Chapter 20: Smaller Page Tables

Adam Disney

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Crux: How To Make Page Tables Smaller?

CRUX: HOW TO MAKE PAGE TABLES SMALLER?

Simple array-based page tables (usually called linear page tables) are too big, taking up far too much memory on typical systems. How can we make page tables smaller? What are the key ideas? What inefficiencies arise as a result of these new data structures?

Simple Solution: Bigger Pages

- Larger pages requires less pages to represent the entire address space.
  - 32-bit AS split into 20-bit VPN + 12-bit offset w/ 4 byte page table entries
    - 2^20 * 4 bytes = 4MB
  - 32-bit AS split into 18-bit VPN + 14-bit offset w/ 4 byte page table entries
    - 2^18 * 4 bytes = 1MB
- But this increases internal fragmentation

Hybrid Approach: Paging and Segments

PFN	valid	prot	present	dirty
10	1	r-x	1	0
-	0	—	-	-
-	0	—	-	-
-	0	—	-	-
23	1	rw-	1	1
-	0	—	-	-
-	0	—	-	-
-	0	—	-	-
-	0	—	-	-
-	0	—	-	-
-	0	—	-	-
-	0	—	-	-
-	0	—	-	-
-	0	—	-	-
-	0	—	-	-
28	1	rw-	1	1
4	1	rw-	1	1

Figure 20.2: **A Page Table For 16KB Address Space**

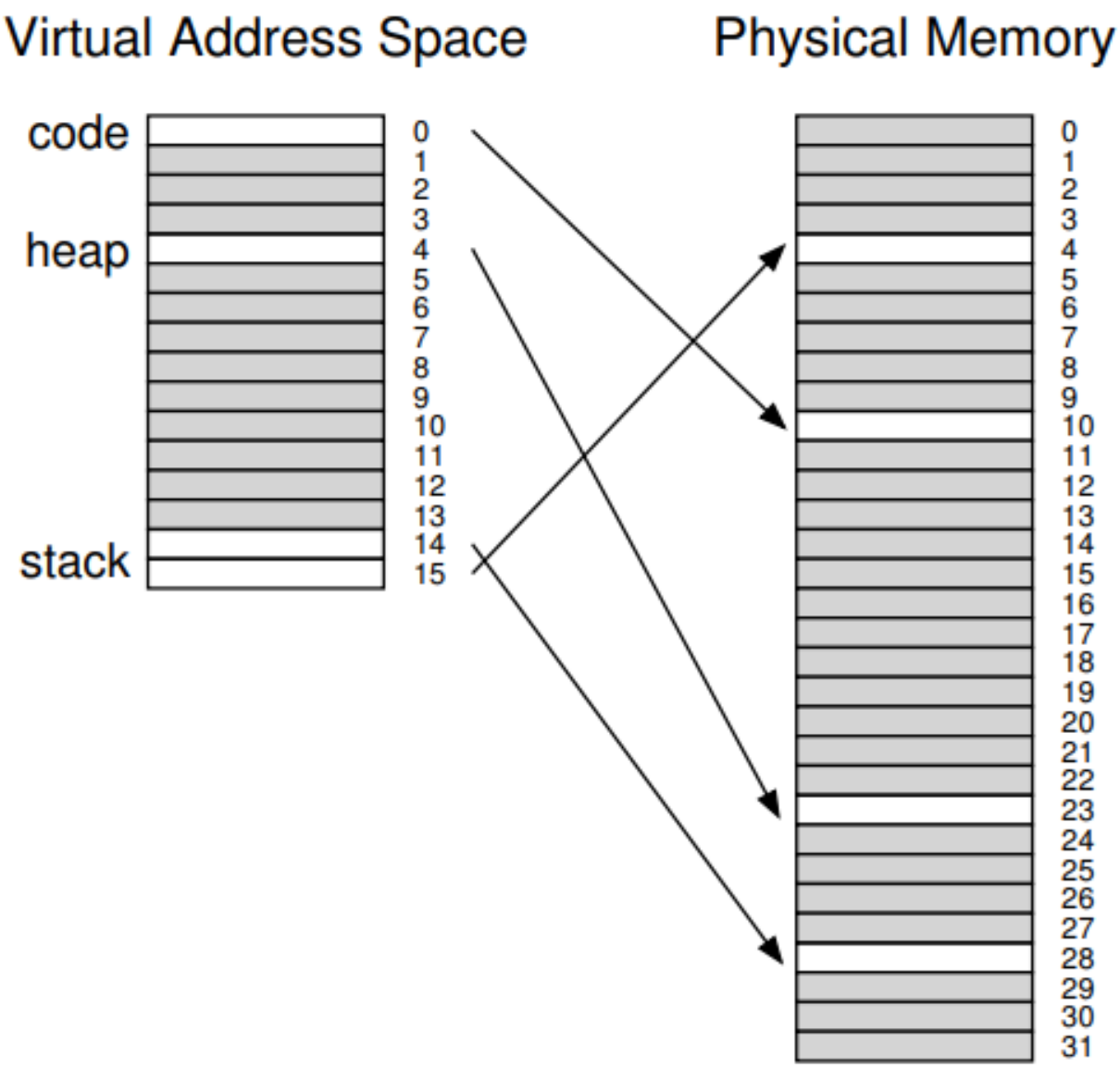
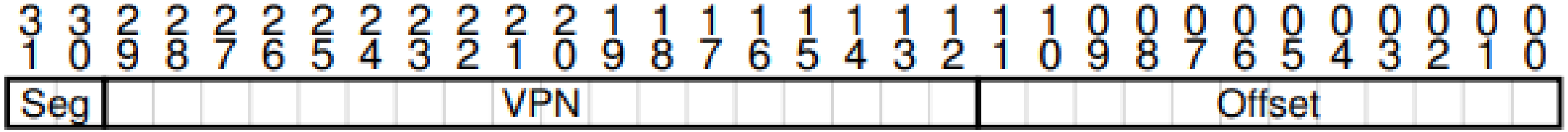


