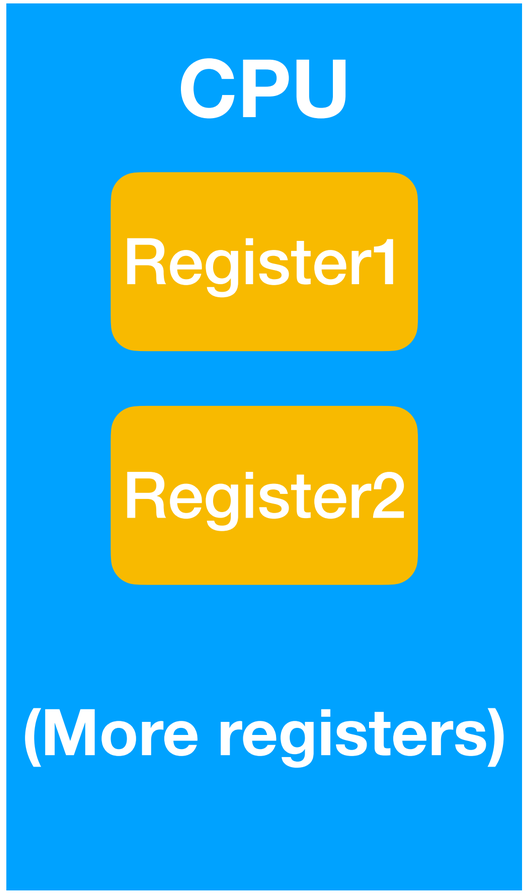
Review: important OS concepts

- Time-sharing, context, context-switch
- Interprocess communication
- Exception control flow
- Priority and scheduling
- **Cache and memory hierarchy (this lecture)**

Cache & Memory Hierarchy

Recall our abstraction: CPU + memory

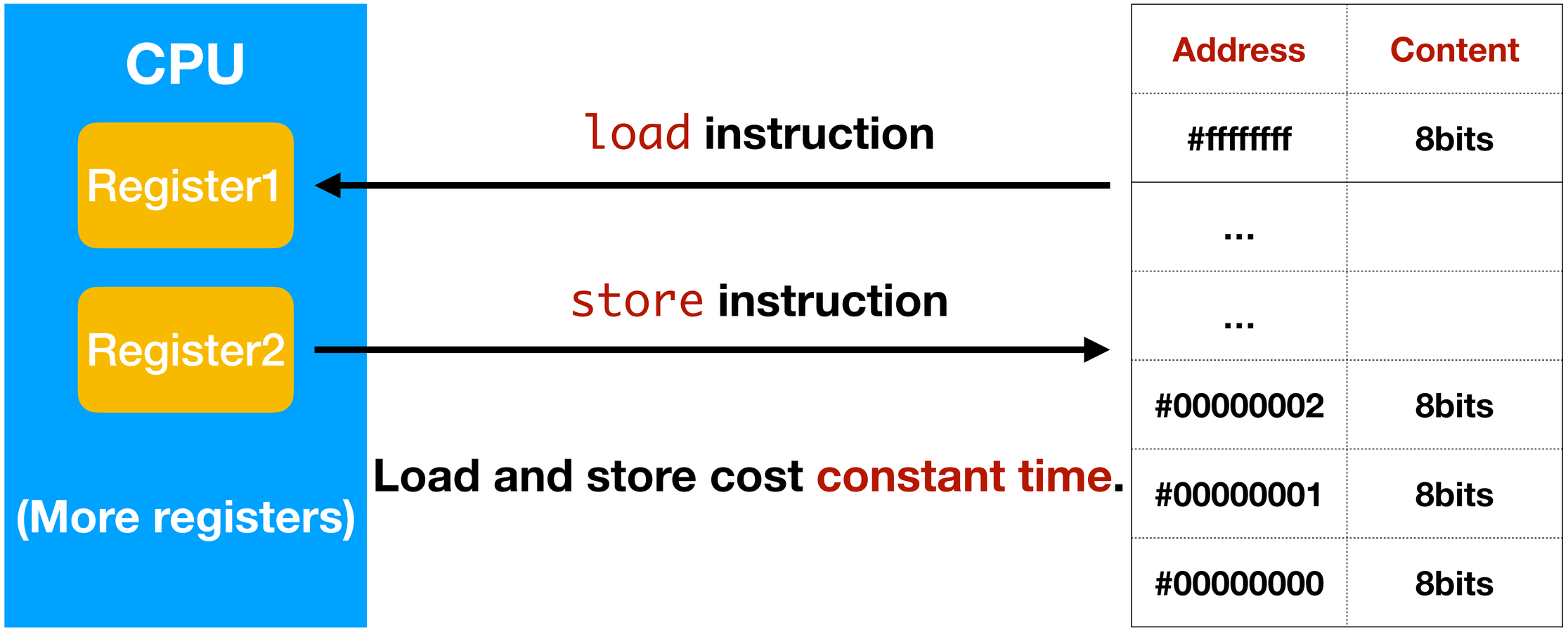


CPU has a number of registers.

