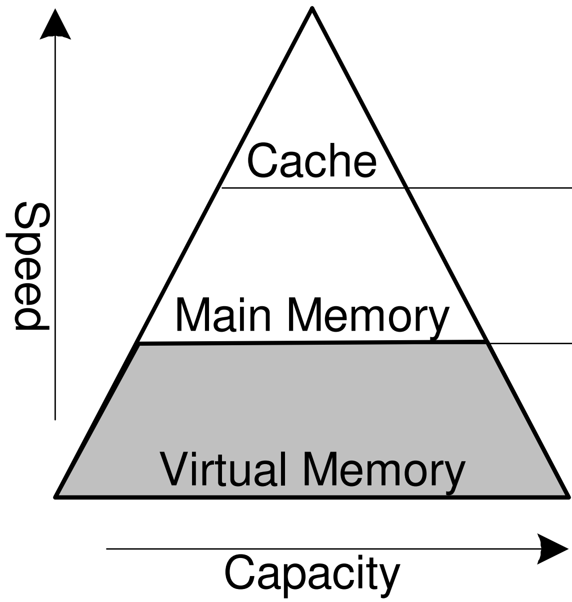
VIRTUAL MEMORY

Prof. Sebastian Eslava M.Sc., Ph.D.

Virtual Memory

- Gives the illusion of a bigger memory without the high cost of DRAM
- Main memory (DRAM) acts as cache for the hard disk

