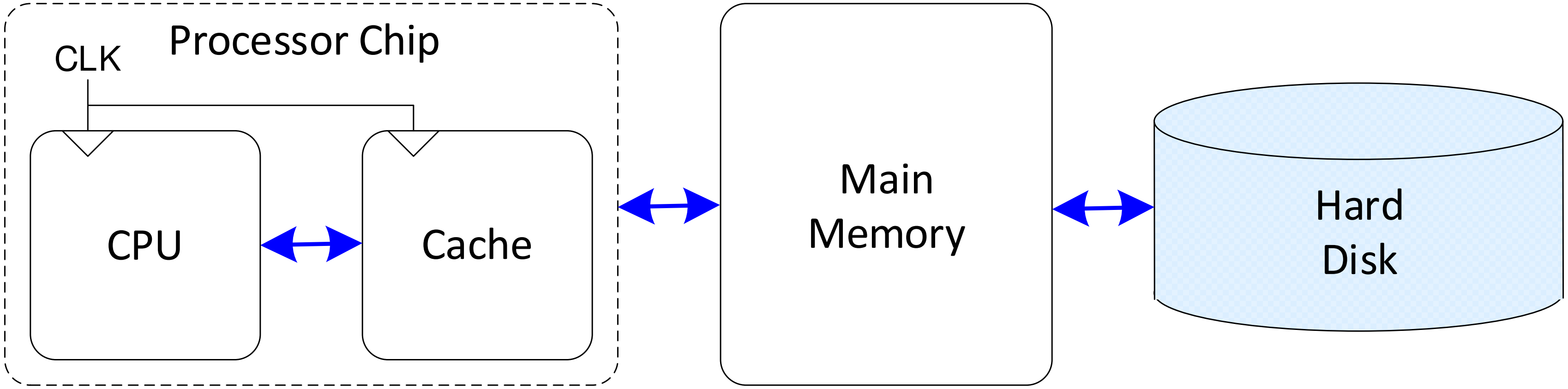
Chapter 7: Microarchitecture

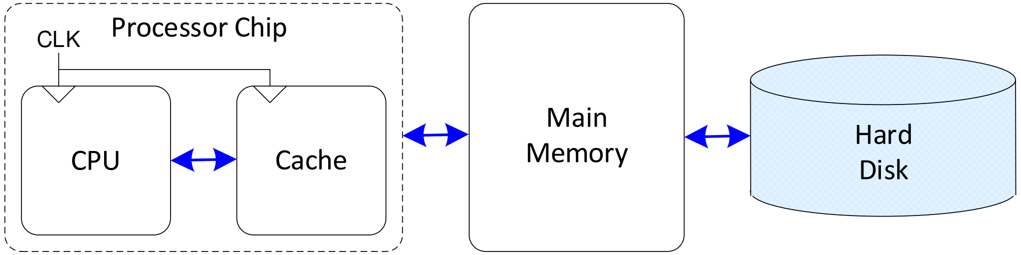
# Virtual Memory

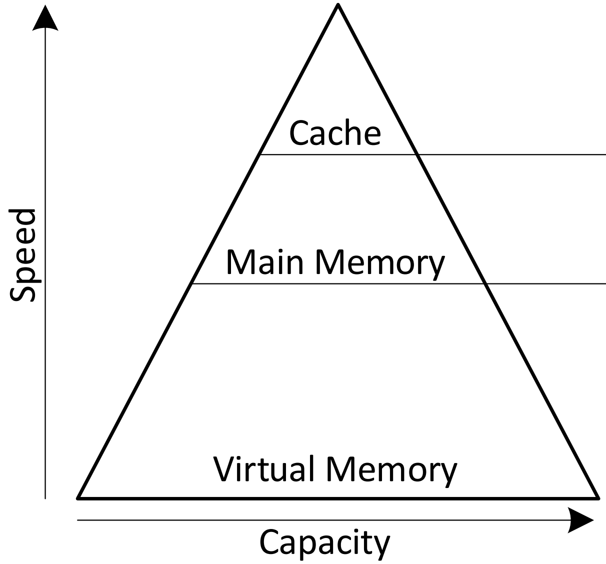
# Virtual Memory

- Gives the illusion of bigger memory
- Main memory (DRAM) acts as cache for hard disk



