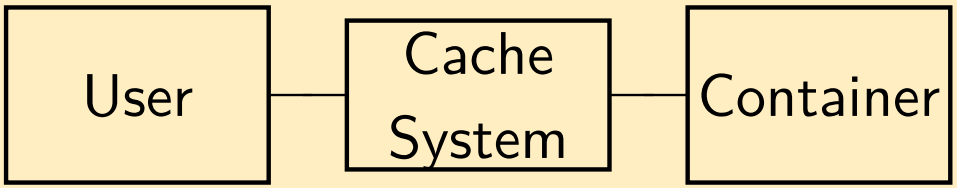
Cache

January 29, 2025

The General Problem

- The container is large and slow
- The user is very fast
- The user has ‘locality of reference’

Solution



- The cache is **faster** than the container
- The cache holds relatively a **small** amount of data
- Temporarily, the cache holds replicas of popular data
- (Thus, per bit, the cache is more expensive)

Without locality of reference the cache is useless

Key

- Each item in the container has an address
- The user uses this address to read/write the item
- This must **not** change with the addition of the cache
- Thus the cache needs to use the address as a key

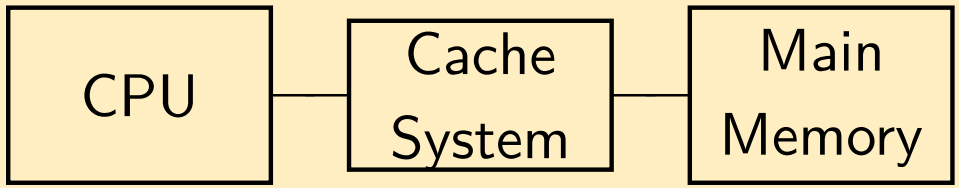
The Dictionary Data Structure

key1	value1
key2	value2
⋮	⋮
⋮	⋮

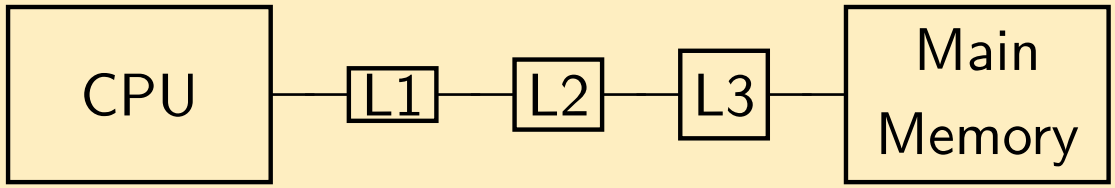
- Hardwarewise, dictionary is ‘associative memory’

The Cache

- Classically **the cache** is an SRAM unit sitting between the CPU and the main memory



- However, nowadays we have a sequence of caches



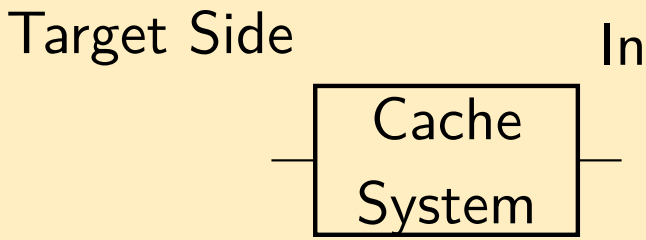
- L4 cache is not unheard of (as of 2023)

All (most) cahches today are on the CPU die

In the olden days, the (one) cache was farther away

Target/Initiator Sides

- Note the cache talks with buses
- So, in princple, it does not know who sits on the other side of the bus
- Most cache operations are indifferent to positioning
  - Single, L1,L2,L3
- For a uniform discussion we use the following:

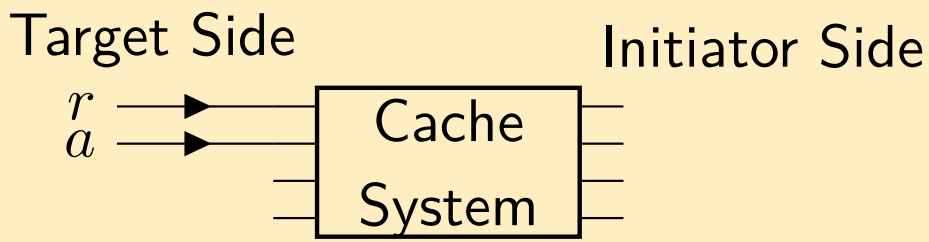


Read Command to the Cache

Read is simple. No variants.

