CS3210 Tutorial 1

Tutorial: Parallel Computer Architecture

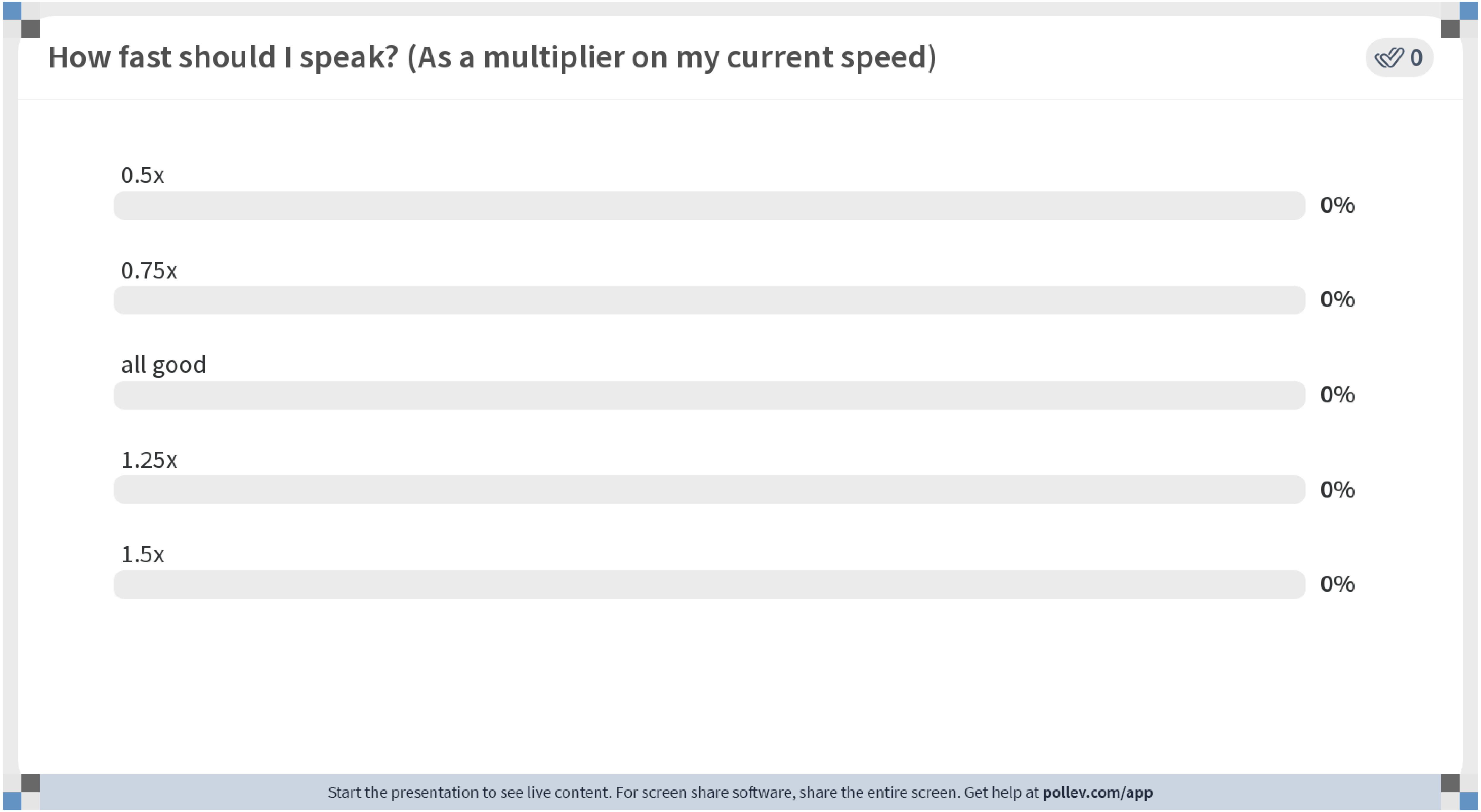
Lab: Slurm

Please don't start on part 2/3 of the tutorial yet.

Theodore's slides; heavily inspired by Sriram's, Richard's, Cristina's and Zhi Heng's slides.

Calibration / Improvements

After first tutorial :)



How fast should I speak? (As a multiplier on my current speed)

0.5x	0%
0.75x	0%
all good	0%
1.25x	0%
1.5x	0%

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Quick Recap

Lecture 2+3, Tutorial 1, Lab 1

So Far: High-Level

Processes and Threads **(Lec 2)**	Processes and Threads **(Lab 1)**

- Process basics and states
- Memory regions
- fork/wait
- IPC (shared memory, message passing)
- Threads as lightweight processes
- Users vs kernel threads
- Synchronization: pthread mutexes, semaphores, condition variables

Parallel Computing Architectures **(Lec 3)**	Parallel Computing Architectures **(Tut 1)**

- Forms of parallelism (bit/instruction/thread/processor)
- Flynn's Taxonomy (SISD/SIMD/...)
- Hierarchical vs Pipelined designs
- How memory / cache is organized (distributed/shared/hybrid)

Focus: Forms of Parallelism

- Bit-level
- Instruction-level
- Thread-level

Single processor

- Processor-level
  - Shared memory
  - Distributed memory

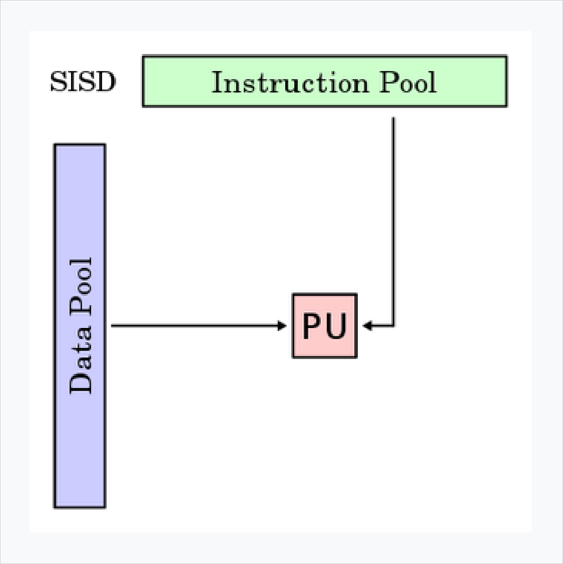
Multiple processors

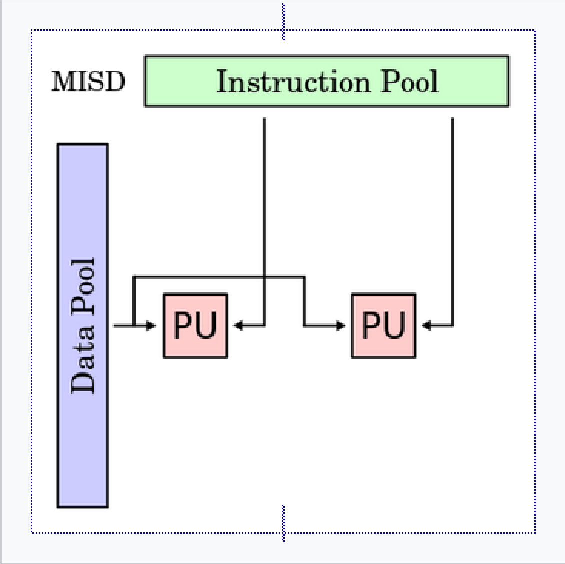
Focus: Flynn’s Taxonomy

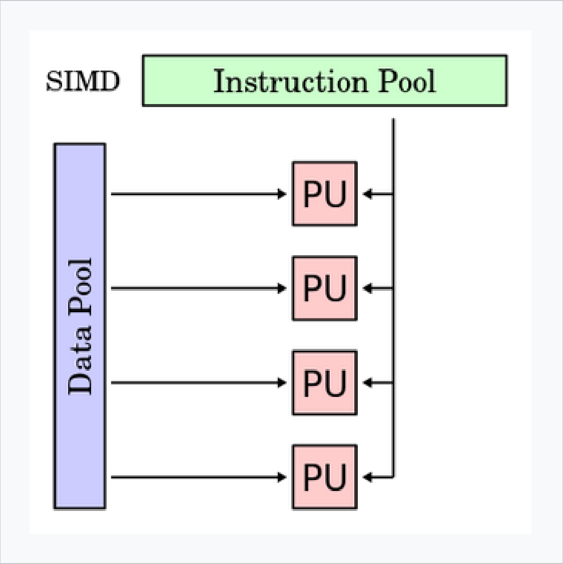
- Terminology to define different types of parallel architectures

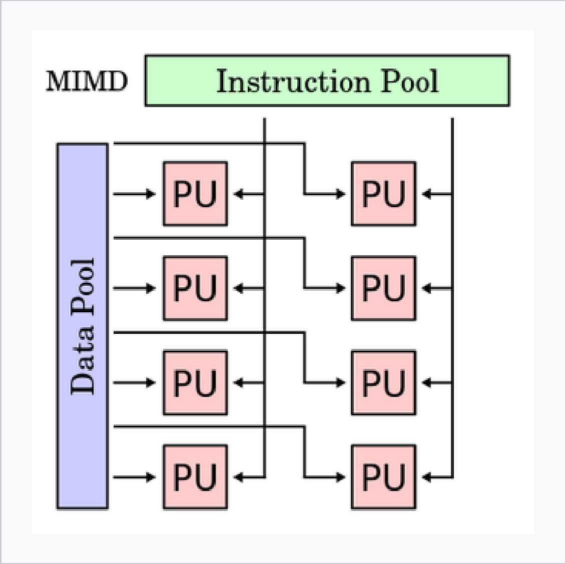
- Single / multi-**instruction**
  - How many **independent** streams of execution are there?

- Single / multi-**data**
  - How many **logical blocks of data** are we trying to operate on?





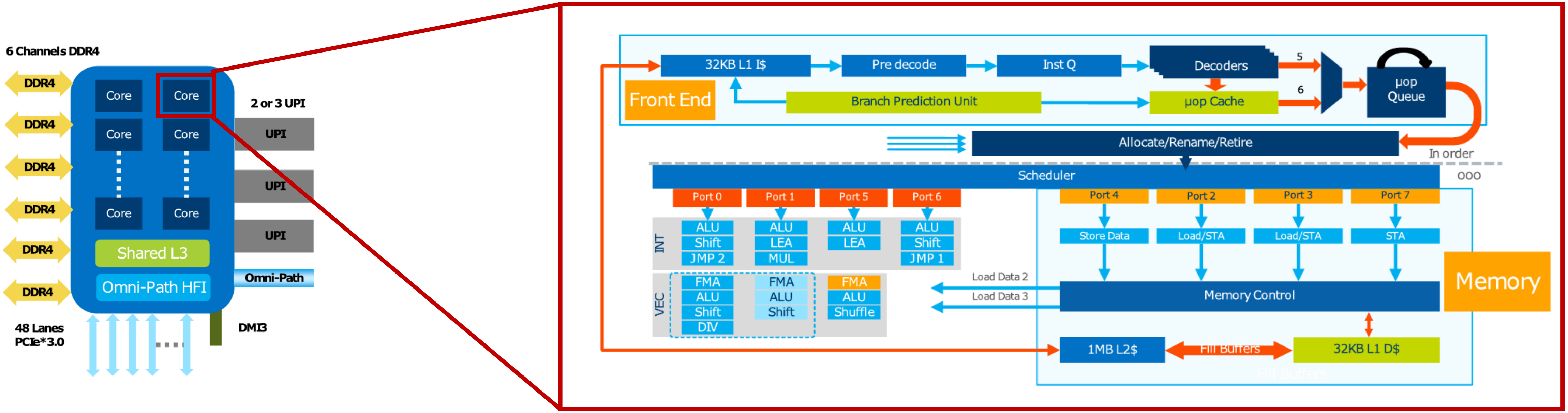




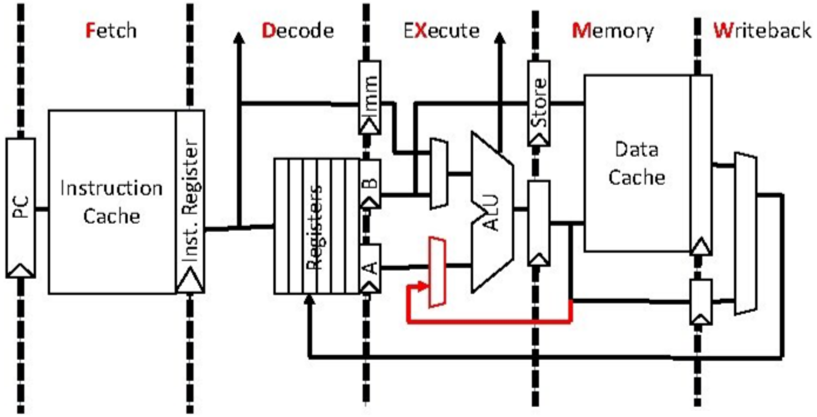
Part 1: Tutorial Questions

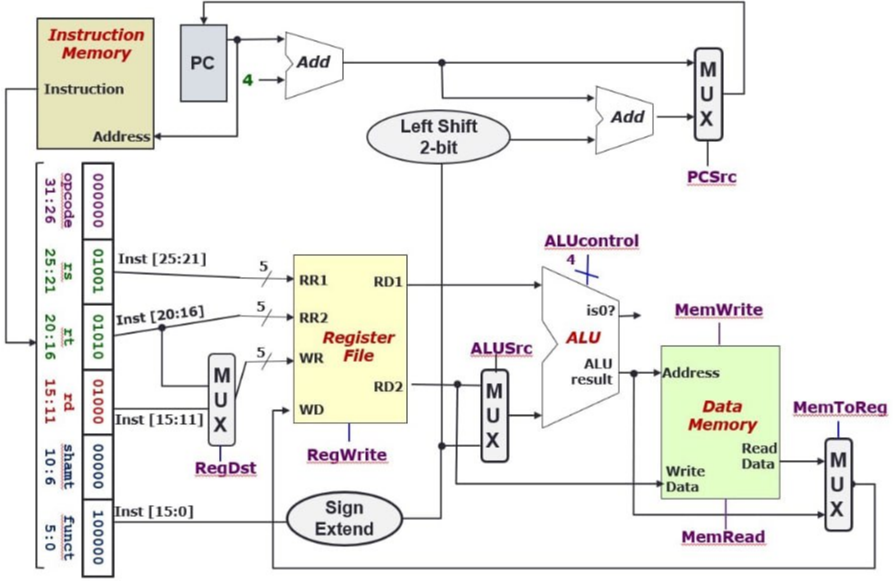
Q1: Identifying Forms of Parallelism

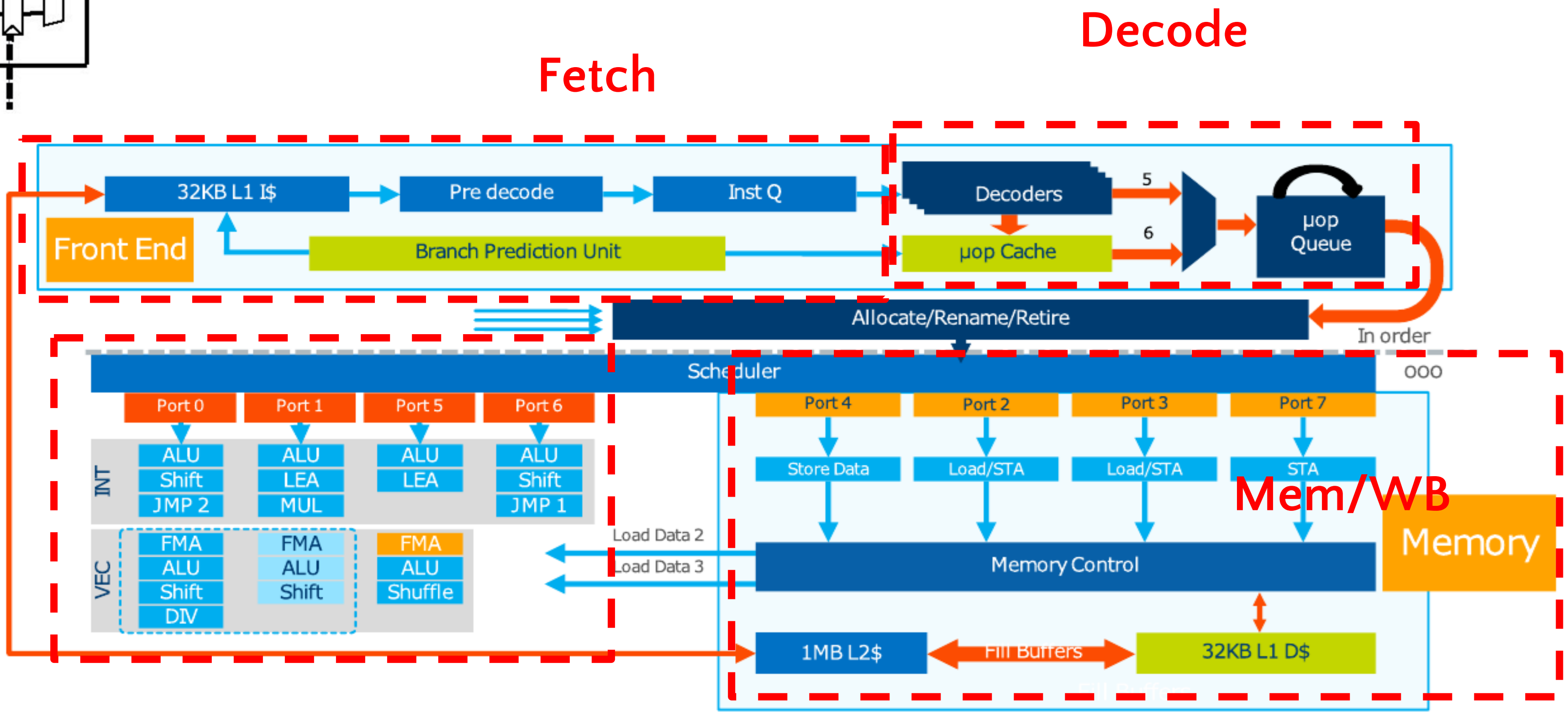
- Determine **where** and **how** our processor supports **parallelism**
- Don’t need to understand low-level details!
- Googling should allow you to match Lecture 3 concepts to these “industry names”



