ECE 2300
Digital Logic & Computer Organization

Spring 2025

Advanced Topics

Cornell University

Announcements

- Lab 5 deadline extended

- Return the FPGA board by Friday May 9th during a TA OH

- Fill out 2300 course evaluation (due Friday 5/9)
  - Comments not required but very welcome

Final Exam

- Saturday May 10, 9:00-10:40AM (100 mins) at Phillips 101
  - Arrive early by 8:50am
  - Closed book, closed notes, closed Internet

  - Coverage: Full course, with a particular emphasis on computer organization (Lectures 15~25)
    - Programmable microprocessor, pipelining, caches, performance measurement, virtual memory, exceptions, I/O
    - Other essential concepts (e.g., 2's complement, timing analysis, and ISA) may still appear in questions related to computer organization
  - Solution to the sample exam is posted on CMS
  - HW 8 solution will be released soon

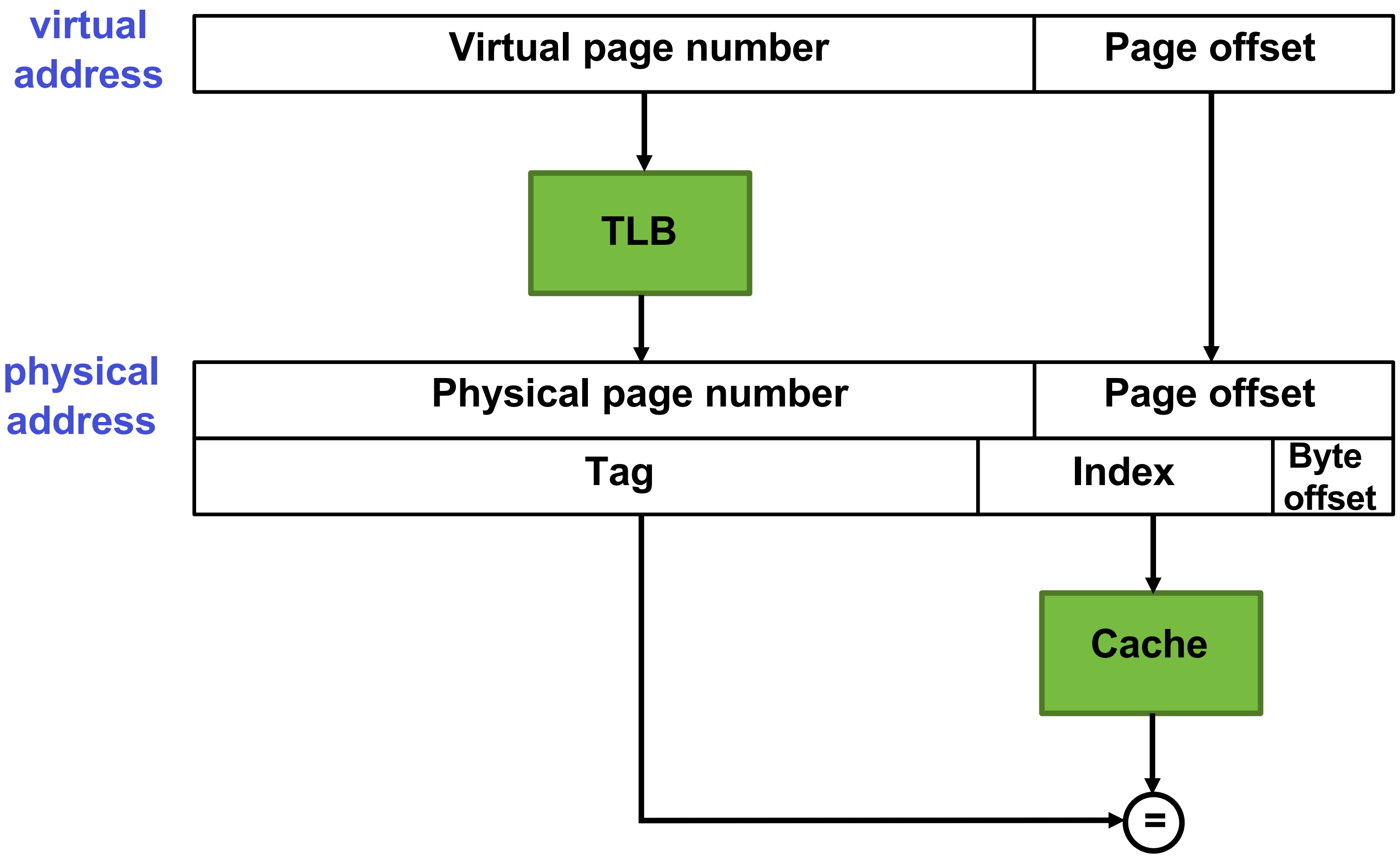
- Same OH schedule during study period (Ed post #21), except Slope Day

