CSE 120 Discussion

Final Review

Wenjia Ouyang

03/13/2015

Announcements

- Final: Saturday, March 14
  - Tomorrow!!
  - 8:00am-11:00am
  - Peterson Hall 108
  - #2 pencil
- Sample Final on Piazza
- PLEASE fill out CAPEs !
  - To fill out the form, go to cape.ucsd.edu and click on "Fill Out Evals" on the right side of the screen.

Final Exam

- The same pattern as Midterm
  - 40~60 Multiple Choice

- It will focus much more on post-midterm material than pre-midterm (like twice as much).

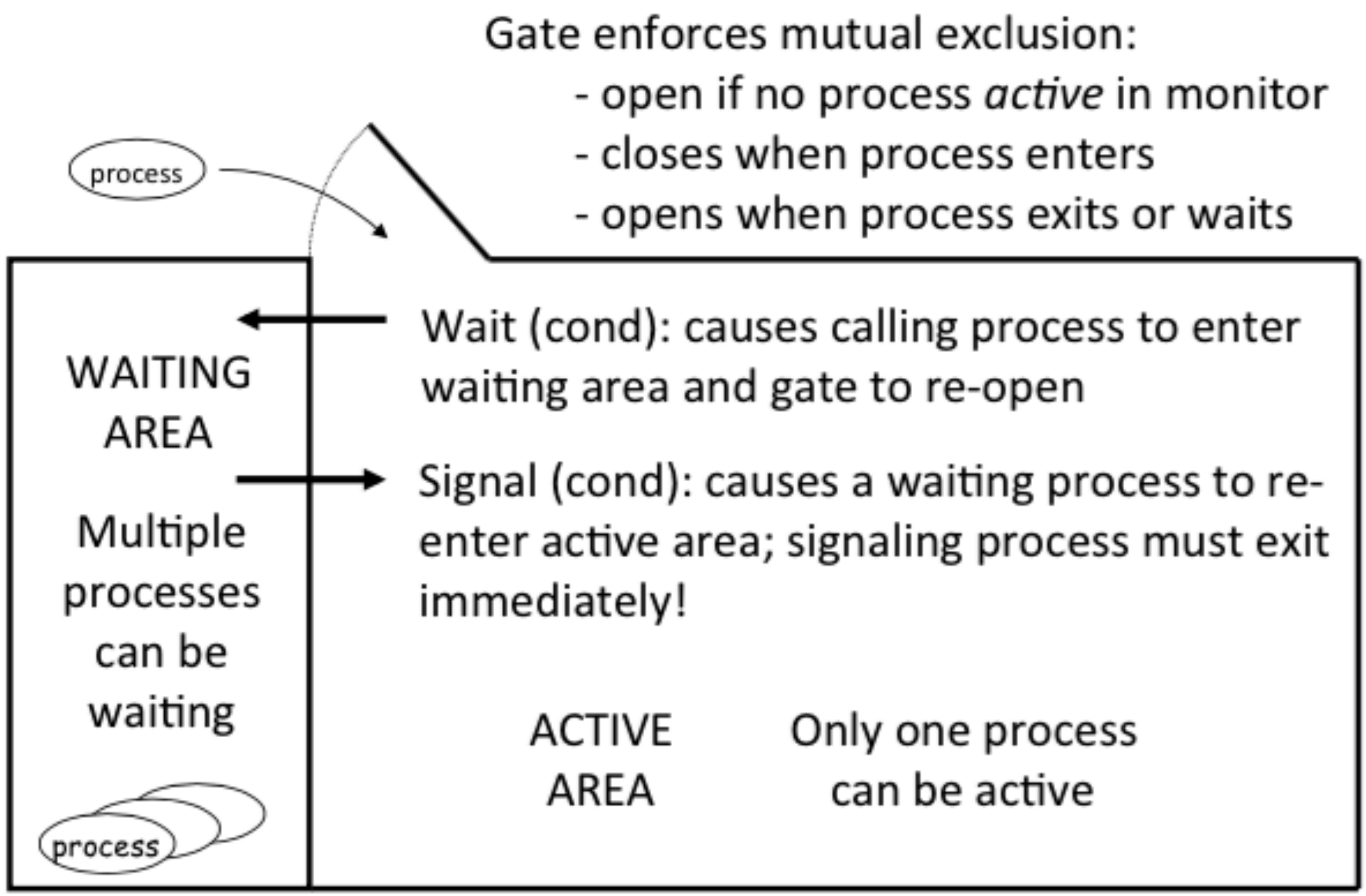
- Projects: PA3 and PA4 (with possibly a few on PA1 and PA2, but mostly the latter PAs).

Pre-Midterm

- Processes
  - What is a process?
  - Scheduling
  - Synchronization
  - Inter-process communication
  - Deadlock

Lecture 6. slide 14

How Synchronization Works



What will happen on the door if calling Signal(cond) by a process that is inside the monitor? The door will keep close!

Issues with Monitors

- Given P_1 waiting on condition c, P_2 signals c
  - P_1 and P_2 able to run: breaks mutual exclusion
  - One solution: Signal just before returning
- Condition variables have no memory
  - Signal without someone waiting does nothing
  - Signal is “lost” (no memory, no future effect)
- Monitors bring structure to IPC
  - Localizes critical sections and synchronization

Memory

- Memory management

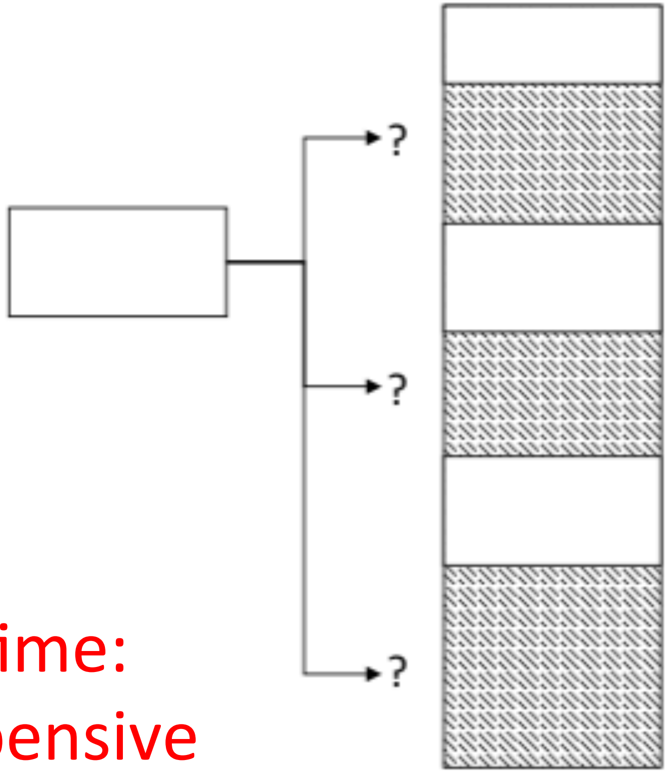
- Logical vs. Physical vs. Virtual
  - Logical Memory: What we'd like them to be
    - Segmentation and paging
  - Physical Memory: What real memory
    - Frame
  - Virtual Memory: Illusion of larger memory to support multi-programming
    - Page replacement algorithms

Memory management

- A program has to be in physical memory for it to run.
- But, since we have multiple processes running, which create and delete lot of data, *the compiler has no idea which part of physical memory is free.*
- Thus compiler generates a logical (virtual address space) for the process from 0 to N-1.

Selecting the Best Hole

- If there are multiple holes, which to select?
- Algorithms
  - First (or next) fit In PA4
  - Best fit
  - Worst fit
- So which is best?



Consider tradeoff: fit vs. search time:
Memory is cheap and time is expensive
→ Waste memory to get speed

Lecture 8. slide 21

50% Rule

- The 50% rule formula: m = n/2
  - What m is? The number of **holes**
  - What n is? The number of **allocated blocks**

- That this is *average* behavior, and not absolutely true
  - Is it possible that there can be 100 blocks and 99 holes?
    - Yes.
  - Is it also possible that there are 100 blocks and 0 holes?
    - Yes. BUT, on average, if there are 100 blocks, there will be 50 holes.

Unused Memory Rule

- $f = k/(k+2)$
  - $k = h/b$, ratio of average block-to-hole size
    - h: average size of holes
    - b: average size of allocated blocks
  - f is fraction space lost to holes

- What does it mean for k to be small, << 1 (avg block size is much larger than avg hole size)

  → f -> 0, which means no space lost to holes

- What does it mean for k to be large, >> 1 (avg block size is much smaller than avg hole size)

  → f -> 1, which means all space lost to holes

- These are *average* values

Buddy System

- Which of the following would ***not*** be a valid chunk size in the buddy system?
  A. 1MB
  B. 2MB
  C. 3MB
  D. 16MB
  Ans: C. Partition into power of 2 size chunks

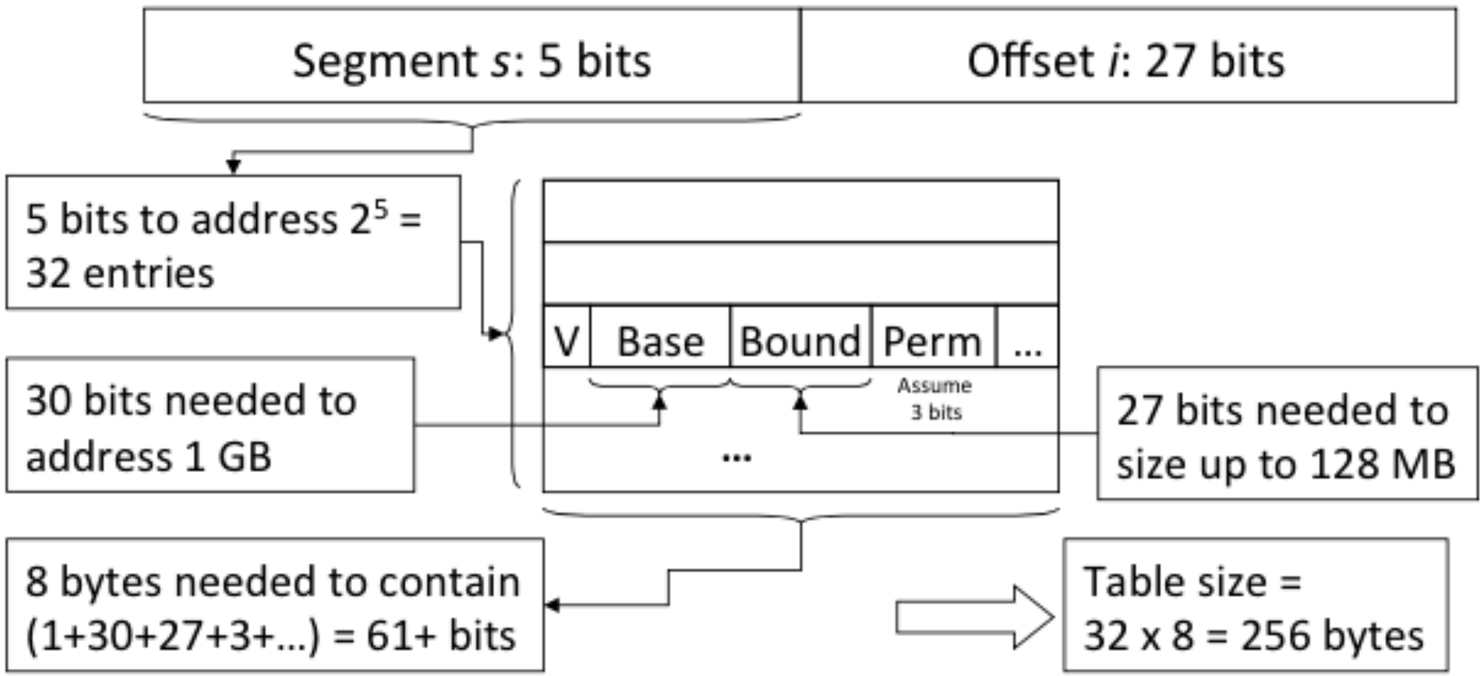
- Which data structure is used to represent the buddy system?
  – Binary tree

