The New Golden Age of Computer Architecture

Hung-Wei Tseng

Recap: 4Cs of cache misses

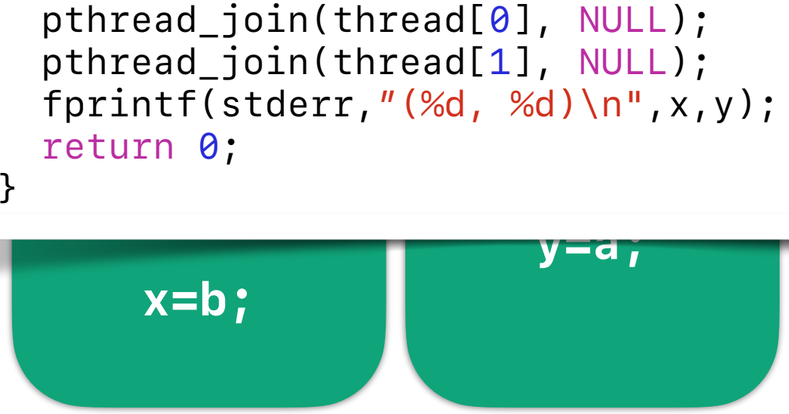
- 3Cs:
  - Compulsory, Conflict, Capacity
- Coherency miss:
  - A “block” invalidated because of the sharing among processors.
- True sharing
  - Processor A modifies X, processor B also want to access X.
- False sharing
  - Processor A modifies X, processor B also want to access Y. However, Y is invalidated because X and Y are in the same block!

Recap: Possible scenarios

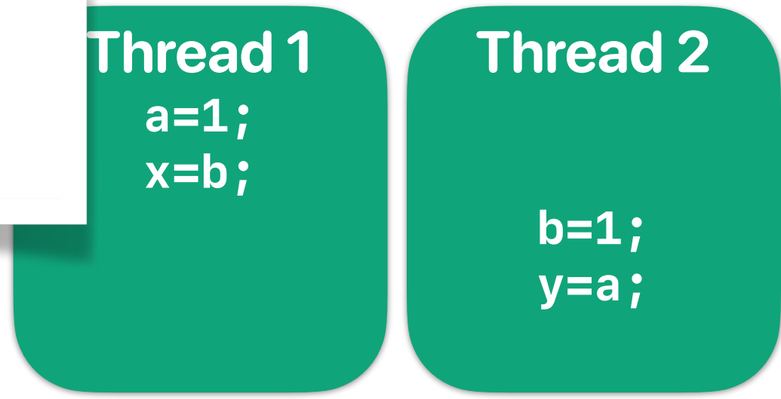
```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

volatile int a,b;
volatile int x,y;
volatile int f;
void* modifya(void *z) {
  a=1;
  x=b;
  return NULL;
}
void* modifyb(void *z) {
  b=1;
  y=a;
  return NULL;
}
```

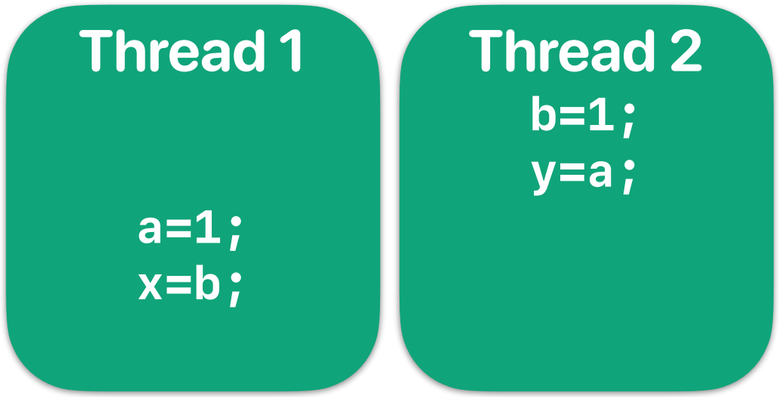
```c
int main() {
  int i;
  pthread_t thread[2];
  pthread_create(&thread[0], NULL, modifya, NULL);
  pthread_create(&thread[1], NULL, modifyb, NULL);
  pthread_join(thread[0], NULL);
  pthread_join(thread[1], NULL);
  fprintf(stderr,"(%d, %d)\n",x,y);
  return 0;
}
```



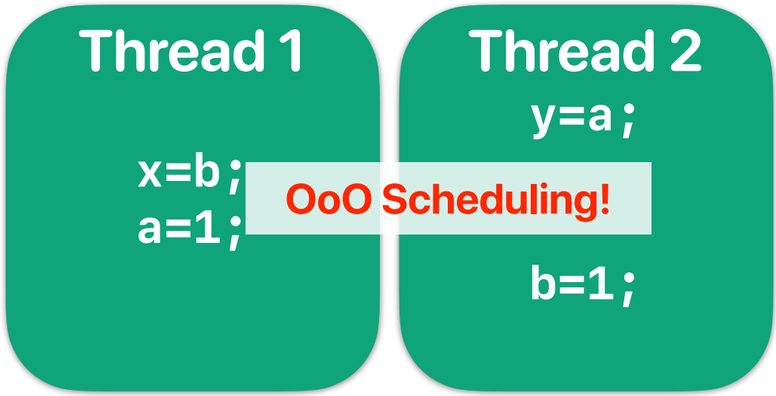
(1,1)



(0,1)

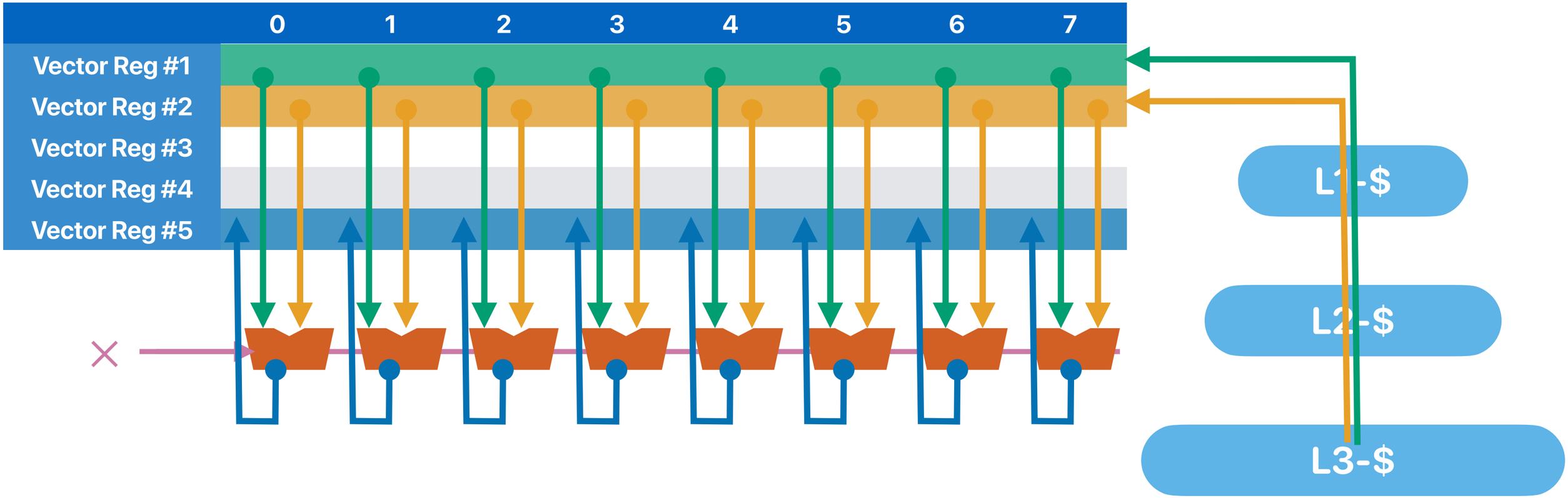


(1,0)



(0,0)

Vector processing architecture



Vectorized matrix multiplications

```
#define VECTOR_WIDTH 4
void vector_blockmm(double **a, double **b, double **c, \
uint64_t tile_size) {
  int i,j,k, ii, jj, kk, x;
  __m256d va, vb, vc;
  for(i = 0; i < ARRAY_SIZE; i+=tile_size) {
    for(j = 0; j < ARRAY_SIZE; j+=tile_size) {
      for(k = 0; k < ARRAY_SIZE; k+=tile_size) {
        for(ii = i; ii < i+tile_size; ii++) {
          for(jj = j; jj < j+tile_size; jj+=VECTOR_WIDTH) {
             vc = _mm256_load_pd(&c[ii][jj]);     // load a vector of 4
             for(kk = k; kk < k+tile_size; kk++) {doubles starting from c[ii][jj] to vc
                va = _mm256_broadcast_sd(&a[ii][kk]); // load a vector of 4 doubles to va
                vb = _mm256_load_pd(&b[kk][jj]);      // load a vector of 4 doubles to vb
                vc = _mm256_add_pd(vc,_mm256_mul_pd(va,vb));  // vc += va * vb
             }
             _mm256_store_pd(&c[ii][jj],vc);
          }                          // store vc to c[ii][jj] to c[ii][jj+3]
        }
      }
    }
  }
}
```

Parallelisms

- Instruction-level parallelism — perform various, independent instructions simultaneously
  - Pipeline
  - OoO/Superscalar
- Data-level parallelism — perform the same operation on multiple data elements in parallel
  - SIMD instructions
  - Compute within an SM in GPUs
- Thread-level parallelism — perform independent computation streams (composed of many instructions or SIMD instructions)
  - Multicore/SMT processors
  - Compute using multiple SMs in GPUs

Power consumption & power density

- $P_{dynamic} \sim \alpha \times C \times V^2 \times f \times N$
  - α: average switches per cycle
  - C: capacitance
  - V: voltage
  - f: frequency, usually linear with V
  - N: the number of transistors
- $P_{leakage} \sim N \times V \times e^{-V_t}$
  - N: number of transistors
  - V: voltage
  - V_t: threshold voltage where transistor conducts (begins to switch)
- Power density:

$$P_{density} = \frac{P}{area}$$

Lower the frequency can reduce active power

Moore's Law allows higher frequencies as transistors are smaller

Moore's Law makes this smaller

Recap: What happens if power doesn't scale with process technologies?

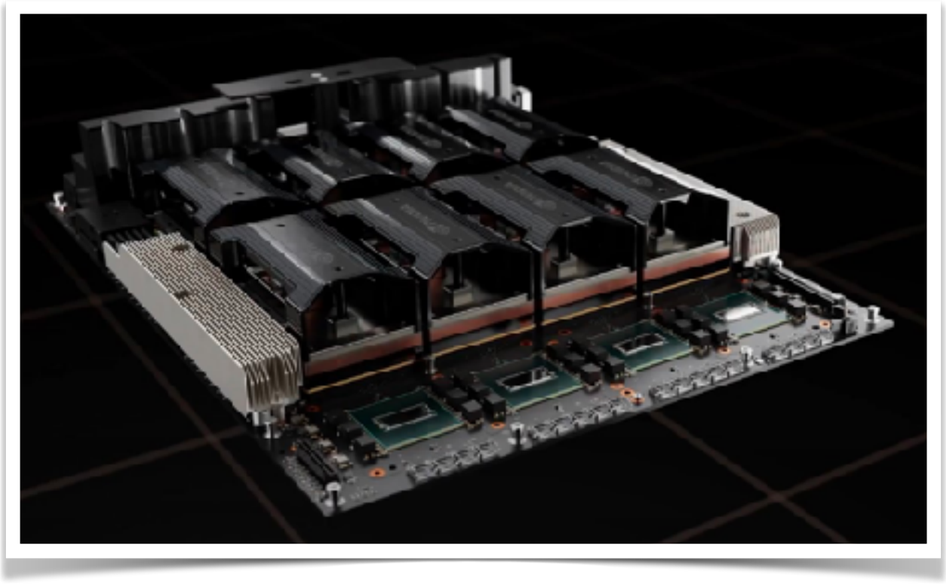
- If we are able to cram more transistors within the same chip area (Moore's law continues), but the power consumption per transistor remains the same. Right now, if put more transistors in the same area because the technology allows us to. How many of the following statements are true?
  - ① The power consumption per chip will increase
  - ② The power density of the chip will increase
  - ③ Given the same power budget, we may not able to power on all chip area if we maintain the same clock rate **— even if we put more cores, we cannot have all of them on!**
  - ④ Given the same power budget, we may have to lower the clock rate of circuits to power on all chip area **— or we have to operate all of them at a slower speed**

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

If you can add power budget…

NVIDIA Accelerator Specification Comparison			
	H100	A100 (80GB)	V100
FP32 CUDA Cores	16896	6912	5120
Tensor Cores	528	432	640
Boost Clock	~1.78GHz (Not Finalized)	1.41GHz	1.53GHz
Memory Clock	4.8Gbps HBM3	3.2Gbps HBM2e	1.75Gbps HBM2
Memory Bus Width	5120-bit	5120-bit	4096-bit
Memory Bandwidth	3TB/sec	2TB/sec	900GB/sec
VRAM	80GB	80GB	16GB/32GB
FP32 Vector	60 TFLOPS	19.5 TFLOPS	15.7 TFLOPS
FP64 Vector	30 TFLOPS	9.7 TFLOPS (1/2 FP32 rate)	7.8 TFLOPS (1/2 FP32 rate)
INT8 Tensor	2000 TOPS	624 TOPS	N/A
FP16 Tensor	1000 TFLOPS	312 TFLOPS	125 TFLOPS
TF32 Tensor	500 TFLOPS	156 TFLOPS	N/A
FP64 Tensor	60 TFLOPS	19.5 TFLOPS	N/A
Interconnect	NVLink 4 18 Links (900GB/sec)	NVLink 3 12 Links (600GB/sec)	NVLink 2 6 Links (300GB/sec)
GPU	GH100 (814mm2)	GA100 (826mm2)	GV100 (815mm2)
Transistor Count	80B	54.2B	21.1B
TDP	700W	400W	300W/350W
Manufacturing Process	TSMC 4N	TSMC 7N	TSMC 12nm FFN
Interface	SXM5	SXM4	SXM2/SXM3
Architecture	Hopper	Ampere	Volta

https://www.workstationspecialist.com/product/nvidia-tesla-a100/



https://www.servethehome.com/wp-content/uploads/2022/03/NVIDIA-GTC-2022-H100-in-HGX-H100.jpg

Recap: Power consumption to light on all transistors

Chip

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

=49W

Dennardian Scaling

Chip

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

=50W

Dennardian Broken

Chip

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

On ~ 50W

Off ~ 0W

Dark!