Figure 20.1: **A 16KB Address Space With 1KB Pages**

Hybrid Approach: Paging and Segments

- How about we have a page table per logical segment?
- Now our base and bound registers are for the page tables themselves
- This allows our page tables to only be as large as they need to be
- Issues:
  - Inflexible, assuming memory usage.
  - External fragmentation due to variable size page tables



Multi-level Page Tables

- Chop up the page table into page-sized units
  - If an entire page of PTEs is invalid, don't allocate that page of the page table
- We use a **page directory** to track which page table pages are valid

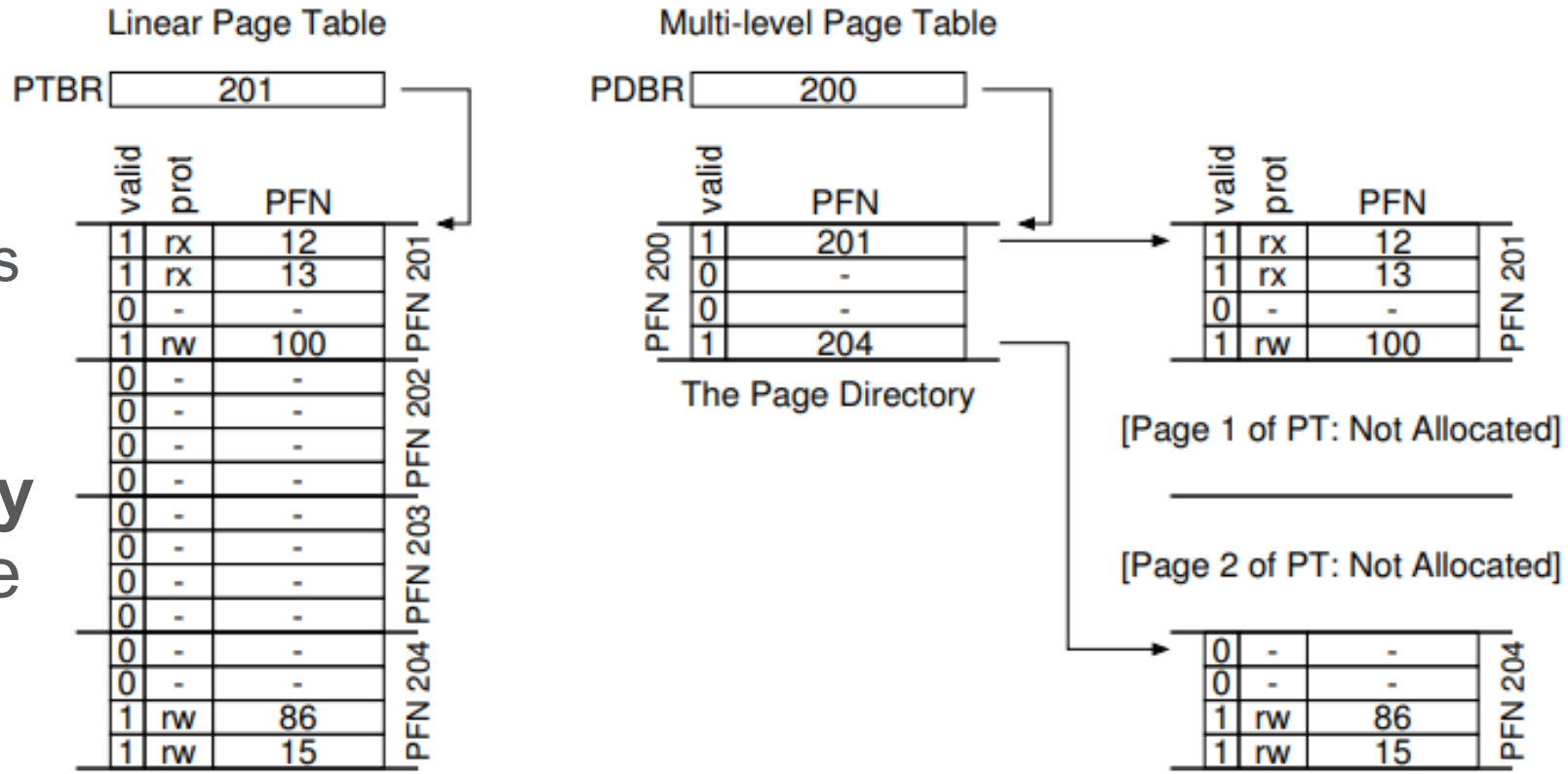


Figure 20.3: **Linear (Left) And Multi-Level (Right) Page Tables**

Multi-level Page Tables (Advantages/Disadvantages)

- Advantages
  - Supports sparse address spaces in a compact page table
  - If carefully constructed, each portion of the page table fits neatly within a page for easy memory management
    - Linear page table must be contiguous in memory. This does not.
- Disadvantages
  - Now requires multiple loads from memory on address translation
    - Time-space trade-offs
  - More complex

Multi-level Page Tables Example

- Address space size is 16KB w/ 64-byte pages
  - 14-bit virtual address space
  - 8-bit VPN + 6-bit offset
- Linear page table must have 2^8 entries
- Assume PTE is 4 bytes that means linear page table would be 1KB