Lecture 8. slide 42

Segmentation and Paging

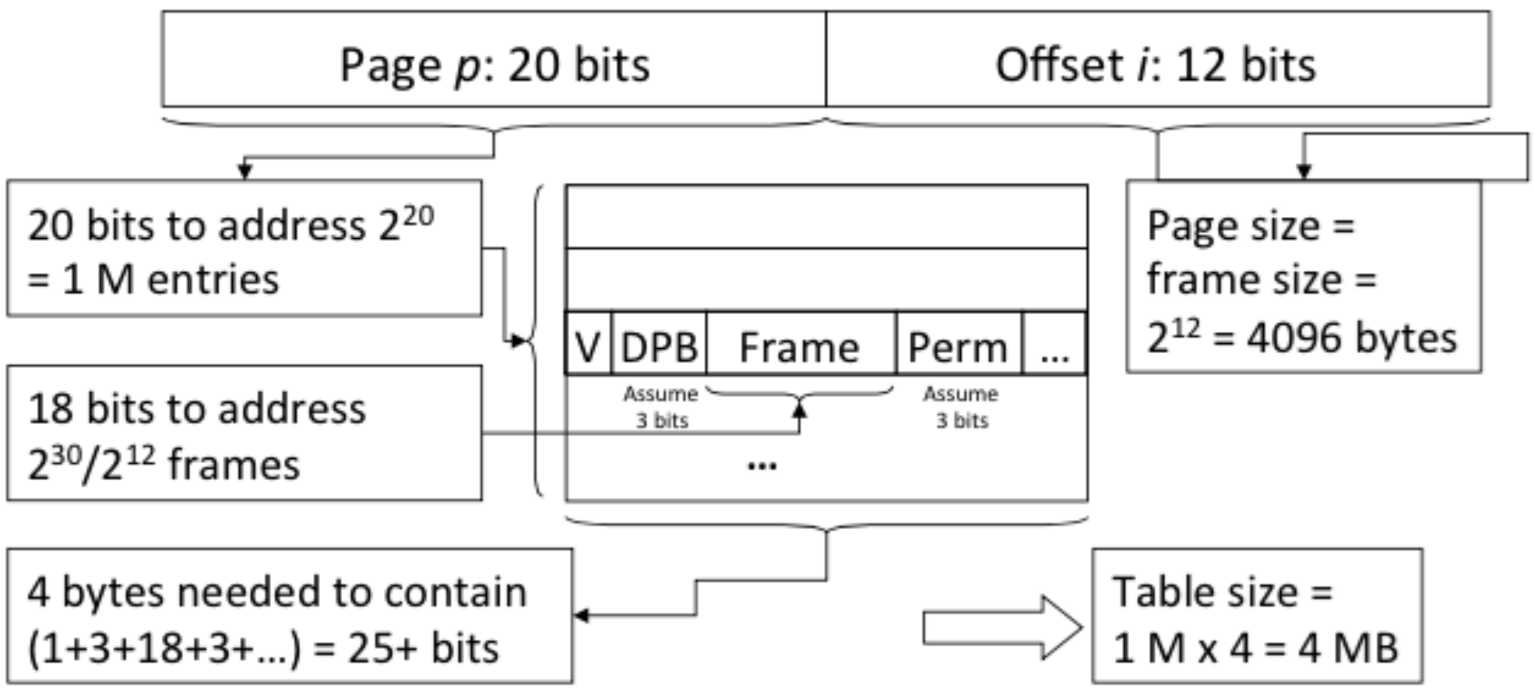
- Segments–Variable sized unit of memory
  – text, data, stack
- Pages–Fixed sized unit of memory access
- Which causes internal fragmentation, and which approach would cause external fragmentation?
  – Segmentation: External fragmentation
  – Paging: Internal fragmentation

Example of Sizing the Segment Table



- Given 32 bit logical, 1 GB physical memory (max)
  - 5 bit segment number, 27 bit offset

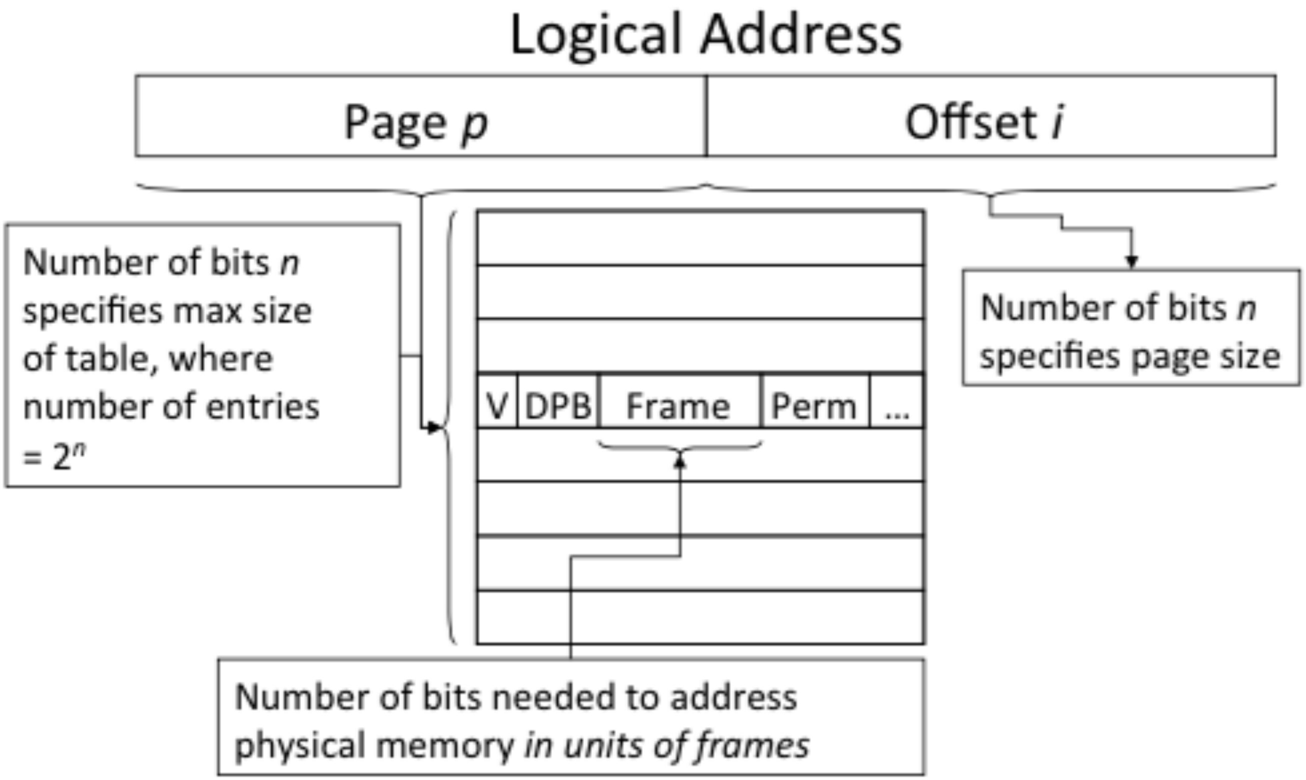
Example of Sizing the Page Table



- Given 32 bit logical, 1 GB physical memory (max)
  - 20 bit page number, 12 bit offset

- Given a 32 bit logical address, page size of 4 KB, 4 GB physical memory:
- what is the size of each entry in the page table?
- What is the total size of the page table?

Sizing the Page Table



Solution:

- Offset: # of bits to represent a page
  - Page size is 4KB → offset i = 12 bits (4KB = 2^{12})
  - Page p + offset i = logical address size
    - → Page p = 20 bits

- Frame bits = PM size/ Frame size

  Frame size = Page size

  → Frame bits = 20 bits, for 4GB/4KB = $2^{32}/2^{12}=2^{20}$

- For **each** entry of the page table, we need:

  1(V)+3(DPB)+ 20(frame)+3(permission) = 27 bits

  → 4 bytes needed for each entry.

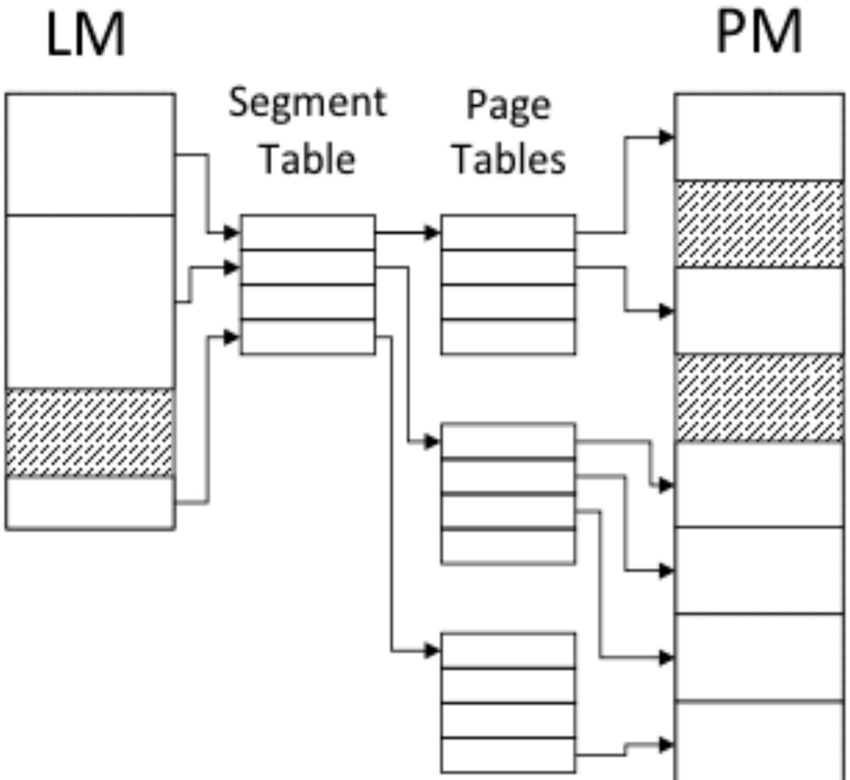
  (convert it into an integer in bytes)

- For the total size of the page table = each entry * # of entries
- # of entries = $2^{\text{page p}}$ = 2^{20}