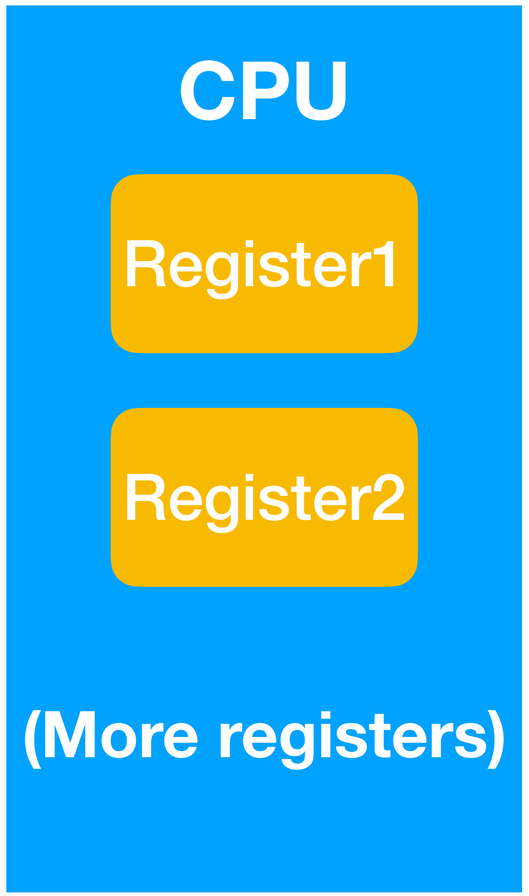
Memory is a two-column table.

Address	Content
#ffffffff	8bits
...	
...	
#00000002	8bits
#00000001	8bits
#00000000	8bits

Recall our abstraction: CPU + memory



Why cache in the middle?



Cache

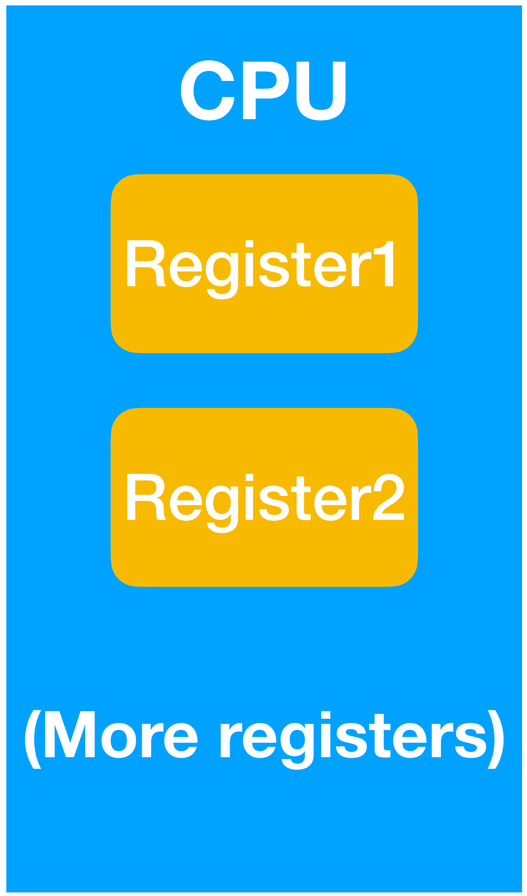
Cache is faster than memory.

Memory has larger capacity than cache.

Memory is cheaper than cache in terms of $/byte.

Address	Content
#ffffffff	byte
...	
...	
#00000002	byte
#00000001	byte
#00000000	byte

What to learn about cache?



Cache

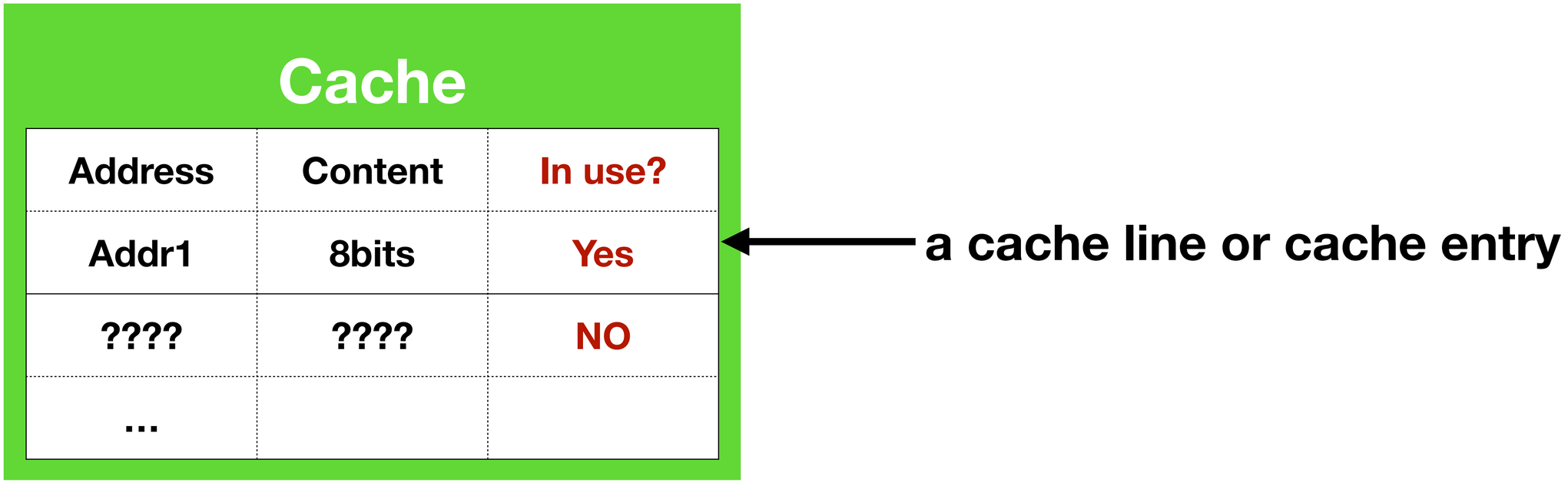
What is the interface of a cache?

