IOWA STATE UNIVERSITY
Department of Electrical and Computer Engineering

Lecture 22:
TLB & Multi-Level Page Tables

Agenda

- Recap
- Translation Lookaside Buffer (TLB)
- Multi-Level Page Tables

Recap

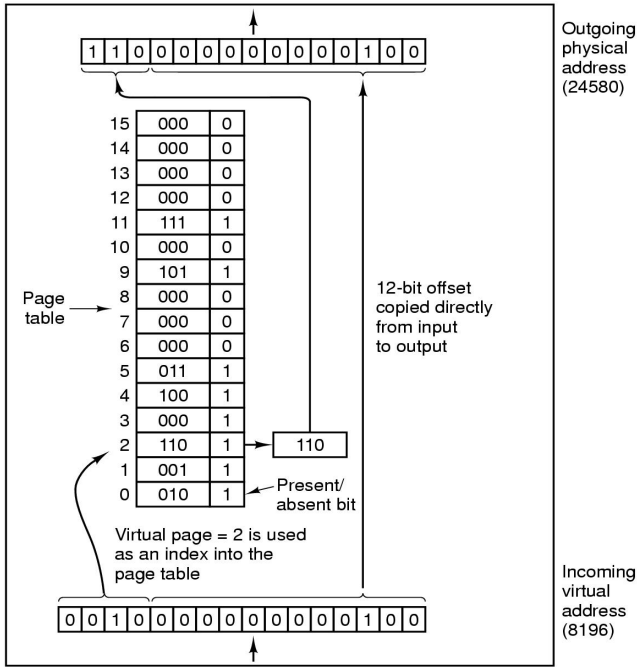
- Where is page table
  - In physical memory
  - accessible by kernel
  - Page-table base register (PTBR) points to the page table
  - Page-table length register (PTLR) indicates size of the page table

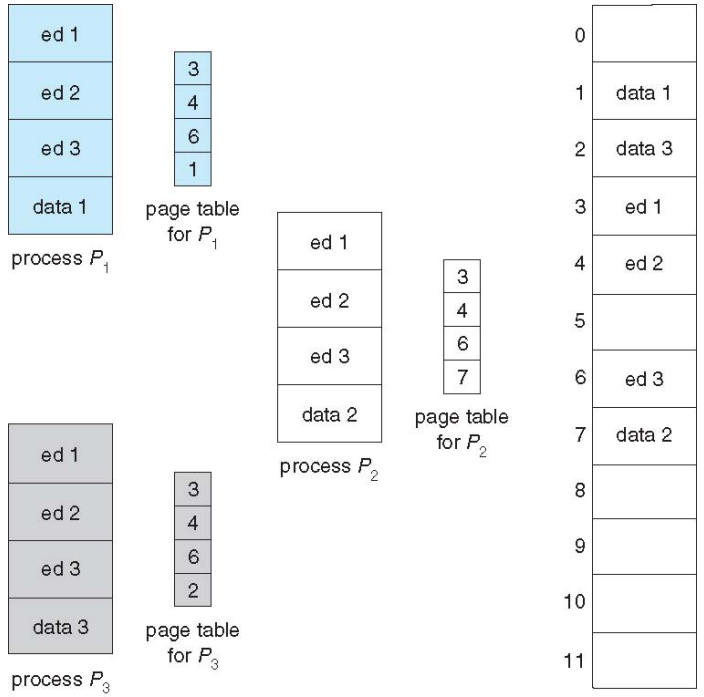
Recap

- What is in the page table
  - Page table entries (PTEs)
    - base address of each page in the physical memory
    - may also contain additional bits for other purposes (e.g., protection)
      - **Valid Bit**: Indicating whether the particular translation is valid
      - **Protection Bit**: Indicating whether the page could be read from, written to, or executed from
      - **Present Bit**: Indicating whether this page is in physical memory or on disk(swapped out)
      - **Dirty Bit**: Indicating whether the page has been modified since it was brought into memory
      - **Reference Bit(Accessed Bit):** Indicating that a page has been accessed

