I/O and Storage:
Flash SSDs

CS 571: Operating Systems (Spring 2020)

Lecture 10a

Yue Cheng

Some material taken/derived from:

- Wisconsin CS-537 materials created by Remzi Arpaci-Dusseau.

Disk vs. Flash

Disk Overview

- I/O requires: seek, rotate, transfer
- Inherently:
  - Not parallel (only one head)
  - Slow (mechanical)
  - Poor random I/O (locality around disk head)
- Random requests each taking ~10+ ms

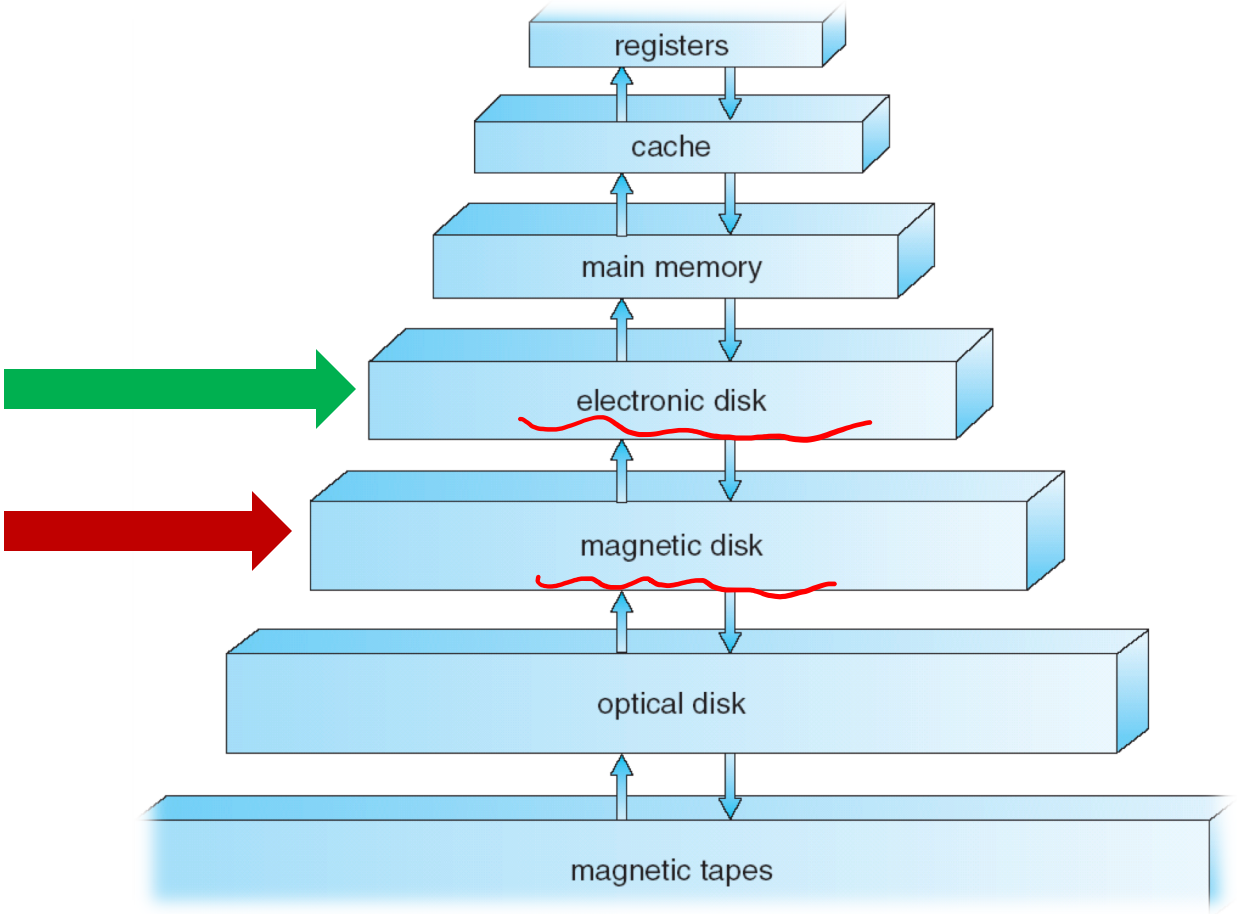
Flash Overview

- Hold charge in cells. No moving (mechanical) parts!
- Inherently parallel!
- No seeks!

Storage Hierarchy Overview

Flash:
Smaller capacity
Faster accesses

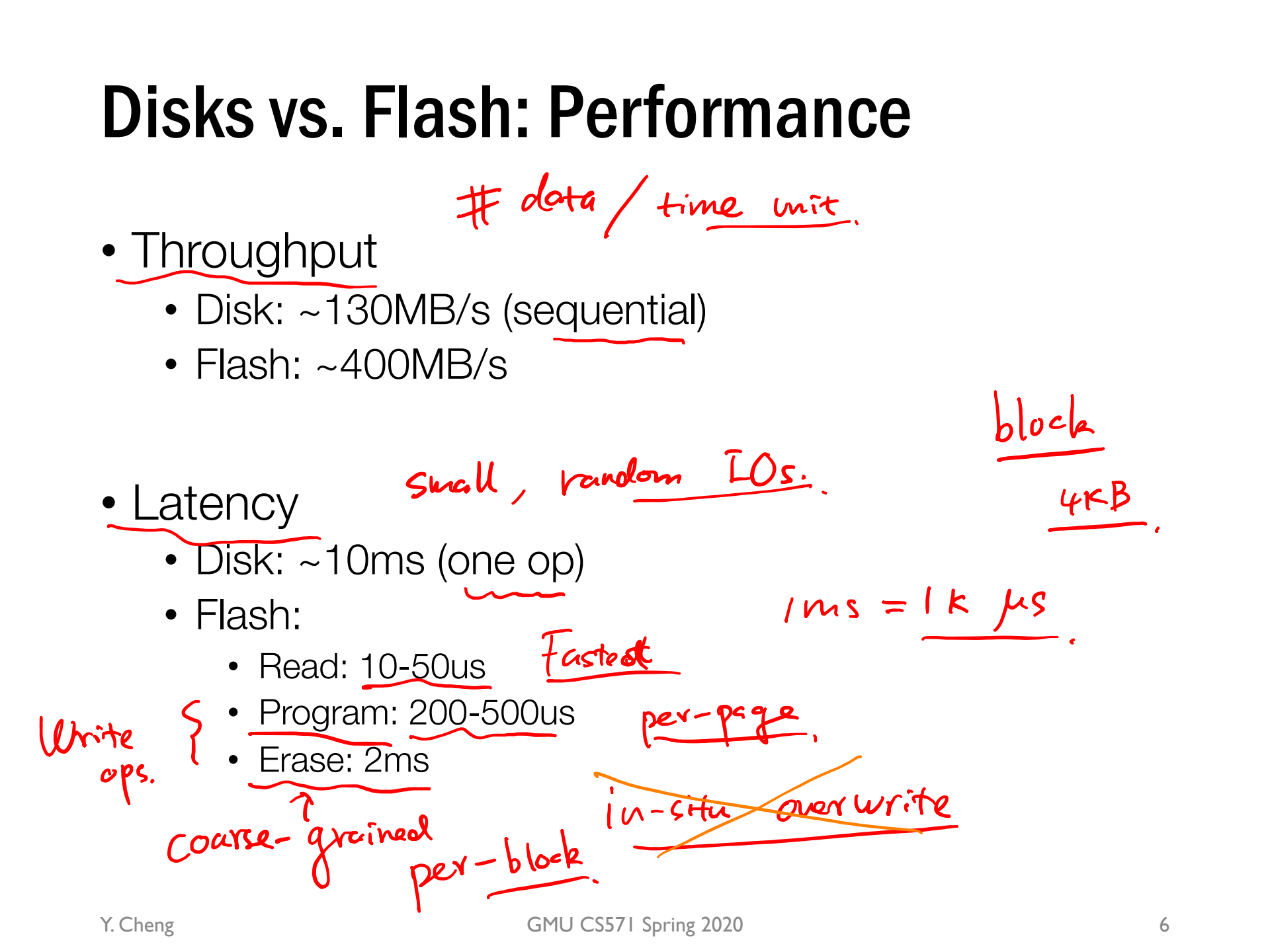
HDD:
Larger capacity
Way slower accesses



Disks vs. Flash: Performance

- Throughput
  - Disk: ~130MB/s (sequential)
  - Flash: ~400MB/s

- Latency
  - Disk: ~10ms (one op)
  - Flash:
    - Read: 10-50us
    - Program: 200-500us
    - Erase: 2ms



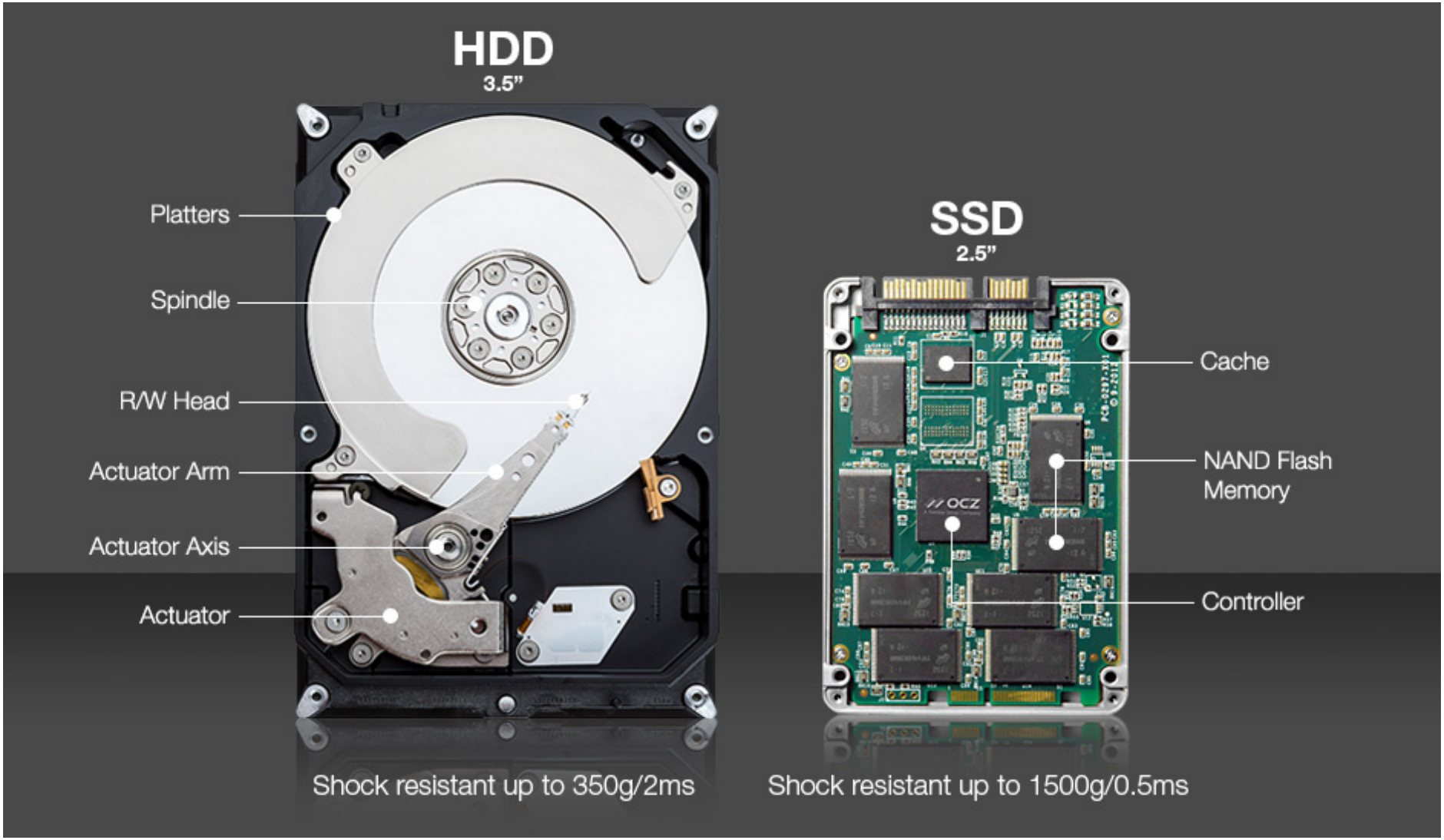
Disks vs. Flash: Performance

- Throughput
  - Disk: ~130MB/s (sequential)
  - Flash: ~400MB/s

- Latency
  - Disk: ~10ms (one op)
  - Flash:
    - Read: 10-50us
    - Program: 200-500us
    - Erase: 2ms

Types of write, more later…

Disks vs. Flash: Internal



* https://www.enterprisestorageforum.com/storage-hardware/ssd-vs-hdd.html

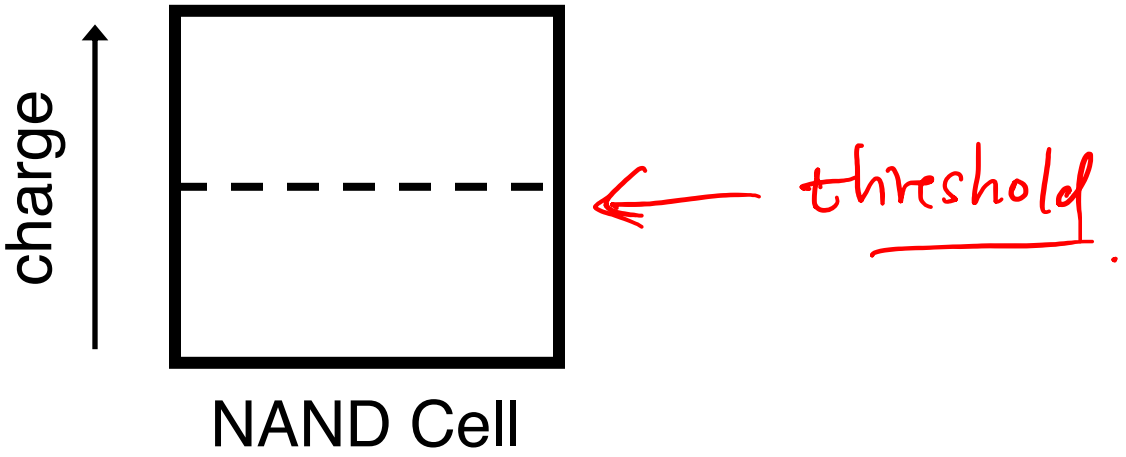
Disks vs. Flash: Summary

	SSD	HDD
Price	$0.25-$0.27 per GB average	$0.2-$0.03 per GB average
Lifespan	30-80% test developed bad block in their lifetime	3.5% developed bad sectors comparatively
Ideal for	High performance processing Residing in APA or Tier 0/1 media in hybrid arrays	High capacity nearline tiers Long-term retained data
Read/write speeds	200 MB/s to 2500 MB/s	up to 200 MB/s
Benefits	Higher performance for faster read/write operations and fast load times	Less expensive Mature technology and massive installed user base
Drawbacks	May not be as durable/reliable as HDDs Not good for long-term archival data	Mechanical components take longer to read-write than SSDs

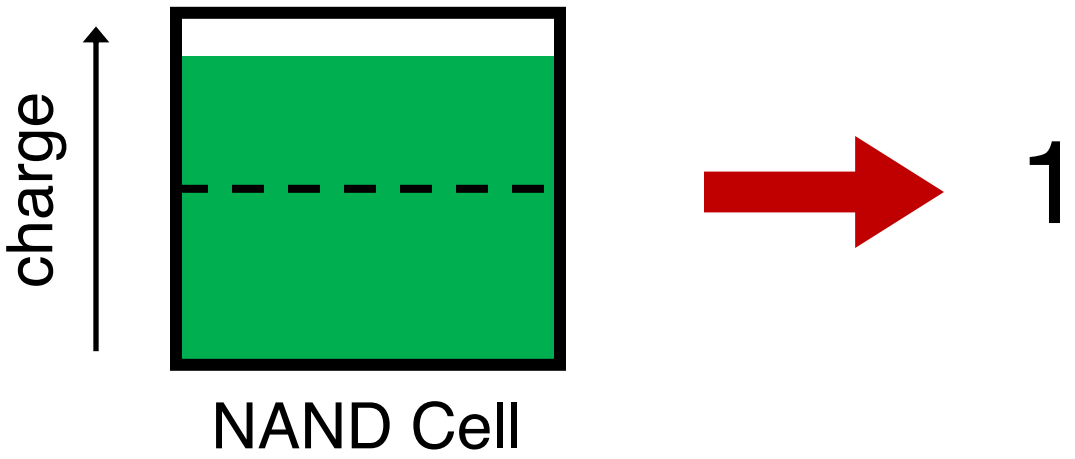
* https://www.enterprisestorageforum.com/storage-hardware/ssd-vs-hdd.html

