# W13.2 - Virtual Memory (Part 2)

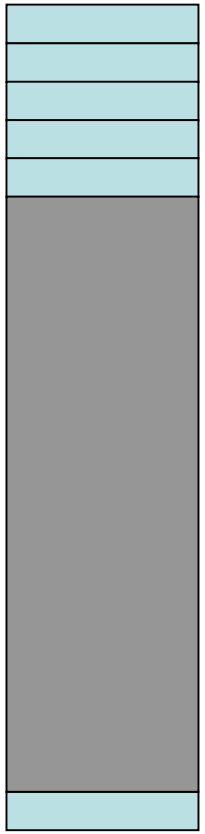
# Acknowledgment


- The material for this topic is primarily assembled from four sources:
  - OSPP Textbook, Anderson and Dahlin, Chapter 8, http://ospp.cs.washington.edu/slides.html
  - Computer Systems: A Programmer's Perspective, 3/E, Bryant and O'Hallaron, Chapter 9, http://www.cs.cmu.edu/afs/cs/academic/class/15213-f15/www/schedule.html
  - High Performance Computer Architecture, Milos Prvulovic, http://www.cc.gatech.edu/~milos/Teaching/CS6290F07/
  - Select slide material of my presentation on Virtual Memory to Intel Colleagues, Jan 1995 (no link ☺)
- The slide material is attributed to the sources (templates). Minor changes to source slides are made where necessary for further explanation

# What's Next?

- ❑The enormity of single-level page table for today's 64-bit machines! Practical considerations and solutions
- ❑Choosing Page Sizes
- ❑Timing of Page Hits and Page Faults and what can be done to improve access times
- ❑Handling Page Faults

# Simple Page Table

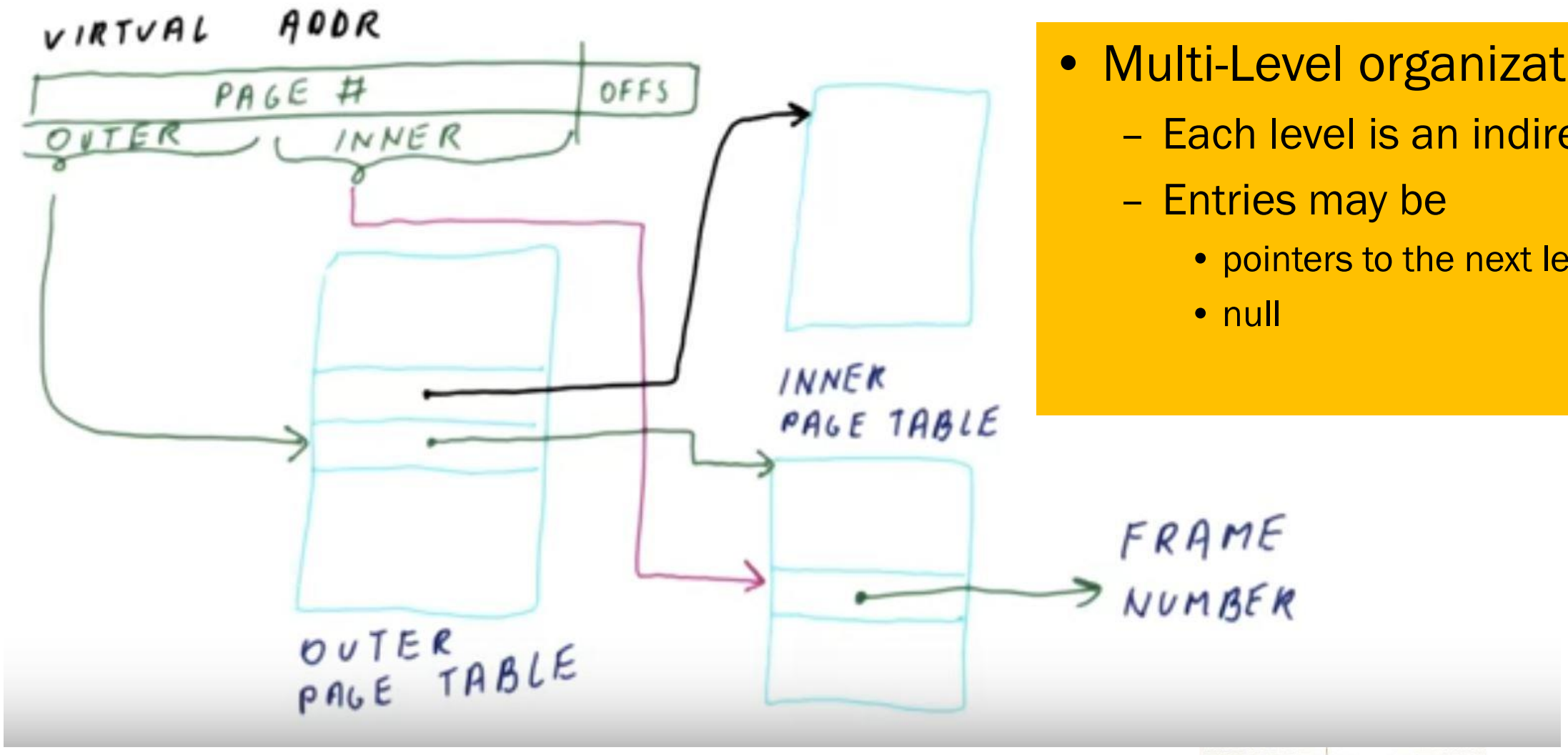


One entry per page

- Flat organization
  - One entry per page
  - Entry contains page frame number or indicates page is on disk or invalid
  - Also meta-data (e.g., permissions, dirtiness, etc.)