=100W!

Recap — Take-aways: parallel programming

- Cache coherency only guarantees that everyone would eventually have a coherent view of data, but not when
- Cache coherency may create unexpected cache invalidations/misses if you do it wrong
- Processor behaviors are non-deterministic
  - You cannot predict which processor is going faster
  - You cannot predict when OS is going to schedule your thread
  - You cannot predict when the processor is going to schedule an instruction
- Cache consistency is hard to support
- **Even if we can address all programming challenges, multi-core performance has stopped to scale due to the Dark Silicon problem**

Outline

- Challenges and state-of-the-art solutions in the dark silicon era
- The new golden age of Computer Architecture

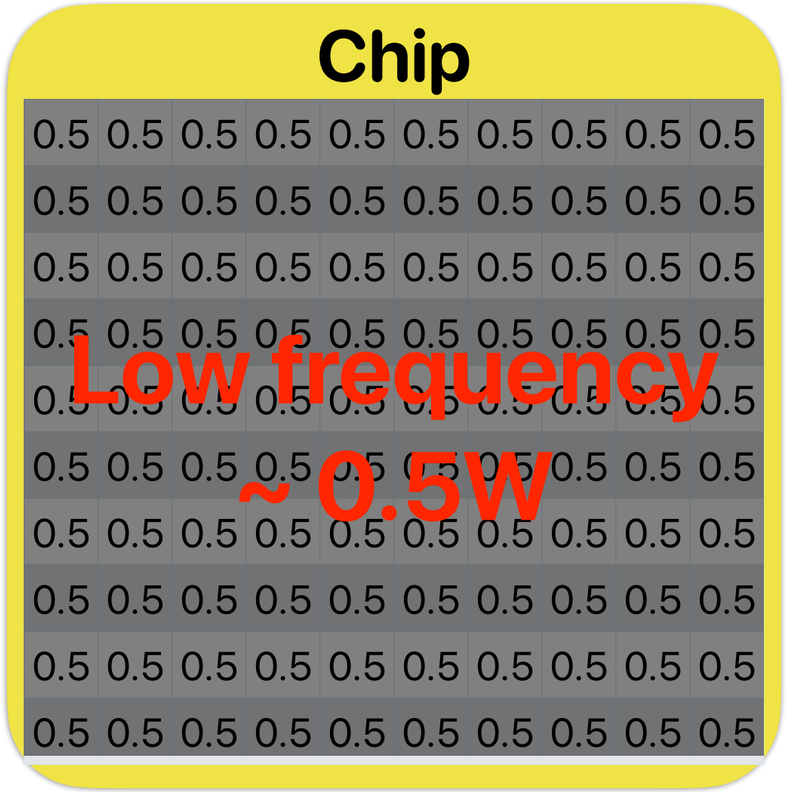
Challenges and state-of-the-art solutions in the dark silicon era

Dark silicon problem

- We are given the same area, twice amount of transistors
- We are given the same power budget
- We are given a certain applications
- How can we maximize the performance for these applications within the given constraints?

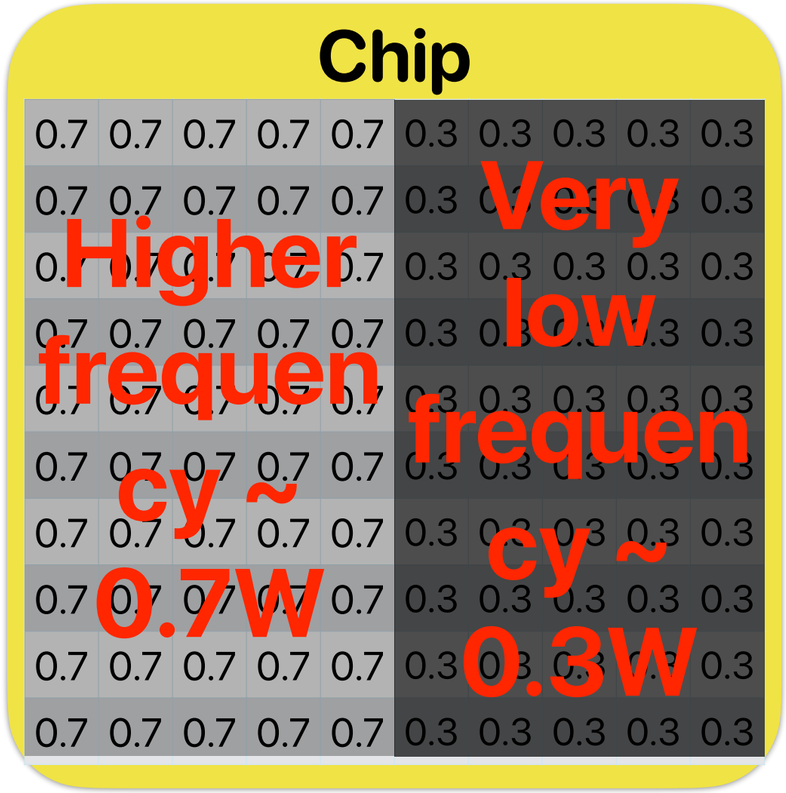
Given the same power budget, maximize the efficiency per chip

Slowdown all of them



=50W!

Some faster, some slower

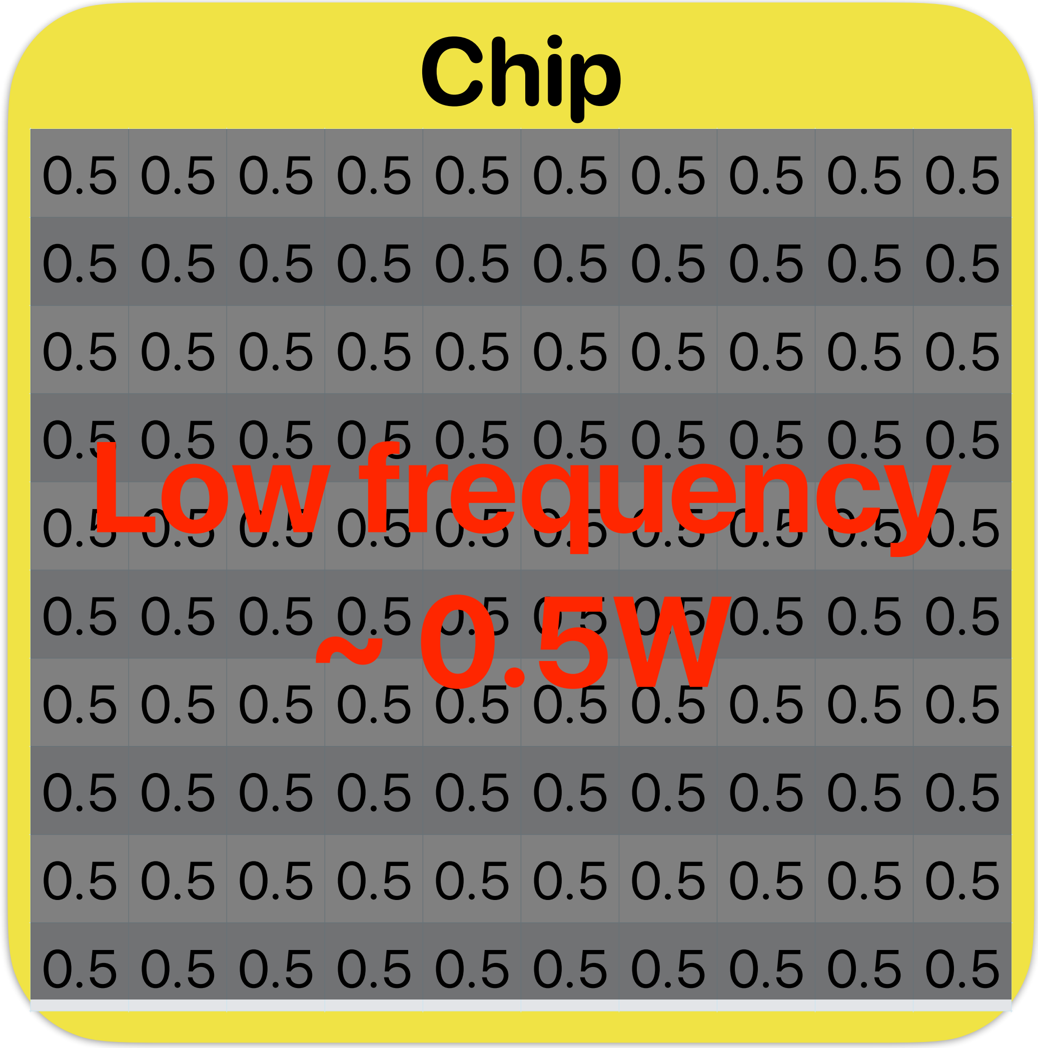


=50W!

Some at top speed, some are not functioning

Given the same power budget, maximize the efficiency per chip

Slowdown all of them



=50W!

More cores per chip, slower per core

Export comparison	Intel® Xeon® Platinum 849...	Intel® Xeon® Platinum 846...	Intel® Xeon® Gold 6448H ...	Intel® Xeon® Platinum 844...	Intel® Xeon® Gold 6434H ...
Total Cores	60	48	32	16	8
Total Threads	120	96	64	32	16
Max Turbo Frequency	3.50 GHz	3.80 GHz	4.10 GHz	4.00 GHz	4.10 GHz
Processor Base Frequency	1.90 GHz	2.10 GHz	2.40 GHz	2.90 GHz	3.70 GHz
Cache	112.5 MB	105 MB	60 MB	45 MB	22.5 MB
Intel® UPI Speed	16 GT/s	16 GT/s	16 GT/s	16 GT/s	16 GT/s
Max # of UPI Links	4	4	3	4	3
TDP	350 W	330 W	250 W	270 W	195 W

https://ark.intel.com/content/www/us/en/ark/products/series/228622/4th-generation-intel-xeon-scalable-processors.html

Xeon Phi

Essentials

Product Collection	Intel® Xeon Phi™ 72x5 Processor Family
Code Name	Products formerly Knights Mill
Vertical Segment	Server
Processor Number	7295
Off Roadmap	No
Status	Launched
Launch Date	Q4'17
Lithography	14 nm

Performance

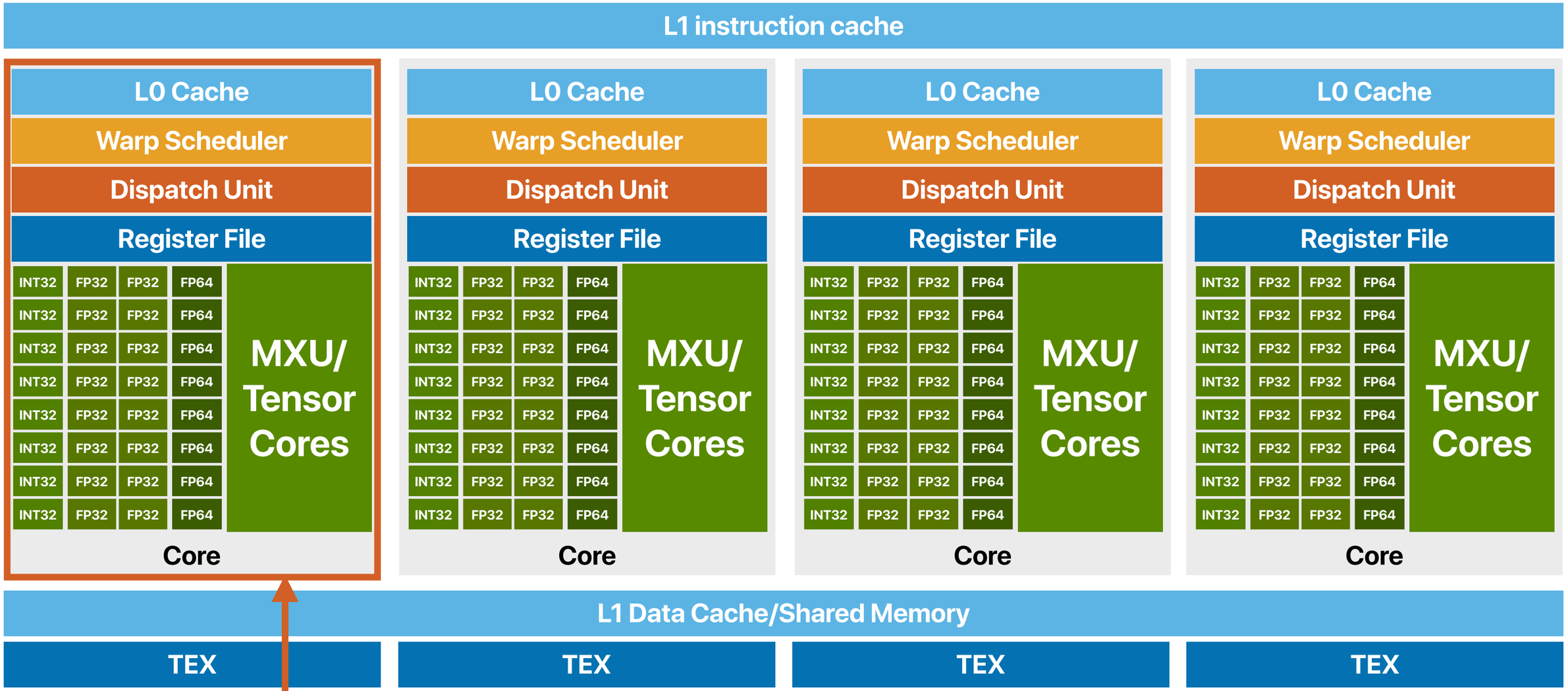
# of Cores	72
# of Threads	72
Processor Base Frequency	1.50 GHz
Max Turbo Frequency	1.50 GHz
Cache	36 MB L2 Cache
TDP	320 W

What's the "appropriate" GPU architecture

- Lots of ALUs to process pixels in parallel — 2M pixels in HD resolution, very regular workloads
  - Vector processing model
- Simple operations
  - The ALUs only supports very few instructions
  - Almost no branches
- Deadline driven and throughput-oriented rather than latency oriented
  - High-bandwidth but also "higher-latency" memory
  - ALUs can be slower

GPU also follows the idea of slower, but more!