Recap

- Address translation using a page table
  - Enable *shared pages*





Recap

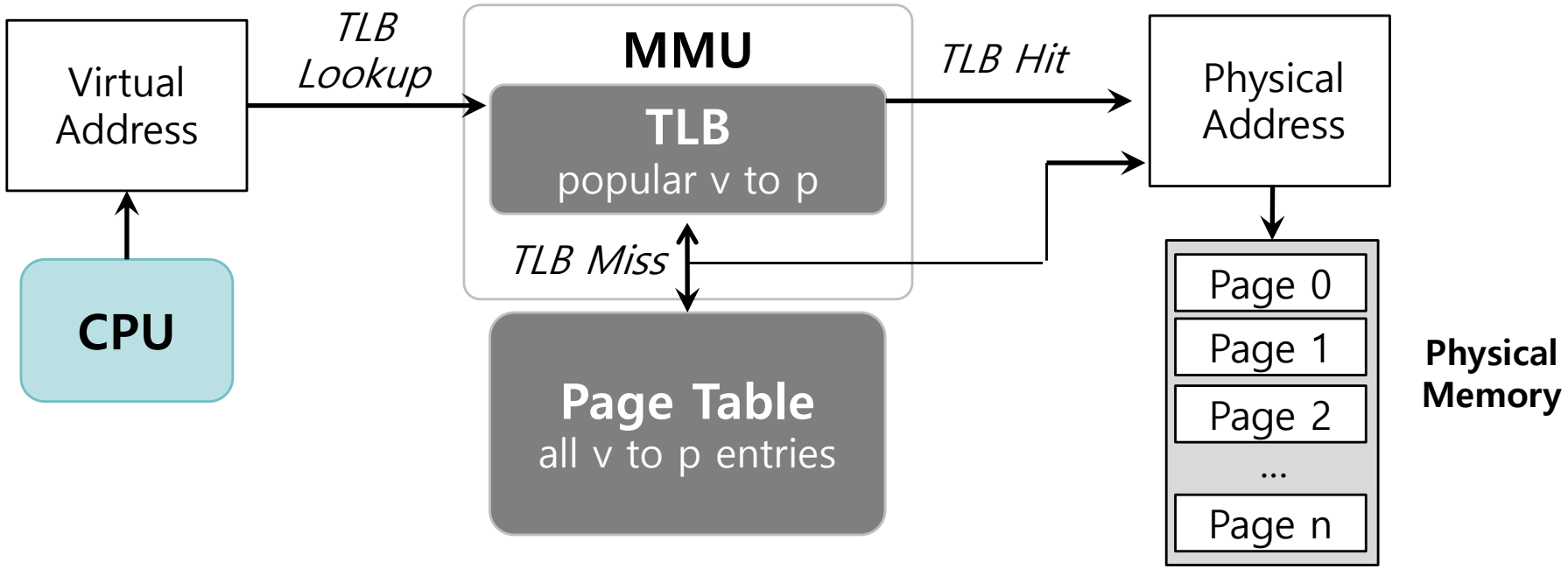
- Page Fault
  - Requested page not in the physical memory
    - MMU reports a page fault to CPU
    - CPU gives control to the OS (page fault handler routine)
    - OS fetches a page from the disk
      - May needs to evict an existing page from memory
    - Instruction is restarted
- Potential Issues of Paging
  - Time
    - for every memory access, one additional access is needed
  - Space
    - A page table can get awfully large
    - Each process needs to have a page table

Agenda

- ~~Recap~~
- Translation Lookaside Buffer (TLB)
- Multi-Level Page Tables

Translation Lookaside Buffer (TLB)

- A hardware cache of popular virtual-to-physical address translation
  - Purpose: to speedup paging
  - Location: part of the memory-management unit (MMU)



Translation Lookaside Buffer (TLB)

