Computer Fundamentals
Processor Architecture (3/3)

Grau en Intel·ligència Artificial

Xavier Martorell

Xavi Verdú

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC)

2021-2022 Q1

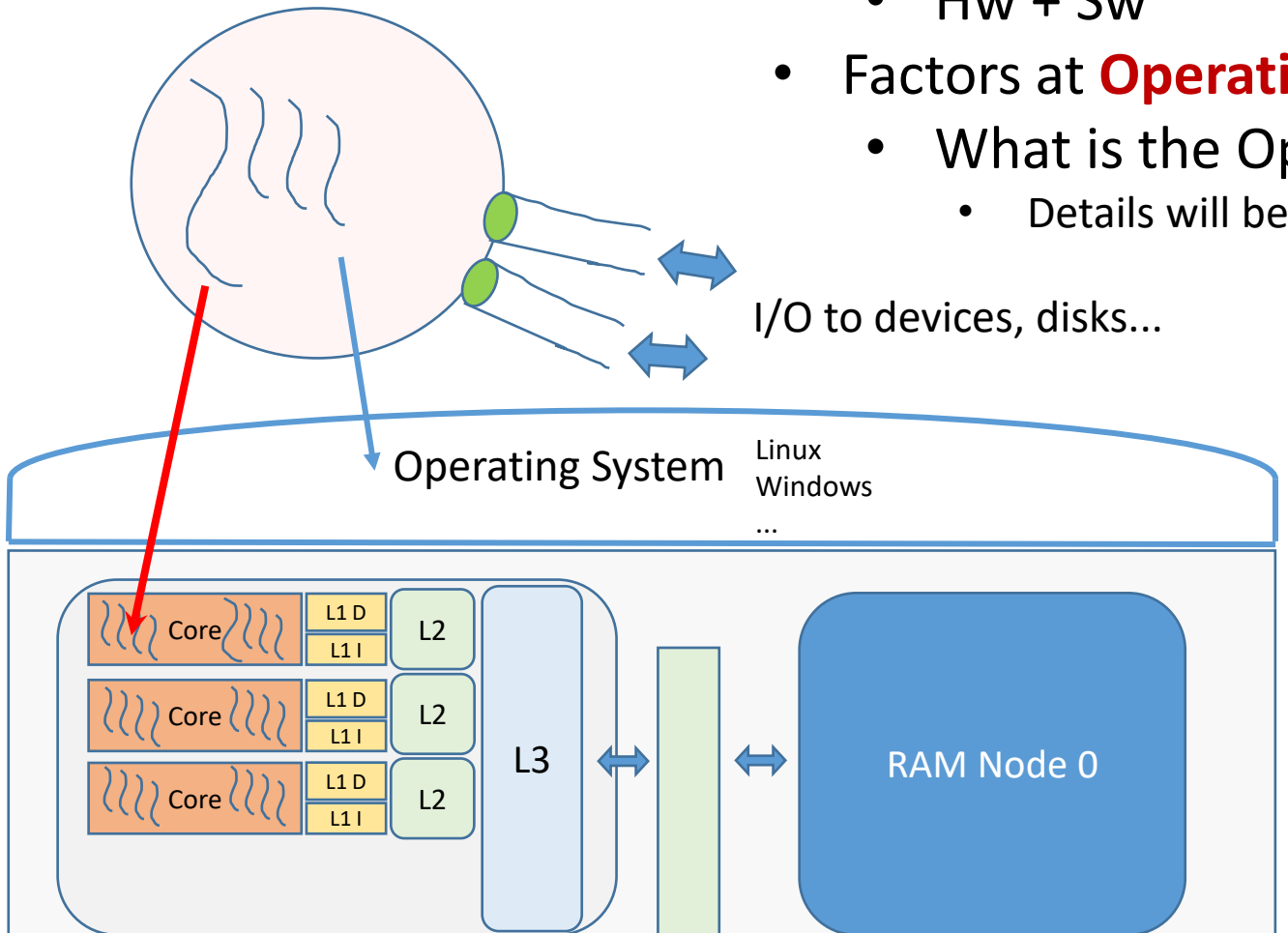
Creative Commons License

Table of Contents

- This lesson consists of three main parts
  1) Basic Concepts
  2) Memory Hierarchy
  3) **Execution flow, performance, bottlenecks, cost and energy consumption**