GPU Architecture



Recap: Demo — changing the max frequency and performance

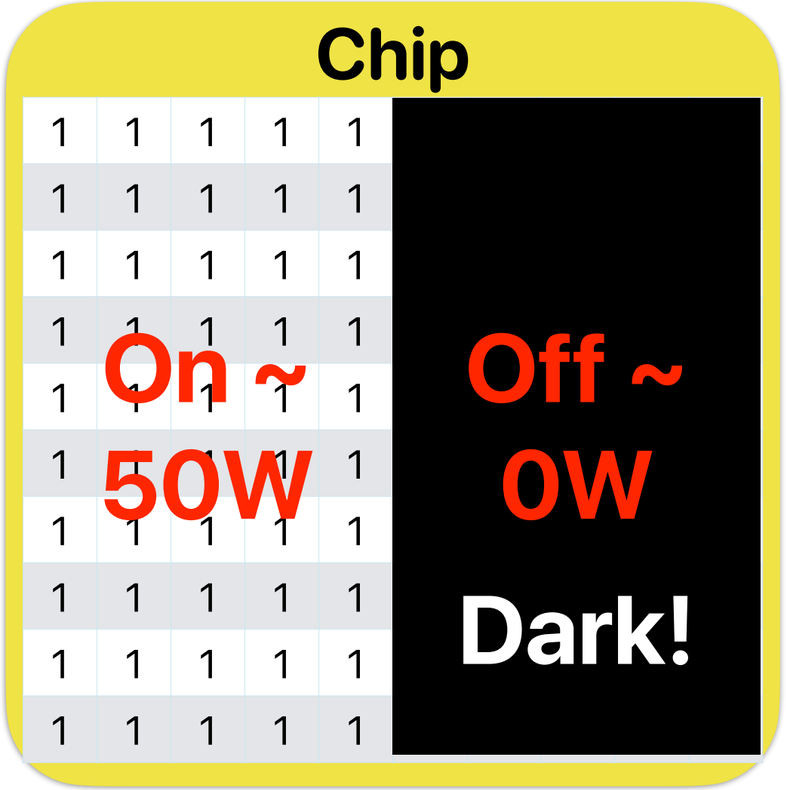
- Change the maximum frequency of the intel processor — you learned how to do this when we discuss programmer's impact on performance
- LIKWID a profiling tool providing power/energy information
  - likwid-perfctr -g ENERGY [command_line]
  - Let's try blockmm and popcount and see what's happening!

Take-aways: the new golden age of computer architectures

- Challenges and SOTA solutions in the dark silicon era
  - GPUs/many-core processors improve the **throughput** per-chip through providing massive parallelism where each processing element operates at a lower speed, but not-ideal for latency-sensitive workloads

Given the same power budget, maximize the efficiency per chip

Some at top speed, some are not functioning



Modern processor's frequency

```
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           151
Model name:                      12th Gen Intel(R) Core(TM) i7-12700KF
Stepping:                        2
CPU MHz:                         1226.409
CPU max MHz:                     5000.0000
CPU min MHz:                     800.0000
BogoMIPS:                        7219.20
```

	i7-12700K
	Intel 7
	$450.00 - $460.00
	PC/Client/Tablet, Workstation

CPU Specifications

Total Cores	12
# of Performance-cores	8
# of Efficient-cores	4
Total Threads	20
Max Turbo Frequency	5.00 GHz
Intel® Turbo Boost Max Technology 3.0 Frequency ‡	5.00 GHz
Performance-core Max Turbo Frequency	4.90 GHz
Efficient-core Max Turbo Frequency	3.80 GHz
Performance-core Base Frequency	3.60 GHz
Efficient-core Base Frequency	2.70 GHz
Cache	25 MB Intel® Smart Cache

https://www.intel.com/content/www/us/en/products/sku/134594/intel-core-i712700k-processor-25m-cache-up-to-5-00-ghz/specifications.html

Modern processor's frequency

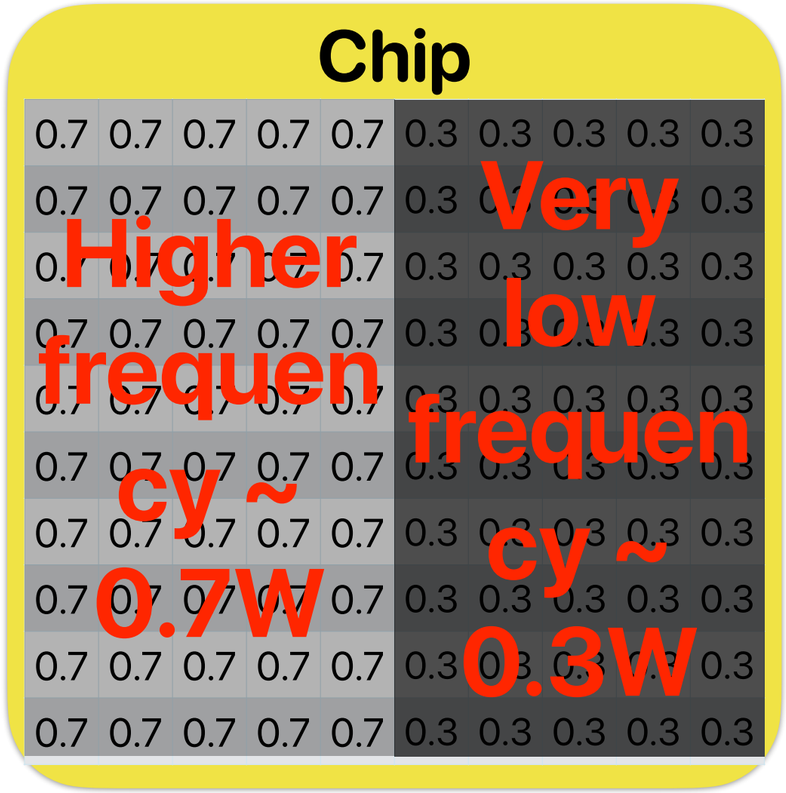
```
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   48 bits physical, 48 bits virtual
CPU(s):                          12
On-line CPU(s) list:             0-11
Thread(s) per core:              2
Core(s) per socket:              6
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       AuthenticAMD
CPU family:                      25
Model:                           80
Model name:                      AMD Ryzen 5 5500
Stepping:                        0
Frequency boost:                 enabled
CPU MHz:                         3600.000
CPU max MHz:                     3600.0000
CPU min MHz:                     1400.0000
```

Take-aways: the new golden age of computer architectures

- Challenges and SOTA solutions in the dark silicon era
  - GPUs/many-core processors improve the **throughput** per-chip through providing massive parallelism where each processing element operates at a lower speed, but not-ideal for latency-sensitive workloads
  - Aggressive dynamic frequency/voltage scaling on CMP to accommodate the demand of **latency**-sensitive, parallelism-limited applications, but the area-efficiency of the slower cores is not great

Given the same power budget, maximize the efficiency per chip

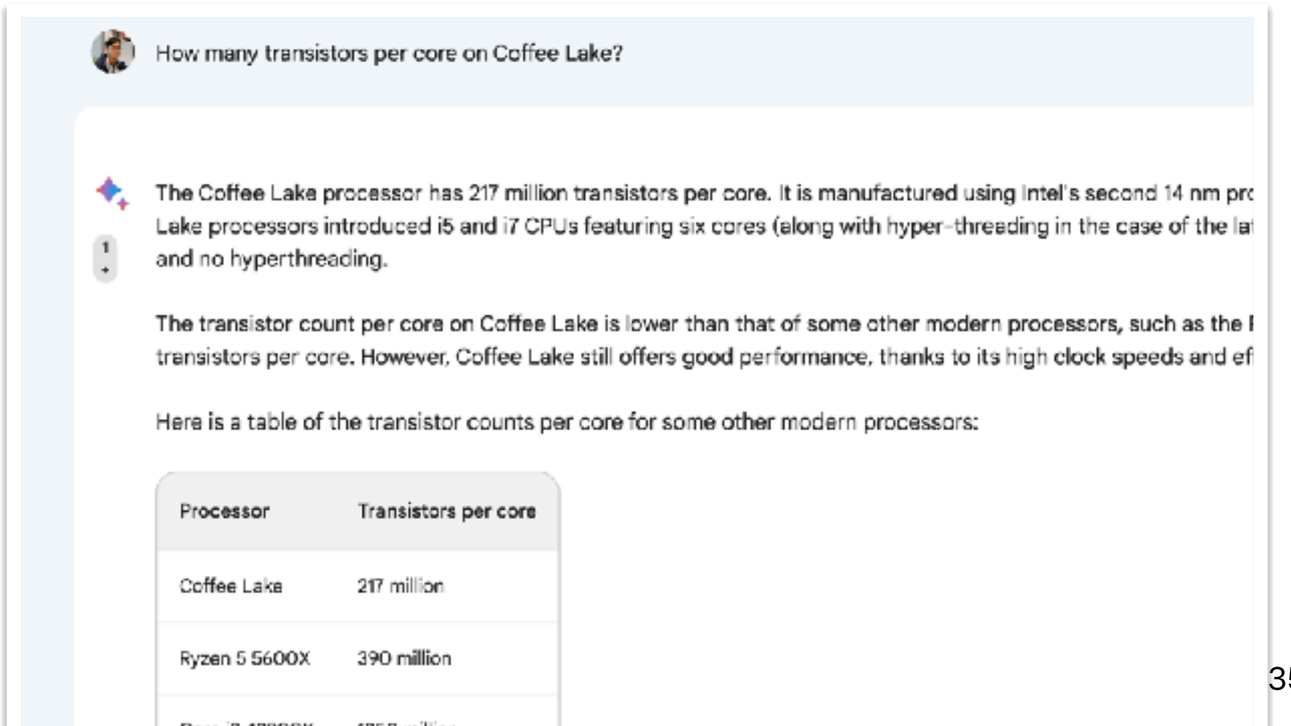
Some faster, some slower



=50W!

Transistor counts

Microarchitecture	Transistor Count	Issue-width	Year
Alder Lake	325 M	5x ALU, 7x Memory	2021
Coffee Lake	217 M	4x ALU, 4x Memory	2017
Sandy Bridge	290 M	3x ALU, 3x Memory	2011
Nehalem	182.75 M	3x ALU, 3x Memory	2008



How many transistors per core on Coffee Lake?

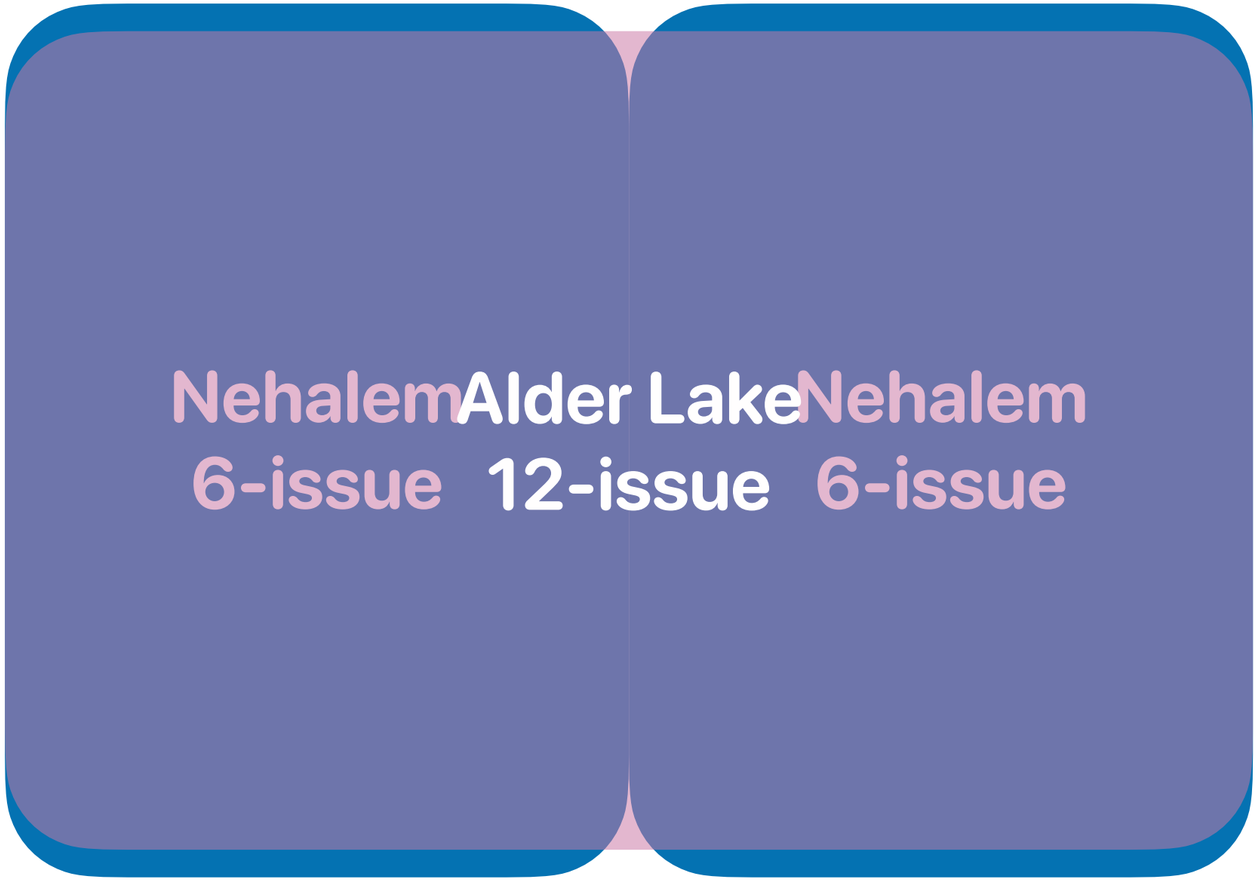
The Coffee Lake processor has 217 million transistors per core. It is manufactured using Intel's second 14 nm pro
Lake processors introduced i5 and i7 CPUs featuring six cores (along with hyper-threading in the case of the lat
and no hyperthreading.

