UNM SCHOOL of ENGINEERING
Department of Computer Science

Lecture #28 – part 1
Virtual Memory (end)

Prof. Soraya Abad-Mota, PhD

Housekeeping

- No more index cards to submit
- Please pick up your old ones today
- Post survey (worth 3 points for simply answering on time) has been published since last week, due date is 12/9/24
- Participation scores have been posted (let me know of discrepancies on the totals)
- Pending scores will be posted as they are ready (this week)

Refresher (1)

1. What is a virtual address (VA)?
2. Which are the components of a VA?
3. What is a Physical Address (PA) and its components?
4. What is a page table and what is it used for?
5. What does it mean for a virtual page to be cached?

Refresher (2)

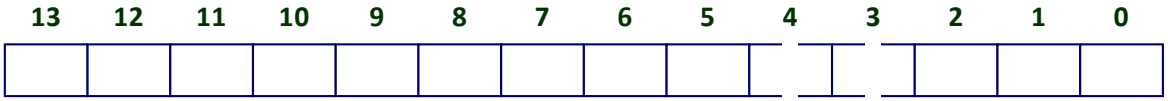
6. What is the MMU?
7. What is the main task performed by the MMU?
8. What is the relationship between a virtual address (VA) and a physical address (PA)?
9. What is address translation in this context?
10. Which components perform the address translation?

Today

- Address translation (example 3)
- Multi-level page tables
- Case study: Core i7/Linux memory system (example to study on your own)
- Lect #28 – part 2:
  - System-Level Input/output (Chap. 10)

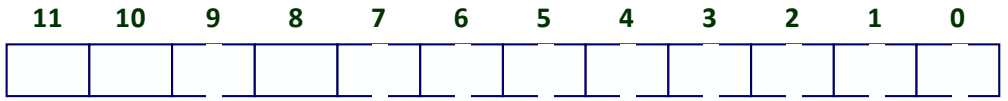
Address Translation Example #3 (problem 9.4)