Flash Architecture

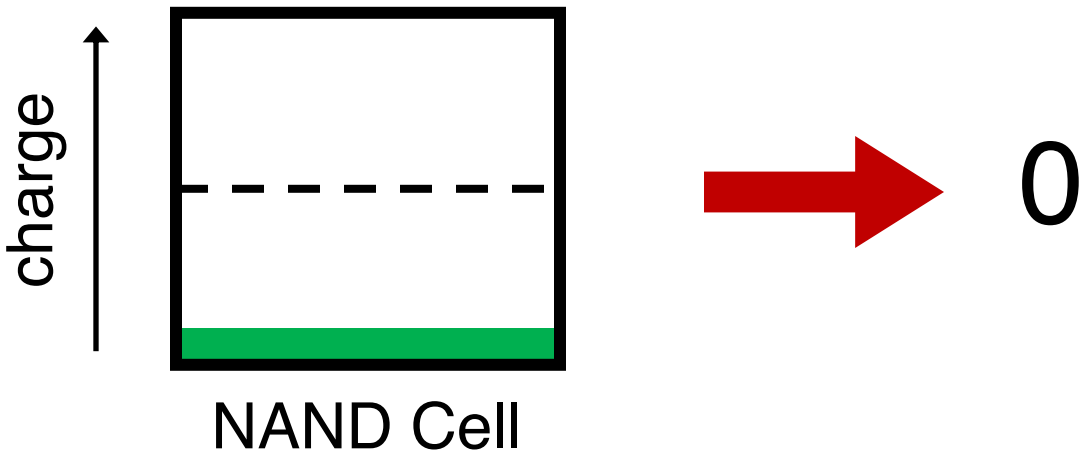
SLC: Single-Level Cell



SLC: Single-Level Cell



SLC: Single-Level Cell

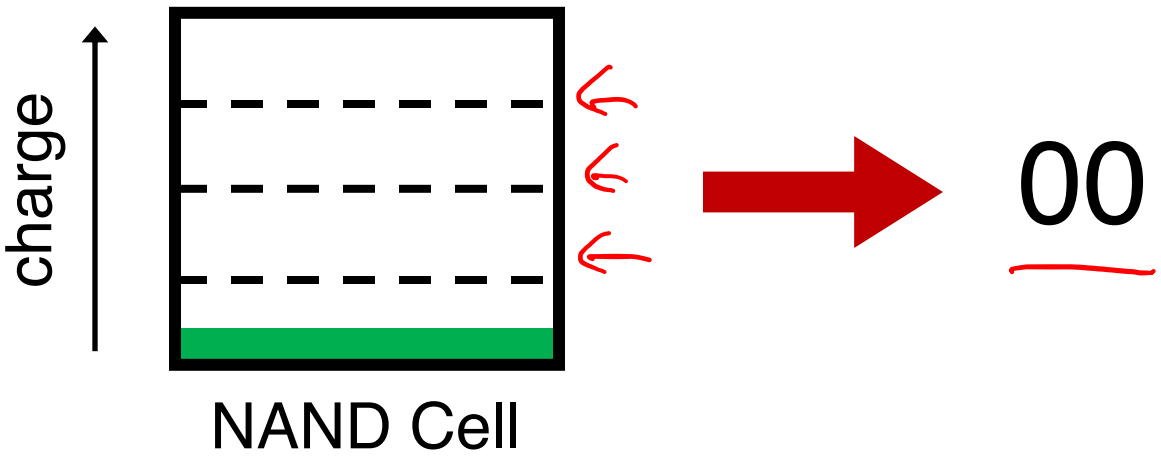


MLC: Multi-Level Cell

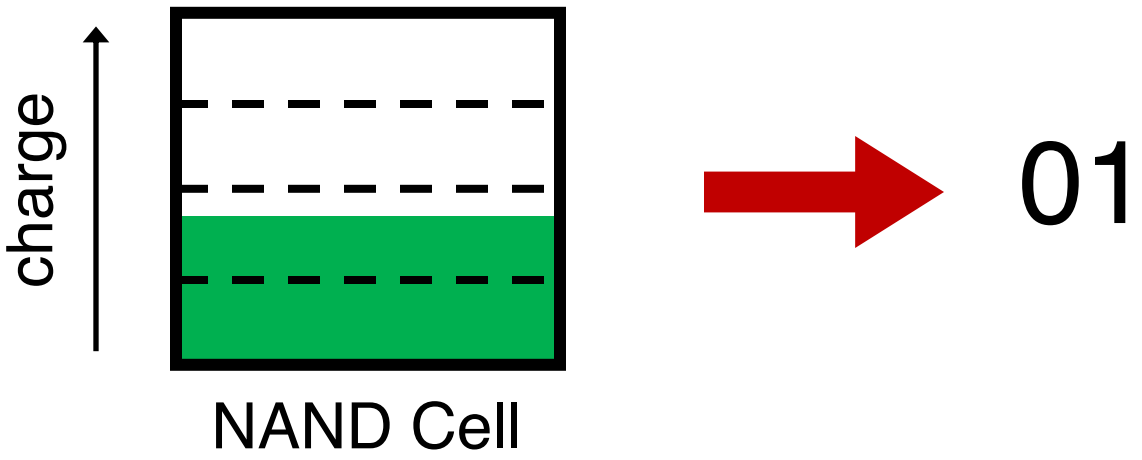
→ two bits

4 states.

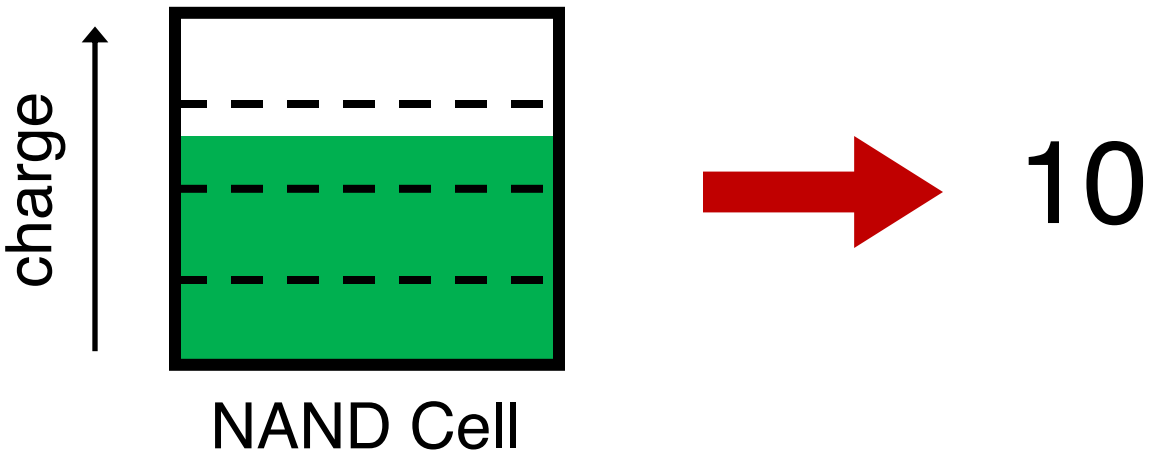
00
01
10
11



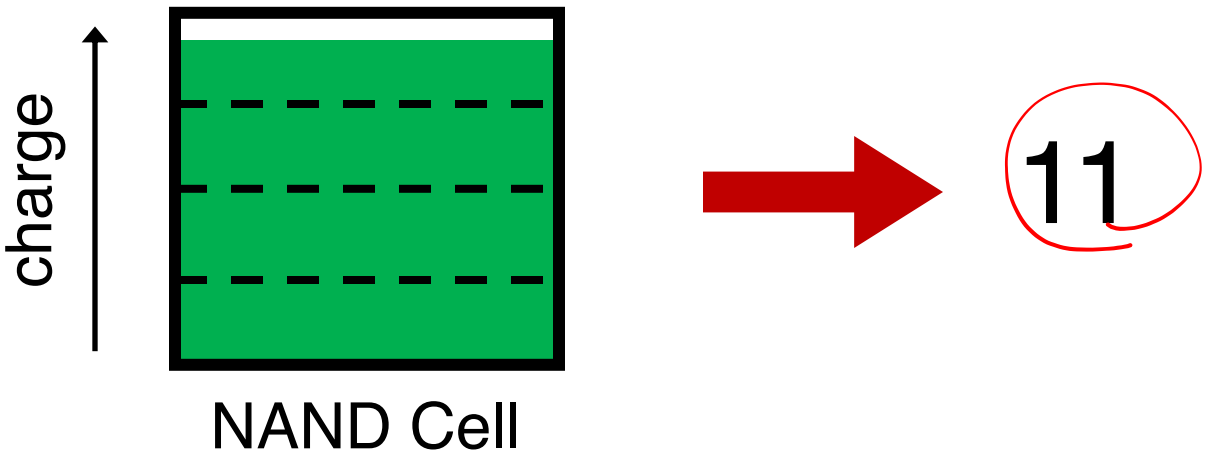
MLC: Multi-Level Cell



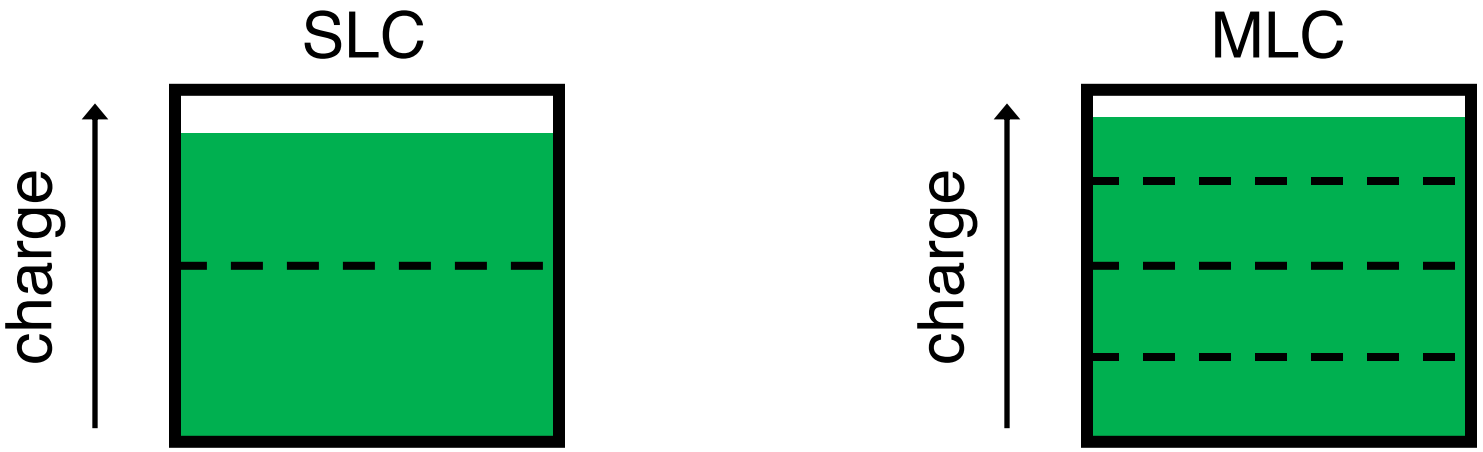
MLC: Multi-Level Cell



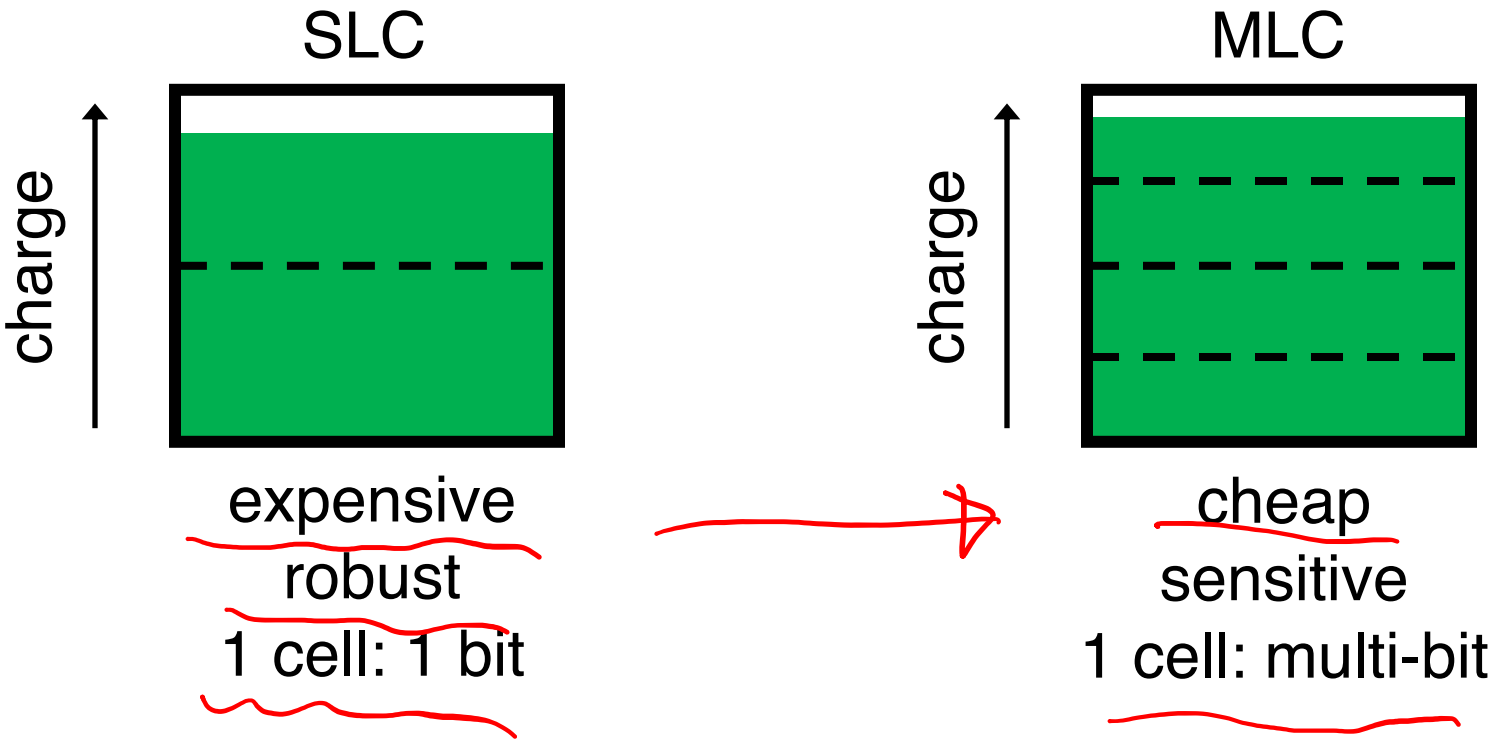
MLC: Multi-Level Cell



Single- vs. Multi-Level Cell



Single- vs. Multi-Level Cell



Wearout

- Problem: flash cells wear out after being erased too many times
- MLC: ~10K times
- SLC: ~100K times
- Usage strategy: ???

Wearout

- Problem: flash cells wear out after being erased too many times

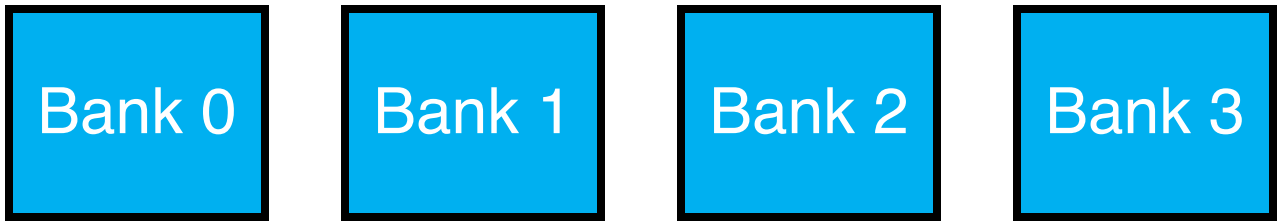
- MLC: ~10K times
- SLC: ~100K times

- Usage strategy: wear leveling
  - Prevents some cells from being wornout while others still fresh

write balancing

Banks

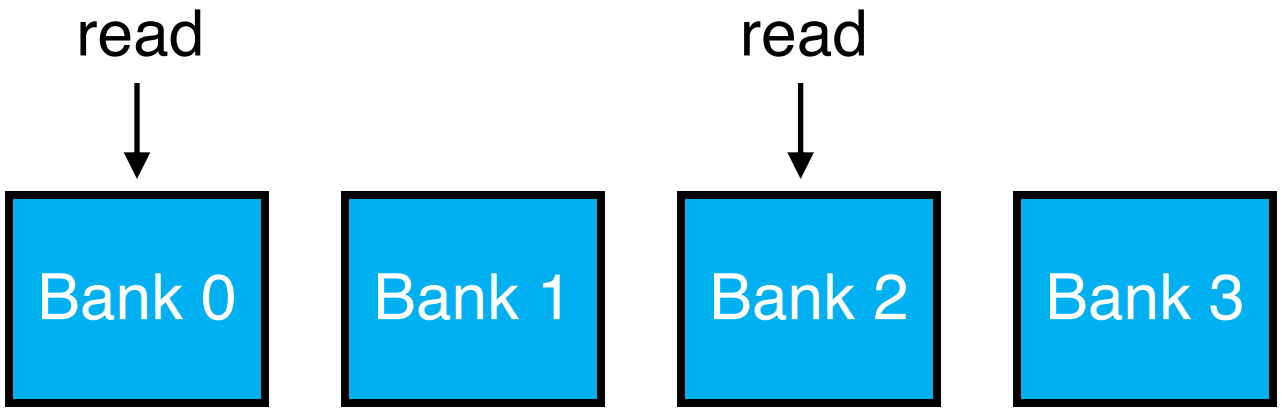
- Flash devices are divided into banks (aka. planes)
- Banks can be accessed in parallel



Banks

- Flash devices are divided into banks (aka. planes)

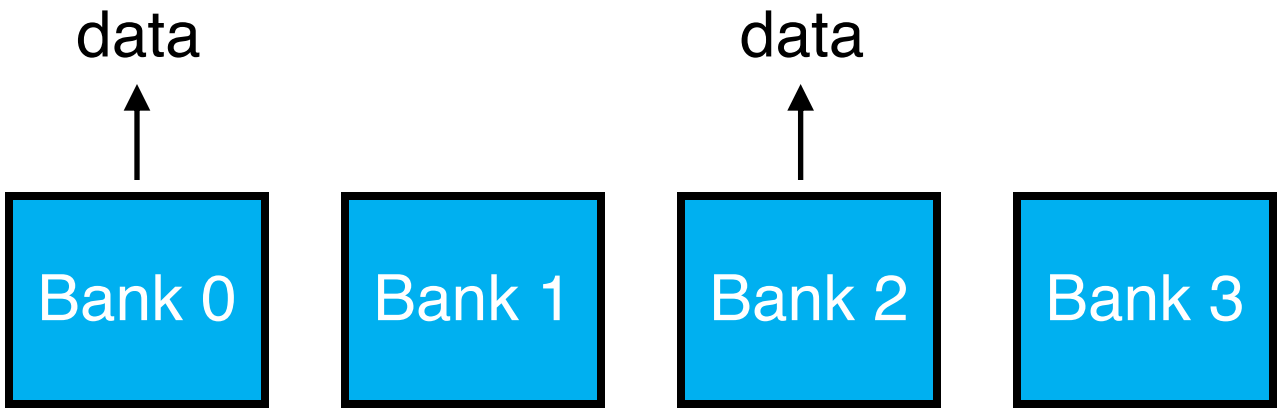
- Banks can be accessed in parallel



Banks

- Flash devices are divided into banks (aka. planes)

- Banks can be accessed in parallel



Flash Writes

- Writing 0's
  - Fast, fine-grained
- Writing 1's
  - Slow, coarse-grained

Flash Writes

- Writing 0's
  - Fast, fine-grained
  - called "program"
- Writing 1's
  - Slow, coarse-grained
  - called "erase"