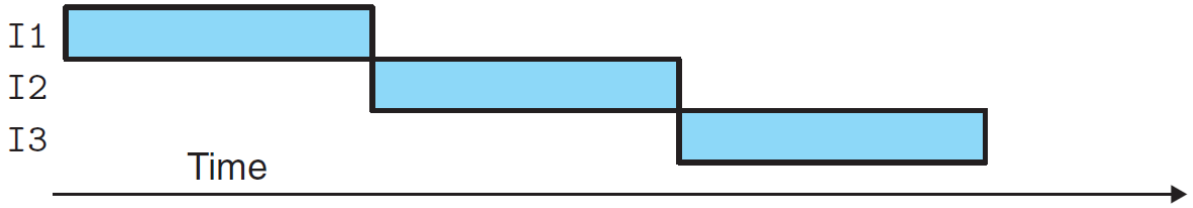
Factors that impact on the performance

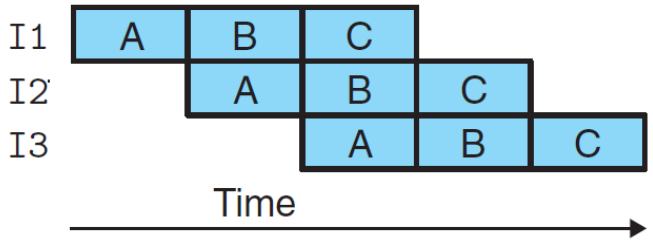
- Factors at **Architecture** Level
  - Hw + Sw
- Factors at **Operating System** Level
  - What is the Operating System (OS)?
    - Details will be addressed at the end of the course



Execution Flow Issues

- Architecture level
  - **Pipelining** principle
    - An instruction execution is divided into multiple stages
      - Modern processors present large number of stages (>=15 stages)





    - Limitations
      - Non-uniform partitioning
        - Throughput is limited to the longest pipeline stage
      - Inherent overhead
        - Communication between two stages (pipeline registers)

Execution Flow Issues

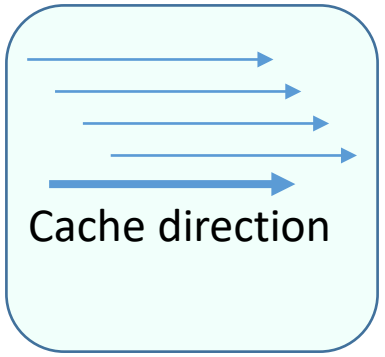
- Pipeline hazards
  - Dependencies can lead to wrong execution
    - Data dependencies: an operand depends on the result of previous calculation
    - Control dependencies: the next instruction to be executed depends on the result of a previous comparison
  - Static approach (by compiler)
    - Introducing delay
      - nop instructions
  - Dynamic approach (by the processor)
    - Introducing stalls (a.k.a. bubbles)
      - The processor holds back one or more instructions in the pipeline until the Hazard is solved

Execution Flow Issues

- Hardware contention in shared resources
  - E.g.: a single floating point unit shared among threads in a single core
- Multiprogramming level (several programs alive at a time)
  - Operating System schedules what programs can use the CPU and what programs have to be stalled
    - Addressed at the end of the course
- Dependencies in parallel software
  - Stalls among threads
    - Addressed in next courses
- Impact of cache misses in the memory hierarchy
  - Miss penalty: additional time due to a miss in the k-Level of cache
  - Types of cache misses
    - **Cold (compulsory) miss, Conflict miss, Capacity miss**

Example cache performance impact (1/6)