The transistor count per core on Coffee Lake is lower than that of some other modern processors, such as the I
transistors per core. However, Coffee Lake still offers good performance, thanks to its high clock speeds and ef

Here is a table of the transistor counts per core for some other modern processors:

Processor	Transistors per core
Coffee Lake	217 million
Ryzen 5 5600X	390 million

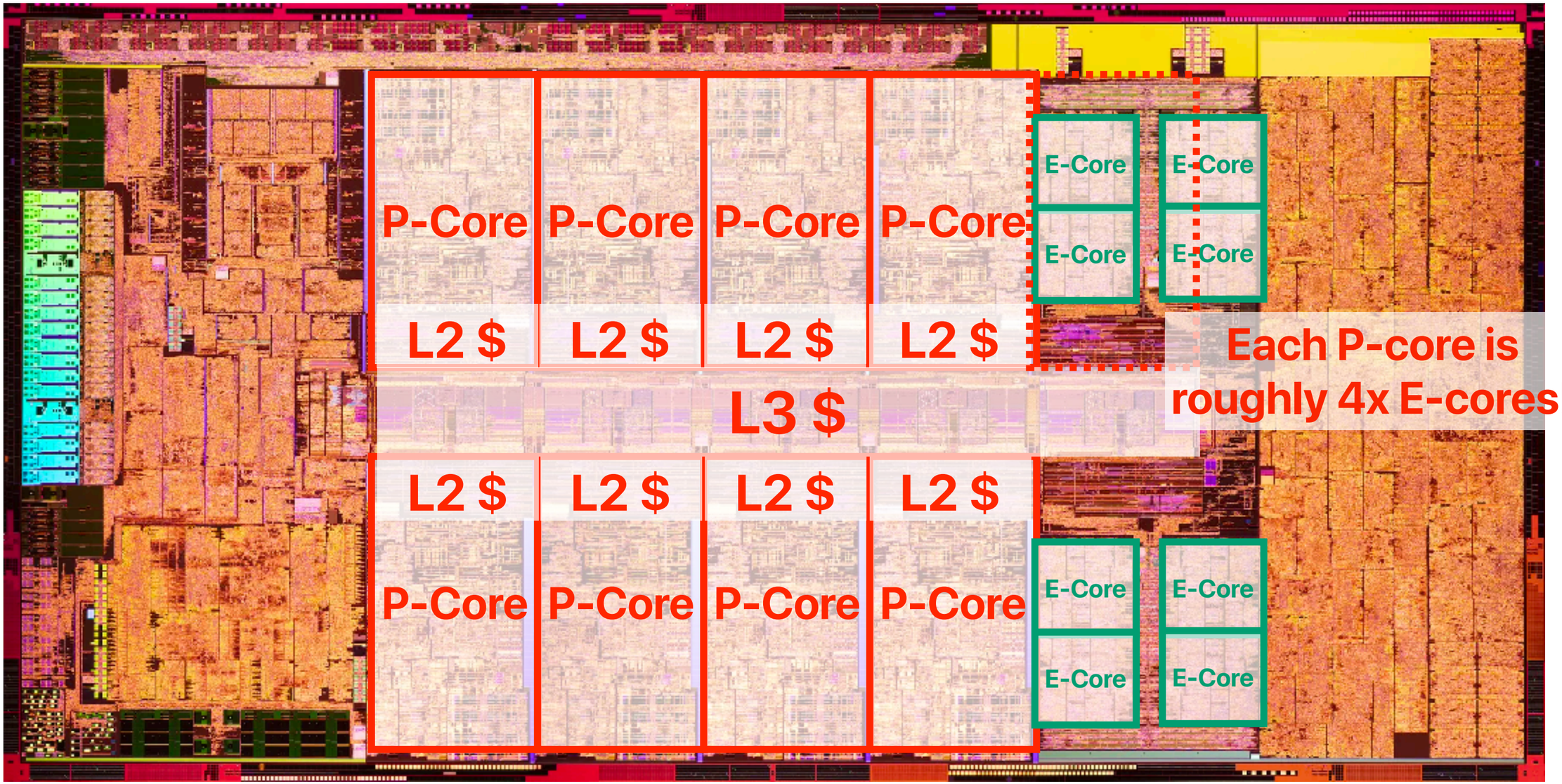
2x 3-issue ALUs Nehalem



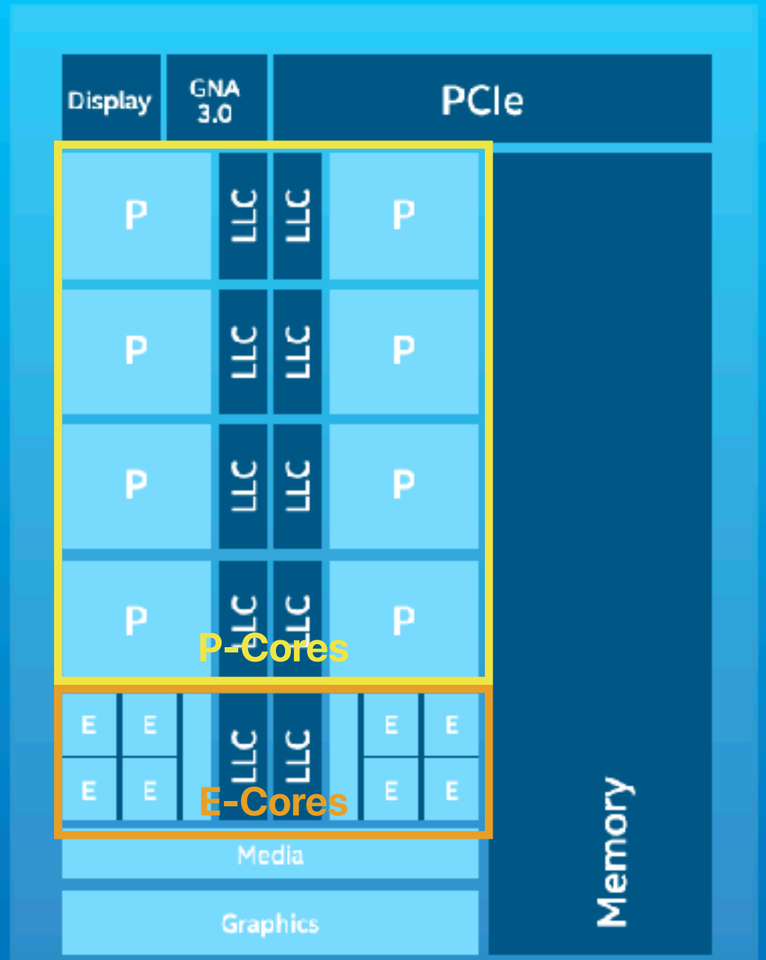
1x 5-issue ALUs Alder Lake

Based on https://en.wikipedia.org/wiki/Transistor_count

Intel Alder Lake



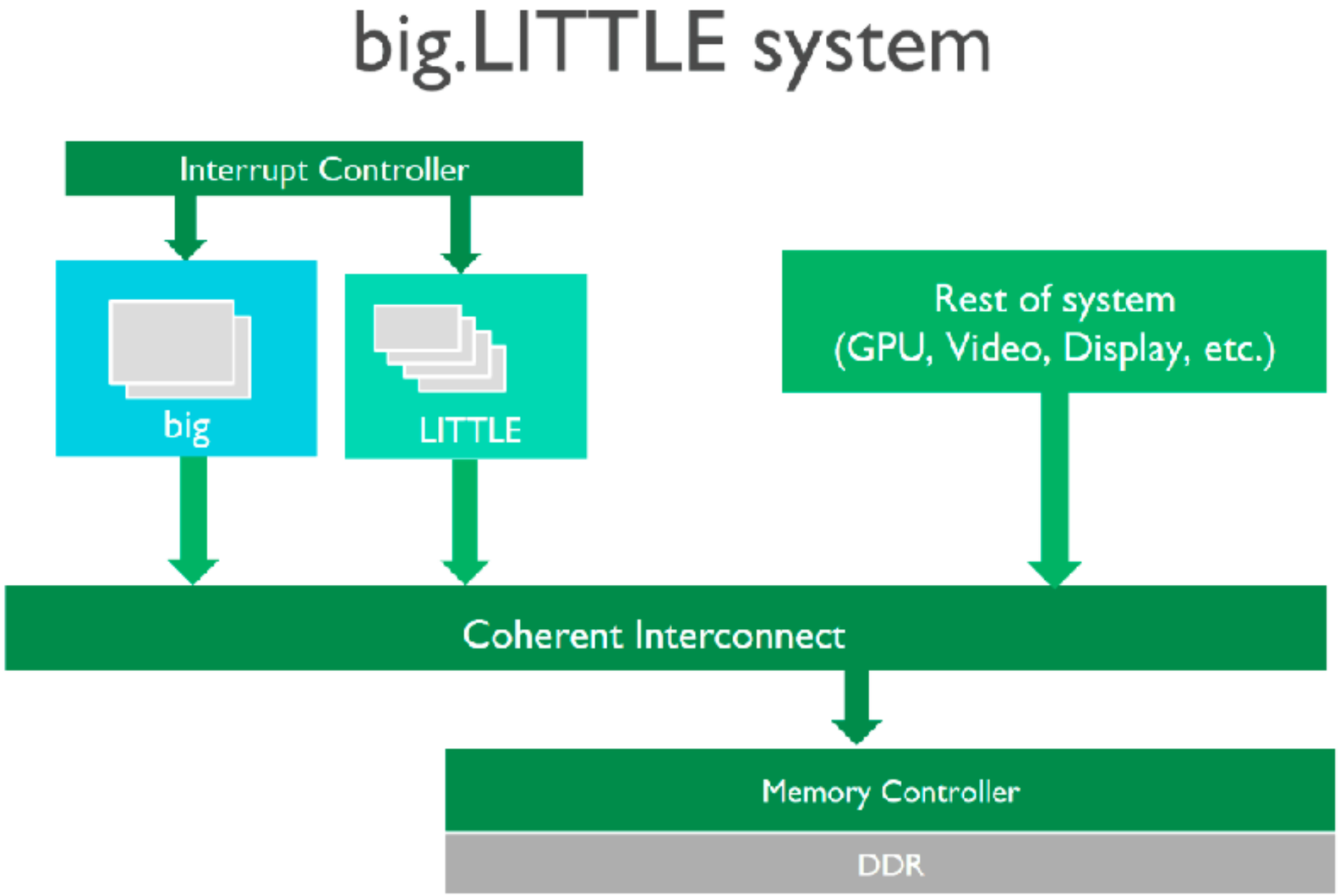
Single ISA heterogeneous CMP in Intel Processors