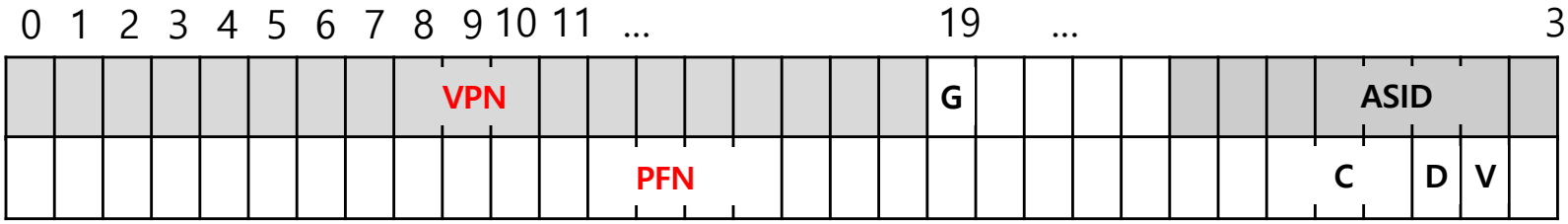
- Structure
  - Associate registers – parallel search

Page #	Frame #

  - Address translation (Virtual Page#, Physical Frame#)
    - If virtual page# is in associative register
      - get the corresponding physical frame# from TLB
    - Otherwise
      - Walk through page table to get the frame#
  - Each entry may also contain other bits
    - E.g., ASID: address space ID

Translation Lookaside Buffer (TLB)

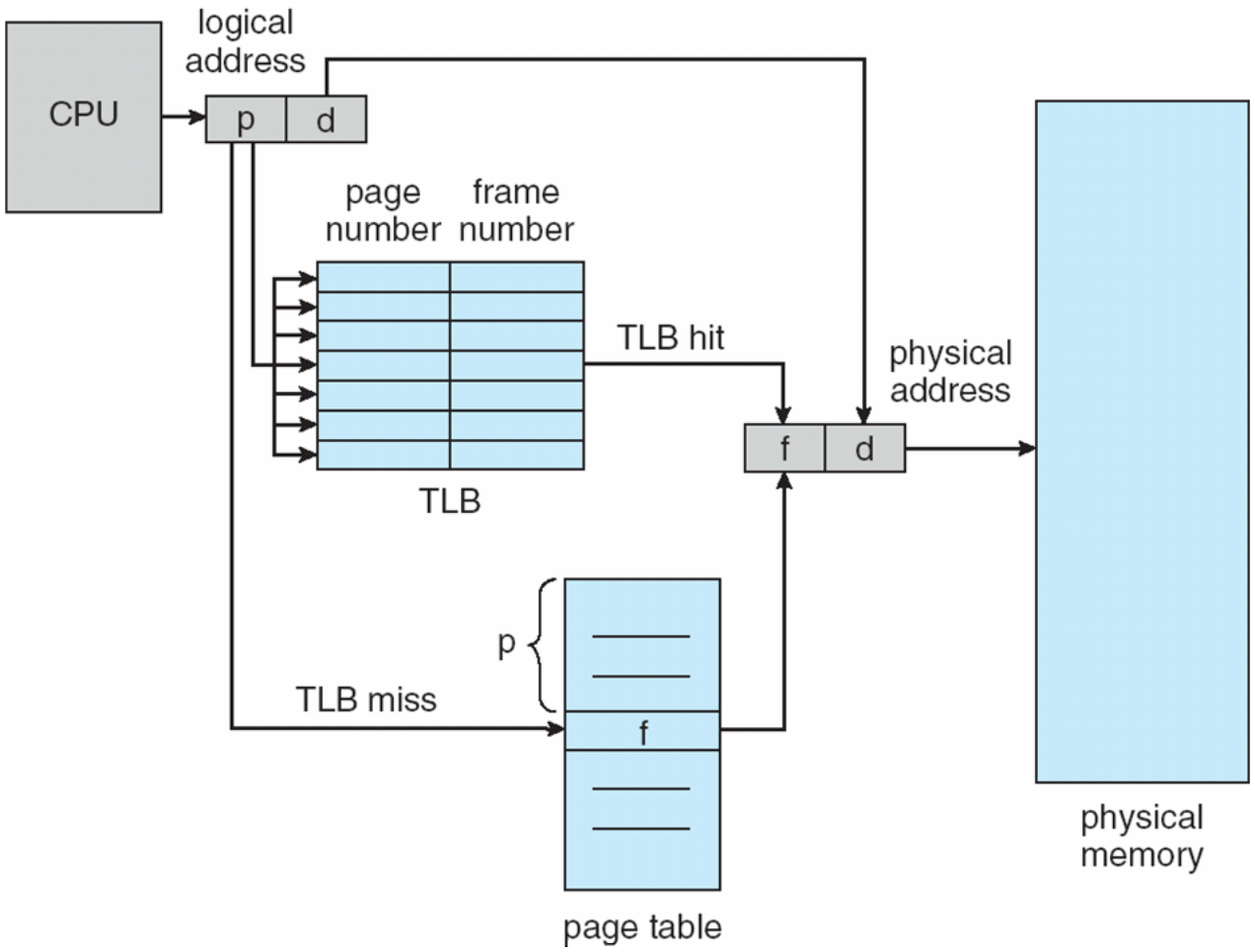
- A Real TLB Entry (MIPS R4000)
  - 64 bits