Review: Context Switching

- **Program counter (PC):** Save
- **Registers in RF:** Save
- **Page table register (PTR):** Save
- **TLB:** Invalidate all entries
- **Caches:** Typically retained; not flushed during context switch as they hold physical addresses

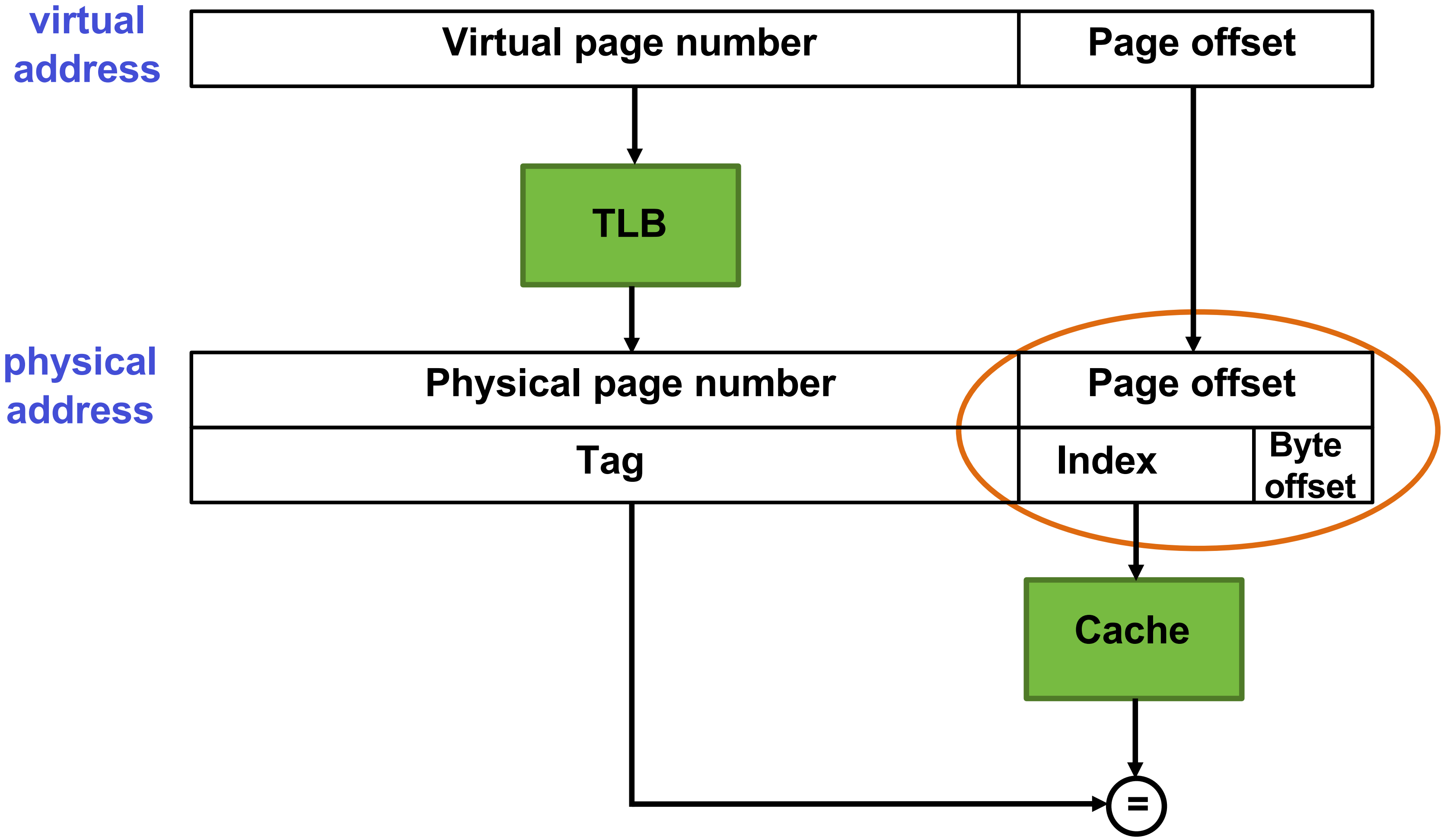
Review: Accessing the TLB and the Cache

- Cache usually uses physical addresses since it holds a subset of what is in MM

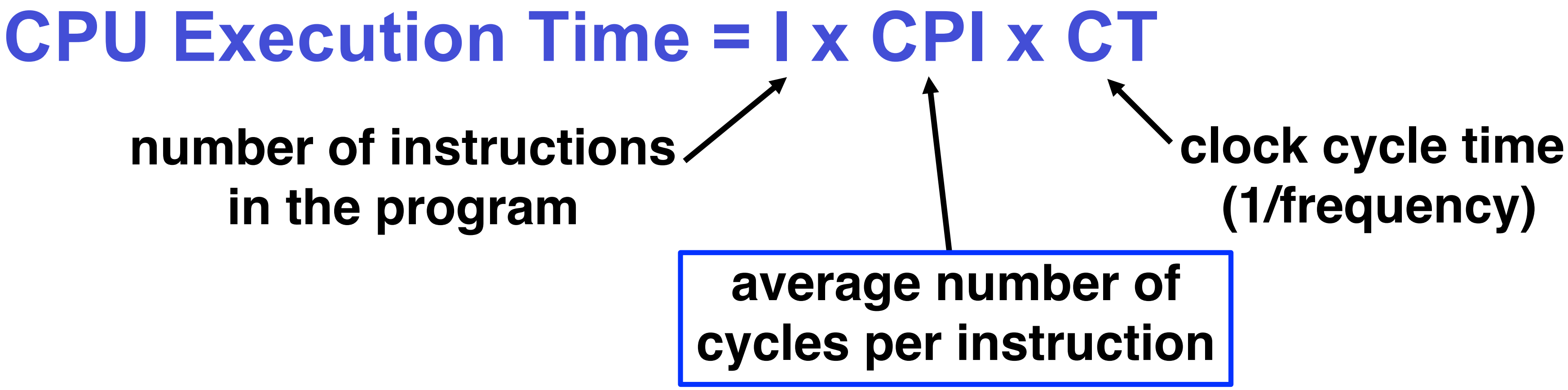


Accessing the TLB and the Cache

- What about this situation with a different cache configuration?
  - We can access the TLB and cache simultaneously because the index bits used for cache addressing don't require translation



Iron law of Processor Performance



CPU Execution Time = I x CPI x CT

number of instructions
in the program

average number of
cycles per instruction

clock cycle time
(1/frequency)

Parallelism as a Path to Lower CPI

- Processor architects improve performance through hardware that exploits the different types of *parallelism* within computer programs

- Instruction-Level Parallelism (ILP)
  - (fine-grain) parallelism within a sequential program
- Thread-level parallelism (TLP)
  - (coarse-grain) parallelism among different threads in a program
- Data-level parallelism (DLP)
  - parallelism among the data within a program

Instruction-Level Parallelism (ILP)