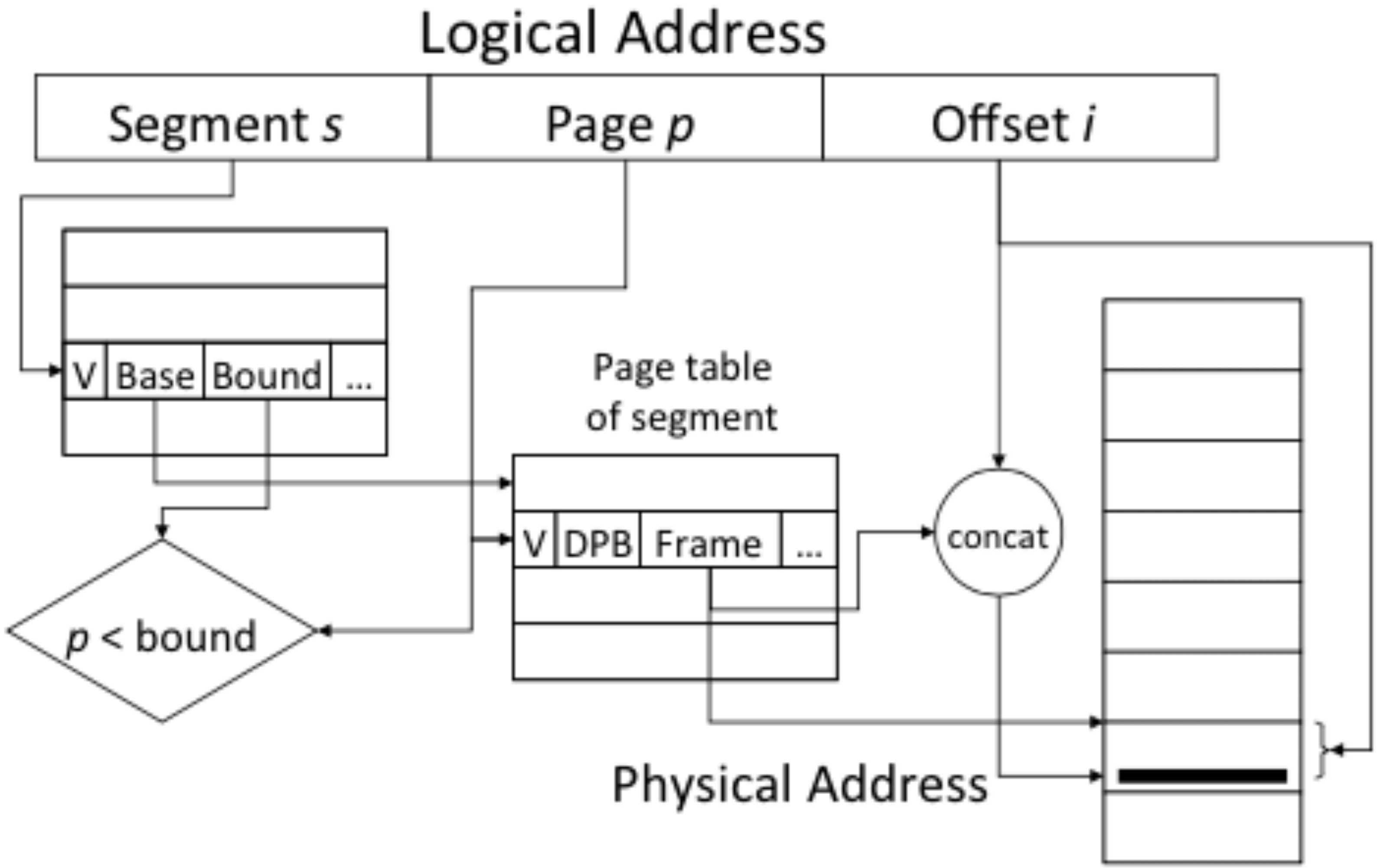
  → 4 * 2^{20} = 4MB

Combining Segments and Pages

- Logical memory composed of segments
- Each segment composed of a set of pages
- Segment table: maps each segment *s* to a page table
- Page tables (like before)



Segment/Page Address Translation



Practice Final #7

TLB – Translation look aside buffer

- Used for logical to physical address lookup on chip.
  - To speed up the translation, a TLB is a separate piece of **hardware** that is used and operates in **parallel** (and works with either paging or segmentation).

- Should be really fast.

- A TLB is normally implemented as a small fully associative cache.

Virtual memory

- Pages are stored on disk by default, and brought into physical memory only when needed.

- This gives us opportunity to allow pages from multiple process to be in physical memory

- Page Fault means page not present in physical memory, and we need to get is from the disk.
  - If valid bit is off, page fault
  - Trap into kernel

Lecture 10, slide 6

Sample Contents of Page Table Entry

Valid	Ref	Mod	Frame number	Prot: rwx

- What will be (V, R, M) bits in case of a memory read of the page?
  - (1, 1, 0)
- Is (1, 0 , 1) a valid combination?
  - No
- Is (0, 1, 1) a valid combination?
  - No

Page Replacement Algorithm

- 1. FIFO
- 2. OPT
- 3. LRU
- 4. Clock Algorithm:
  - Who set the reference bit to 0?
    - OS
  - Who set the reference bit to 1?
    - Hardware

Denning's Working Set Model

- The working set W(t, D), which of following is ***not*** correct?
- A. The set of pages referenced during the last D time units
- B. Working set is a local replacement policy
- C. Compare to Clock, it is easier to implement

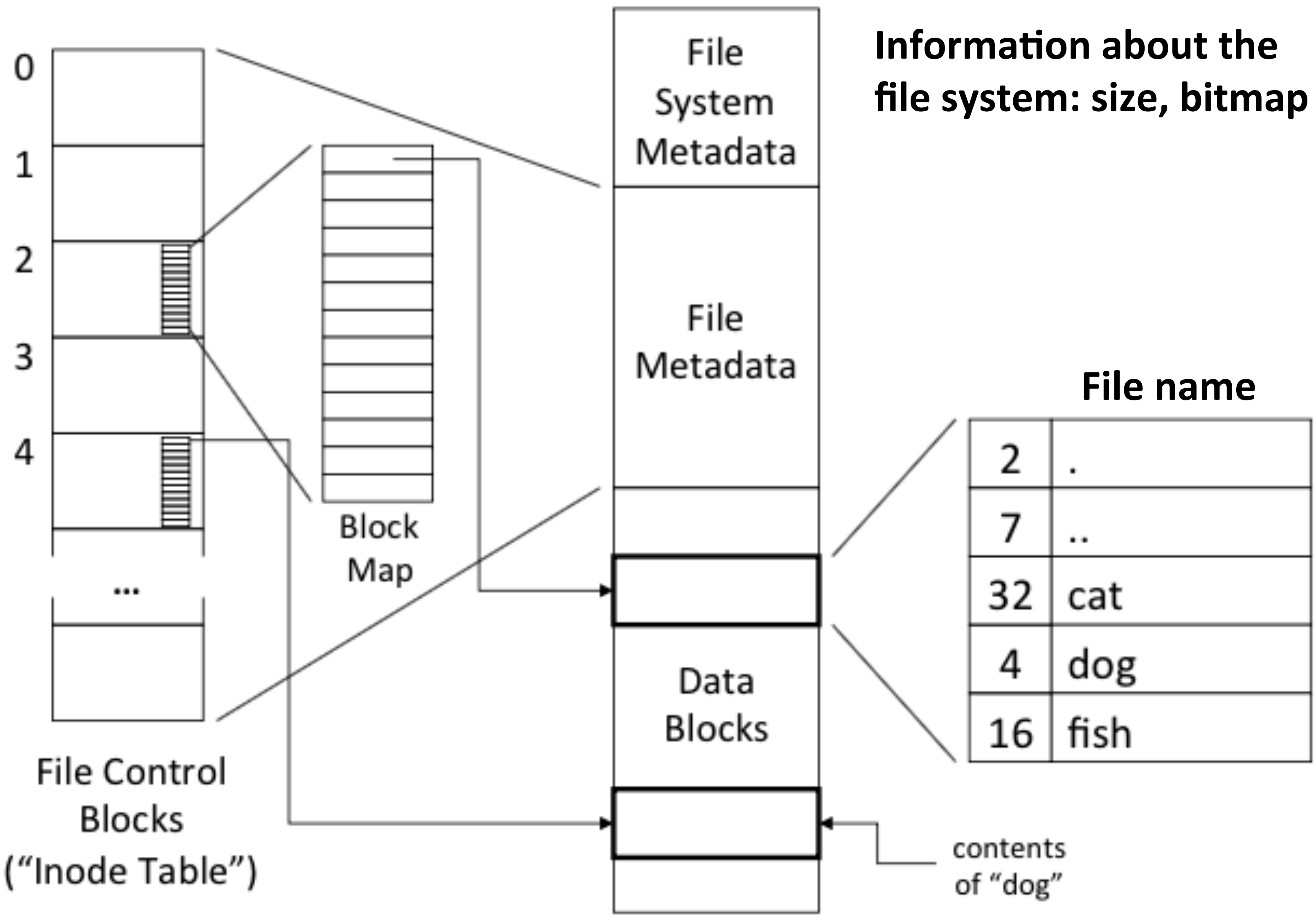
- Ans: C.
- It is difficult to implement. Must timestamp pages in working set, and must determine if timestamp older than t-Δ. Also, how should Δ be determined?

Lecture 11

File Systems

- File system abstraction
- File system structure
- Block allocation and management
- Block cache