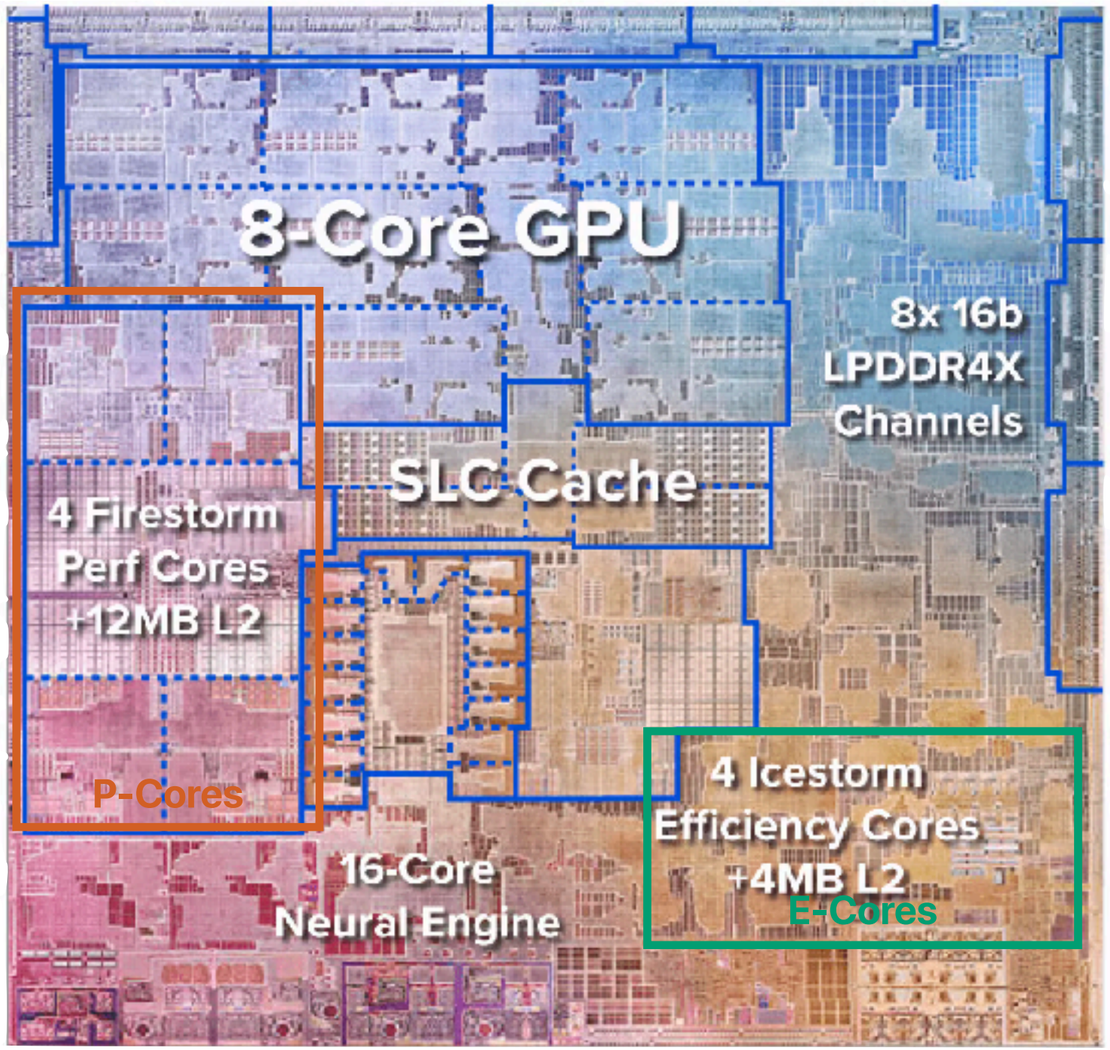
In a game-changing advance for core performance, 12th Gen Intel® Core™ desktop processors power a revolutionary approach to the x86 architecture. Its Performance-cores—or "P-cores"—are optimized for single & lightly-threaded performance, while its Efficient-cores—or "E-cores"—are optimized for scaling highly-threaded workloads. Intel® Thread Director helps to monitor and analyze performance data in real-time to seamlessly place the right application thread on the right core and optimize performance per watt.[1] That means gamers, creators, and professionals can harness both intelligence and power to enhance the experiences that matter most.

https://download.intel.com/newsroom/2021/client-computing/12th-gen-processor-product-brief.pdf

Single ISA heterogeneous CMP in ARM's big.LITTLE architecture



Single ISA heterogeneous CMP in Apple's M1



https://www.toptal.com/apple/apple-m1-processor-compatibility-overview

Take-aways: the new golden age of computer architectures

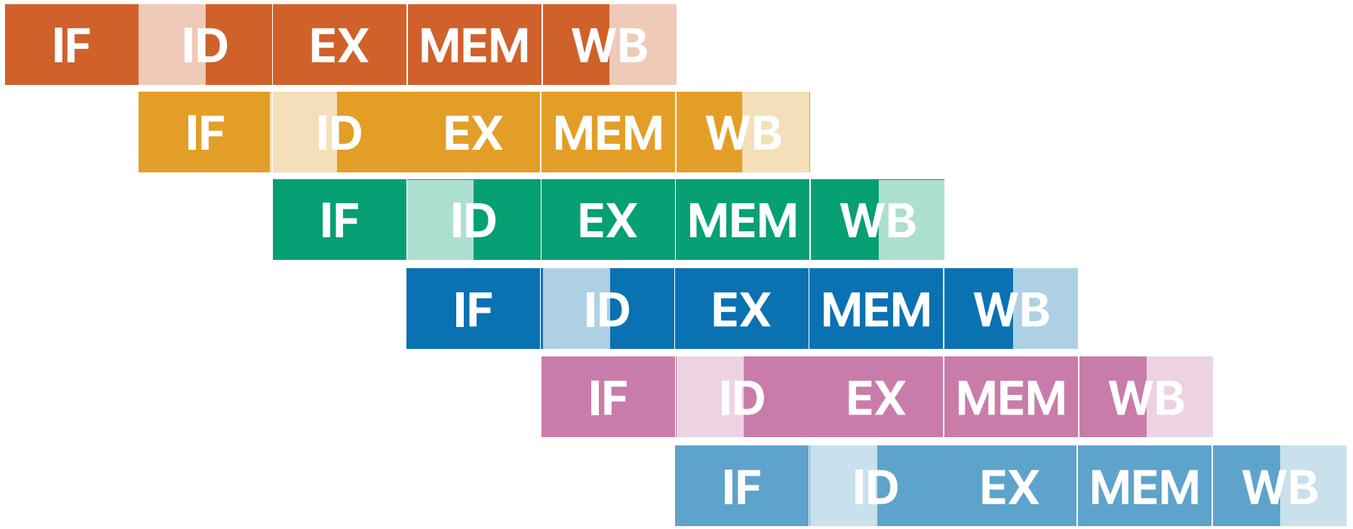
- Challenges and SOTA solutions in the dark silicon era
  - GPUs/many-core processors improve the **throughput** per-chip through providing massive parallelism where each processing element operates at a lower speed, but not-ideal for latency-sensitive workloads
  - Aggressive dynamic frequency/voltage scaling on CMP to accommodate the demand of **latency**-sensitive, parallelism-limited applications, but the area-efficiency of the slower cores is not great
  - Single ISA, heterogeneous CMPs (e.g., big.Little cores, Intel/Apple's P-cores/E-cores) find a balance the trade-offs of general-purpose workloads, but won't be ideal if your applications go to either extreme of throughput or latency

The Rise of ASICs/Accelerators — A New “Golden” Age of Architects

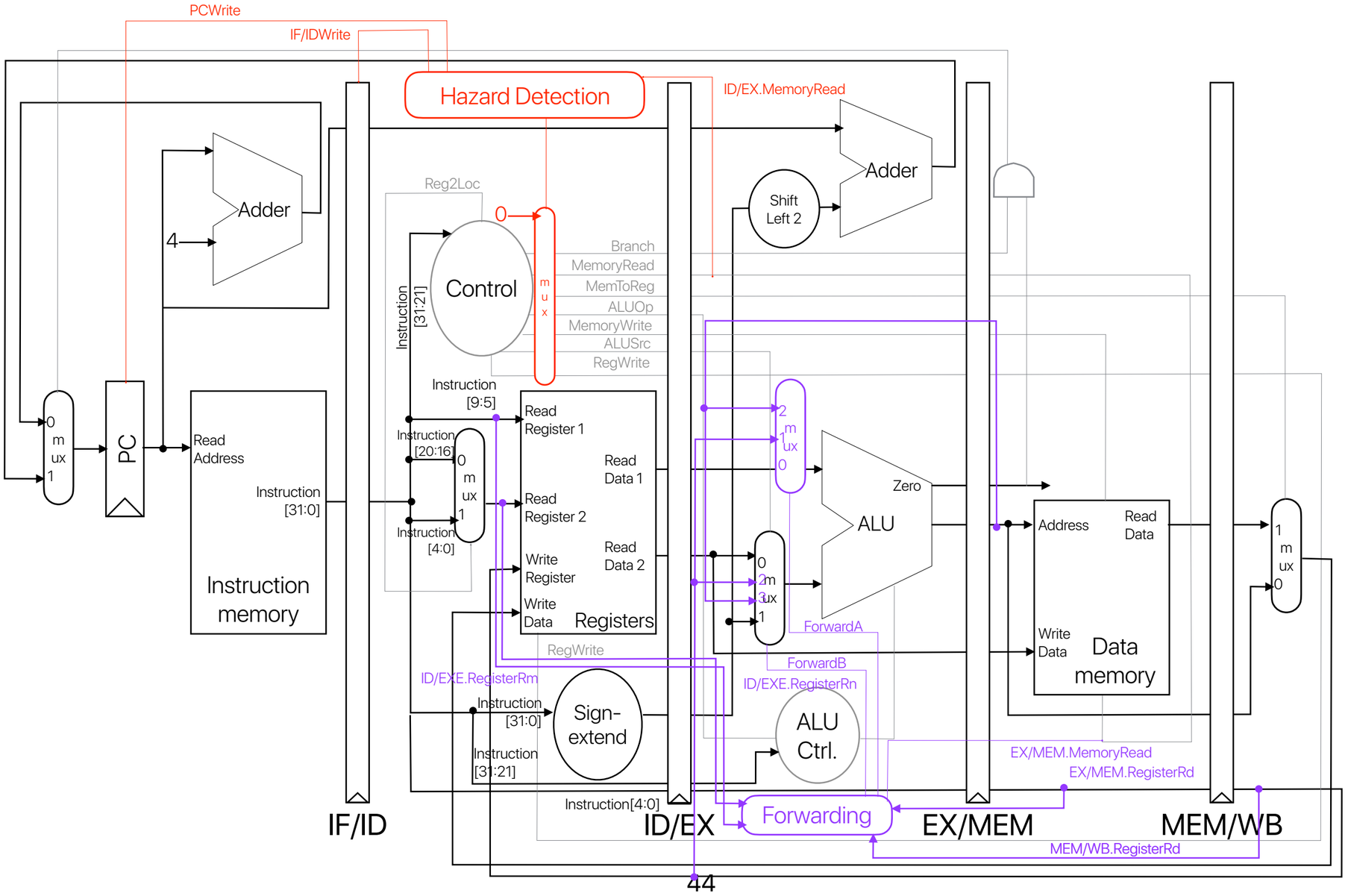
Say, we want to implement a[i] += a[i+1]*20

- This is what we need in RISC-V in each iteration

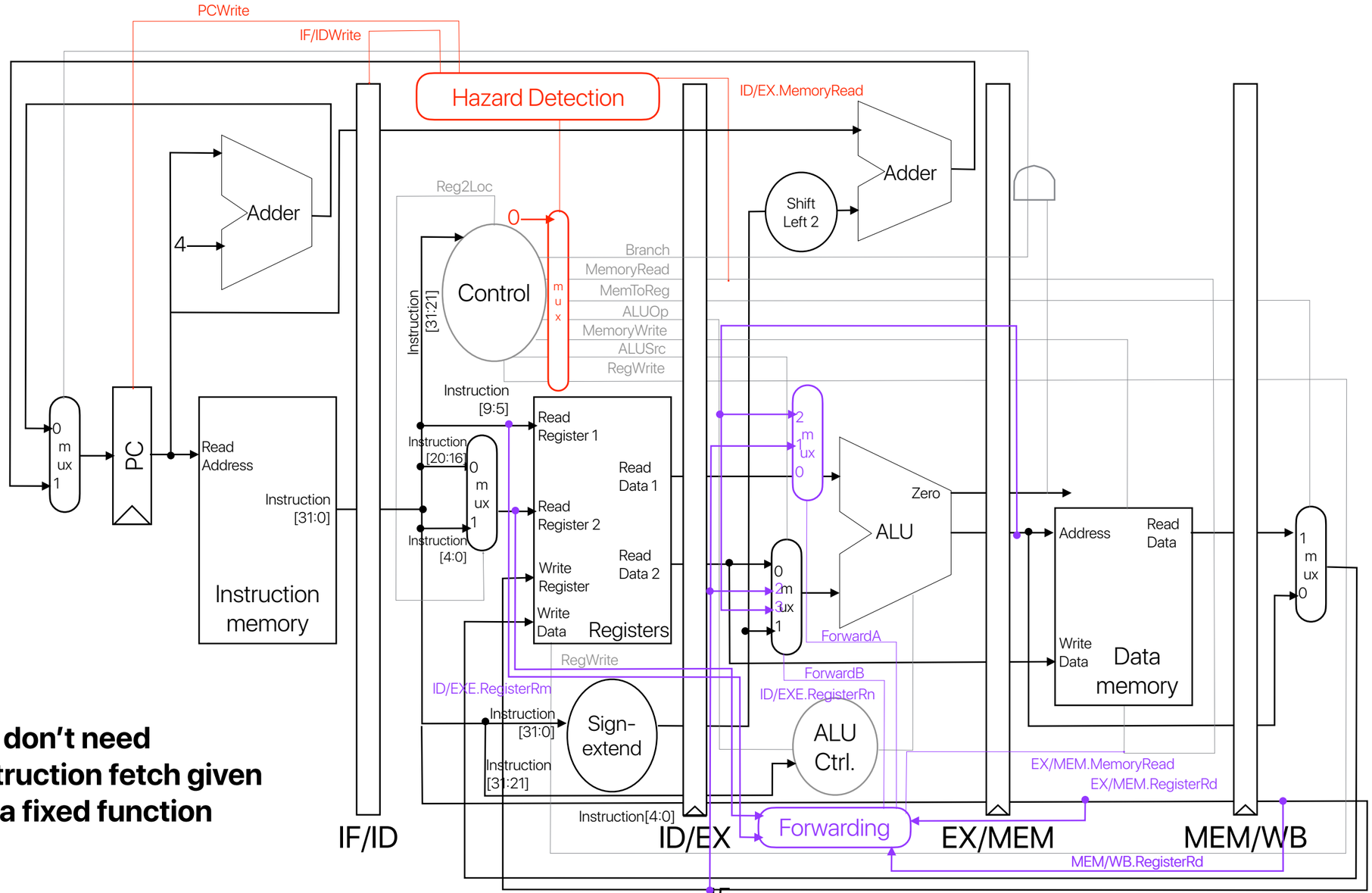
```
ld    X1, 0(X0)
ld    X2, 8(X0)
add   X3, X31, #20
mul   X2, X2, X3
add   X1, X1, X2
sd    X1, 0(X0)
```