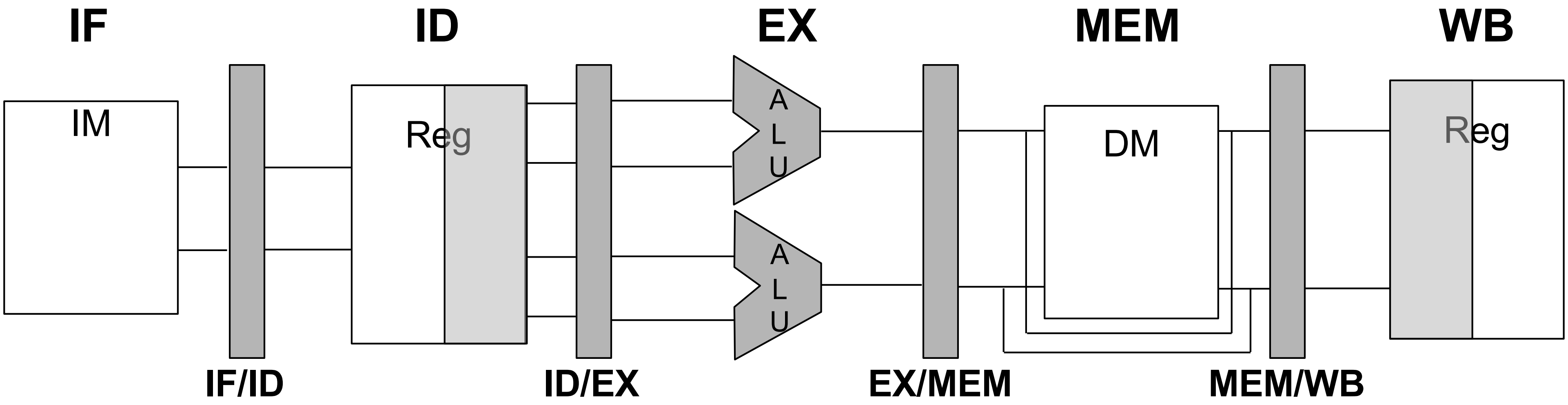
- Refers to the parallelism found within a sequential program

- Consider the ILP in this program segment

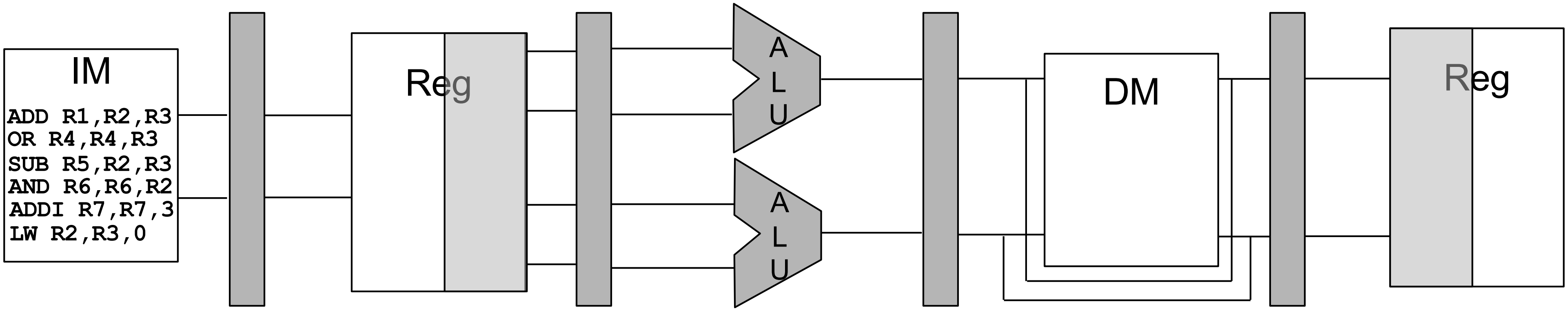
```
ADD R1,R2,R3
OR R4,R4,R3
SUB R5,R2,R3
AND R6,R6,R2
ADDI R7,R7,3
LW R2,R3,0
```