# Memory Hierarchy



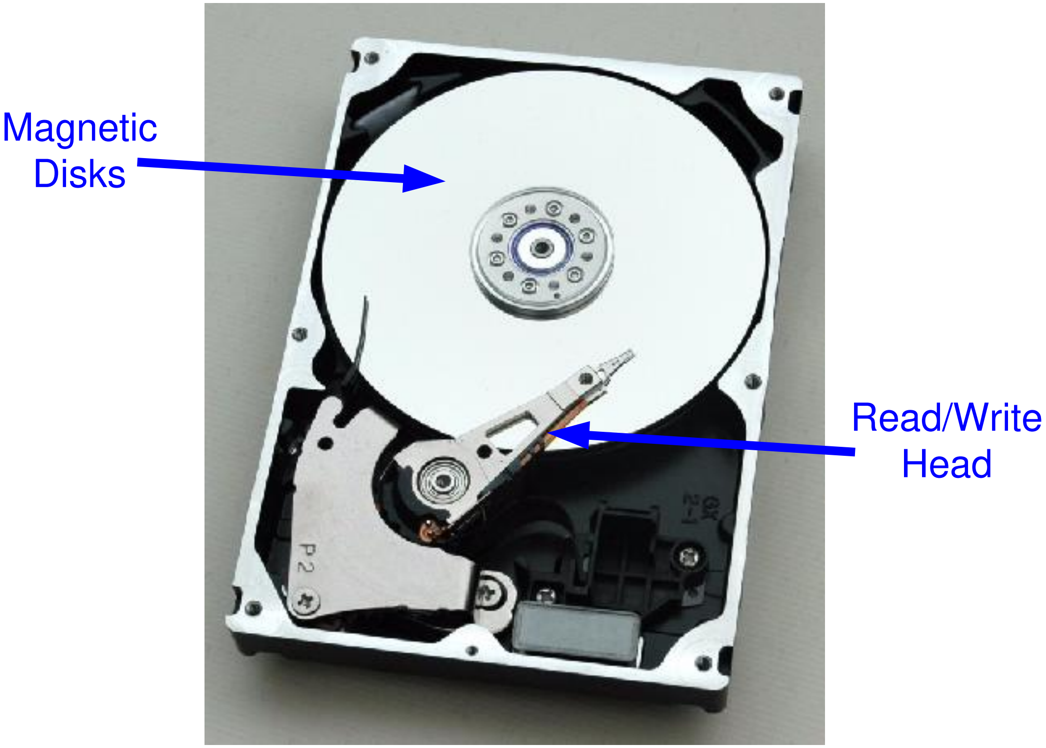


| Technology | Price / GB | Access Time (ns) | Bandwidth (GB/s) |
|---|---|---|---|
| SRAM | $100 | 0.2 - 3 | 100+ |
| DRAM | $3 | 10 - 50 | 30 |
| SSD<br>HDD | $0.10<br>$0.03 | 20,000<br>5,000,000 | 0.05 - 3<br>0.001 - 0.1 |

- **Physical Memory:** DRAM (Main Memory)
- **Virtual Memory:** Hard drive
  - Slow, Large, Cheap

# Memory Hierarchy

## Hard Disk Drive



Magnetic Disks

Read/Write Head

Takes milliseconds to *seek* correct location on disk

## Solid State Drive

*Arshane88 / CC BY-SA 4.0 / Wikimedia Commons*

# Virtual Memory

- **Virtual addresses**
  - Programs use virtual addresses
  - Entire virtual address space stored on a hard drive
  - Subset of virtual address data in DRAM
  - CPU translates virtual addresses into ***physical addresses*** (DRAM addresses)
  - Data not in DRAM fetched from hard drive

# Virtual Memory

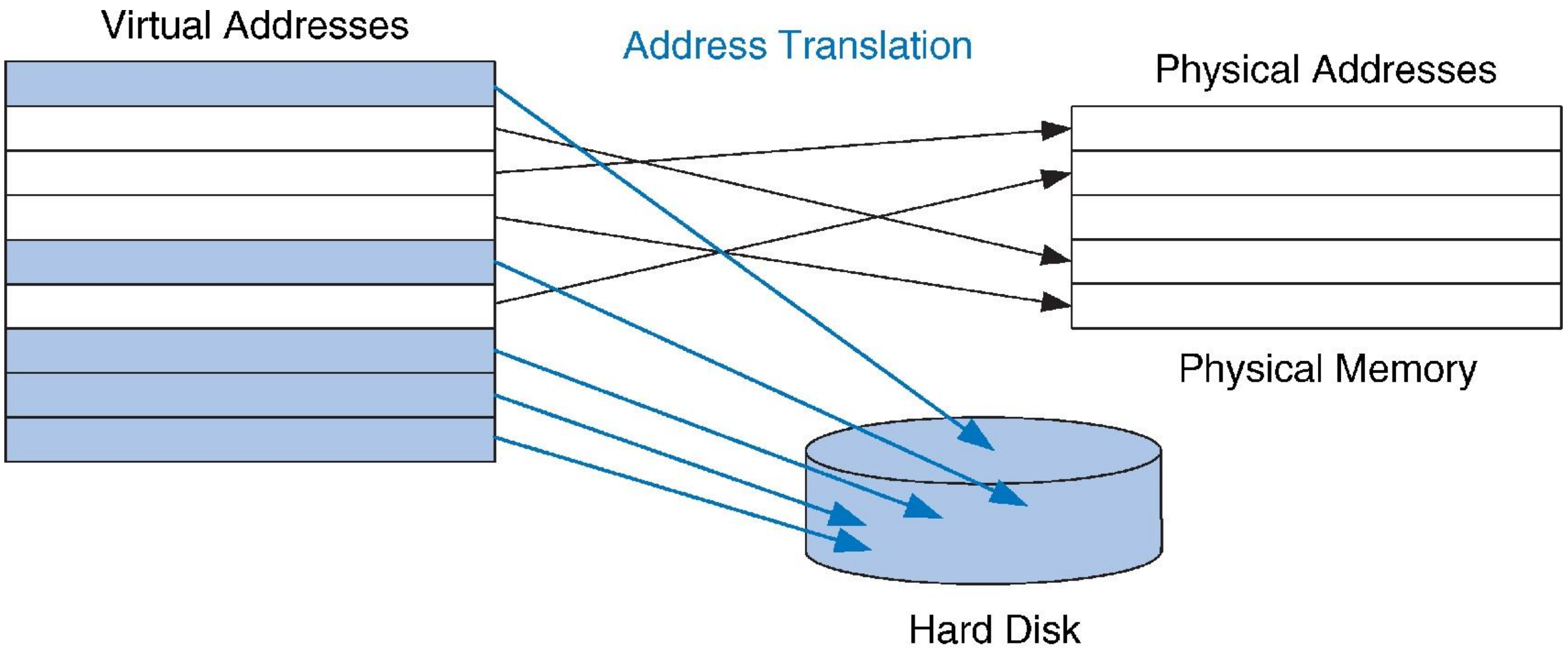
| Cache | Virtual Memory |
|---|---|
| Block | Page |
| Block Size | Page Size |
| Block Offset | Page Offset |
| Miss | Page Fault |
| Tag | Virtual Page Number |