This is what you need for these instructions



Specialize the circuit

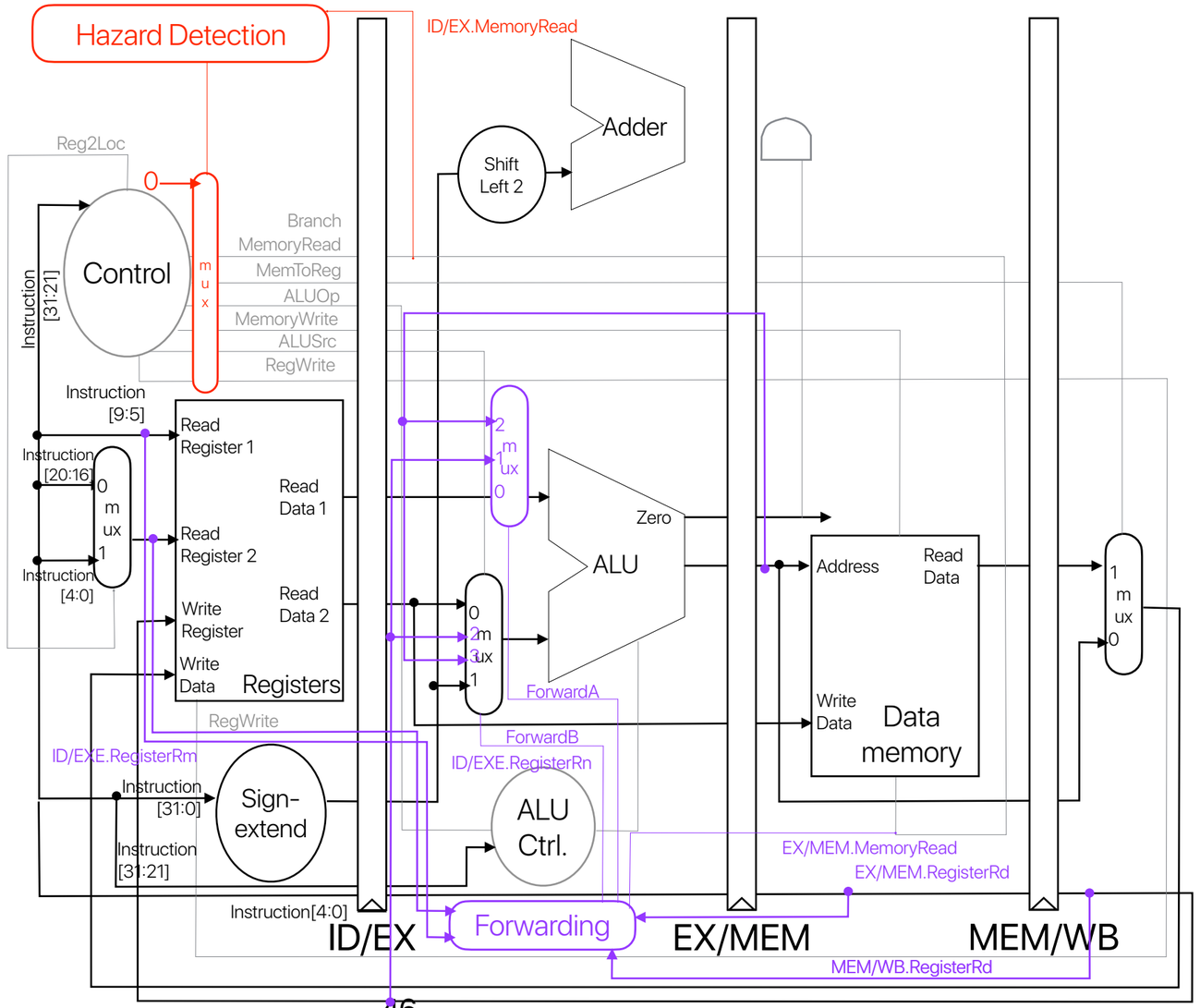


We don't need instruction fetch given it's a fixed function

Specialize the circuit

We don’t need these many registers, complex control, decode

We don’t need instruction fetch given it’s a fixed function

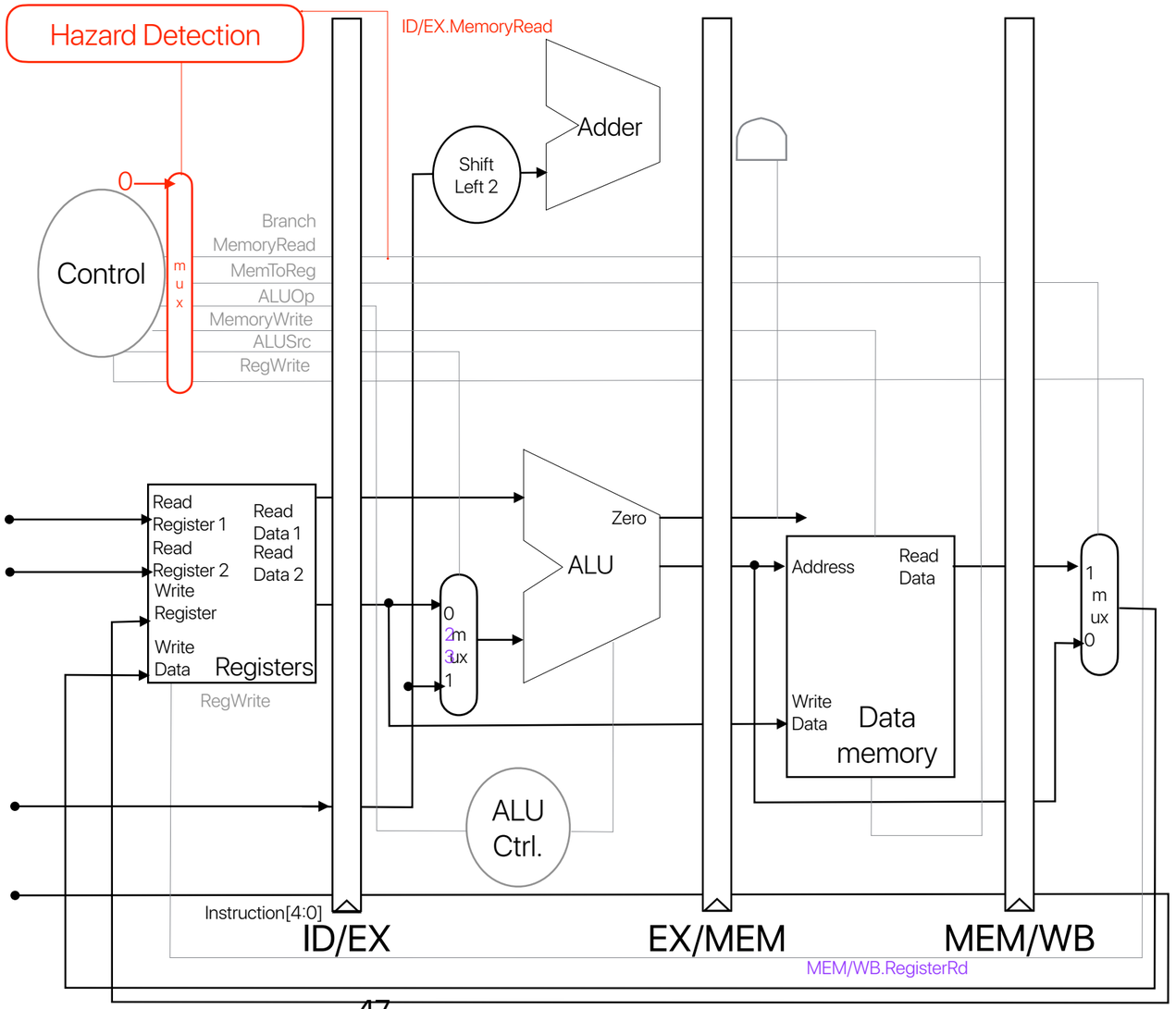


Specialize the circuit

We don't need ALUs, branches, hazard detections…

We don't need these many registers, complex control, decode

We don't need instruction fetch given it's a fixed function

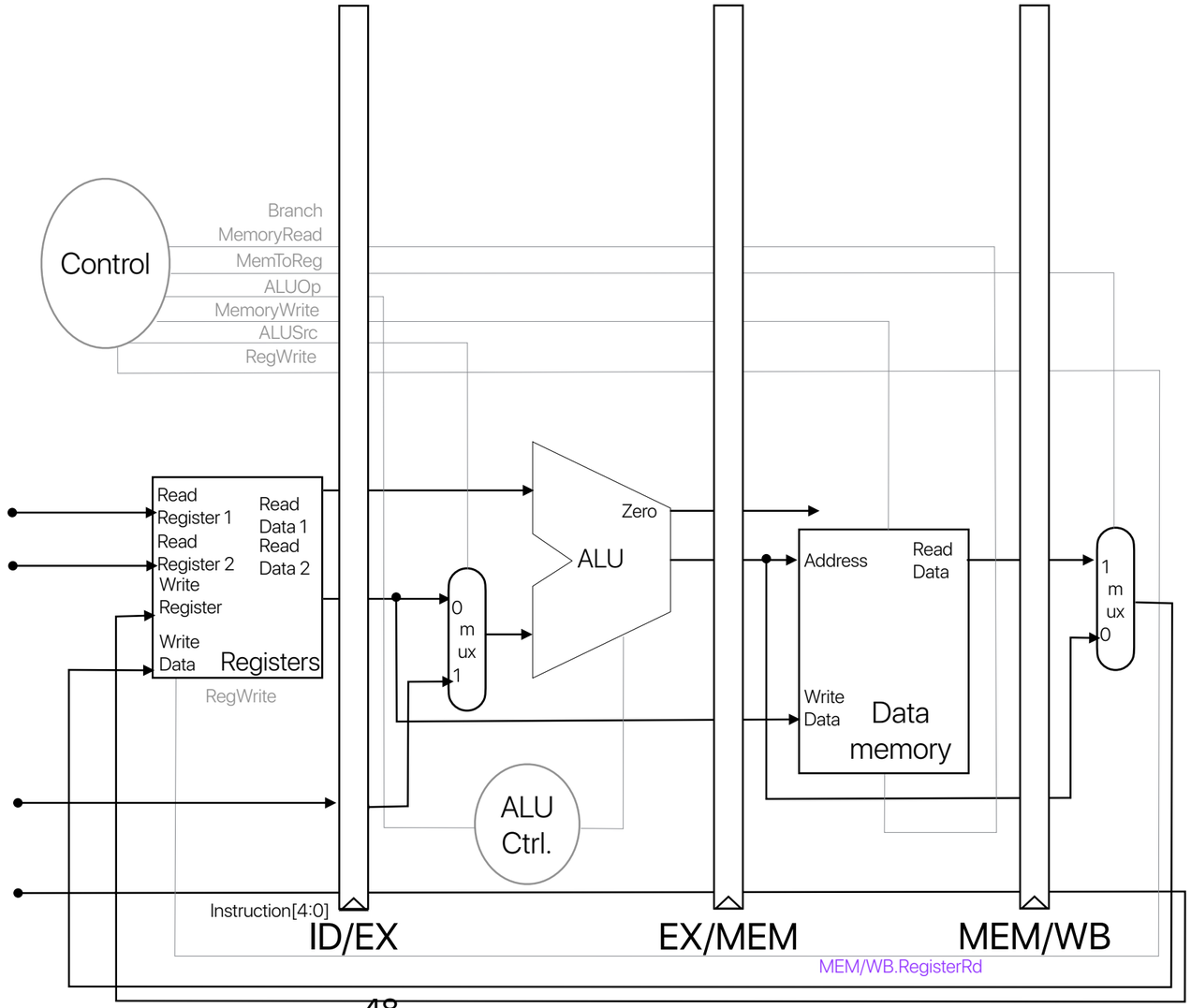


Specialize the circuit

We don’t need big ALUs, branches, hazard detections…

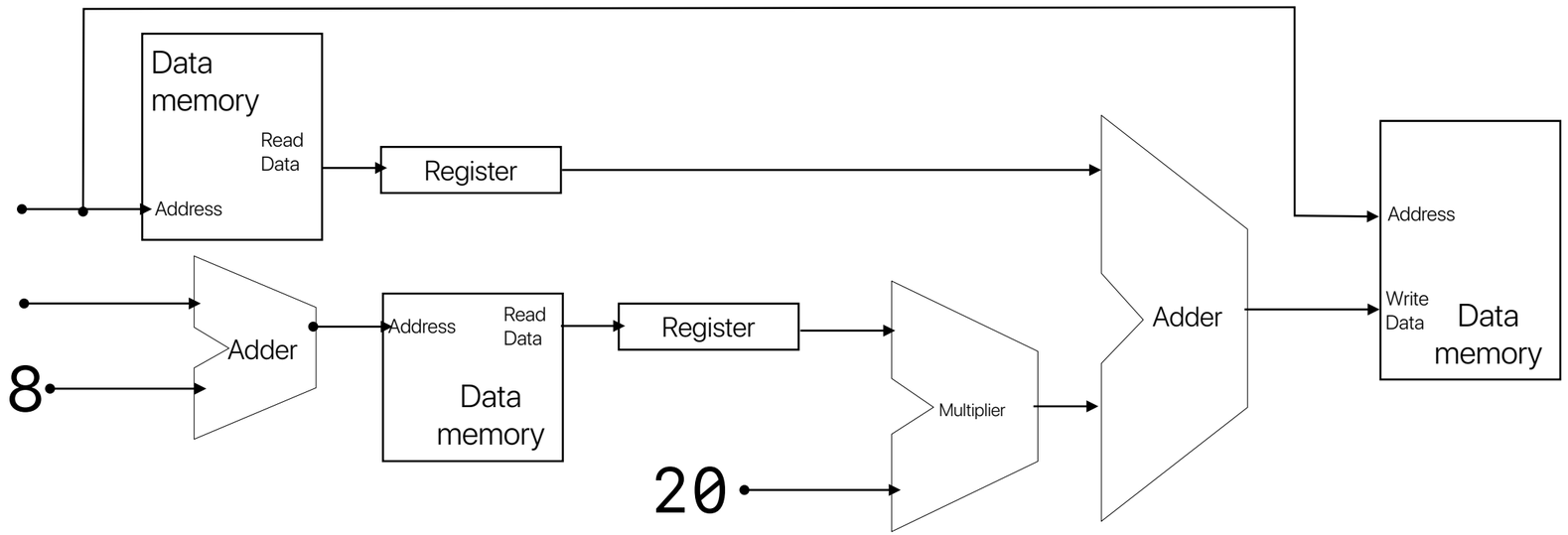
We don’t need these many registers, complex control, decode