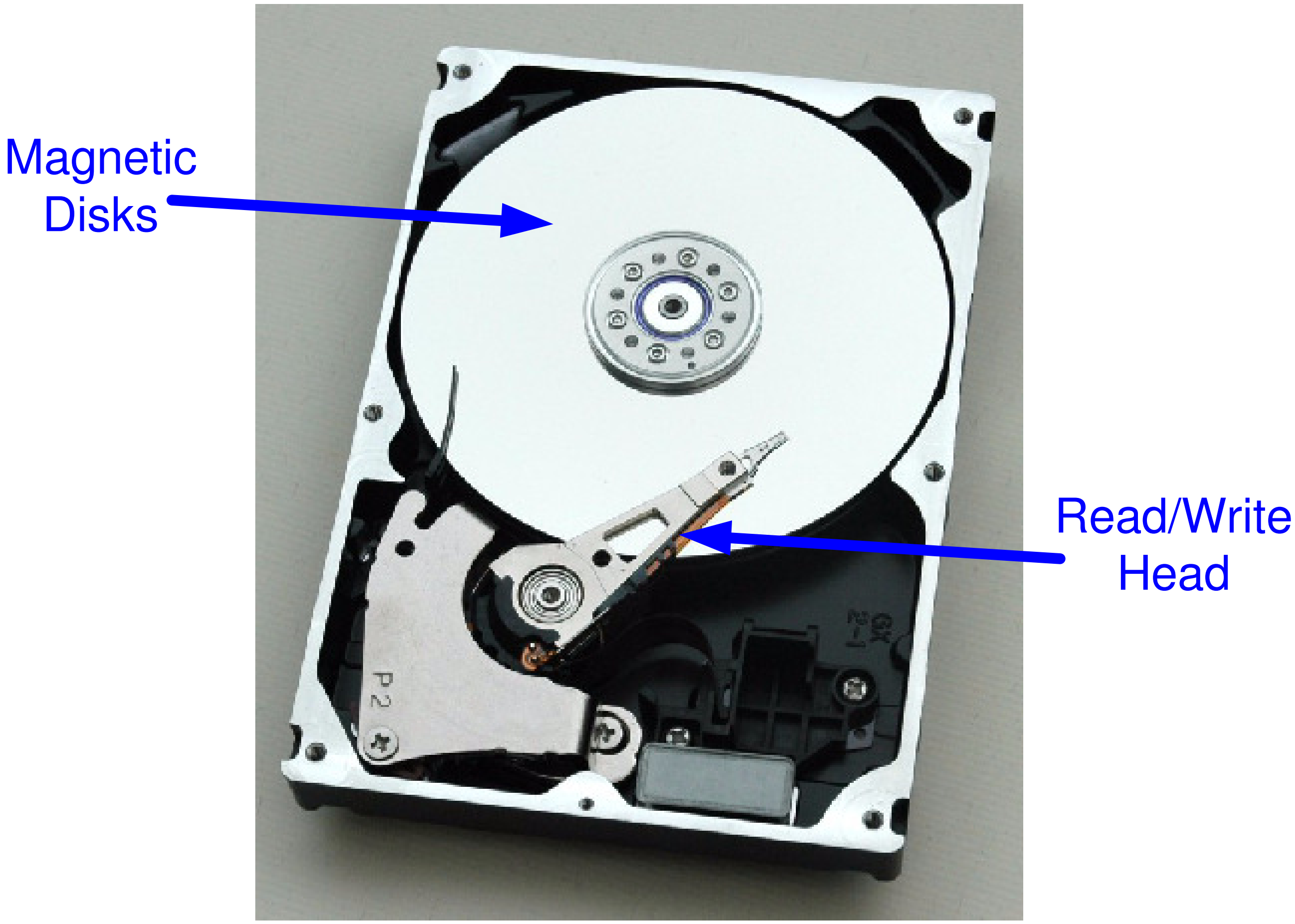
The Memory Hierarchy



Technology	cost / GB	Access time
SRAM	~ $10,000	~ 1 ns
DRAM	~ $100	~ 100 ns
Hard Disk	~ $1	~ 10,000,000 ns

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

The Hard Disk



Takes milliseconds to *seek* correct location on disk

Virtual Memory

- Each program uses ***virtual addresses***
  - Entire virtual address space stored on a hard disk.
  - Subset of virtual address data in DRAM
  - CPU translates virtual addresses into ***physical addresses***