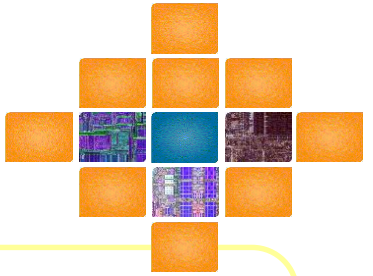
Question: How large could this flat page table become (assume 4KB Pages)?
→ 32-bit virtual address space?
→ 64-bit virtual address space?

# Multi-Level Page Tables



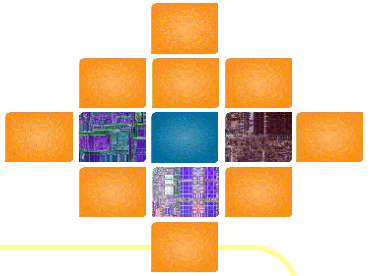

- Multi-Level organization
  - Each level is an indirection
  - Entries may be
    - pointers to the next level, or
    - null

# Choosing a Page Size



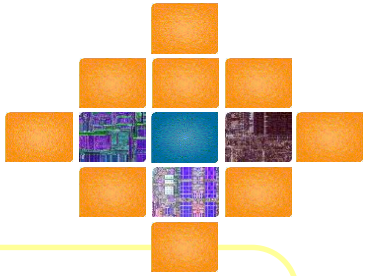
- Page size inversely proportional to page table overhead
- Large page size permits more efficient transfer to/from disk
  - vs. many small transfers
  - like downloading from Internet
- Small page leads to less fragmentation
  - Big page likely to have more bytes unused

# CPU Memory Access



- Program deals with virtual addresses
  - "Load R1 = [R2]" *//meaning load the content of memory location pointed to by register R2//*
- On memory load instruction
  1. Compute virtual address ([R2])
  2. Compute virtual page number
  3. Compute physical address of page table entry
  4. Read page table entry
  5. Compute physical address
  6. Access Cache (and sometimes memory)