**Physical memory acts as cache for virtual memory**

# Virtual Memory Definitions

- **Page size:** amount of memory transferred from hard disk to DRAM at once
- **Address translation:** determining physical address from virtual address
- **Page table:** lookup table used to translate virtual addresses to physical addresses

# Virtual Memory Definitions



© 2007 Elsevier, Inc. All rights reserved

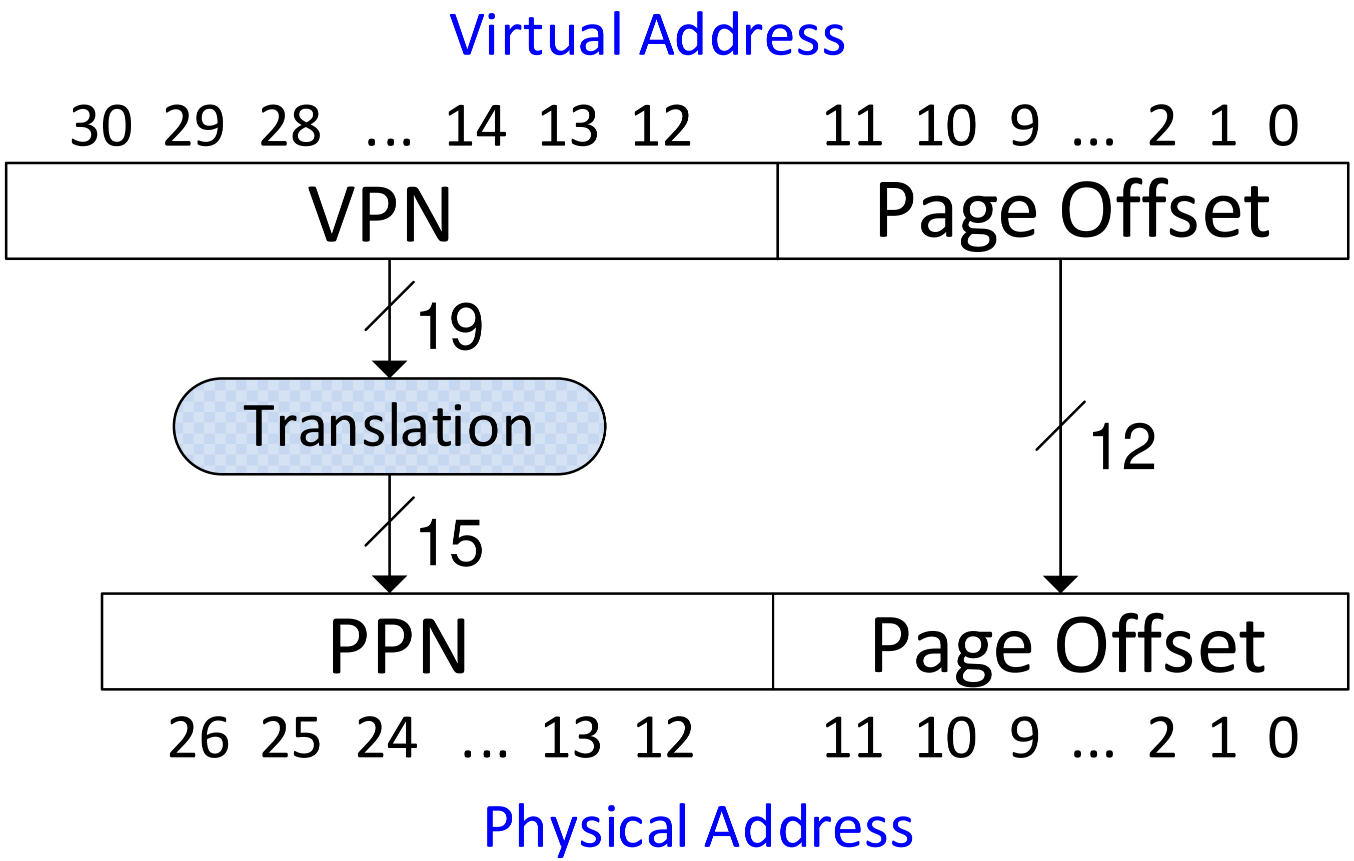
**Most accesses hit** in physical memory