  - Data not in DRAM is fetched from the hard disk

- Each program has its own virtual to physical mapping
  - Two programs can use the same virtual address for different data
  - Programs don't need to be aware that others are running
  - One program (or virus) can't corrupt the memory used by another
  - This is called ***memory protection***

Cache/Virtual Memory Analogues

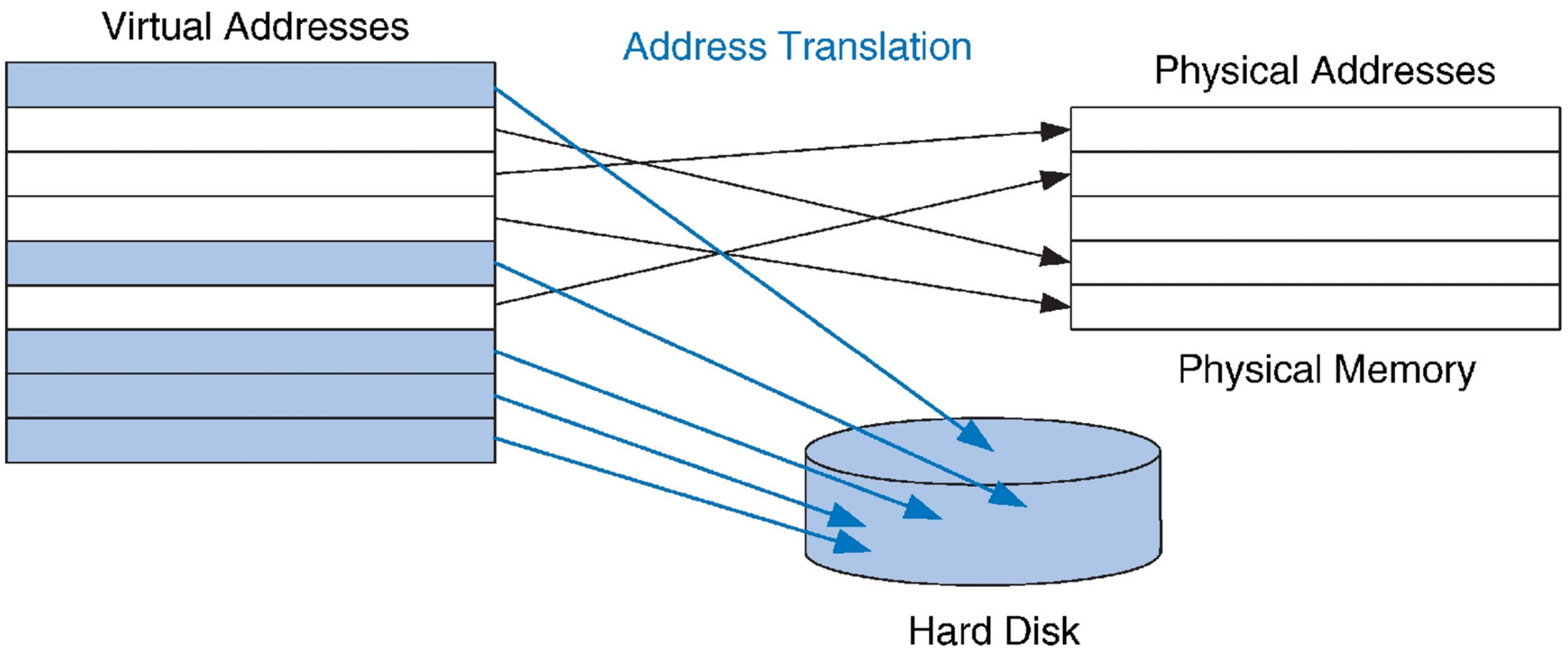
Physical memory acts as cache for virtual memory