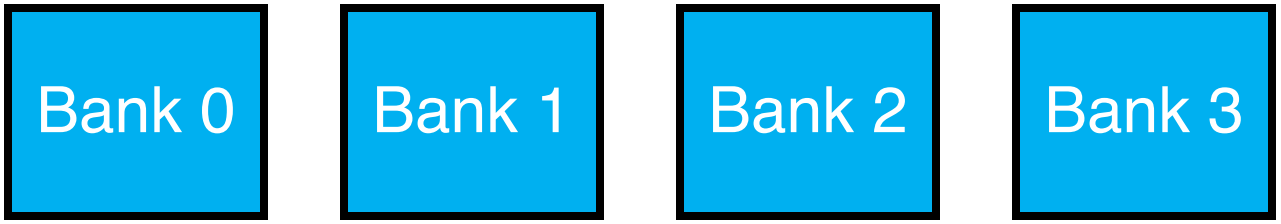
Flash Writes

- Writing 0's
  - Fast, fine-grained [page-level]
  - called "program"
- Writing 1's
  - Slow, coarse-grained [block-level]
  - called "erase"

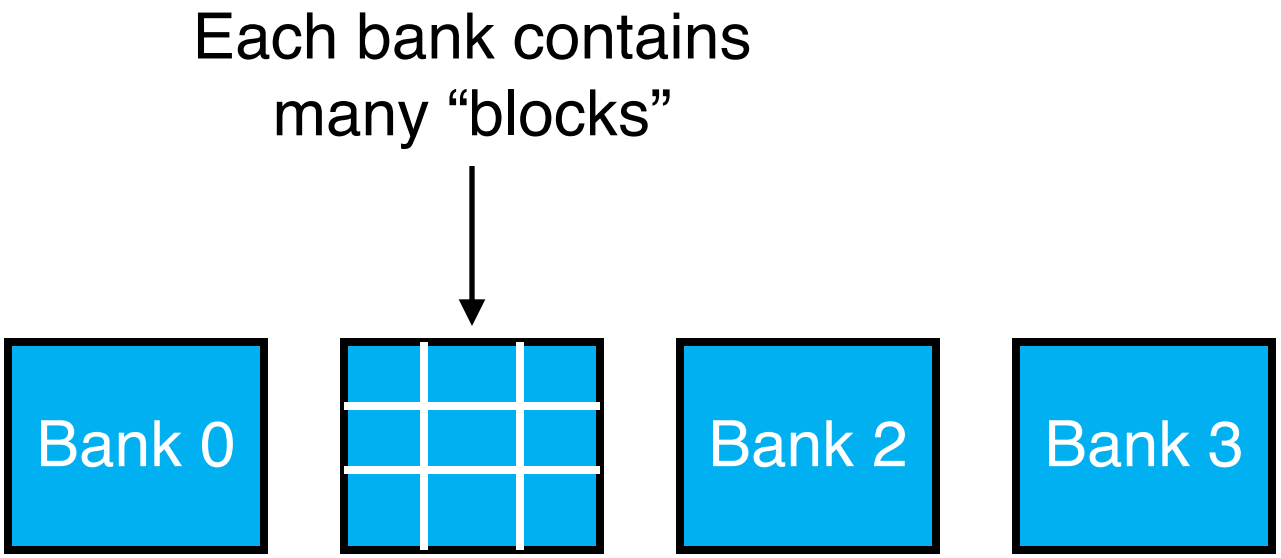
Flash Writes

- Writing 0's
  - Fast, fine-grained [page-level]
  - called "program"
- Writing 1's
  - Slow, coarse-grained [block-level]
  - called "erase"
- Flash can only "write" (program) into clean pages
  - "clean": pages containing all 1's (pages that have been erased)
  - Flash does not support in-place overwrite!

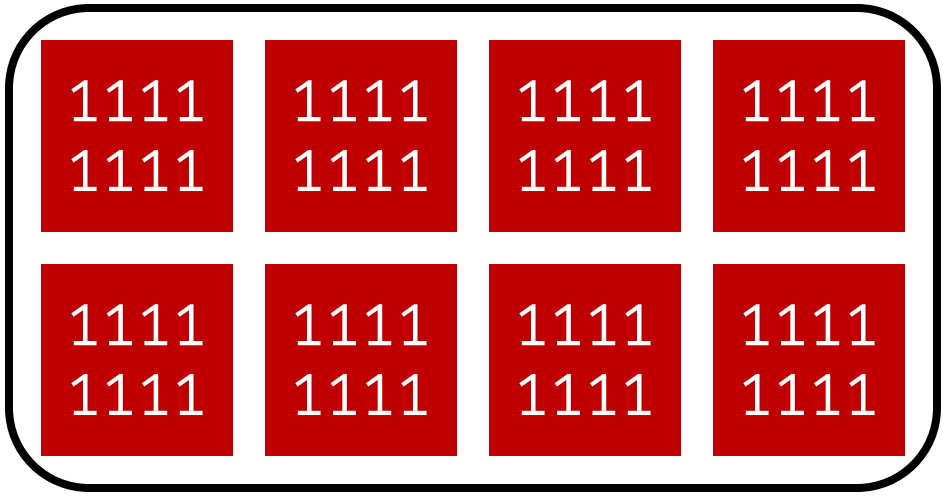
Banks and Blocks



Banks and Blocks



Block and Pages

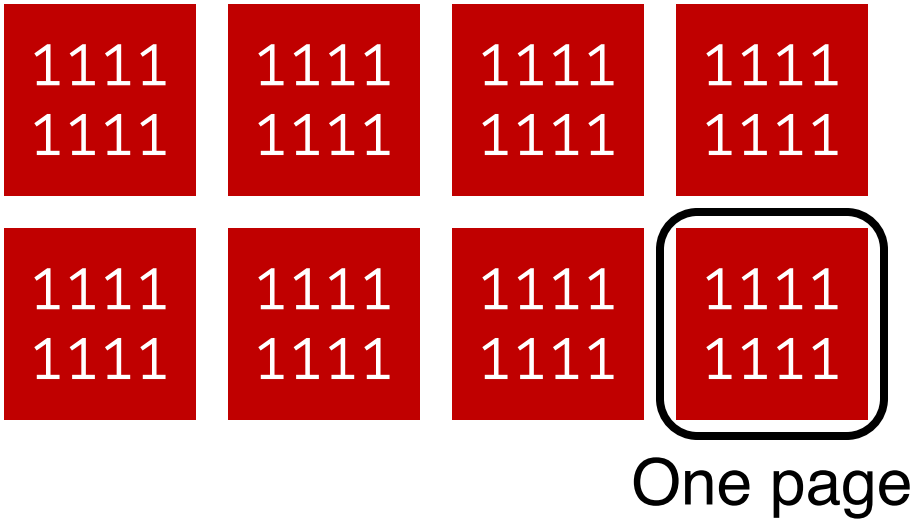


One block

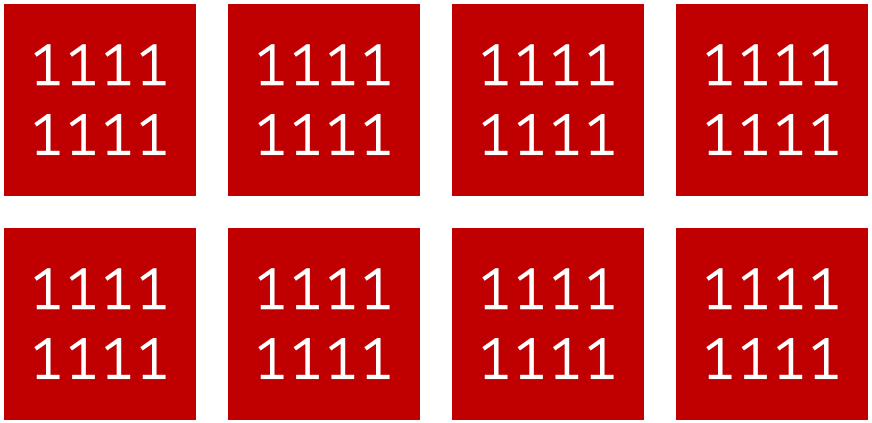
Block and Pages

8 pages.

basic unit of I/O.

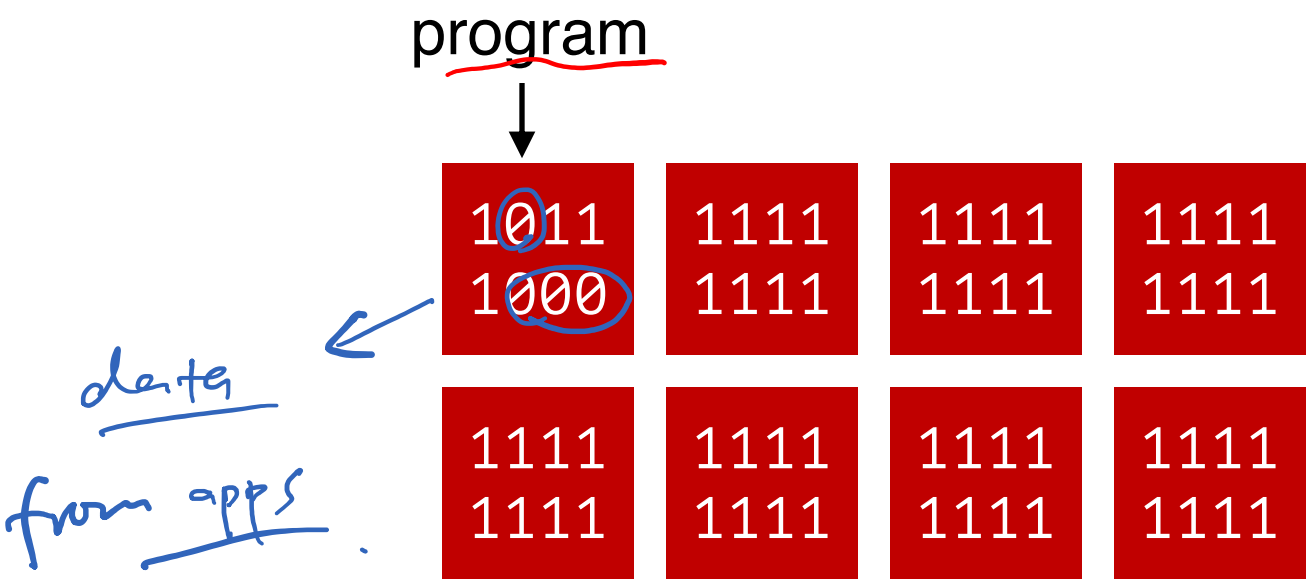


Block and Pages

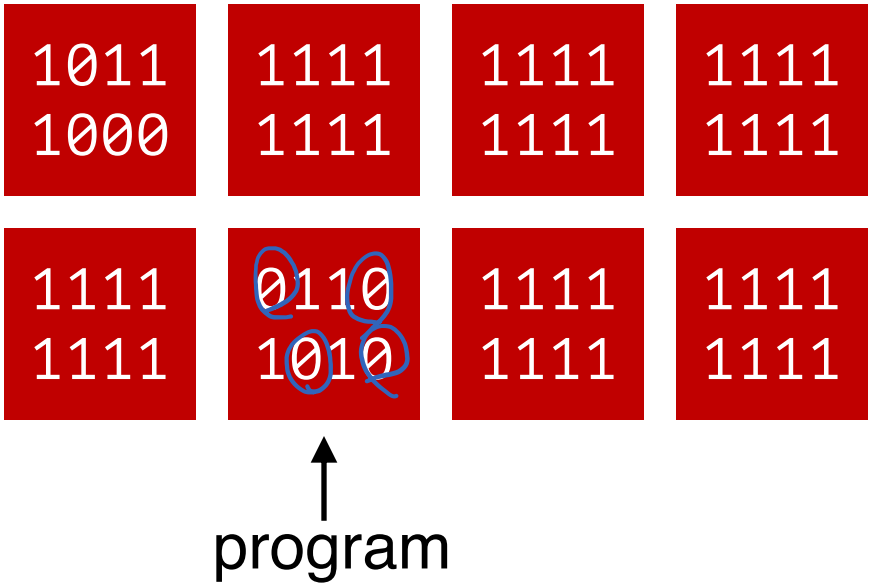


All pages are clean
("programmable")

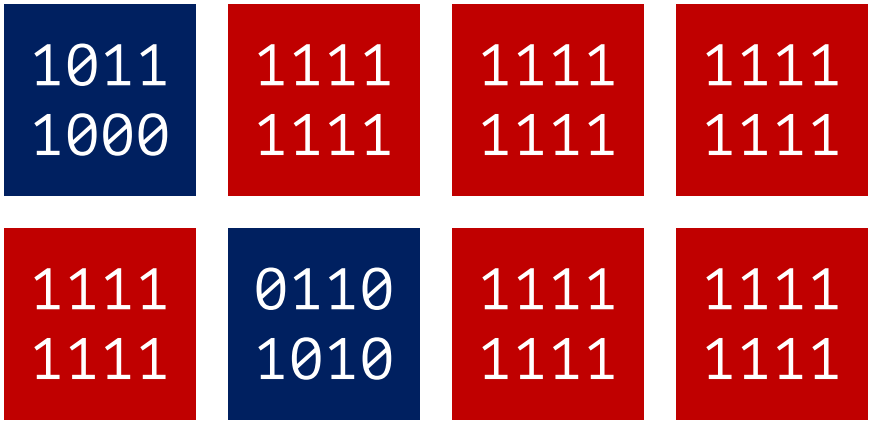
Block


program
1011
1000
1111
1111
1111
1111
1111
1111
1111
1111
1111
1111
1111
1111
data
from apps

Block


1011
1000
1111
1111
1111
1111
1111
1111
1111
1111
0110
1010
1111
1111
1111
1111
program

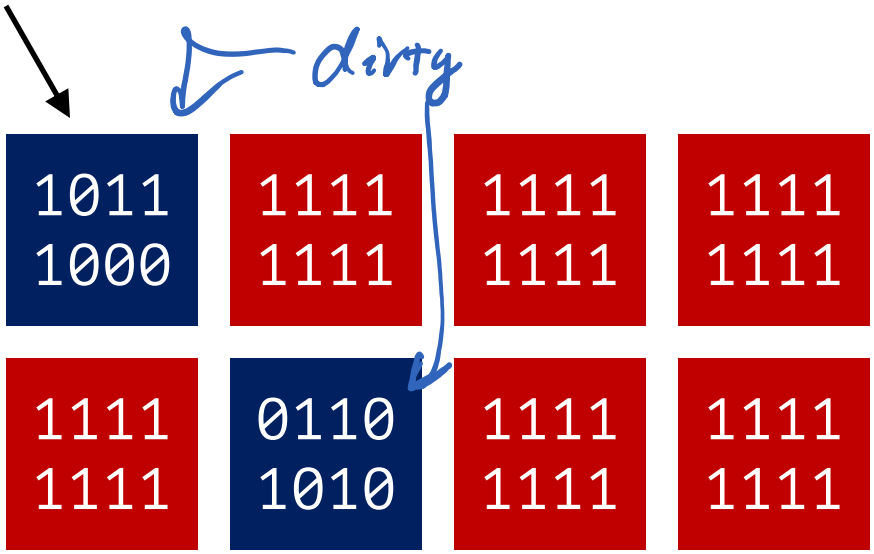
Block



Two pages hold data
(**cannot be overwritten**)

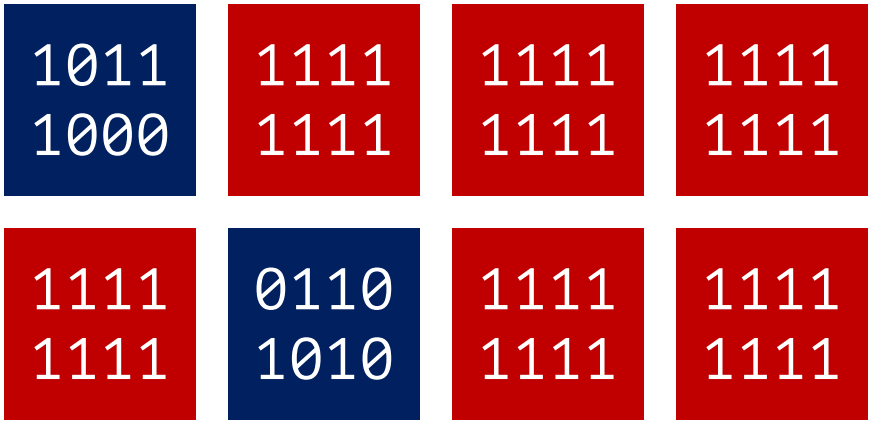
Block

still want to write data into this page???



Two pages hold data
(**cannot be overwritten**)

Block

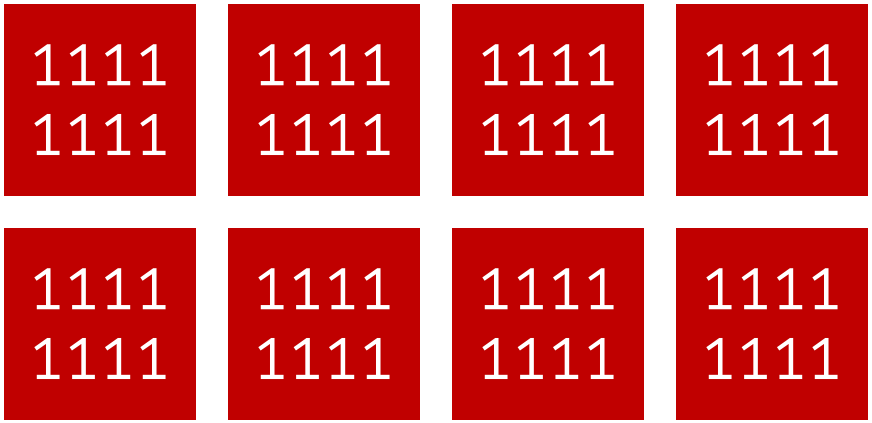


erase

the whole block!!