What is the structure of a cache?

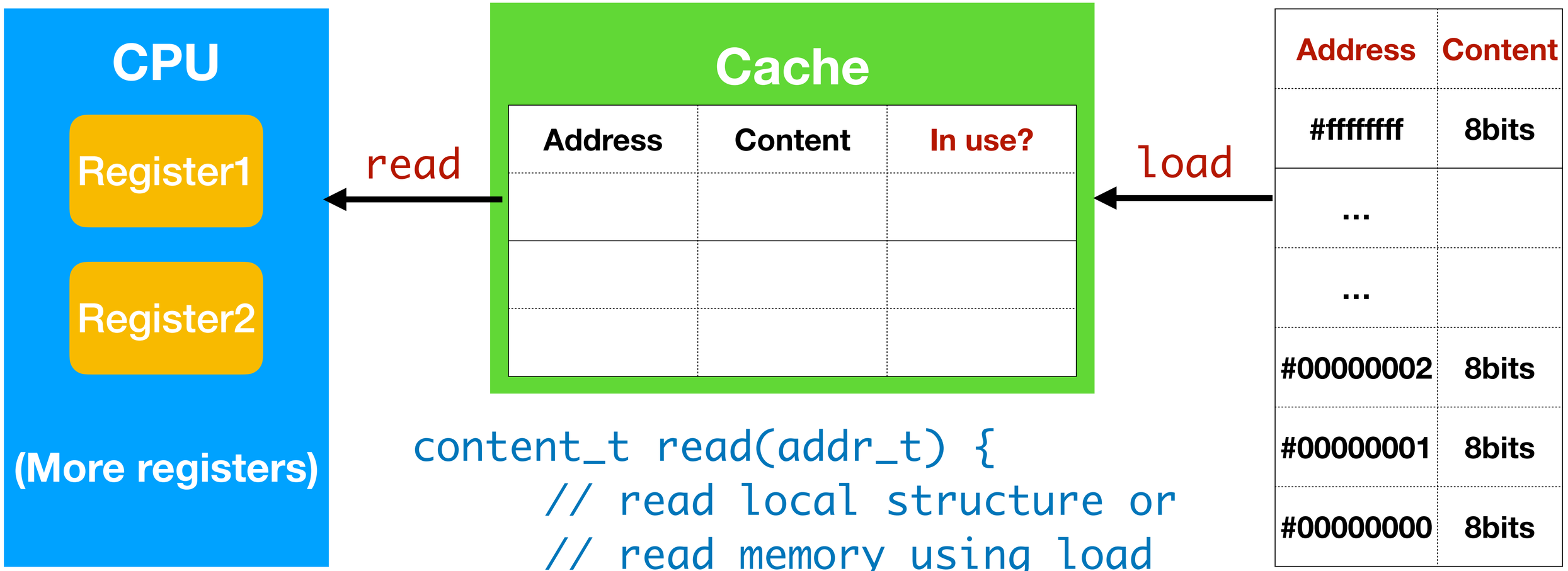
Address	Content
#ffffffff	8bits
...	
...	
#00000002	8bits
#00000001	8bits
#00000000	8bits

Write-through and Write-back cache

- Write-through and write-back are two types of cache.
  - You are going to implement both in P3.
  - The general structure is a 3-column table.