Mapping to CS2100

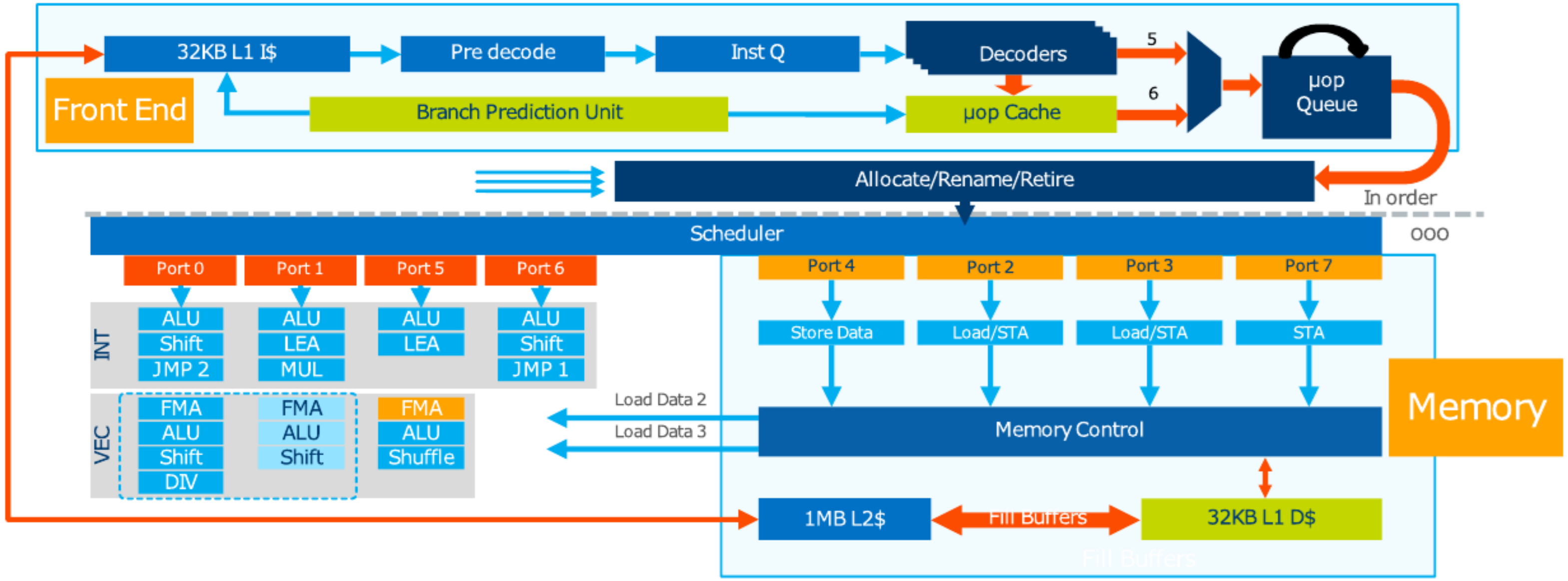






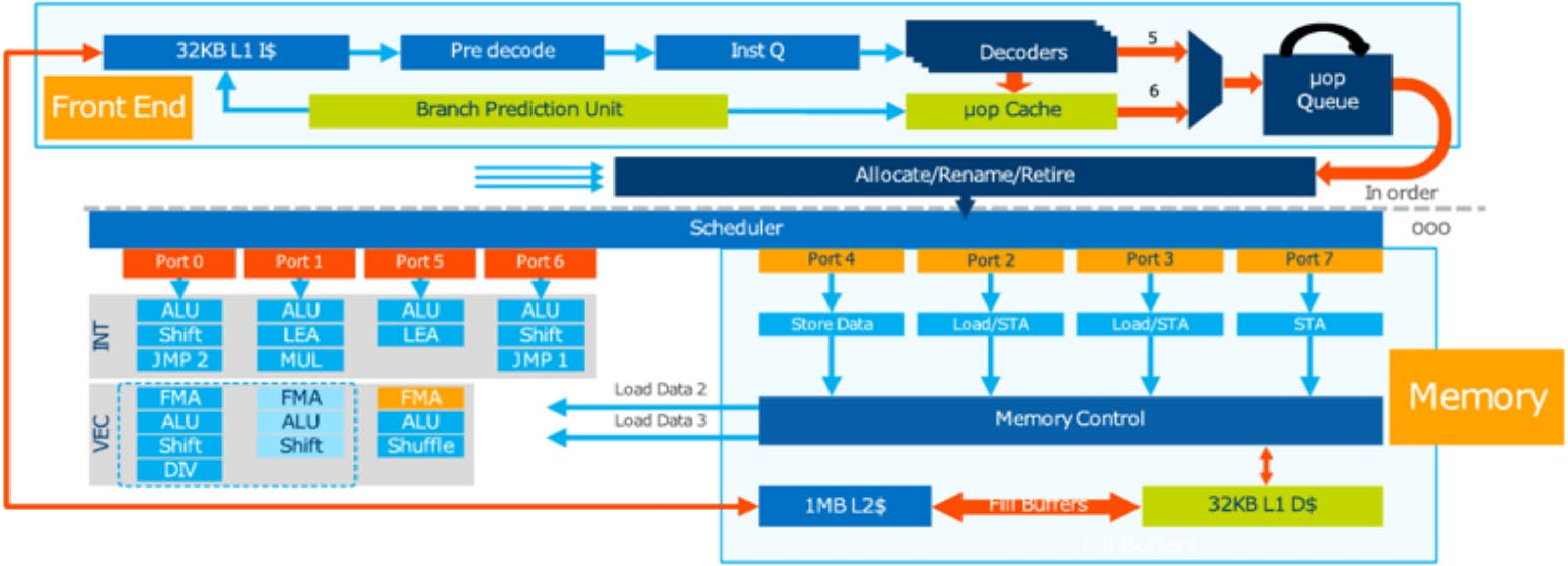
Q1: Identifying forms of parallelism

- Q: Where do we have *instruction-level* parallelism? [p]



Where do we have instruction-level parallelism?

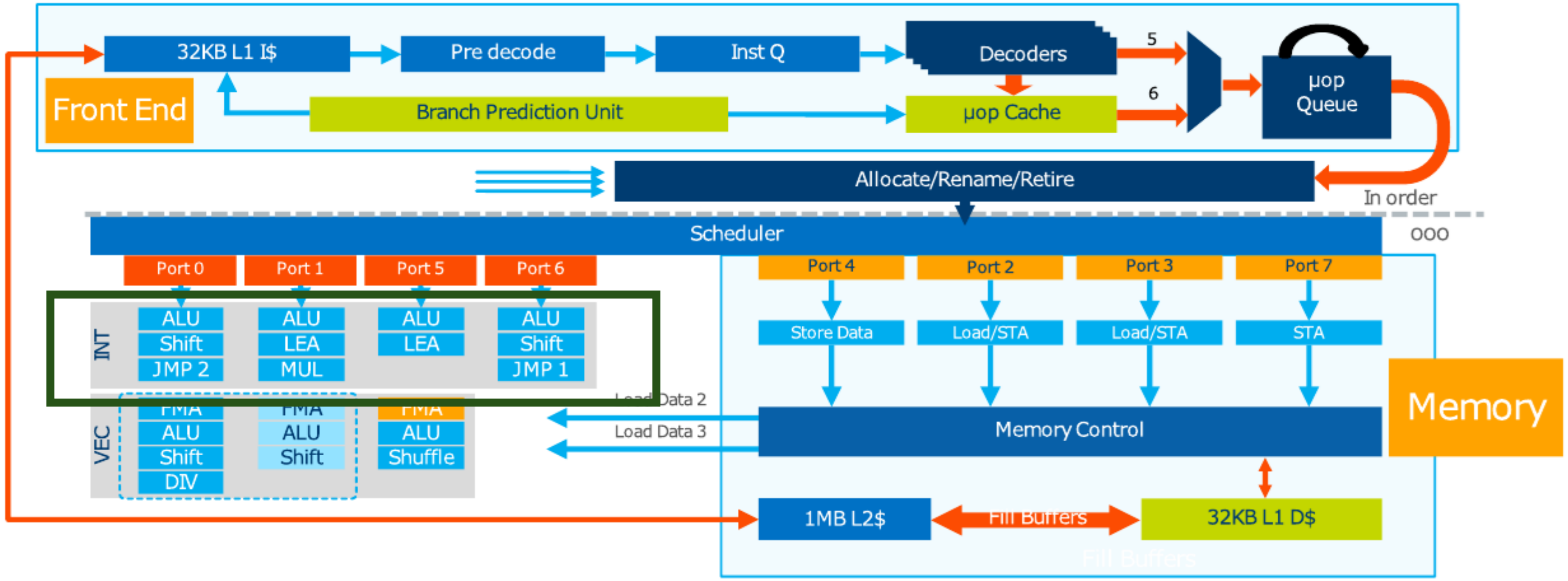
0



Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

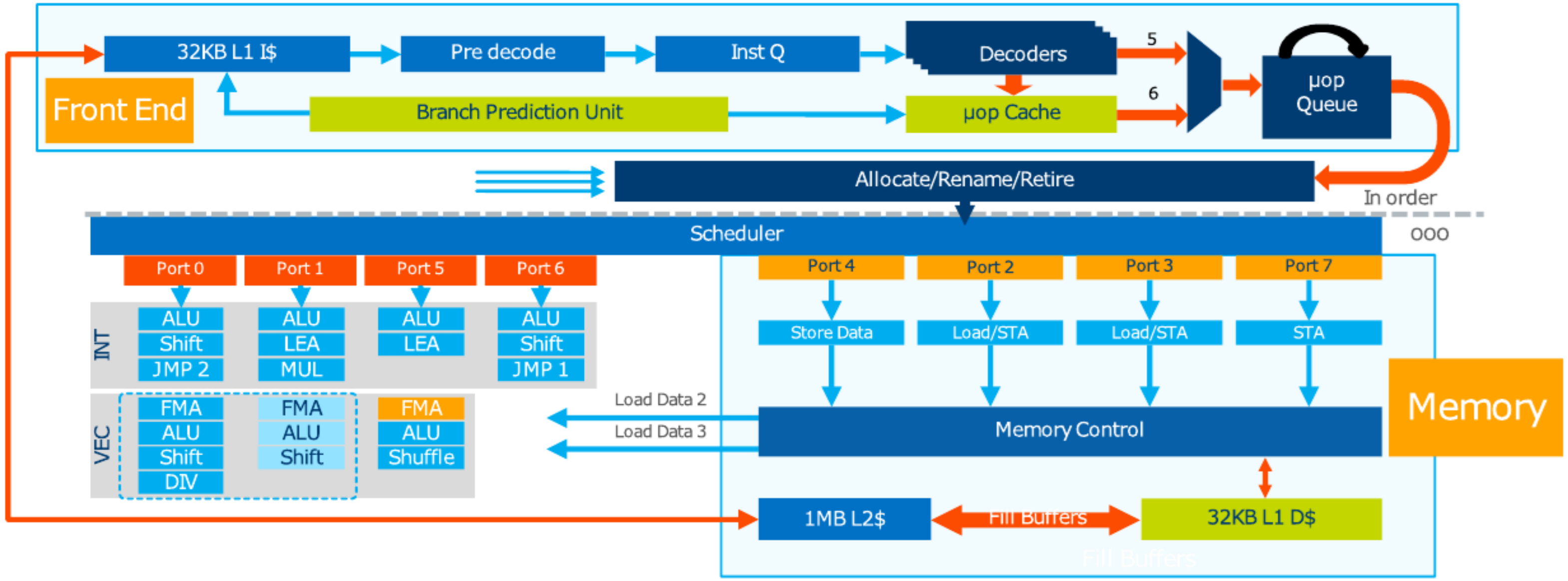
Q1: Identifying forms of parallelism

- Q: Where do we have *instruction-level* parallelism?
- One example of superscalar processing:
  multiple duplicated portions of the processor pipeline



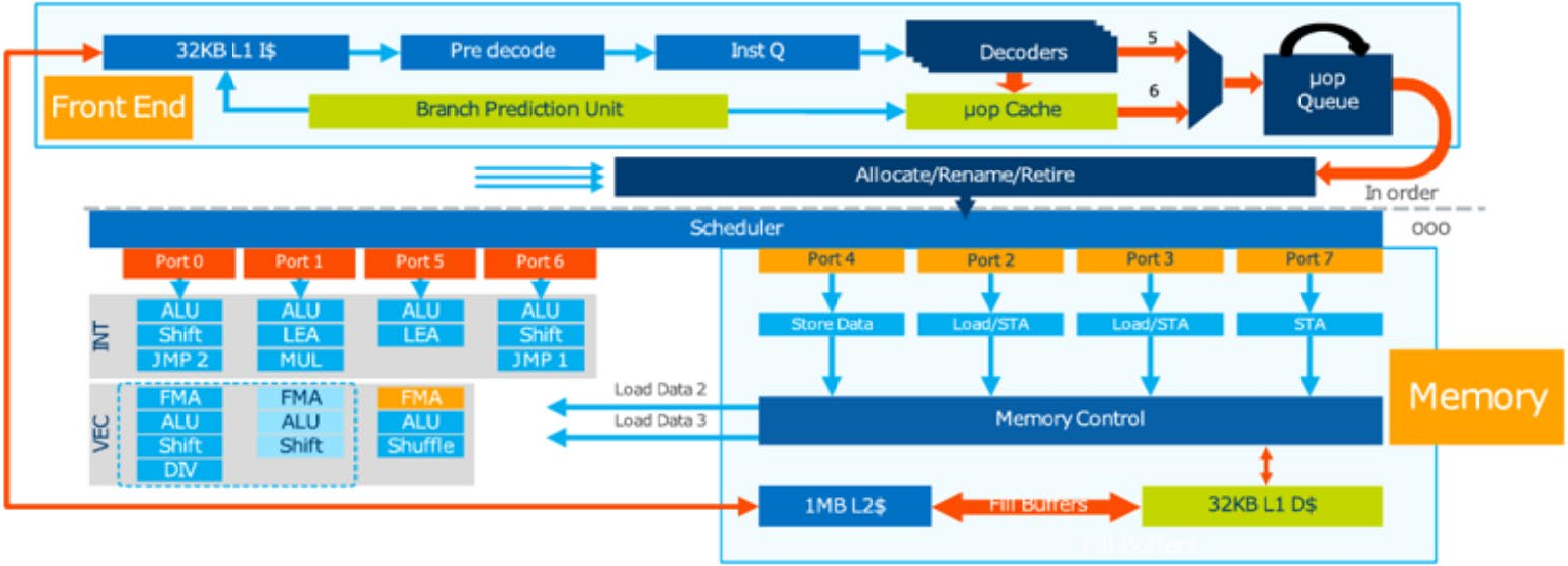
Q1: Identifying forms of parallelism

- Q: Where do we have *thread-level* parallelism? [p]



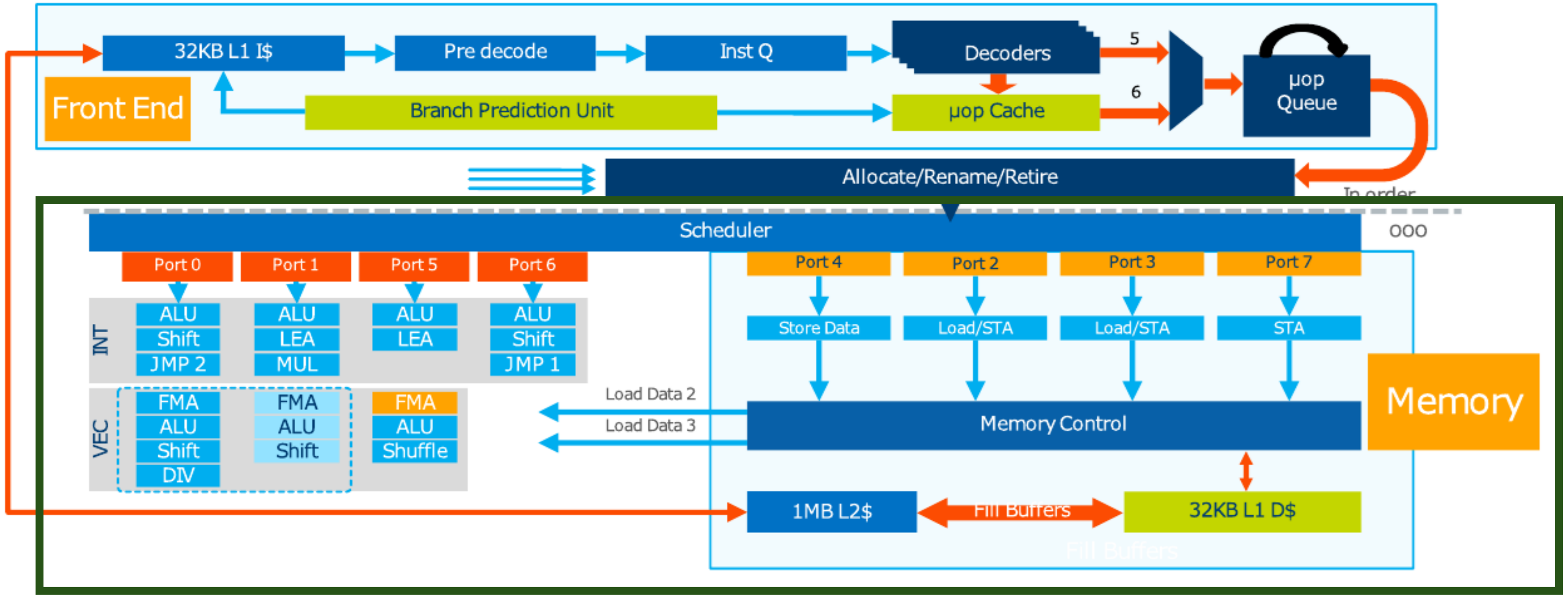
Where do we have thread-level parallelism?