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

Virtual Memory Definitions

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

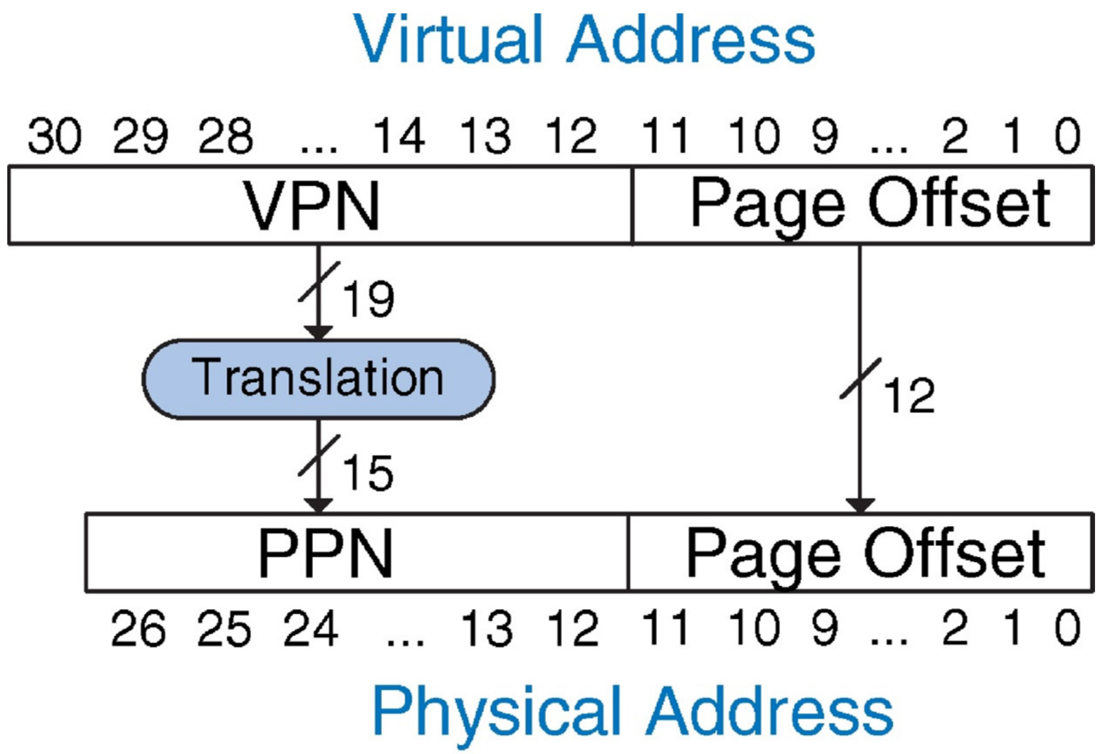
Virtual and Physical Addresses



Most accesses hit in physical memory

But programs have the large capacity of virtual memory

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

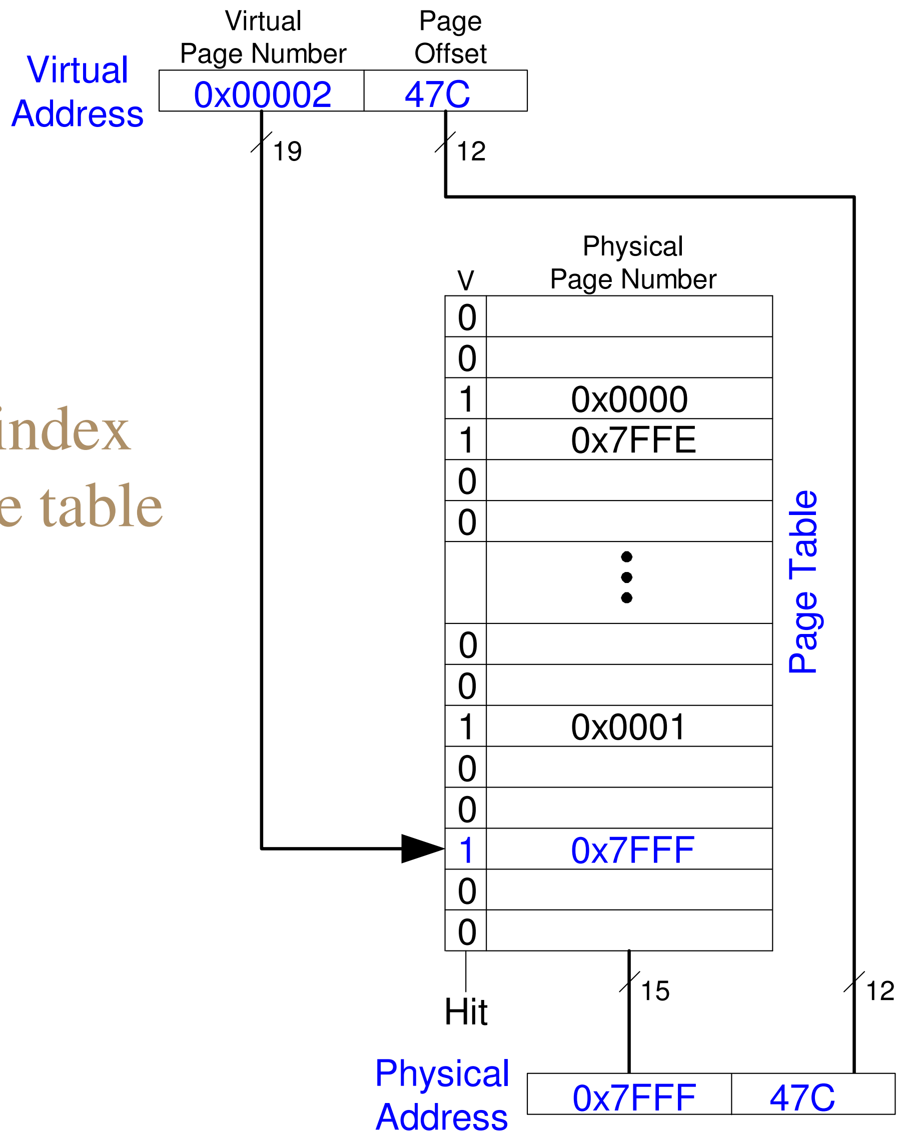
Virtual Memory Example

- **System:**
  - Virtual memory size: 2 GB = 2^{31} bytes
  - Physical memory size: 128 MB = 2^{27} bytes
  - Page size: 4 KB = 2^{12} bytes
- **Organization:**
  - Virtual address: **31** bits
  - Physical address: **27** bits
  - Page offset: **12** bits
  - # Virtual pages = $2^{31}/2^{12}$ = $\mathbf{2^{19}}$ (VPN = 19 bits)
  - # Physical pages = $2^{27}/2^{12}$ = $\mathbf{2^{15}}$ (PPN = 15 bits)

How do we translate addresses?