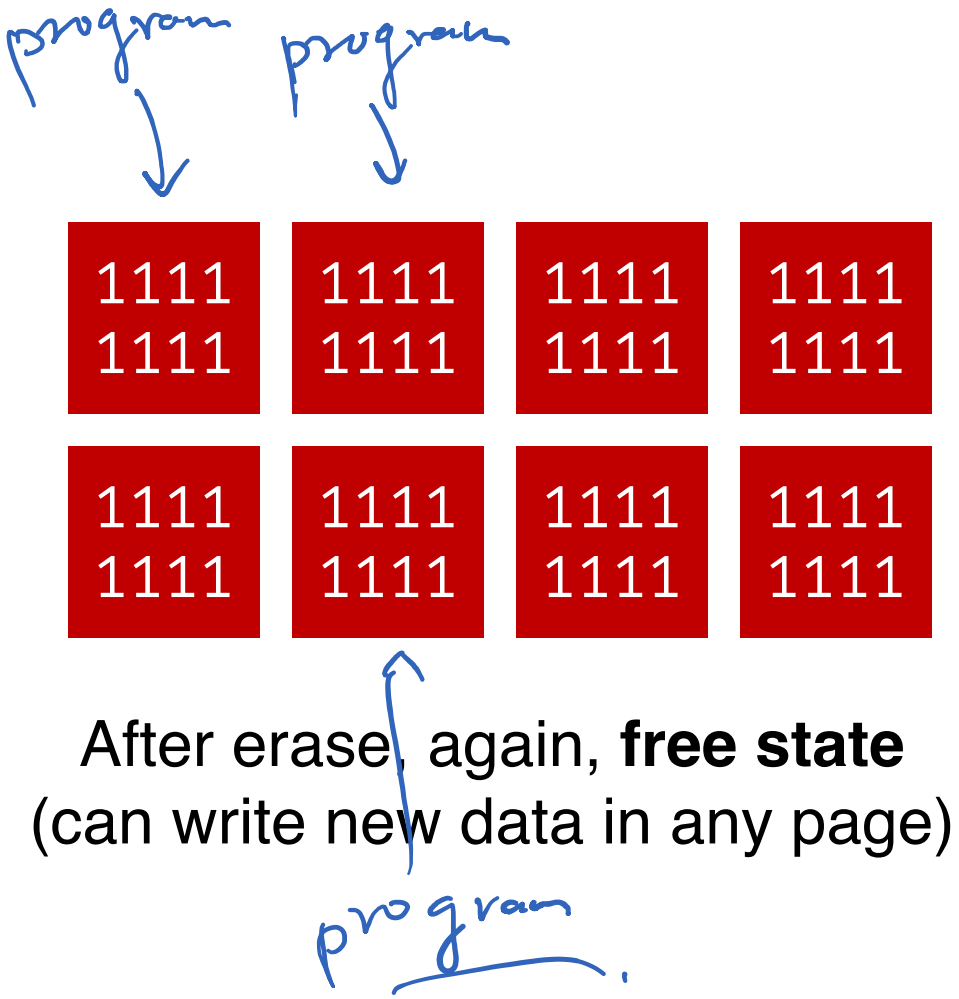
Block

Flip 0's into 1's.

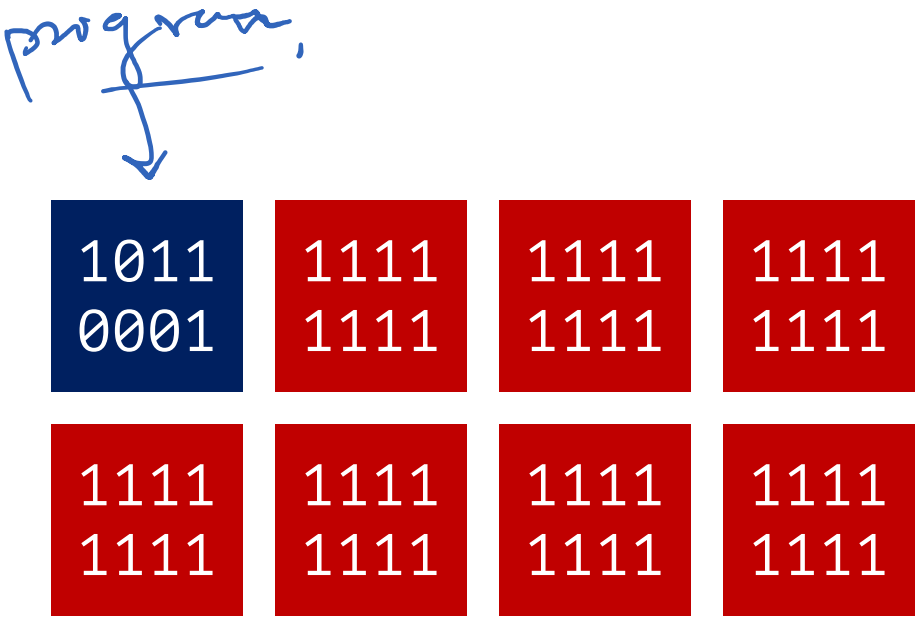


erase
(the whole block)

Block

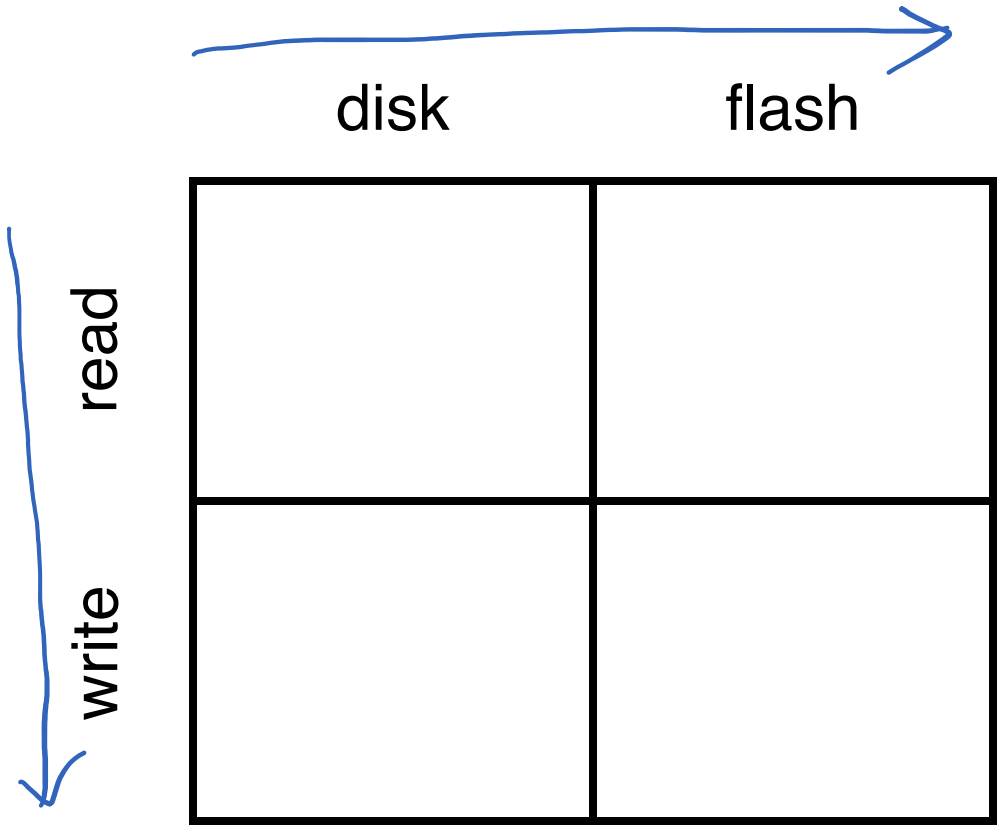

program
program
1111 1111
1111 1111
1111 1111
1111 1111
1111 1111
1111 1111
1111 1111
1111 1111
After erase, again, **free state**
(can write new data in any page)
program.

Block

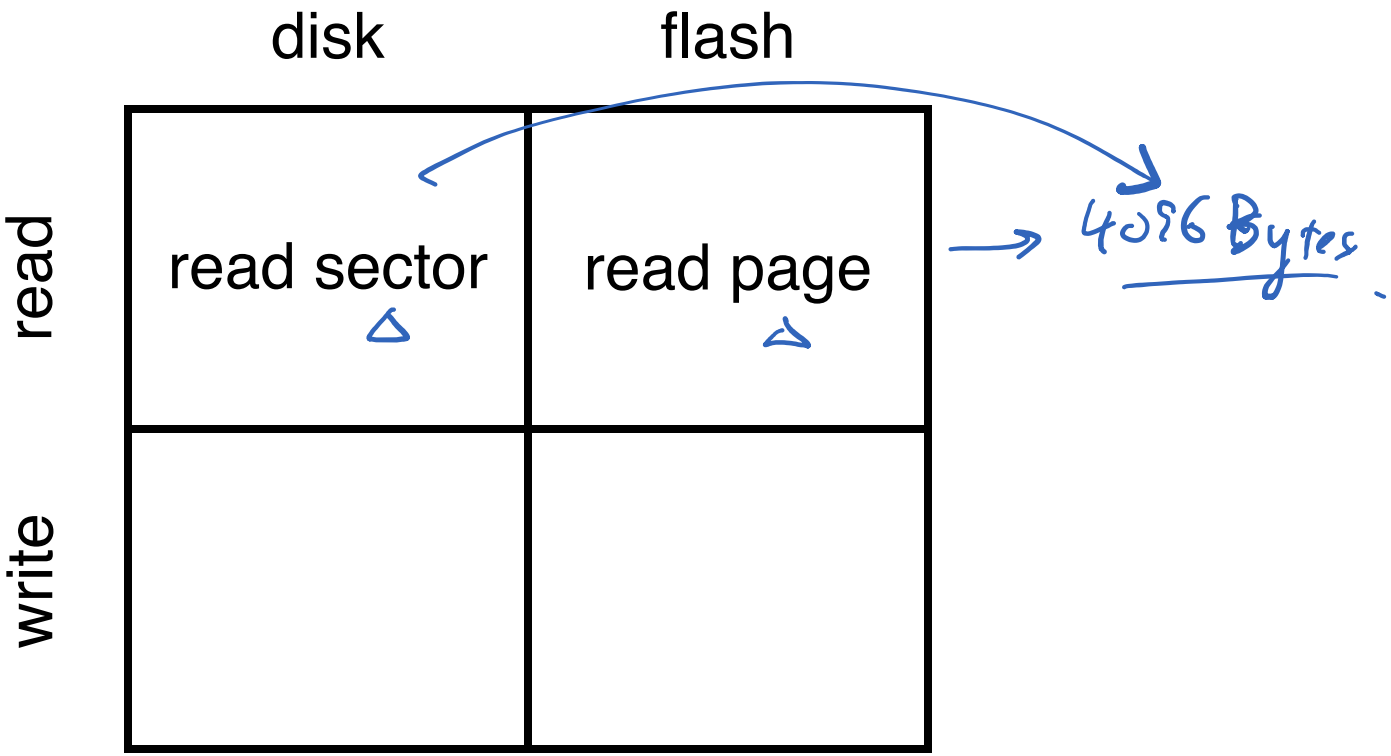


This blue page holds data

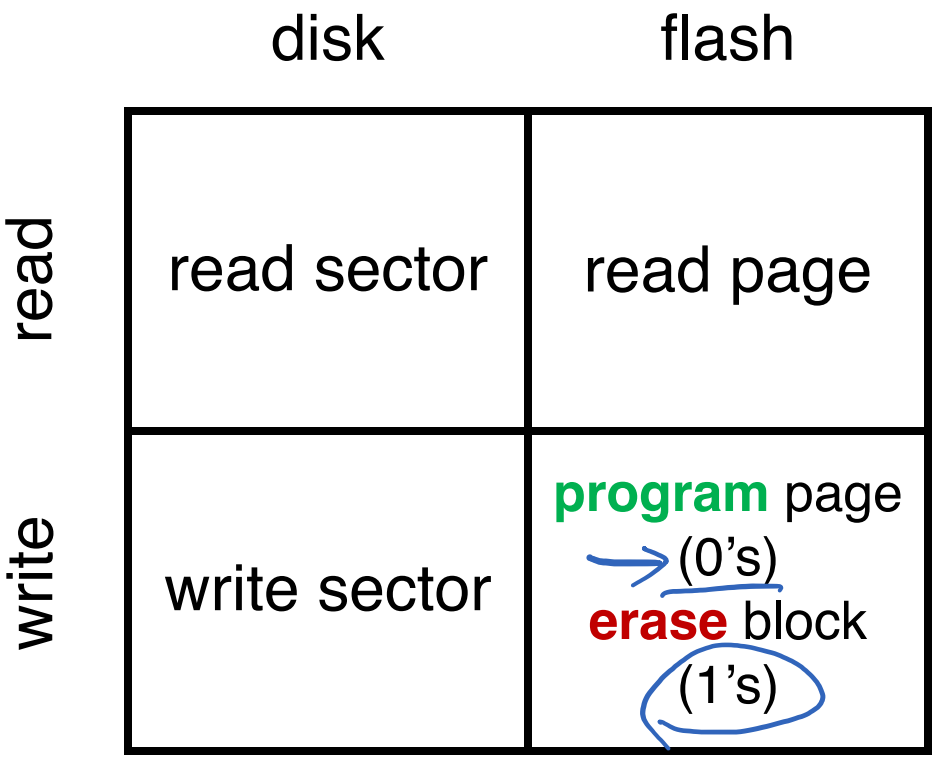
Flash vs. Disks: APIs



Flash vs. Disks: APIs



Flash vs. Disks: APIs



program ops.
1 → 0
erase op.
0 → 1

Flash Architecture

- Bank/plane: 1024 to 4096 blocks
  - Banks accessed in parallel
- Block: 64 to 256 pages
  - Unit of erase
- Page: 2 to 8 KB
  - Unit of read and program

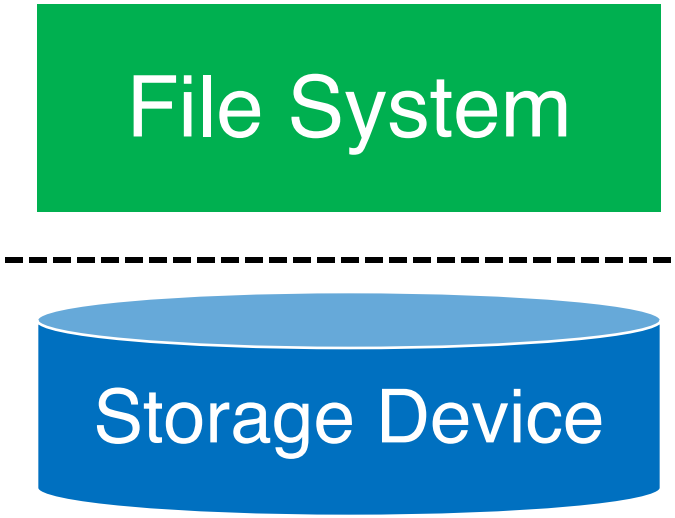
Disks vs. Flash: Performance

- Throughput
  - Disk: ~130MB/s (sequential)
  - Flash: ~400MB/s

- Latency
  - Disk: ~10ms (one op)
  - Flash:
    - Read: 10-50us
    - Program: 200-500us
    - Erase: 2ms

Working with File System

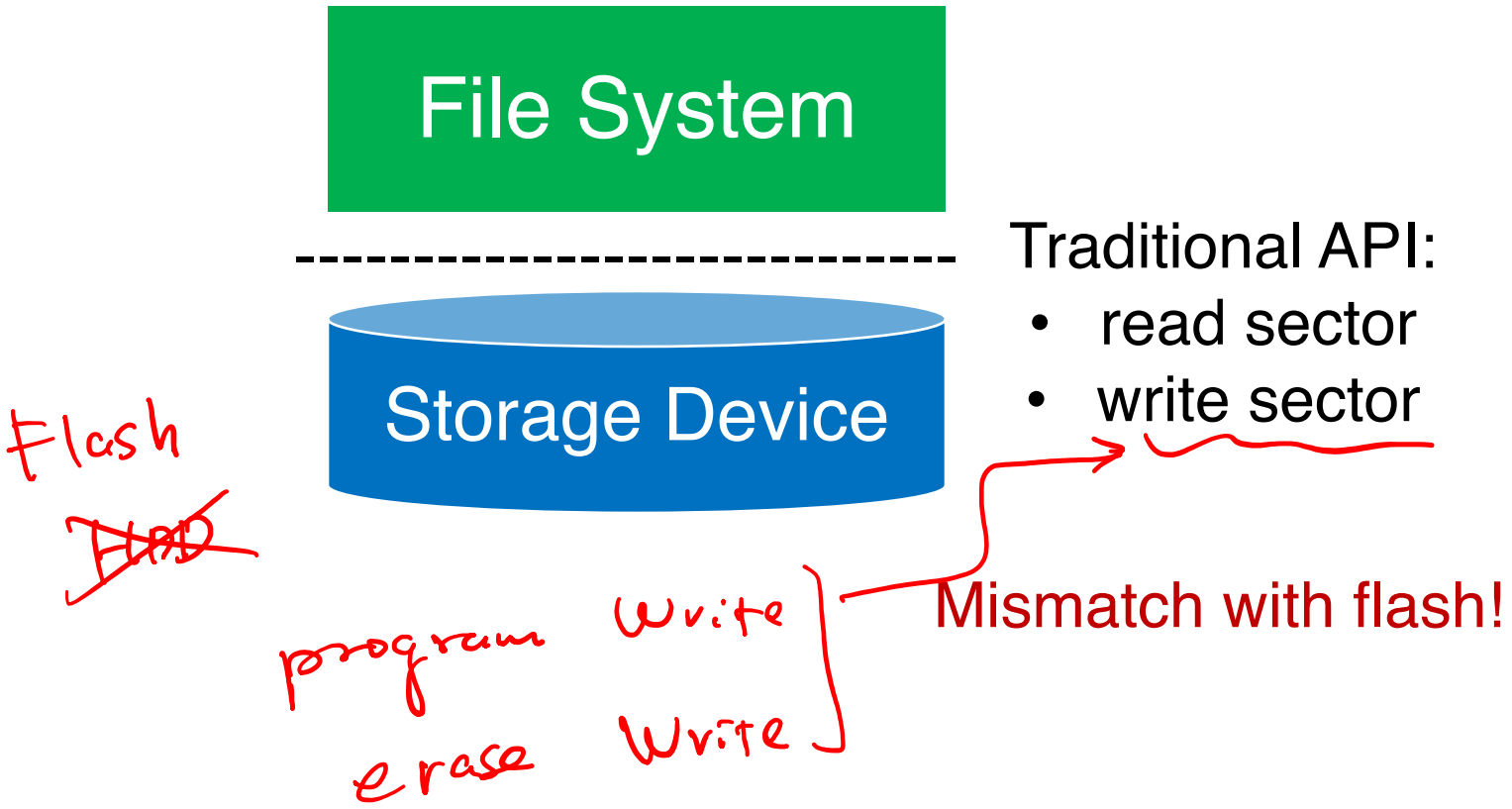
Traditional File Systems



Traditional API:

- read sector
- write sector

Traditional File Systems

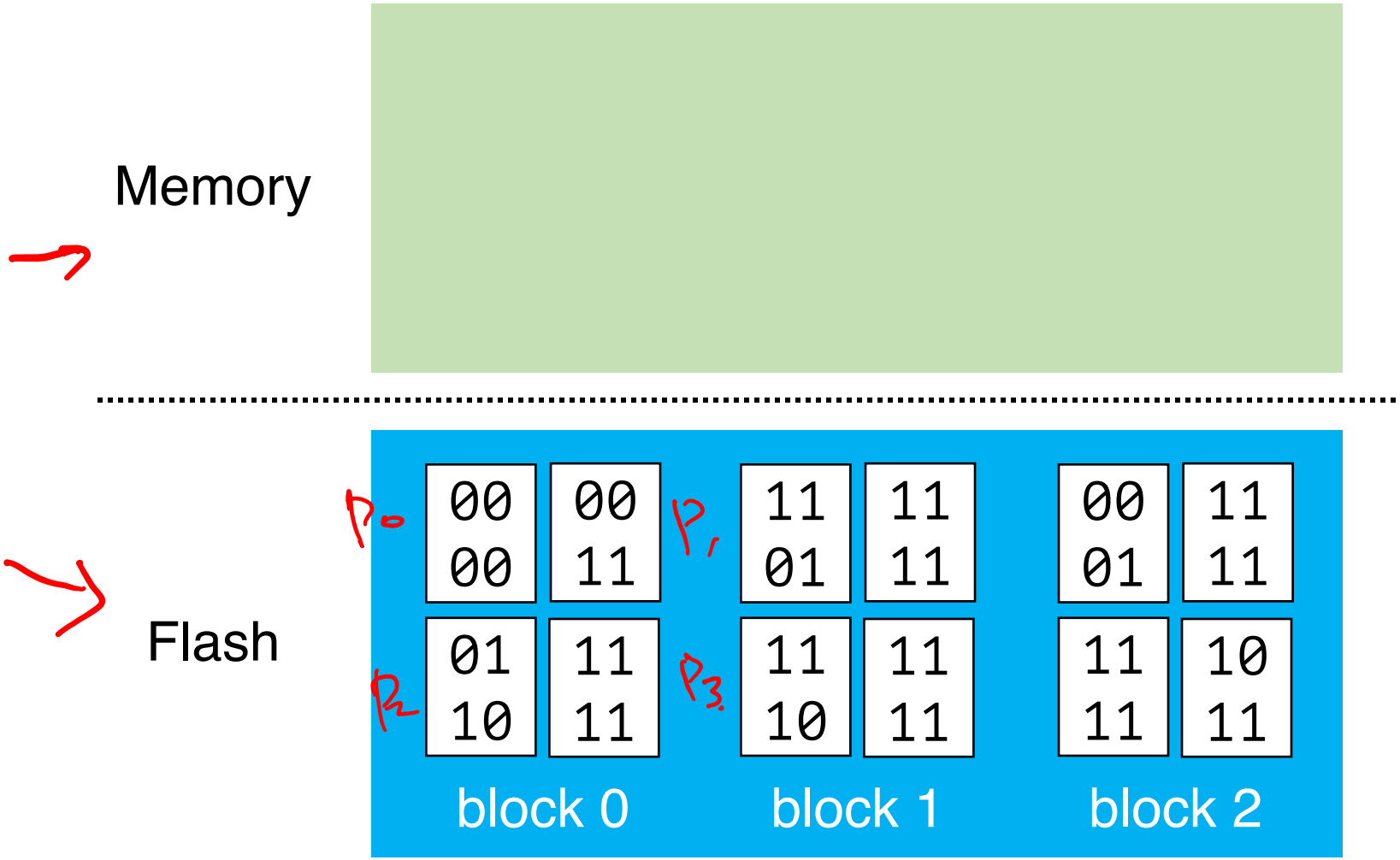


Traditional APIs wrapping around Flash APIs

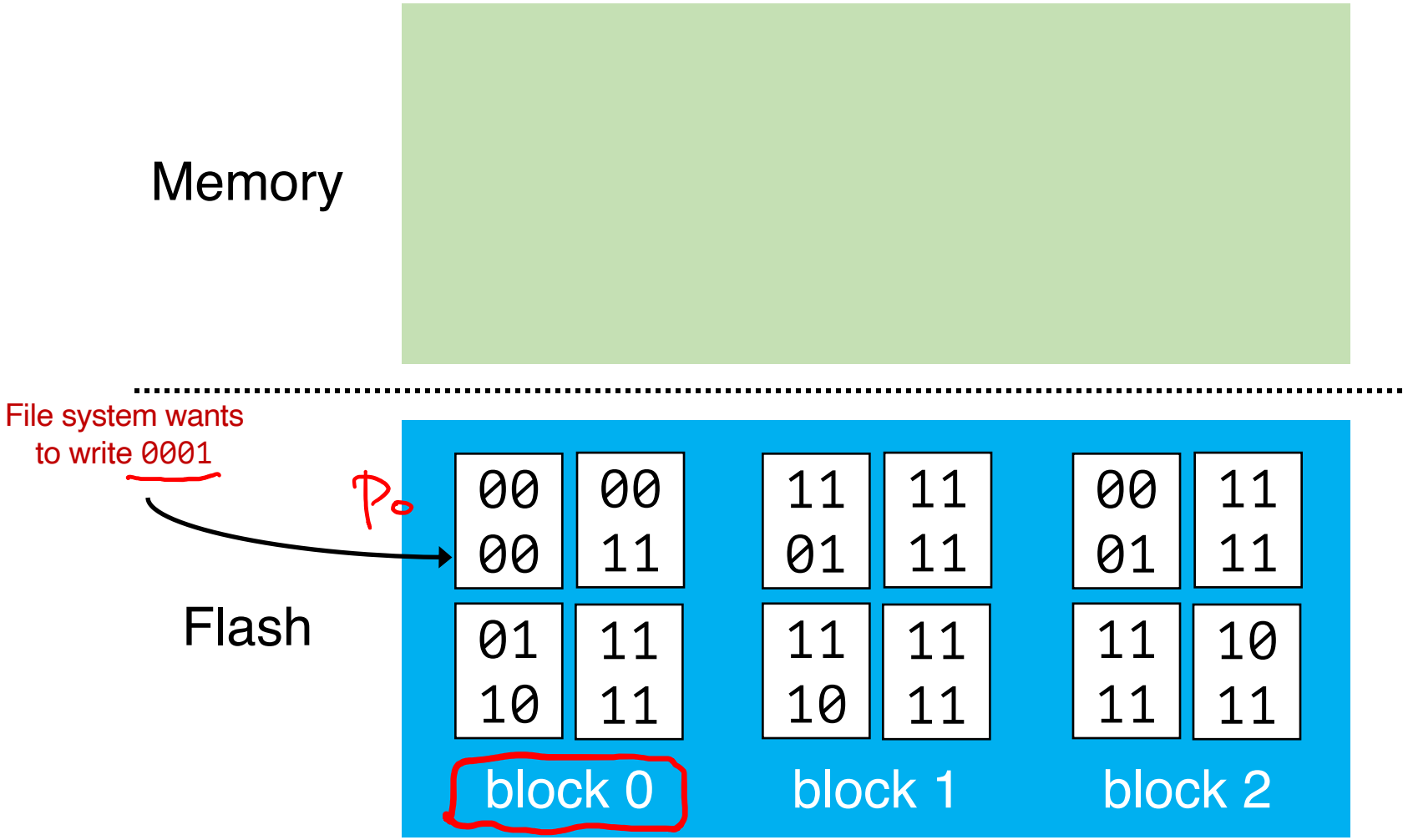
```
read(addr):
    return flash_read(addr)

write(addr, data):
    block_copy = flash_read(all pages of block)
    modify block_copy with data
    flash_erase(block of addr)
    flash_program(all pages of block_copy)
```