Virtual Address: `0x03D7`



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

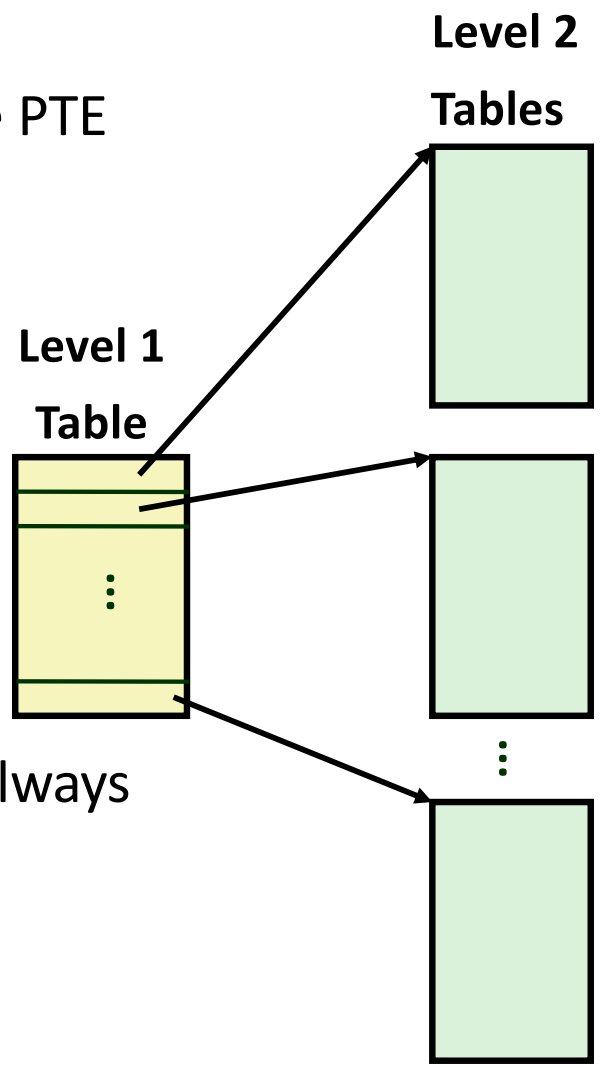
Physical Address



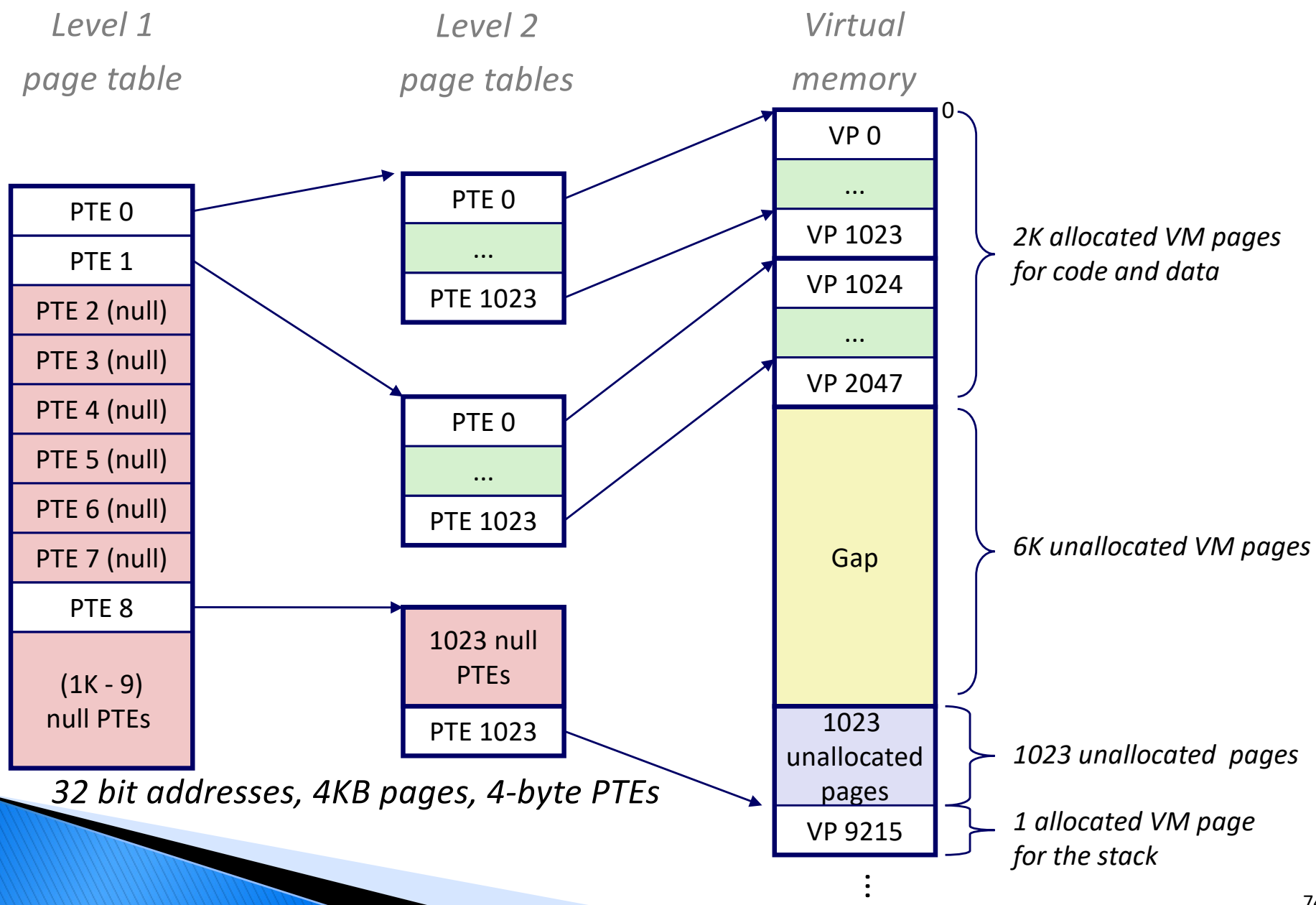
CO ___ CI__ CT ___ Hit? __ Byte: ____

Multi-Level Page Tables

- Suppose:
  - 4KB (2^{12}) page size, 48-bit address space, 8-byte PTE
- Problem:
  - Would need a 512 GB page table!
    - $2^{48} * 2^{-12} * 2^{3} = 2^{39}$ bytes
- Common solution: Multi-level page table
- Example: 2-level page table
  - Level 1 table: each PTE points to a page table (always memory resident)
  - Level 2 table: each PTE points to a page (paged in and out like any other data)



A Two-Level Page Table Hierarchy



Translating with a k-level Page Table