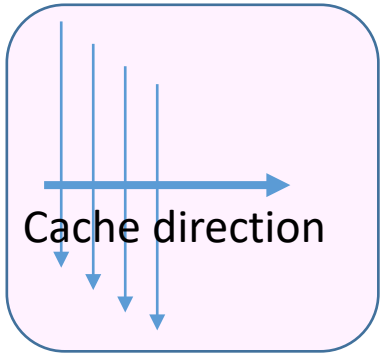
- ACCESS(i,j)



Row-wise access

- Locality of access

ACCESS(j,i)

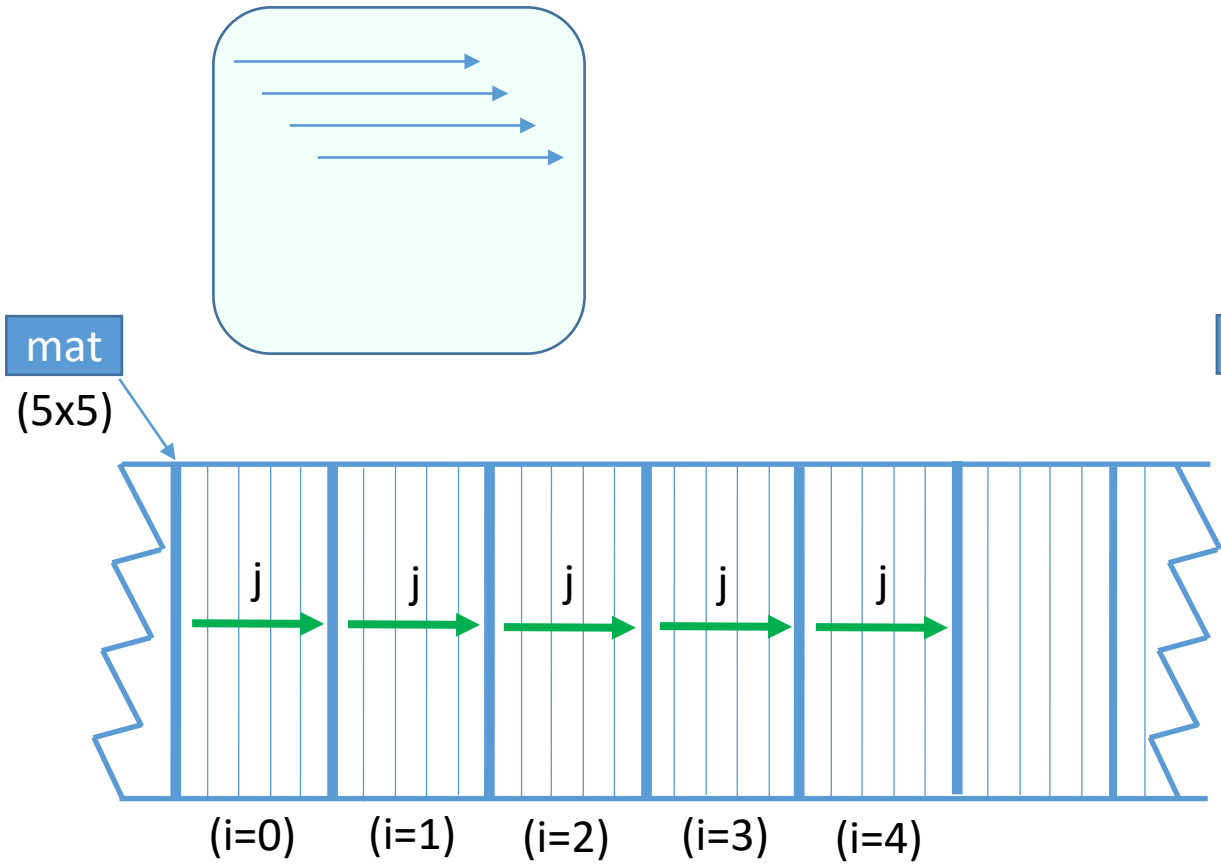


Column-wise access