# Impact on Performance?

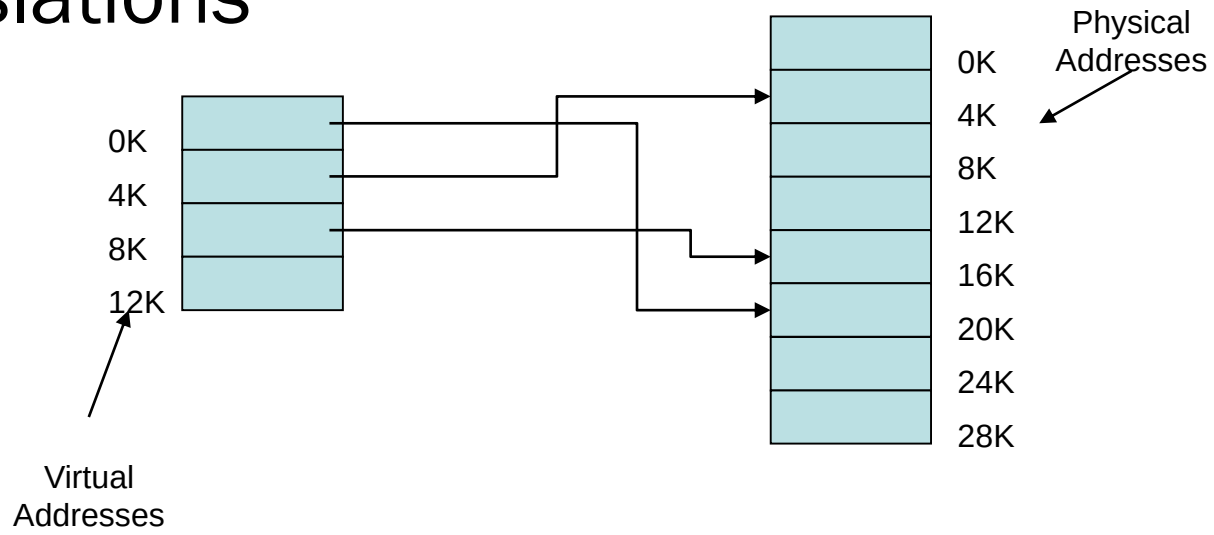


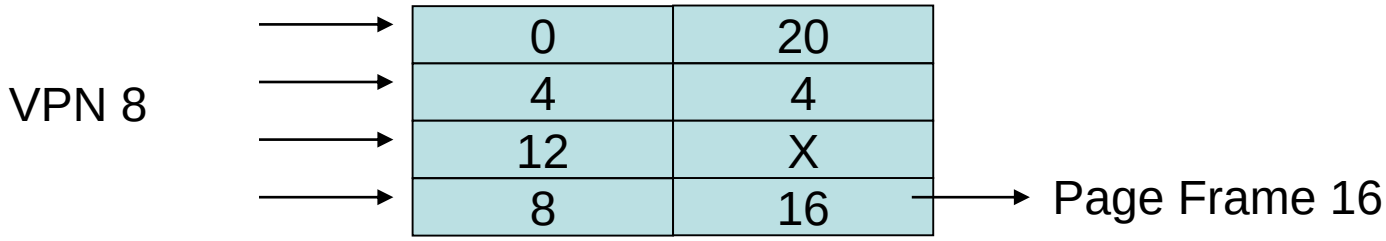
- Every time you load/store, the CPU must perform two (or more) accesses!
- Even worse, every *fetch* requires translation of the PC!

- Observation:
  - Once a virtual page is mapped into a physical page, it'll likely stay put for quite some time

# Idea: Caching!

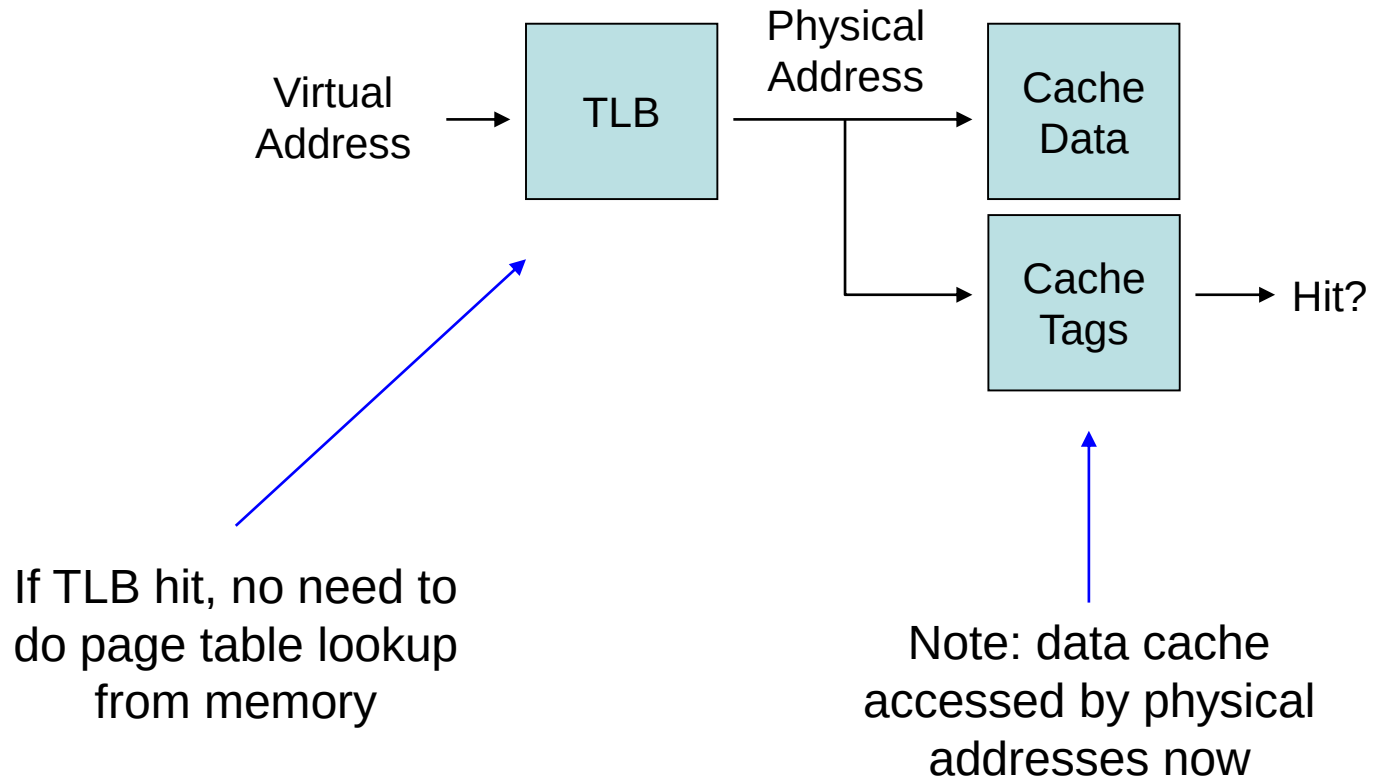
- Not caching of data, but caching of translations



