Virtual Memory
Part II
CS 105: Computer Systems
Lecture 23

Melissa O'Neill

April 20, 2026

Learning Goals

- **Review virtual memory concepts**
- **Motivate and reason about speeding up memory address translation with translation lookaside buffers (TLBs)**
- **Practice address translation, including TLBs and caches**

Exercise: reasoning about page size

Suppose that 7 bits of a 10-bit memory address are used to encode page offset.

1. **What is the page size?**

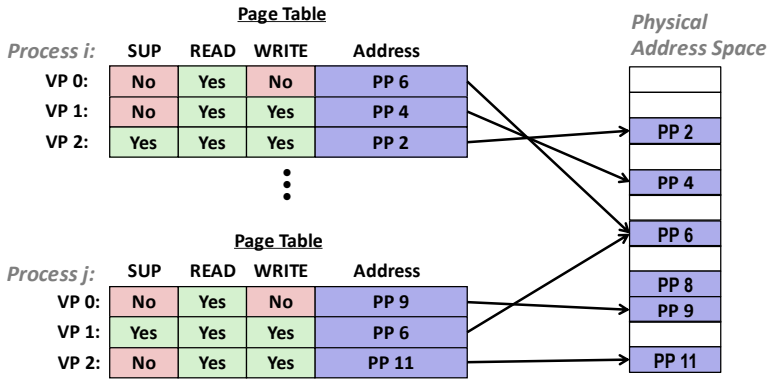
2. **Determine whether each pair of memory addresses fall within the same page.**

   a. 0x155 and 0x105

   b. 0x2AA and 0x255

VM as: a Tool for Memory Protection

- **Operating system must control access to memory**
- **Extend page table entries (PTEs) with *permission bits***
  - MMU checks these bits on each access



Memory Mapping

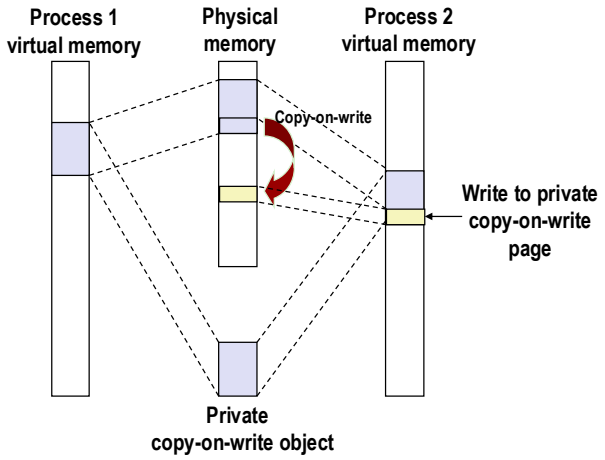
- **A process' VM content is associated with *objects* on disk**
  - This is known as ***memory mapping***
- **VM pages can be *backed by* (i.e., get values from):**
  - An executable object file on disk
    Initial page bytes come from a section of a file
  - Or, for VM pages allocated while process is running, a ***swap file*** on disk (managed by operating system)

On demand paging!

VM as: a Tool for Memory Management

- **Simplifying memory allocation**
  - Each virtual page can be mapped to any physical page
  - A virtual page can be stored in different physical pages at different times
- **Share code and data amongst processes!**

Sharing Pages: Private Copy-on-write (COW) Objects



- **Instruction writing to private page triggers protection fault.**
- **Handler creates new R/W page.**
- **Instruction restarts upon handler return.**
- **Copying deferred as long as possible!**

Exercise: physical memory accesses

Consider this assembly code and setup for the following two questions.
Setup: 13-bit virtual addresses, 12-bit physical addresses, page size is 2^{10} bytes.

```
# instruction address     # instruction
0x410:                    mov    0xc08,%rdi
0x418:                    mov    0xc10,%rsi
```

1. **List the virtual page number (VPN) for each virtual address (VA) in this code that will need to be translated into a physical address (PA).**

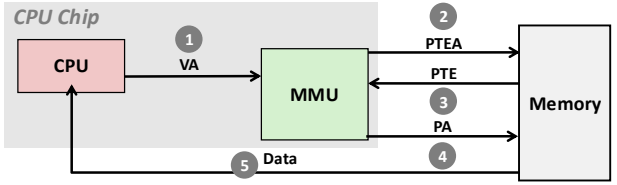
Exercise: physical memory accesses (cntd)

2. **Assume each VPN you listed from question #1 is currently mapped to a physical page, i.e., it is resident in physical memory. Also assume the page table is in physical memory. When the code runs, how many accesses to physical memory will occur? *Assume no caches.***

