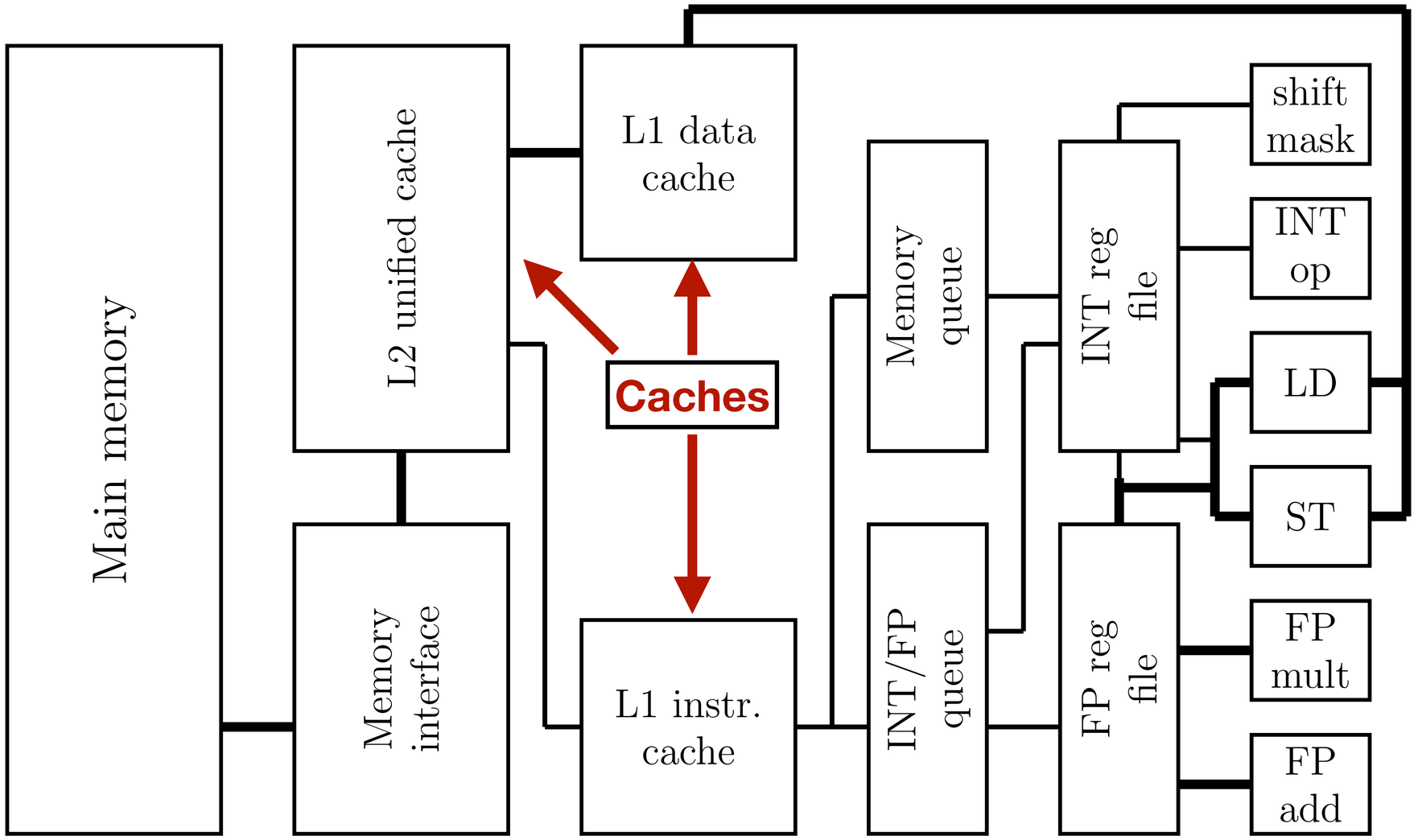
