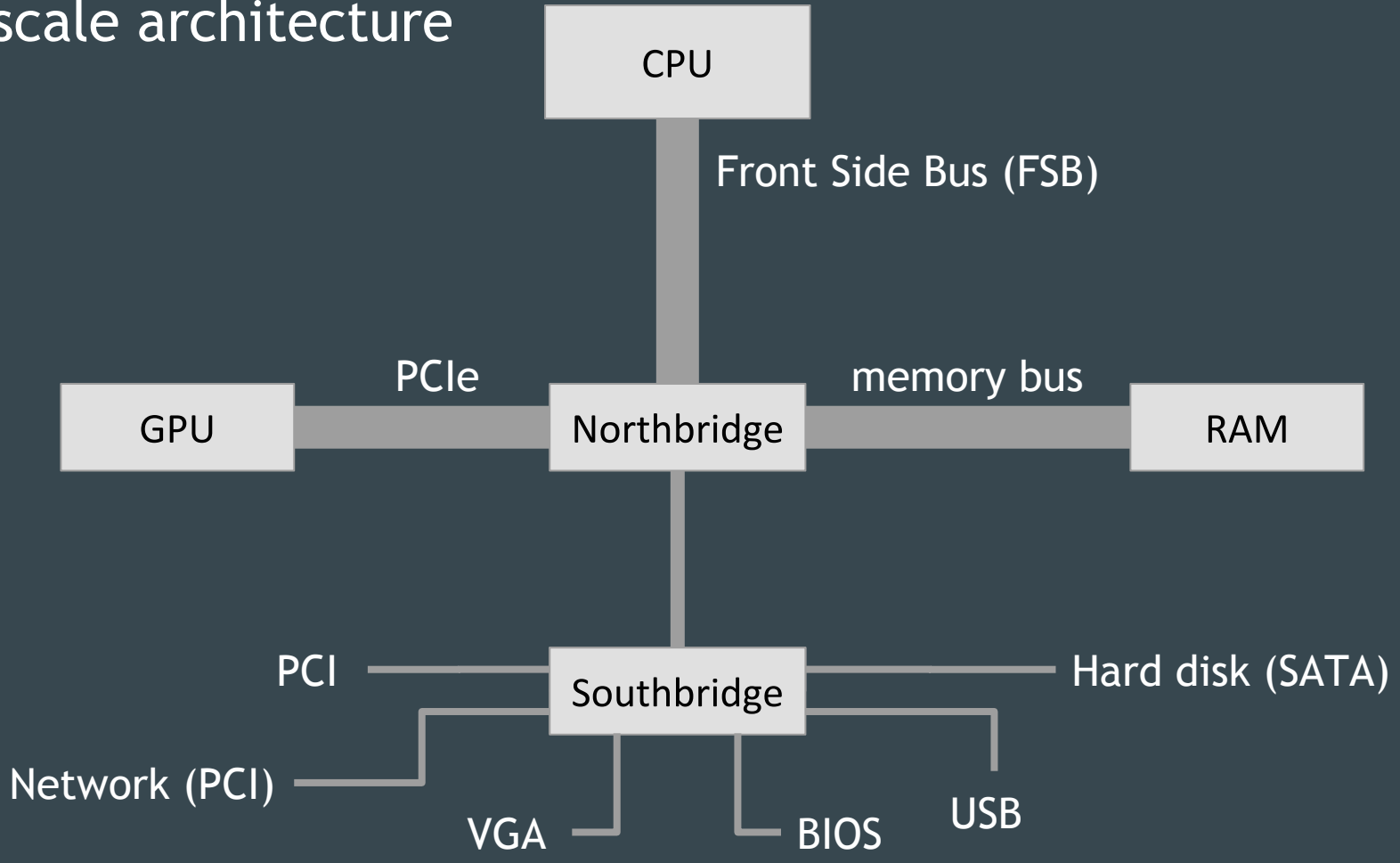
Astro 528
Week 4: Computer Architectures