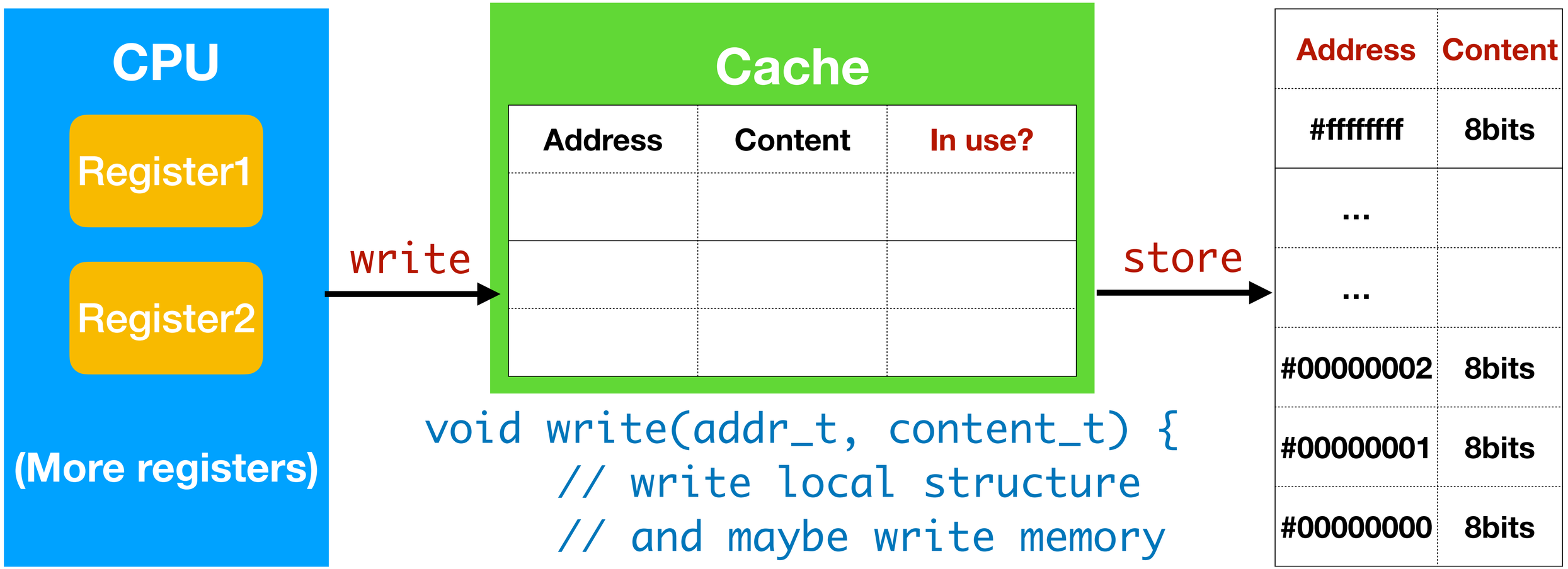


Read in Write-back and Write-through



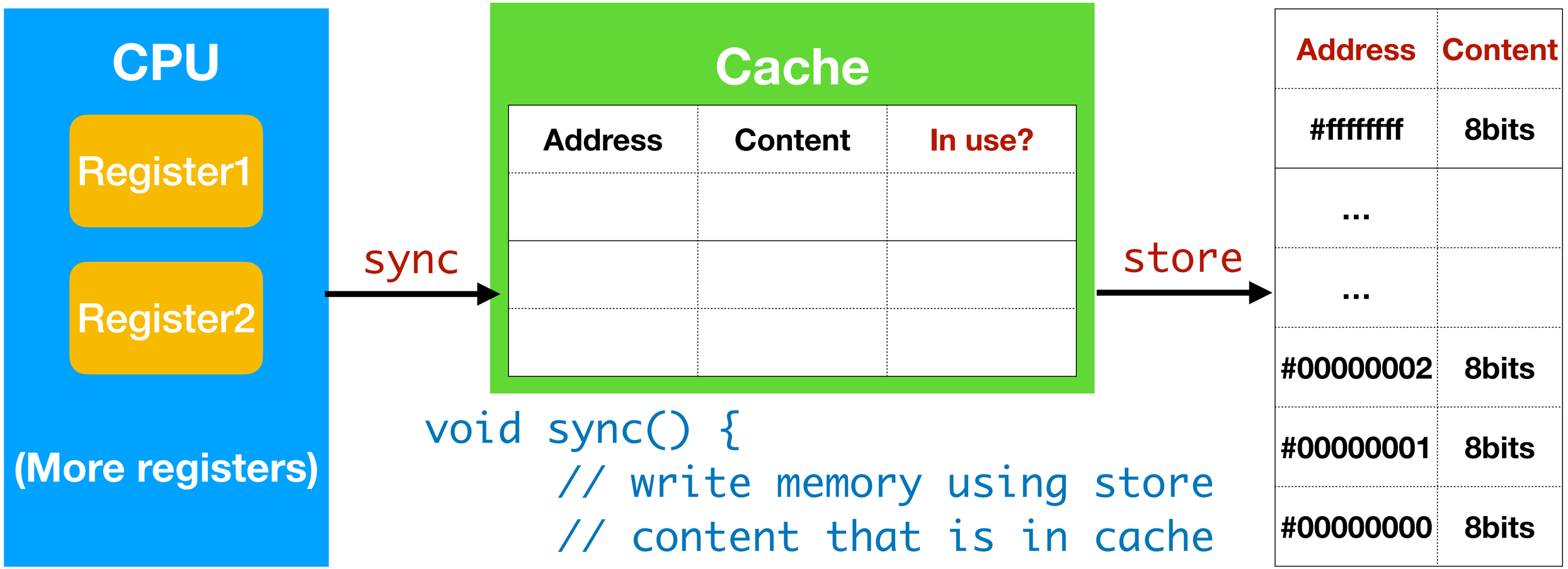

```
content_t read(addr_t) {
        // read local structure or
        // read memory using load
}
```

Write in both Write-back and Write-through




```
void write(addr_t, content_t) {
        // write local structure
        // and maybe write memory
        // using store
}
```

Sync (or flush) in Write-back cache




```
void sync() {
        // write memory using store
        // content that is in cache
        // but not yet in memory
}
```

Write-through and Write-back cache

- Write-through and write-back are two types of cache.
  - You are going to implement both in P3.
  - The general structure is a 3-column table.
  - Write-through cache:
    - read + write using load + store
  - Write-back cache:
    - read + write + sync using load + store

Cache

Address	Content	In use?
Addr1	8bits	Yes
????	????	NO
...		

Question: what about dirty bit?

When is a dirty bit useful?