- *Superscalar pipelines* exploit ILP by duplicating the pipeline hardware

Two-Way Superscalar Pipeline

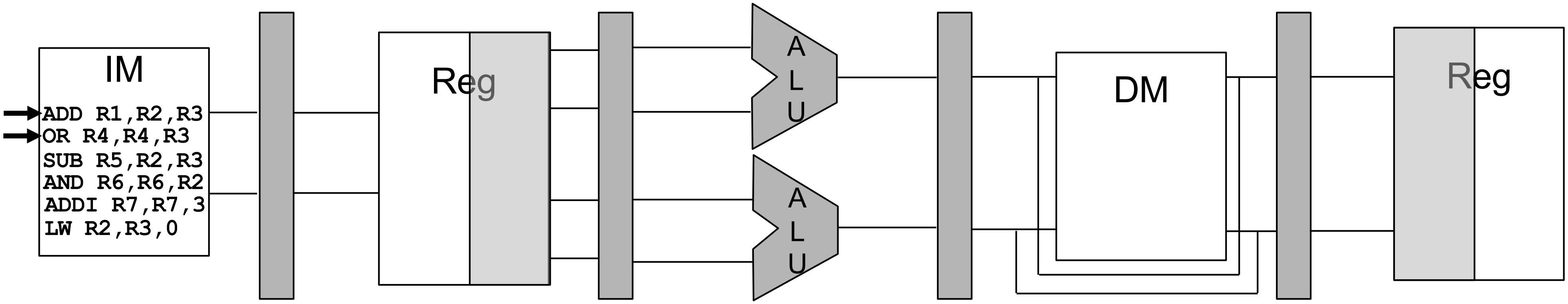


Instruction Sequence on 2W SS



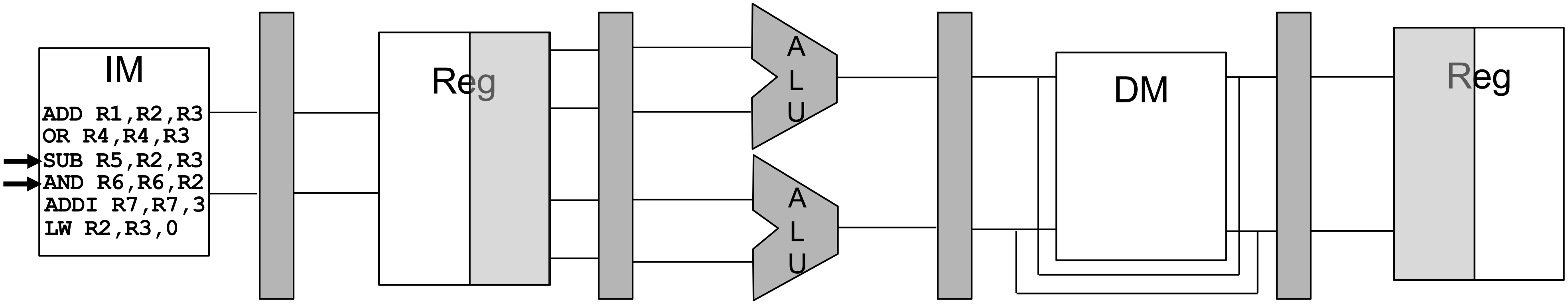
```
ADD R1,R2,R3
OR R4,R4,R3
SUB R5,R2,R3
AND R6,R6,R2
ADDI R7,R7,3
LW R2,R3,0
```