The Big Picture



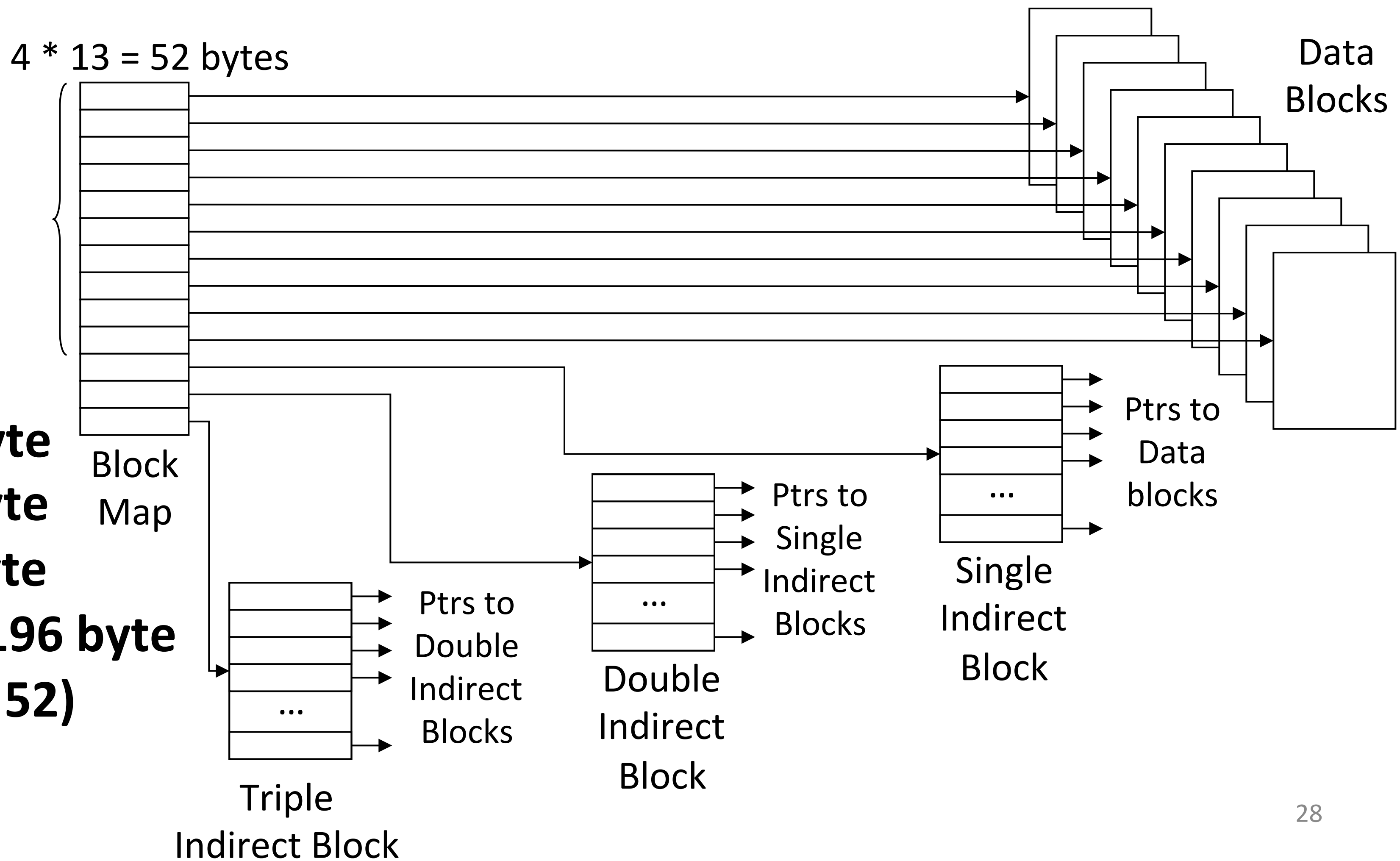
Unix Block Map

- Fixed size
- Part of the file metadata
- Array of pointers to data blocks.
  13 pointers:
  - 10 direct: references 10 data blocks
  - 1 singly-indirect: references N data blocks
  - 1 doubly-indirect: references N^2 data blocks
  - 1 triply-indirect: references N^3 data blocks

(N = Block Size / Pointer Size =1KB/4 Byte = 256)

We want to store a 5 KB file. How many bytes of metadata will we need for our block map?

Assume that Block size: 1 KB (2^{10}) and Pointer size: 4 bytes



A: 20 byte

B: 40 byte

C: 52 byte

D: 262,196 byte (256K + 52)

Answer: C. 52 byte

- A 5 KB file is "broken up" into 5 blocks because each block is of size 1KB. So we need 5 pointers for the 5 blocks.

  →Is A the right answer?

  4 byte *5 = 20 bytes of metadata.

- ***No!***
  - Because you have to initialize **that whole block map(13 pointers)** even if you are only use a part of it. You may think 20 of those bytes of meta data are ACTUALLY being used, **BUT you NEED 52 bytes to represent those 5 blocks.**

We want to store a **100 KB** file. How many bytes of metadata will we need for our block map?

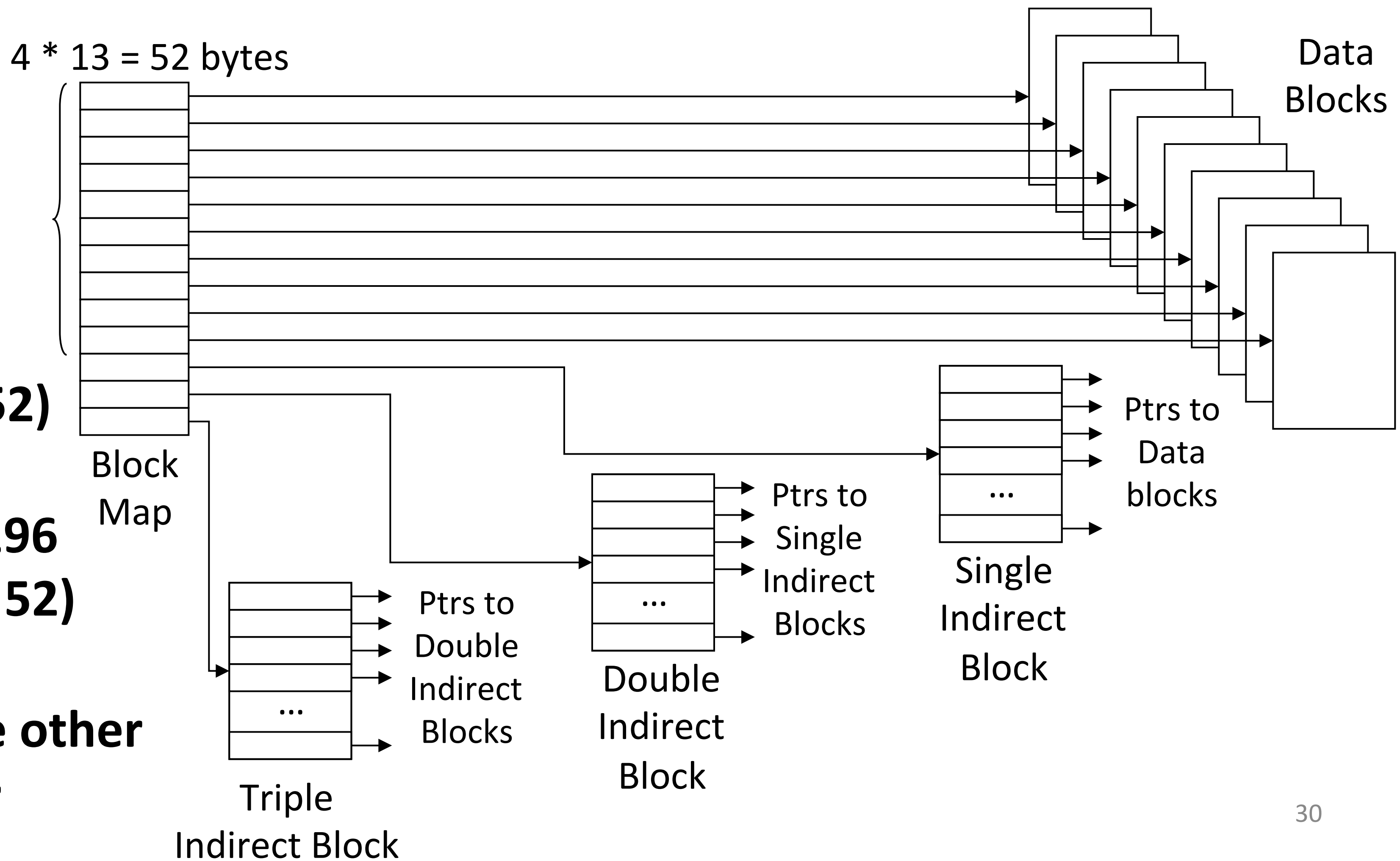
Assume that Block size: 1 KB (2^{10}) and Pointer size: 4 bytes

A: 52

B: 1076 (1KB + 52)

C: 262,196 (256K + 52)

D: some other number



Answer: B. 1KB+52

Assume that Block size: 1 KB (2^{10}) and Pointer size: 4 bytes

- 100 KB file → broken up into 100 data blocks
- We can use the 10 direct pointers to point 10 data blocks

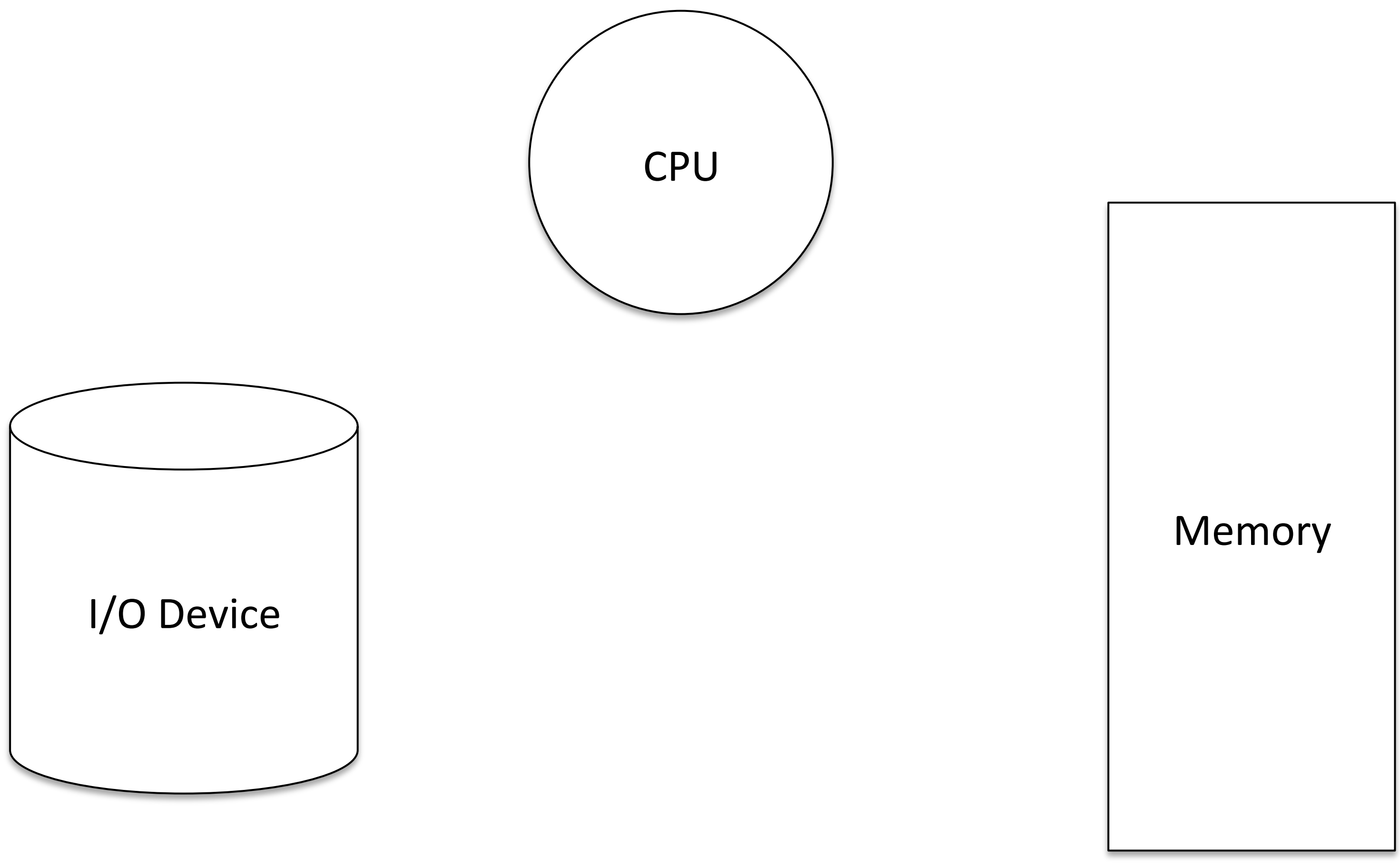
  →but we are still left with 90 unreferenced blocks.
- So we need to use a singly-indirect to references N=256 data blocks. This is enough to hold those 90 pointers.
- 4 byte * 13 (block map) + 1024 byte (first level)