- **Page table**
  - Has entry for each virtual page
  - Each entry has:
    - **Valid bit:** whether the virtual page is located in physical memory (if not, it must be fetched from the hard disk)
    - **Physical page number:** where the page is located

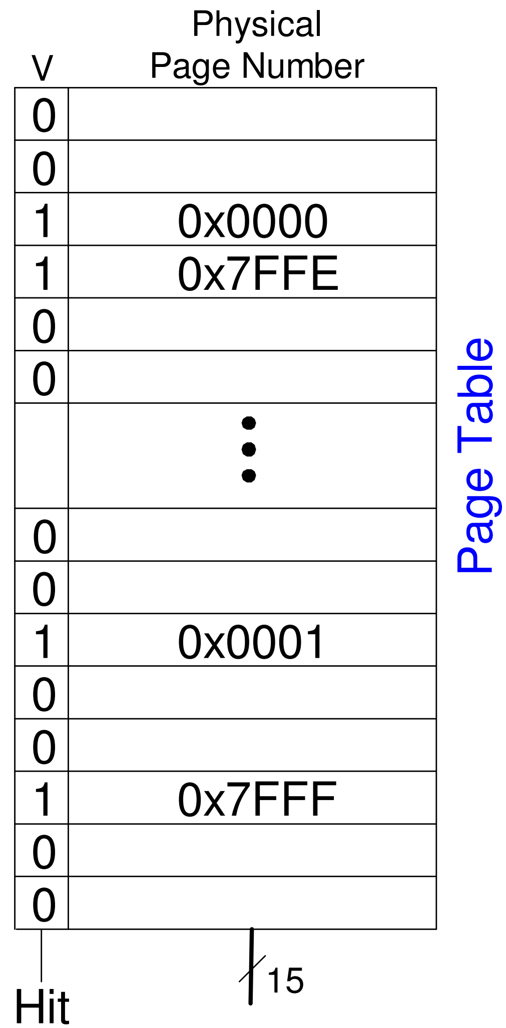