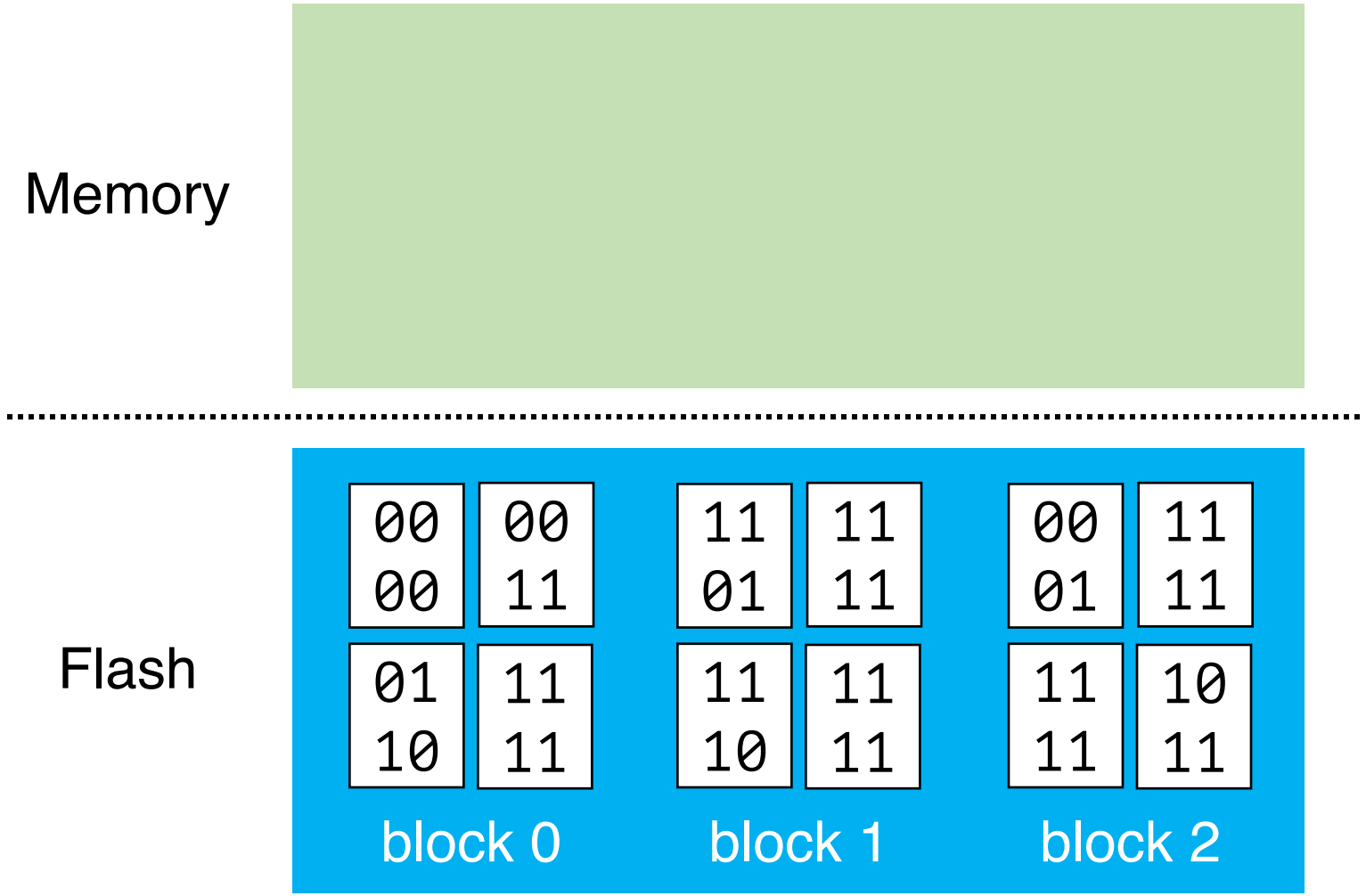
Awkward Flash Write



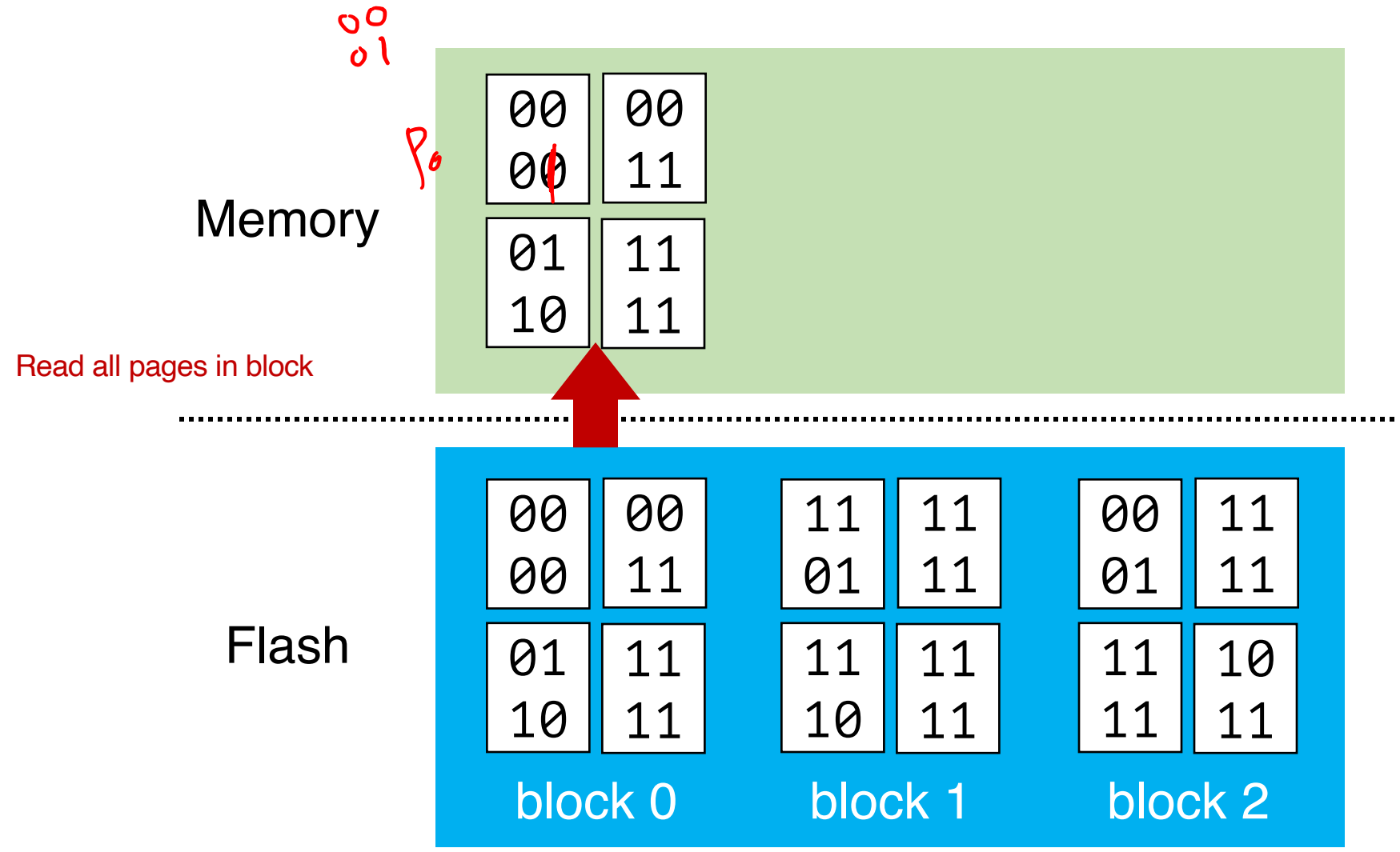
Awkward Flash Write



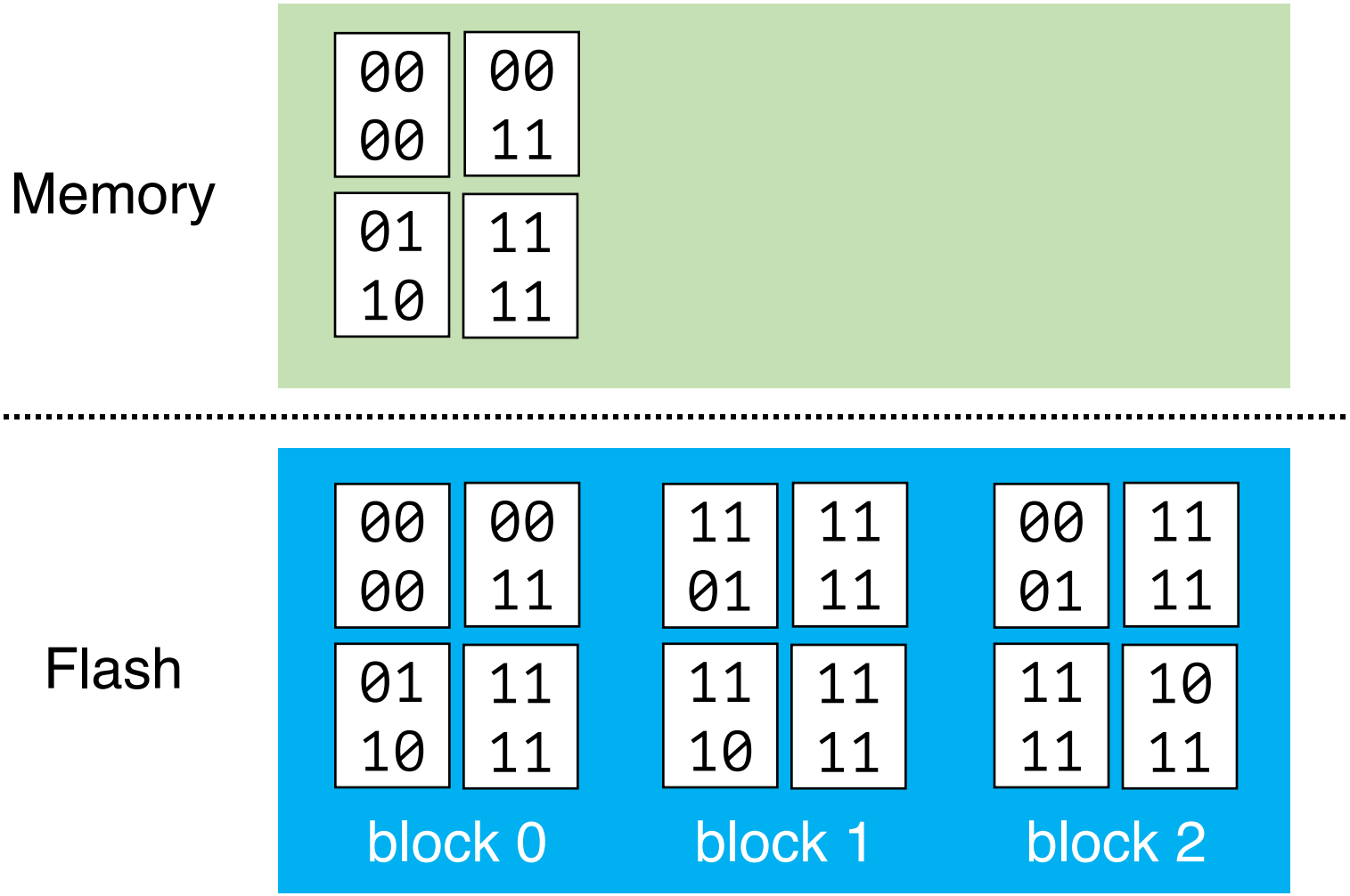
Awkward Flash Write



Awkward Flash Write

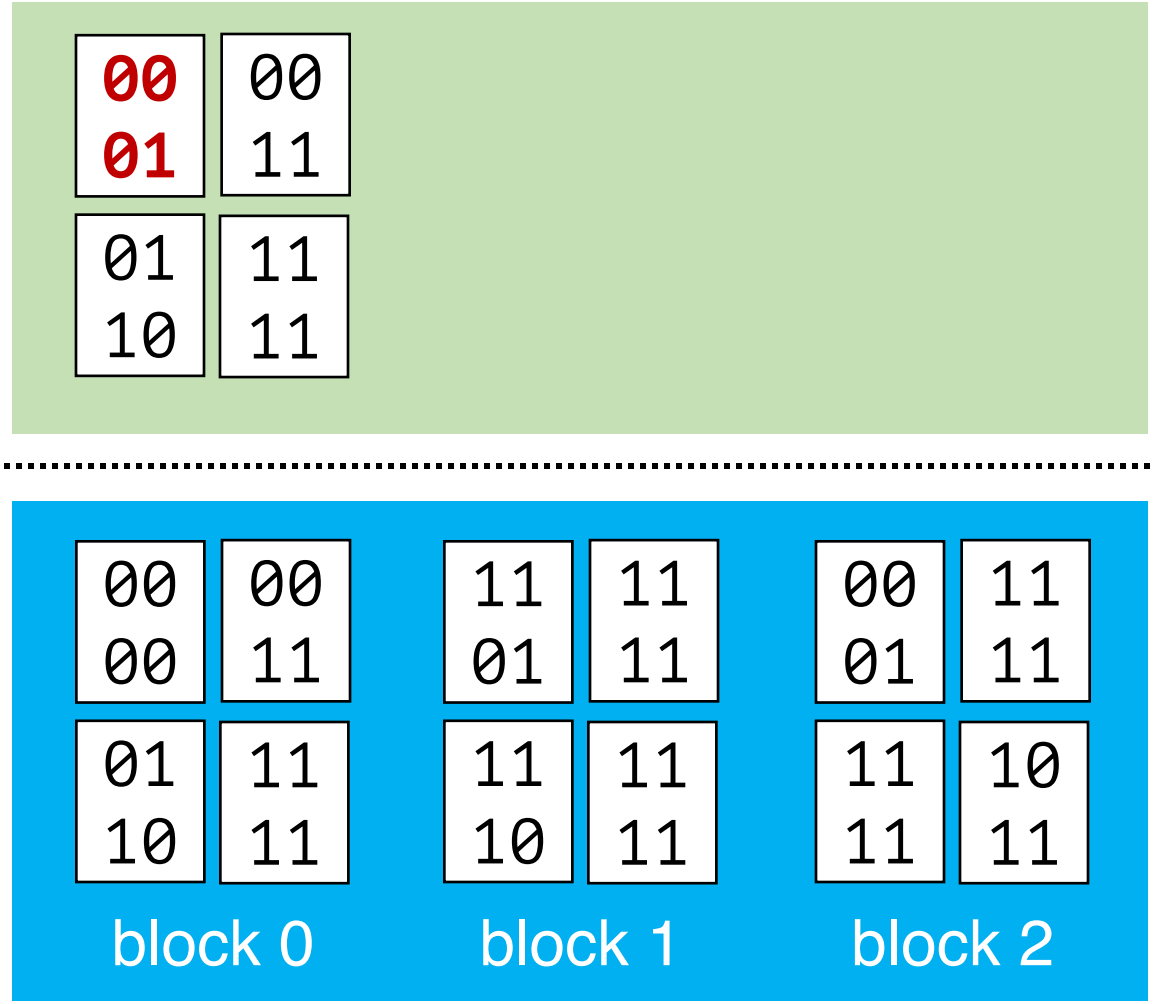


Awkward Flash Write

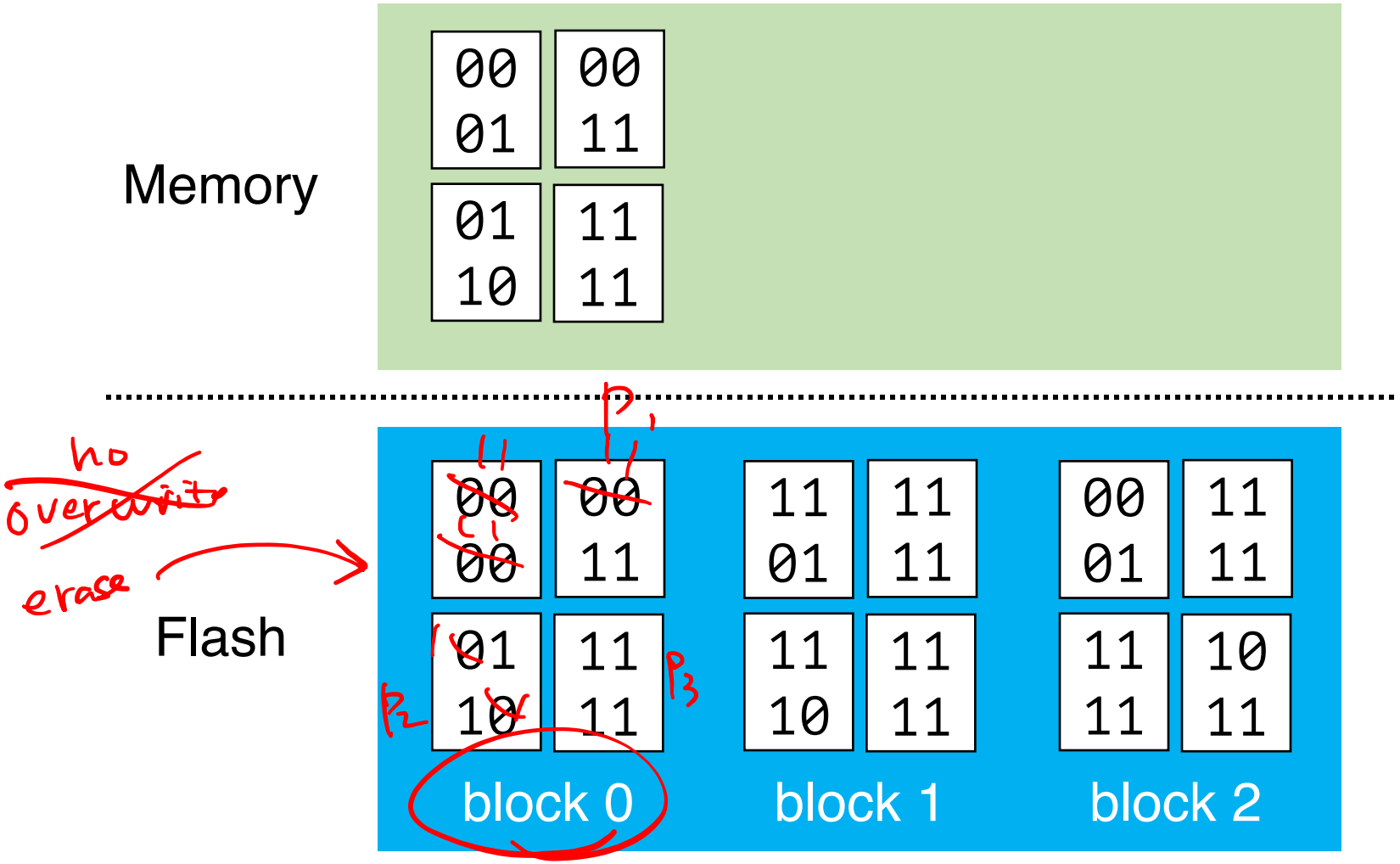


Awkward Flash Write

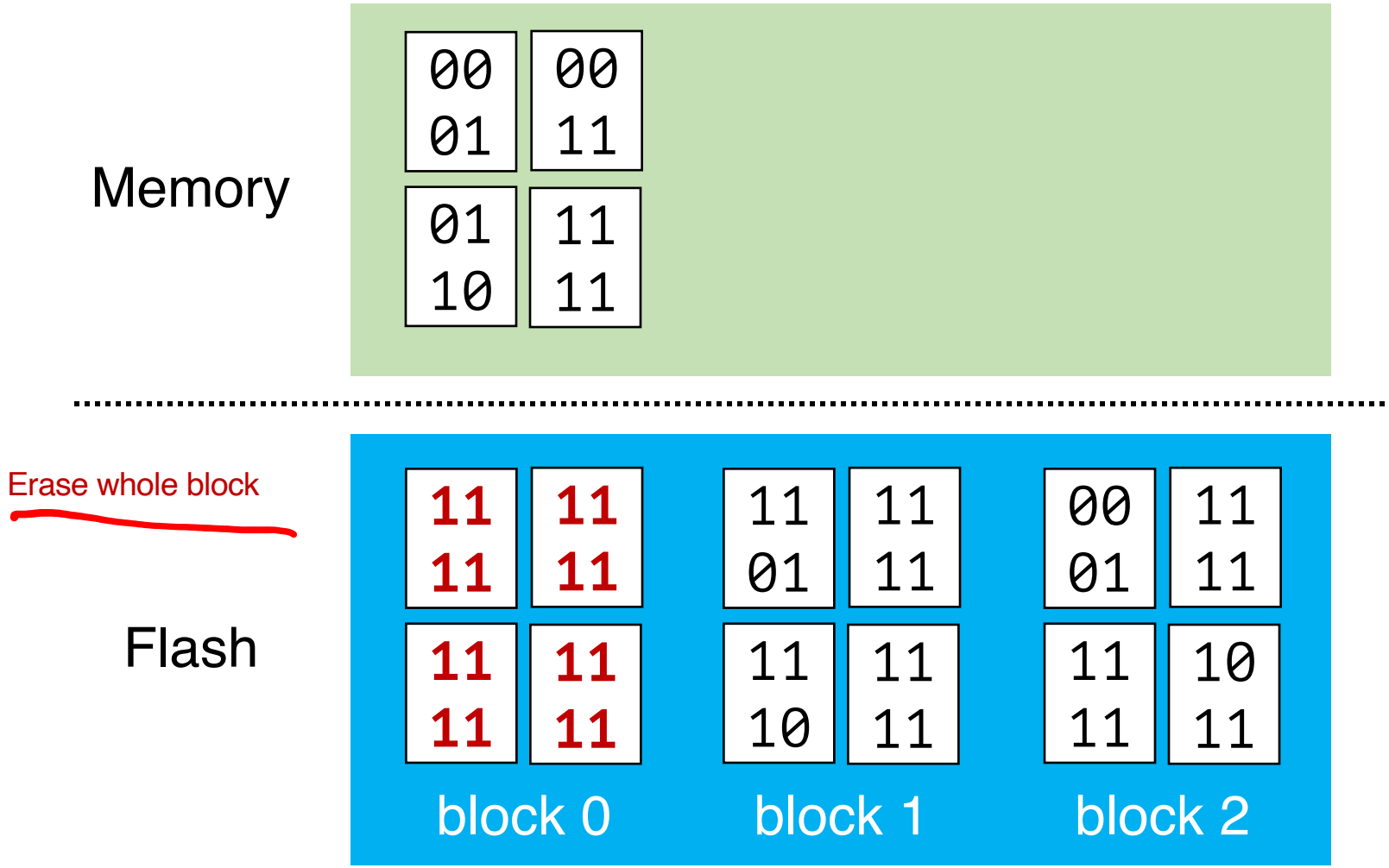
Modify target page in memory



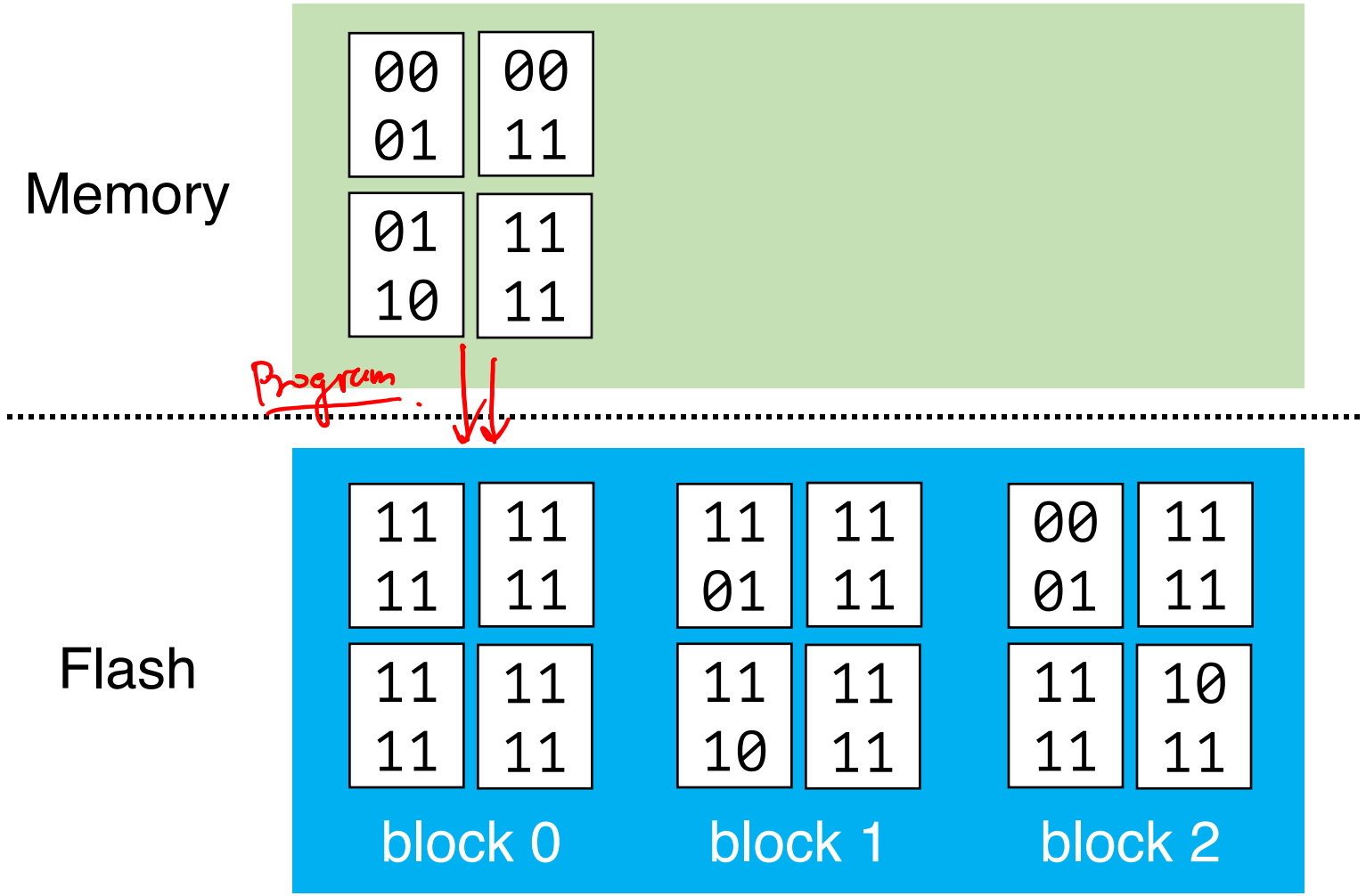
Awkward Flash Write



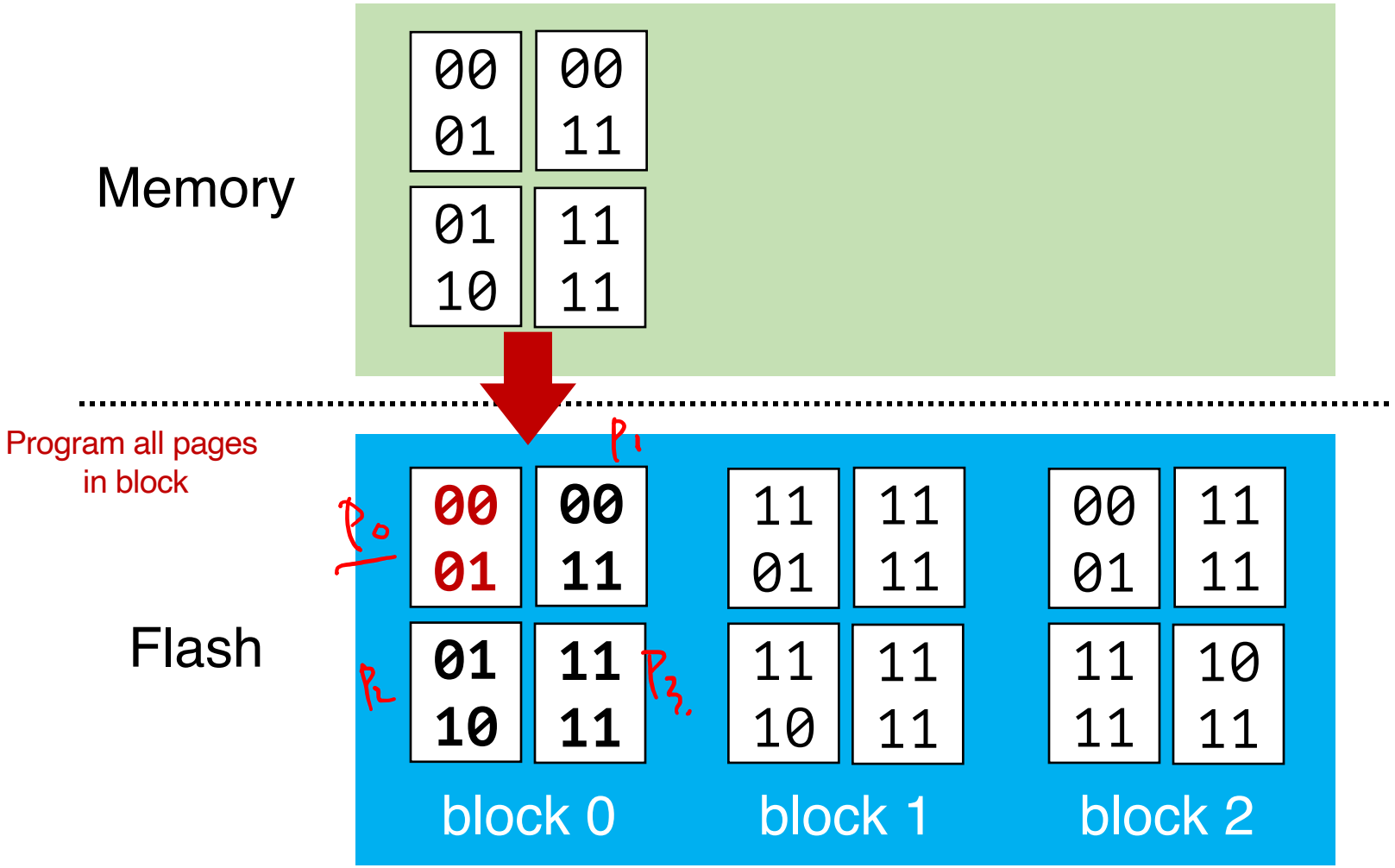
Awkward Flash Write



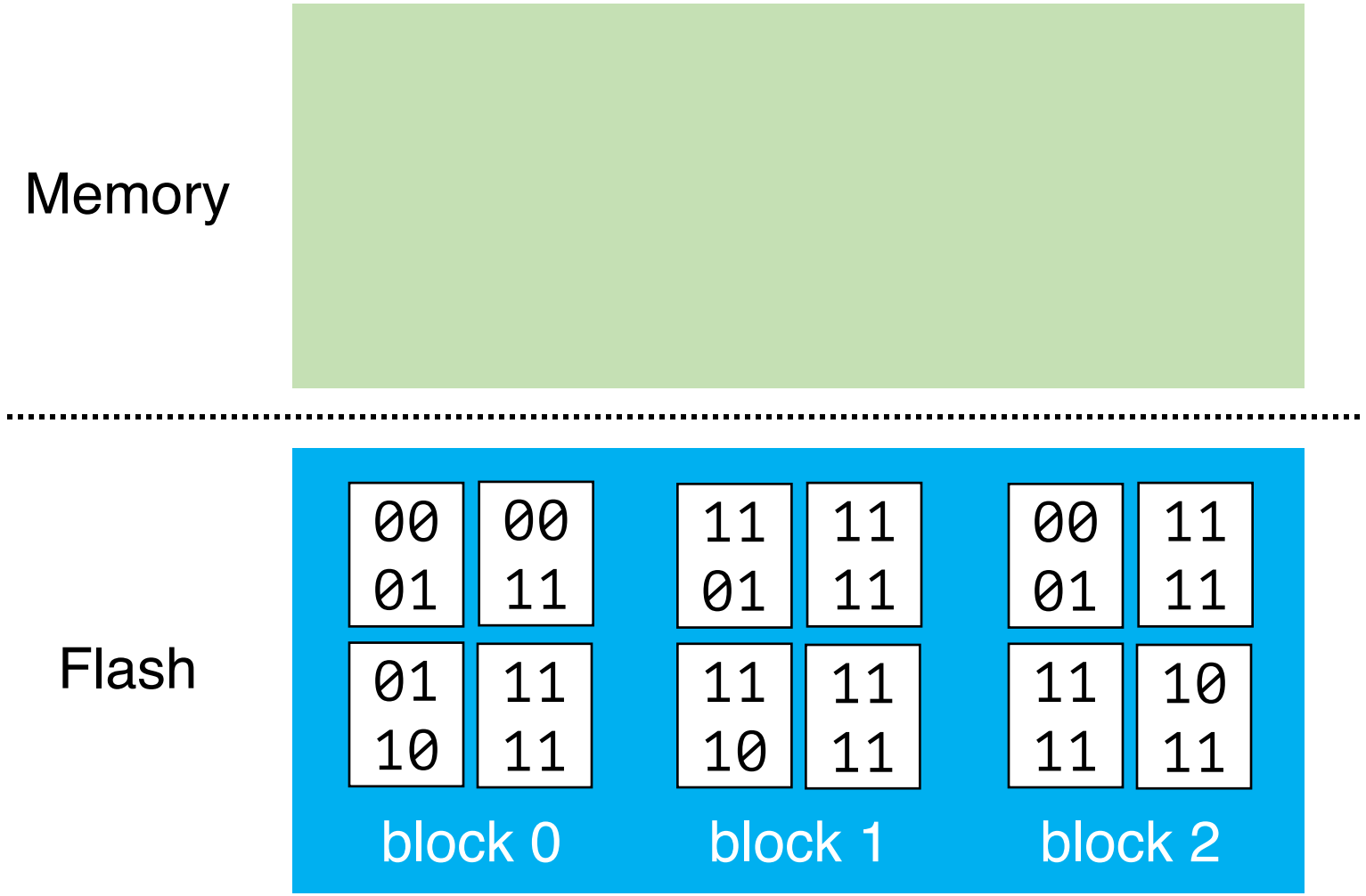
Awkward Flash Write



Awkward Flash Write



Awkward Flash Write



Issue: Write Amplification

- Random writes are expensive for flash!