Guest Lecture by Daniel Carrera

Large scale architecture



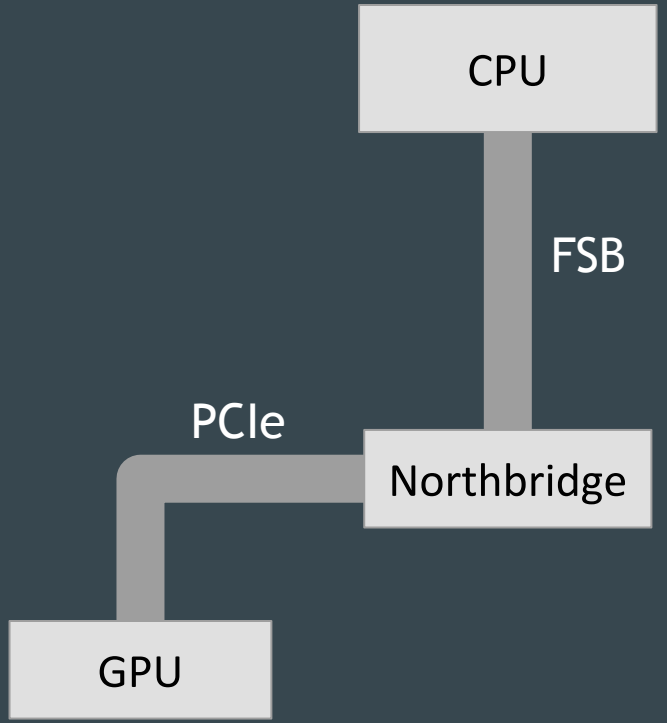
Large scale architecture: bus

- A "*bus*" is just a circuit to transfer data between components.

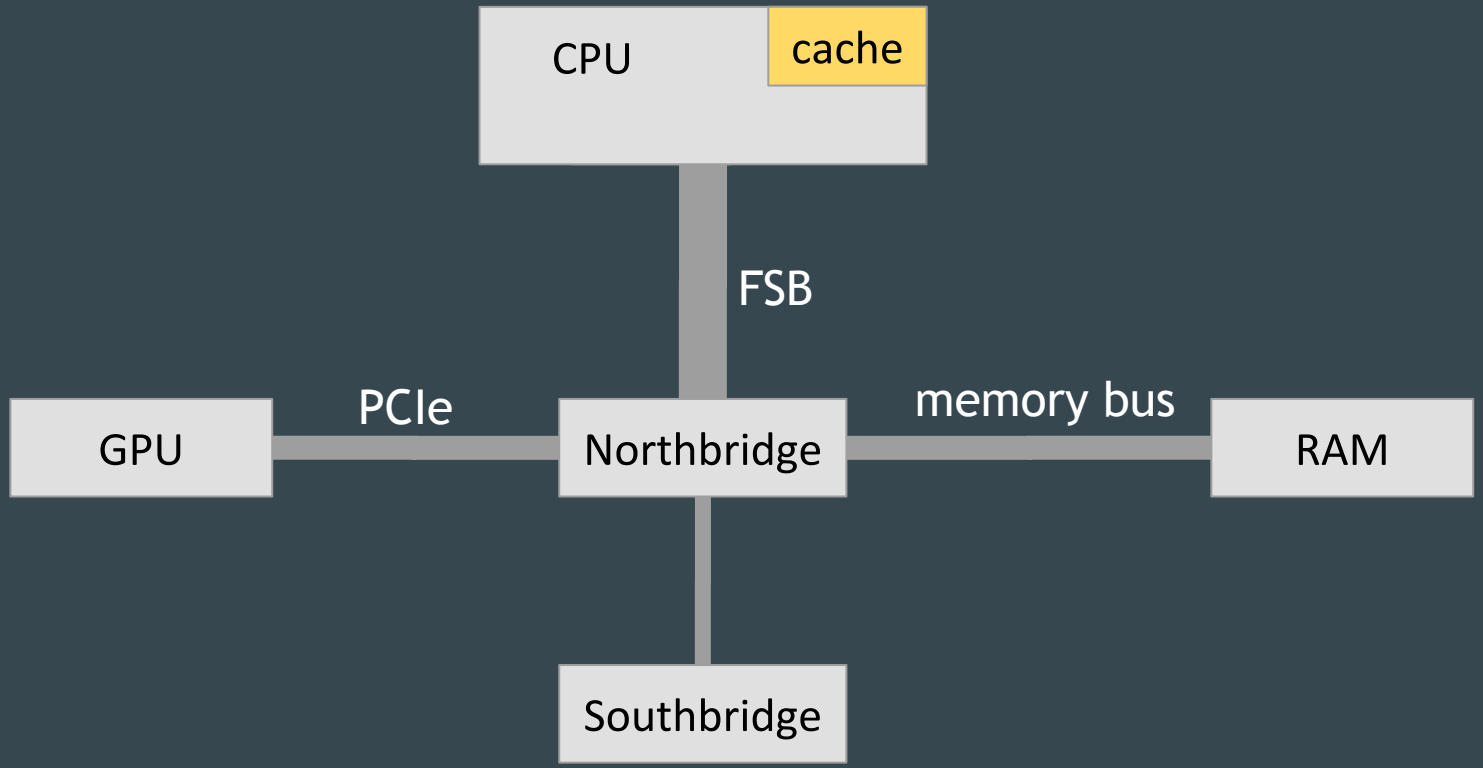
Ex:

USB	Universal Serial Bus
SATA	Serial ATA
PCI	Peripheral Component Interconnect
PCIe	PCI-Express
FSB	Front Side Bus

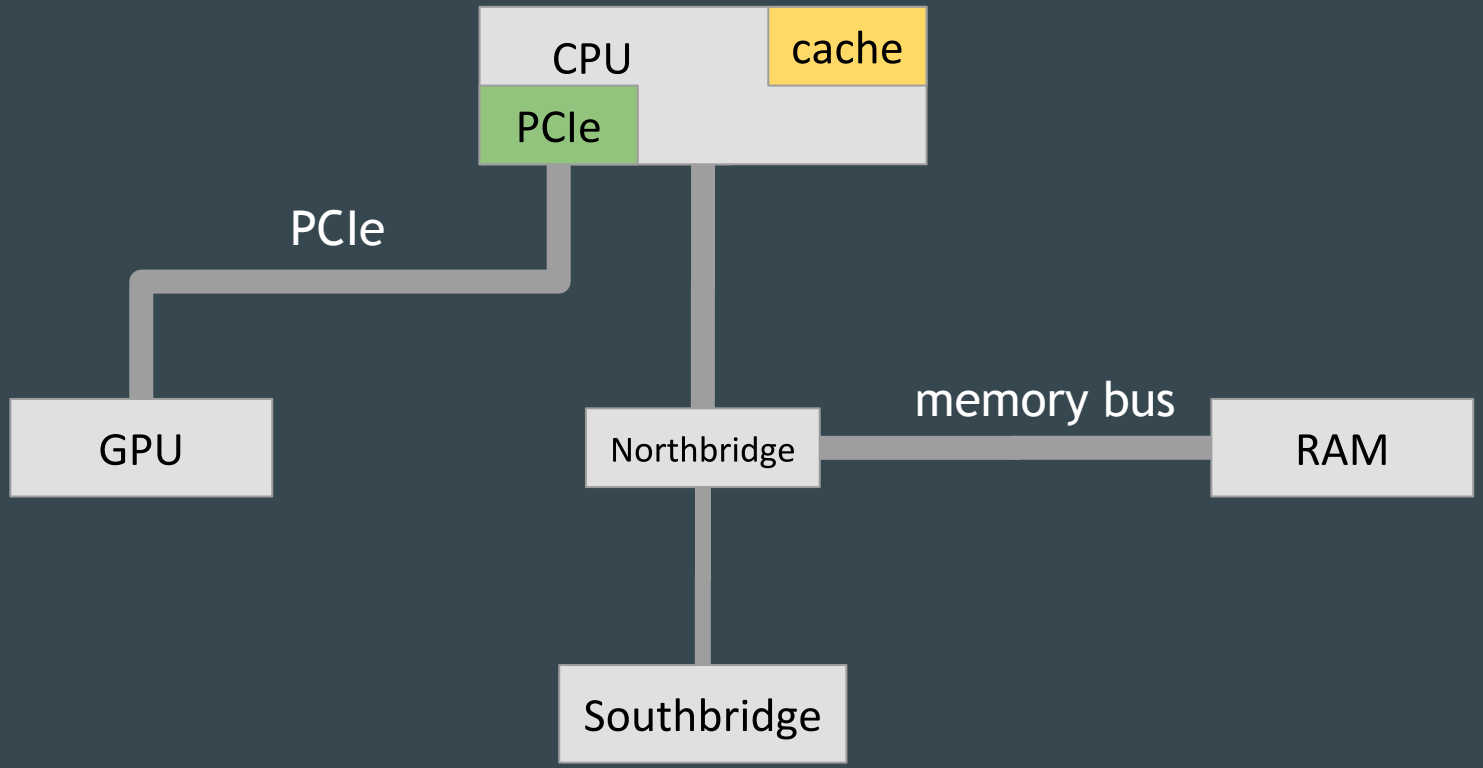
- All buses are much slower than the CPU.
- A lot of hardware and software design choices are entirely about avoiding the FSB.