Page Table Example



VPN is index into page table

Page Table Example 1

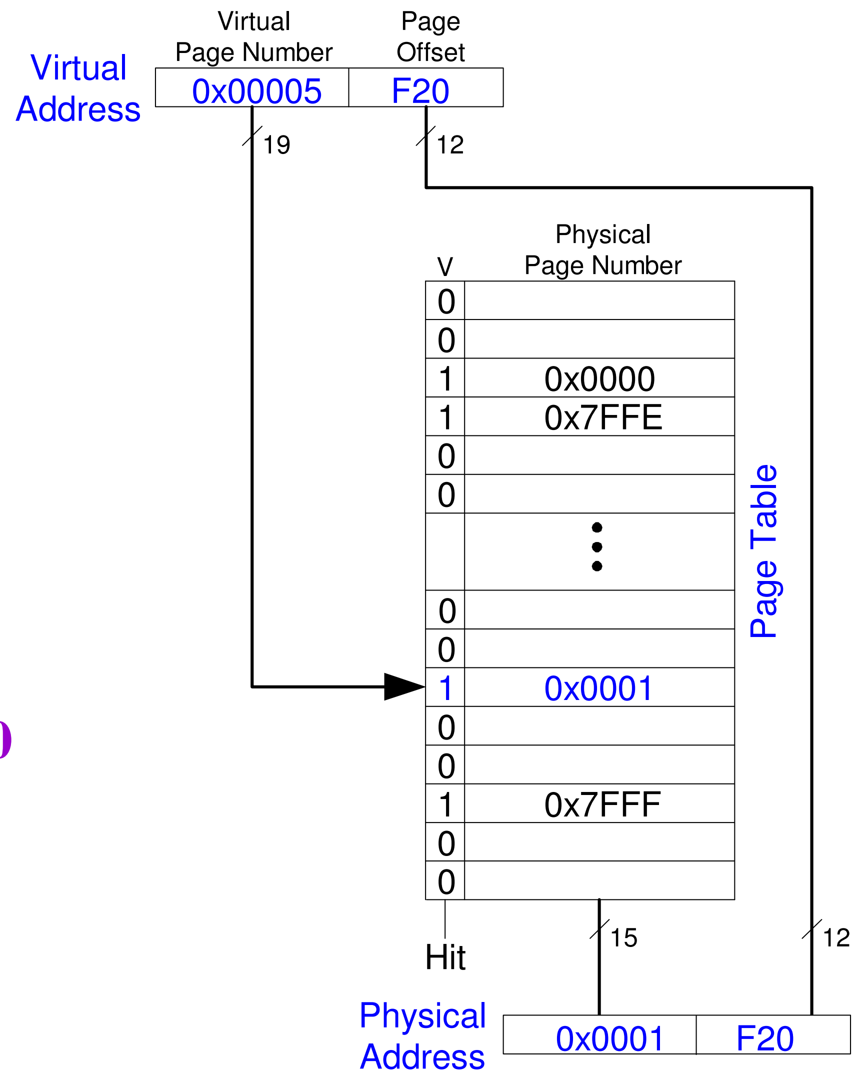
What is the physical address of virtual address **0x5F20**?



Page Table Example 1

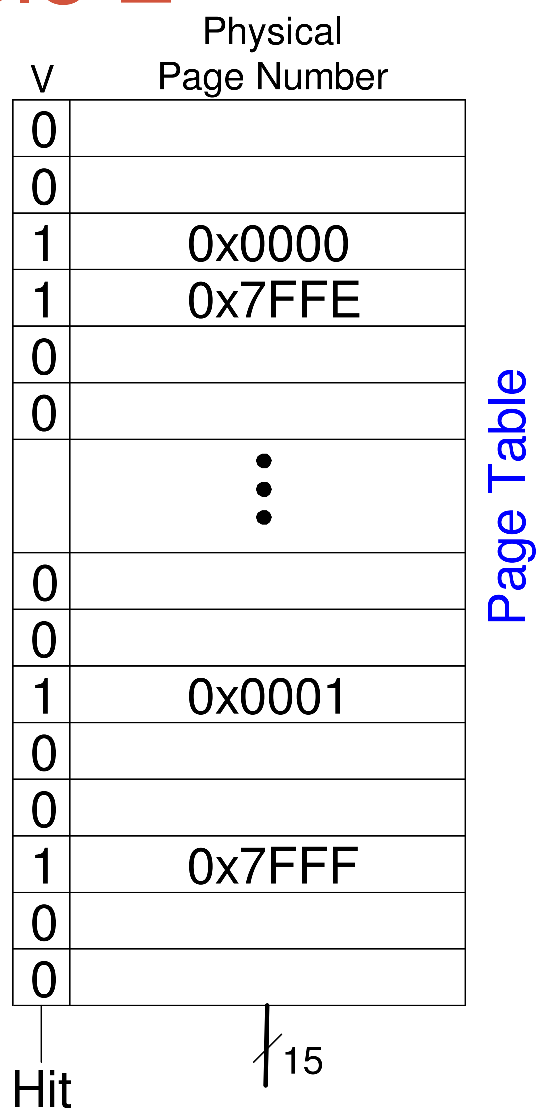
What is the physical address of virtual address **0x5F20**?