We don’t need instruction fetch given it’s a fixed function



Rearranging the datapath

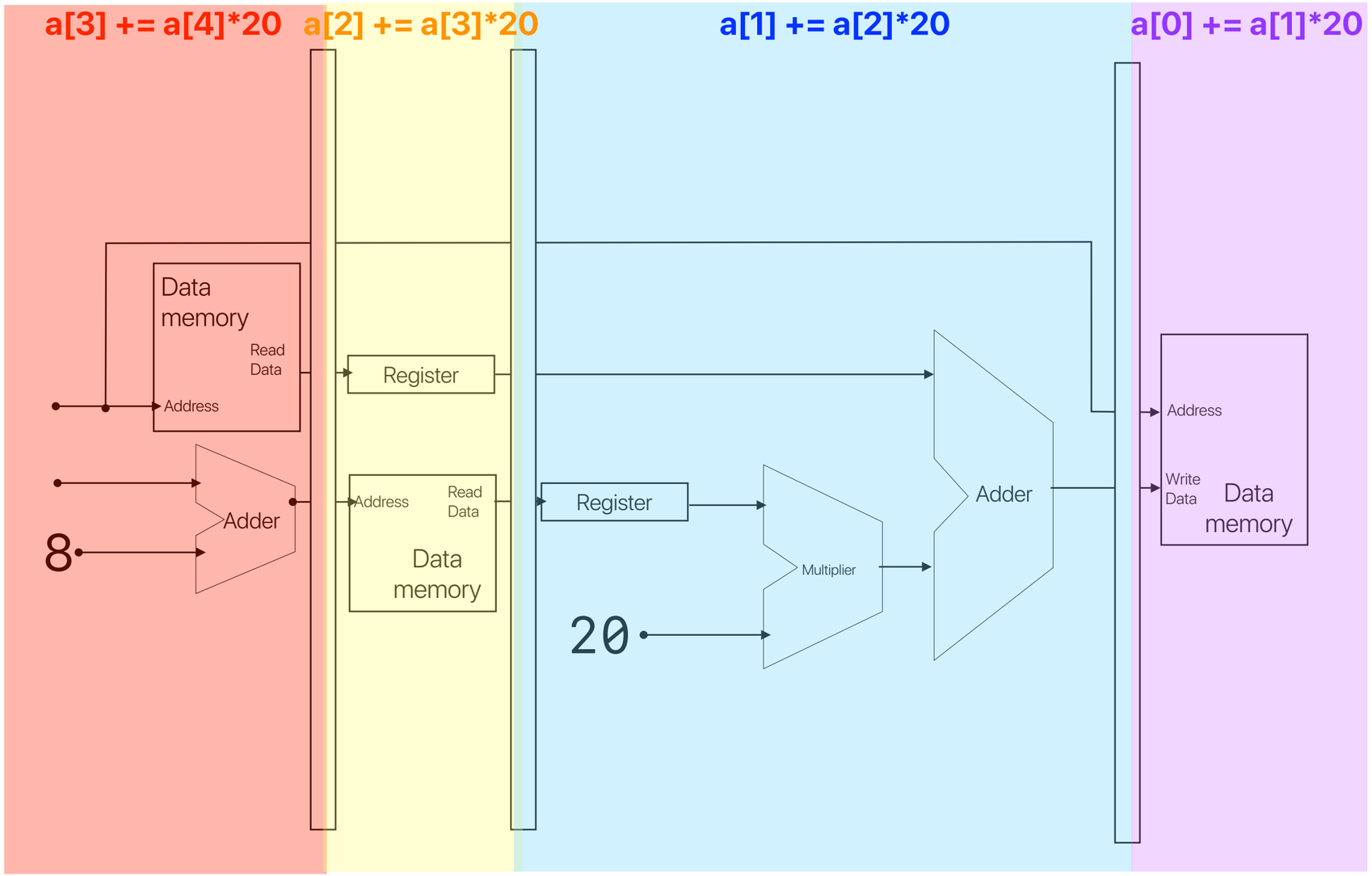
```
ld    X1, 0(X0)
ld    X2, 8(X0)
add   X3, X31, #20
mul   X2, X2, X3
add   X1, X1, X2
sd    X1, 0(X0)
```



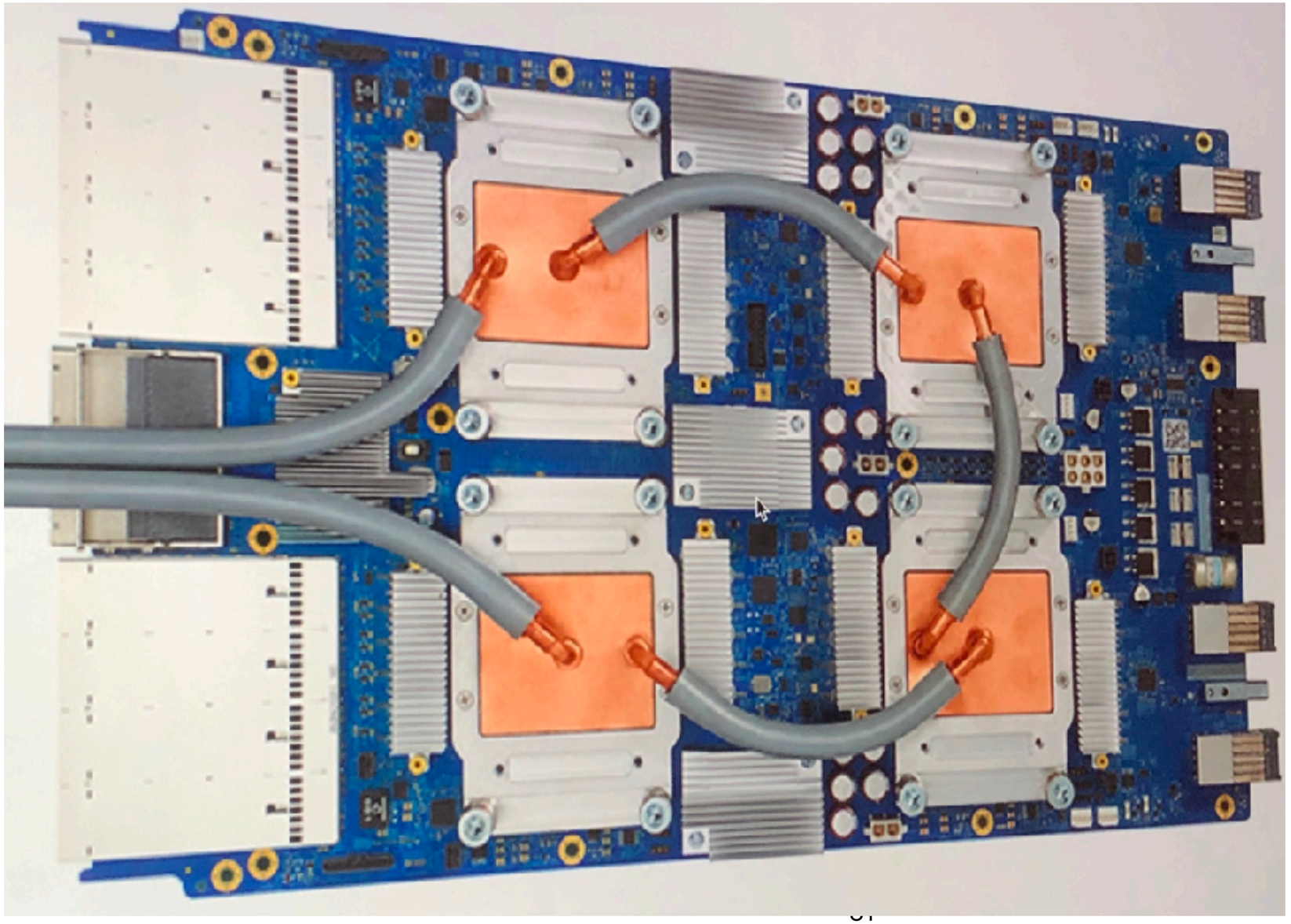
The pipeline for a[i] += a[i+1]*20

Each stage can still be as fast as the pipelined processor

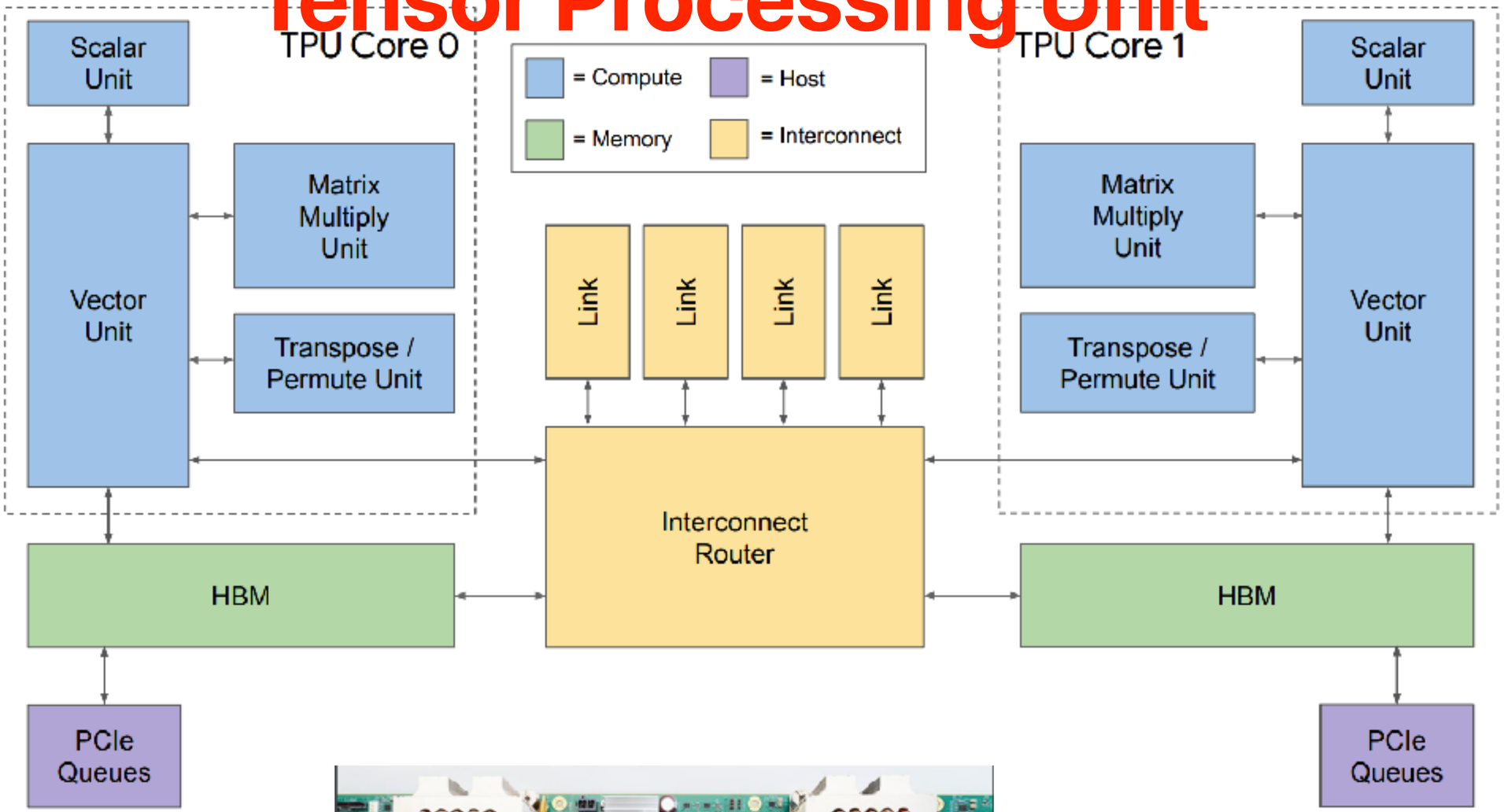
But each stage is now working on what the original 6 instructions would do