0



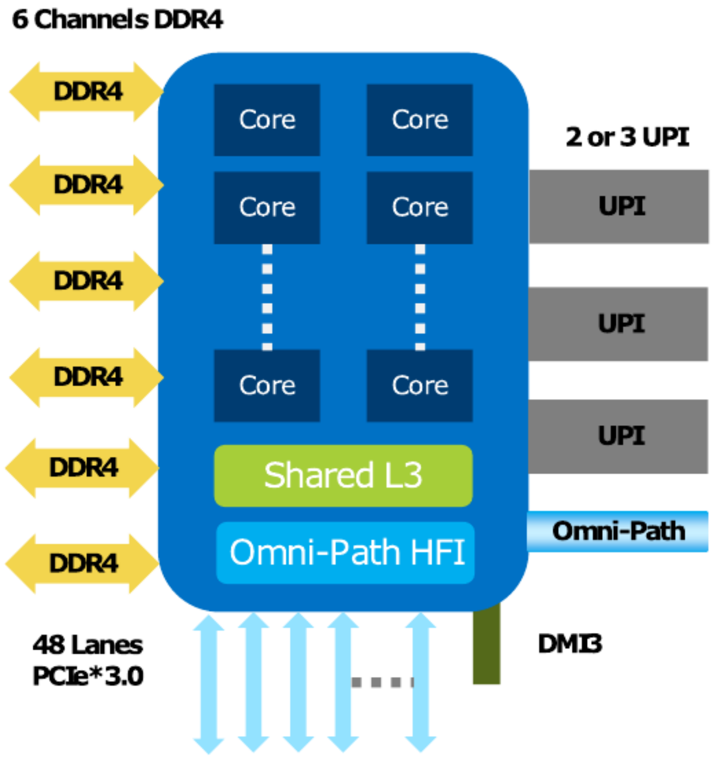
Q1: Identifying forms of parallelism

- Q: Where do we have *thread-level* parallelism?
- This entire diagram represents the pipeline for one core with 2 *hardware threads! They share the resources in the core.*



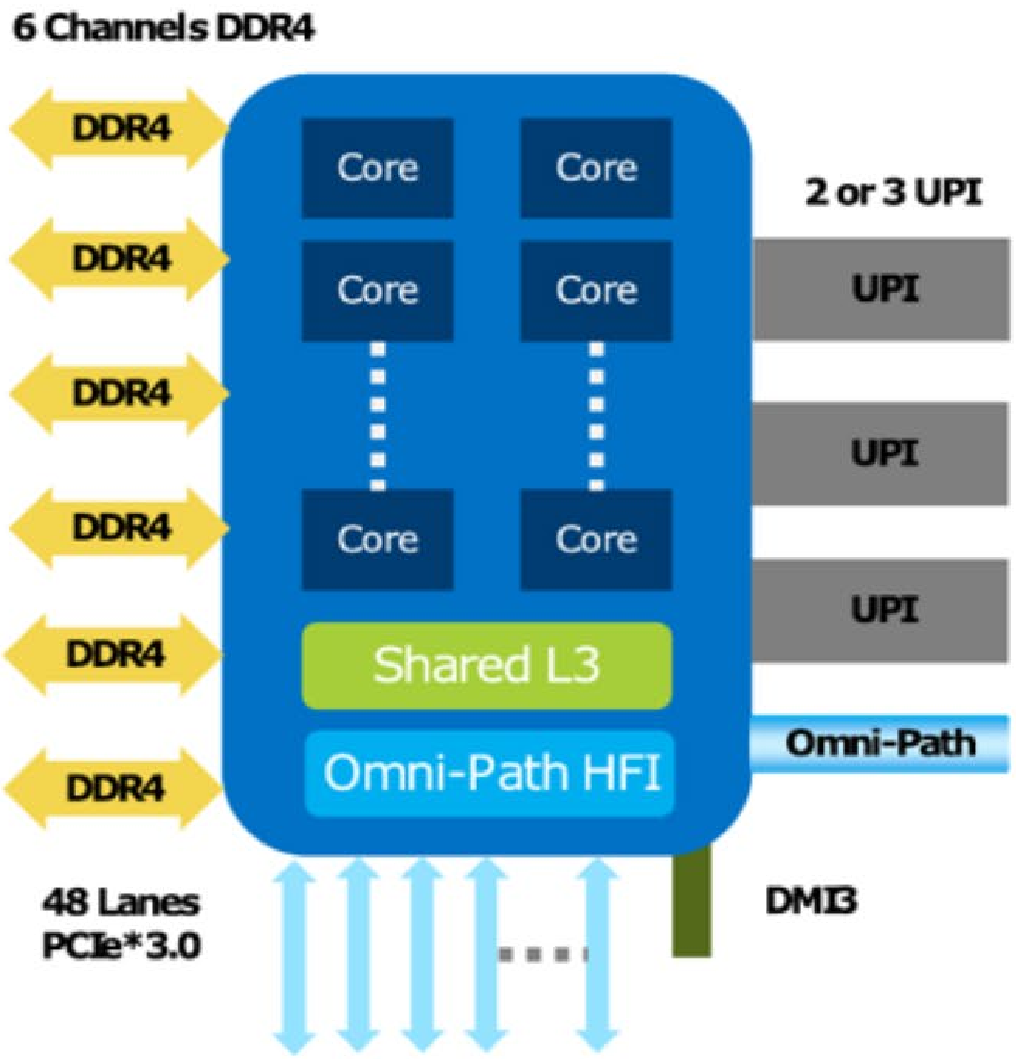
Q1: Identifying forms of parallelism

- Q: Where do we have *processor-level* parallelism? [p]



Where do we have processor-level parallelism?

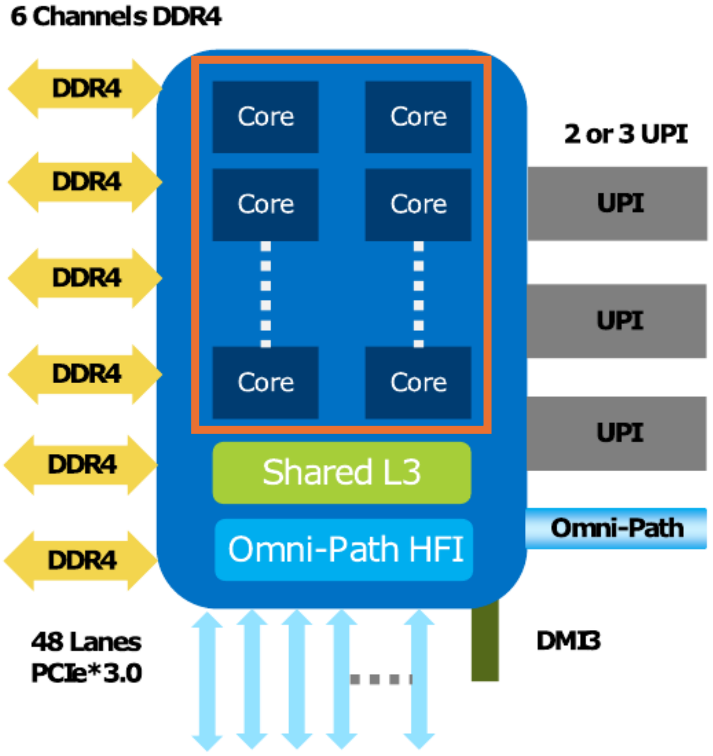
0



Q1: Identifying forms of parallelism

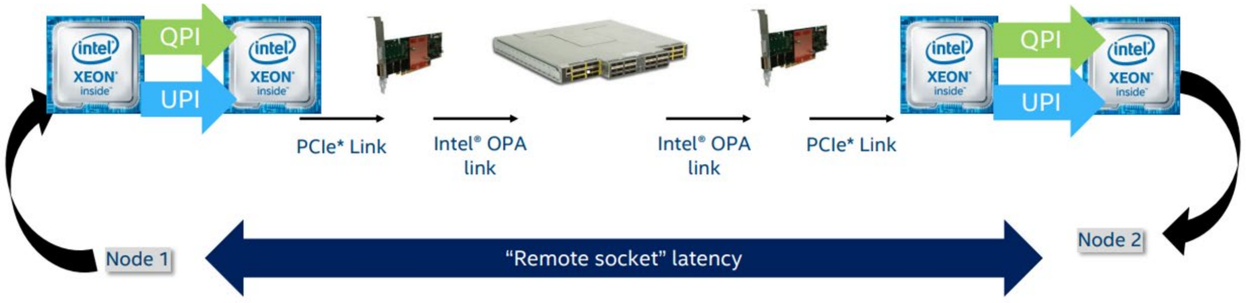
- Q: Where do we have *processor-level* parallelism?
- Each processor core runs individually: parallelism

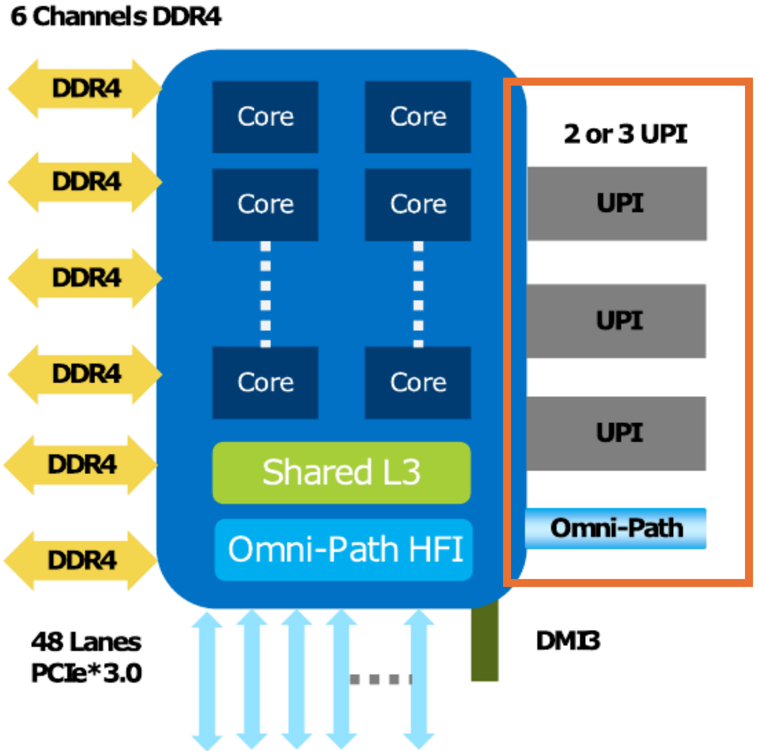
- Any more?



Q1: Identifying forms of parallelism

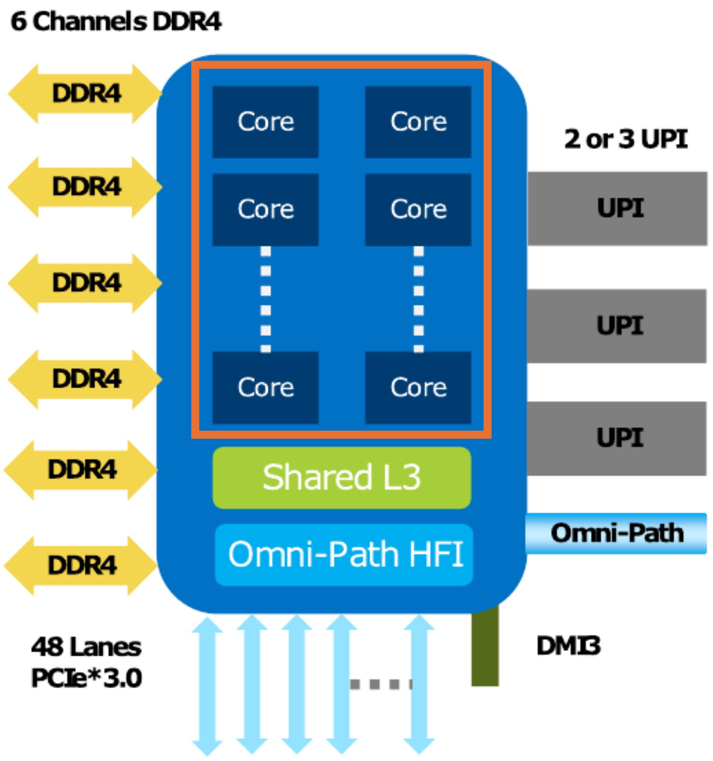
- Q: Where do we have *processor-level* parallelism?
  - Intel UltraPath Interconnect (UPI): Processor to Processor *(shared memory)*
  - Intel OmniPath Architecture (OPA): Node to node *(distributed memory)*





Q1: Identifying forms of parallelism

- Bonus: Where can we find I/O parallelism?
- Multiple DDR4 channels/ PCIexpress lanes



Why does this matter?

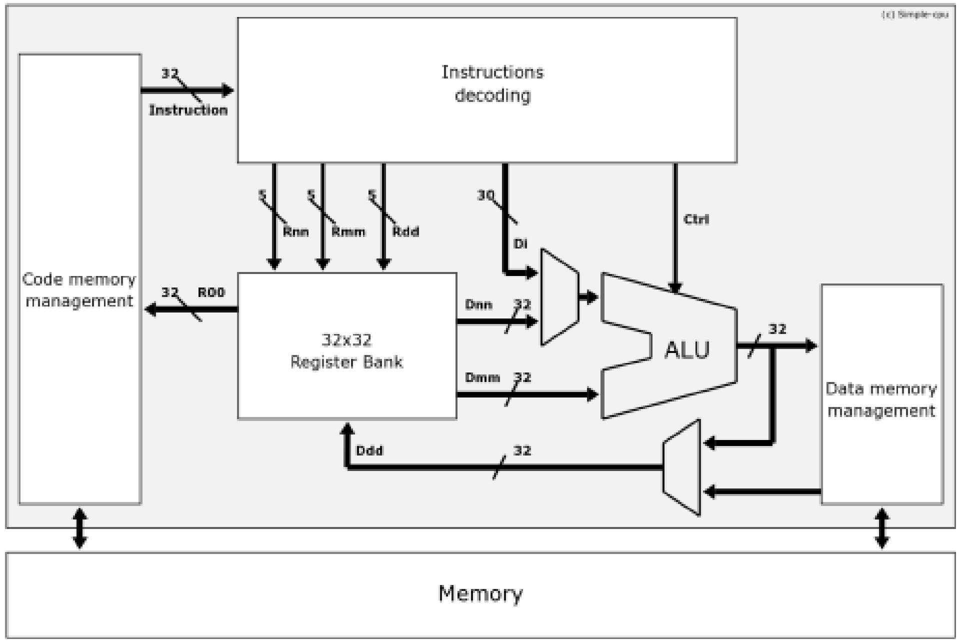
- When you run your code on certain hardware, you can make decisions about **how to parallelize.** Examples:
    - Node has multiple processors connected without high-speed interconnect?
      Maybe keep tasks to 1 processor.
    - Cores don't have much superscalar ability?
      Maybe best to use simpler tasks over many cores.