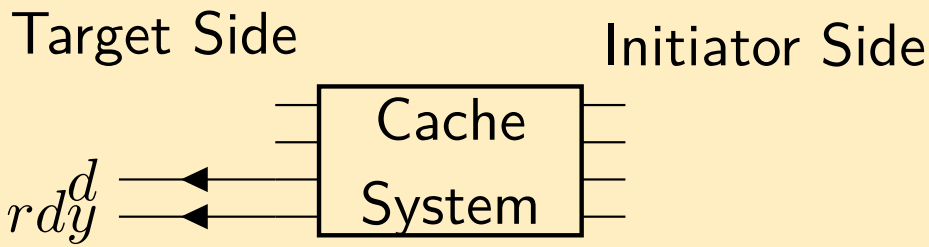
Read and Hit

Read request



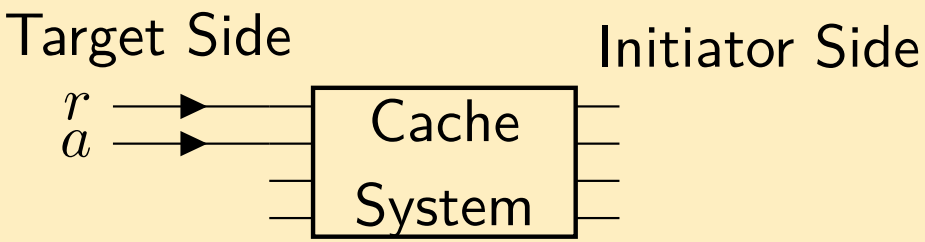
Item found in cache memory

Data sent to faster system



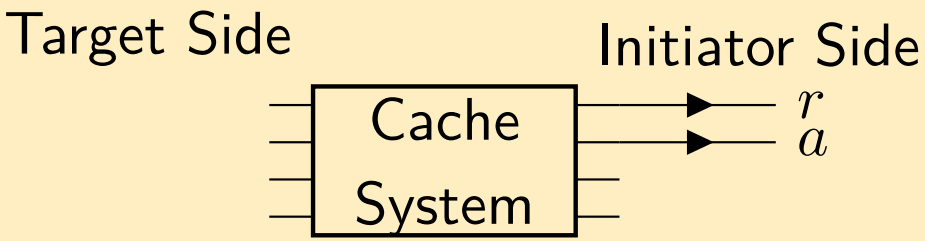
Read and Miss (1)

Read request



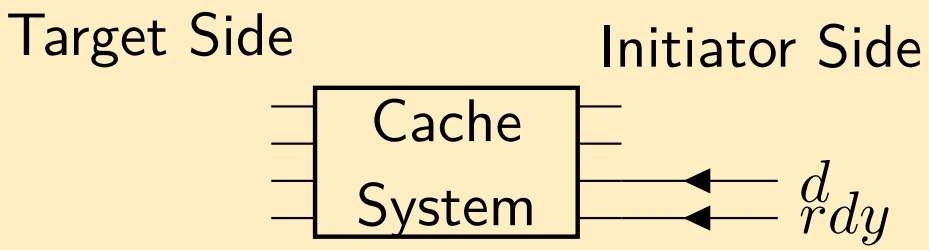
Item is not in cache memory

Request passed to slower system



Read and Miss (2)

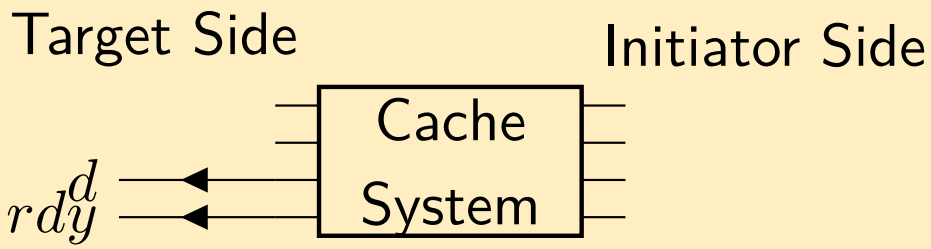
Data received from slower system



Item is saved in the Cache **Memory**

This might lead to some other item to be **evicted**

Data sent to faster system



Notes on Read

- Accessing items radomly once will probably yield **misses**
- Repeatedly accessing an item will probably yield miss followed by hits
- For large blocks, sequential access will also yield miss followe by hits

Direct Mapping Cache

Cache Memory in the Cache System

- The structure of the cache memory

tag	block/line	flags
⋮	⋮	⋮
⋮	⋮	⋮

- The flags contains at least the $\mathrm{V}alid$ flag
- The block size is always $2^n B$ for some n
- The number of blocks is always 2^k for some k
- The size of the cache is $2^k \times 2^n$B
- (The tag and flag fields are ignored for size calculations)
- The cache memory has its own addresses: $0 - (2^k - 1)$
- We call this address **the index**
- (This helps us)