- VPN = **5**
- Entry 5 in page table indicates VPN 5 is in physical page **1**
- Physical address is **0x1F20**



Page Table Example 2

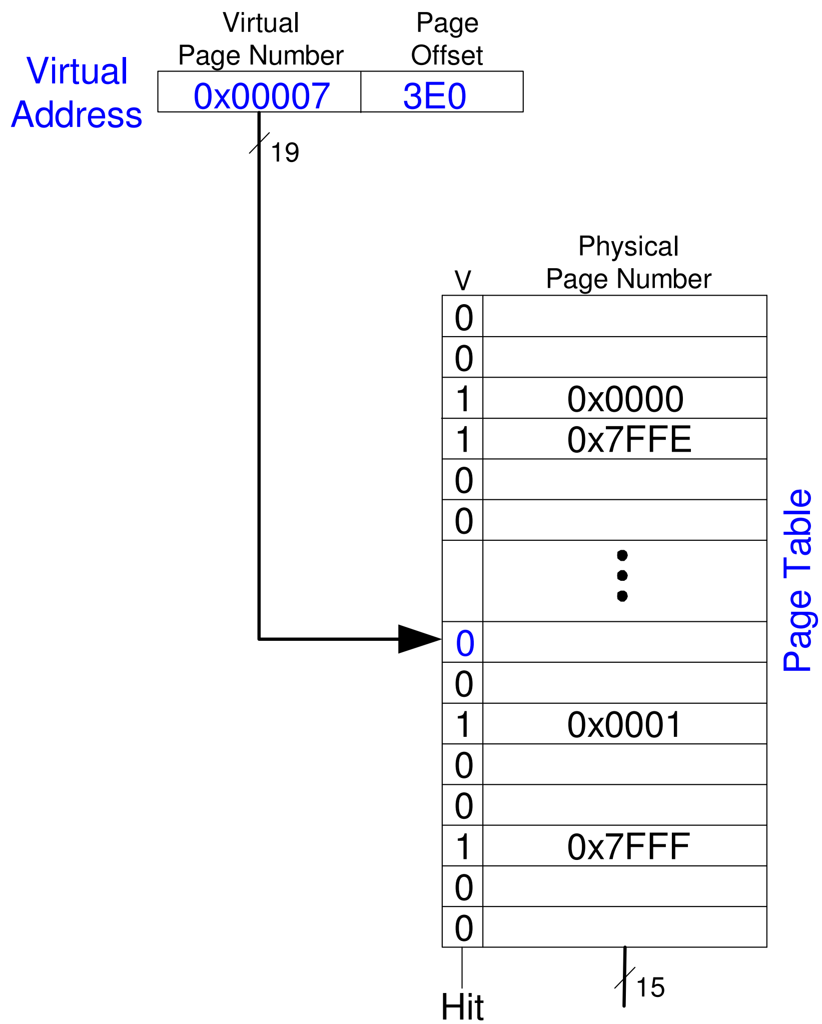
What is the physical address of virtual address **0x73E0**?



Page Table Example 2

What is the physical address of virtual address **0x73E0**?

- VPN = **7**
- Entry 7 in page table is invalid, so the page is not in physical memory
- The virtual page must be ***swapped*** into physical memory from disk



