Introduction to Parallel Processing

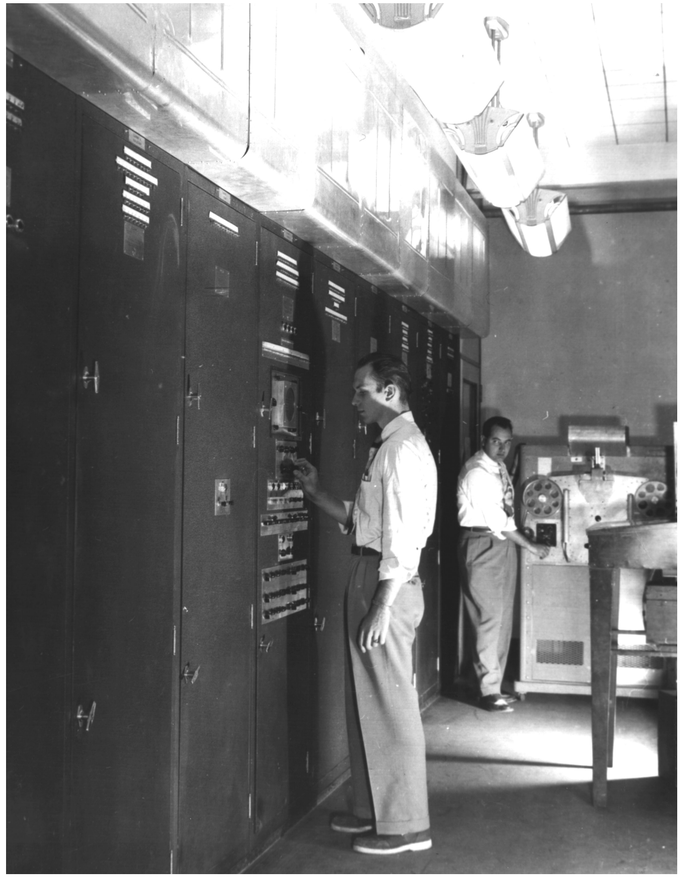
Lecture 2 : Serial Performance

08/29/2022

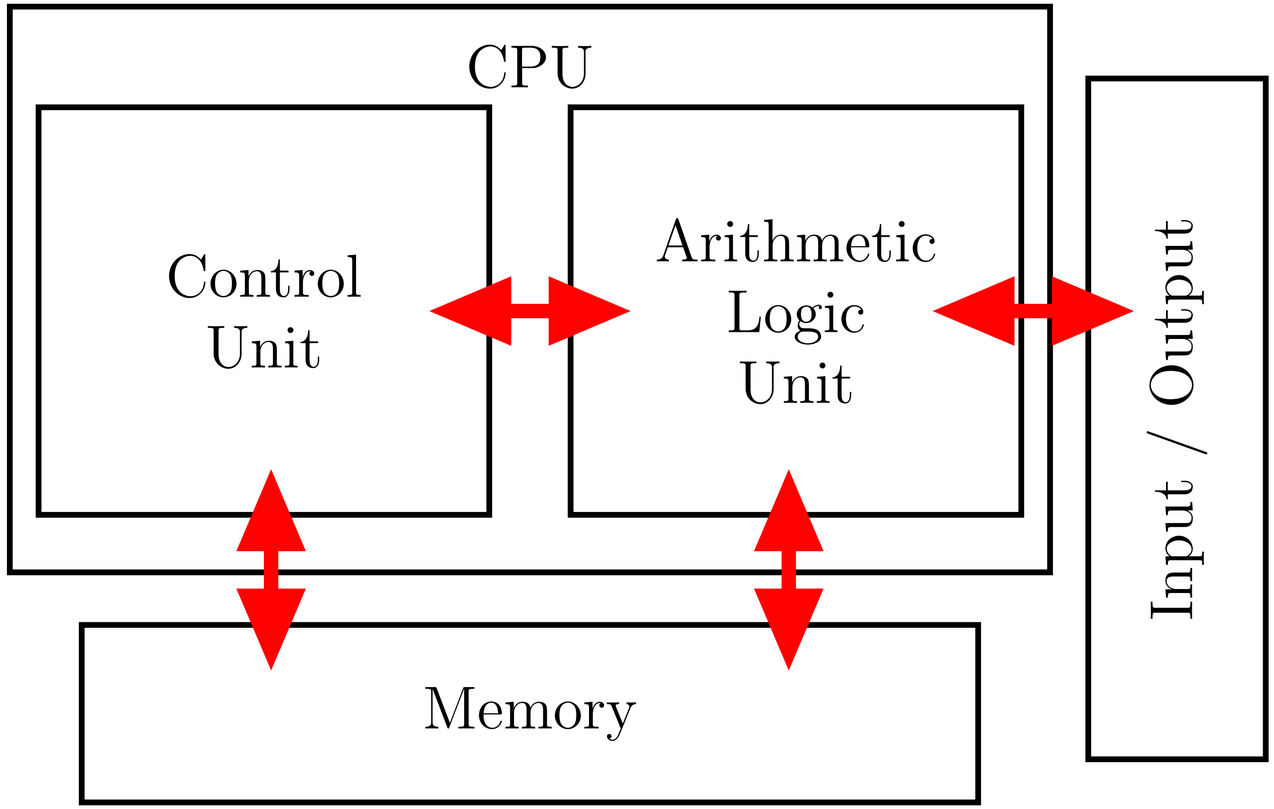
Professor Amanda Bienz