Flag	Content
19-bit VPN	The rest reserved for the kernel.
24-bit PFN	**Systems can support with up to 64GB of main memory($2^{24} * 4KB$ pages).**
Global bit(G)	Used for pages that are globally-shared among processes.
ASID	**OS can use to distinguish between address spaces.**
Coherence bit(C)	determine how a page is cached by the hardware.
Dirty bit(D)	marking when the page has been written.
Valid bit(V)	tells the hardware if there is a valid translation present in the entry.

Translation Lookaside Buffer (TLB)

- Address translation with TLB & page table

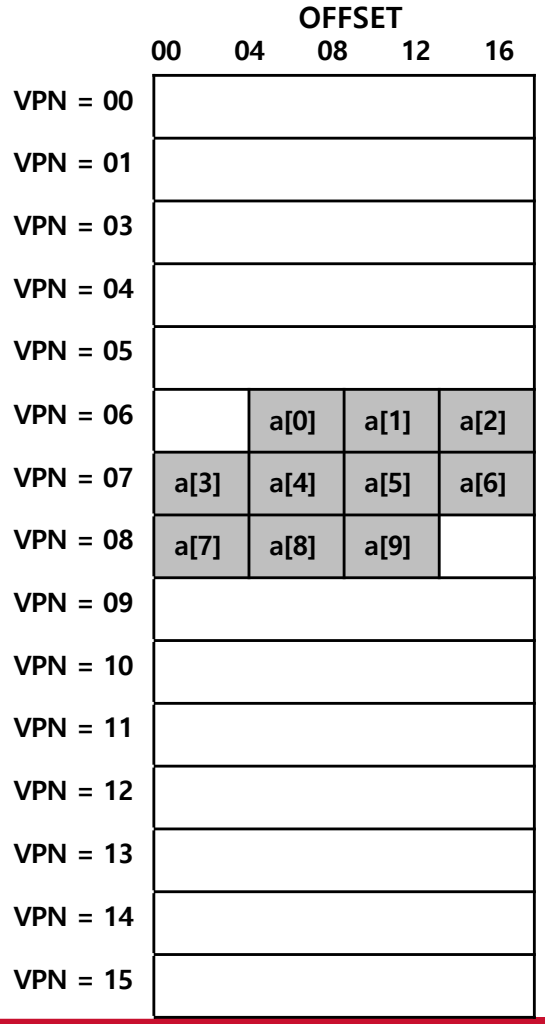


Translation Lookaside Buffer (TLB)

- Effective Access Time (EAT)
  - effective time for accessing memory with TLB
  - TLB Hit ratio = α
    - percentage of times that a page number is found in the TLB
    - TLB hit: one memory access
    - TLB miss: two memory accesses
  - Consider α = 80%, 100ns for each memory access
    - EAT = 0.80 x 100 + 0.20 x 200 = 120ns
  - Consider a more realistic hit ratio α = 99%; still 100ns for each memory access
    - EAT = 0.99 x 100 + 0.01 x 200 = 101ns