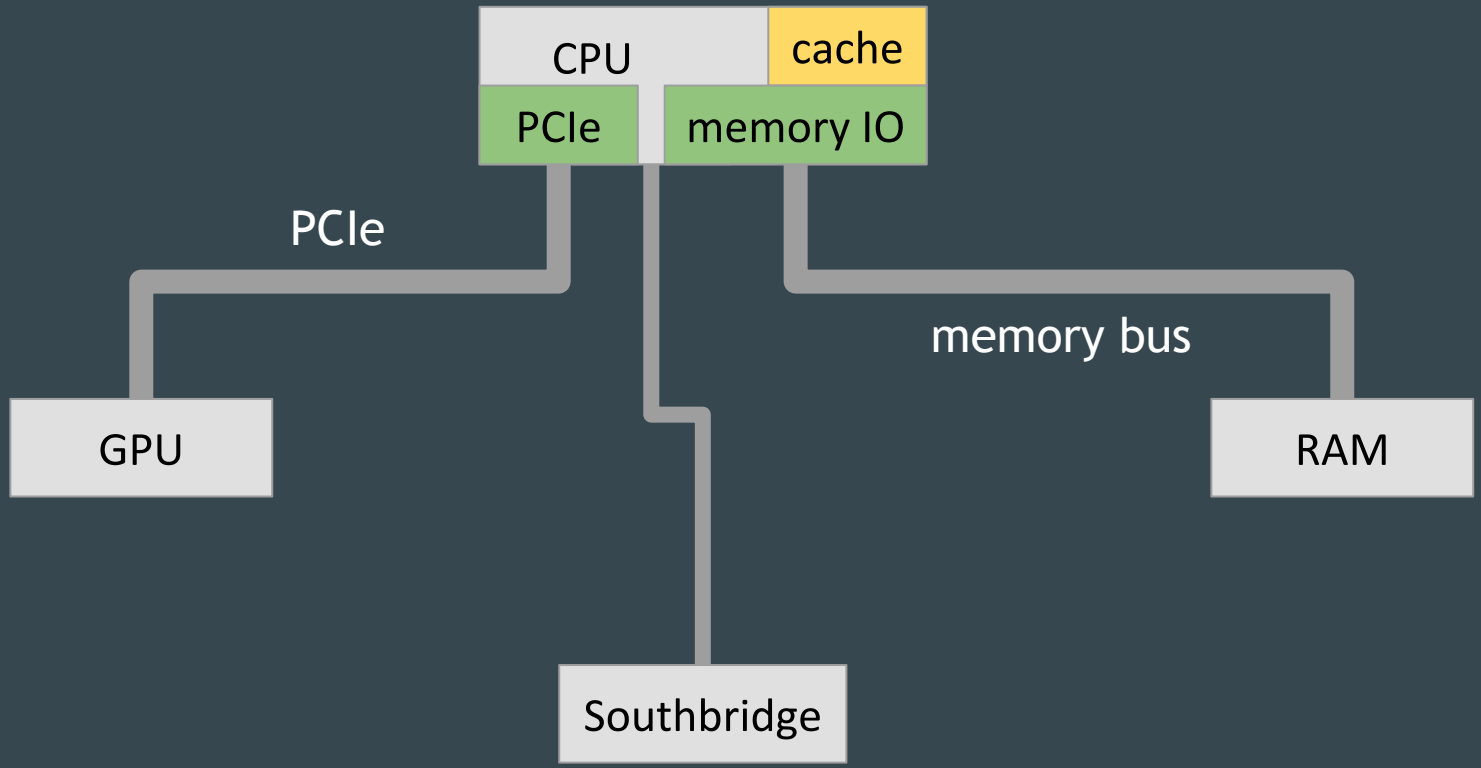
Large scale architecture: avoiding the FSB



