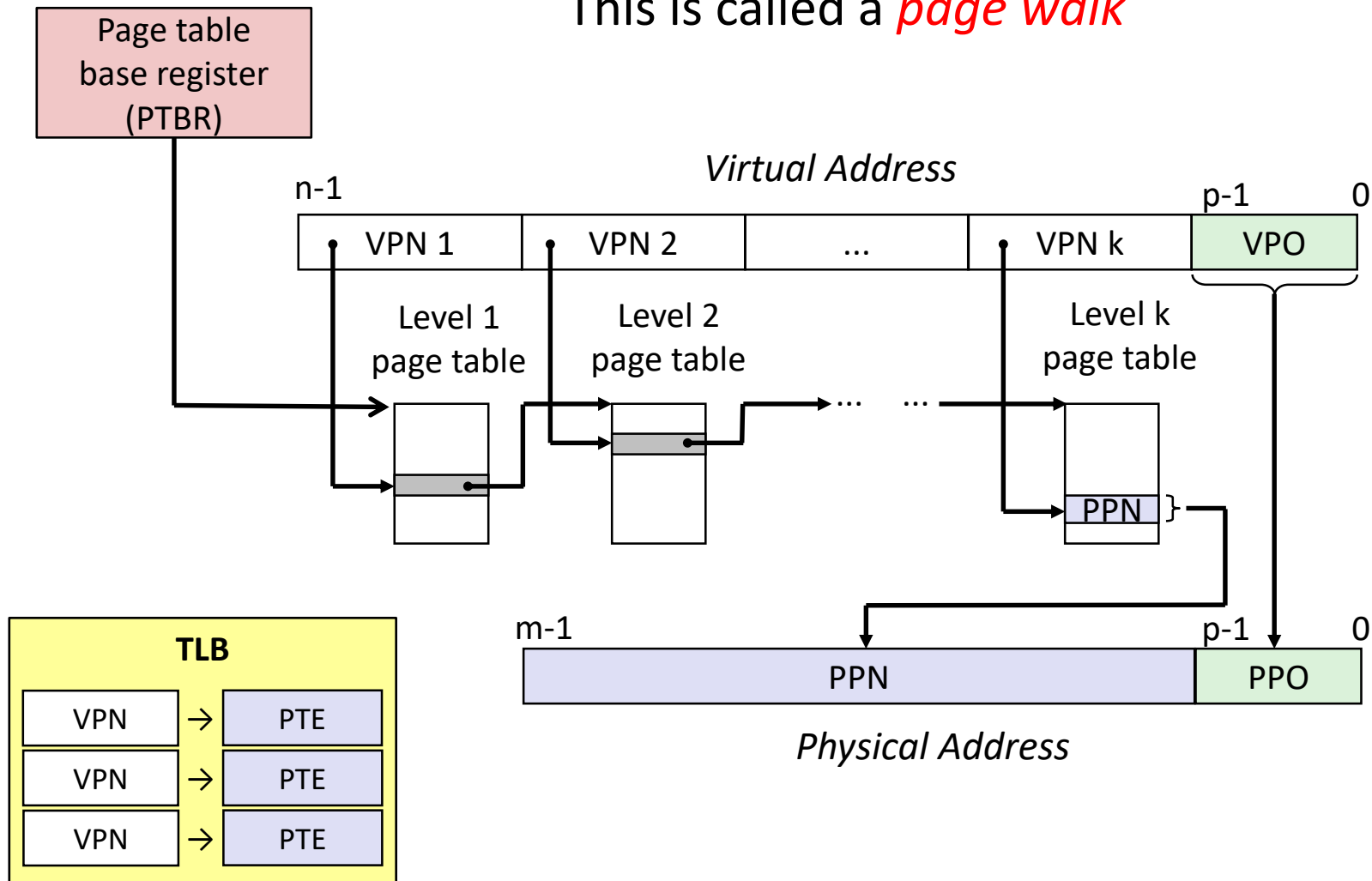 - Can leave alone because storing based on PAs – good for shared data

Page Table Reality

❖ Just one issue… the numbers don't work out for the story so far!

❖ The problem is the page table for each process:
- Suppose 64-bit VAs, 8 KiB pages, 8 GiB physical memory
- How many page table entries is that?

- About how long is each PTE?

- **Moral:** Cannot use this naïve implementation of the virtual→physical page mapping – it's *way* too big

A Solution: Multi-level Page Tables

This is extra (non-testable) material

This is called a *page walk*

Page table base register (PTBR)

Virtual Address

n-1			p-1	0
VPN 1	VPN 2	...	VPN k	VPO

Level 1 page table   Level 2 page table   Level k page table

PPN

m-1	p-1	0
PPN	PPO	

Physical Address

TLB

VPN	→	PTE
VPN	→	PTE
VPN	→	PTE

Multi-level Page Tables

This is extra (non-testable) material

- ❖ A tree of depth k where each node at depth i has up to 2^j children if part i of the VPN has j bits

- ❖ Hardware for multi-level page tables inherently more complicated
  - But it's a necessary complexity – 1-level does not fit

- ❖ Why it works: Most subtrees are not used at all, so they are never created and definitely aren't in physical memory
  - Parts created can be evicted from cache/memory when not being used
  - Each node can have a size of ~1-100KB

- ❖ But now for a k-level page table, a TLB miss requires $k + 1$ cache/memory accesses
  - Fine so long as TLB misses are rare – motivates larger TLBs

Practice VM Question

❖ Our system has the following properties

- 1 MiB of physical address space

- 4 GiB of virtual address space

- 32 KiB page size

- 4-entry fully associative TLB with LRU replacement

a) Fill in the following blanks:

_____	Total entries in page table	_____	Minimum bit-width of PTBR
_____	TLBT bits	_____	Max # of valid entries in a page table

Practice VM Question

❖ One process uses a page-aligned *square* matrix `mat[]` of 32-bit integers in the code shown below:

```
#define MAT_SIZE = 2048
for(int i=0; i<MAT_SIZE; i++)
    mat[i*(MAT_SIZE+1)] = i;
```

b) What is the largest stride (in bytes) between successive memory accesses (in the VA space)?

Practice VM Question

❖ One process uses a page-aligned *square* matrix `mat[]` of 32-bit integers in the code shown below:

```
#define MAT_SIZE = 2048
for(int i=0; i<MAT_SIZE; i++)
    mat[i*(MAT_SIZE+1)] = i;
```

c) What are the following hit rates for the *first* execution of the for loop?

_____ TLB Hit Rate   _____ Page Table Hit Rate

Quick Review Answers

❖ What do Page Tables map?
  ▪ VPN → PPN or disk address

❖ Where are Page Tables located?
  ▪ In physical memory

❖ How many Page Tables are there?
  ▪ One per process

❖ Can your program tell if a page fault has occurred?
  ▪ Nope, but it has to wait a long time

❖ What is thrashing?
  ▪ Constantly paging out and paging in

❖ True / False: Virtual Addresses that are contiguous will always be contiguous in physical memory
  ▪ Could fall across a page boundary

❖ TLB stands for Translation Lookaside Buffer and stores page table entries