How can TLB improve performance

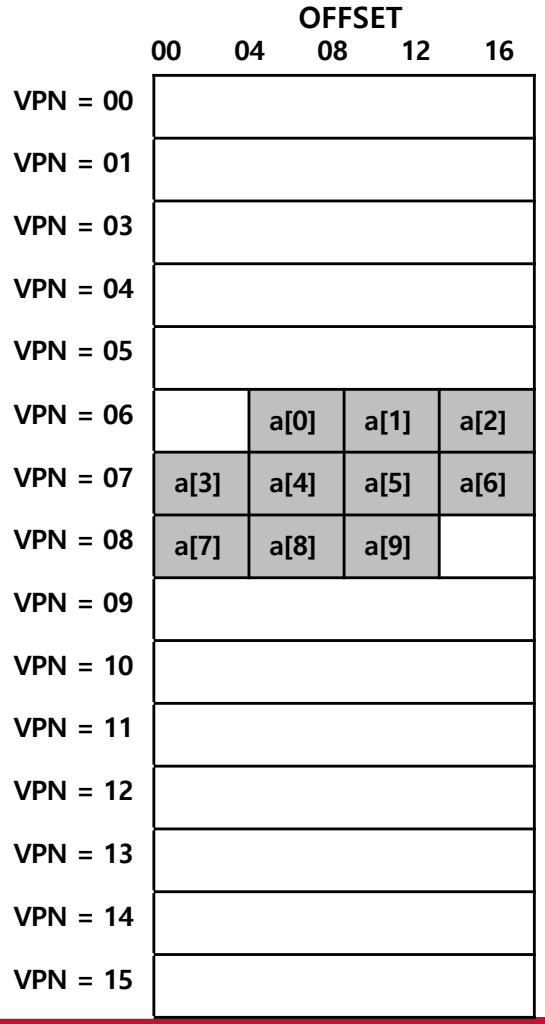
- Example: accessing an array




```
0:      int sum = 0 ;
1:      for( i=0; i<10; i++){
2:              sum+=a[i];
3:      }
```

How can TLB improve performance

- Example: accessing an array



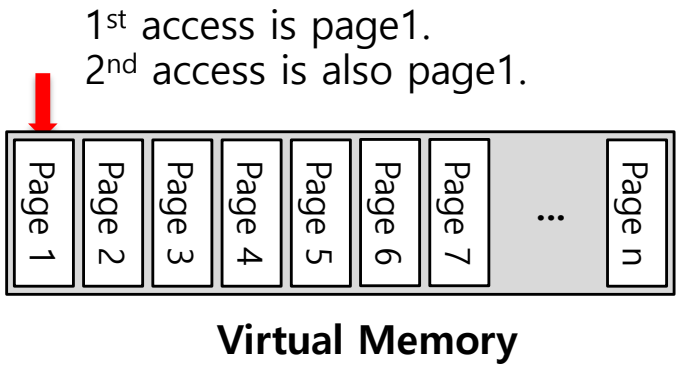

```
0:      int sum = 0 ;
1:      for( i=0; i<10; i++){
2:              sum+=a[i];
3:      }
```

The TLB improves performance due to spatial locality

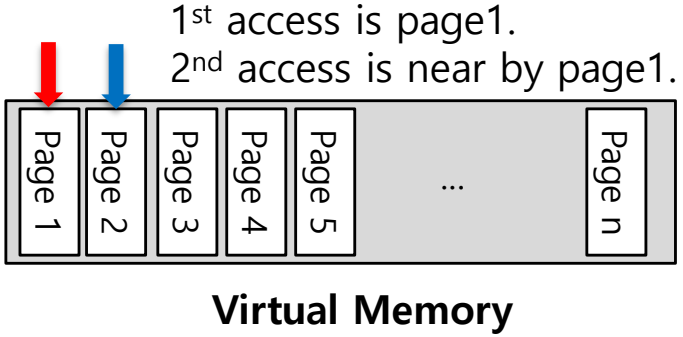
3 misses and 7 hits.
Thus TLB hit rate is 70%.

How can TLB improve performance

- Locality
  - Temporal Locality
    - An instruction or data item that has been recently accessed will likely be re-accessed soon in the future.



  - Spatial Locality
    - If a program accesses memory at address x, it will likely soon access memory near x.