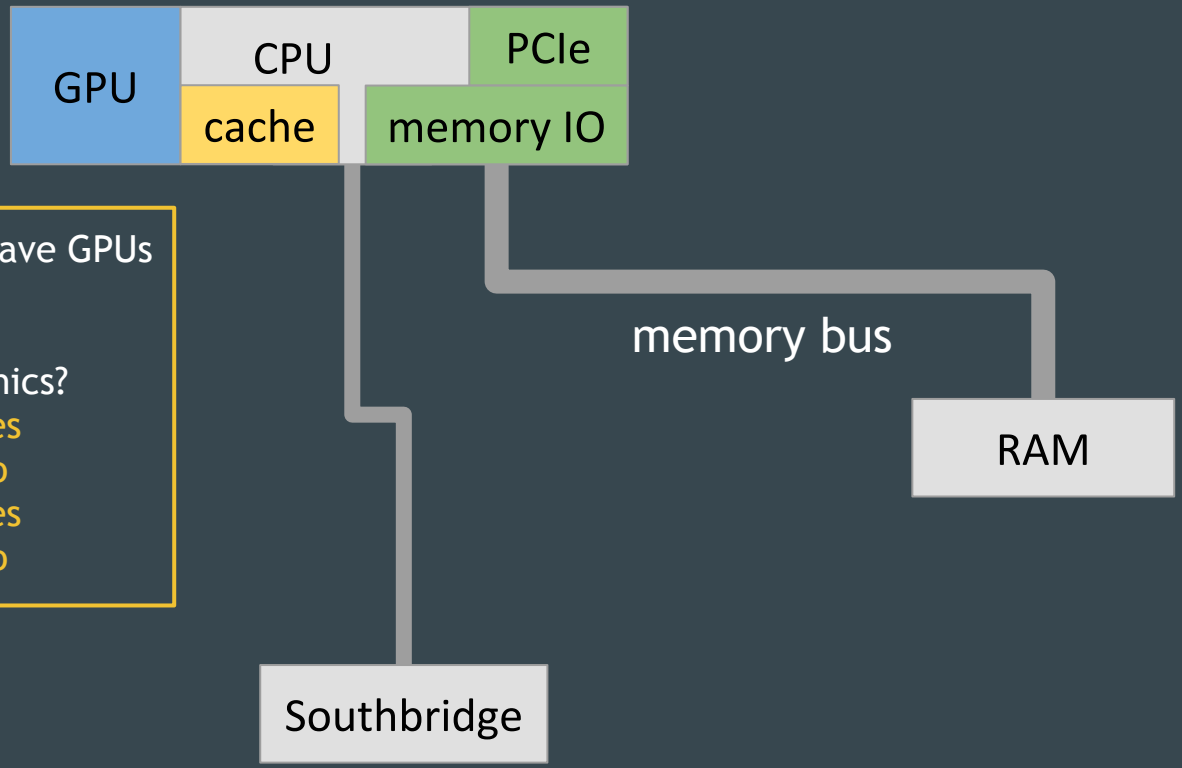
Large scale architecture: avoiding the FSB



Large scale architecture: avoiding the FSB



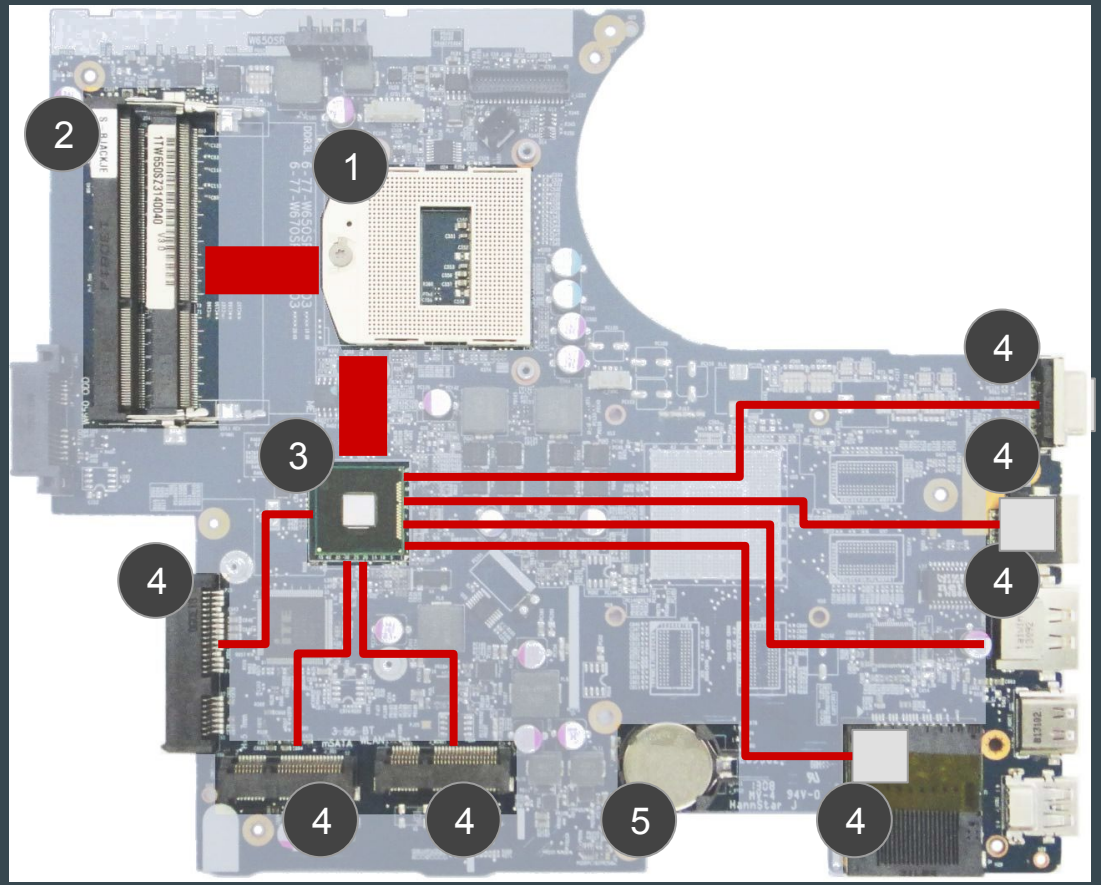
Large scale architecture: avoiding the FSB



Note: Some CPUs have GPUs and some don't.