Instruction Sequence on 2W SS



```
ADD R1,R2,R3
OR R4,R4,R3
```

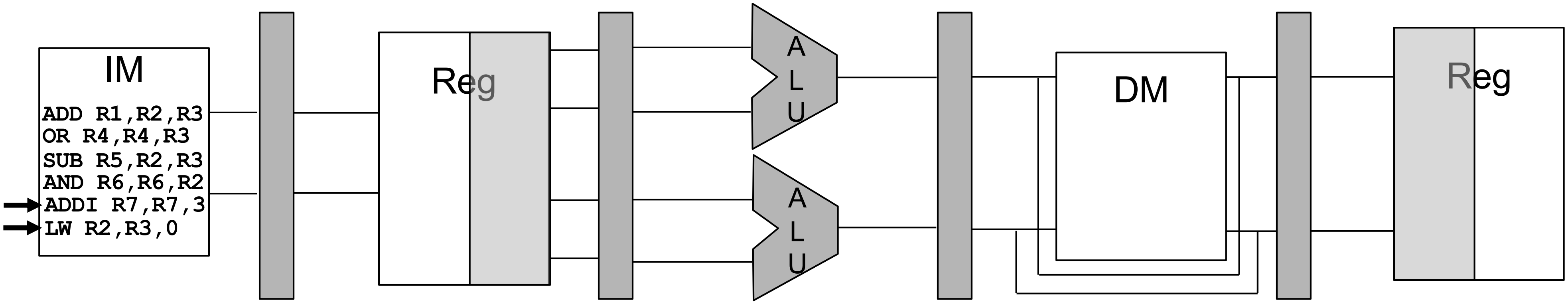
Instruction Sequence on 2W SS



```
ADD R1,R2,R3
OR R4,R4,R3
SUB R5,R2,R3
AND R6,R6,R2
ADDI R7,R7,3
LW R2,R3,0
```

```
SUB R5,R2,R3     ADD R1,R2,R3
AND R6,R6,R2     OR R4,R4,R3
```

Instruction Sequence on 2W SS



```
ADD R1,R2,R3
OR R4,R4,R3
SUB R5,R2,R3
AND R6,R6,R2
ADDI R7,R7,3
LW R2,R3,0
```

```
ADDI R7,R7,3    SUB R5,R2,R3    ADD R1,R2,R3
LW R2,R3,0      AND R6,R6,R2    OR R4,R4,R3
```

ARM Cortex-A8 Microprocessor

Apple iPhone 4, iPod Touch (3rd & 4th gen), iPad; Motorola Droid, Droid X, Droid 2; Palm Pre, Pre 2; Samsung Omnia HD, Wave 8500, i9000 Galaxy S, P1000 Galaxy Tab; HTC Desire; Google Nexus S; Nokia N900; Sony Ericsson Satio, Xperia X10, etc, etc

Cortex-A8 Processor Pipeline

- 2-way superscalar
  - With some dual issue restrictions: only one multiplier and one load/store unit
- Average CPI of 1.1
- 13 stages for integer instructions, 3 major sections
  - Instruction Fetch, Instruction Decode, Execute
- Up to 1GHz clock frequency
- ~0.5W @ 1GHz (processor core only)

Caches and TLBs

- Identical L1 instruction and data caches
  - 16KB or 32KB
  - 4-way set associative
  - 64-byte block size
  - Random replacement policy
- 32 entry, fully associative ITLB and DTLB
- L2 cache
  - Up to 1MB, 8-way set associative, 64 byte block size, random replacement

Data Dependency Limits SS Execution

- Consider this program sequence

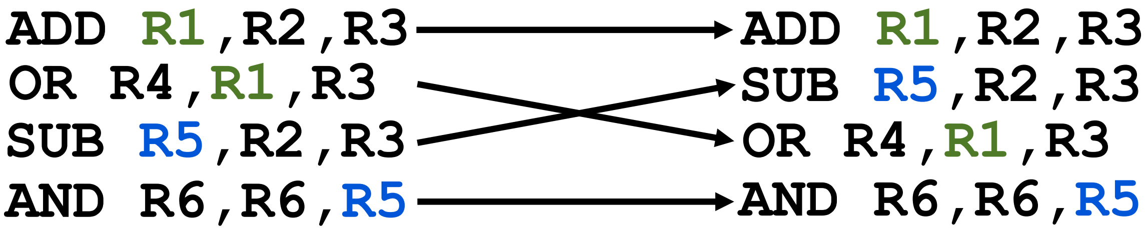
```
ADD R1,R2,R3
OR R4,R1,R3
SUB R5,R2,R3
AND R6,R6,R5
```

- The ADD and the OR, and the SUB and AND, cannot execute at the same time
  - Cortex-A8 limits dual issue in this case