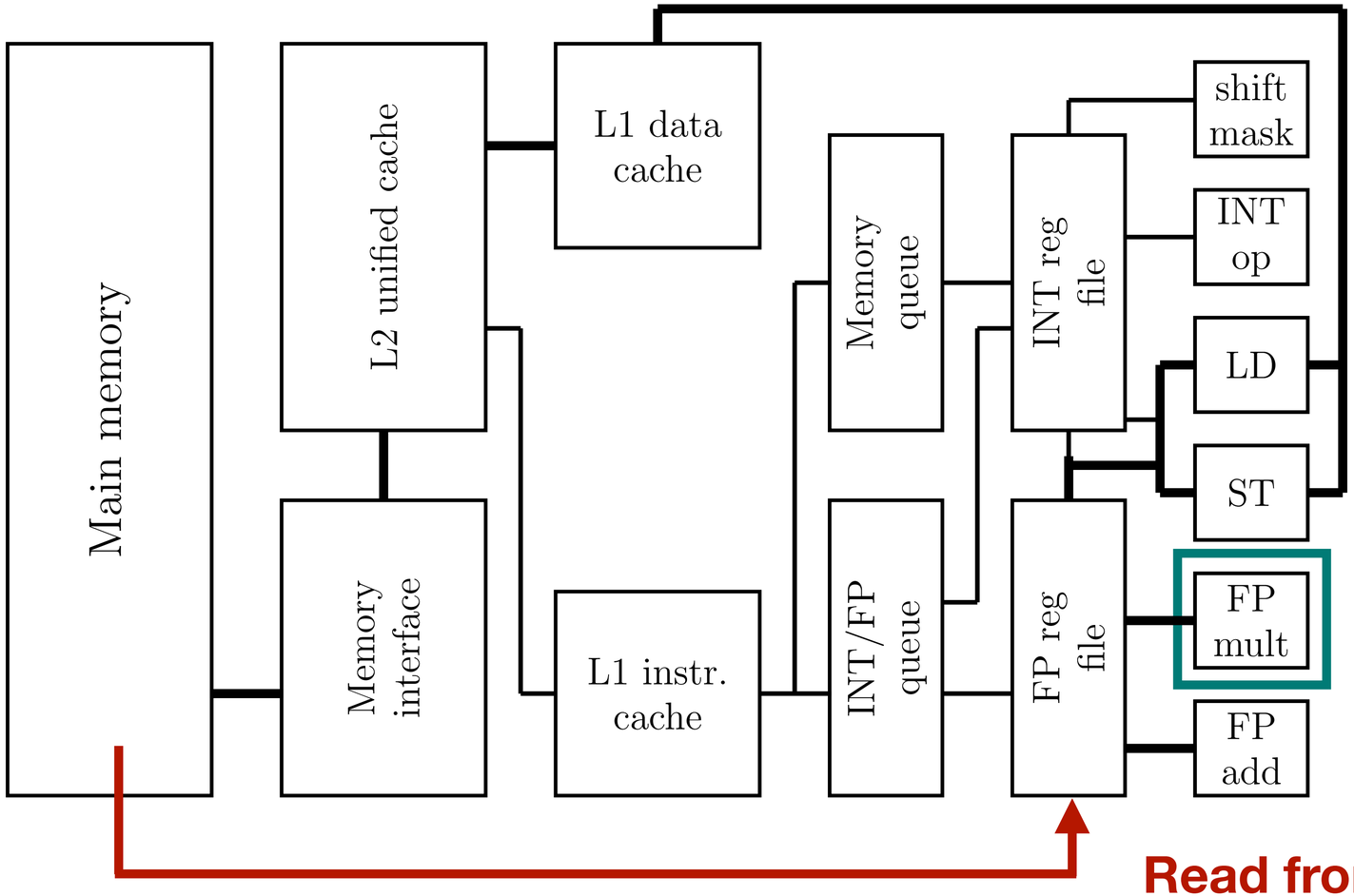
Serial Optimizations