  = 52 byte + 1024 byte = 52 byte +1KB

I/O

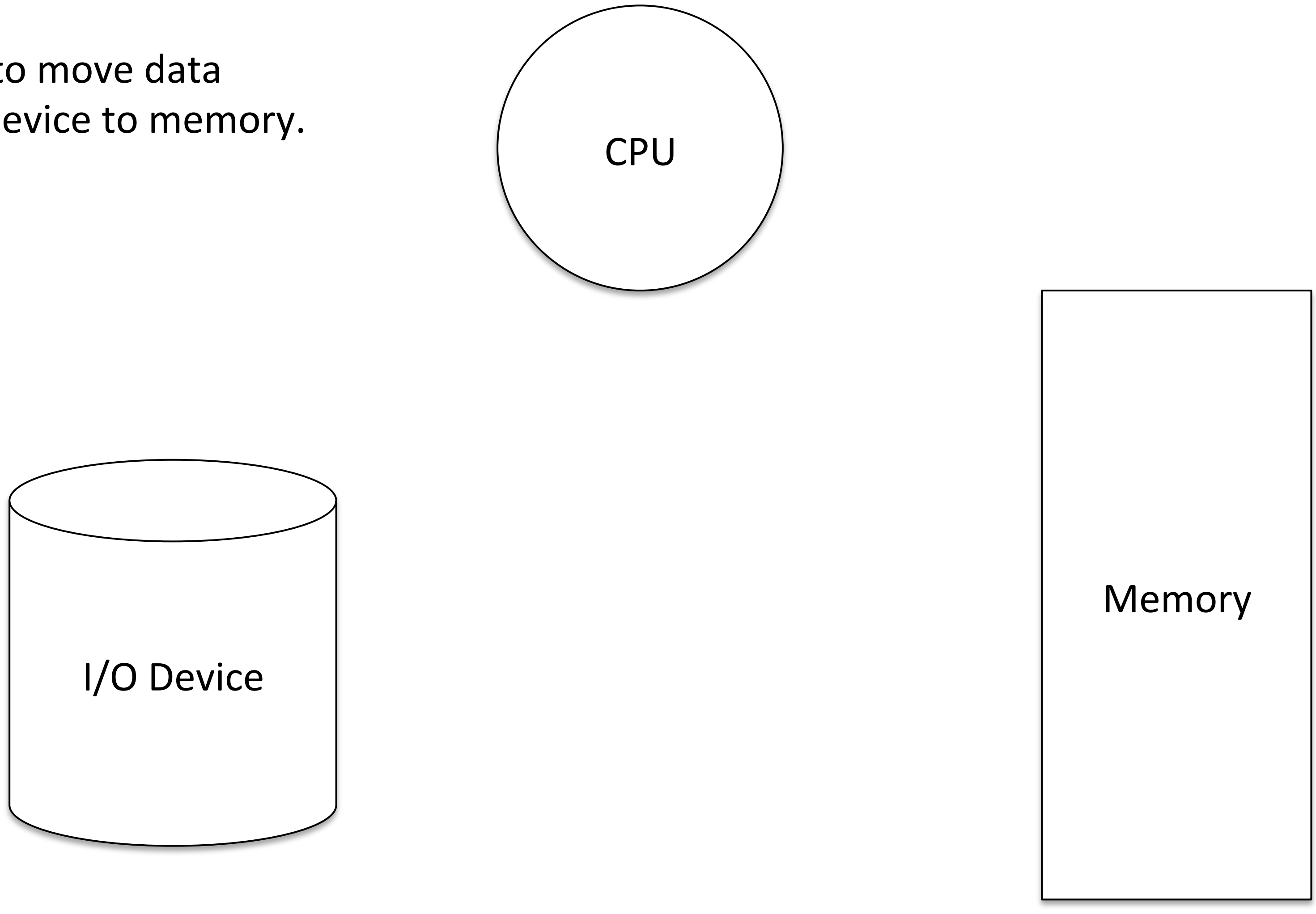
- Structure of I/O system software
  - Functionality of layer
  - Interoperation
- Device drivers
- Buffering
  - Why (and why not) buffer
  - Where to buffer

PIO (programmed IO) vs. DMA (direct memory access)



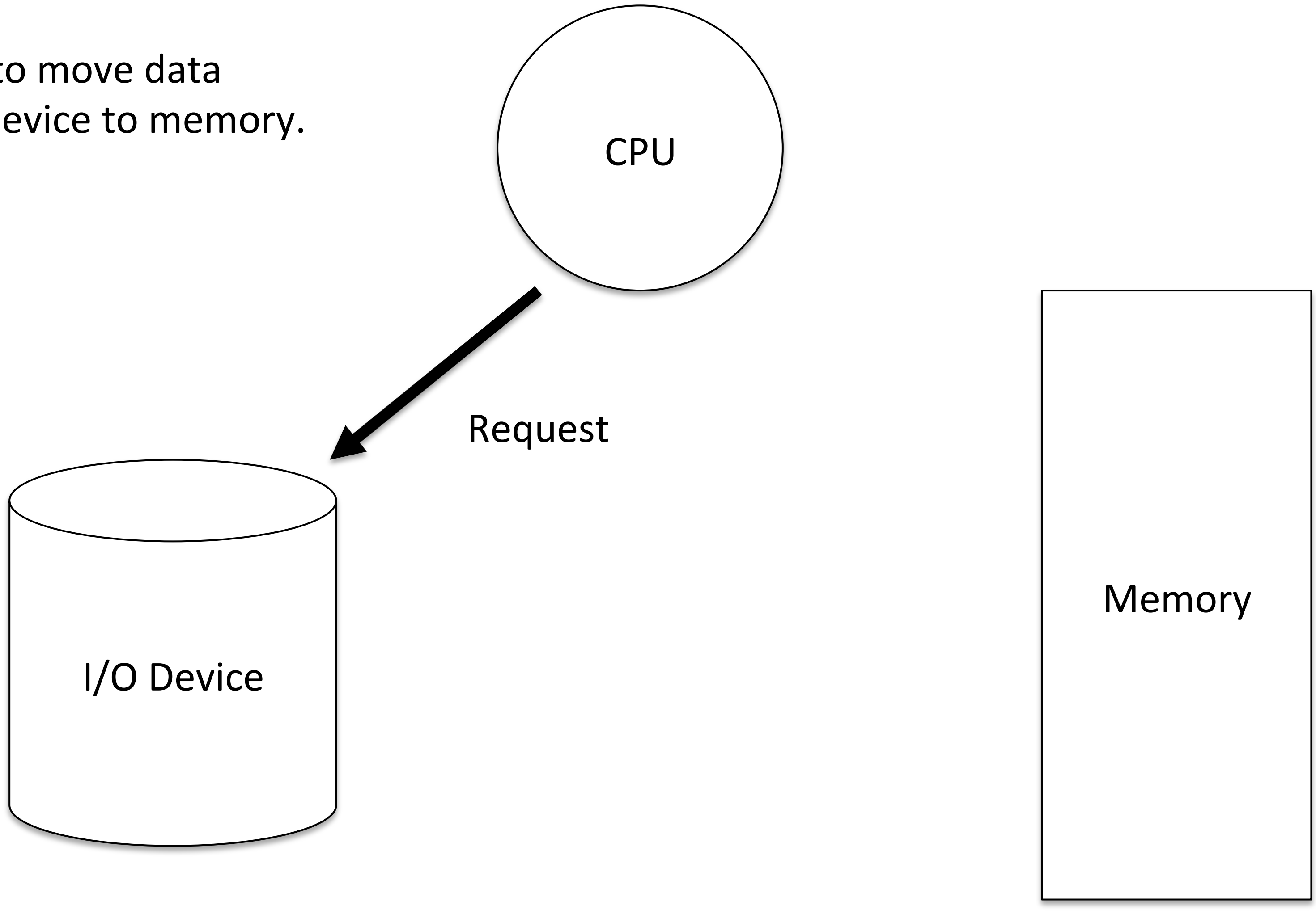
PIO vs. DMA

Want to move data
from device to memory.



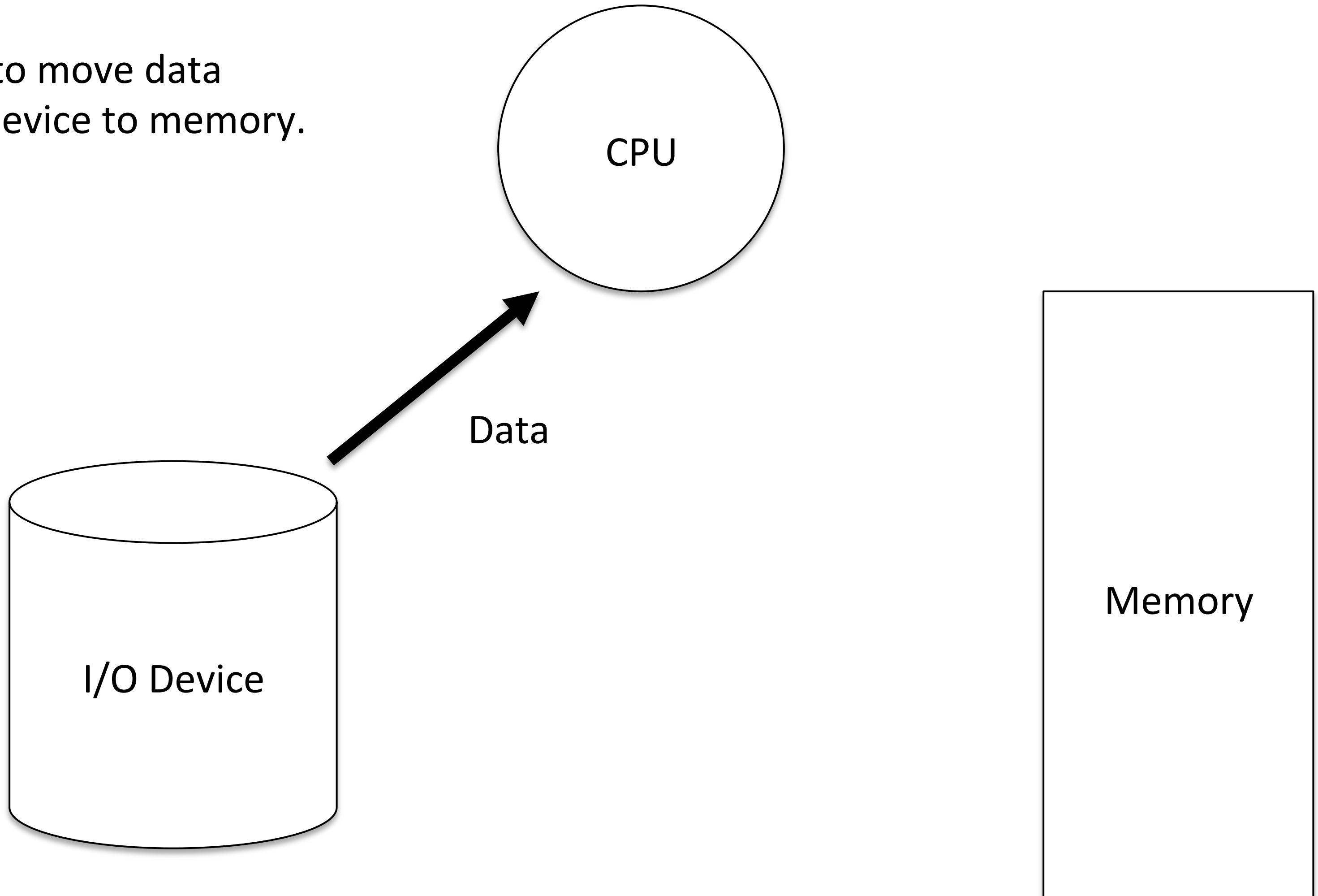
PIO vs. DMA

Want to move data
from device to memory.