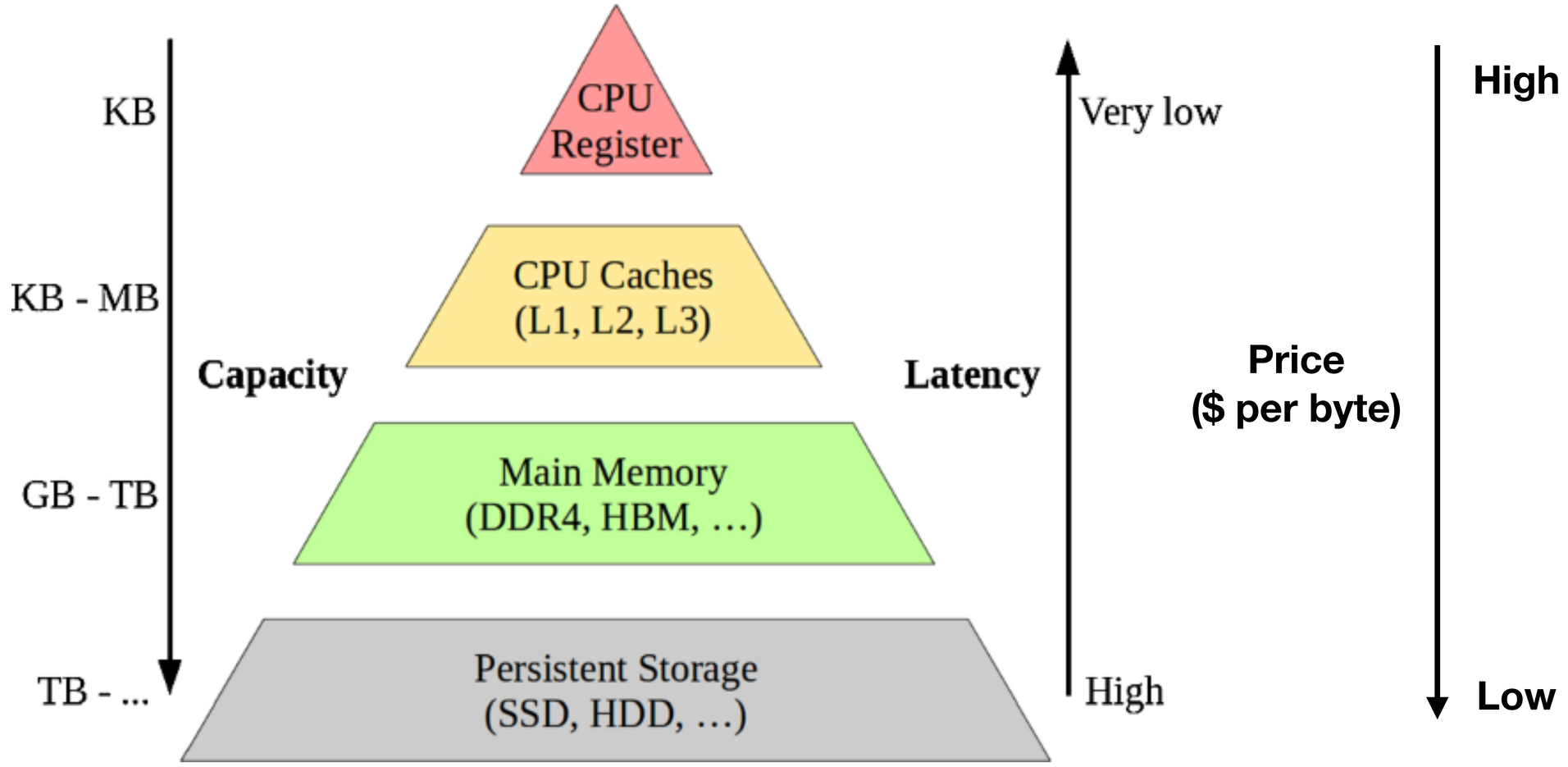
You may recall something called dirty bit that you learned in 3410.

Cache eviction

- When the cache is full and a new entry needs to be added, the cache evicts an entry back to the memory.
  - In write-through cache, the evicted cache entry does NOT need to be stored back to memory.
  - In write-back cache, the evicted cache entry, if dirty, needs to be stored back to memory.
- In P3, you will implement the CLOCK algorithm for cache eviction which will be taught in 4410 (Oct 27).