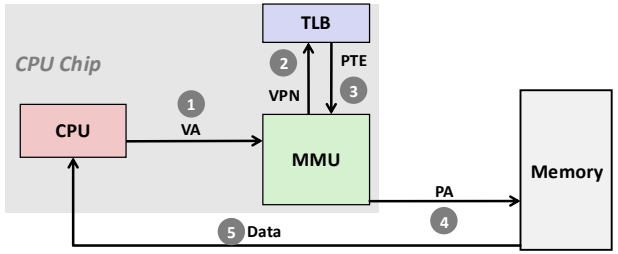
Address Translation: Page Hit

… with a cache for address translations (the TLB)

Instead of:



Do this:

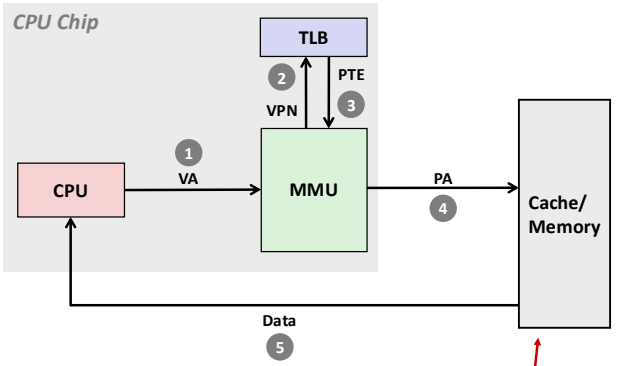


A TLB hit eliminates a memory access!

Speeding up Translation by caching PTEs

- **Cache for PTEs separate from cache(s) for data/instructions. Why?**
  - PTEs could be be evicted from shared L1 cache by other data references
  - PTE hit would still require a small L1 delay
- **Solution: *Translation Lookaside Buffer* (TLB)**
  - Small hardware cache in MMU
  - Maps virtual page numbers to physical page numbers
  - Contains complete page table entries for small number of pages

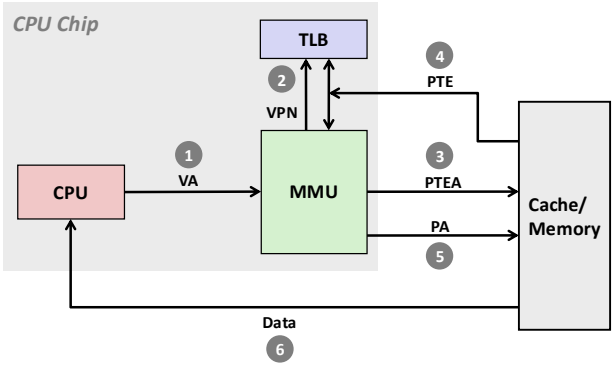
Address Translation: TLB Hit



A TLB hit eliminates a memory access

The data that PA refers to might be found in hardware cache(s)!

Address Translation: TLB Miss

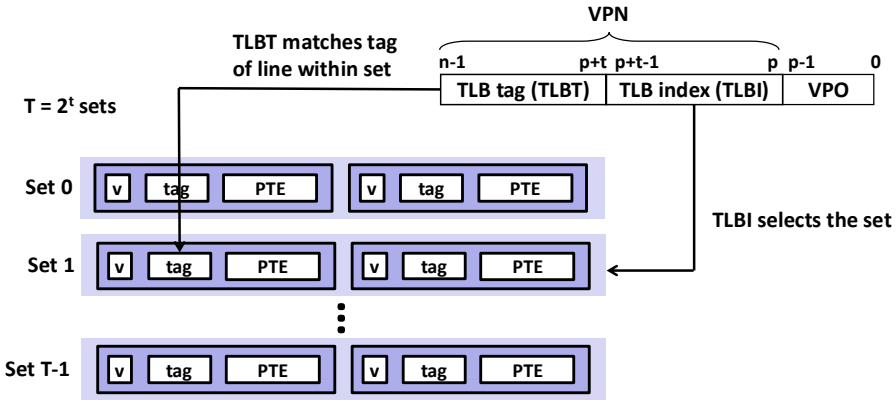


A TLB miss incurs an additional memory access (the PTE)

Fortunately, TLB misses are rare. (Think about why)

Accessing the TLB

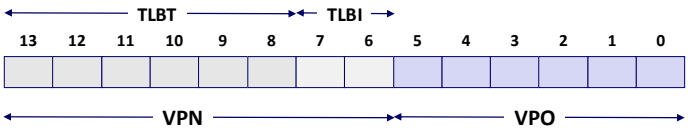
- MMU uses the VPN portion of the virtual address to access the TLB:



Exercise: TLB concepts

1. **What does the Translation look-aside buffer (TLB) cache?**

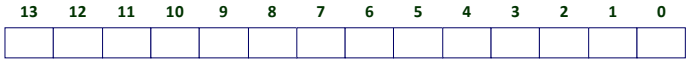
2. **A lookup in the TLB is very similar to a lookup in other hardware caches: we use a set index and a tag. However, unlike other caches, there is no block offset needed for the TLB. Why?**



Example: Address Translation #1

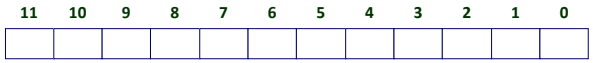
Virtual Address: 0x03D4

Convert VA to binary



VPN ___ TLBI ___ TLBT ____ TLB Hit? __ Page Fault? __ PPN: ____

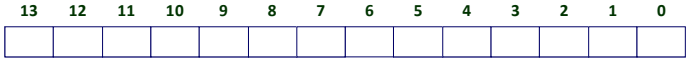
Physical Address



CO ___ CI___ CT ____ Hit? __

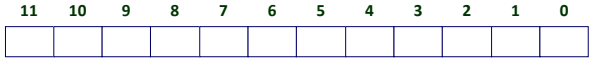
Exercise: Address Translation #2

Virtual Address: 0x0020



VPN ___ TLBI ___ TLBT ____ TLB Hit? __ Page Fault? __ PPN: ____

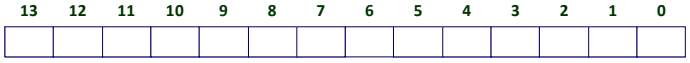
Physical Address



CO ___ CI___ CT ____ Hit? __

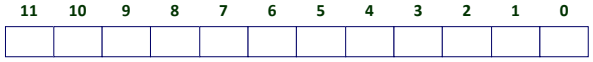
Exercise: Address Translation #3

Virtual Address: 0x0316



VPN ___ TLBI ___ TLBT ____ TLB Hit? __ Page Fault? __ PPN: ____

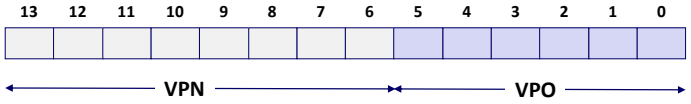
Physical Address

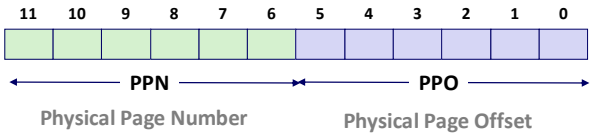


CO ___ CI___ CT ____ Hit? __

Example: Virtual and Physical addresses

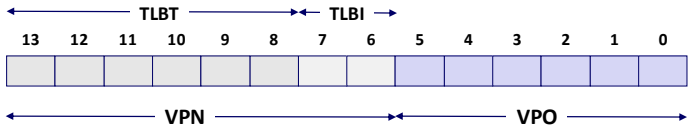
- **Addressing**
  - 14-bit virtual addresses
  - 12-bit physical address
  - Page size = 64 bytes





Example: TLB configuration and contents

- **4 sets, 4-way associative**
  - 16 entries
  - Note: values below (besides valid bit) are in hexadecimal



Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

Example: Page Table contents

Only showing first 16 entries (out of 256)

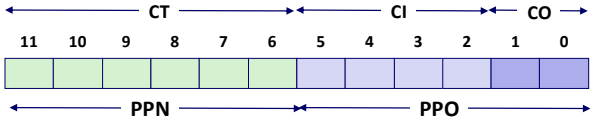
Note: values below (besides valid bit) are in hexadecimal

VPN	PPN	Valid
00	28	1
01	–	0
02	33	1
03	02	1
04	–	0
05	16	1
06	–	0
07	–	0

VPN	PPN	Valid
08	13	1
09	17	1
0A	09	1
0B	–	0
0C	–	0
0D	2D	1
0E	11	1
0F	0D	1

Example: Cache config and contents

- **16 sets, 4-byte block size**
- **Caches based on *physical address***
- **Direct mapped**
- Note: values below (besides valid bit) are in hexadecimal



Idx	Tag	Valid	B0	B1	B2	B3
0	19	1	99	11	23	11
1	15	0	–	–	–	–
2	1B	1	00	02	04	08
3	36	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	–	–	–	–
7	16	1	11	C2	DF	03

Idx	Tag	Valid	B0	B1	B2	B3
8	24	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–