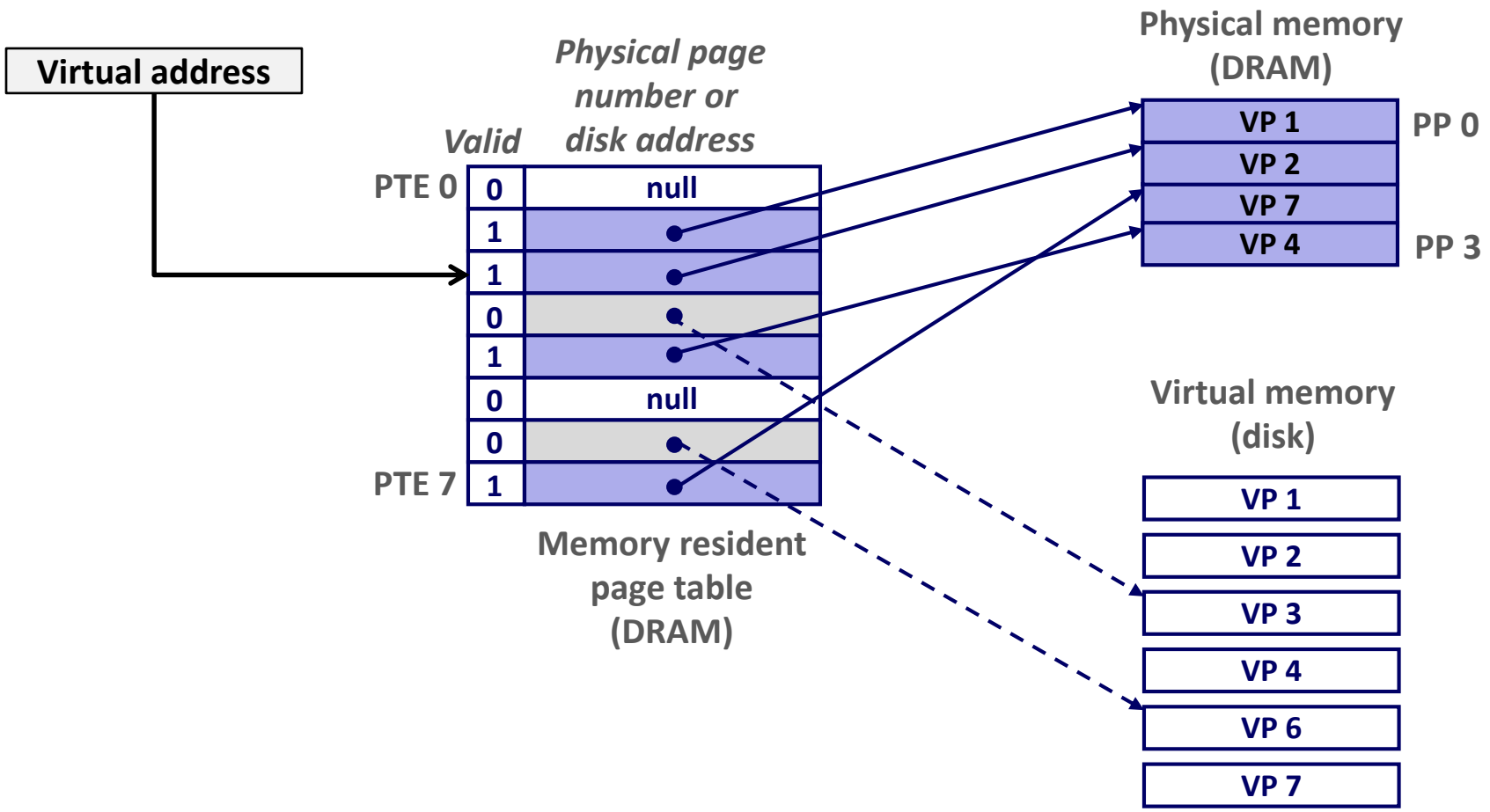
# Translation Cache: TLB

- TLB = Translation Look-aside Buffer



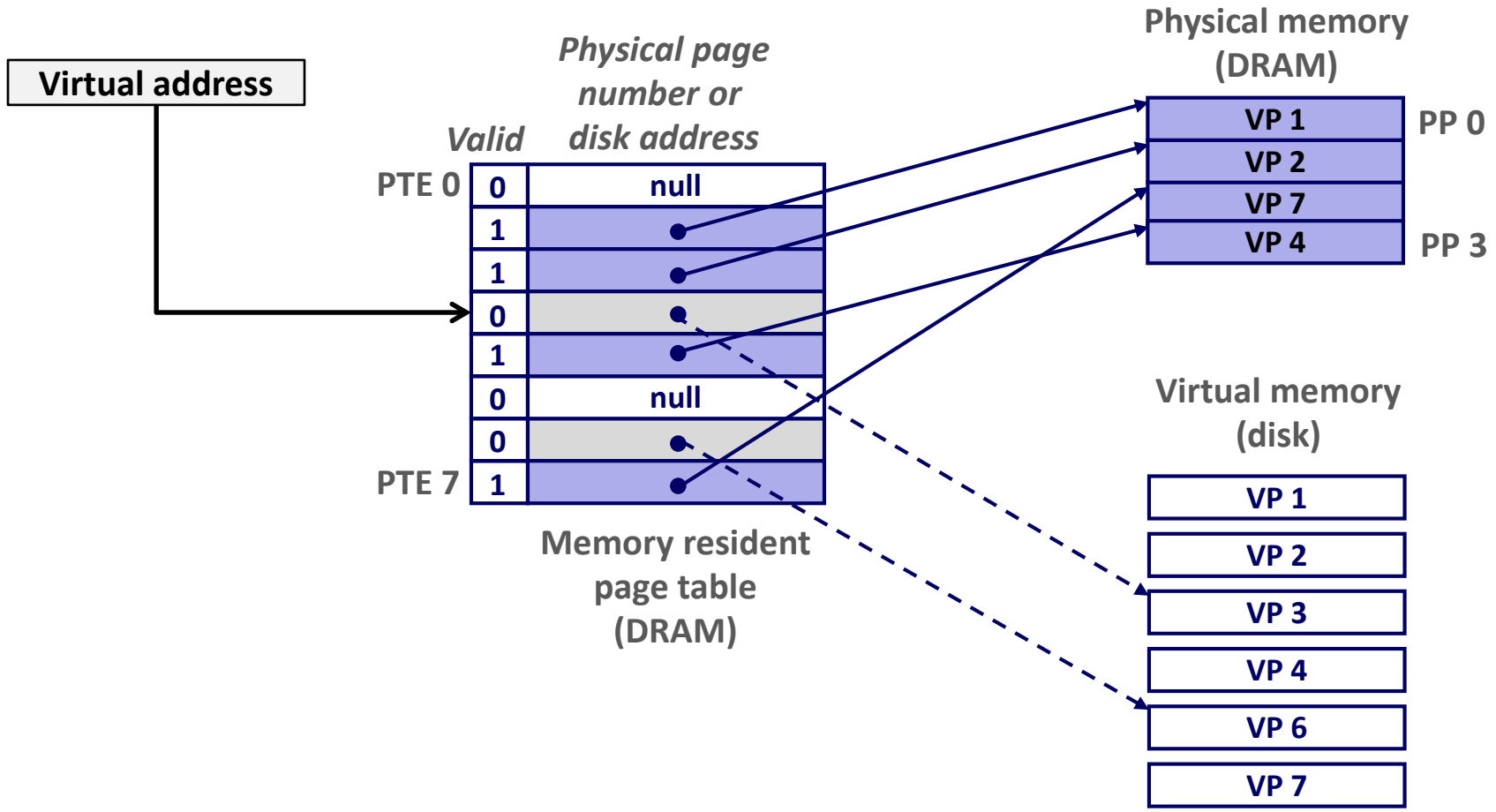
# Page Hit

- ***Page hit:*** **reference to