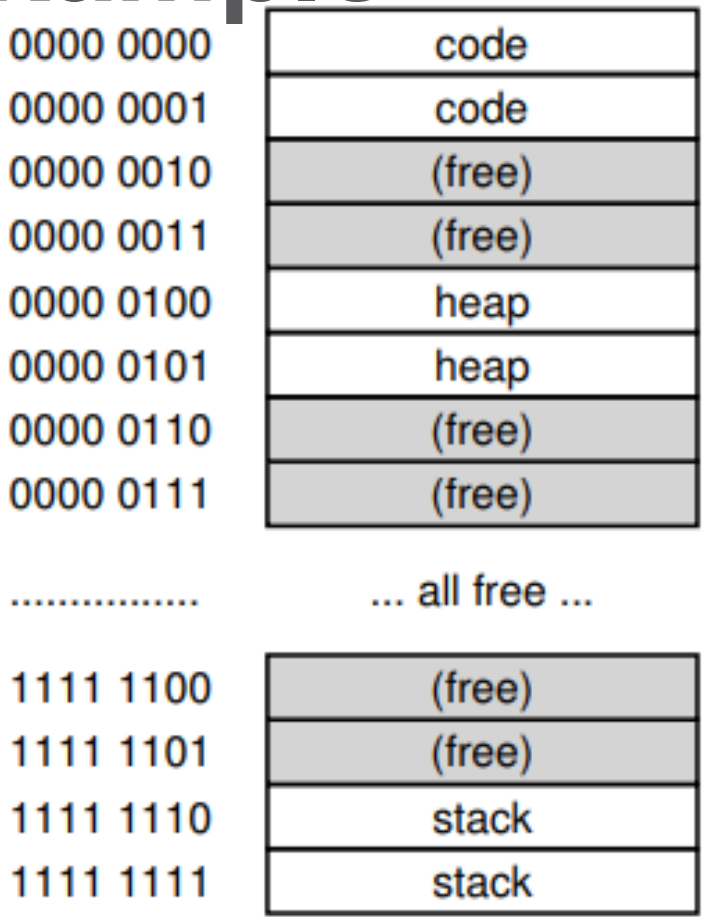
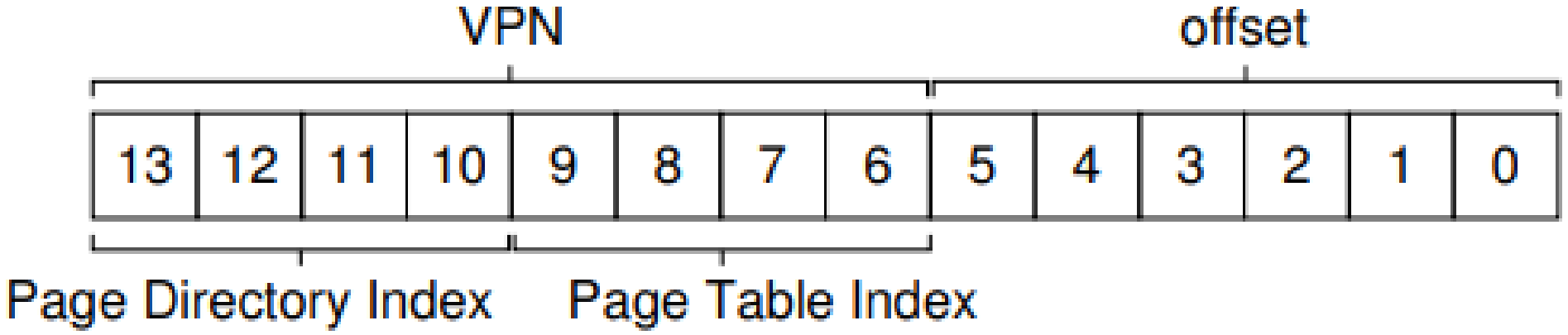


Figure 20.4: **A 16KB Address Space With 64-byte Pages**

Multi-level Page Tables Example

- With 64-byte pages, we can divide the 1KB table into 16 64-byte Page Table Pages
- Thus, the page directory needs 16 entries
- Now we look up the page in the page directory
  - If it's valid, look up the entry in that page of the page table