Matrix transposed

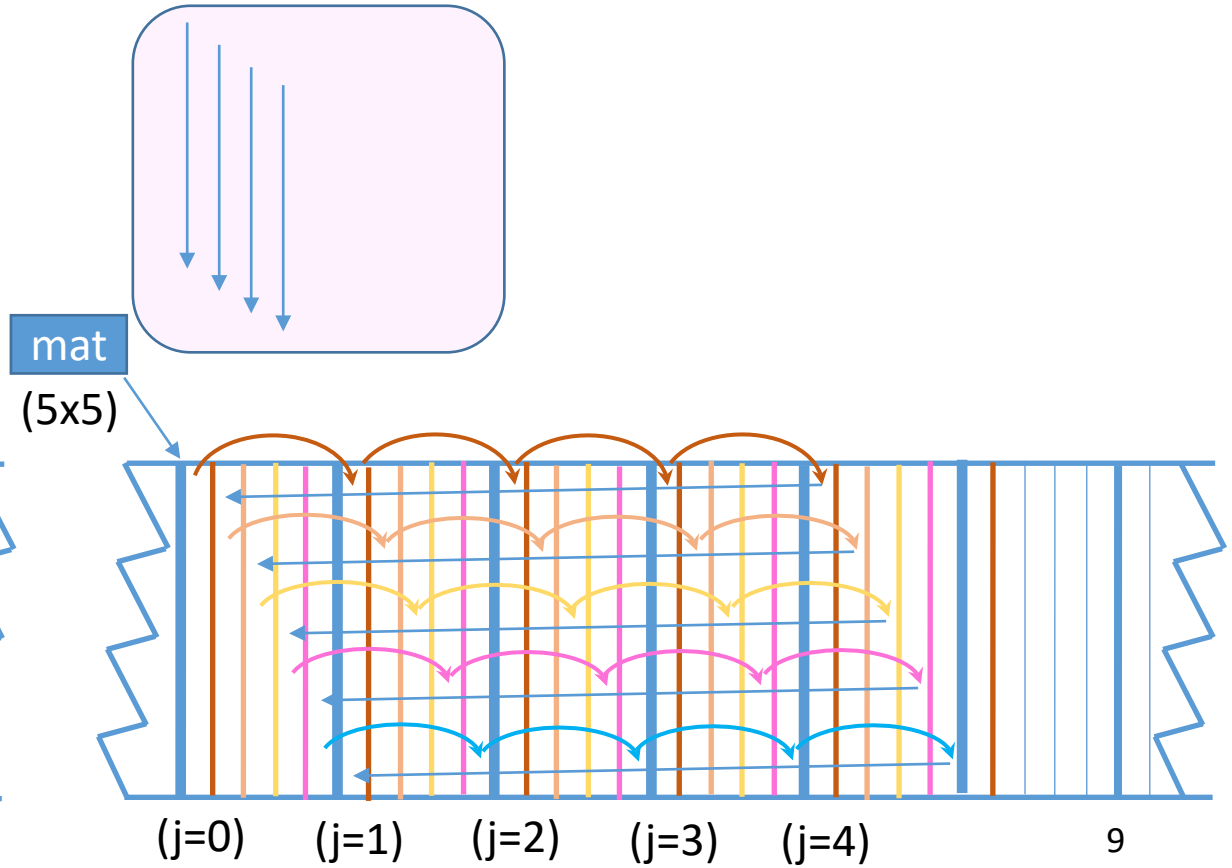
- Accesses are disjoint between each other – low locality

Example cache performance impact (2/6)

- ACCESS(i,j)