08/21/2020

Cache-Based Microprocessor



Why Do We Have Caches

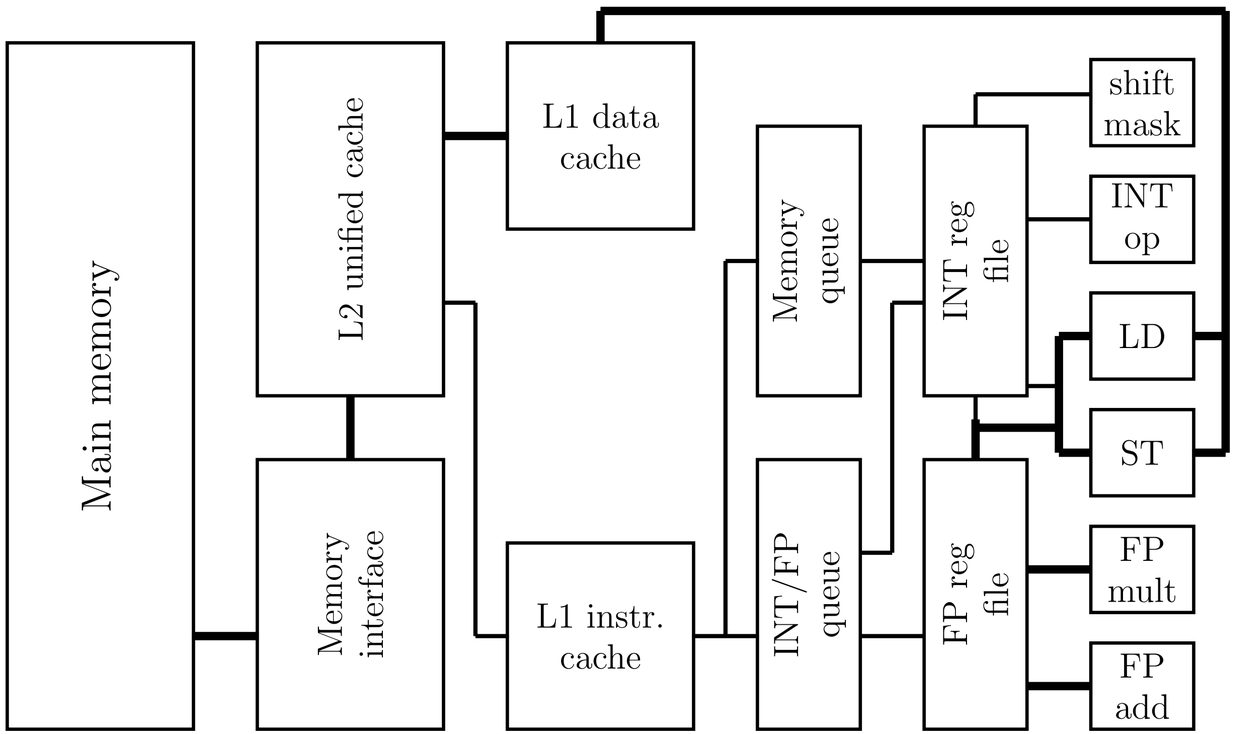


Floating-Point Mult: Several GFlops/Sec

Read from memory: A few GBytes/Sec

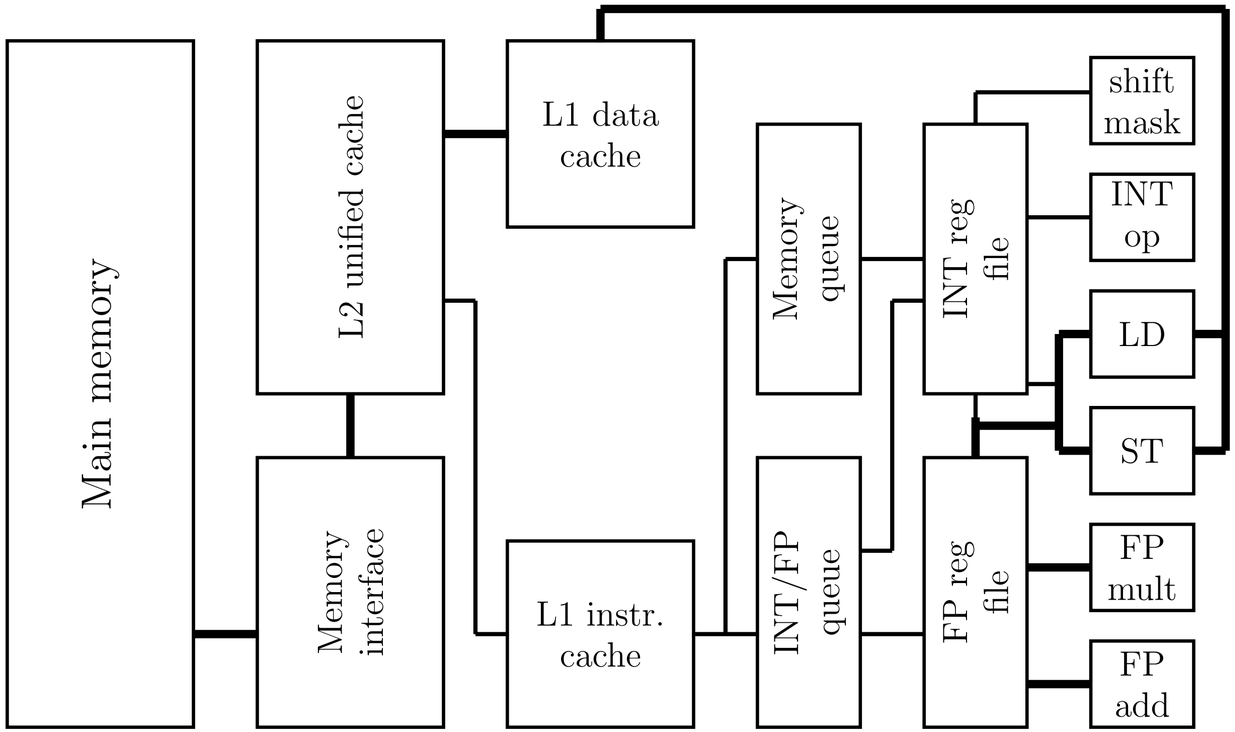
Cache Overview

- Typically at least 2 caches
- L1 is split (data, instr.)
- Others are unified
- Cost of reading data:
  - Latency
  - Bandwidth
- **Closer the cache is to register, lower latency, higher bandwidth**



Cache Overview

- When reading data:
  - **Cache hit** : data in cache
  - **Cache miss** : not in cache
    - Outer cache or main memory
- If cache is full, data is evicted



Memory / Cache Latency

- Overhead for reading a single byte
- Not dependent on the number of bytes being read
- **Cache Lines :** a group of data is read from memory (say 64 bytes)
- During a cache miss, fetch entire line from memory