Multi-level Page Tables Example

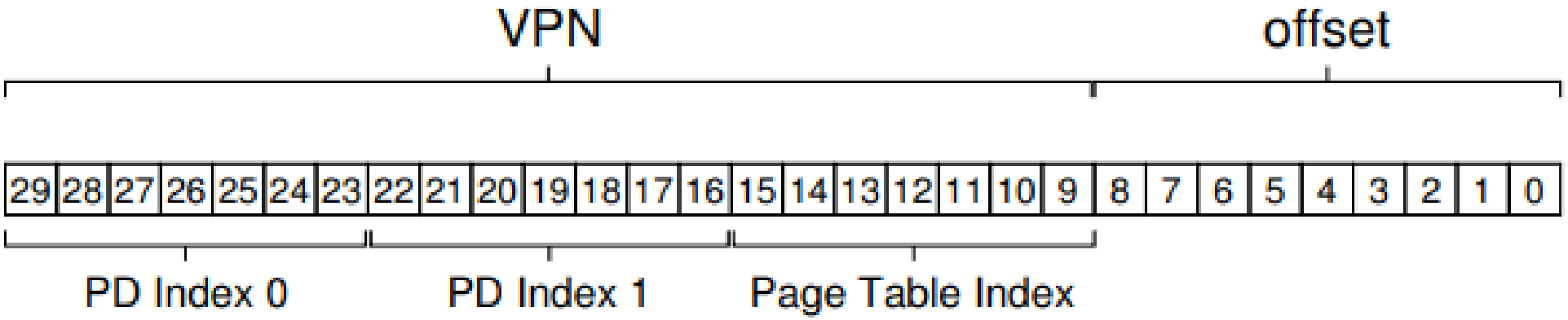
- Notice we only need 3 pages instead of 16 in this example
  - Obviously, 32-bit/64-bit AS would save much more space
- Let's translate an address!

Page Directory		Page of PT (@PFN:100)			Page of PT (@PFN:101)		
PFN	valid?	PFN	valid	prot	PFN	valid	prot
100	1	10	1	r-x	—	0	—
—	0	23	1	r-x	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	80	1	rw-	—	0	—
—	0	59	1	rw-	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	—	0	—
—	0	—	0	—	55	1	rw-
101	1	—	0	—	45	1	rw-

Figure 20.5: **A Page Directory, And Pieces Of Page Table**

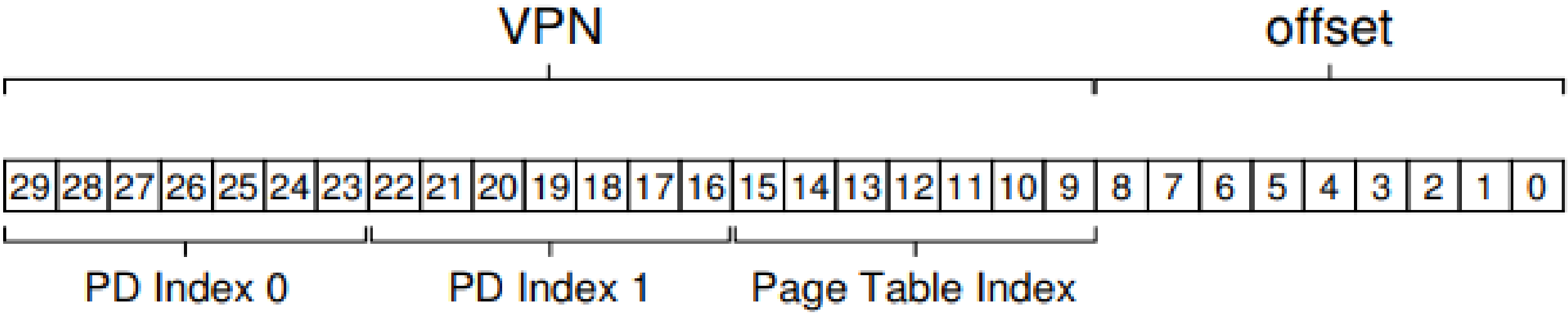
More Than Two Levels

- Generally, we want all parts of the page table to fit in pages
- Thus, if the page directory gets too large, we might need more levels



More Than Two Levels

- Questions!



Inverted Page Tables

- Instead of having many page tables (1 per process), have a single page table representing the ***physical pages***
- Now to know if a page is present in memory, we must scan the table
- Usually paired with a hash table to speed up lookups
- Again, page table is just a data structure which we could represent in many different ways.

Swapping the Page Tables to Disk

- Thus far, we have assumed page tables reside in physical memory
- Some systems place page tables in kernel virtual memory to allow them to swap pages of the page tables to disk
- More on that next time!