Serial Computer Architecture

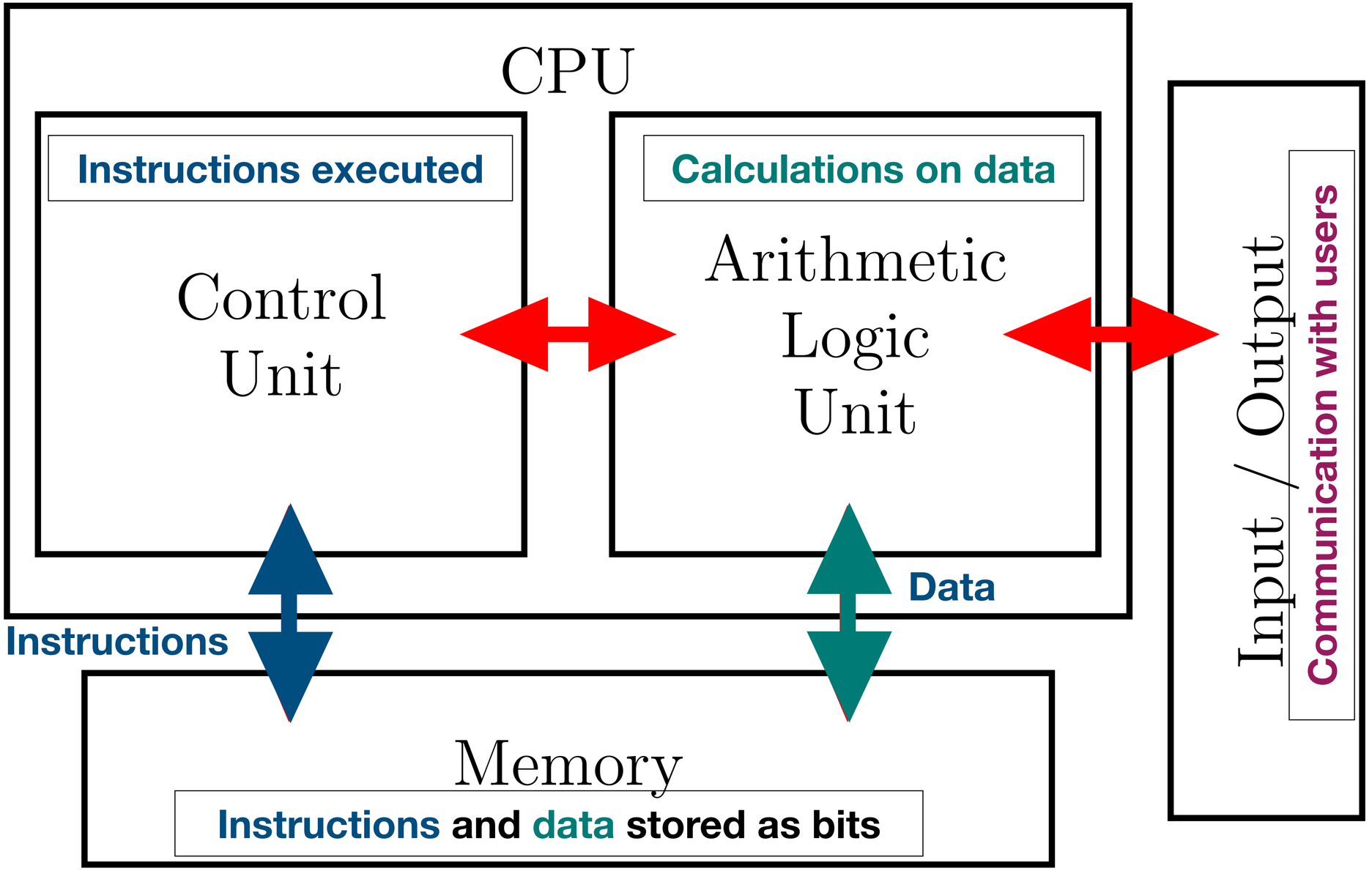
EDVAC : First Binary Computer



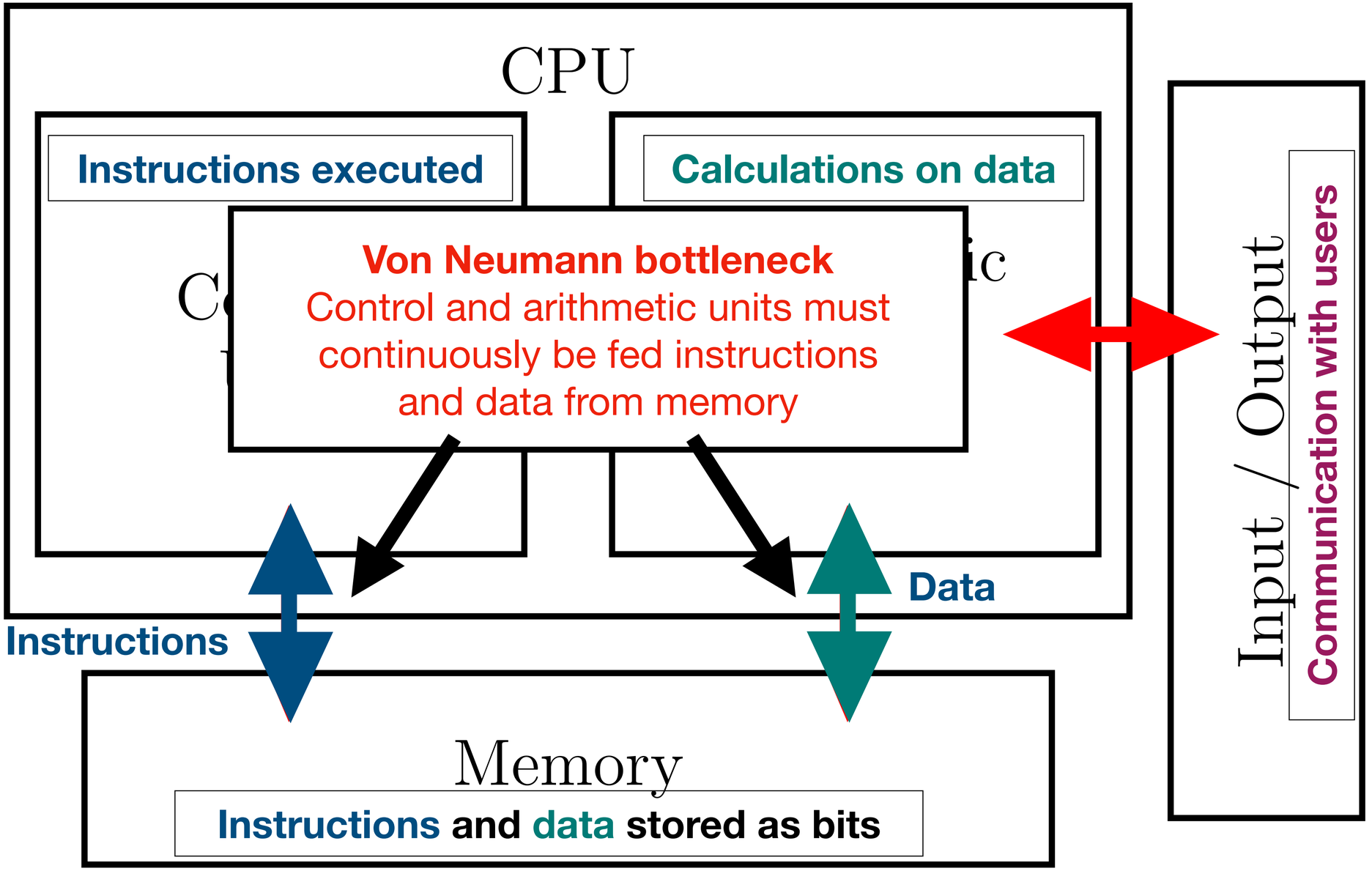
http://ftp.arl.mil/ftp/historic-computers/