But programs have the **large capacity** of virtual memory

Chapter 7: Microarchitecture

# Address Translation

# Address Translation

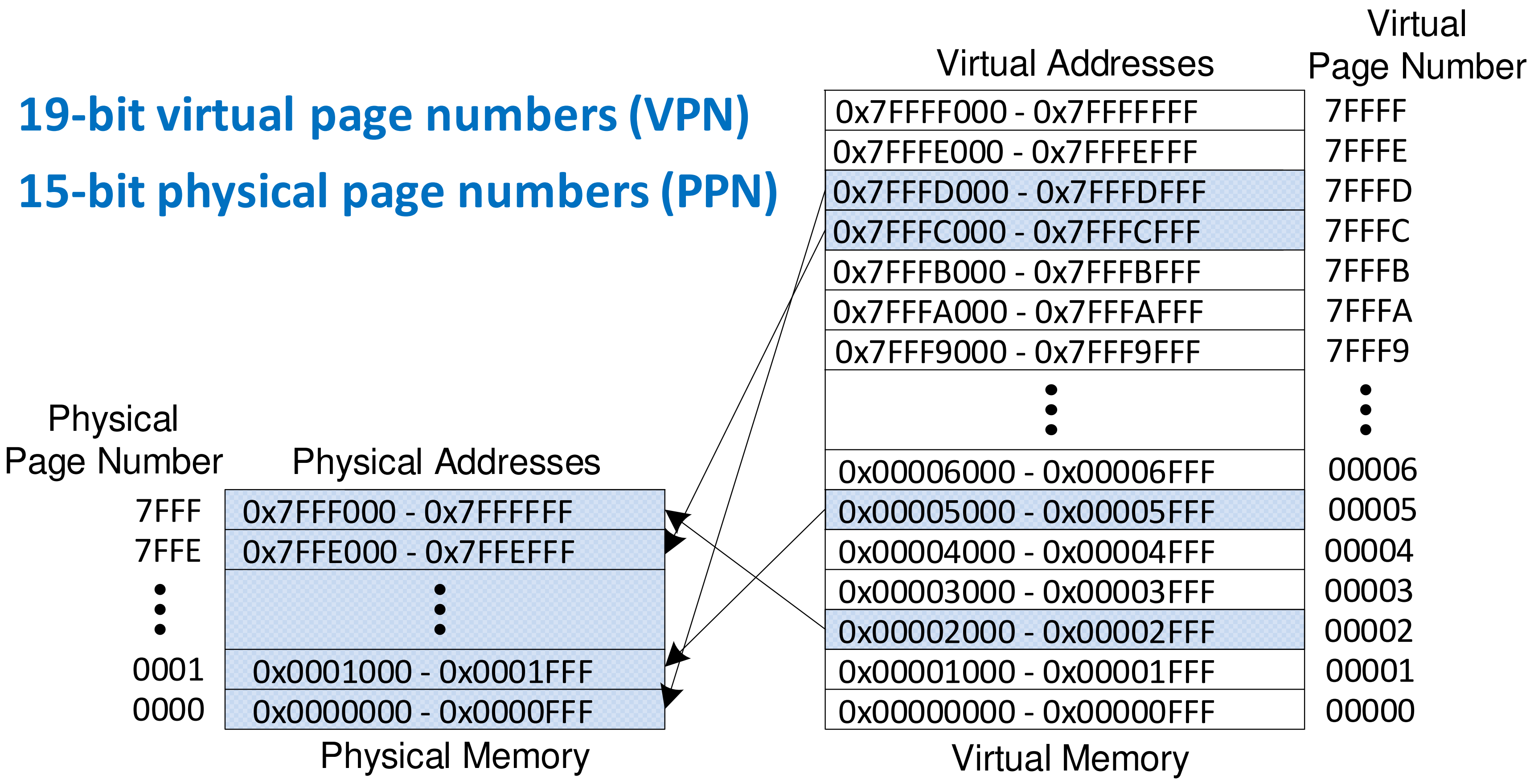


# Virtual Memory Example

- **System:**
  - Virtual memory size: 2 GB = $2^{31}$ bytes
  - Physical memory size: 128 MB = $2^{27}$ bytes
  - Page size: 4 KB = $2^{12}$ bytes