- Writing one 4KB page may cause:
  - `read`, `erase`, and `program` of the whole 256KB block

Random I/Os → Sequential Appends.

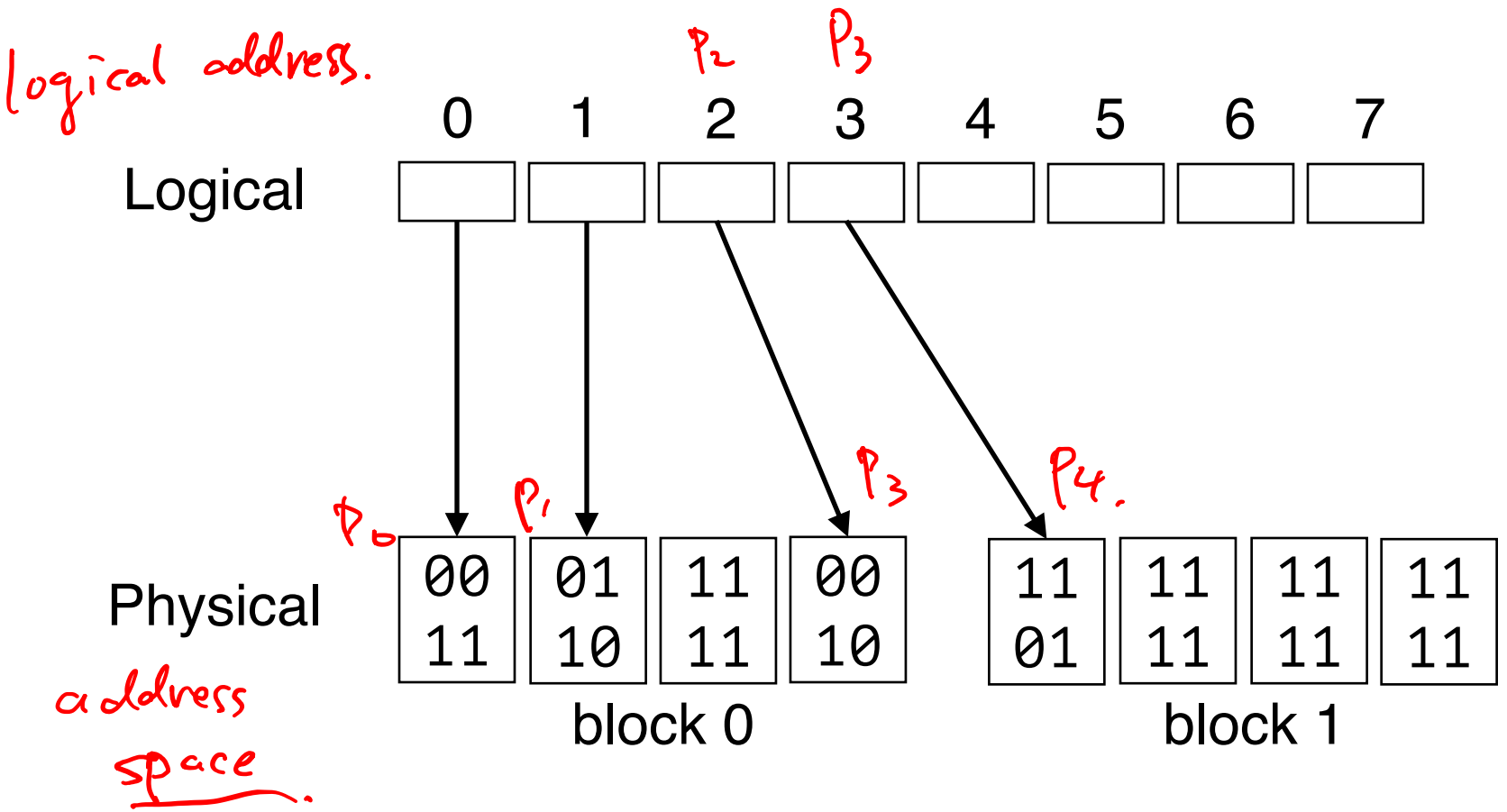
By adding one more abstraction layer!

Flash Translation Layer

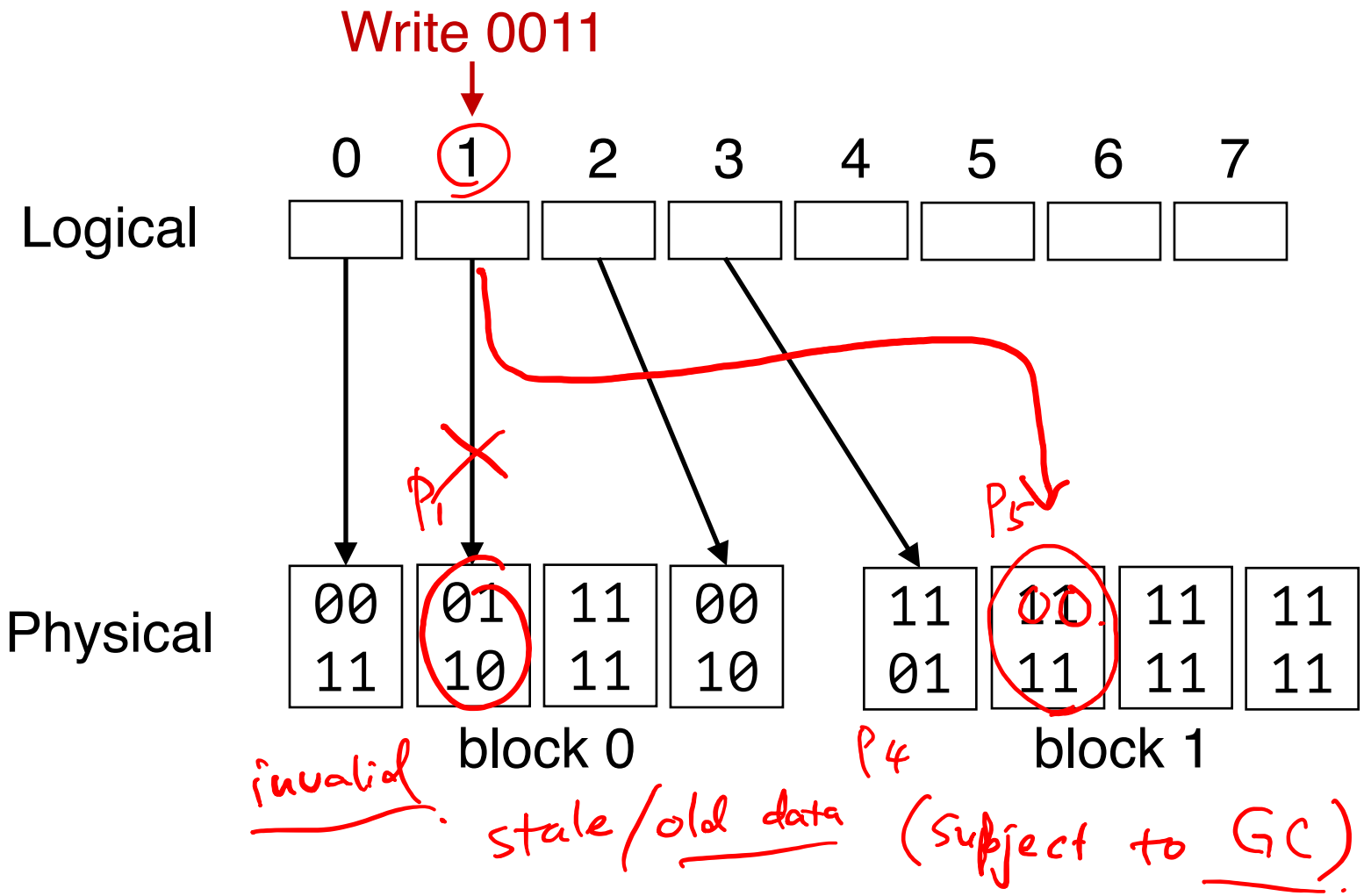
Flash Translation Layer (FTL)

- Add an address translation layer between upper-level file system and lower-level flash
  - Translate logical device addresses to physical addresses
  - Convert in-place write into append-write
  - Essentially, a virtualization & optimization layer

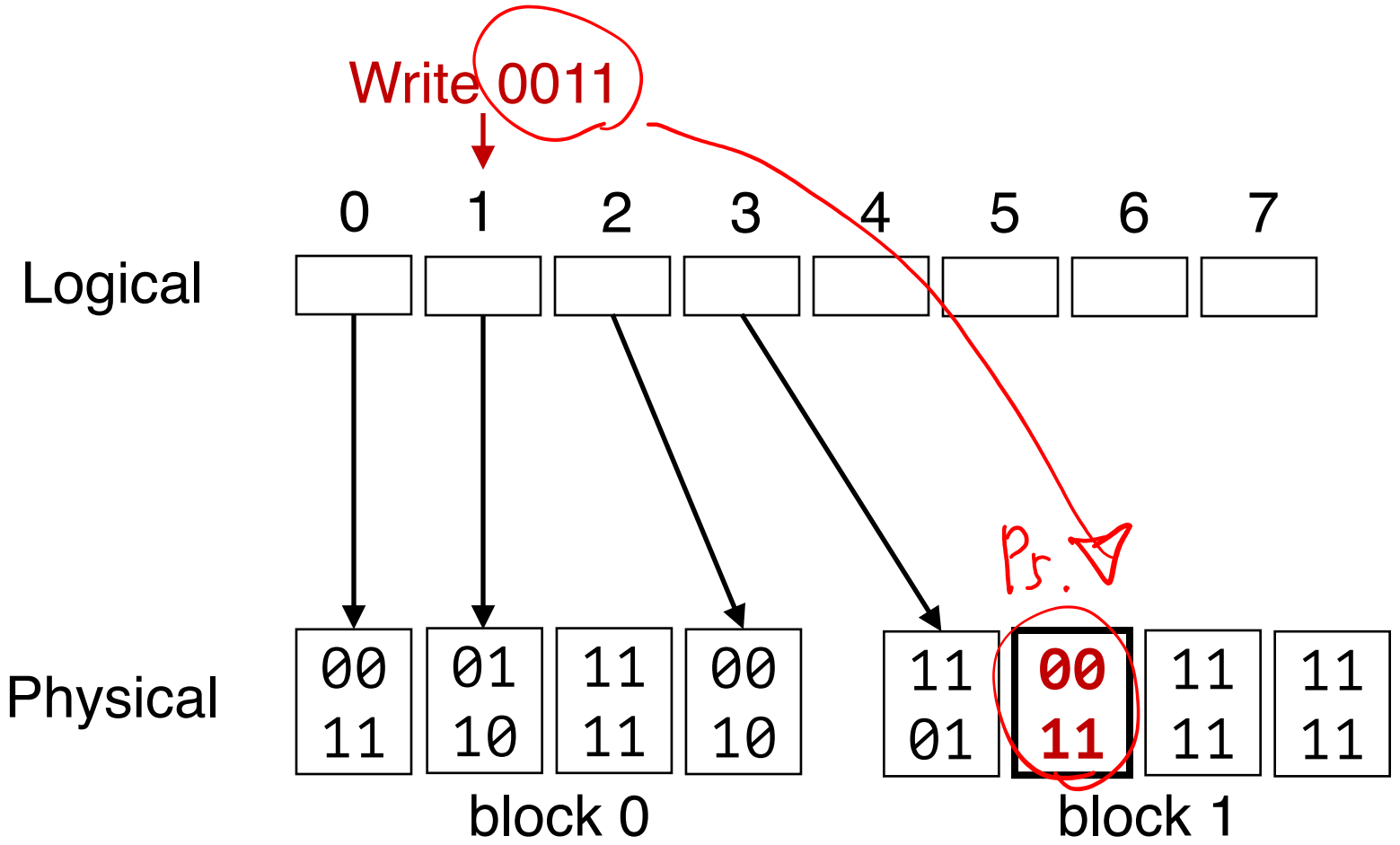
Flash Translation Layer (FTL)



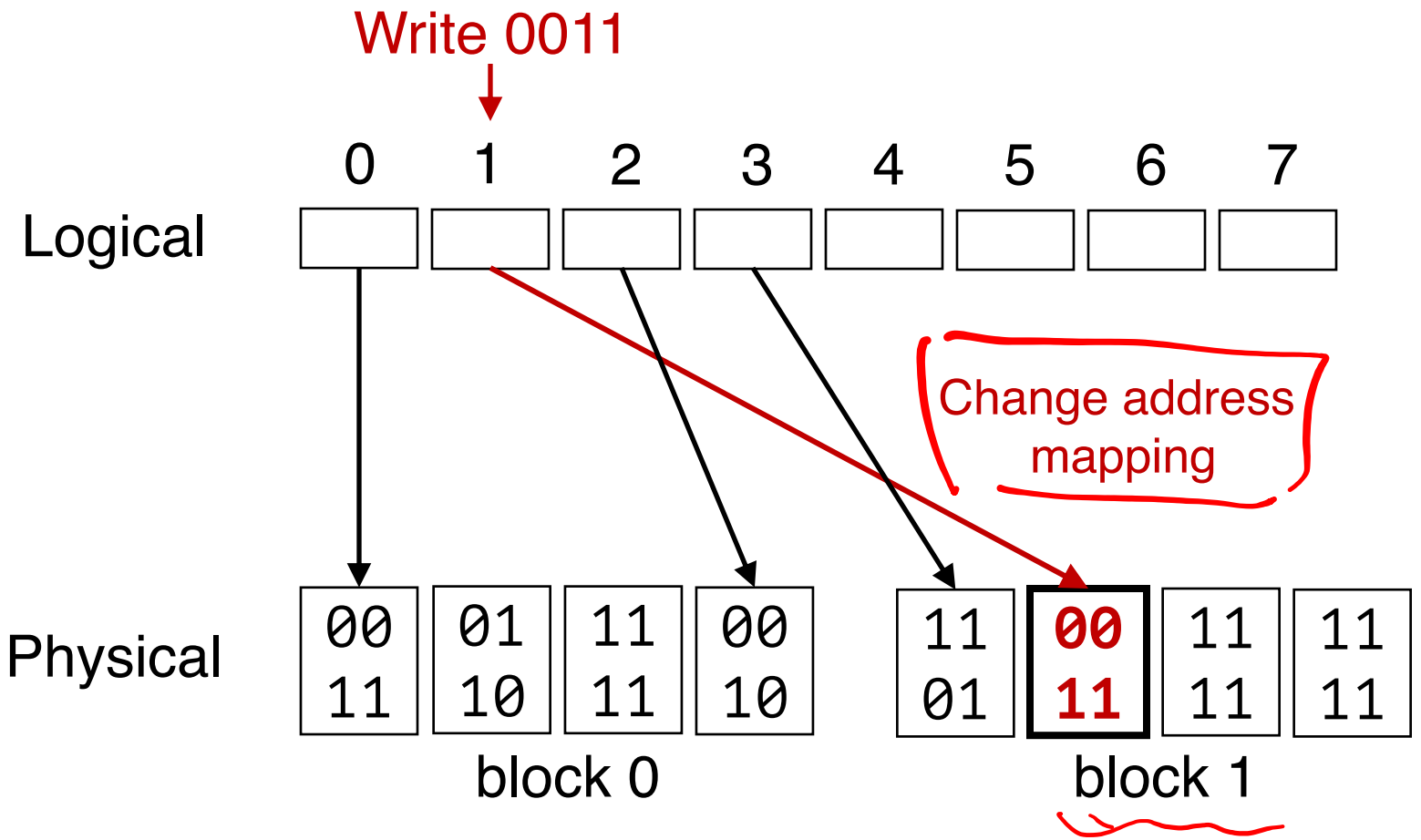
Flash Translation Layer (FTL)