- Addressed by *out-of-order execution*

Out-of-Order Execution

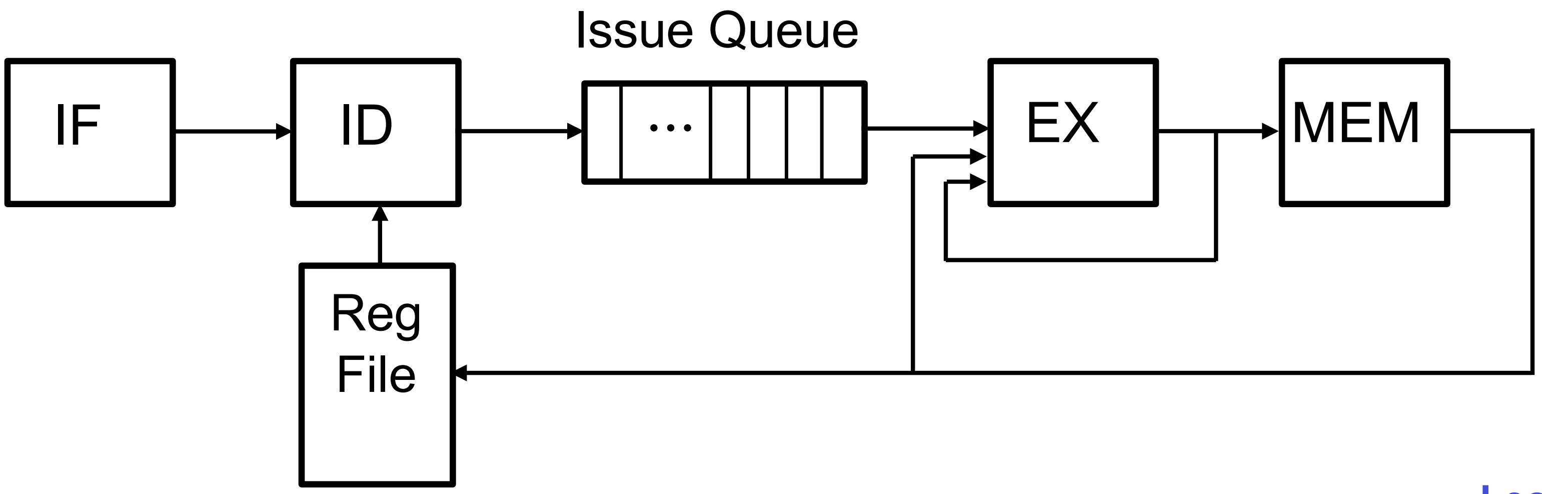
- Processor can execute instructions out of the original program order



```
ADD R1,R2,R3
OR R4,R1,R3
SUB R5,R2,R3
AND R6,R6,R5
```

```
ADD R1,R2,R3
SUB R5,R2,R3
OR R4,R1,R3
AND R6,R6,R5
```

- One key component is an *issue queue* that tracks the availability of source operands



ARM Cortex-A9

- Successor to the Cortex-A8
- Superscalar pipeline with out-of-order execution
  - Issues up to 4 instruction each clock cycle
- ITLB and DTLB + L2 TLB

Apple iPhone 4S, iPad2; Motorola Droid Bionic, Altrix 4G, Xoom; Blackberry Playbook; Samsung Galaxy S II, Galaxy S III; HTC Sensation, EVO 3D; LG Optimus 2X, Optimus 3D; Lenovo IdeaPad K2, ASUS Eee Pad Transformer; Acer ICONIA TAB A-series, etc, etc

A More Troublesome Piece of Code

- Now consider this program sequence

```
Loop1: ADD R1,R2,R3
       OR R4,R1,R3
       SUB R5,R4,R3
       AND R1,R6,R5
       BEQ R2,R1,Loop1
```

- Superscalar pipeline would send instructions one by one through EX, MEM, and WB
  - 1 ALU, 1 memory port, and 1 RF port would sit idle, perhaps through 10000+ loop iterations!

- How to improve the hardware utilization?