Serial Computer Architecture

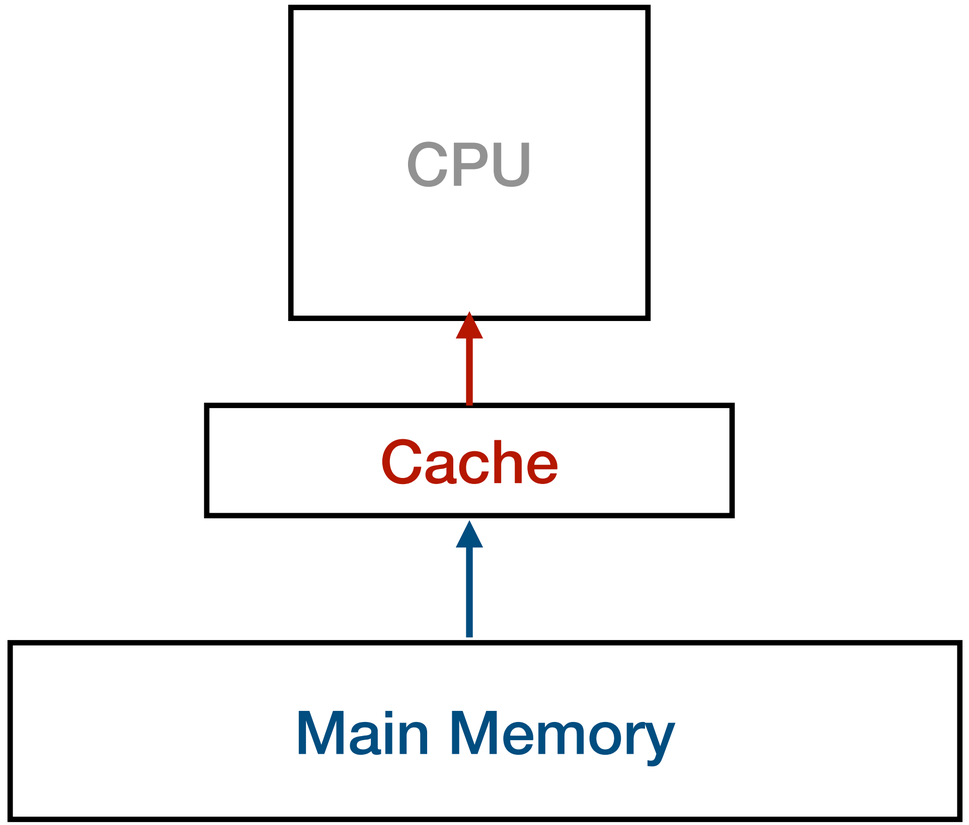


Serial Computer Architecture



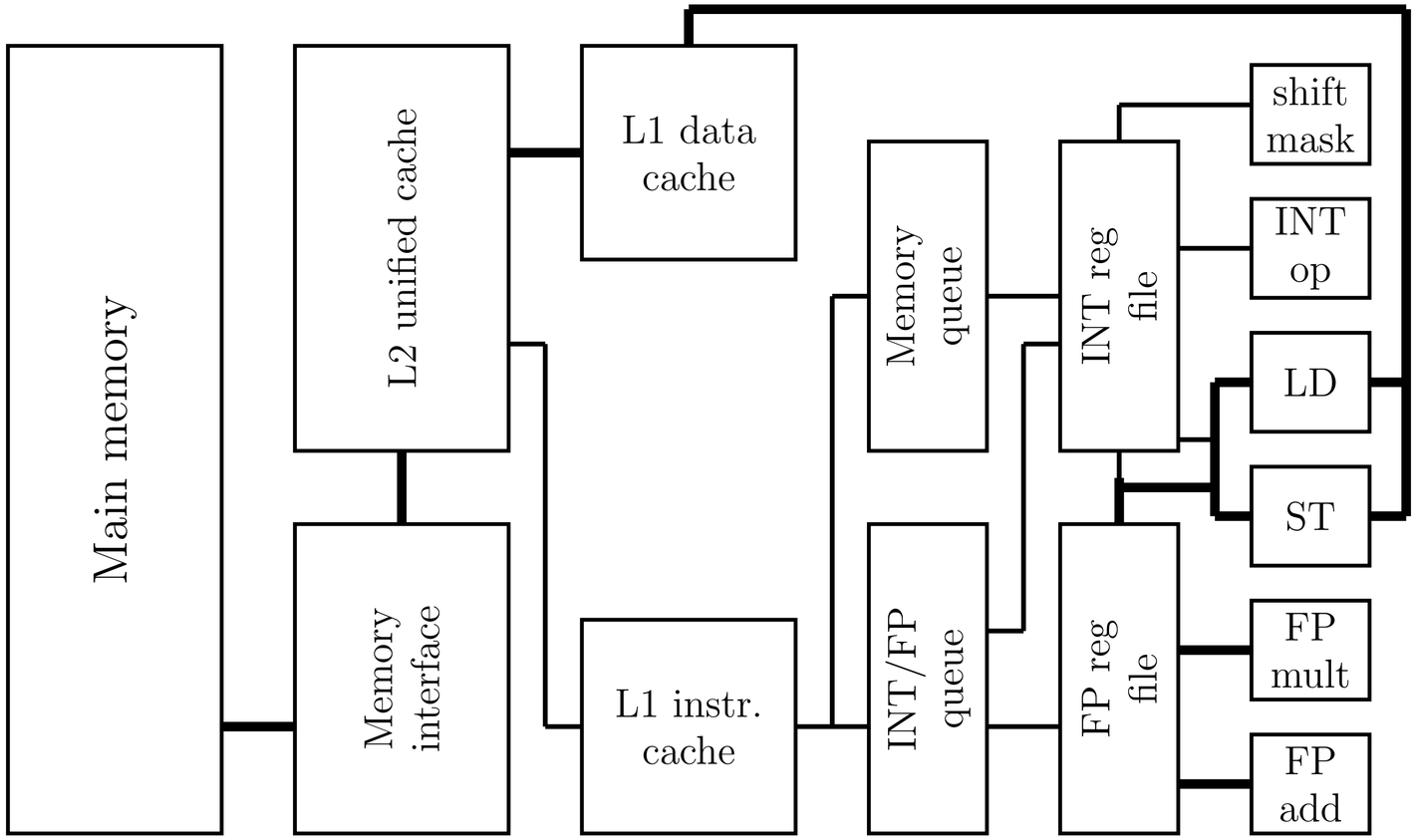
One Fix : Cache

- Main memory:
  - Large
  - Far from CPU
  - Slow



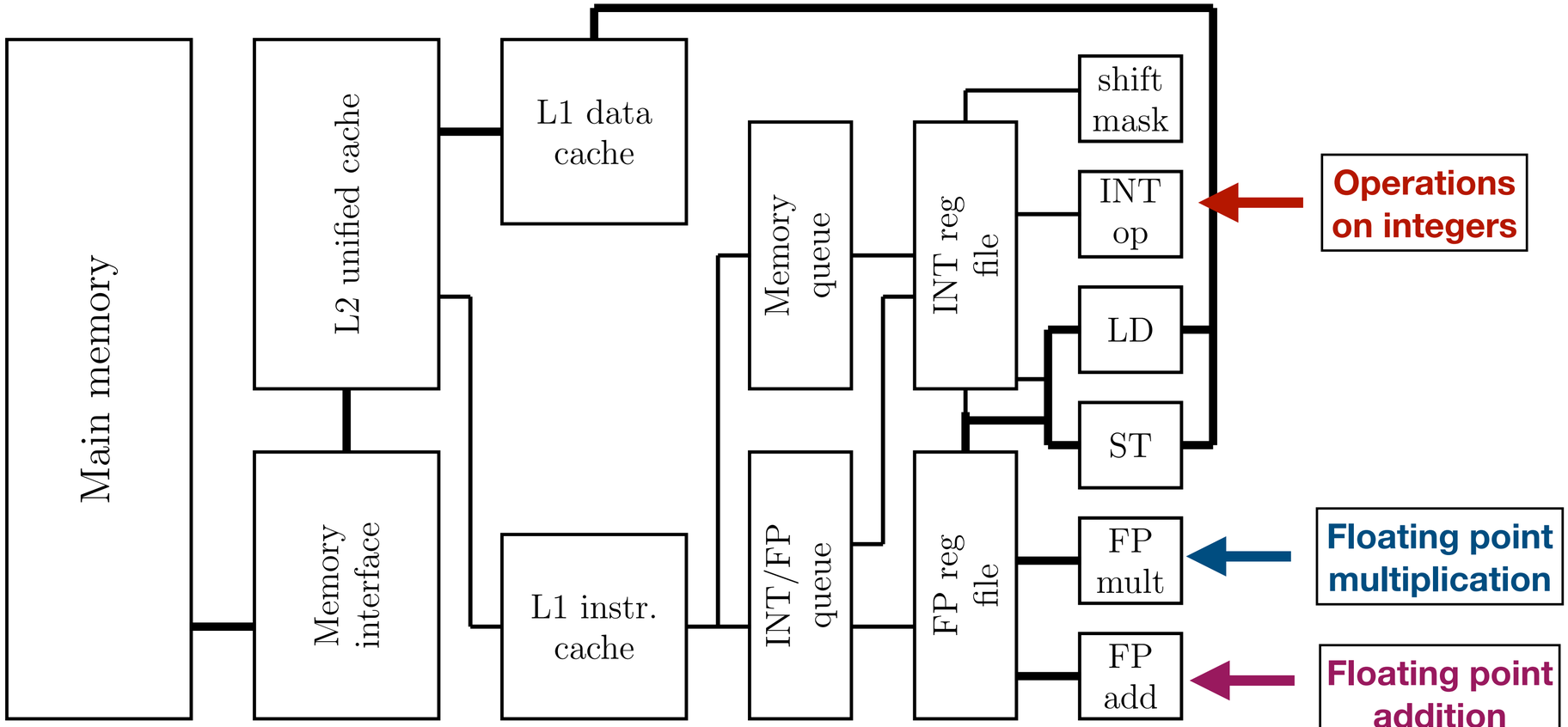
- Cache:
  - Smaller
  - Closer to CPU
  - Faster