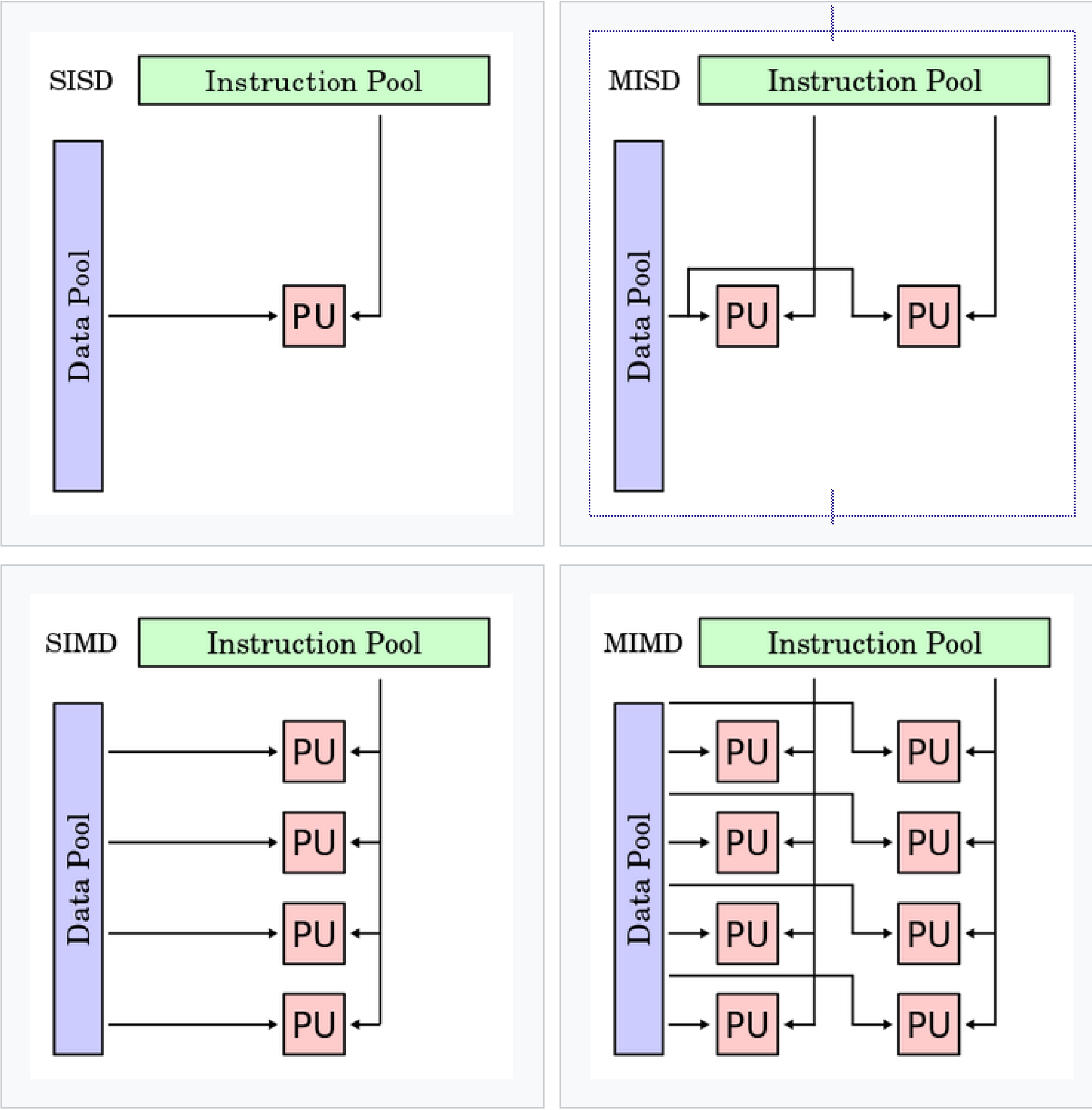
Summary

- Connecting to lab machines
- Processes and threads
- Mutexes / Semaphores / Condition Variables

Q2: Flynn's Taxonomy

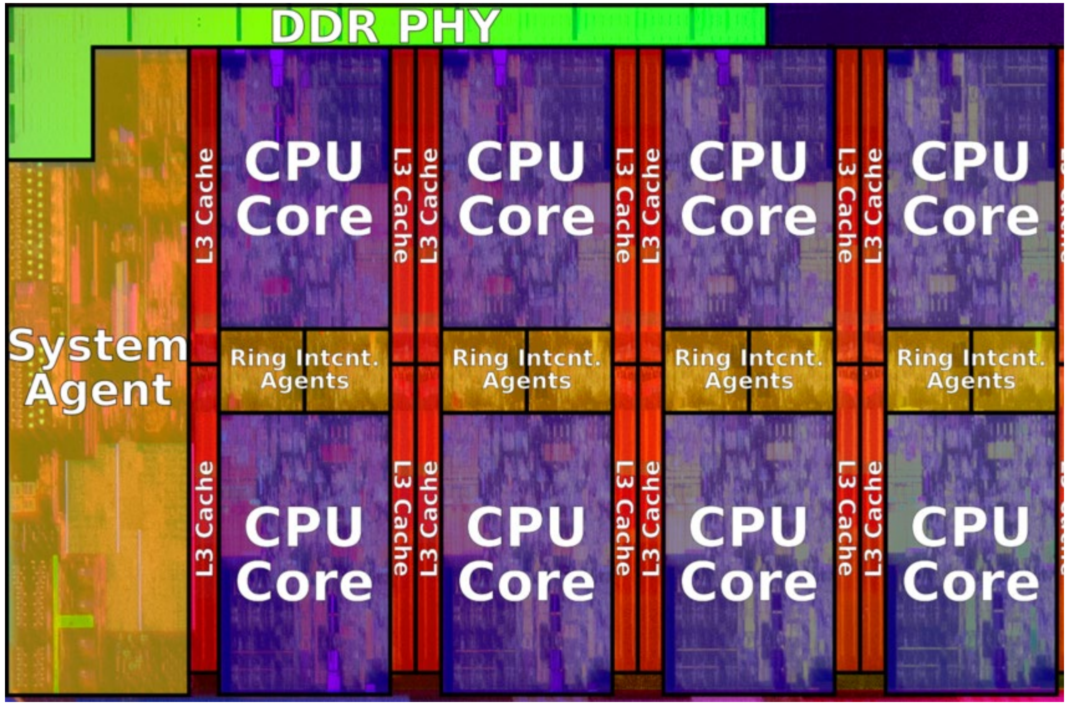
- What architecture for a personal computer from the 1980's?
- SISD: one instr and data stream

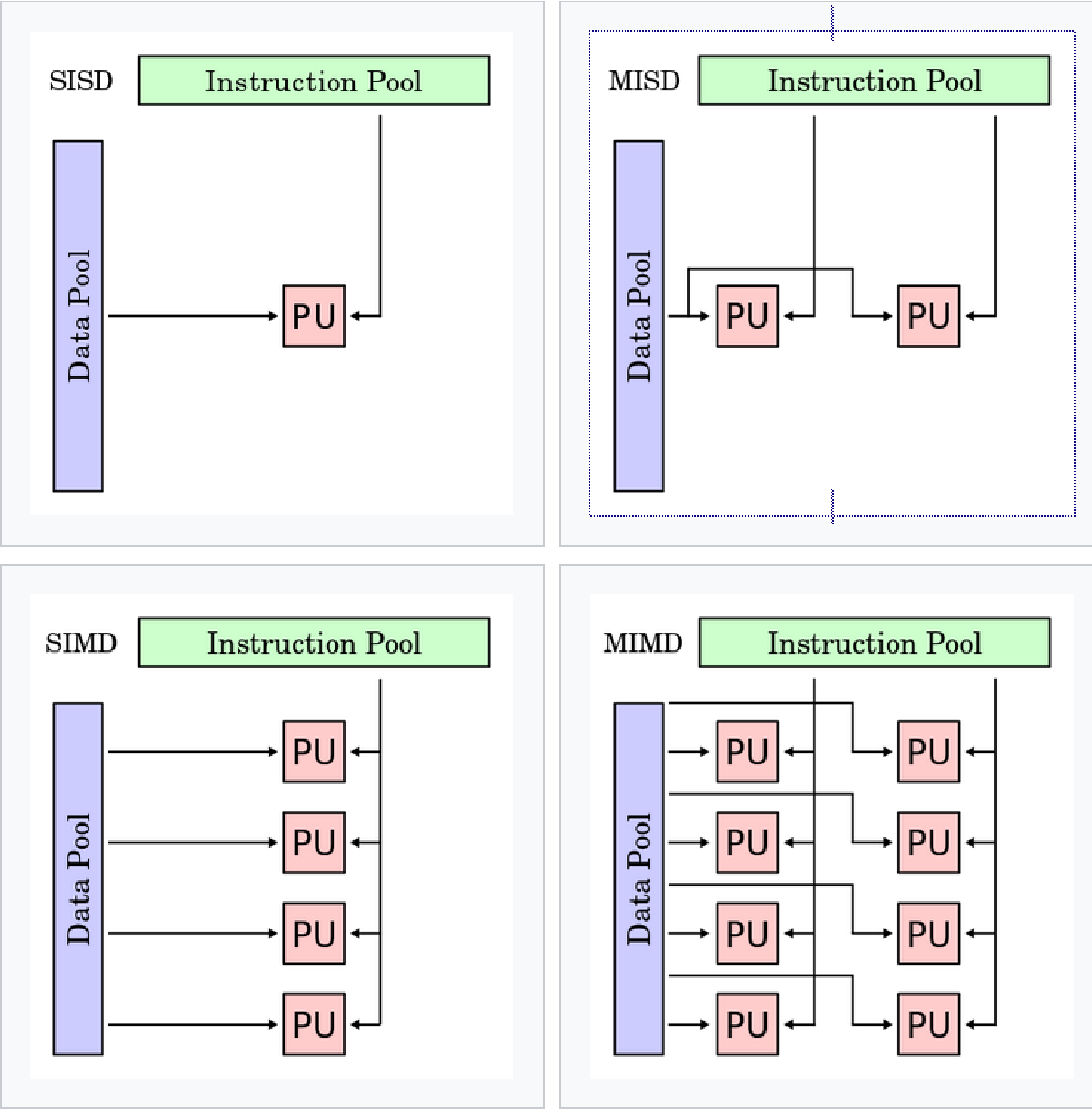




Q2: Flynn’s Taxonomy

- What architecture for a multi-core processor laptop?
- MIMD: Each core runs in parallel, independent data, or…





Q2: Flynn's Taxonomy

- What architecture for the Intel AVX instruction set?
- SIMD: one instruction operates on multiple data blocks

```
double *x, *y, *z;
for (i=0;  i<n;  i++)      z[i] = x[i] + y[i];
```

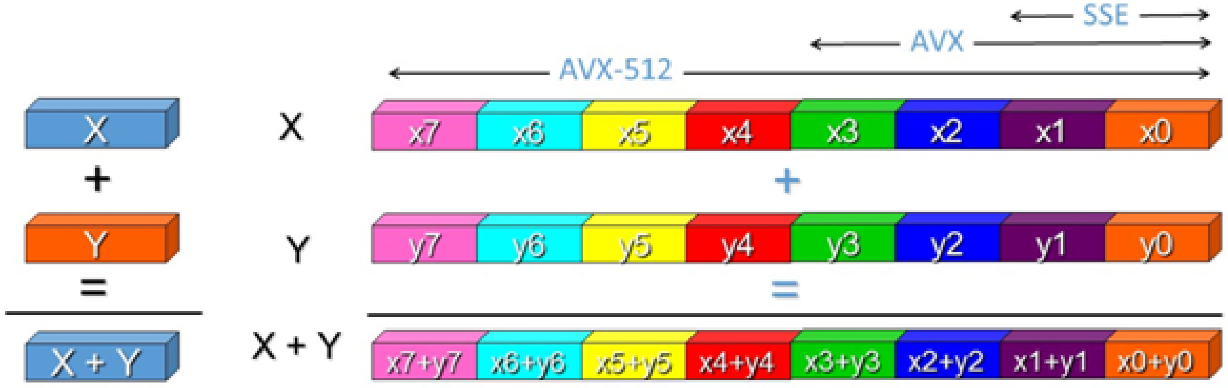
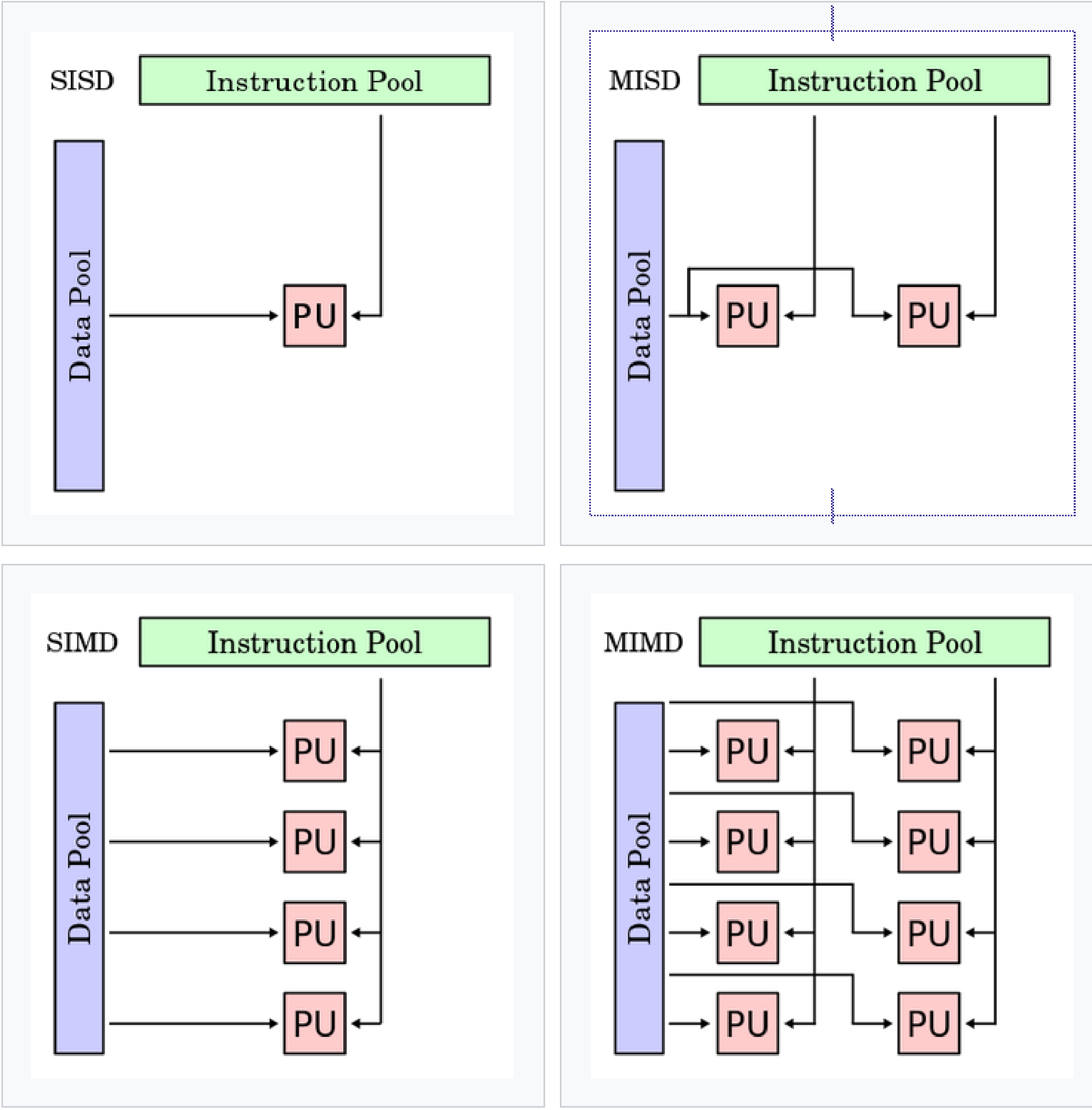
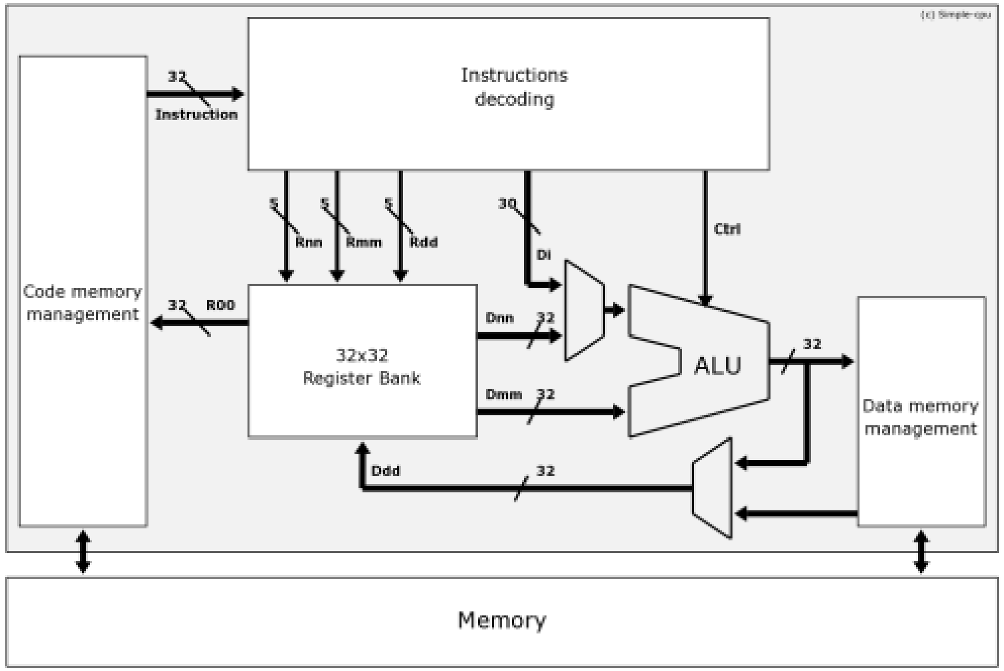