  - Neighboring items will then be read from cache

Writing Data is More Complicated

- **Write Hit :** data to be written already in cache
  - **Write-back :** modify cache line in cache, write to memory whole cache line when evicted
- **Write-Miss :** Cache line first transferred from memory to cache before being modified
  - Increased data transfer cost

Getting the Most Out of Cache

T_m	Time to access main memory
T_c	Time to access cache
β	Cache reuse ratio : fraction of loads and stores already in cache
τ	Factor by which cache is faster than memory

How Much Does Cache Help?

Component	Required Cycles	Cost, relative to previous
Register	1	
L1 Cache	A few	1-3x
L2 Cache	10	3-10x
Memory	250	25x

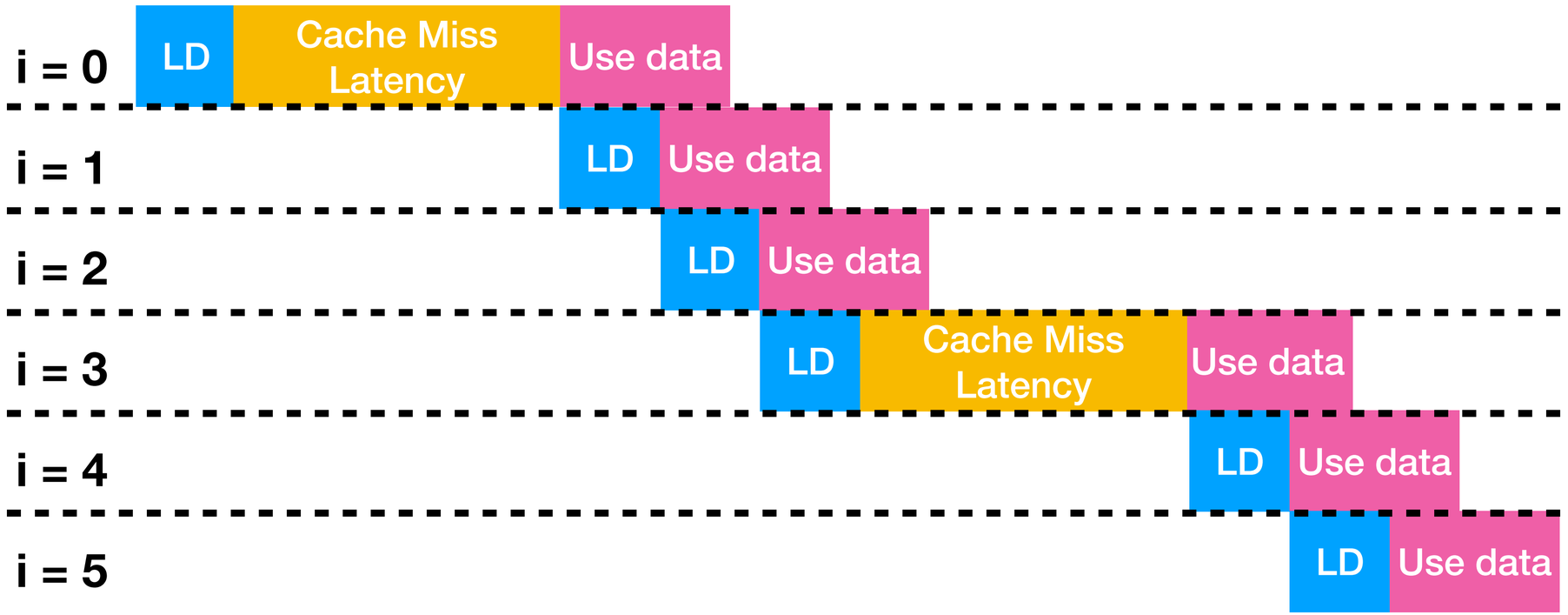
http://wgropp.cs.illinois.edu/courses/cs598-s16/lectures/lecture05.pdf