Cache-Based Microprocessor



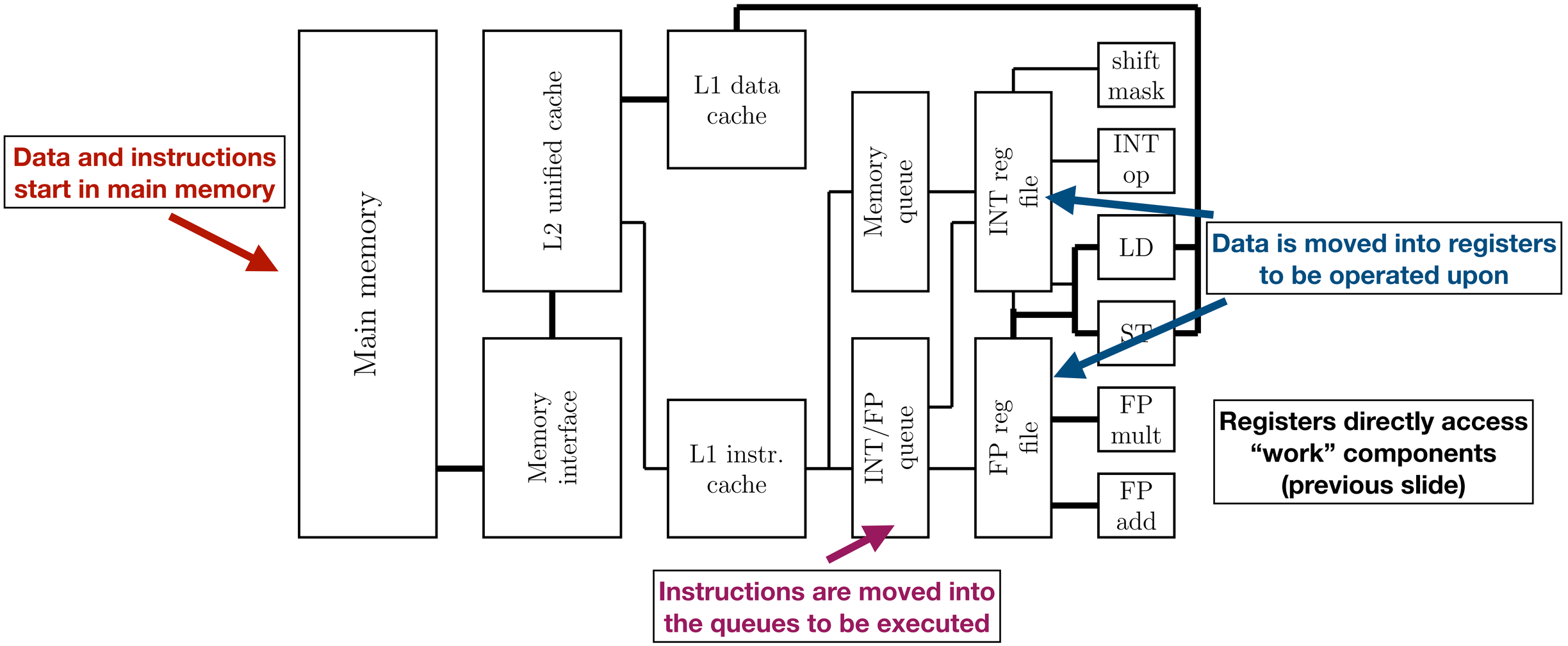
Cache-Based Microprocessor

Three little components perform all operations

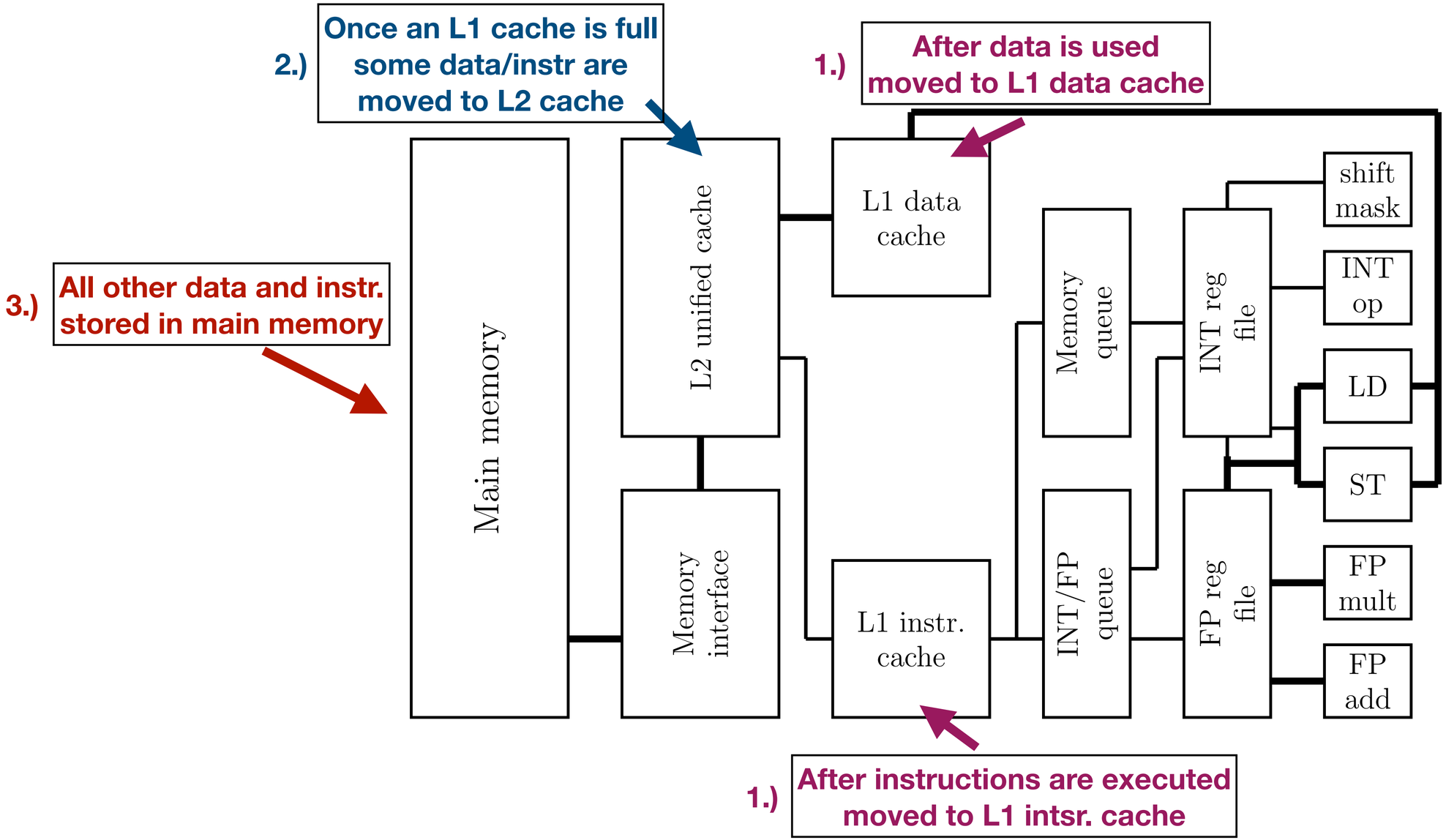


Everything else used to efficiently move data and instructions

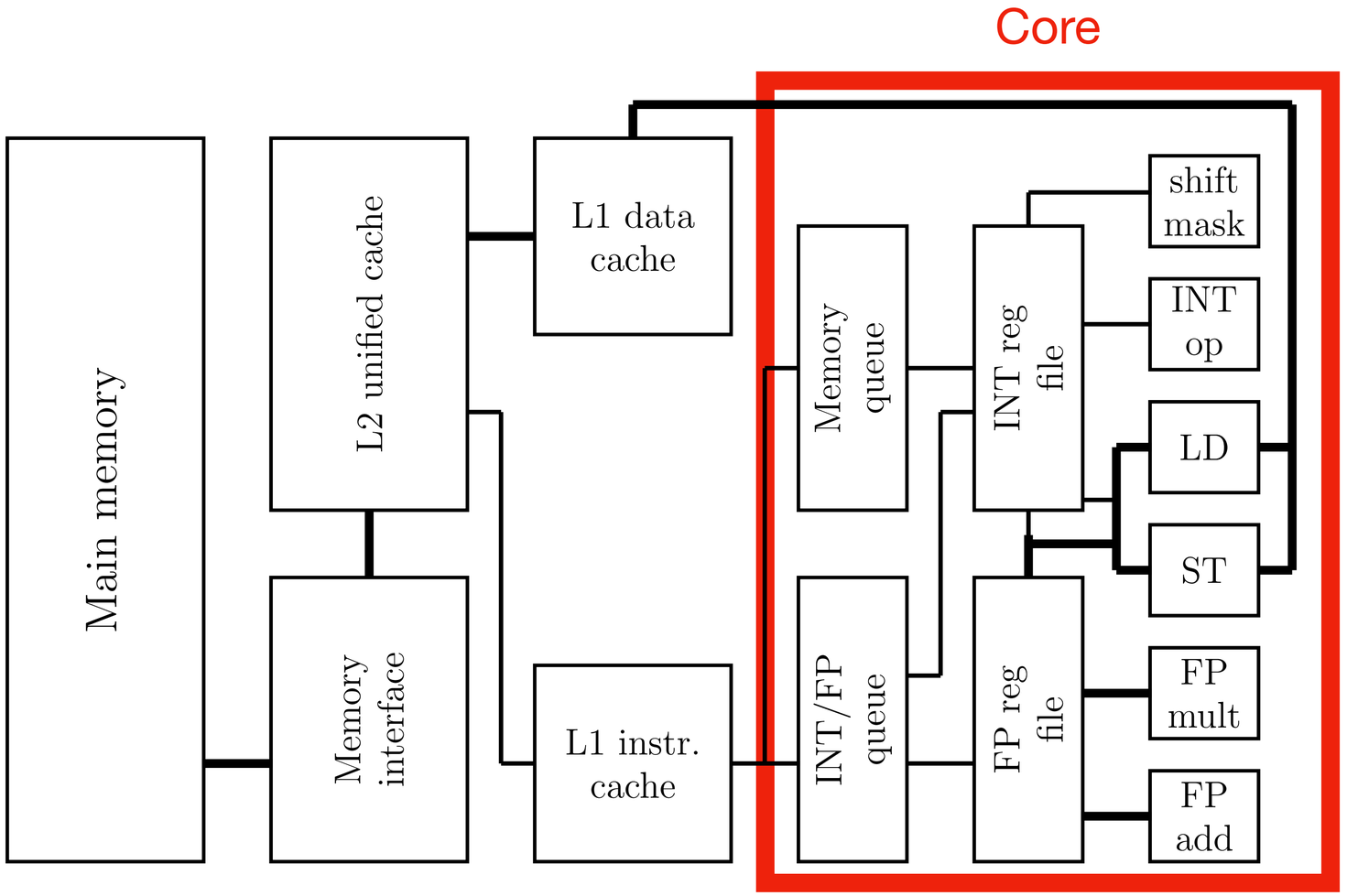
Cache-Based Microprocessor



Cache-Based Microprocessor



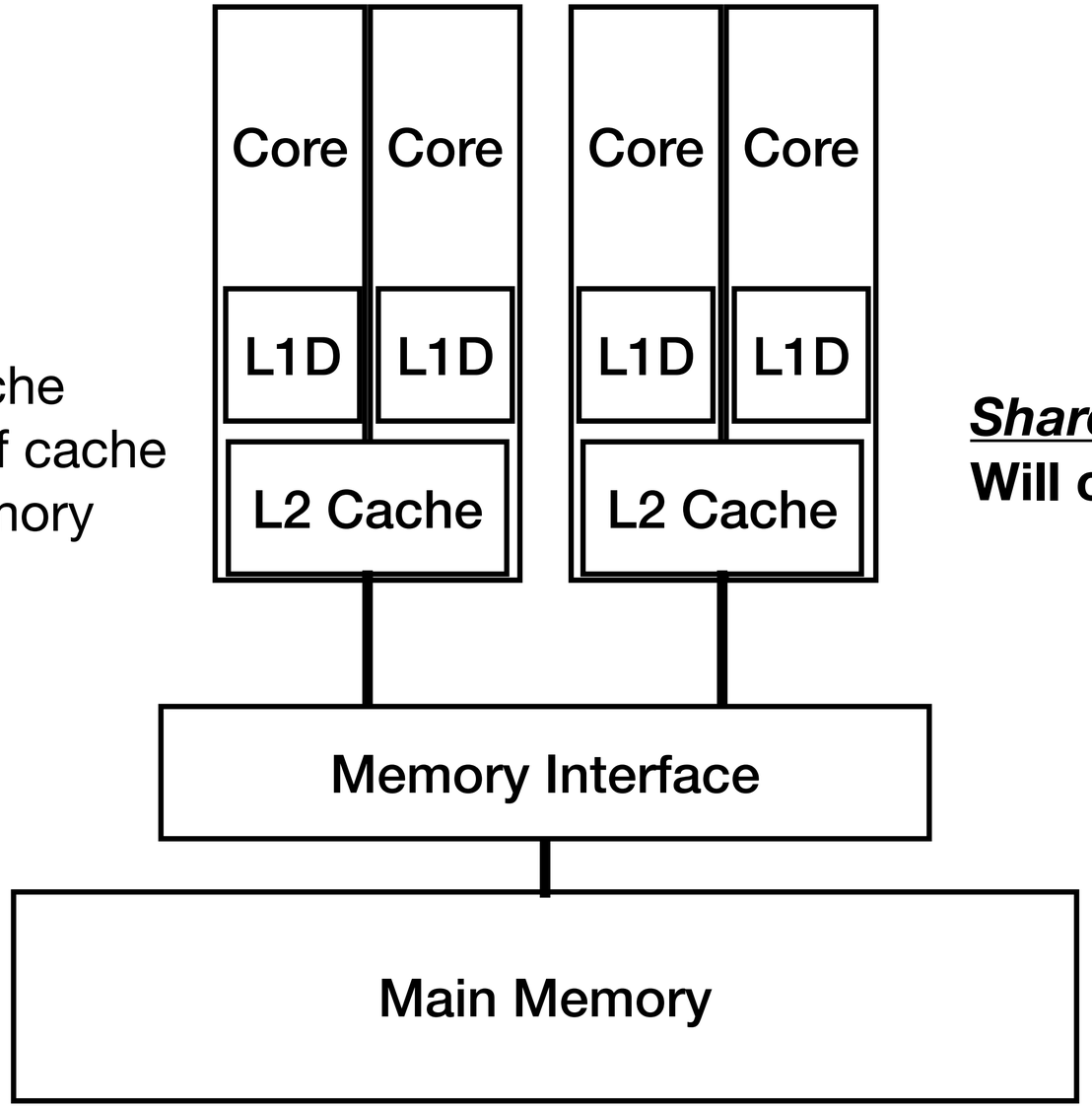
Cache-Based Microprocessor



Multicore processors have multiple cores that share main memory, and maybe a level of cache