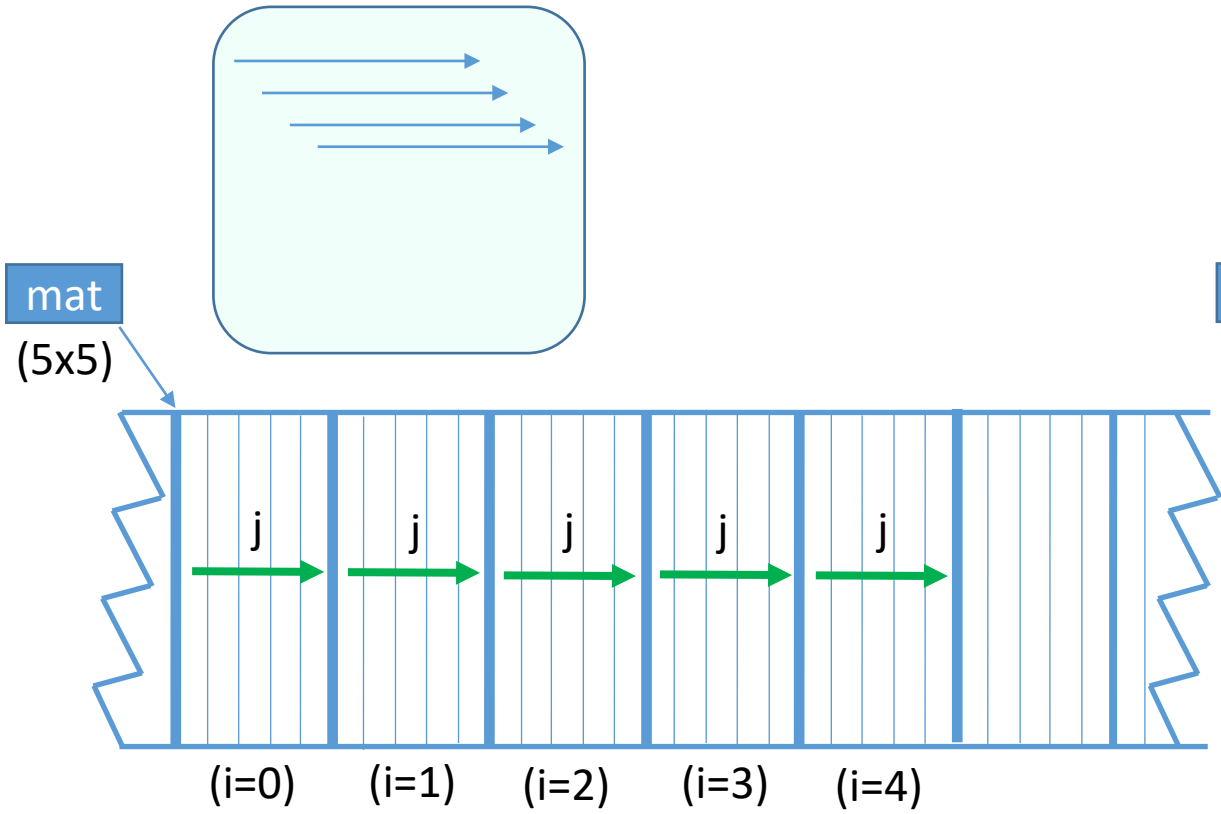


ACCESS(j,i)