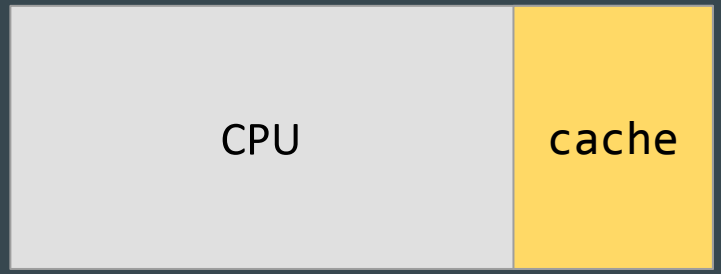
	graphics?
Intel Core	yes
Intel Xeon	no
AMD APU	yes
AMD Ryzen	no

Large scale architecture: sample motherboard

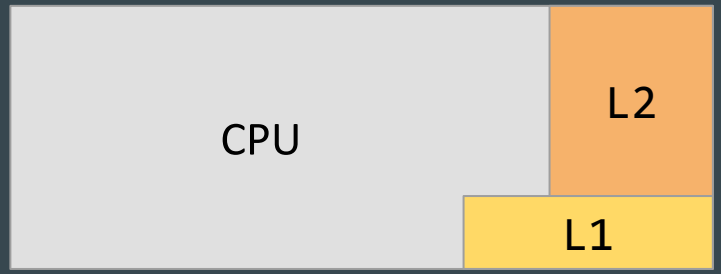


1 CPU + northbridge
2 RAM
3 Southbridge
4 IO ports
5 CMOS battery

CPU architecture



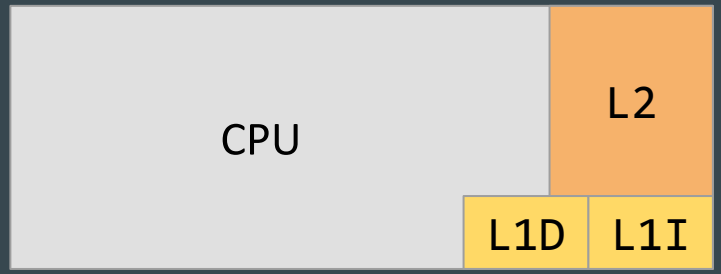
CPU architecture: cache levels



L1 Designed for speed at the expense of memory density.

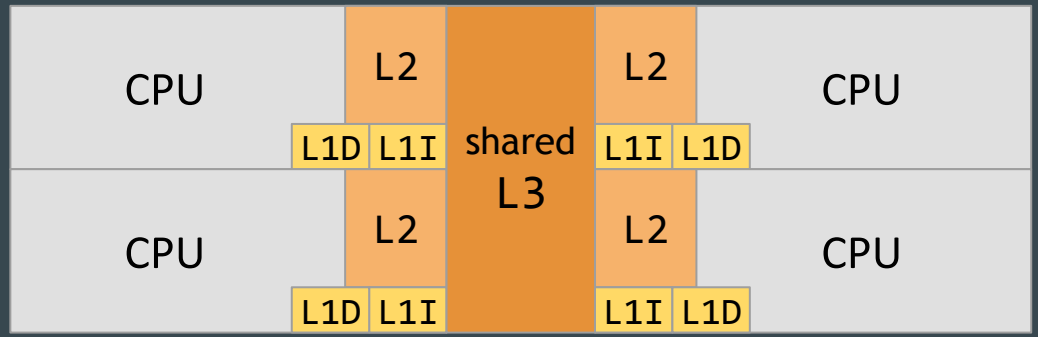
L2 (the opposite)

CPU architecture: cache levels



		Example (Xeon E5-2650)
L1I	L1 instruction cache	32 kB
L1D	L1 data cache	32 kB
L2	instruction + data	256 kB