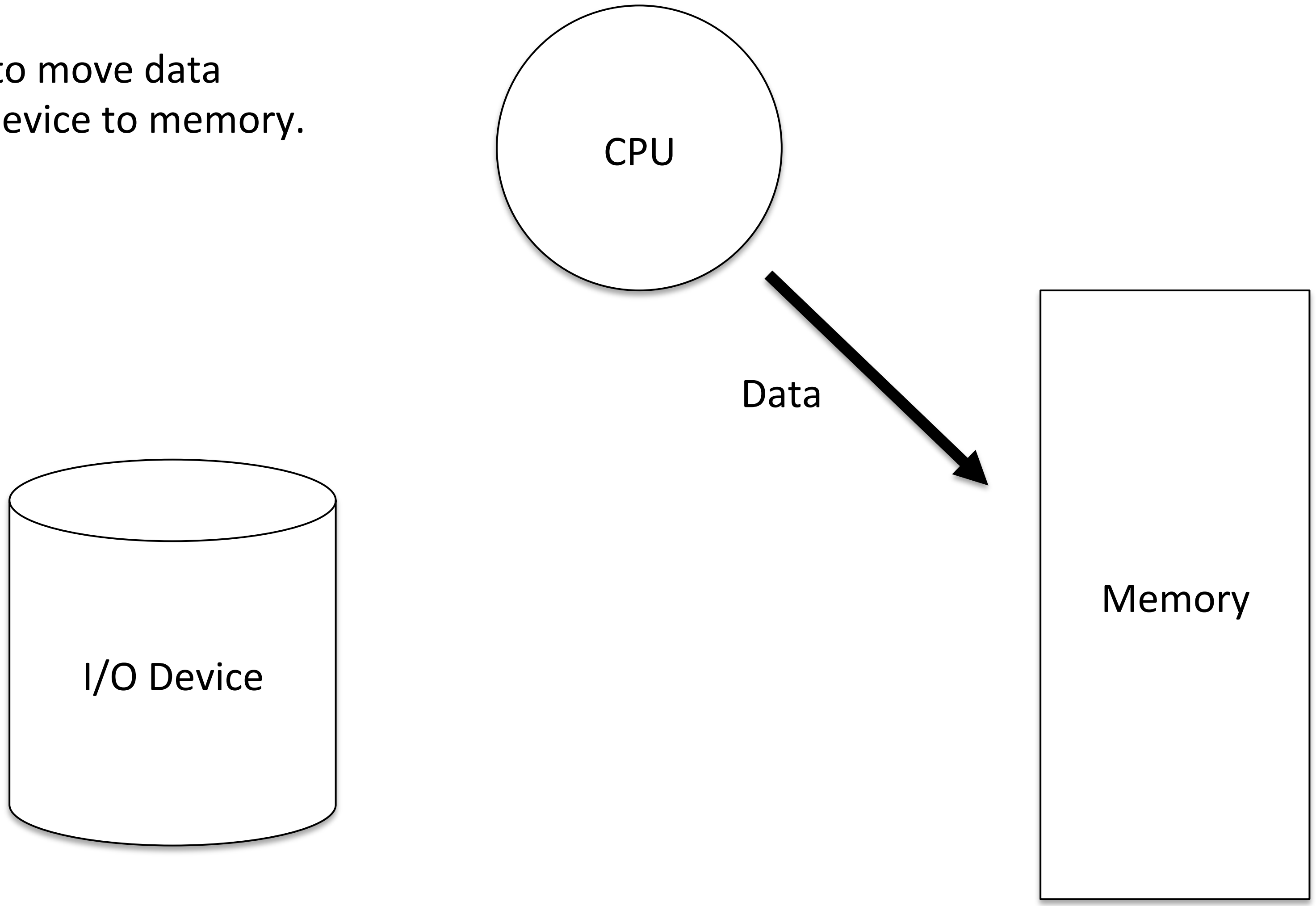
PIO vs. DMA

Want to move data
from device to memory.



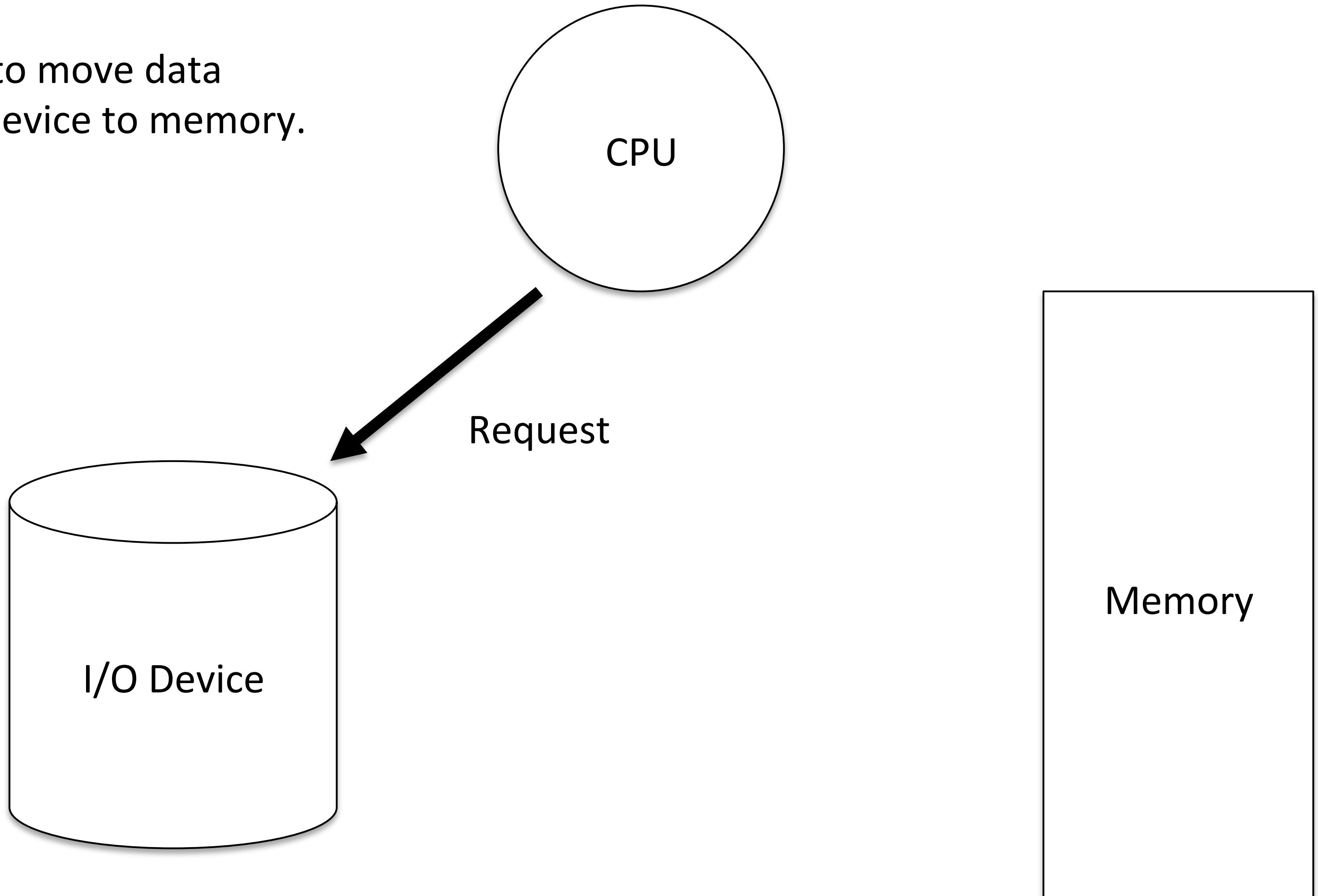
PIO vs. DMA

Want to move data
from device to memory.