Multicore Processor

- Cores may have own cache
- May share some levels of cache
- All cores share main memory



Shared Memory Programming

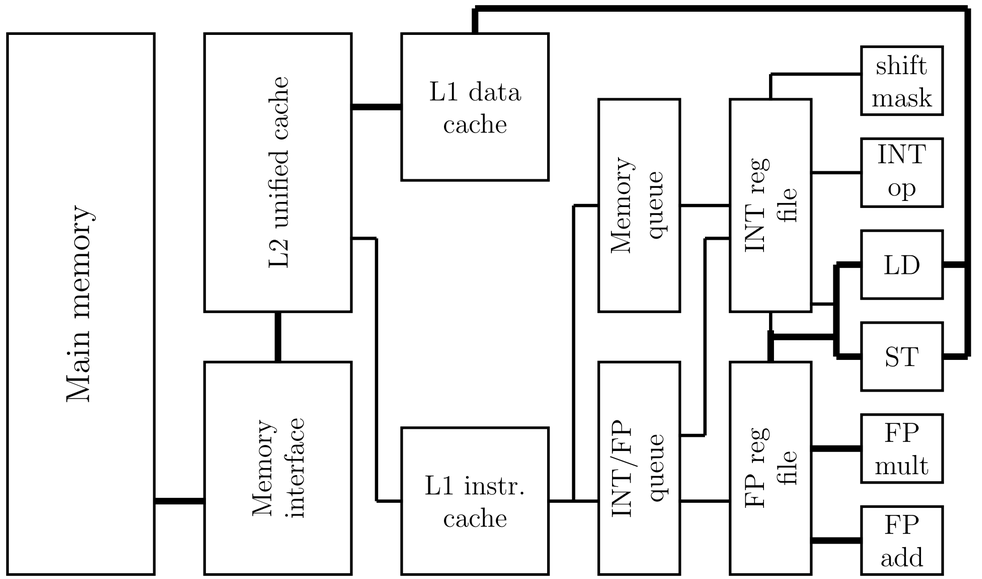
Will come back to this next week

Superscalar Processor

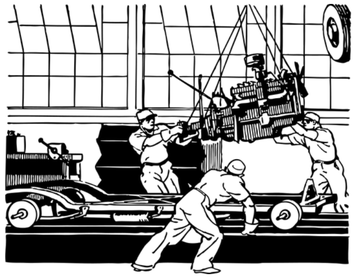
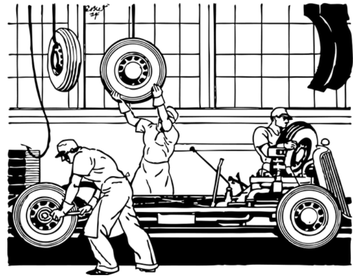
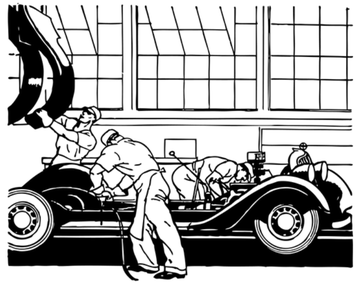
- Capable of executing more than one instruction per cycle
- Typical quality of today's computers
- Some details :
  - Multiple instructions fetched, decoded concurrently
  - Fast caches : > 1 load or store per cycle
  - Multiple floating-point pipelines run in parallel (explanation to follow)

Pipelining : Instruction-level parallelism

- Parallelism occurs in serial computers
- To complete an instruction:
  - Fetch data from memory
  - Decode instruction
  - Execute operation
- A program will have many, many operations.
- Can pipeline the fetch - decode - execute process