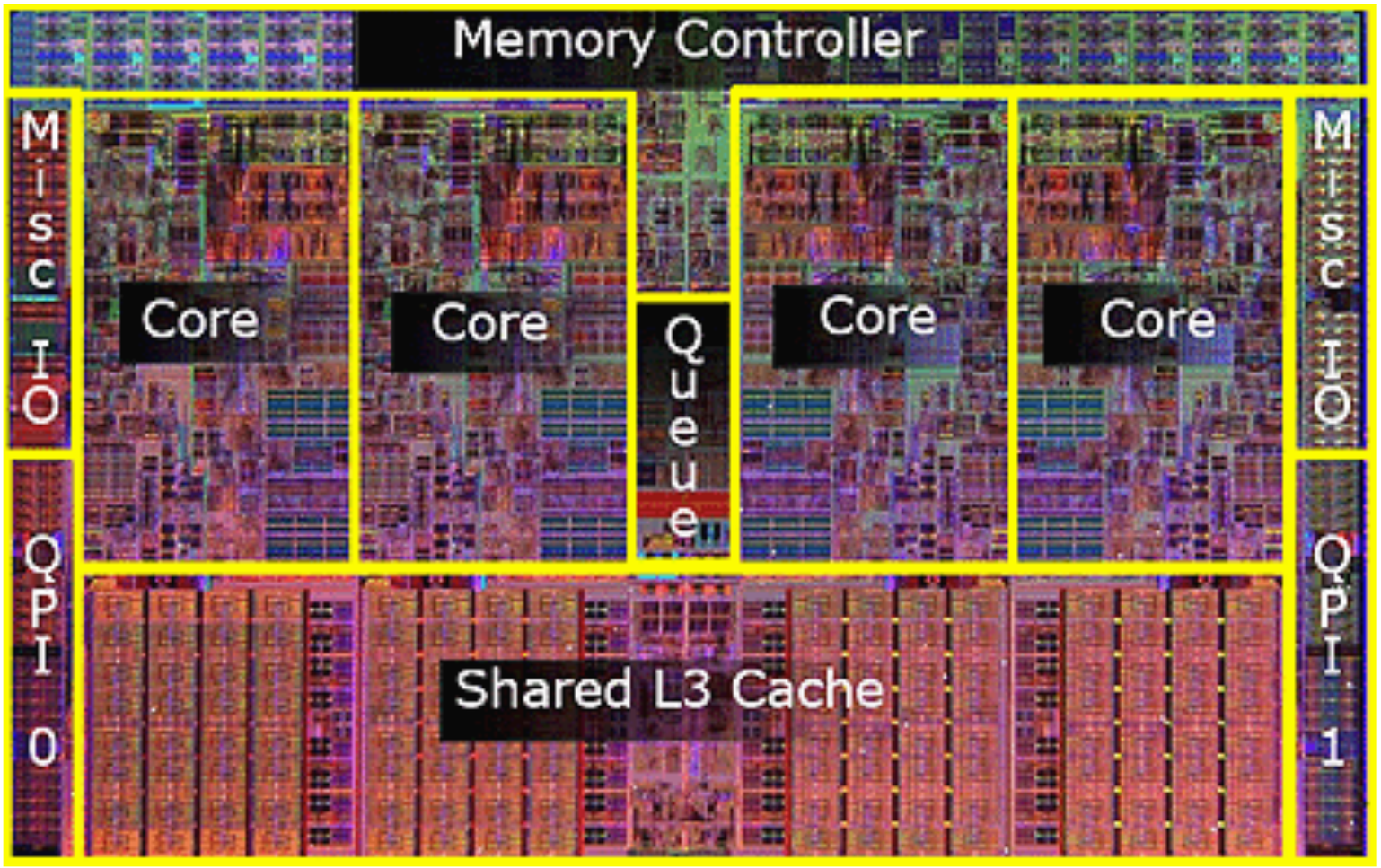
Cache & Memory Hierarchy

Memory Hierarchy



Picture source: https://link.springer.com/article/10.1007/s00778-019-00546-z

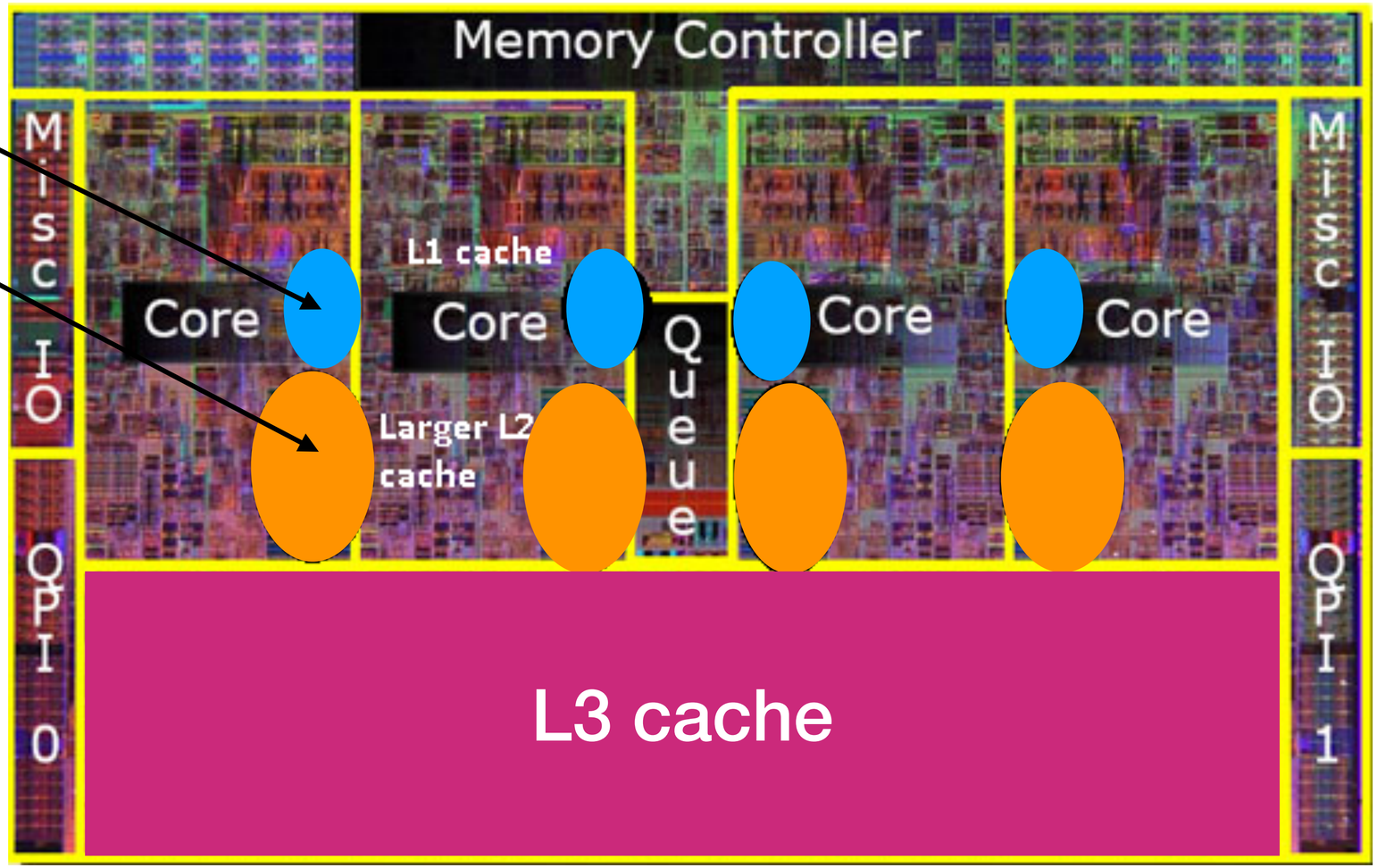
Example: internal of Intel i7 CPU



CPU cache hierarchy

L1 cache

L2 cache



CPU cache hierarchy

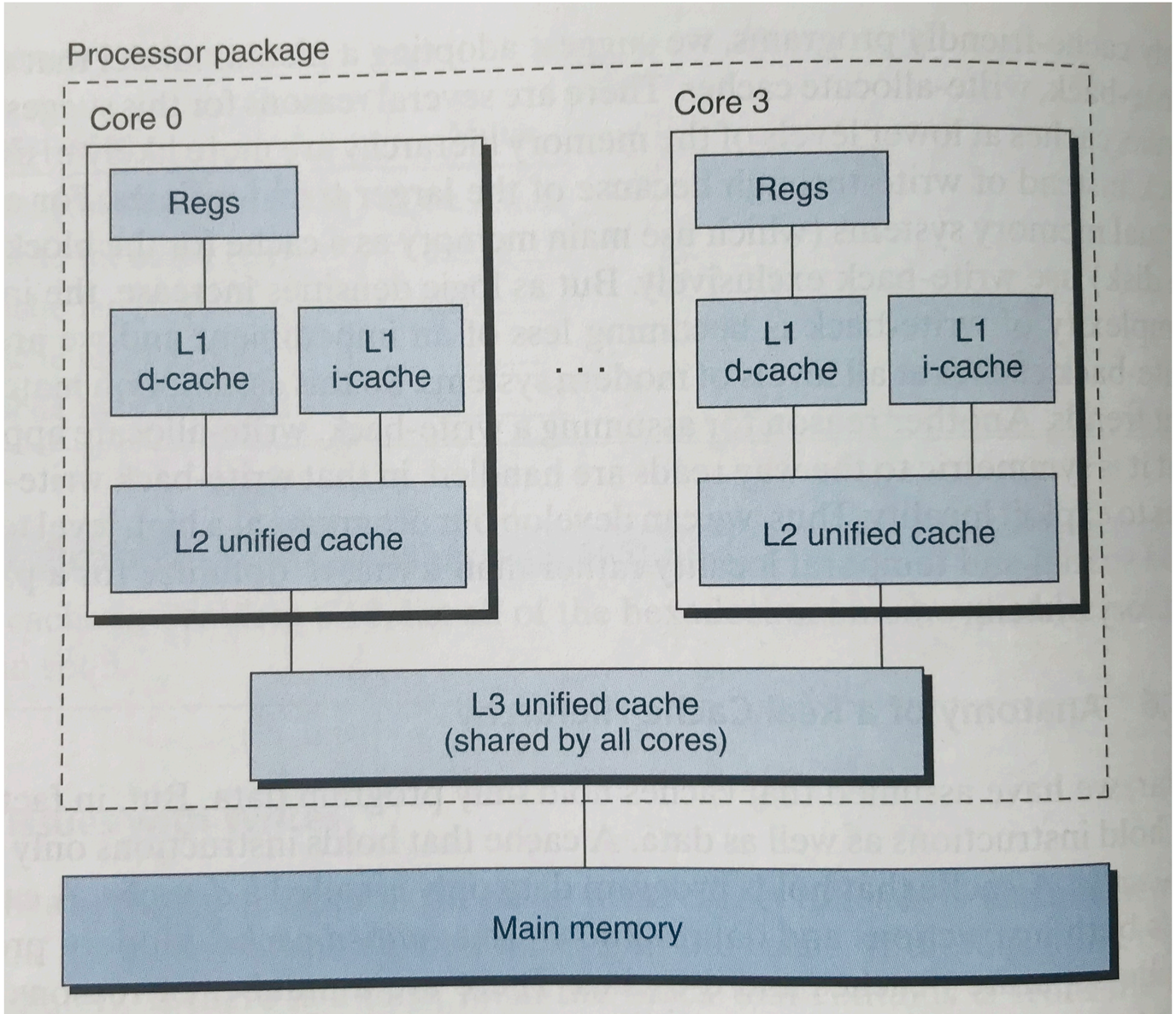
Registers

L1 cache

L2 cache

L3 cache

Main memory



From Figure 6.39 of Computer Systems A Programmer's Perspective

Memory hierarchy performance and capacity

Cache level	Access time	Capacity