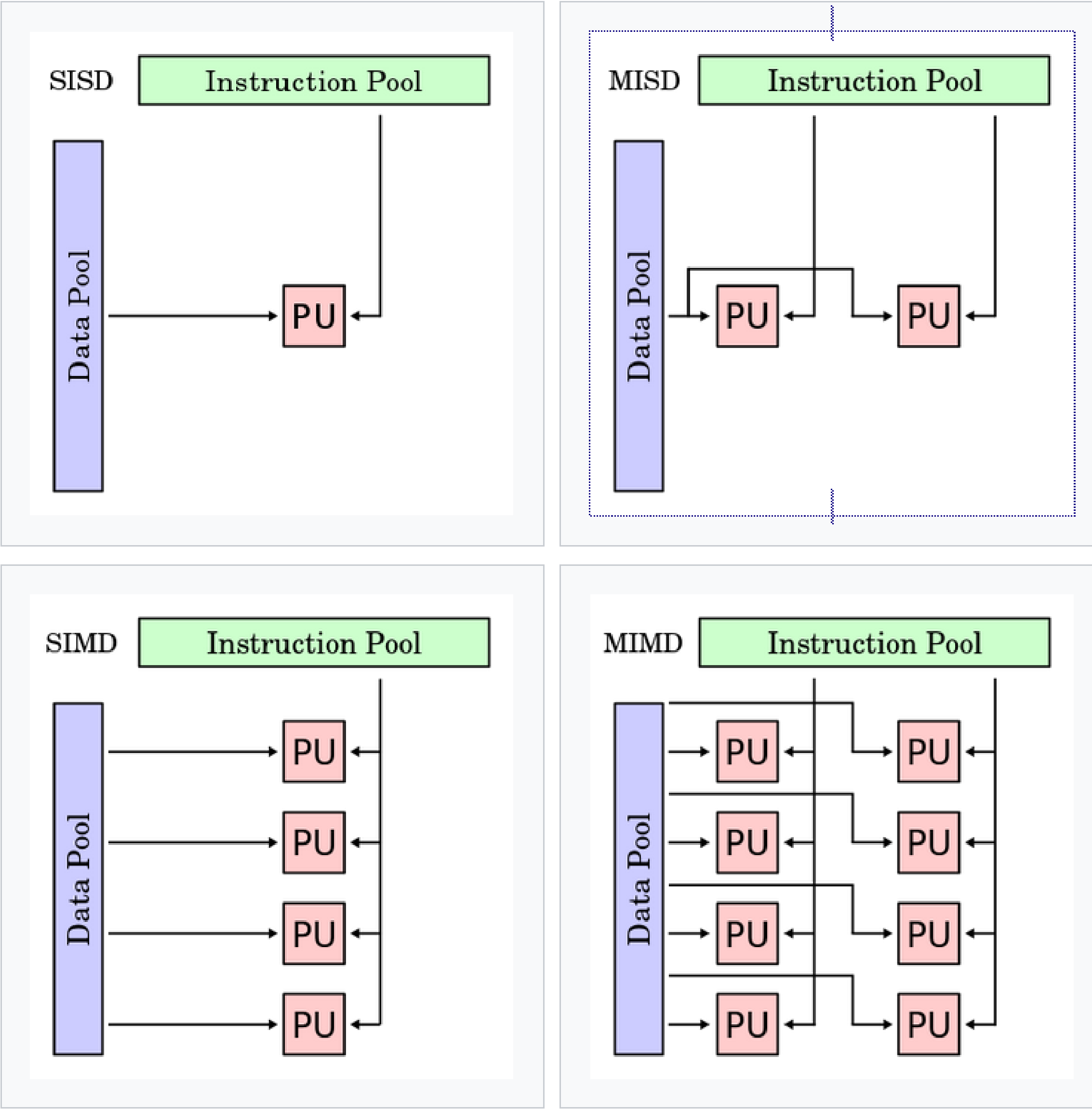
Figure 1 Scalar and vectorized loop versions with Intel® SSE, AVX and AVX-512.



Q2: Flynn's Taxonomy

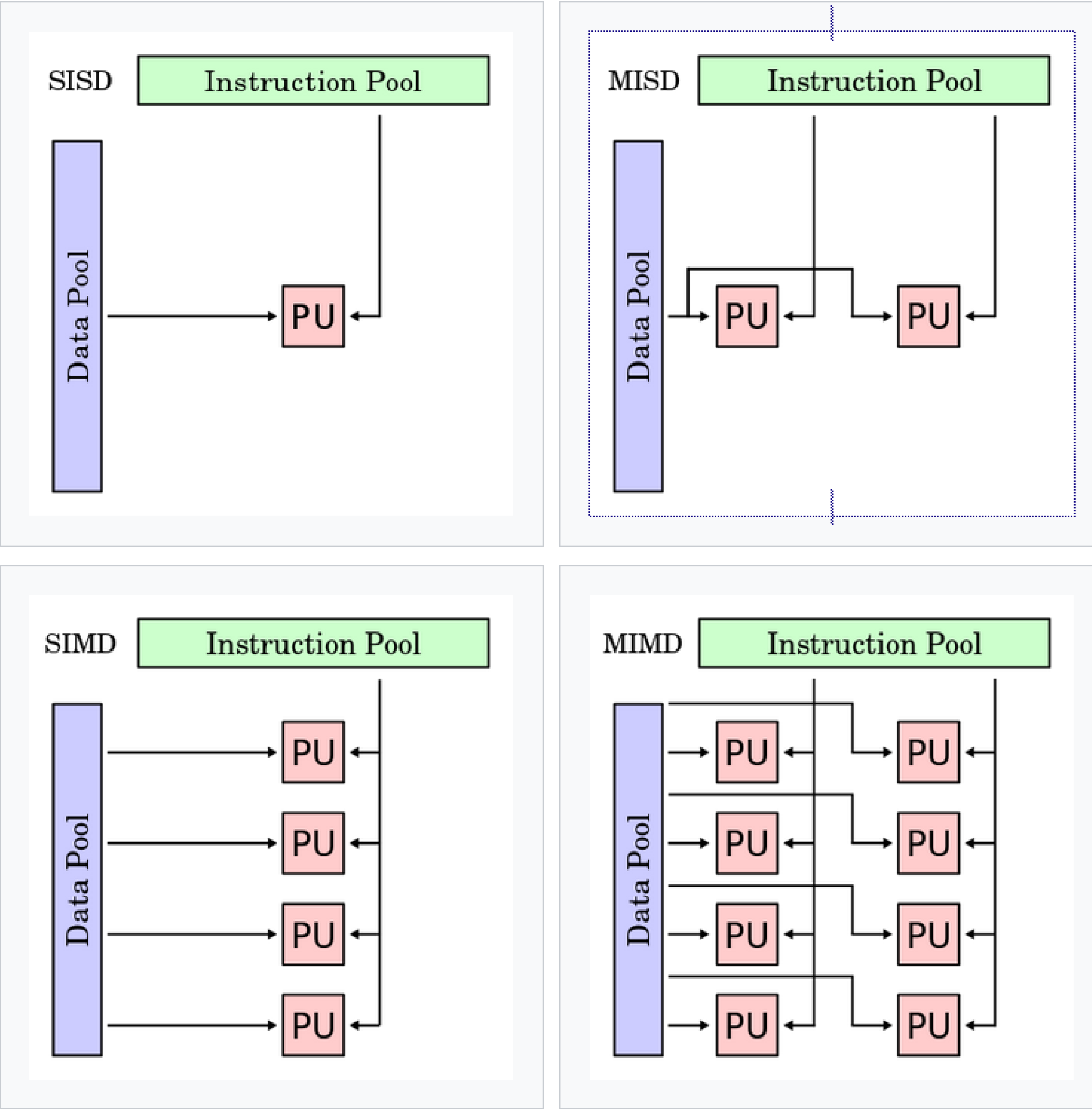
- What architecture for a single core processor with pipelining?
- SISD: same as first question



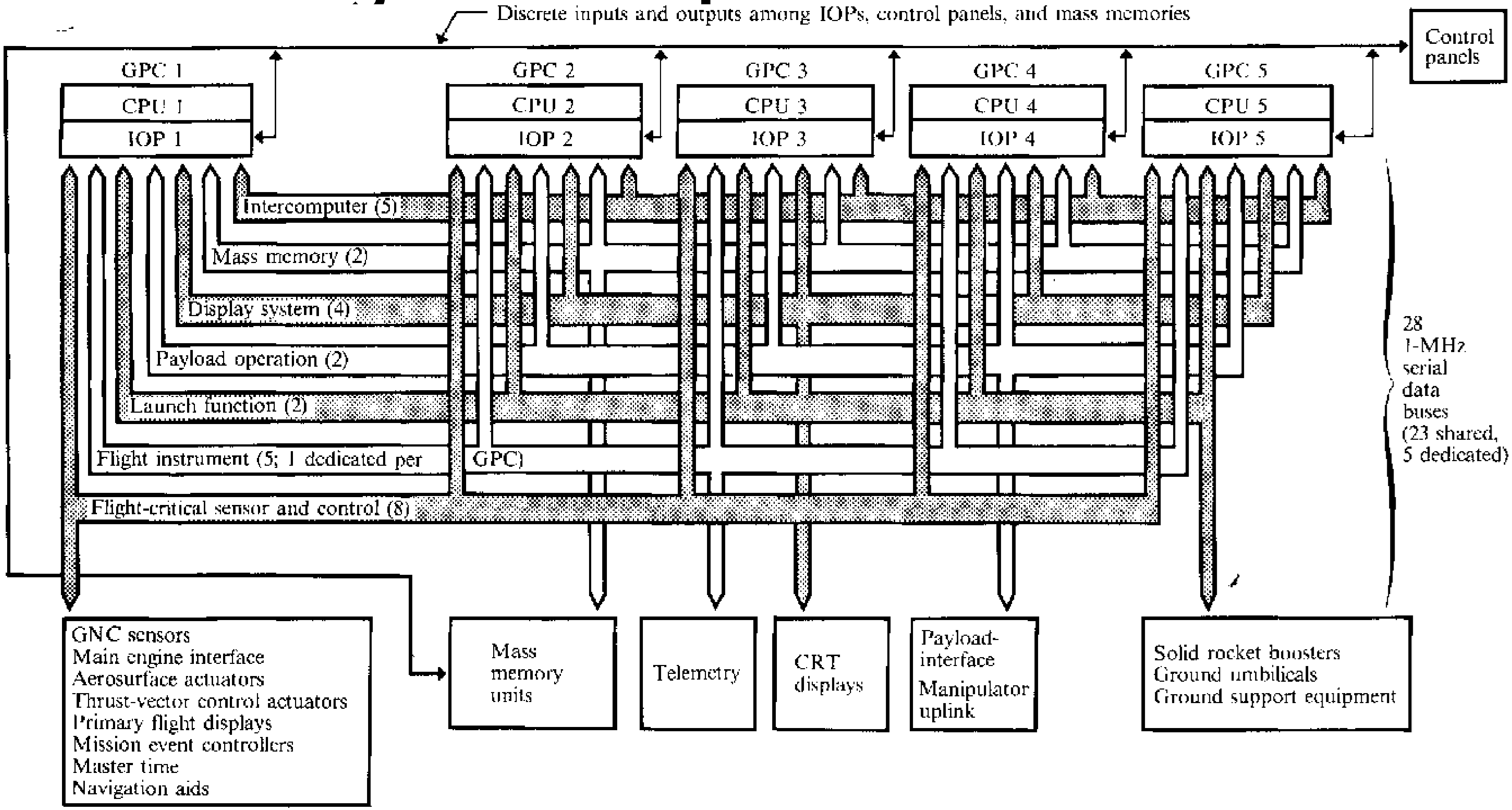


Q2: Flynn’s Taxonomy

- Students in an exam hall?
- MISD: same data (exam paper) but multiple independent streams (students)



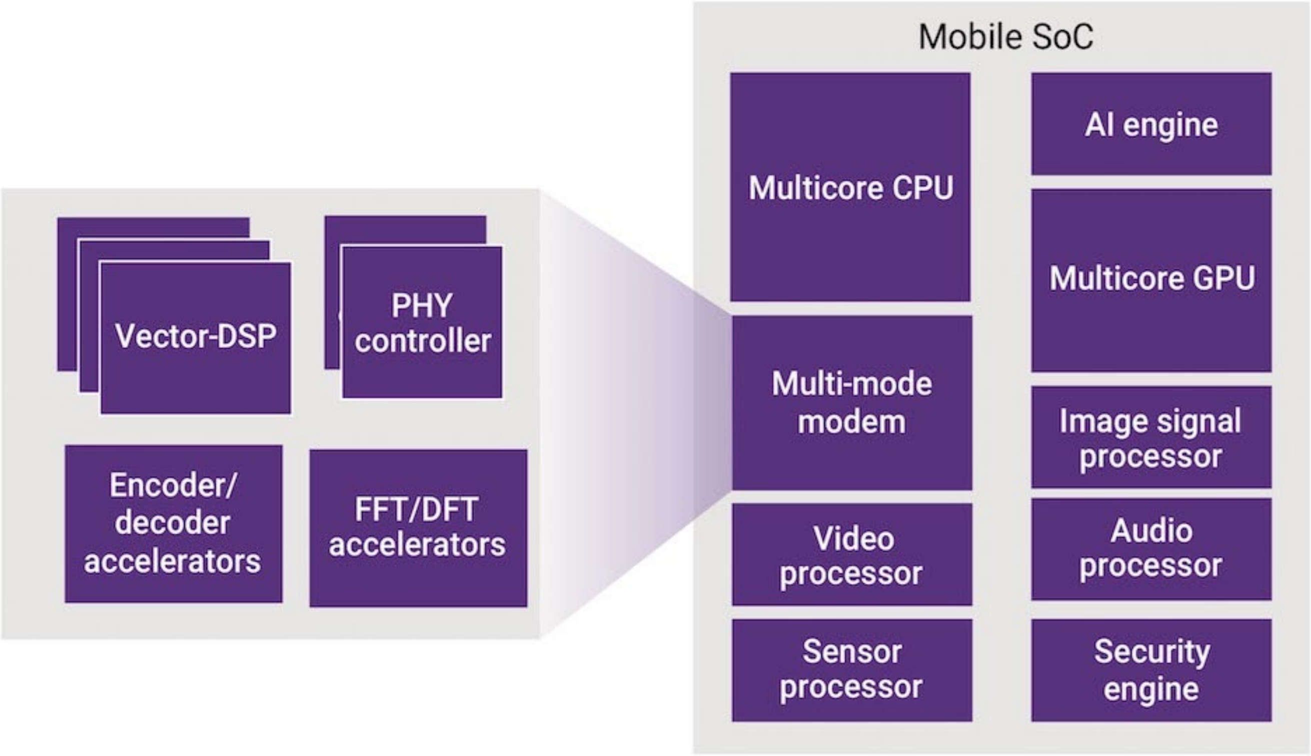
Interesting MISD: Space Shuttle CPUs



https://www.semanticscholar.org/paper/Redundancy-Management-Technique-for-Space-Shuttle-Sklaroff/eb225e9ad32eaeb29583ee0806d9381522de4959

Why does this matter?