VM word that is in physical memory**



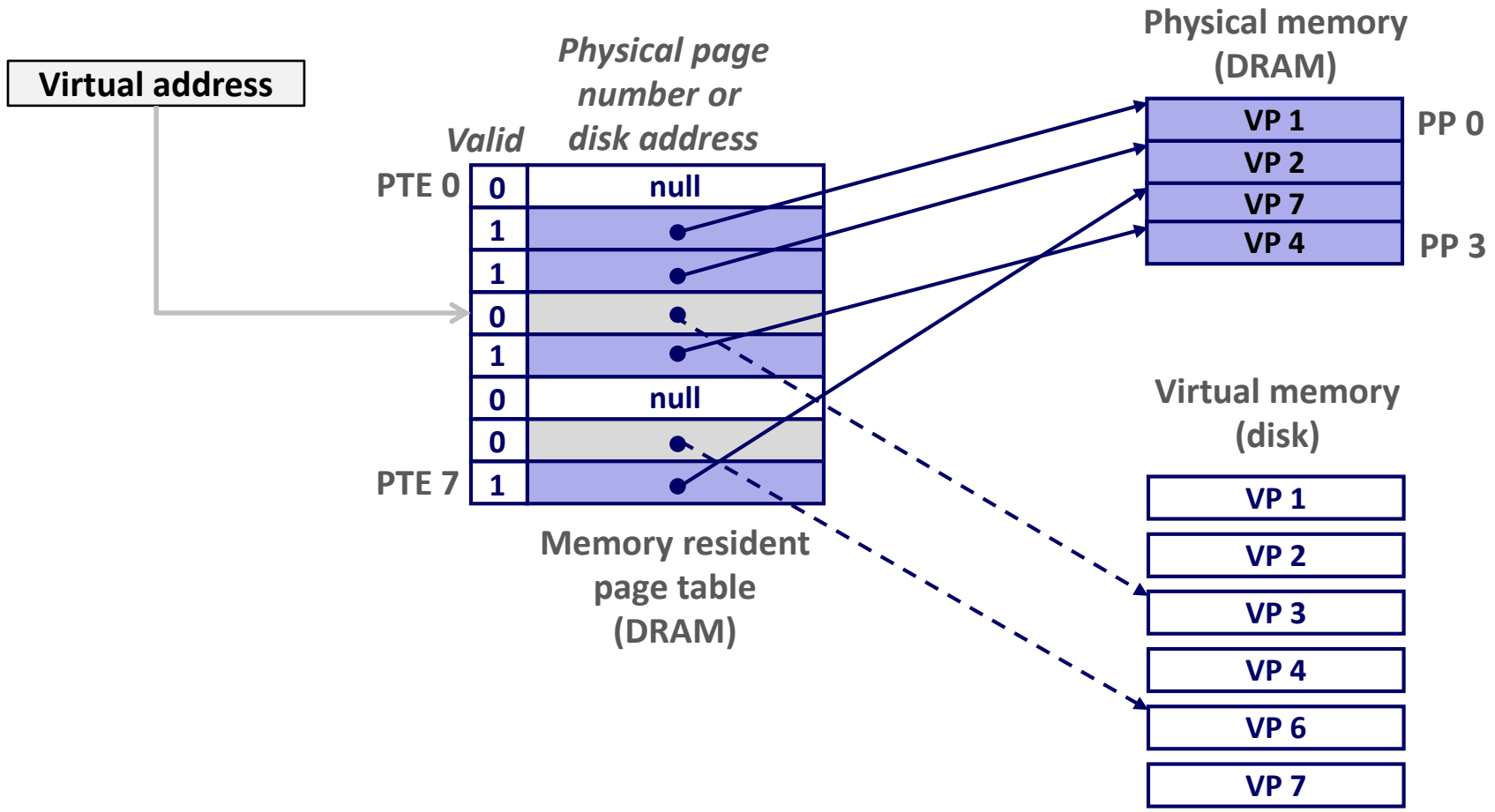
# Page Fault

- ***Page fault:*** **reference to VM word that is not in physical memory (DRAM cache miss)**