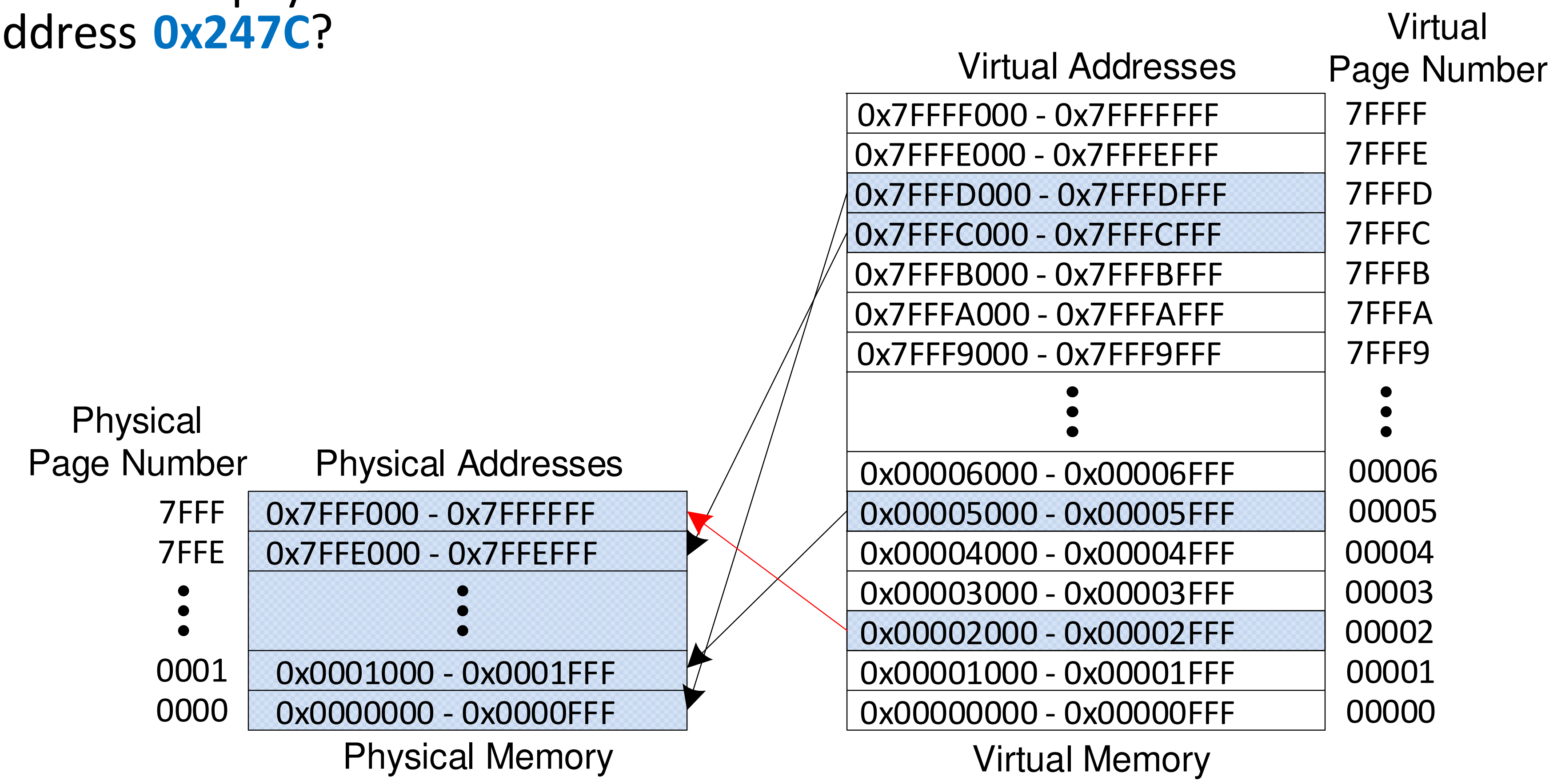
# Virtual Memory Example

- **19-bit virtual page numbers (VPN)**
- **15-bit physical page numbers (PPN)**



# Virtual Memory Example

What is the physical address of virtual address **0x247C**?