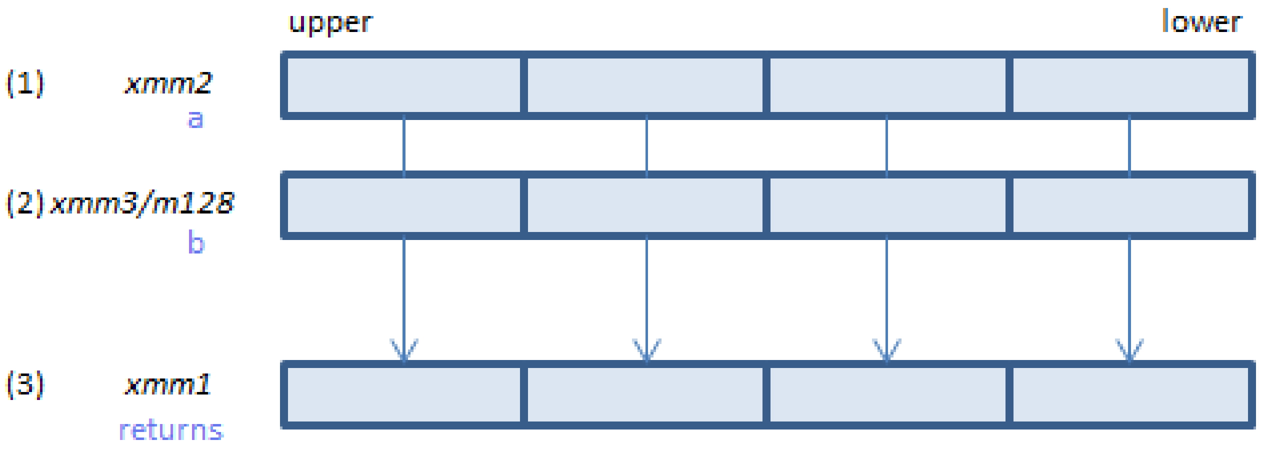
- We see all kinds of ***specialized processors*** nowadays
- Tradeoff between speed and generality
- Taxonomies help us to understand the rough **capabilities** of new processing units



Bonus: Quick look at AVX (SIMD) Instructions

- Note - you don't have to know details!
- Just that such things (vector instructions) exist

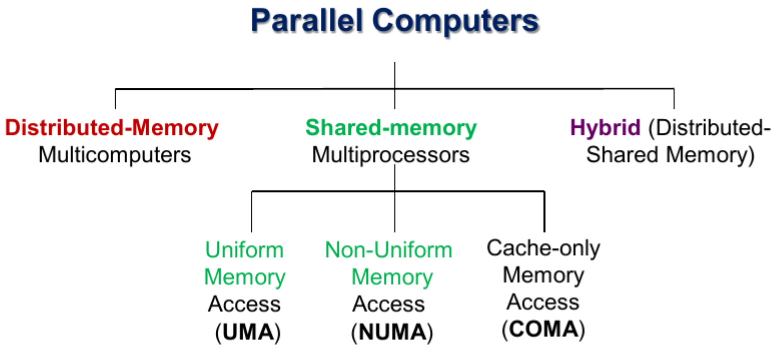
https://godbolt.org/z/fbTeE4jb7

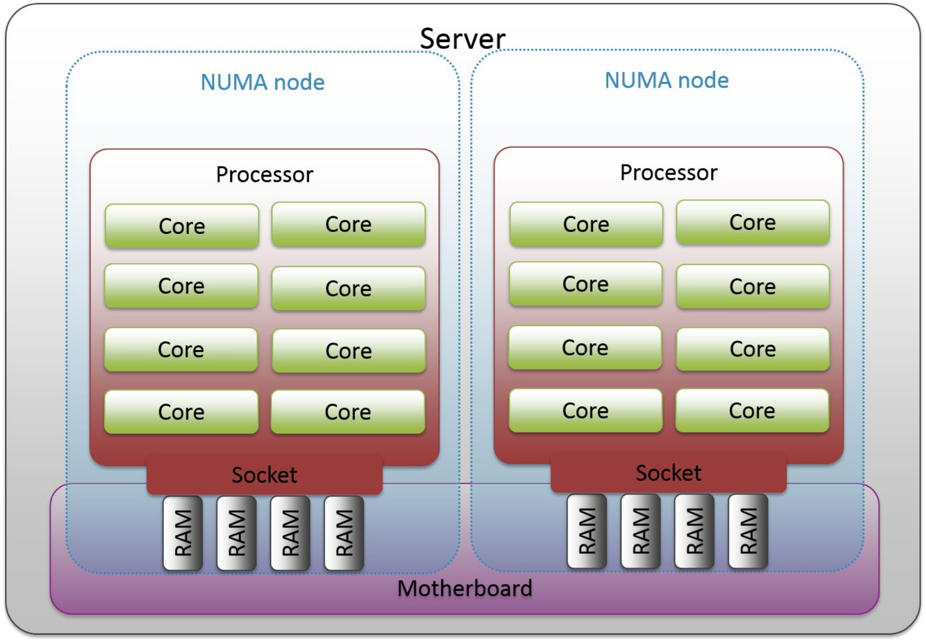


vpaddd

Q3: Memory & Multicore Architectures

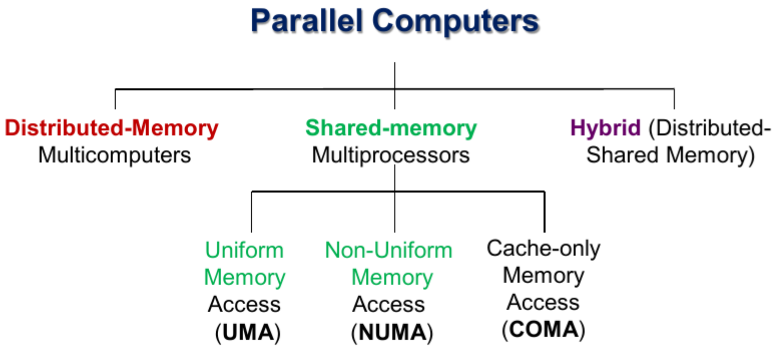
- Shared memory implies a UMA architecture? (T/F?)
- No! Memory access need not be uniform!


Parallel Computers
Distributed-Memory Multicomputers
Shared-memory Multiprocessors
Hybrid (Distributed-Shared Memory)
Uniform Memory Access (UMA)
Non-Uniform Memory Access (NUMA)
Cache-only Memory Access (COMA)

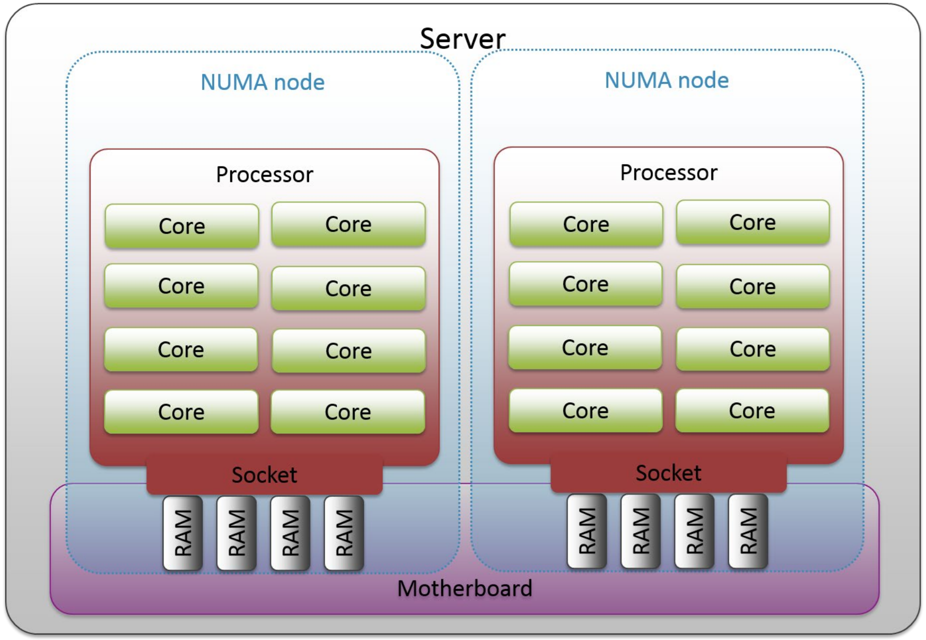


Q3: Memory & Multicore Architectures

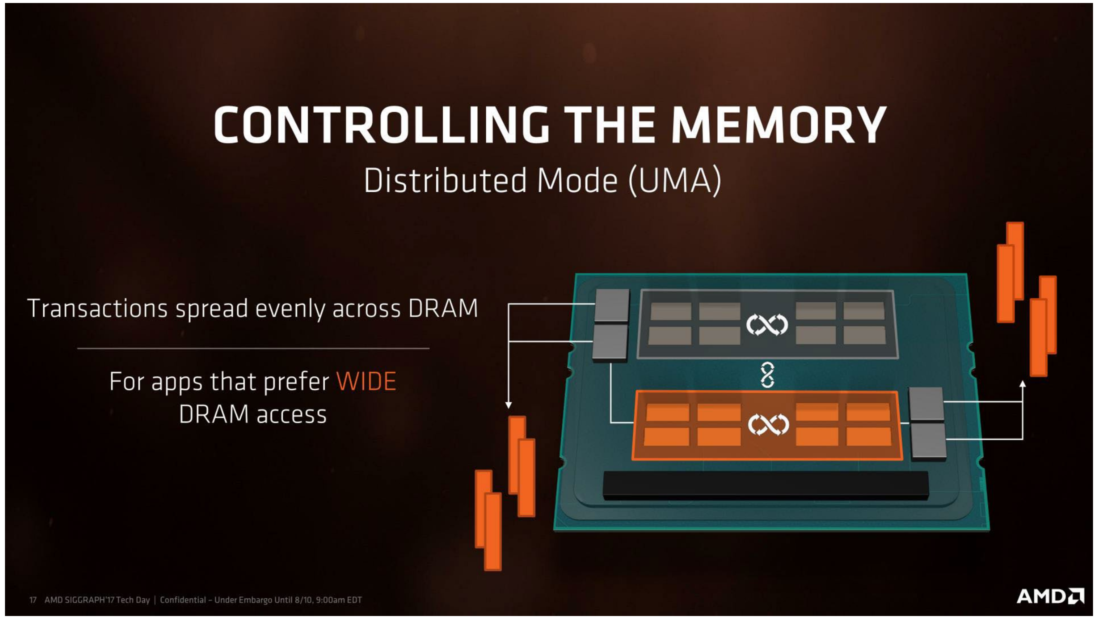
- Answering Q2 at the same time: **where your data is, is important in a NUMA architecture!**



Parallel Computers
Distributed-Memory Multicomputers
Shared-memory Multiprocessors
Hybrid (Distributed-Shared Memory)
Uniform Memory Access (UMA)
Non-Uniform Memory Access (NUMA)
Cache-only Memory Access (COMA)



Bonus: AMD Ryzen Threadripper



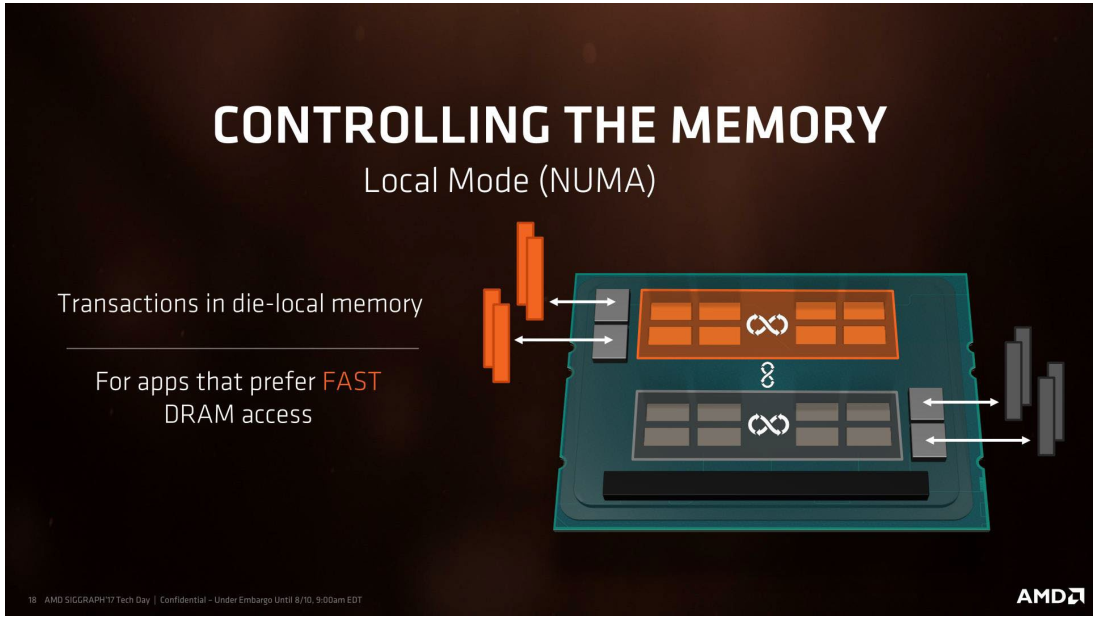
CONTROLLING THE MEMORY

Distributed Mode (UMA)

Transactions spread evenly across DRAM

For apps that prefer WIDE DRAM access

17 AMD SIGGRAPH'17 Tech Day | Confidential – Under Embargo Until 8/10, 9:00am EDT



CONTROLLING THE MEMORY

Local Mode (NUMA)

Transactions in die-local memory

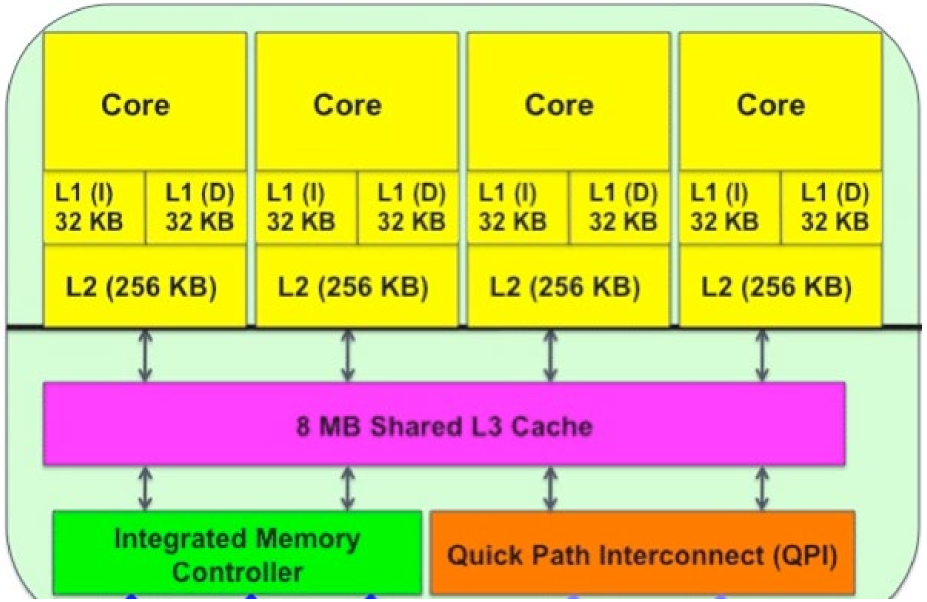
For apps that prefer FAST DRAM access

18 AMD SIGGRAPH'17 Tech Day | Confidential – Under Embargo Until 8/10, 9:00am EDT

Extra reading: https://en.wikichip.org/wiki/amd/microarchitectures/zen%2B

Q3: Memory & Multicore Architectures

- In hierarchical multicore designs, is the memory organization **hybrid** (distributed + shared memory)? [p]



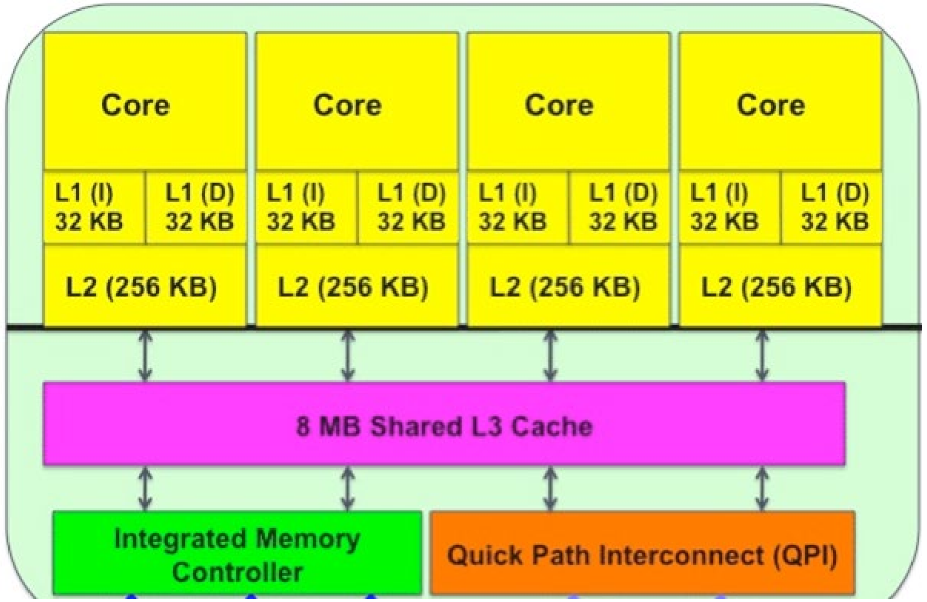
In hierarchical multicore designs, is the memory organization hybrid (distributed + shared memory)?