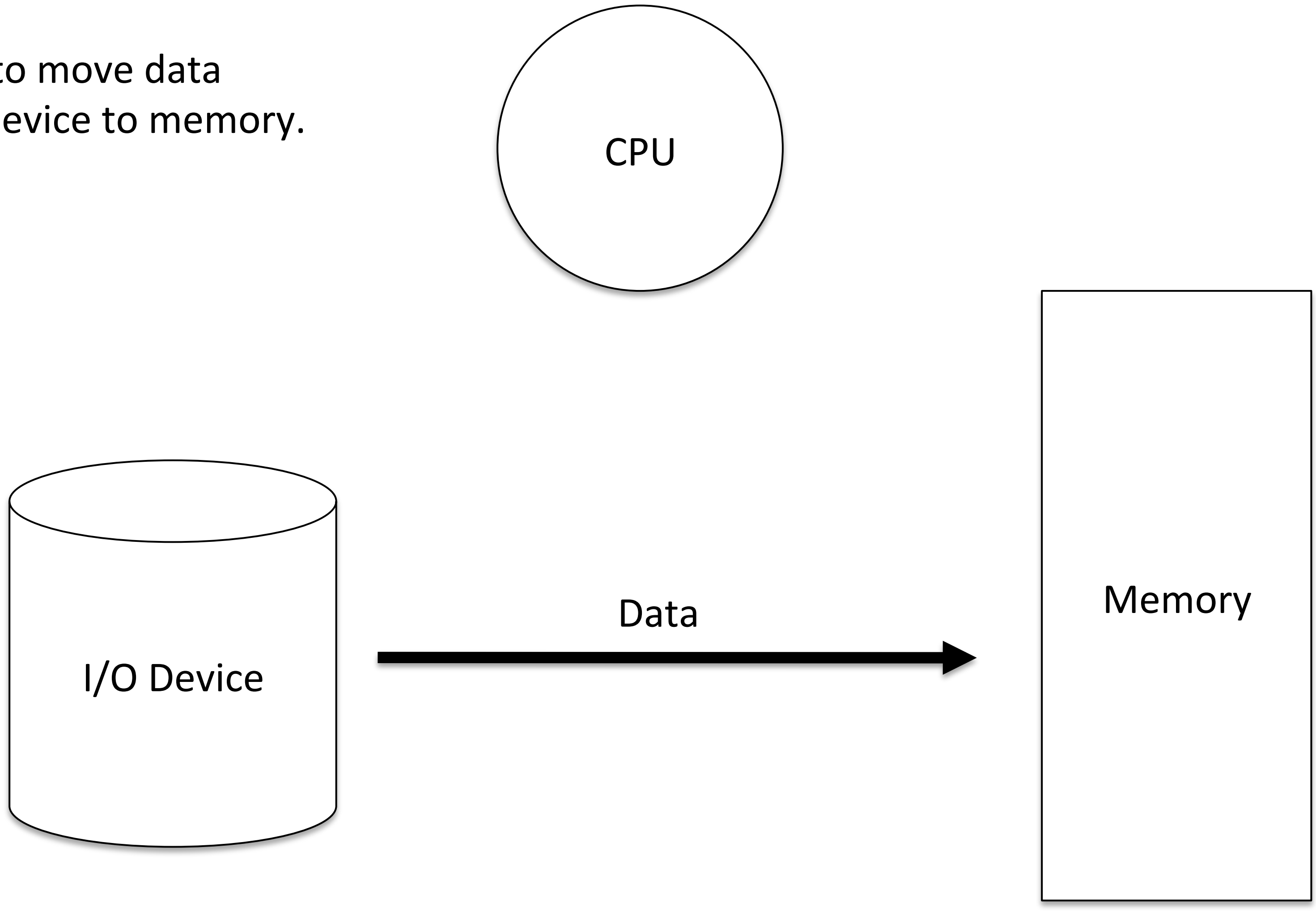
PIO vs. DMA

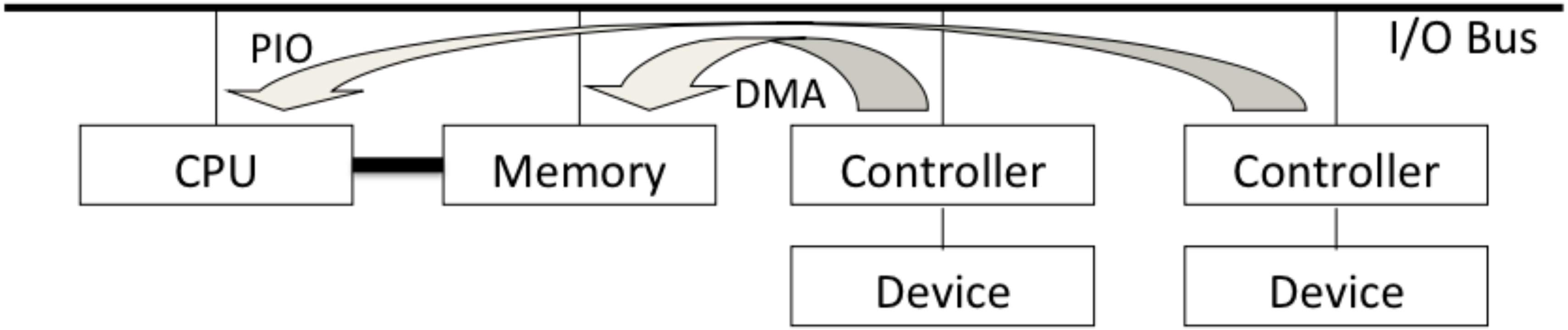
Want to move data
from device to memory.



PIO vs. DMA

Want to move data
from device to memory.





- PIO
  - CPU involved in each byte/word of I/O access
  - If device is slow, CPU will be wasted ---busy waiting
  - Simpler, preferred for low-volume transfers (Keyboard)

- DMA
  - Move data between controller and memory
  - Faster, preferred for large transfers (Disk)

How does OS get data from the IO device

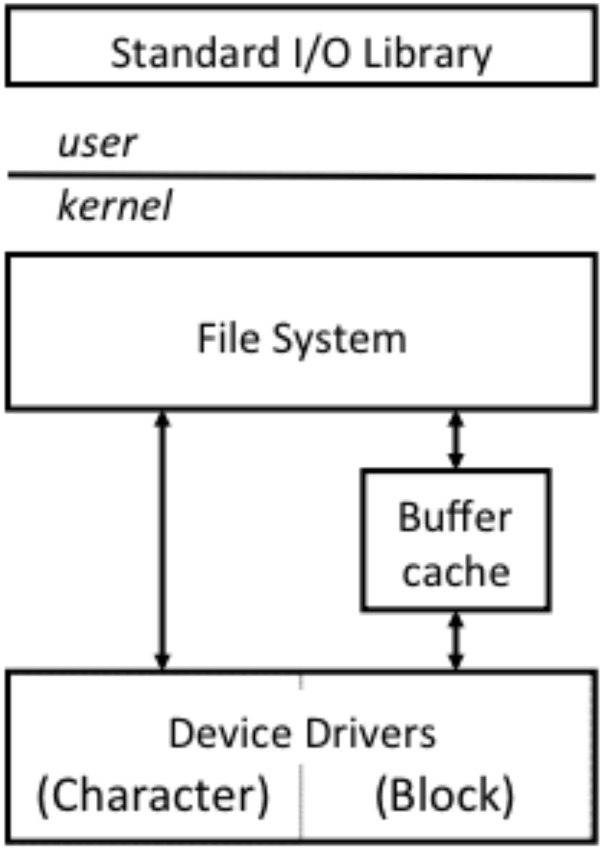
- Two options:
  - Polling: Ask device repeatedly if data is ready
    - Useful when device would be supplying a continuous stream of data
  - Interrupt driven: Device signals data is ready
    - Useful when the input is not continuous

Lecture 12: slide 24

Example: UNIX I/O Model

- Uses file system interface
- stdio.h: C standard I/O library
- Block devices
  - Fixed-size blocks
  - Randomly addressable
  - Uses buffer cache
- Character devices
  - Variable sequence of bytes
  - For non-block devices

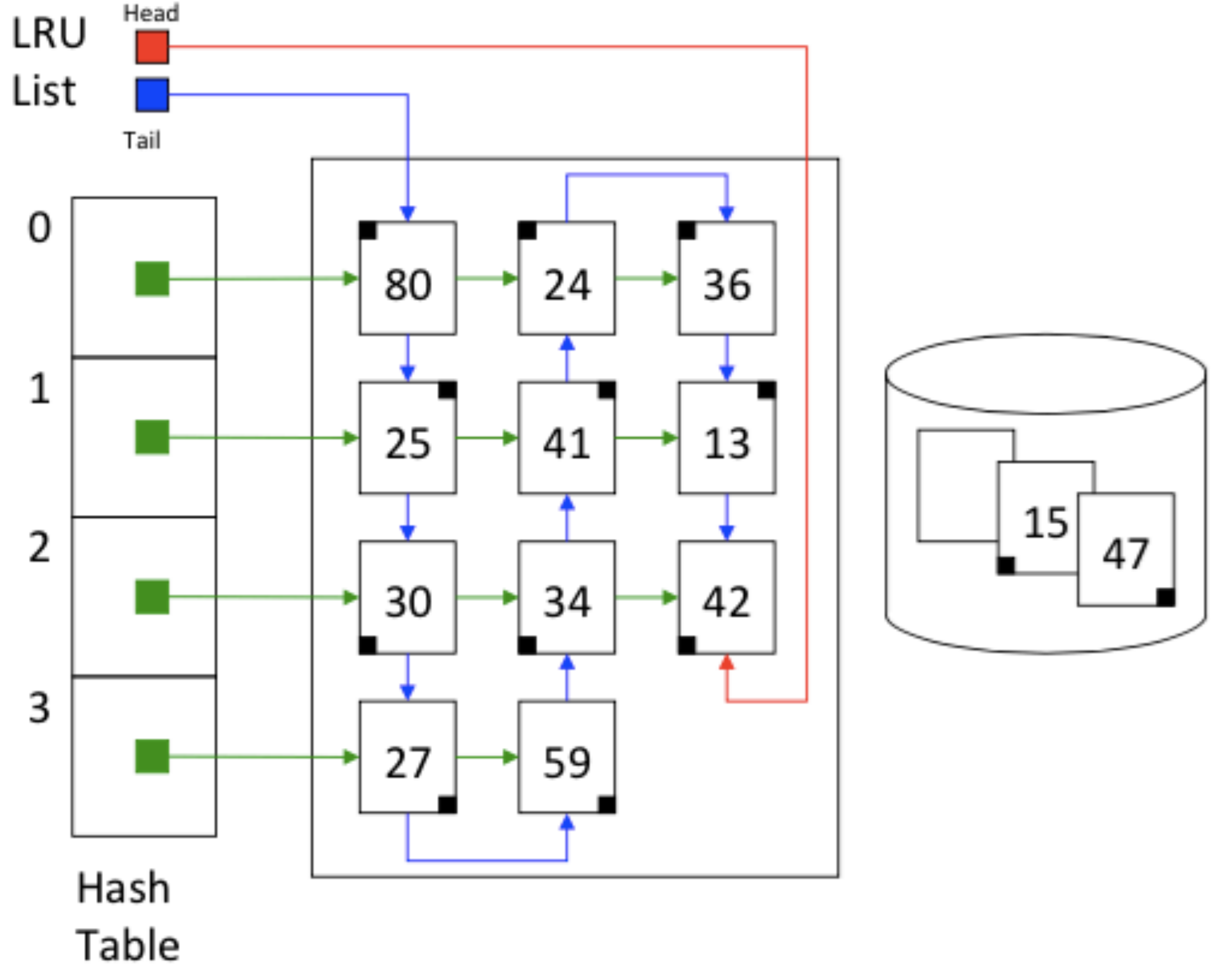
Private buffer keep in user space



Searching the Buffer Cache

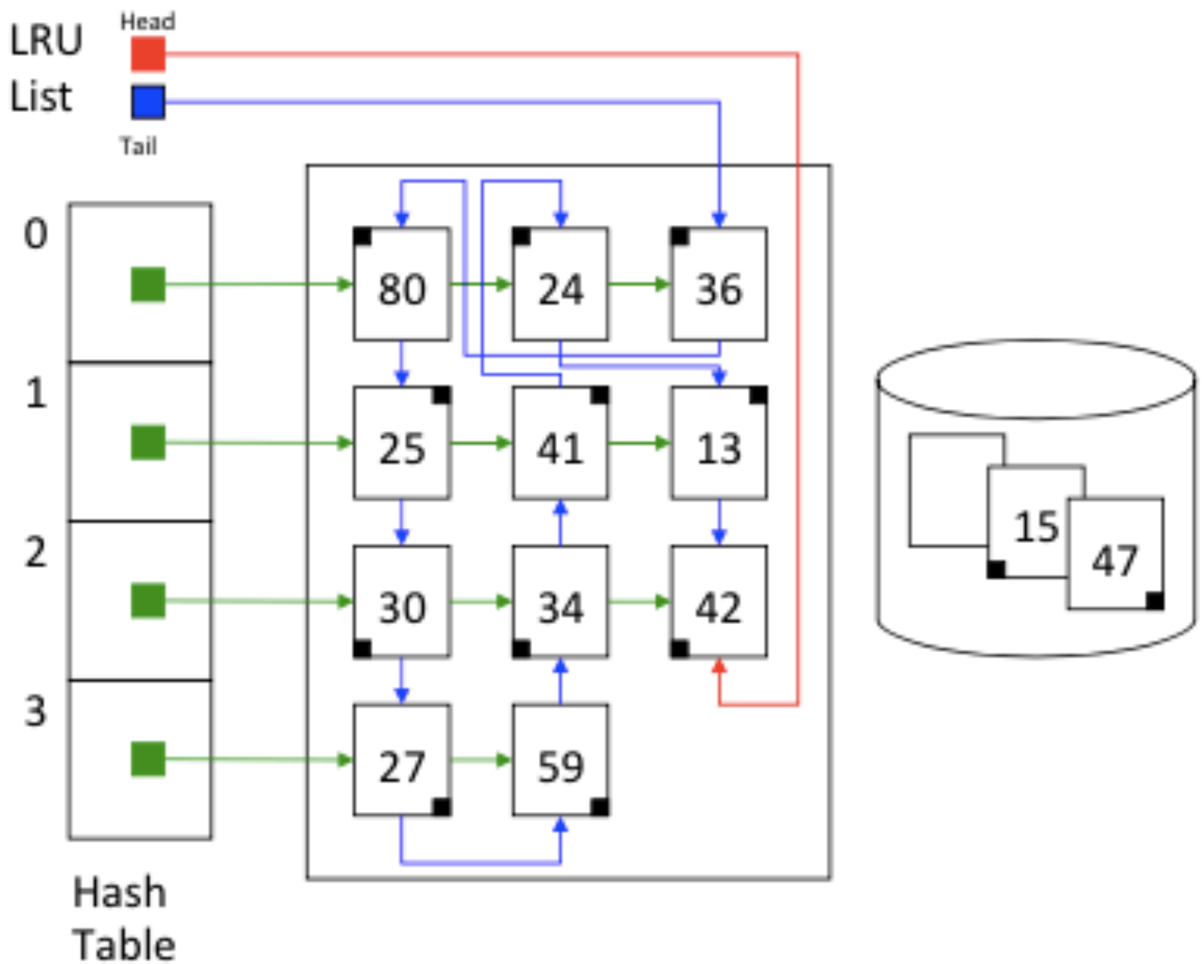
- Read (36)
- 36%4 = 0
- Search list 0 for 36
- Cache hit!