Physical Memory

| Physical Page Number | Physical Addresses |
|---|---|
| 7FFF | 0x7FFF000 - 0x7FFFFFF |
| 7FFE | 0x7FFE000 - 0x7FFEFFF |
| ⋮ | ⋮ |
| 0001 | 0x0001000 - 0x0001FFF |
| 0000 | 0x0000000 - 0x0000FFF |

Virtual Memory

| Virtual Addresses | Virtual Page Number |
|---|---|
| 0x7FFFF000 - 0x7FFFFFFF | 7FFFF |
| 0x7FFFE000 - 0x7FFFEFFF | 7FFFE |
| 0x7FFFD000 - 0x7FFFDFFF | 7FFFD |
| 0x7FFFC000 - 0x7FFFCFFF | 7FFFC |
| 0x7FFFB000 - 0x7FFFBFFF | 7FFFB |
| 0x7FFFA000 - 0x7FFFAFFF | 7FFFA |
| 0x7FFF9000 - 0x7FFF9FFF | 7FFF9 |
| ⋮ | ⋮ |
| 0x00006000 - 0x00006FFF | 00006 |
| 0x00005000 - 0x00005FFF | 00005 |
| 0x00004000 - 0x00004FFF | 00004 |
| 0x00003000 - 0x00003FFF | 00003 |
| 0x00002000 - 0x00002FFF | 00002 |
| 0x00001000 - 0x00001FFF | 00001 |
| 0x00000000 - 0x00000FFF | 00000 |

Chapter 7: Microarchitecture

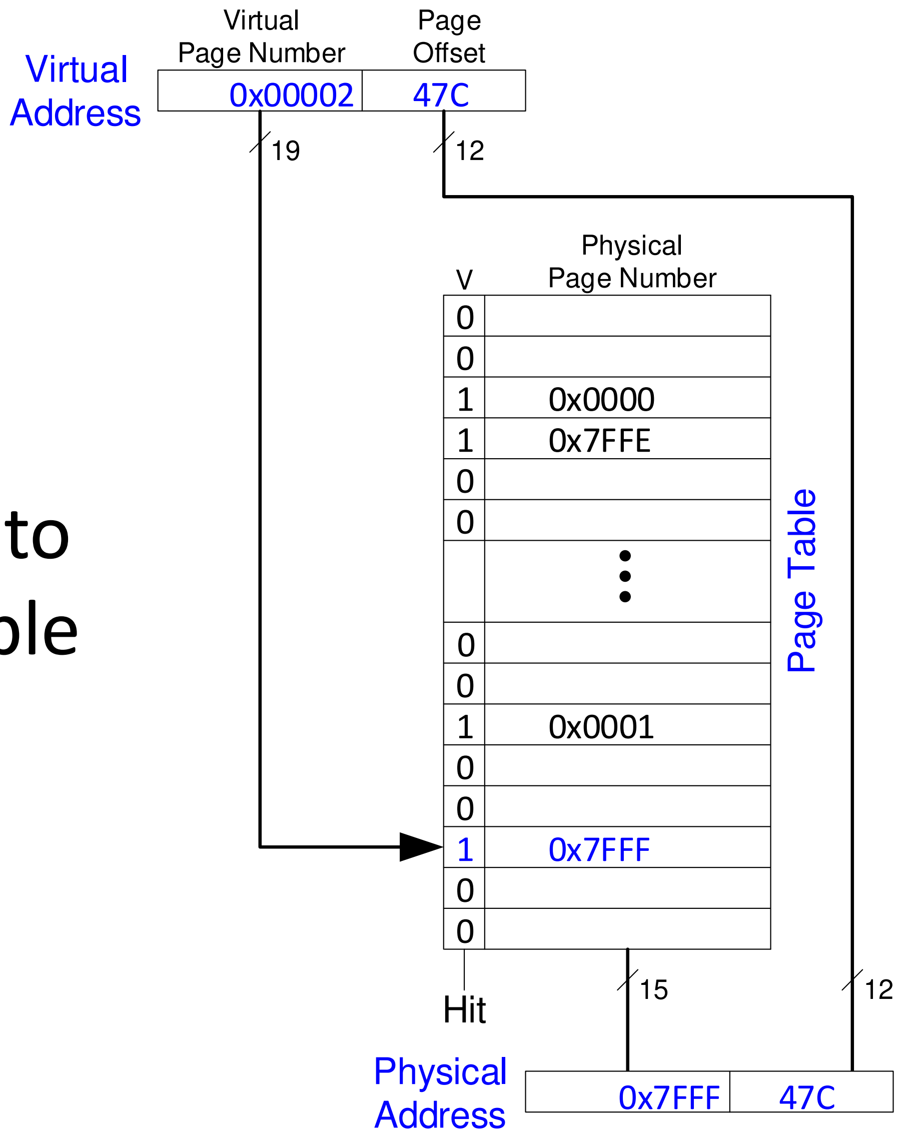
# Page Table

# How to Perform Translation

- **Page table**
  - Entry for each virtual page
  - Entry fields:
    - **Valid bit:** 1 if page in physical memory
    - **Physical page number:** where the page is located

# Page Table Example

**VPN** is **index** into page table



| V | Physical Page Number |
|---|---|
| 0 | |
| 0 | |
| 1 | 0x0000 |
| 1 | 0x7FFE |
| 0 | |
| 0 | |
| | ⋮ |
| 0 | |
| 0 | |
| 1 | 0x0001 |
| 0 | |
| 0 | |
| 1 | 0x7FFF |
| 0 | |
| 0 | |