✓ 0

Q3: Memory & Multicore Architectures

- In hierarchical multicore designs, is the memory organization **hybrid** (distributed + shared memory)?
- Both answers arguable!

- Not hybrid: everyone just sees one shared memory
- Hybrid: could see cache as its own memory that can be out of sync



Why does this matter?

- The quirks of distributed / shared memory are **crucial** for fast parallel programs
  - Very small L1 caches?
    Your parallel programs may not benefit from accessing lots of individual data per core.
  - NUMA processors?
    Best to keep data organization in mind

- A lot of performance can be squeezed out of good cache usage!

Q4: Processes and Threads

Q: Can a semaphore replace a mutex without affecting correctness? [p]

Can a semaphore replace a mutex without affecting correctness?

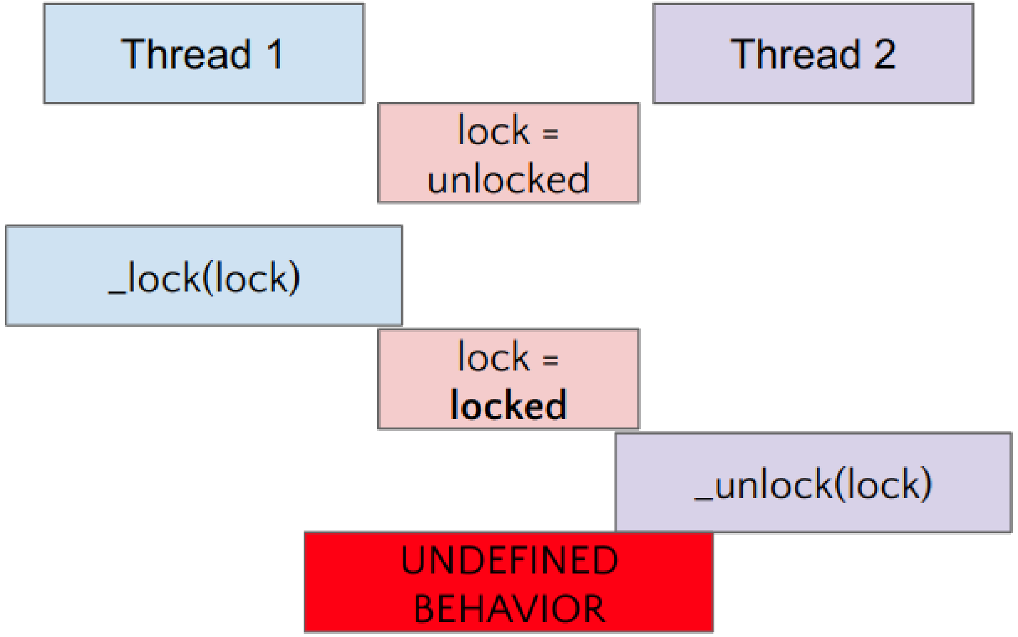
✓ 0

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Q4: Processes and Threads

- Can a **semaphore** replace a **mutex** without affecting correctness?
- Yes! Semaphores are more general than mutexes, just set S = 1

- pthread mutexes cannot do this



Thread 1

Thread 2

lock = unlocked

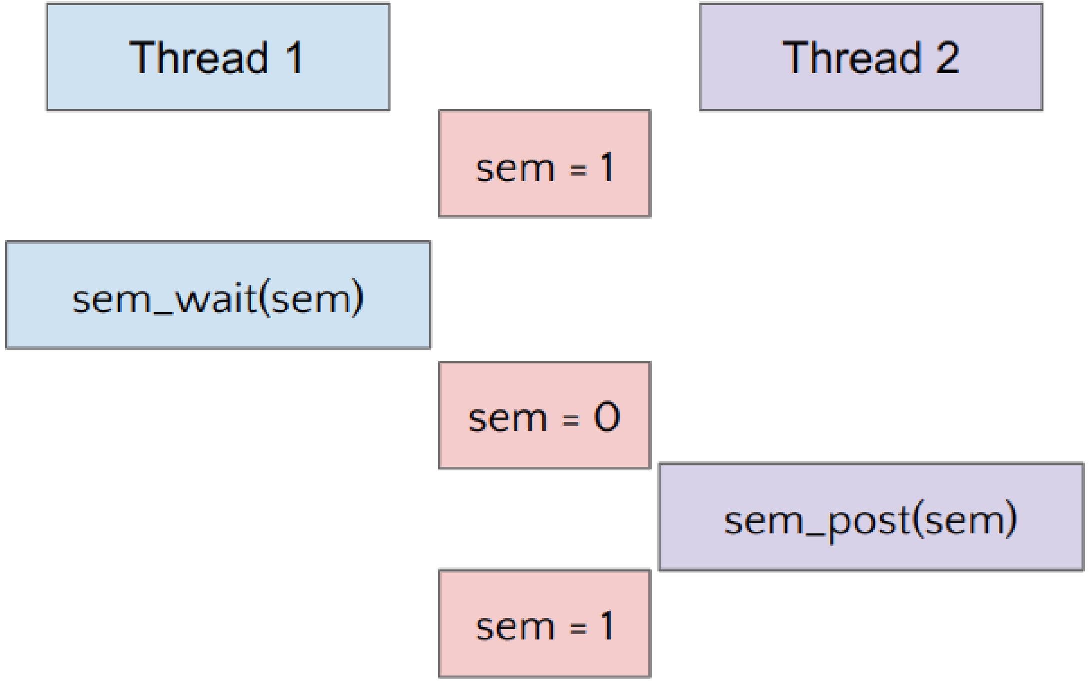
_lock(lock)

lock = **locked**

_unlock(lock)

UNDEFINED BEHAVIOR

- Semaphores can do this:



Thread 1

Thread 2

sem = 1

sem_wait(sem)

sem = 0

sem_post(sem)

sem = 1

Q4: Processes and Threads

- Can a **semaphore** replace a **mutex** without affecting correctness?
- If a program implemented with a mutex *is already correct*, we can replace it with a semaphore without affecting correctness!

Q4: Processes and Threads

- Is a program implemented with **threads** faster than **processes?**

Q4: Processes and Threads

- Is a program implemented with **threads** faster than **processes?**

- **Not necessarily, because:**
  - Could be only **user threads**
  - Condition variable usage / other logic could be slower than semaphores sometimes
  - **Rarely have such guarantees in systems work**

- **Your mileage might vary for Lab 1!**

Short break

Stretch, go to the toilet, buy drinks, or ask me questions

5-10 mins :D

Part 2+3: Slurm usage

Please follow at the same pace until later!

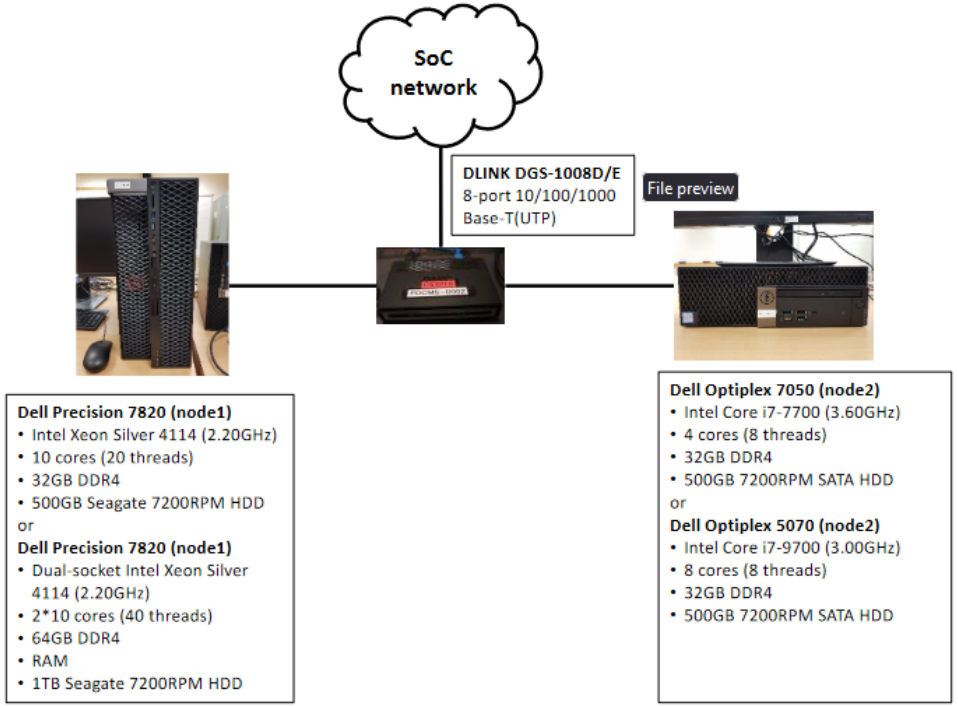
Reminder to self: attendance

Part 2 & 3: Practical Slurm Usage

Please follow at the same pace until later

ex1: Login to a machine and run `lstopo/lscpu`

- What’s the difference between sockets/cores/threads?



DLINK DGS-1008D/E
8-port 10/100/1000
Base-T(UTP)

Dell Precision 7820 (node1)
- Intel Xeon Silver 4114 (2.20GHz)
- 10 cores (20 threads)
- 32GB DDR4
- 500GB Seagate 7200RPM HDD

or

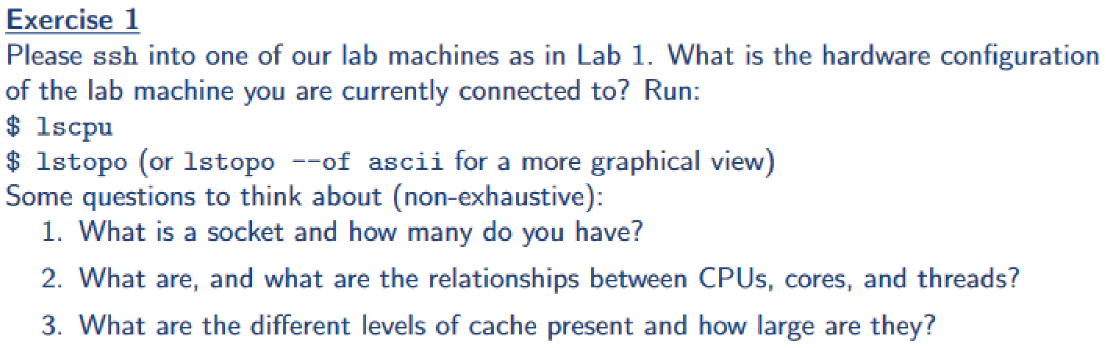
Dell Precision 7820 (node1)
- Dual-socket Intel Xeon Silver 4114 (2.20GHz)
- 2*10 cores (40 threads)
- 64GB DDR4
- RAM
- 1TB Seagate 7200RPM HDD

Dell Optiplex 7050 (node2)
- Intel Core i7-7700 (3.60GHz)
- 4 cores (8 threads)
- 32GB DDR4
- 500GB 7200RPM SATA HDD

or

Dell Optiplex 5070 (node2)
- Intel Core i7-9700 (3.00GHz)
- 8 cores (8 threads)
- 32GB DDR4
- 500GB 7200RPM SATA HDD

Figure 1: Example layout of machines on each workbench (not all machines shown)



Exercise 1

Please ssh into one of our lab machines as in Lab 1. What is the hardware configuration of the lab machine you are currently connected to? Run:

```
$ lscpu
```

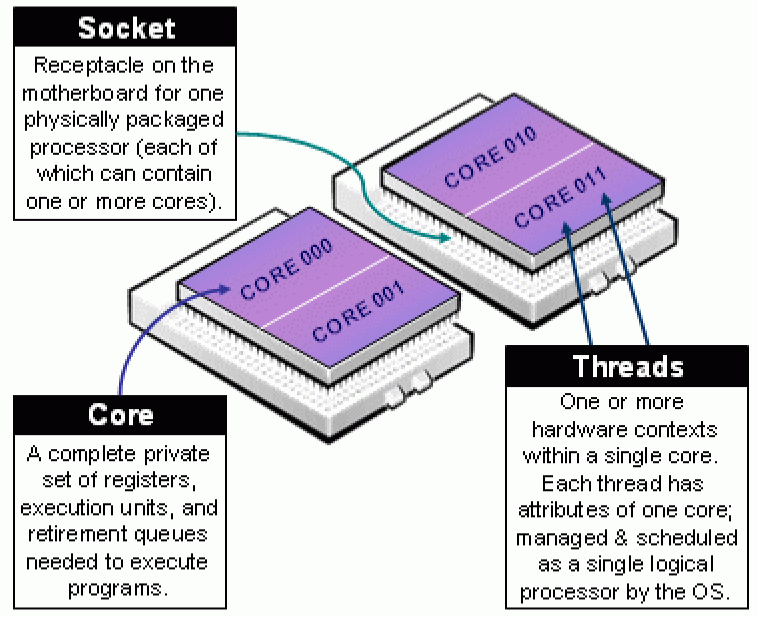
`$ lstopo` (or `lstopo --of ascii` for a more graphical view)

Some questions to think about (non-exhaustive):

1. What is a socket and how many do you have?
2. What are, and what are the relationships between CPUs, cores, and threads?
3. What are the different levels of cache present and how large are they?

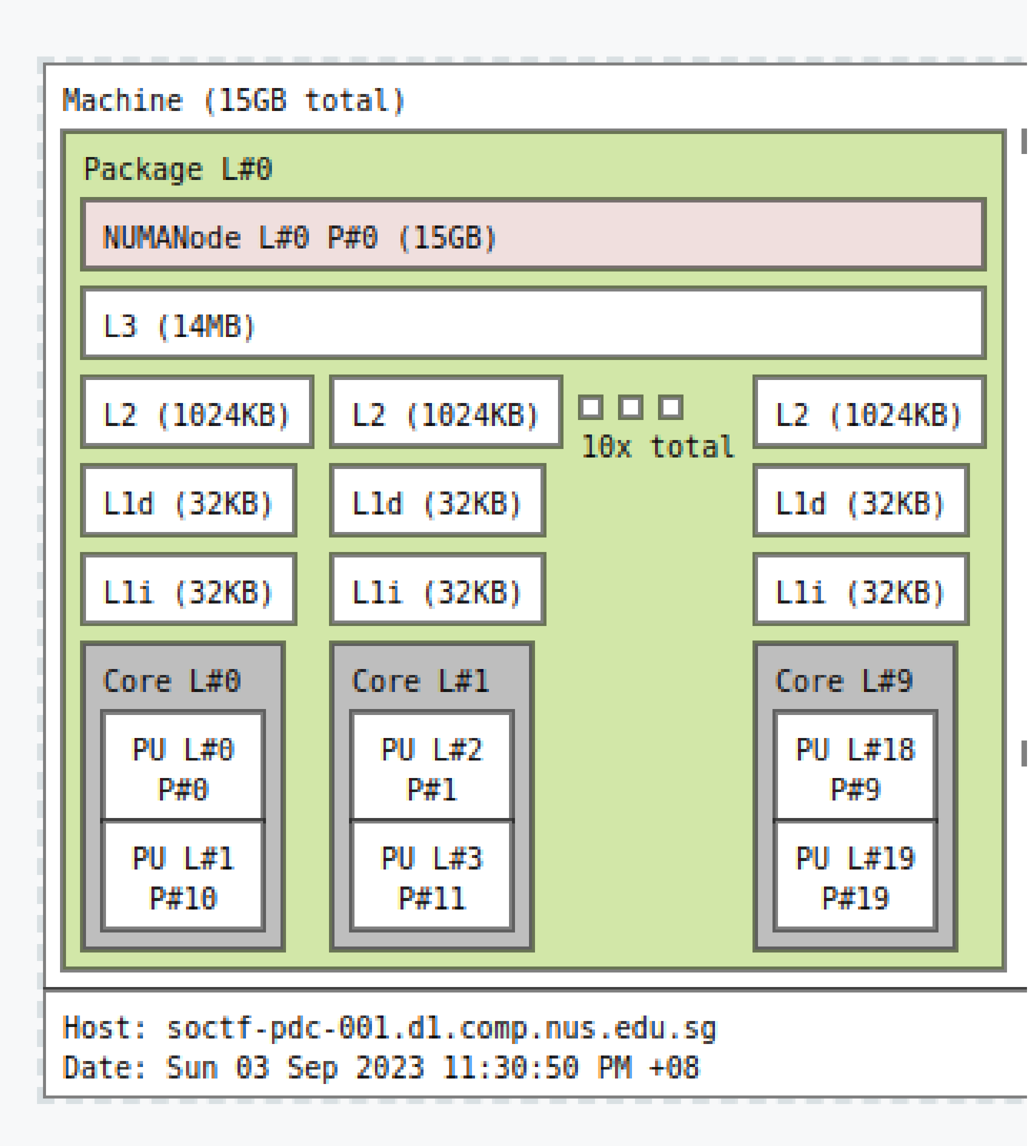
Socket vs Core vs Thread..