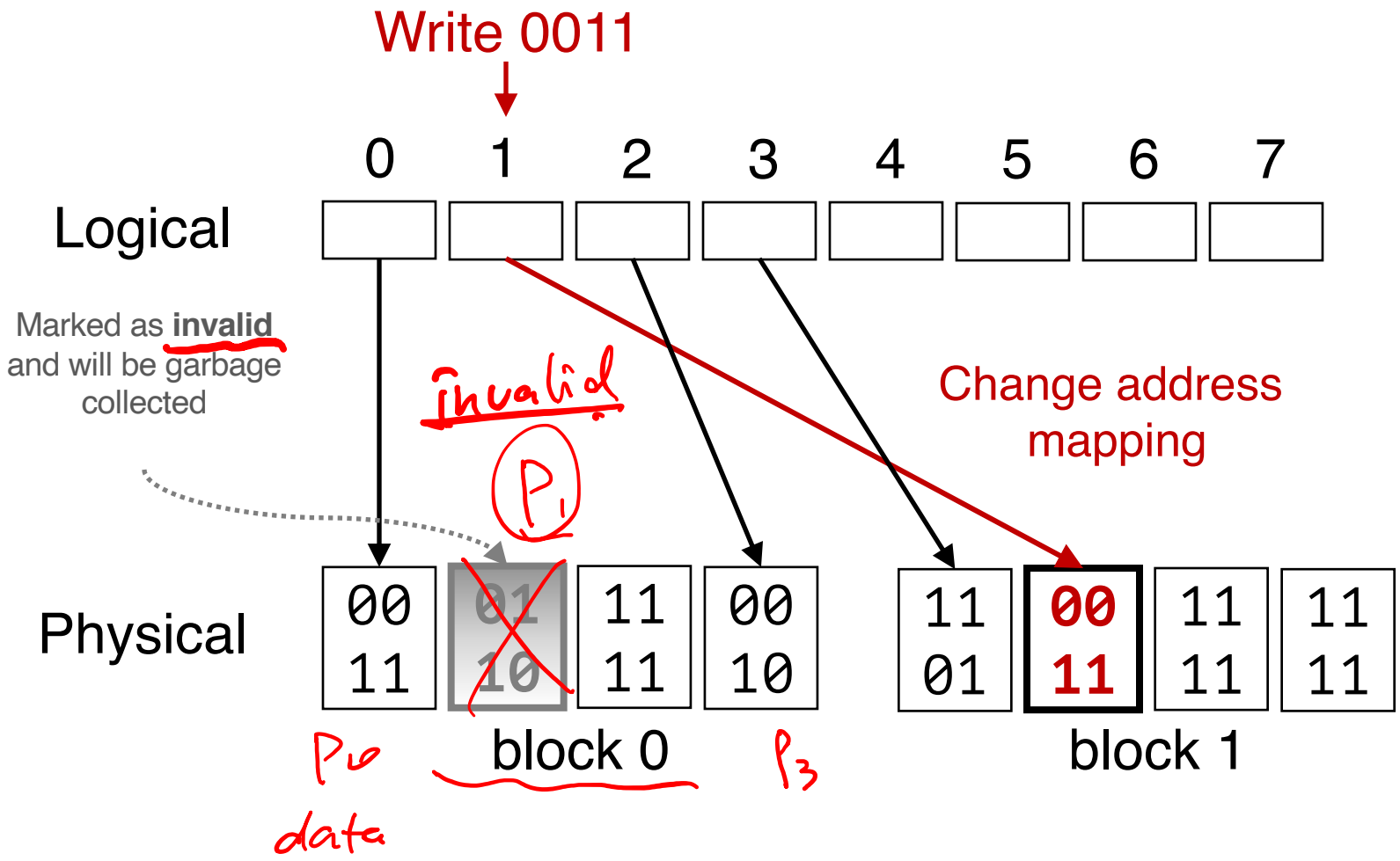
Flash Translation Layer (FTL)



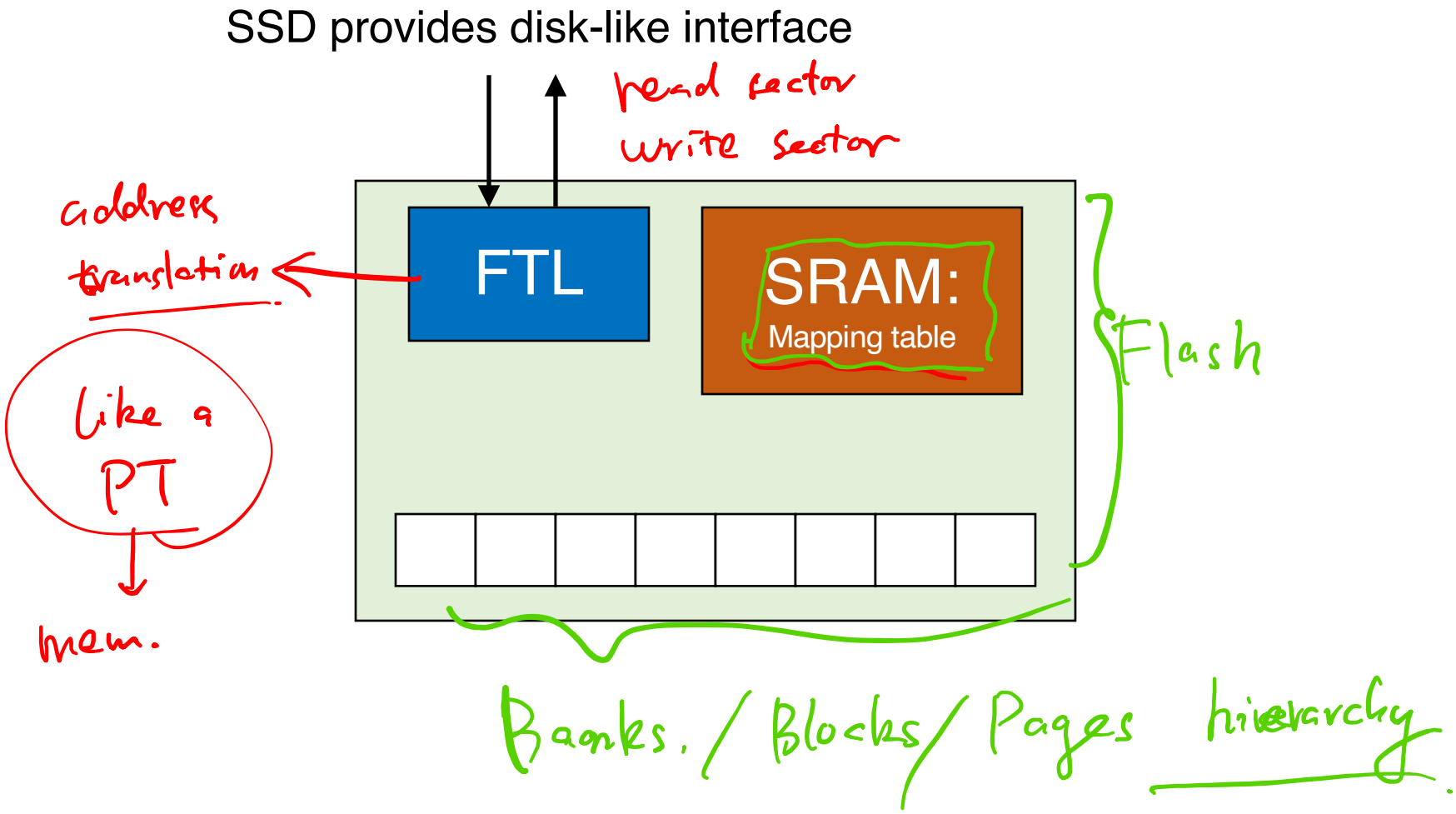
Flash Translation Layer (FTL)



Flash Translation Layer (FTL)



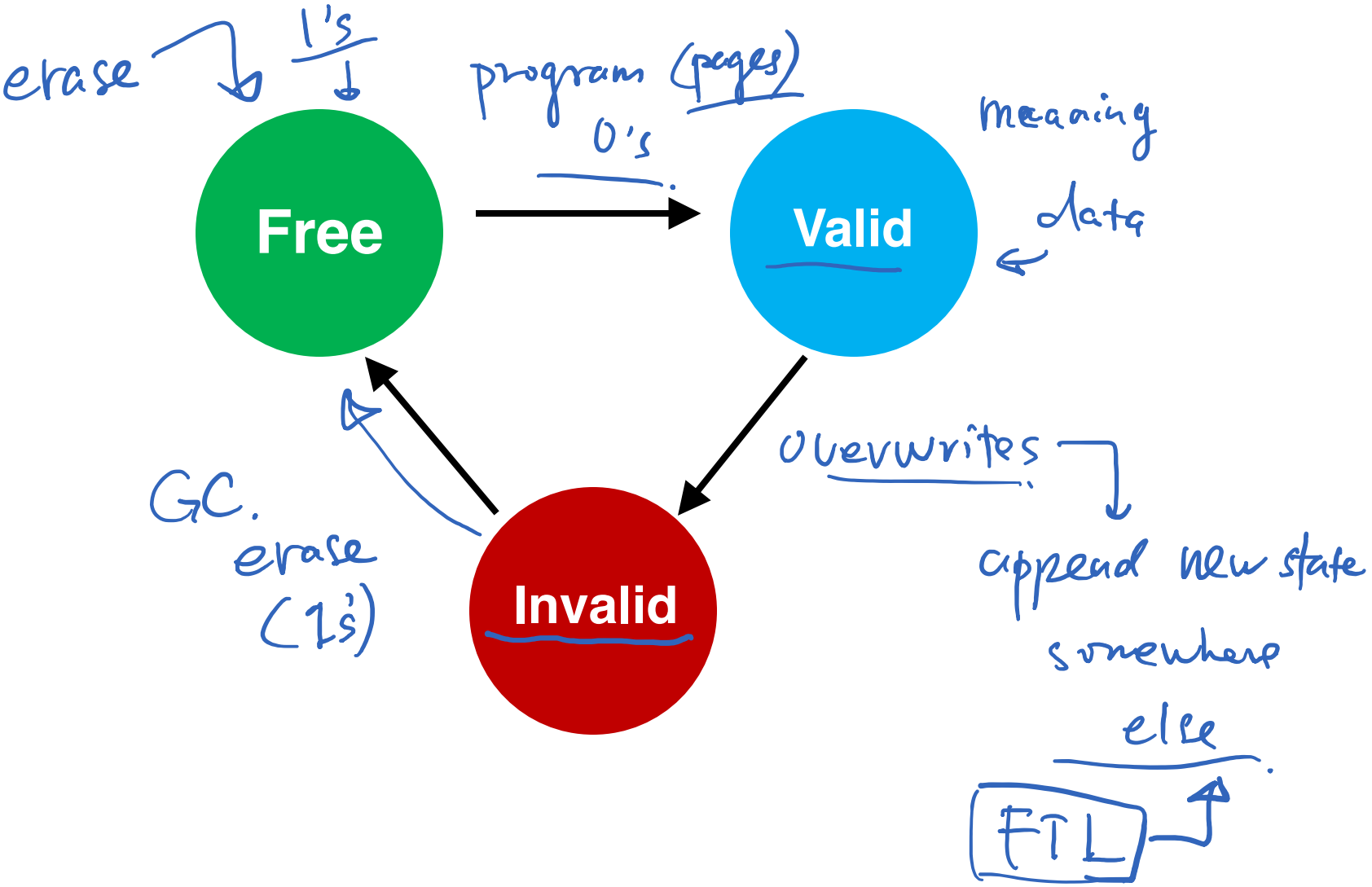
SSD Architecture with FTL



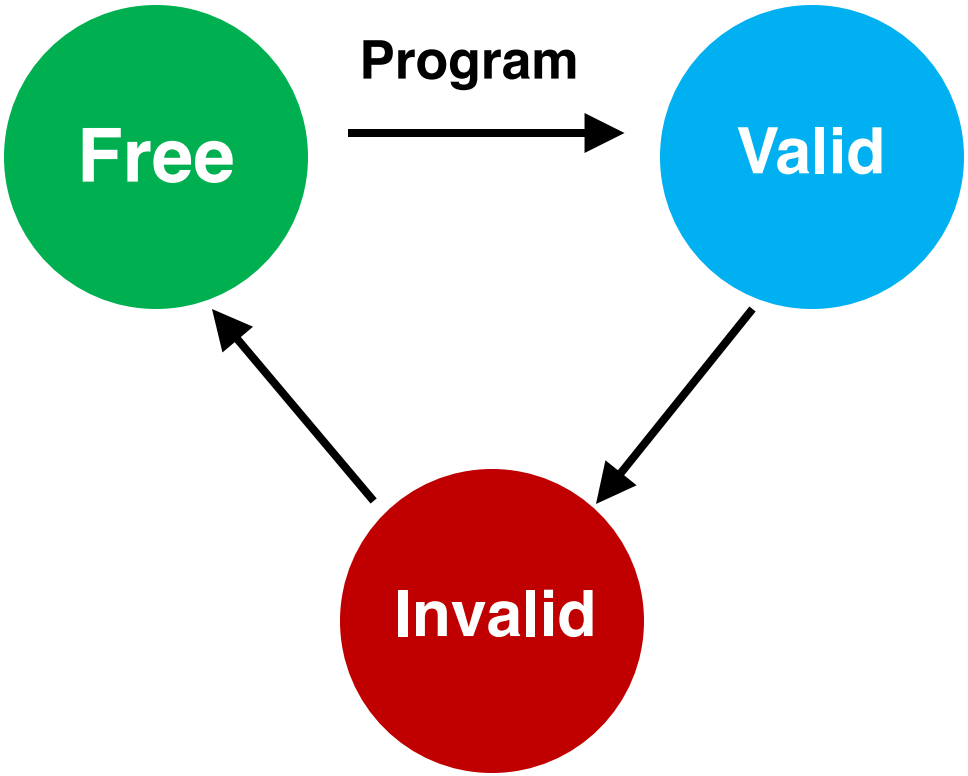
Flash Translation Layer (FTL)

- Usually implemented in flash device's firmware
- Where to store mapping?
  - SRAM
- Physical pages can be in three states
  - valid, invalid, free

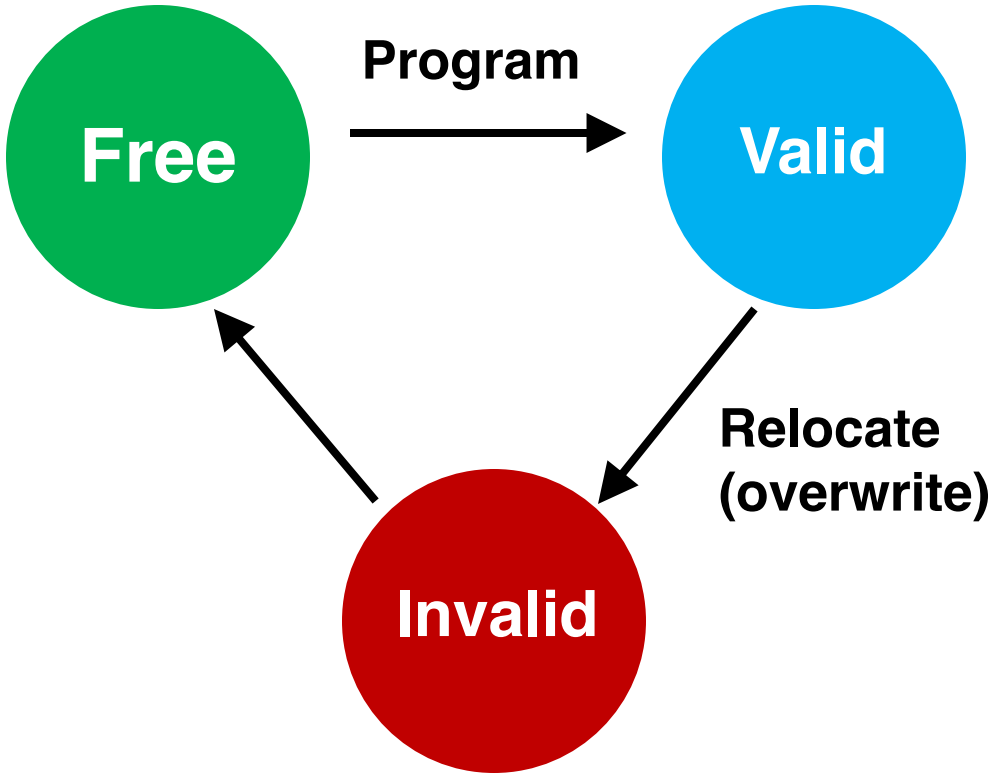
State Transition of Physical Pages



State Transition of Physical Pages



State Transition of Physical Pages



State Transition of Physical Pages