Prefetching

Cache miss has high latency, even with use of cache lines
Let's assume cache line size is 4 doubles below

Vector Norm :

```
for i = 0…n:
    sum += v[i]*v[i]
```

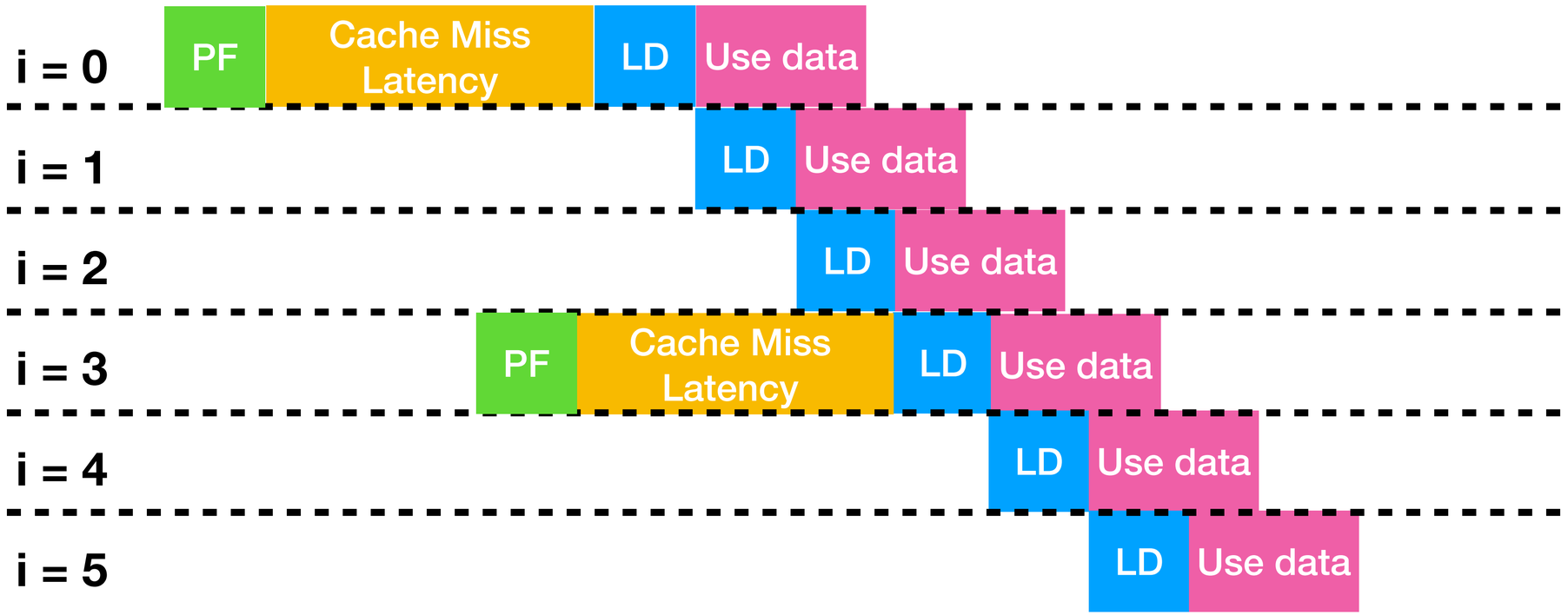


Prefetching

Load data into cache before it is needed by the application
Compiler adds instructions to touch cache lines early
Hardware prefetcher detects regular access patterns

Vector Norm :

```
for i = 0…n:
    sum += v[i]*v[i]
```



Vastly improves performance, but depends on regular access patterns

If you want to learn more…

- “Introduction to High Performance Computing for Scientists and Engineers” by Georg Hager and Gerhard Wellein
- “When Prefetching Works, When It Doesn’t, and Why”
- Some great slides on caches by Bill Gropp