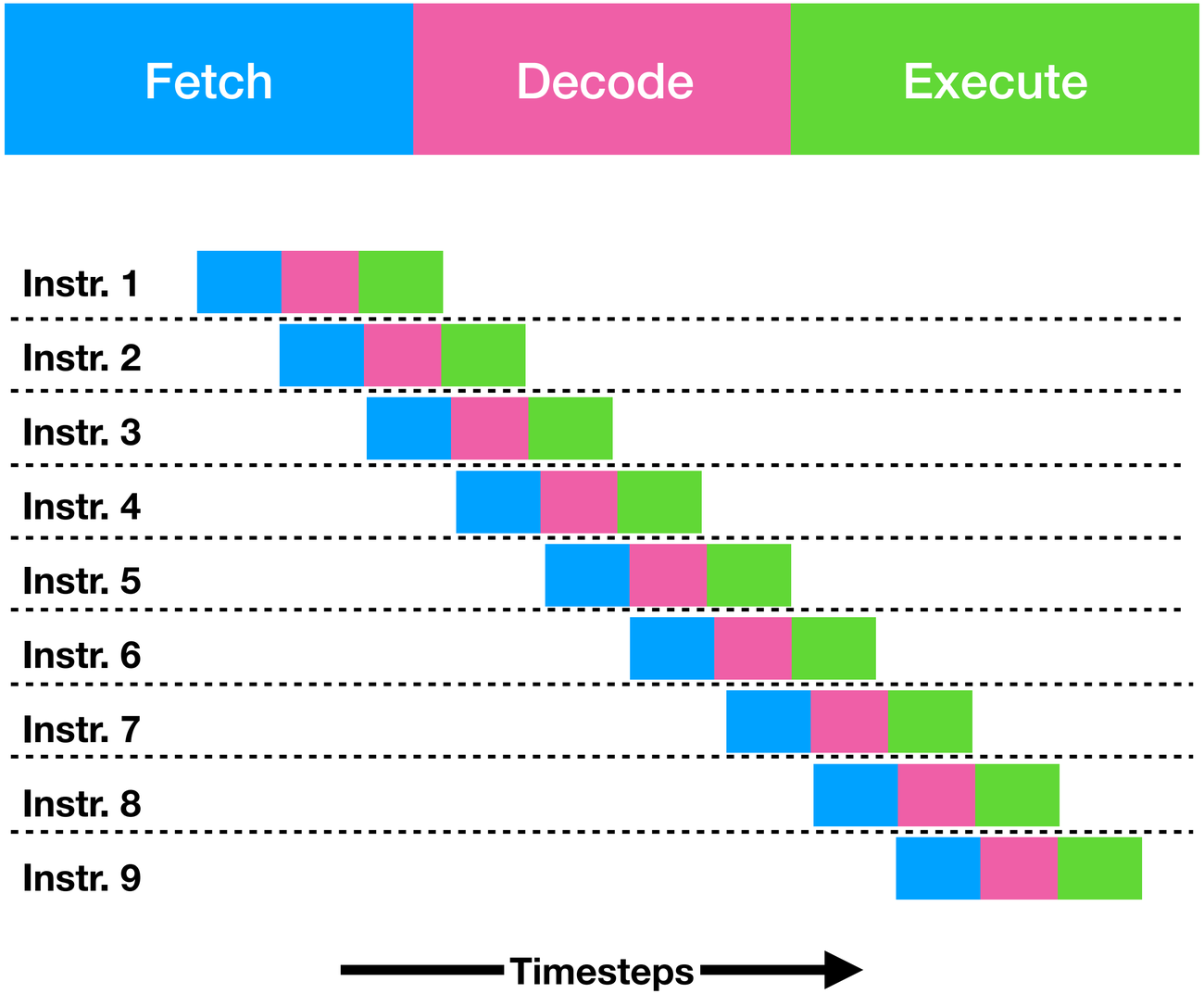


Pipelining

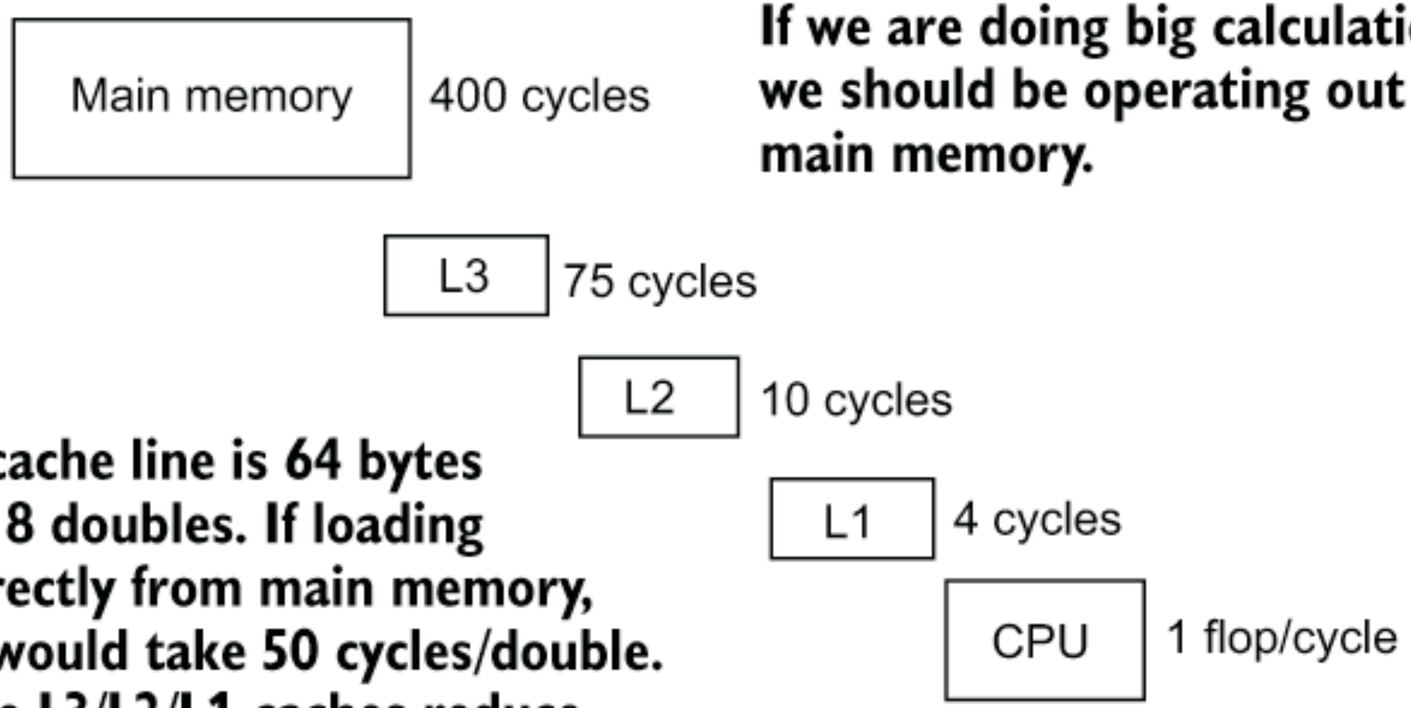




- Similar to assembly lines
- Workers only need to know about their specific task
- Each worker executes his/her task over and over on successive objects
- Each object is then moved to the next worker