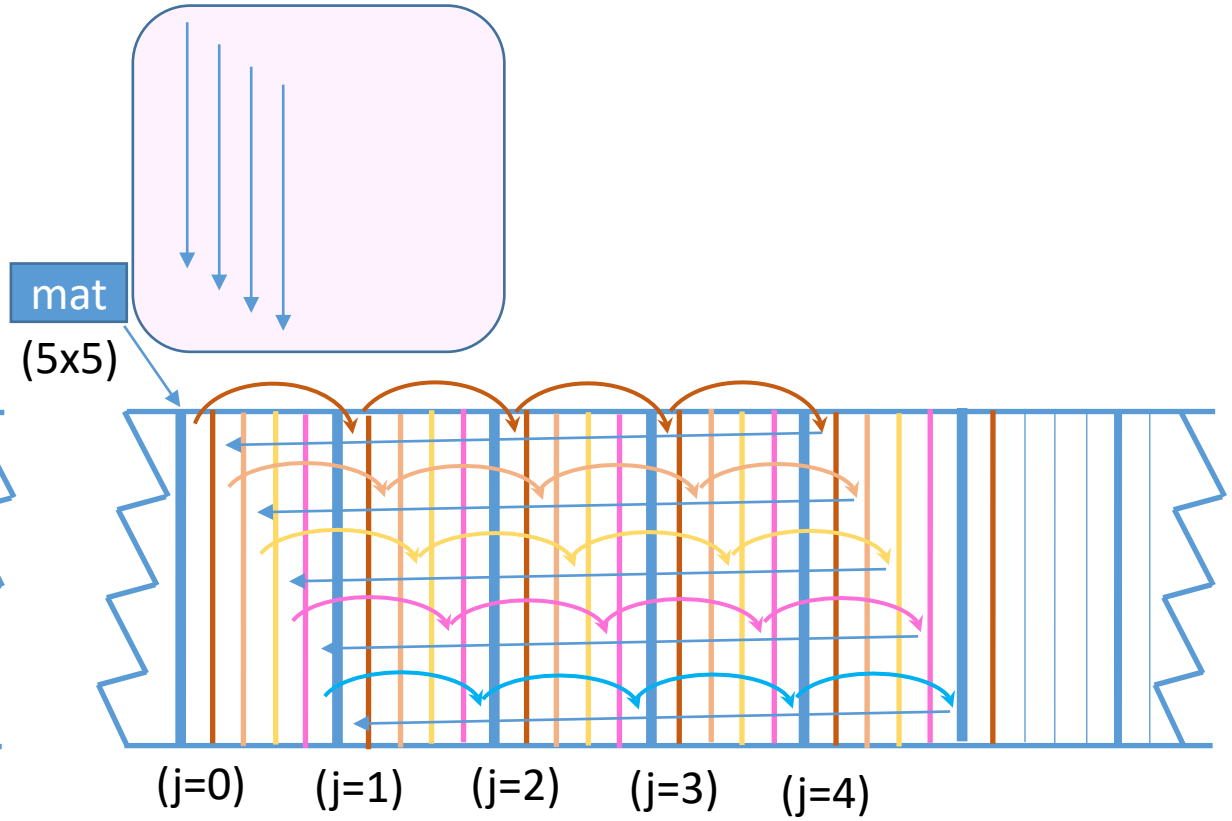
Example cache performance impact (3/6)

- ACCESS(i,j)



~5 misses

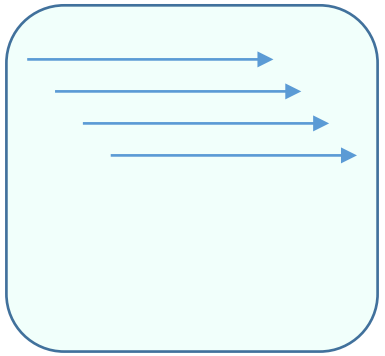
ACCESS(j,i)



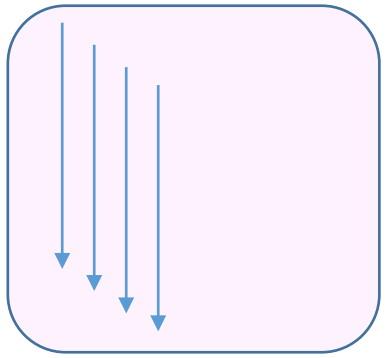
>5 misses (capacity)

Example cache performance impact (4/6)

- ACCESS(i,j)



ACCESS(j,i)



- Matrix 10240 x 10240, single precision floating point values
- MN4 processor, Intel(R) Xeon(R) Platinum 8160 CPU @ 2.10GHz
- Initialization times
  - loop =0: 148 ms
  - loop >0: 51ms
- Cache friendly
  - Spatial and temporal locality