# Page Table Example 1

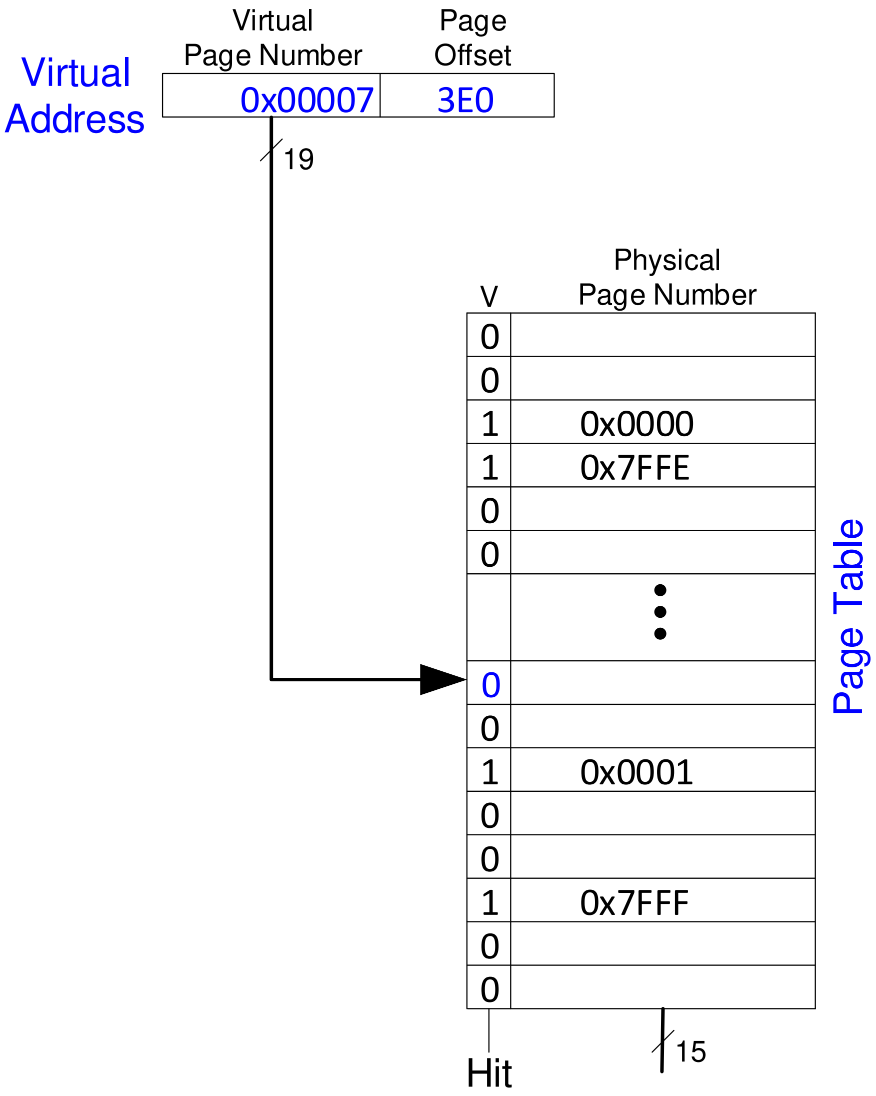
What is the physical address of virtual address **0x5F20**?

| V | Physical Page Number | Virtual Page Number |
|---|---|---|
| 0 |  | 7FFFF |
| 0 |  | 7FFFE |
| 1 | 0x0000 | 7FFFD |
| 1 | 0x7FFE | 7FFFC |
| 0 |  | 7FFFB |
| 0 |  | 7FFFA |
|  | ⋮ | ⋮ |
| 0 |  | 00007 |
| 0 |  | 00006 |
| 1 | 0x0001 | 00005 |
| 0 |  | 00004 |
| 0 |  | 00003 |
| 1 | 0x7FFF | 00002 |
| 0 |  | 00001 |
| 0 |  | 00000 |

Page Table

# Page Table Example 2

What is the physical address of virtual address **0x73E4**?



| V | Physical Page Number |
|---|---|
| 0 | |
| 0 | |
| 1 | 0x0000 |
| 1 | 0x7FFE |
| 0 | |
| 0 | |
| | ⋮ |
| 0 | |
| 0 | |
| 1 | 0x0001 |
| 0 | |
| 0 | |
| 1 | 0x7FFF |
| 0 | |
| 0 | |

# Page Table Challenges

- Page table is **large**
  - usually located in physical memory
- Load/store requires **2 main memory accesses**:
  - one for translation (page table read)
  - one to access data (after translation)
- Cuts memory performance in half
  - ***Unless we get clever…***