Pipelining

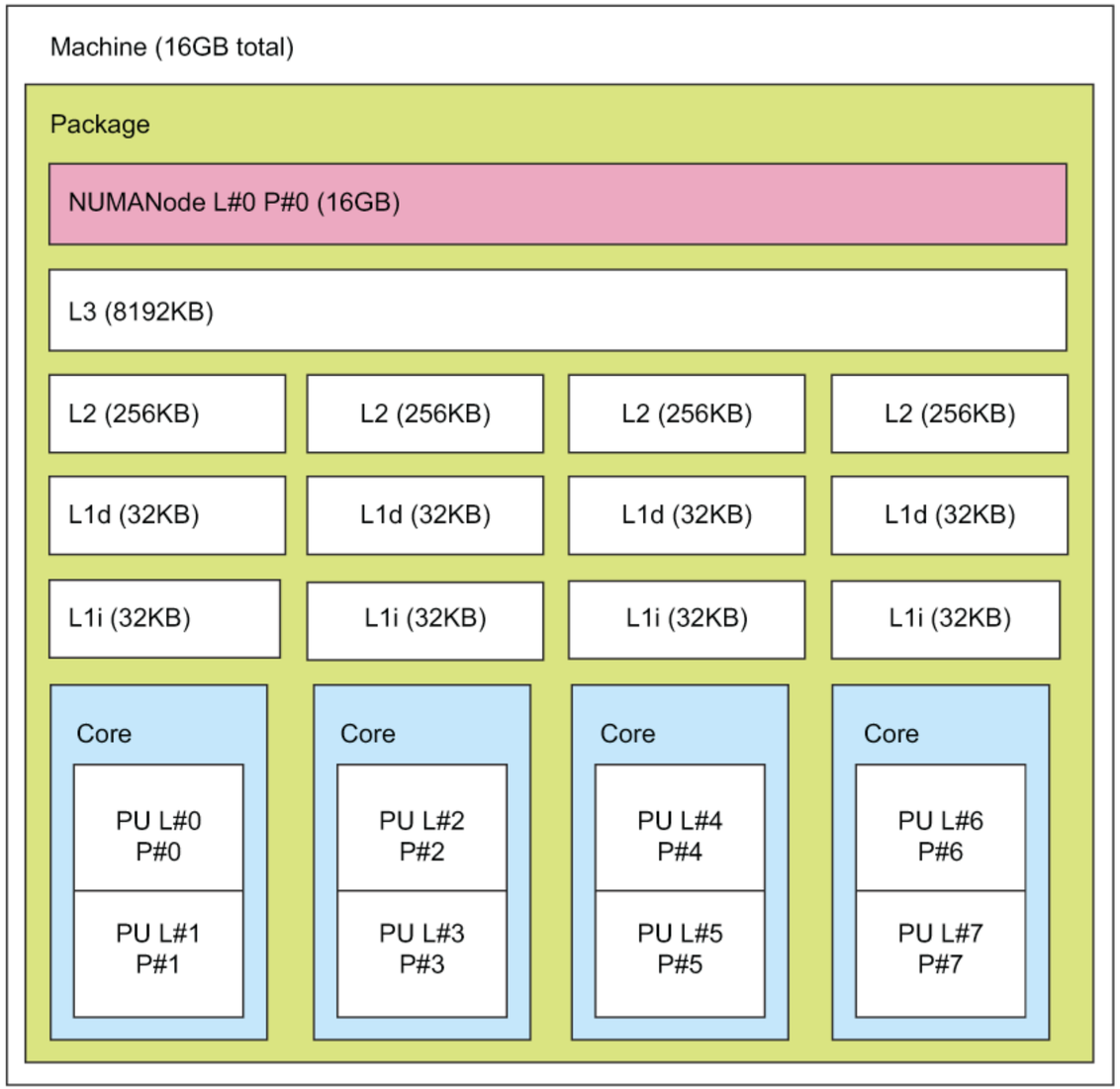

Fetch
Decode
Execute
Instr. 1
Instr. 2
Instr. 3
Instr. 4
Instr. 5
Instr. 6
Instr. 7
Instr. 8
Instr. 9
Timesteps

Memory Hierarchy

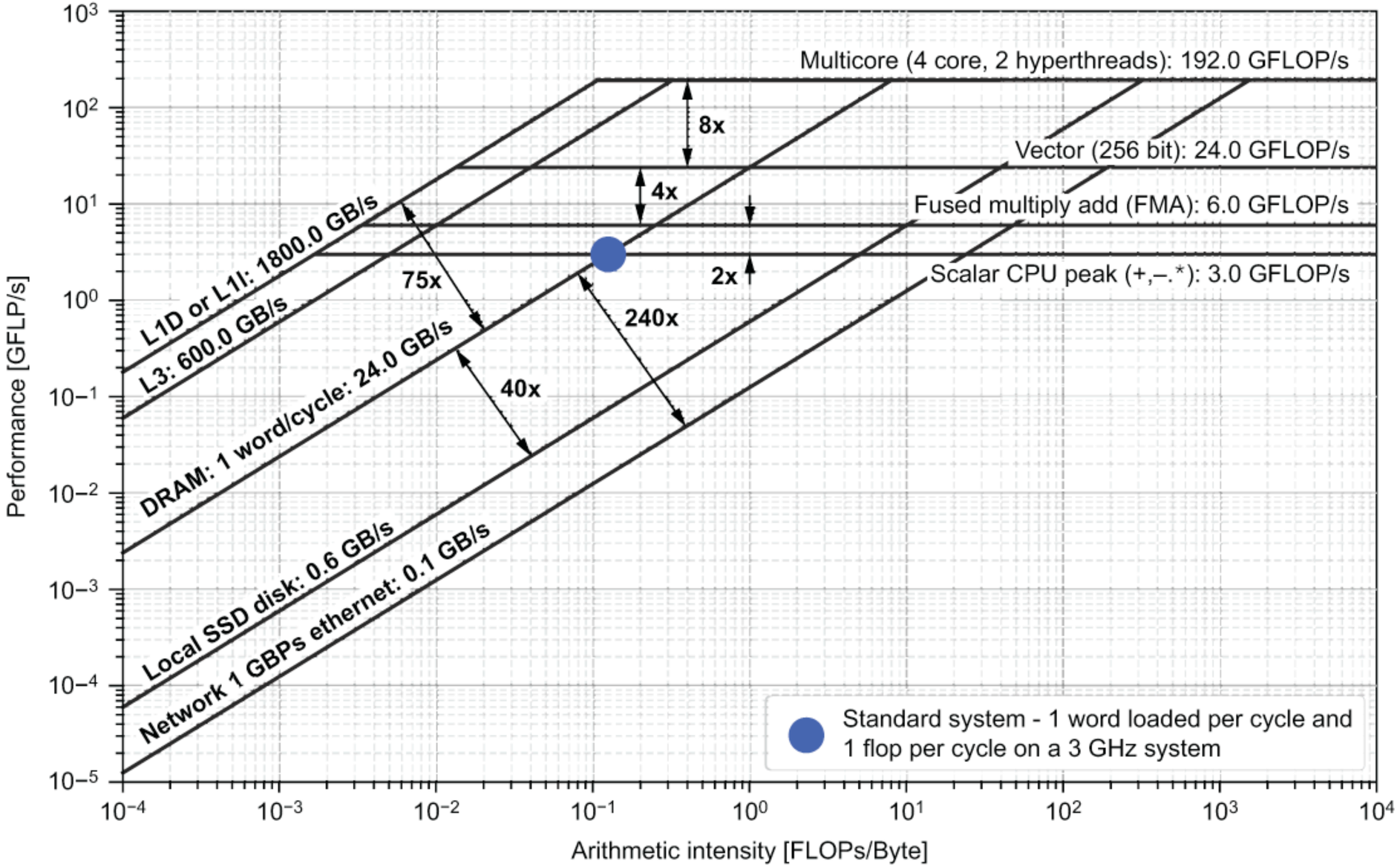


If we are doing big calculations, we should be operating out of main memory.

A cache line is 64 bytes or 8 doubles. If loading directly from main memory, it would take 50 cycles/double. The L3/L2/L1 caches reduce this by an order of magnitude.


Machine (16GB total)
Package
NUMANode L#0 P#0 (16GB)
L3 (8192KB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L1d (32KB)
L1d (32KB)
L1d (32KB)
L1d (32KB)
L1i (32KB)
L1i (32KB)
L1i (32KB)
L1i (32KB)
Core
PU L#0 P#0
PU L#1 P#1
Core
PU L#2 P#2
PU L#3 P#3
Core
PU L#4 P#4
PU L#5 P#5
Core
PU L#6 P#6
PU L#7 P#7

Roofline Models



If you want to learn more…

- “Introduction to High Performance Computing for Scientists and Engineers” by Georg Hager and Gerhard Wellein
- We will talk about caches more in depth next lecture