Page Table Challenges

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

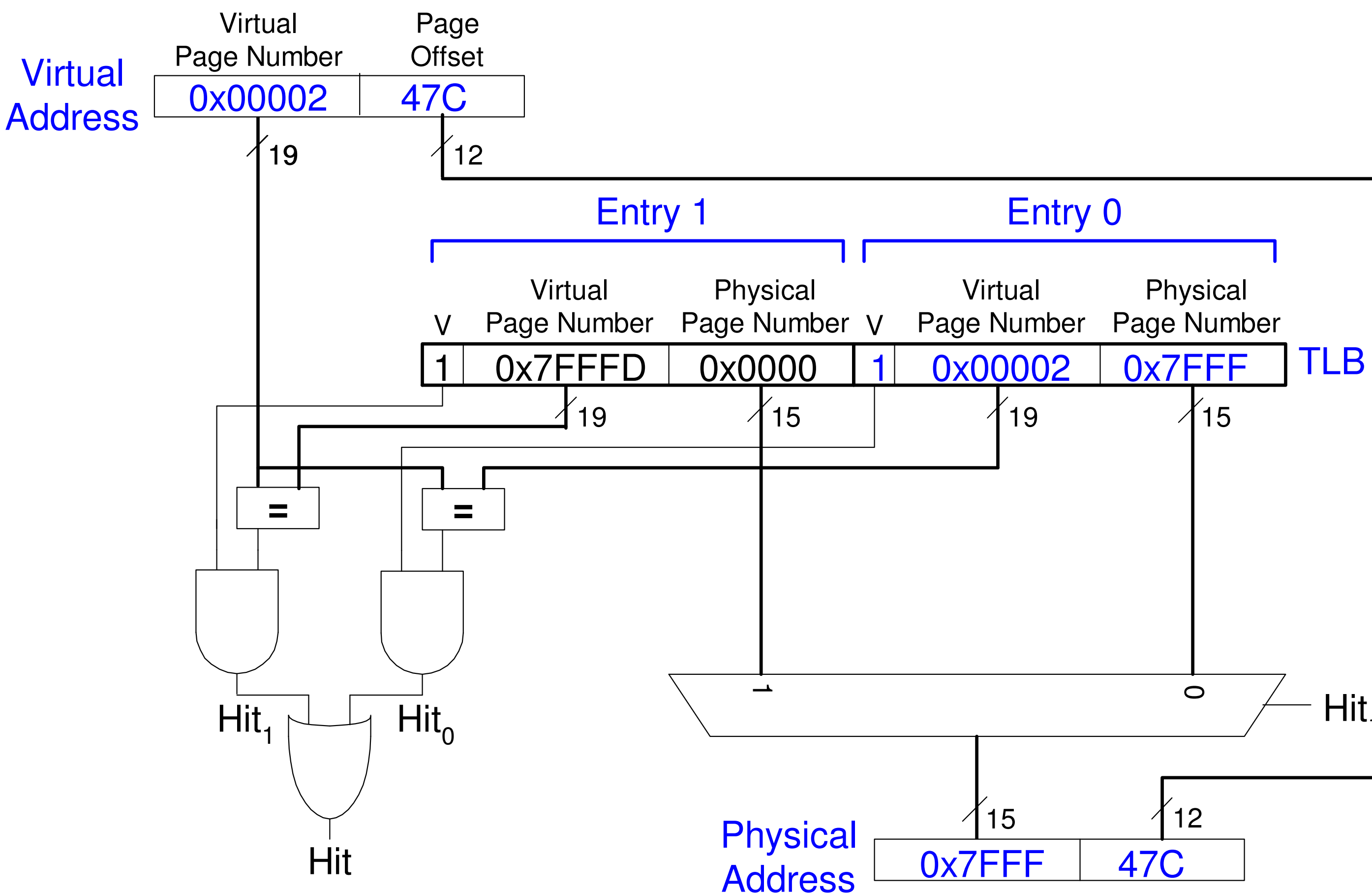
Translation Lookaside Buffer (TLB)

- Use a *translation lookaside buffer* (TLB)
  - Small cache of most recent translations
  - Reduces number of memory accesses required for *most* loads/stores from two to one

Translation Lookaside Buffer (TLB)

- Page table accesses have a lot of temporal locality
  - Data accesses have temporal and spatial 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 # of memory accesses for most loads and stores from 2 to 1

Example Two-Entry TLB



Memory Protection

- Multiple programs (*processes*) run at once
- Each process has its own page table
- Each process can use entire virtual address space without worrying about where other programs are
- A process can only access physical pages mapped in its page table – can't overwrite memory from another process

Virtual Memory Summary

- Virtual memory increases **capacity**
- A subset of virtual pages are located in physical memory
- A **page table** maps virtual pages to physical pages – this is called address translation
- A **TLB** speeds up address translation
- Using different page tables for different programs provides **memory protection**