CPU architecture: multi-core CPUs



L1I	per core
L1D	per core
L2	per core
L3	shared between cores

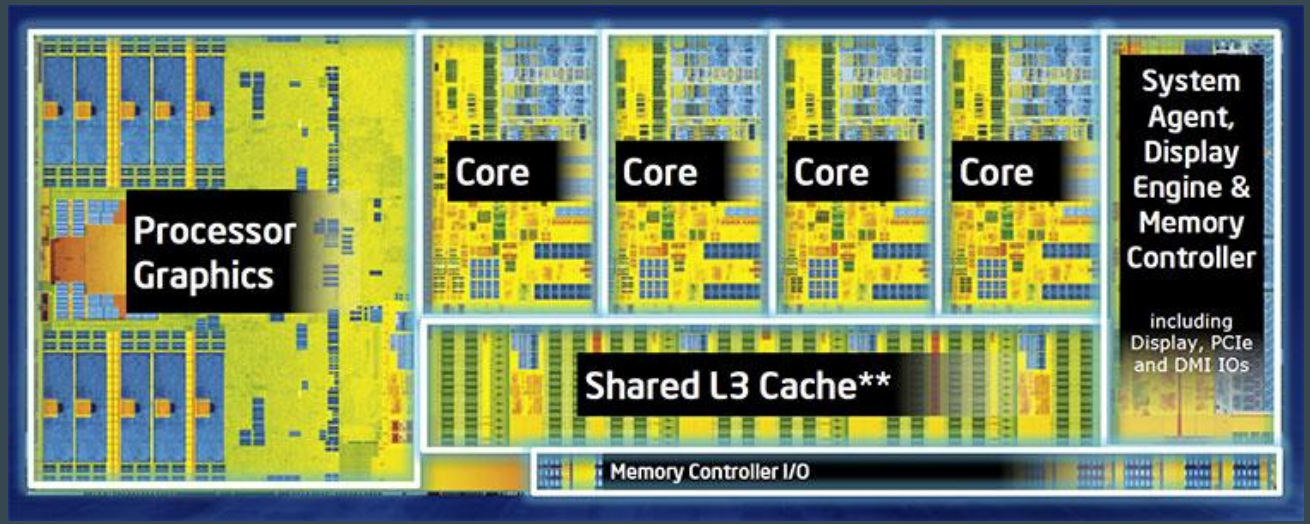
Q: Why is this important?

Example (Xeon E5-2650)

```
 32 kB / core
 32 kB / core
256 kB / core
30 MB shared / 12 cores
```

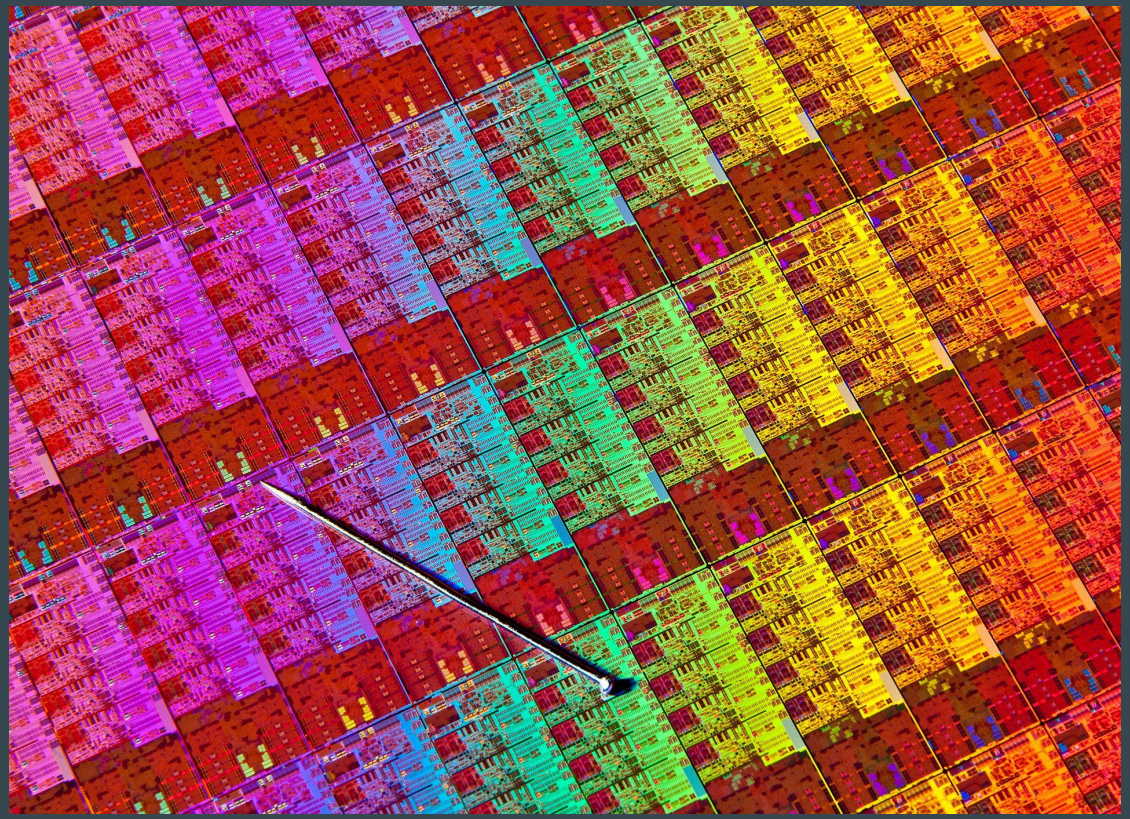
CPU architecture: examples

Haswell CPU (Intel, 2013, 22nm process)



CPU architecture: examples

Haswell wafer (Intel, 2013, 22nm process)



CPU architecture: the core

Skylake core (Intel, 2015, 14nm process)

CPU architecture: pre-fetching

Skylake core

The CPU pre-loads cache with data and instructions that it thinks is most likely to be needed soon.