Agenda

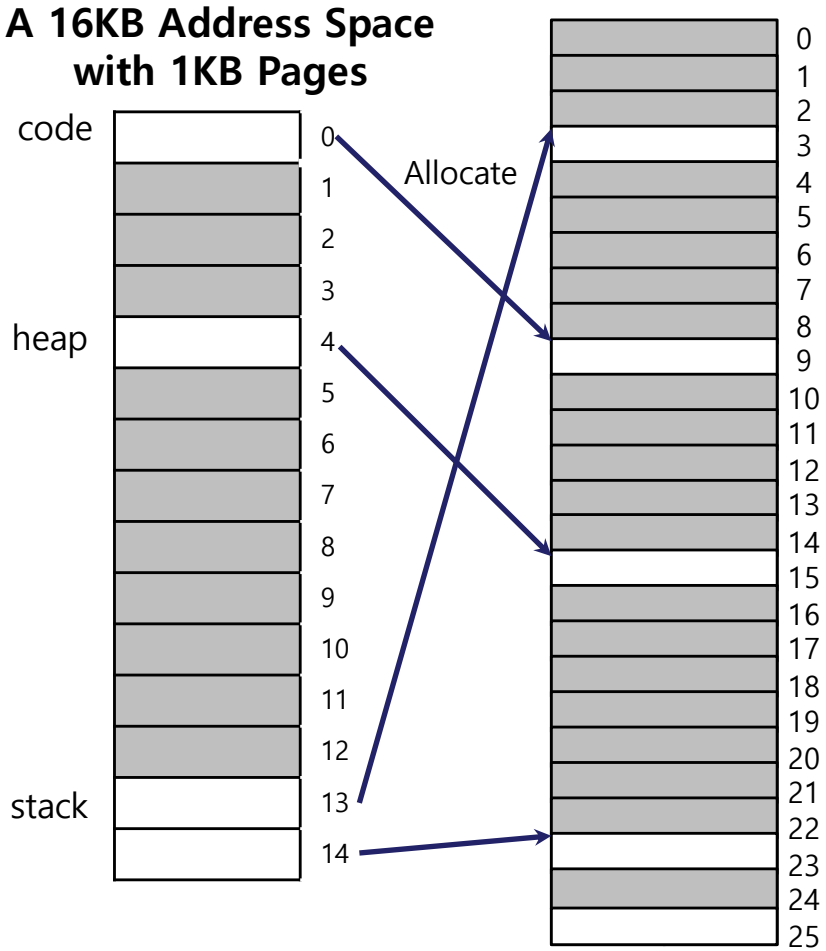
- ~~Recap~~
- ~~Translation Lookaside Buffer (TLB)~~
- Multi-Level Page Tables

Space Overhead of Page Tables

- A (linear) page table can be large
  - Example (revisit):
    - 32-bit address space (4GB) with 4KB pages
    - 12 bits for offset within a page (4K=2^12)
    - 20 bits for VPN (32 – 12 = 20)
    - $4MB$= 2^20 $entries$ $*4$ $Bytes\ per\ page\ table\ entry$
- Each process needs to have a page table
  - 100 processes needs 4MB * 100 = 400MB