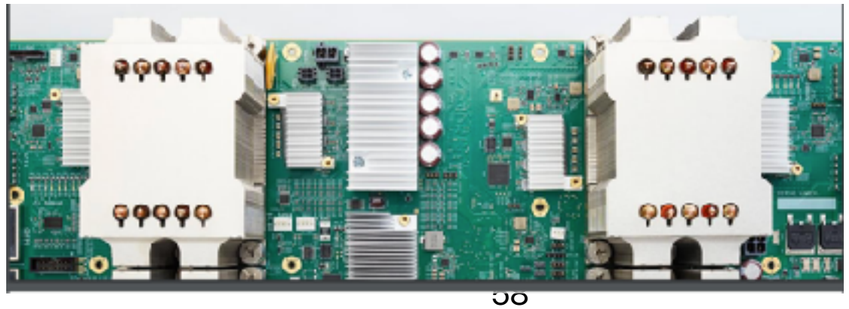


Tensor Processing Units

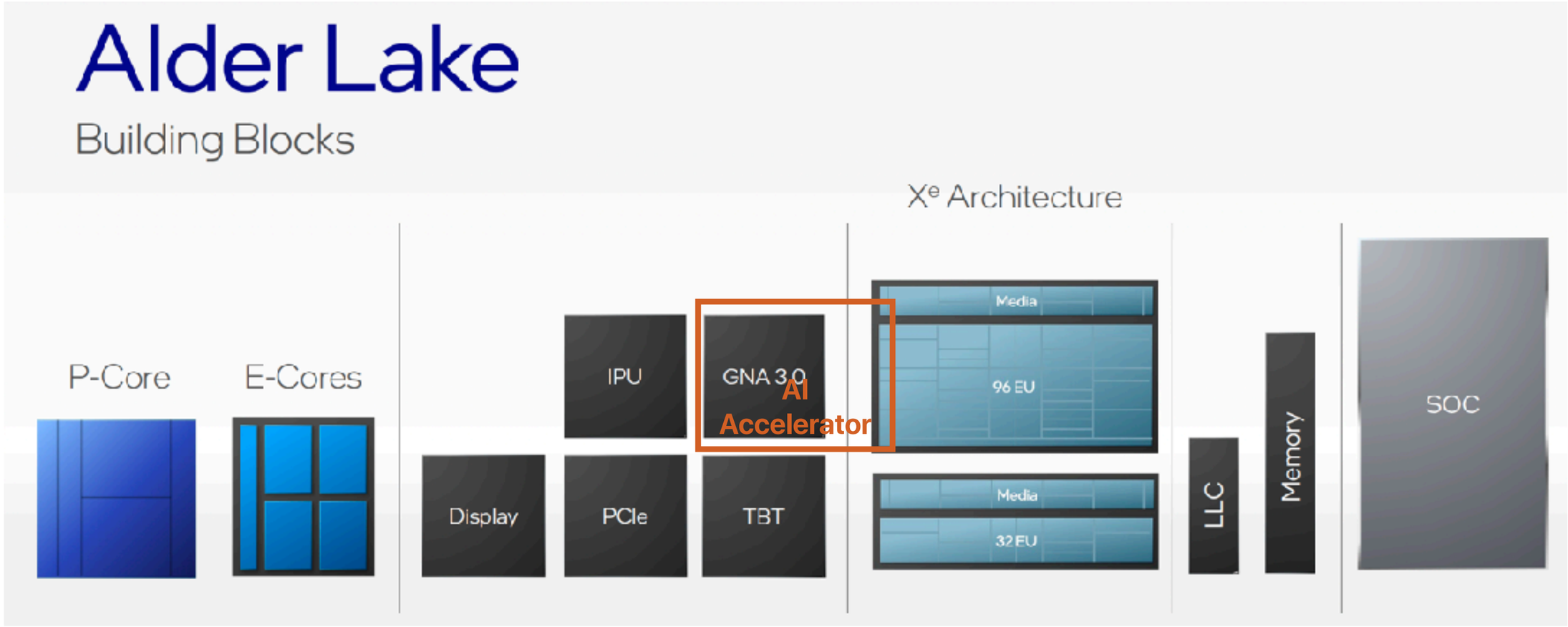


Tensor Processing Unit

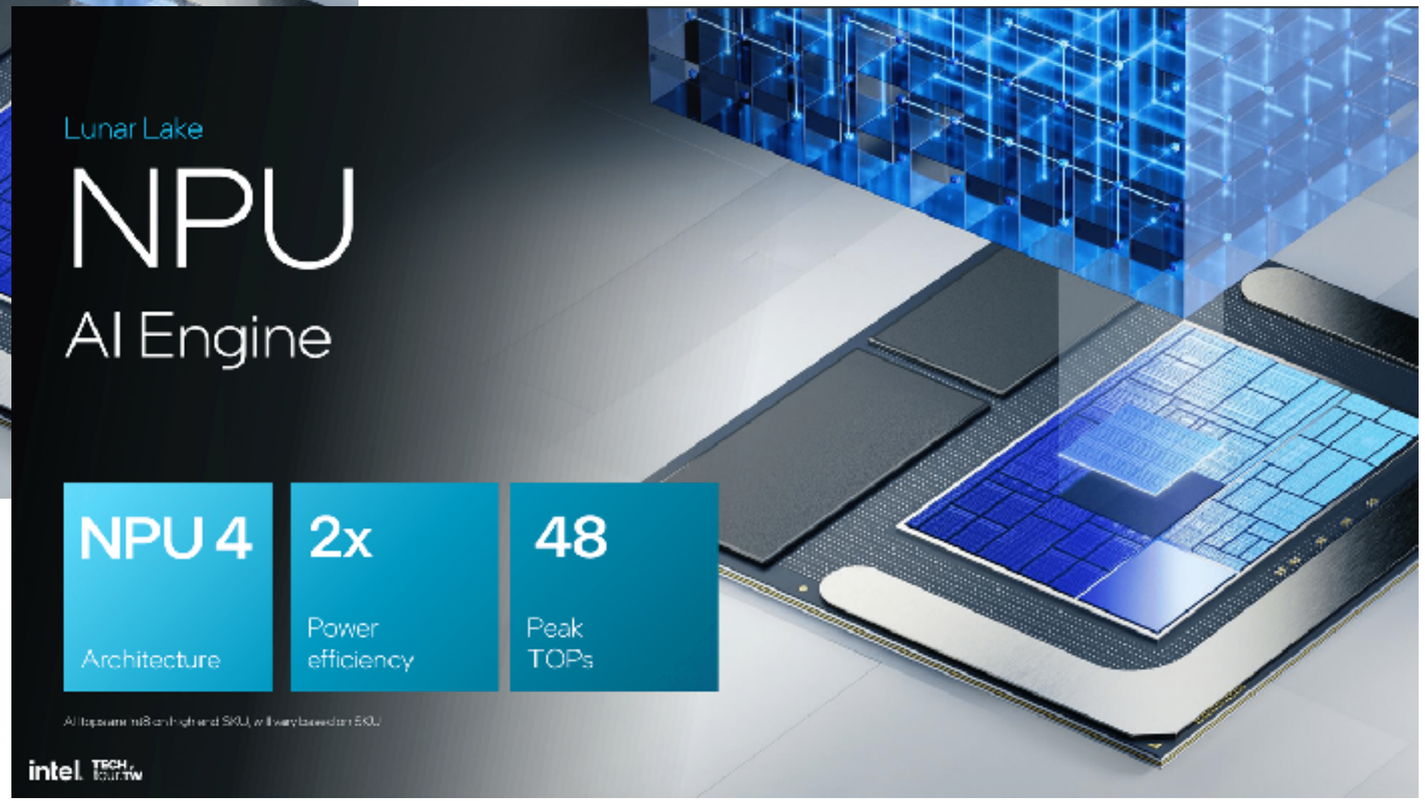




Modern processors also contain “accelerators”

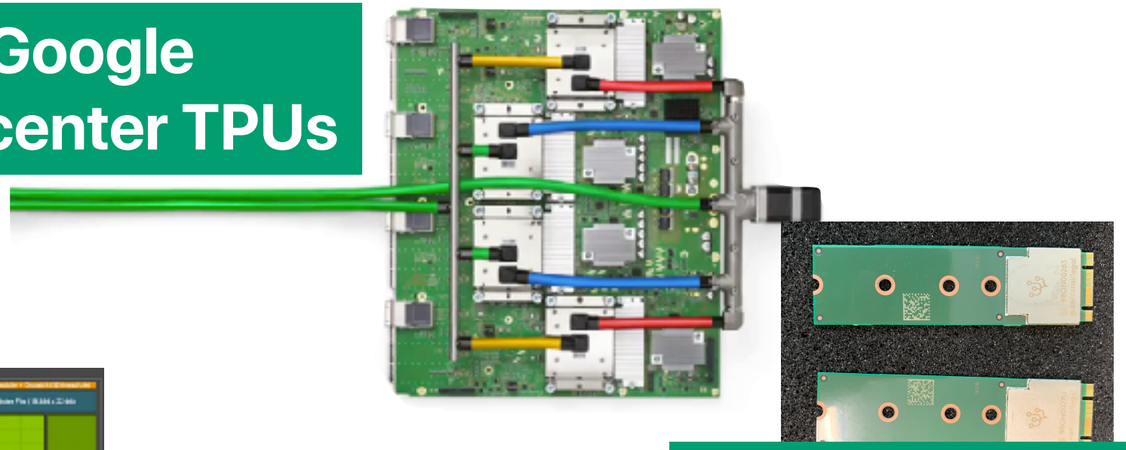



Lunar Lake
CPU
AI Engine
P-core & E-core
CPU architecture
VNNI & AMX
AI instructions
5
peak TOPs
intel.


Lunar Lake
NPU
AI Engine
NPU 4
Architecture
2x
Power efficiency
48
Peak TOPs
intel.

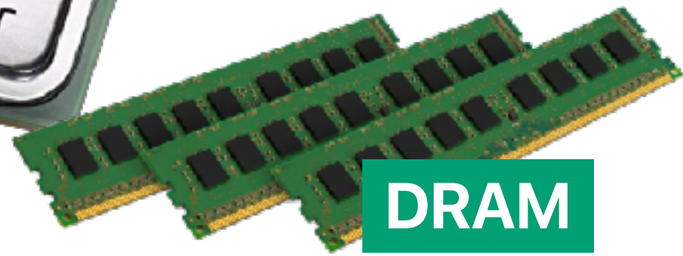
The “landscape” of modern computers

Google Datacenter TPUs



Google Edge TPUs

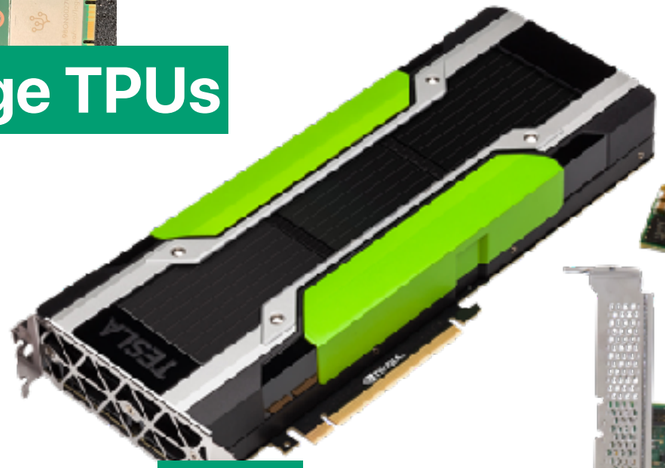
CPU



DRAM

NVIDIA Tensor Core Units

AI/ML Accelerators



GPU

SSD

Apple Neural Engines

NIC

Take-aways: the new golden age of computer architectures

- Challenges and SOTA solutions in the dark silicon era
  - GPUs/many-core processors improve the **throughput** per-chip through providing massive parallelism where each processing element operates at a lower speed, but not-ideal for latency-sensitive workloads
  - Aggressive dynamic frequency/voltage scaling on CMP to accommodate the demand of **latency**-sensitive, parallelism-limited applications, but the area-efficiency of the slower cores is not great
  - Single ISA, heterogeneous CMPs (e.g., big.Little cores, Intel/Apple's P-cores/E-cores) find a balance the trade-offs of general-purpose workloads, but won't be ideal if your applications go to either extreme of throughput or latency
  - Domain-specific accelerators or application-specific ICs make more efficient use of chip areas to fulfill the latency or throughput demand of applications, but sacrifices flexibility and application designers are now also hardware designers

Conclusion: A New Golden Age

Conclusion

- Computer architecture is now more important than you could ever imagine in the dark silicon era + the "new" golden age of computer architecture
- Just being a "programmer" is easy. You need to know architecture a lot to be a "performance programmer"
  - Optimizing locality and footprint for caching
  - Make your code or data more branch prediction friendly
  - Exploiting more instruction-level parallelism via carefully revisiting the critical path of execution
- Multicore era — to get your multithreaded program correct and perform well, you need to take care of coherence and consistency
- We're now in the "dark silicon era"
  - Single-core isn't getting any faster
  - Multi-core doesn't scale anymore
  - We will see more and more ASICs
  - You need to write more "system-level" programs to use these new ASICs.
- Can we be more creative in using ASICs?

Announcements

- **Course evaluation** started and ends on 9/05/2024
  - Submit the prove of your participation in course evaluation through Gradescope
  - It can become a full credit reading quiz (it helps to amortize the penalty of another least performing one) — we will drop a total of three now
- **Assignment 5** is **due 9/05/2024**
  - The due date is earlier to allow publication of solutions before the deadline
- **Final exam**
  - 9/6 3p-6p @ **PCH 121**
  - Closed book, no cheatsheet — the same rules as the midterm
  - Pizza party after the examine


Computer
Science &
Engineering
142
ありがとう
Prof. Usagi