CPU architecture: registers

Skylake core

Registers are part of the execution unit.

Instructions and data must be copied to a register before the CPU can act, and any result must be copied back to cache.

x86-64 has just 16 registers (64 bits each).

CPU architecture: pipelining

Even the simplest operation requires at least 4 steps (fetch from cache, decode, execute, write back to cache). ***Pipelining*** keeps all the CPU components busy.

Without Pipelining

Fetch	A				B			
Decode		A				B		
Execute			A				B	
Write				A				B

With Pipelining

Fetch	A	B						
Decode		A	B					
Execute			A	B				
Write				A	B			

CPU architecture: branch prediction

Skylake core

When encountering a conditional:

```
statement 1
statement 2
if cond
    statement 3
else
    statement 4
```

The processor will guess which branch is most likely. It then fetches the data and ***speculatively executes*** the instruction (i.e. puts it in the pipeline).

CPU architecture: out of order execution

Skylake core

Pipelining requires knowledge of data dependencies, so it relies on OoO.

OoO = out of order execution

Processor executes instructions when data becomes available, rather than in their original order in the program.

Ex:

```
1.  a = b + c
2.  h = e + g
3.  d = a + c
```

Lines 2 and 3 can be done in any order.

CPU architecture: cache misses

- AMD and Intel spend a lot of effort in trying to minimize ***cache misses***.
- A ***cache miss*** is when the data you need is not in cache (esp. L1D).
- Both companies routinely achieve L1 ***cache hit*** rates of > 95%.

CPU architecture: the cost of a cache miss

Zen architecture (AMD, 2017, 14 nm process)

	Latency
L1D	~1.0ns
L2	~5.3ns
L3	~21.6ns
RAM	~84.0ns

Assuming a 95% L1 hit rate,

Q: What fraction of the time is spent in cache misses if…

- the other 5% is in L2 ?
- the other 5% is in L3 ?
- the other 5% is in RAM?

(ignore execution time and bandwidth)

CPU architecture: the cost of a cache miss

Zen architecture (AMD, 2017, 14 nm process)

	Latency	Answer
L1D	~1.0ns	
L2	~5.3ns	22%
L3	~21.6ns	53%
RAM	~84.0ns	82%

Assuming a 95% L1 hit rate,

Q: What fraction of the time is spent in cache misses if…

- the other 5% is in L2 ?
- the other 5% is in L3 ?
- the other 5% is in RAM?

(ignore execution time and bandwidth)

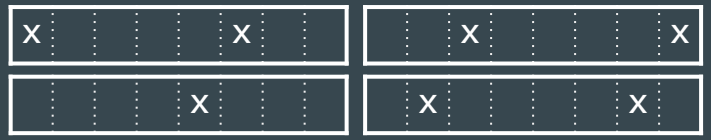
Software design: cache lines

- Data moves across cache in units of 64 bytes (usually), or 8 x Float64. This is called a ***cache line***.

- Design your code to access data sequentially inside a cache line to minimize cache misses.

```
GOOD:   for j in 1:7
            … = … + x[j]
```


```
BAD:    for j in 1:5:31
            … = … + x[j]
```



Software design: cache lines

	Xeon E5-2650
L1D	32 kB / core
L2	256 kB / core
L3	30 MB shared

Exercise 3 from last Thursday (matrix multiply: A * b)

N	Storage	column_inner	row_inner	Ratio
64	33 kB	4.564 µs	4.146 µs	1.1
128	129 kB	22.037 µs	14.847 µs	1.5
256	514 kB	114.570 µs	57.350 µs	2.0
512	2 MB	448.123 µs	229.487 µs	2.0
1024	8 MB	1.8 ms	885.461 µs	2.0
2048	32 MB	35.9 ms	3.8 ms	9.4

Q: Why does the ratio suddenly jump at the last line? (speculate)

Q: Why is Julia's implementation faster than row_inner? (speculate)

Software design: block matrix multiply - why Julia is fast

Software design: block matrix multiply - why Julia is fast