Observation

- Many pages are unused, many PTEs are empty



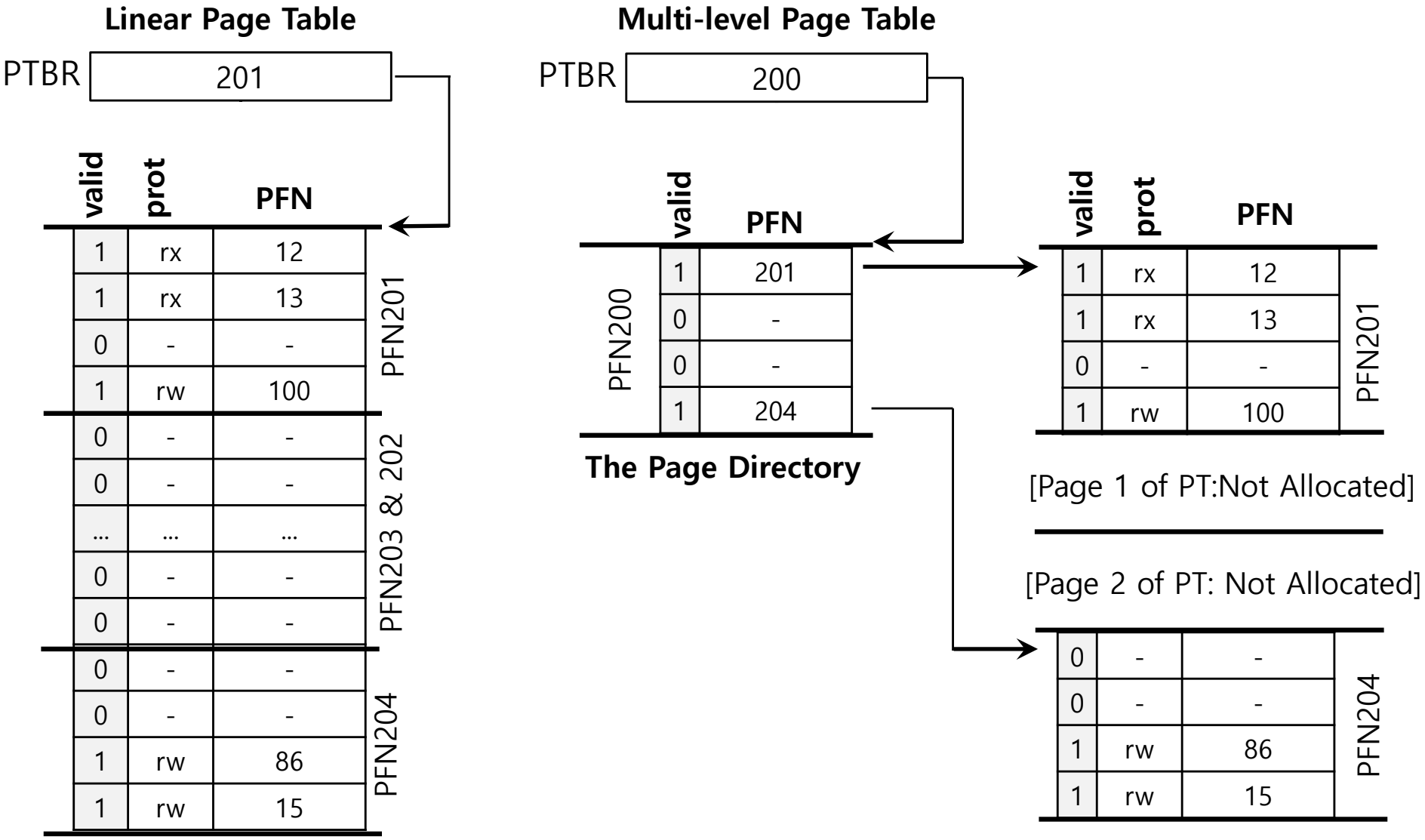
PFN	valid	prot	present	dirty
10	1	r-x	1	0
-	0	-	-	-
-	0	-	-	-
-	0	-	-	-
15	1	rw-	1	1
...	...	...	...	...
-	0	-	-	-
3	1	rw-	1	1
23	1	rw-	1	1

A Page Table For 16KB Address Space

Multi-Level Page Tables

- Paging the (linear) page table
  - Chop up the page table into page-sized units
  - If an entire page of page-table entries is invalid, don't allocate that page of the page table at all
  - To track whether a page of the page table is valid, use a new structure, called page directory
    - It consists of a number of page directory entries (PDE)
      - one PDE per page of the page table
      - PDE has a valid bit and page frame number (PFN)

Multi-Level Page Tables



Agenda

- ~~Recap~~
- ~~Translation Lookaside Buffer (TLB)~~
- ~~Multi-Level Page Tables~~

Questions?

*acknowledgement: slides include content from “Modern Operating Systems” by A. Tanenbaum, “Operating Systems Concepts” by A. Silberschatz etc., “Operating Systems: Three Easy Pieces” by R. Arpaci-Dusseau etc., and anonymous pictures from internet.