Are we done?



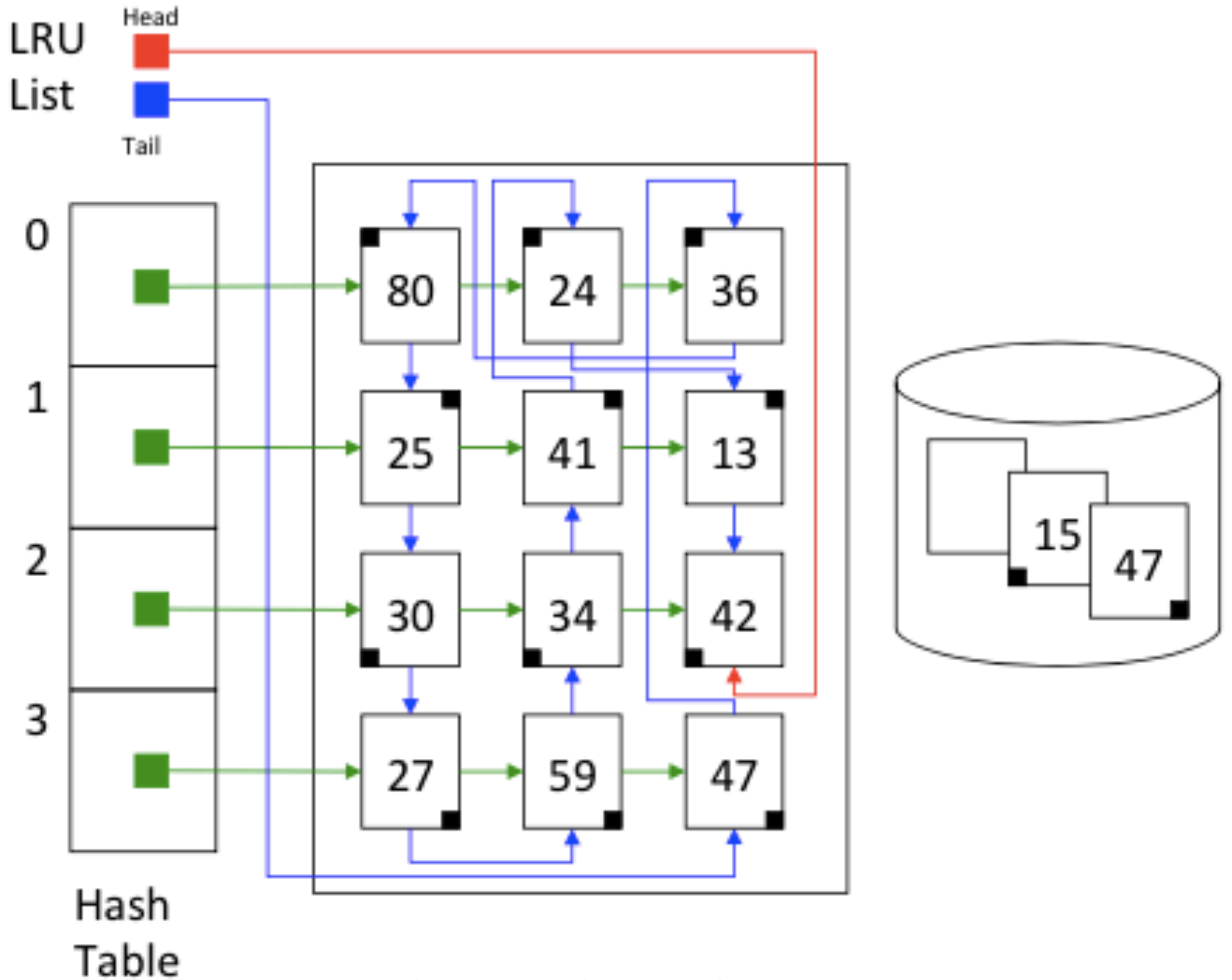
Searching the Buffer Cache

- Read (36)
- 36%4 = 0
- Search list 0 for 36
- Cache hit!
- *Update LRU list*



Searching the Buffer Cache

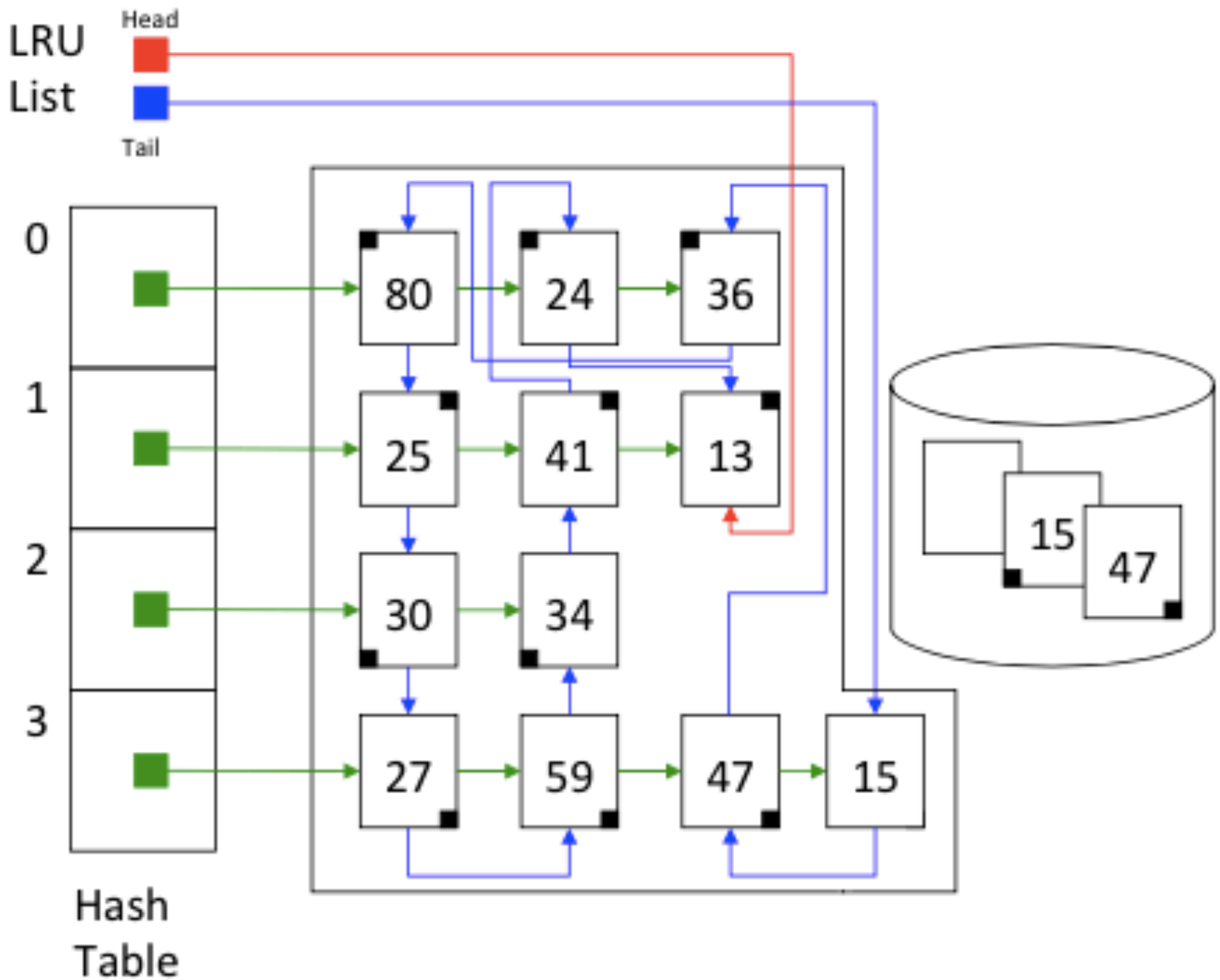
- Read (15)
- 15%4 = 3
- Search list 3 for 15
- Cache miss!



What should we do now?

Searching the Buffer Cache

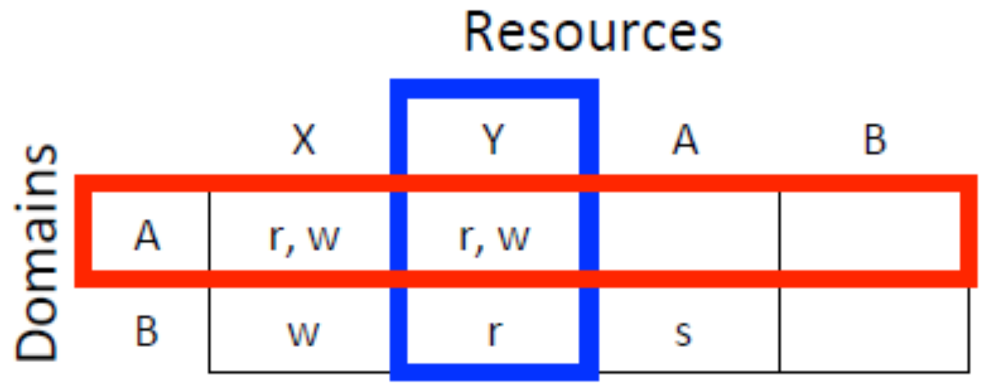
- Read (15)
- 15%4 = 3
- Search list 3 for 15
- Cache miss!
- *Remove 42*
- *Retrieve 15*



- Protection
- Security
- ~~Networks~~
- ~~Distributed Systems~~

Lecture 13: slide 1-25 only

Access Control Lists and Capabilities Lists



- Access Control Lists

  owner/group/world

  – For each resource, list (domain, permissions) pairs
- Capability Lists
  – For each domain, list (resource, permissions) pairs

  X,Y

ACLs vs. Capabilities

- Access control lists (ACLs)
  - Slow lookup
  - Easy to manage (and revoke)
  - The rwx mechanism in Unix
- Capabilities
  - Fast access
  - Hard to manage (revoke)
  - File descriptor returned after a open system call

Public Key vs. Secret Key

- Secret key (symmetric)
  - Same key K is used to encrypt and decrypt
  - Operates fast
  - Difficult to distribute keys

- Public key (asymmetric)
  - Different keys to encrypt and decrypt
  - Time consuming operation
  - Convenient for key distribute

Any question?

GOOD LUCK!