loop =0: 465 ms

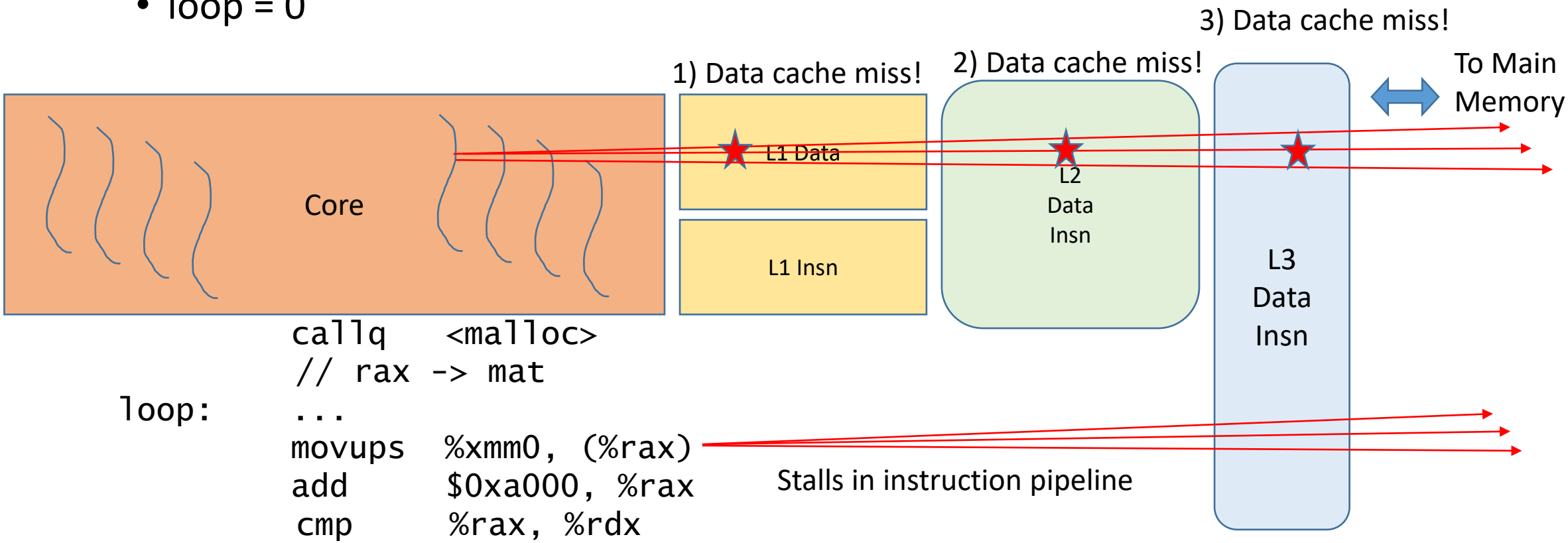
loop >0: 361ms

Cache unfriendly

~Temporal locality, no spatial locality

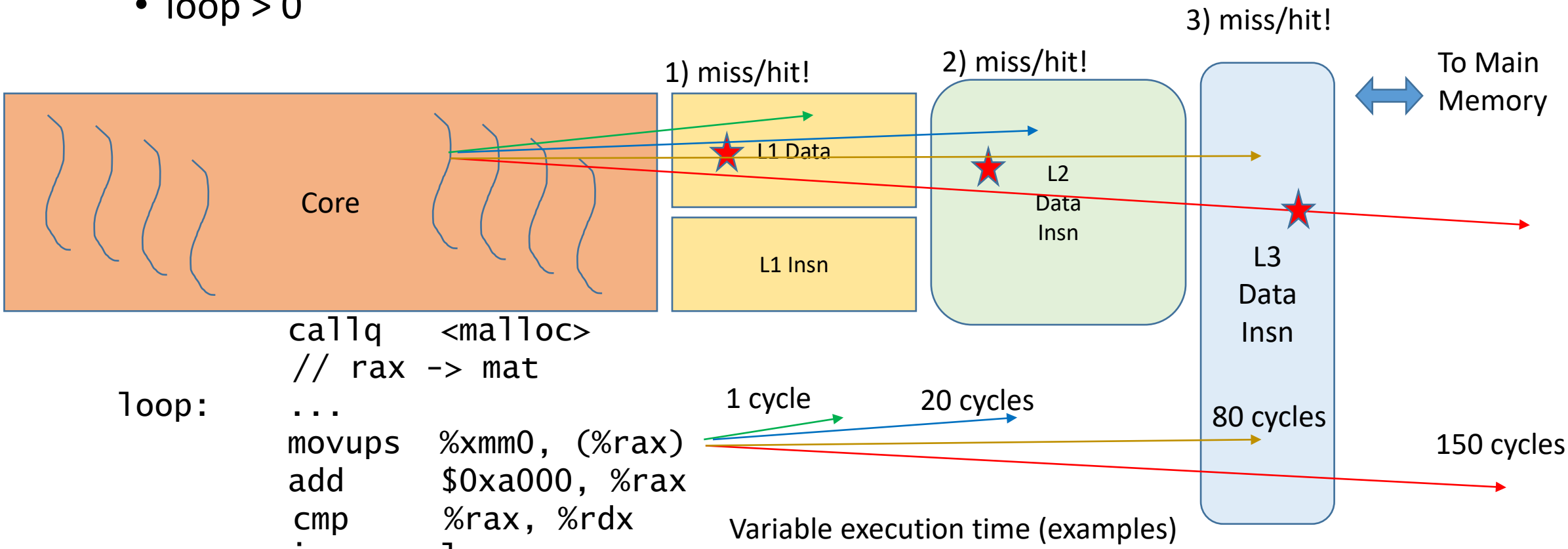
Example cache performance impact (5/6)