- **Thread**: single hardware thread of execution
- **Core:** single set of usable hardware for running code (registers, pipeline stages, some cache)
    - **Cores * threads per core == lscpu CPUs → logical CPUs**
- **Socket:** where a full physical process fits (many cores)


Socket
Receptacle on the motherboard for one physically packaged processor (each of which can contain one or more cores).
CORE 010
CORE 011
CORE 000
CORE 001
Core
A complete private set of registers, execution units, and retirement queues needed to execute programs.
Threads
One or more hardware contexts within a single core. Each thread has attributes of one core; managed & scheduled as a single logical processor by the OS.

Understanding lstopo

- `lstopo --of svg`



Machine (15GB total)

Package L#0

NUMANode L#0 P#0 (15GB)

L3 (14MB)

L2 (1024KB) | L2 (1024KB) | 10x total | L2 (1024KB)

L1d (32KB) | L1d (32KB) | L1d (32KB)

L1i (32KB) | L1i (32KB) | L1i (32KB)

Core L#0: PU L#0 P#0, PU L#1 P#10

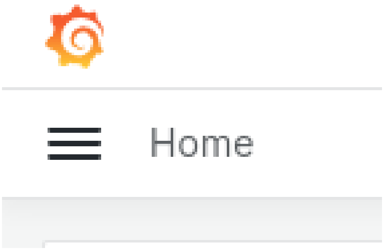
Core L#1: PU L#2 P#1, PU L#3 P#11

Core L#9: PU L#18 P#9, PU L#19 P#19

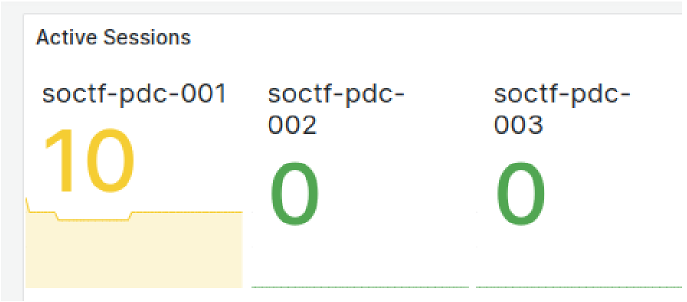
Host: soctf-pdc-001.dl.comp.nus.edu.sg
Date: Sun 03 Sep 2023 11:30:50 PM +08

ex2: Lab Monitoring

- Go to https://pdc.comp.nus.edu.sg/grafana
- Login with your lab username and password
- Click the “hamburger” menu on the top left side



- Check out Home and Dashboards
  - Home - Overview of all nodes, including **number of users logged in**
    - **“Sessions”**
    - **Login to nodes with fewer active sessions!**
  - Dashboards - detailed “Node Exporter” dashboard


Active Sessions

soctf-pdc-001 10

soctf-pdc-002 0

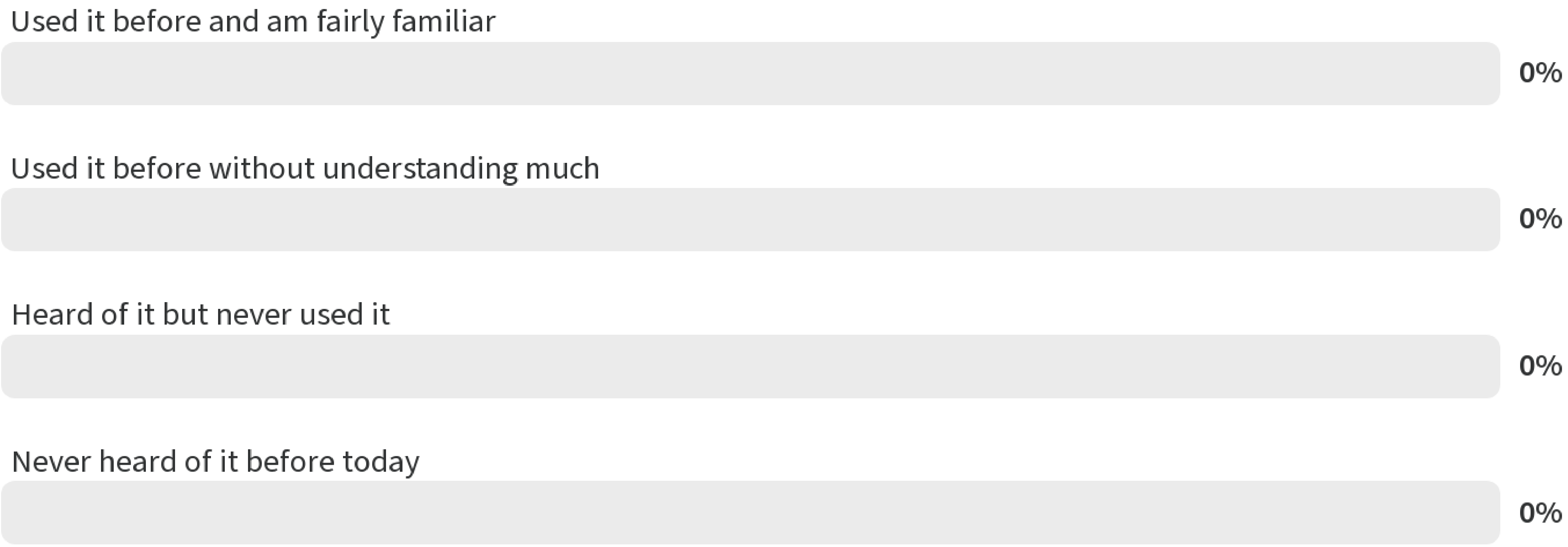
soctf-pdc-003 0

Introducing:
Slurm Workload Manager

How you’re going to run most of your stuff in the lab machines

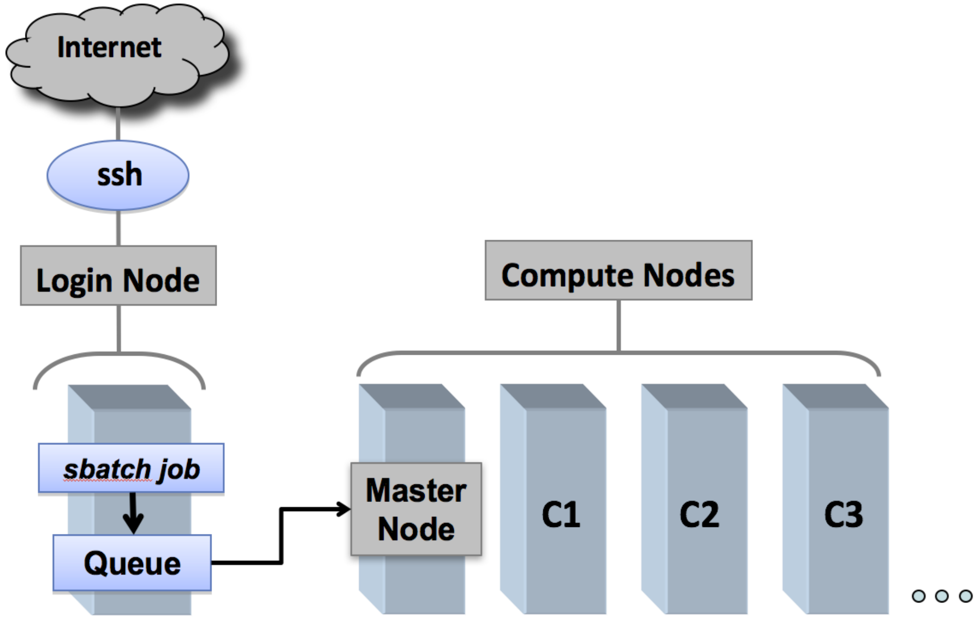
What is your familiarity with Slurm?

✓ 0



What is Slurm / Why Slurm?

- **Job allocation system**
- **Fair allocation** of compute resources
- **Exclusive access** to nodes for **performance measurements**
- Wide variety of machines to test on and **run distributed programs** on



Why use Slurm in CS3210?

- SoC uses Slurm for our entire compute cluster
- The best parallel computing systems use Slurm (>60% of TOP500)
- A way to get good performance measurements when **#hardware << #users** → good for CS3210!

- Do not run long jobs on login nodes anymore!

Disclaimers

- **Please let us know if there are any issues**
  - Contact is in the tutorial sheet; it's Sriram and Peigeng
  - If you contact me…
    my help is probably relaying to them unlesss it's node down
- **Our "best practices" and policies may change.**
  Please refer to https://bit.ly/cs3210-student-guide for updates

Go forth and try!

It’s not a race	Be curious	Screw things up

- ***ex3*** **-** ***ex12***: Small guided *exercises* working through the basics
- ***ex13***: Basic performance *evaluation* with Slurm

Please bring up any interesting observations!

For me:

```
watch -n 0.5 squeue
watch -n 0.5 sprio
```

Special Topics in Slurm

Network Filesystem (NFS)

Special Topics in Slurm: NFS

It’s not a race **Be curious** **Screw things up**

- Your home directory is configured in a network filesystem
  - Try: `pwd` (print working directory)
- Pro: you get to access it from every node
- Pro: as we have 6 nodes, one goes down your file is safe :D
- Cons: might be slower

- **How much slower is NFS than node-local storage?**

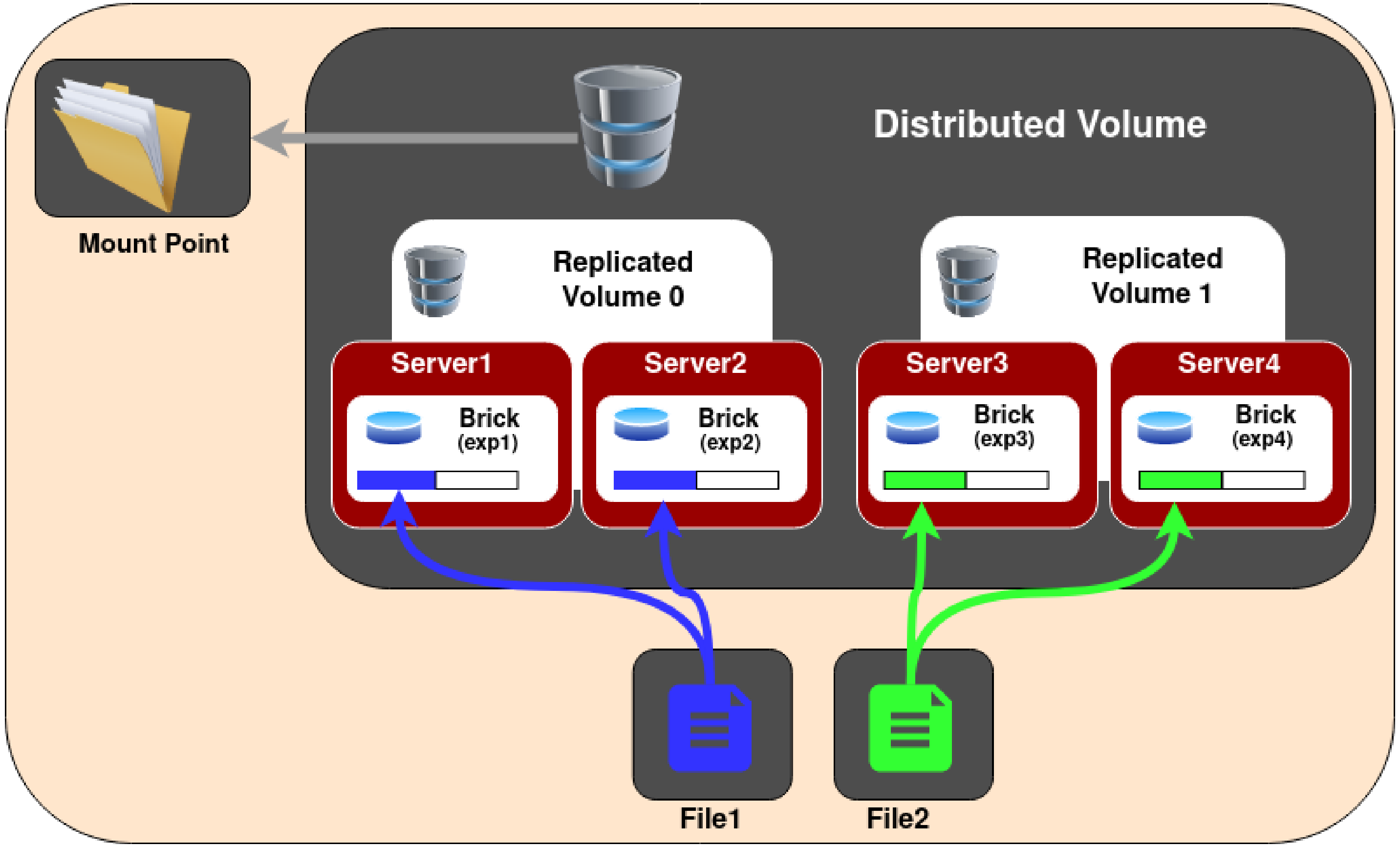
```
dd if=/dev/zero of=/nfs/home/theo/test.img bs=100MB count=1 oflag=dsync
dd if=/dev/zero of=/tmp/test.img bs=100MB count=1 oflag=dsync
```

Special Topics in Slurm: NFS

It's not a race	Be curious	Screw things up

- We run a **distributed + replicated** NFS with "GlusterFS"

- Many other alternatives! Go look it up :D



Special Topics in Slurm

Priority and Share

Special Topics: Priority and Share

It’s not a race **Be curious** **Screw things up**

- Everyone run together: `srun -w soctf-pdc-005 sleep 10`
- `sprio -l`
  - Who has more priority?
- `sshare –A students -a`
  - Who’s been using the most?

Special Topics in Slurm

Slurm’s own broad runtime statistics

Special Topics: Broad runtime statistics

It’s not a race **Be curious** **Screw things up**

- You can find *some* information about running job statistics with `sstat`
  - ○ `sbatch cond.sh`
  - ○ `sstat -o jobid,nodelist,ntasks,avecpu,averss,maxrss,maxdiskread,maxdiskwrite <jobid>`

- But clearly better to use your own performance tracking for single jobs (perf, etc)

Special Topics II

Sneak Peek at Performance Evaluation

Sneak Peek at Performance Evaluation

It’s not a race **Be curious** **Screw things up**

- Let’s try to measure how long a program takes
    - `srun -p xs-4114 time ./pthread_addsub`
    - `srun -p xs-4114 /usr/bin/time -vvv ./pthread_addsub`
    - `srun -p xs-4114 perf stat ./pthread_addsub`
    - `srun -p xs-4114 perf stat -r 3 ./pthread_addsub`
    - `srun -p xs-4114 hyperfine -M 5 ./pthread_addsub`

Sneak Peek at Performance Evaluation

It’s not a race **Be curious** **Screw things up**

- Are there hardware differences?

```
srun -p xs-4114 hyperfine ./pthread_addsub > xs4114.out
srun -p dxs-4114 hyperfine ./pthread_addsub > dxs4114.out
srun -p i7-7700 hyperfine ./pthread_addsub > i77700.out
srun -p i7-9700 hyperfine ./pthread_addsub > i79700.out
srun -p xw-2245 hyperfine ./pthread_addsub > xw2245.out
```

- `tail -n +1 *.out` (to see result)

Results

4C/8T	i7-7700 CPU	@ 4.20GHz: **4.054s**
8C/8T	i7-9700 CPU	@ 4.70GHz: **5.416s (?)**
10C/20T	Xeon Silver 4114	@ 3.00GHz: **6.266s**
2x10C/20T	Dual-Socket XS4114	@ 3.00GHz: **5.082s**
8C/16T	Xeon W-2245	@ 4.50GHz: **8.541s (???)**
12C/24T	Xeon w5-3245	@ 4.60GHz: **2.502s**
16C/24T	i7-13700	@ 5.20GHz(P), 4.10GHz(E): **1.500s**

Admin & Feedback

- 100% anonymous, anytime feedback at last slide

- If you haven’t submitted Lab 1 - you still can!
  Just submit your best attempt :)

Summary

- Parallel architectures
- Flynn’s taxonomy
- Memory, Processes vs threads
- Slurm usage

End of tutorial 1

- If you haven't joined the telegram group: you're missing out

- Slides uploaded!
- Feedback: bit.ly/feedback-theodore or scan below
- Email: theo@comp.nus.edu.sg