L1	4 cycles	32KB
L2	10 cycles	256KB
L3	40-75 cycles	8MB
Main memory	200 cycles	4-16GB
Disk	>1M cycles	>1TB

Take-aways

- Cache makes memory access faster, but cache has smaller capacity and is more expensive.
- Different levels of cache form a memory hierarchy.
  - CPU cache hosts KB and costs tens of CPU cycles
  - Main memory hosts GB and costs hundreds of CPU cycles
  - Disks hosts TB and costs millions of CPU cycles

Homework

- P3 is released today due on Nov 6. Implement write-back and write-through cache with the CLOCK algorithm.
- Read page241 of the Intel's IA-32 manual Volume2 (https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-instruction-set-reference-manual-325383.pdf) about the CLFLUSH instruction.

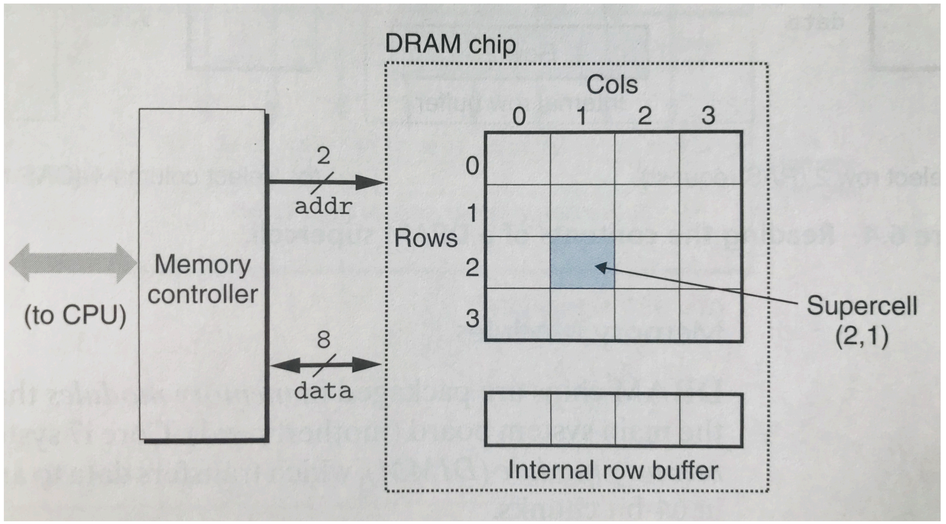
Just for fun

- Main memory internal structure and row-hammer attack

What is inside here?

Just for fun

- Main memory internal structure and row-hammer attack
- Further reading: section 6.1 of Computer Systems A Programmer's Perspective.



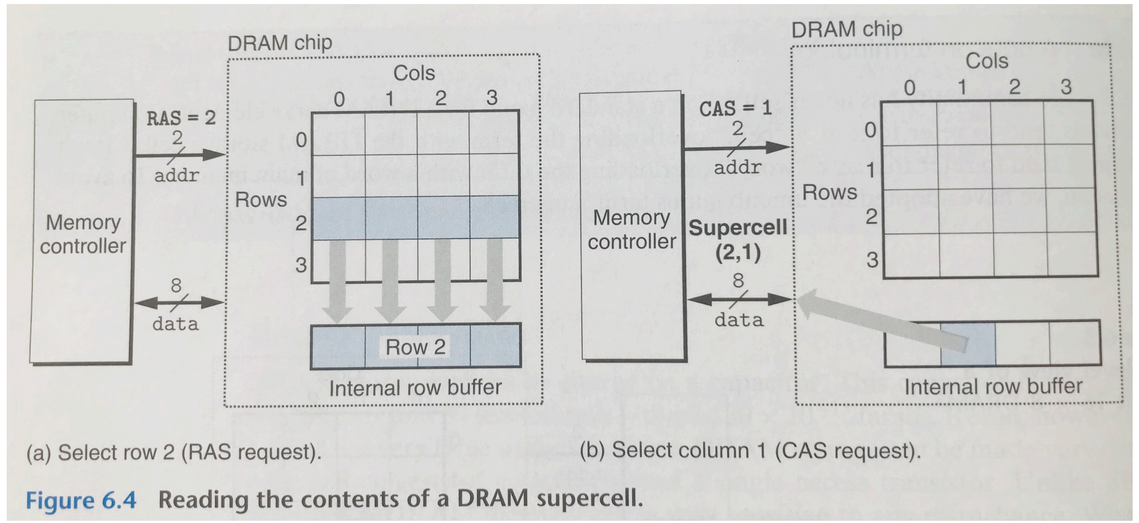


Figure 6.4 Reading the contents of a DRAM supercell.