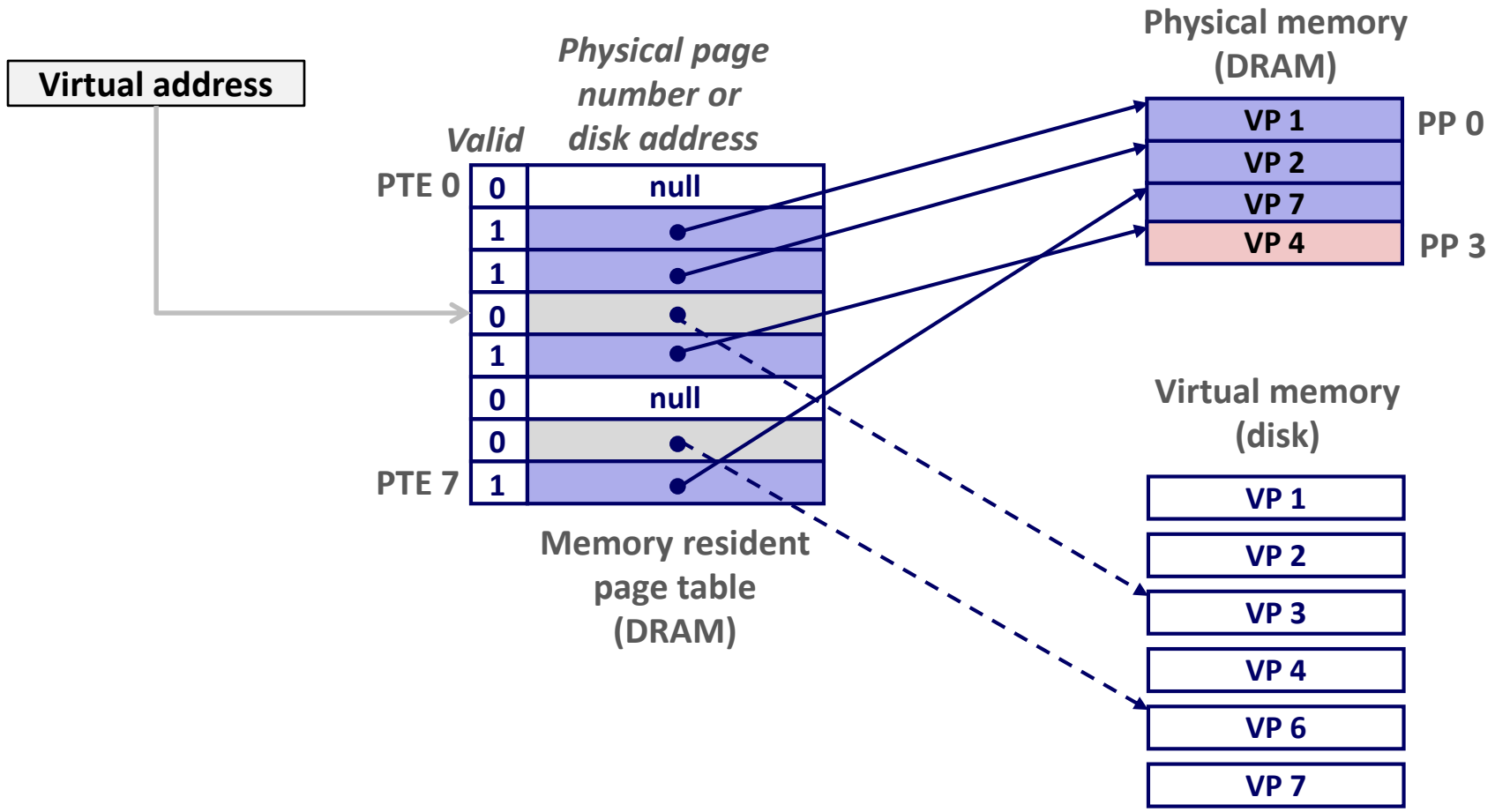
# Handling Page Fault

- Page miss causes page fault (an exception)



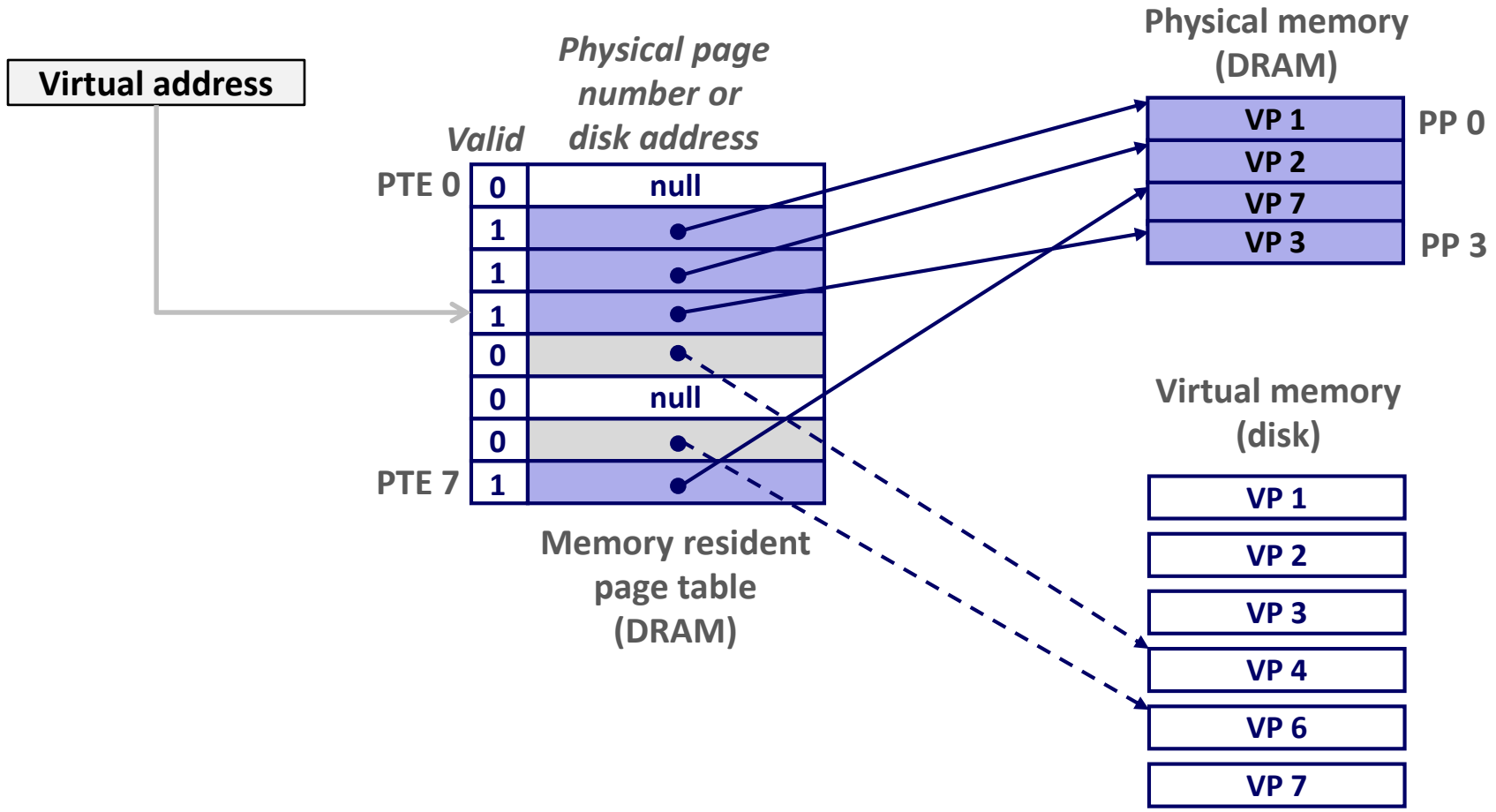
# Handling Page Fault

- Page miss causes page fault (an exception)
- Page fault handler selects a victim to be evicted (here VP 4)