Chapter 7: Microarchitecture

# Translation Lookaside Buffer (TLB)

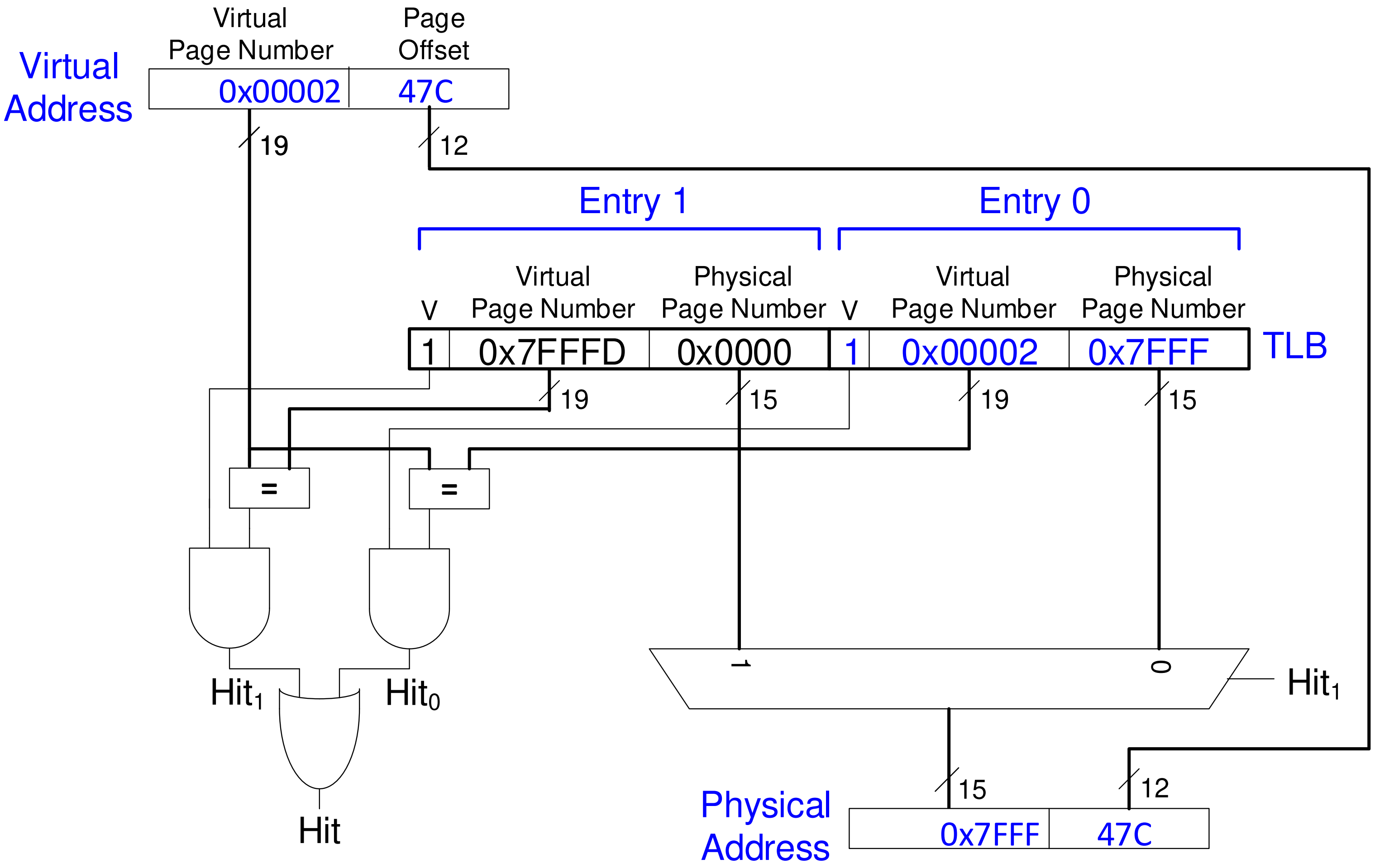
# Translation Lookaside Buffer (TLB)

- Small cache of most recent translations
- Reduces number of memory accesses for *most* loads/stores from 2 to **1**

# TLB

- **Page table accesses:** high temporal locality
  - Large page size, so consecutive loads/stores likely to access same page
- **TLB**
  - **Small:** accessed in < 1 cycle
  - Typically **16 - 512 entries**
  - **Fully associative**
  - **> 99%** hit rates typical
  - **Reduces number of memory accesses** for most loads/stores from 2 to 1

# Example: 2-entry TLB



Chapter 7: Microarchitecture

# Virtual Memory Summary

# Memory Protection

- **Multiple processes** (programs) run at once
- Each process has its **own page table**
- Each process can use **entire virtual address space**
- A process can only access a **subset of physical pages**: those mapped in its own page table

# Virtual Memory Summary

- Virtual memory increases **capacity**
- A subset of virtual pages in physical memory
- **Page table** maps virtual pages to physical pages – address translation
- A **TLB** speeds up address translation
- Different page tables for different programs provides **memory protection**

# About these Notes

**Digital Design and Computer Architecture Lecture Notes**

**© 2021 Sarah Harris and David Harris**

**These notes may be used and modified for educational and/or non-commercial purposes so long as the source is attributed.**