- Cache accesses
  - loop = 0



Example cache performance impact (6/6)

- Cache accesses
  - loop > 0



Performance Metrics

- Architectural based
  - CPI: cycles per instruction
    - CPI = $(C_i+C_b) / C_i = 1 + C_b/C_i$ → CPI = 1 + penalties
  - IPC: instructions per cycle
    - IPC = #instructions/cycle = 1/CPI

Performance Metrics

- Execution time (s)
  - Actual wall-clock time
  - Affected by hardware events, all of them count as time spent in the app
    - Requesting services to the OS (Interrupts, Exceptions, System calls)
  - ... and OS events
    - **Multiprogramming** level
- CPU time (s)
  - Amount of time spent running on a CPU (hw thread), in user and/or system mode
- Speed-up (no units)
  - Relation between the serial execution time and the parallel execution time
  - Usually applied to wall-clock time

Performance Metrics

- Bandwidth (bytes/s)
  - Relation between the amount of data transmitted and the time invested
- Latency (s)
  - Amount of time to start operations or communications
- Throughput (elements/s)
  - Maximum amount of operations, applications, units per second
  - Maximum rate of production / processing / consumption
- Power consumption (W)
  - Work done per unit of time
  - It depends on the complexity of the instruction execution
    - Simple integer ALU, Floating-point, branches, …
- And many many more...

Statistics

- Average
  - Provides a more stable and realistic measurement
  - Sum of samples, divided by the number of samples
  - Also possible:
    - Harmonic mean (average of rates)
    - Geometric mean (when comparing different items, with different numeric ranges)
- Standard deviation
  - Indicates how much variability there is among the results
  - $s = \sqrt{\frac{\sum_{i=0}^{N-1} (x_i - \bar{x})^2}{N-1}}$

How to use the statistics

- Execution time
  - Average* of the N results obtained
- Speed-up
  - Average sequential execution time over average parallel execution time
- Bandwidth
  - Average* of the bandwidth obtained in N experiments
- Latency
  - Average* of the latencies obtained in N experiments

* Average can be changed by standard deviation

Bibliography

- Computer Systems – A Programmer's perspective (3rd Edition)
  - Randal E. Bryant, David R. O'Hallaron, Person Education Limited, 2016
  - https://discovery.upc.edu/permalink/34CSUC_UPC/11q3oqt/alma991004062589706711
    - Several Chapters

- Computer Organization and Design (5th Edition)
  - D. Patterson, J. Hennessy, and P. Alexander
  - http://cataleg.upc.edu/record=b1431482~S1*cat
    - Several Chapters