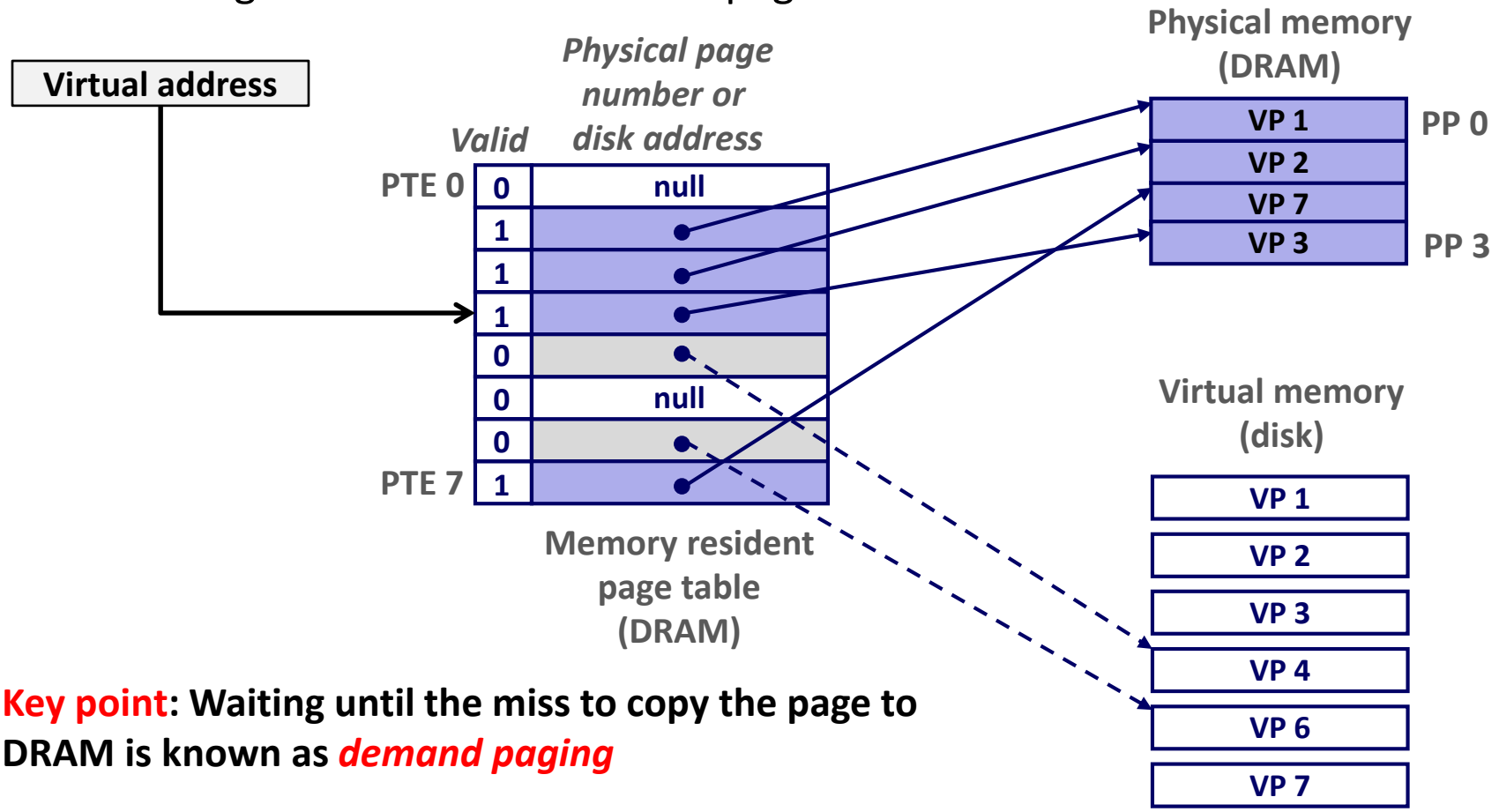
# Handling Page Fault

- Page miss causes page fault (an exception)
- Page fault handler selects a victim to be evicted (here VP 4)



# Handling Page Fault

- Page miss causes page fault (an exception)
- Page fault handler selects a victim to be evicted (here VP 4)
- Offending instruction is restarted: page hit!




**Key point: Waiting until the miss to copy the page to DRAM is known as *demand paging***

# Summary

- Virtual Memory (programmer's view) versus Physical Memory (processor's view)
- Cross-referencing of virtual to physical memory
  - Done in the form of pages
  - Done by virtue of page table
- A flat page table for today's 64-bit machines is ginormous! Hence we implement multi-level page tables
- Choosing Page Sizes pros and cons
- Timing of Page Hits and Page Faults and what can be done to improve access times (Translation Lookaside Buffer Cache)
- Handling Page Faults