Thread-Level Parallelism (TLP)

- Refers to the parallelism among different *threads* (usually identified by the programmer)
  - A thread is a path of execution within a program
  - Each uses its own registers but they share the memory

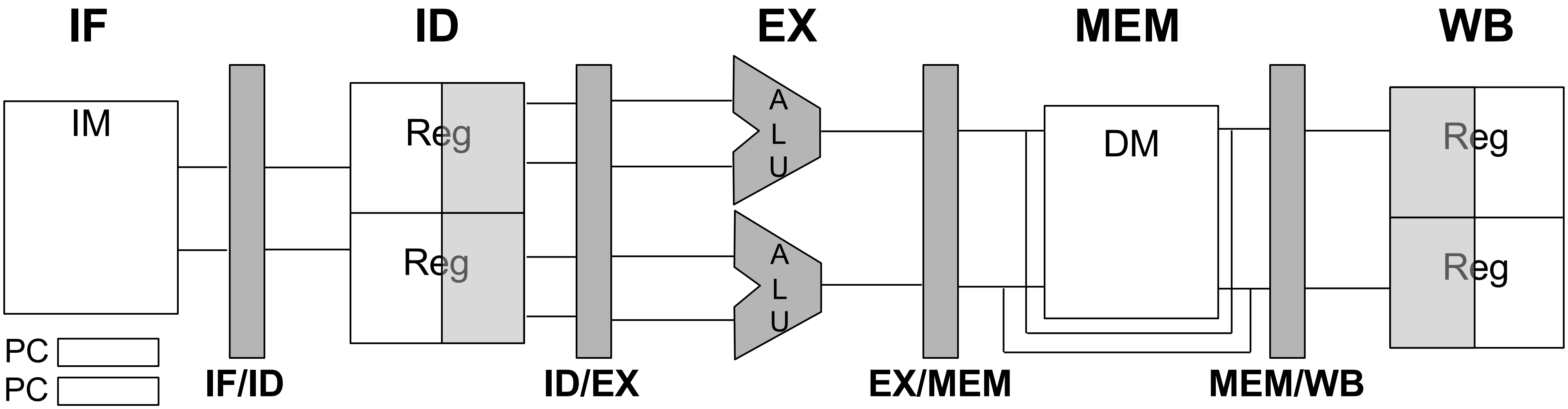
- Consider two threads that we want to run

```
             Thread 1                        Thread 2

Loop1: ADD R1,R2,R3             Loop2: LW R7,0(R1)
       OR R4,R1,R3                     ADD R4,R7,R2
       SUB R5,R4,R3                    SUBI R5,R4,1
       AND R1,R6,R5                    SW R5, 0(R1)
       BEQ R2,R1,Loop1                 BGEZ R5, Loop2
```

- We can run them on separate cores, or create a superscalar pipeline that can run them both at the same time

Two-Way *Multithreaded* Pipeline



- **Two threads share many of the pipeline resources (and virtual memory space)**
- **Each thread has its own PC and (physical) registers**
- **This is one example of multithreading, called *Simultaneous Multithreading (SMT)***

Data-Level Parallelism

- Consider the following C code

```
char A[4], B[4], C[4];
for (i = 0; i < 4; i++)
  A[i] = B[i] + C[i];
```

- Arrays *a, b, and c* contain four 8-bit elements
  - e.g., A[0], A[1], A[2], A[3] for array A

- Same operation is done for each data element

- Can replace the 4 add operations in the loop above by 1 ***SIMD*** add instruction

SIMD Instructions

- SIMD: *Single Instruction, Multiple Data*
- Special instructions for *vector* data (arrays)
- Identical operation is performed on each of the corresponding data elements
- Data elements are stored contiguously
  - 1 load (store) can read (write) all the elements at once
  - Register file is wide enough to hold all the elements in one register

Implementing the *for* Loop Using SIMD

```
char A[4], B[4], C[4];


for (i = 0; i < 4; i++)
  A[i] = B[i] + C[i];
```

↓

```
; load b and c from memory
LW      R0, 0(R4)   ; R4 points to B
LW      R1, 0(R5)   ; R5 points to C
; vector add
ADD.V   R2, R0, R1  ; one inst does four 8-bit adds!
; store result
SW      R2, 0(R3)   ; R3 points to A
```

Assume 32-bit registers and a 32-bit memory word (also note each `char’ variable holds 8 bits)

ILP, TLP, and DLP

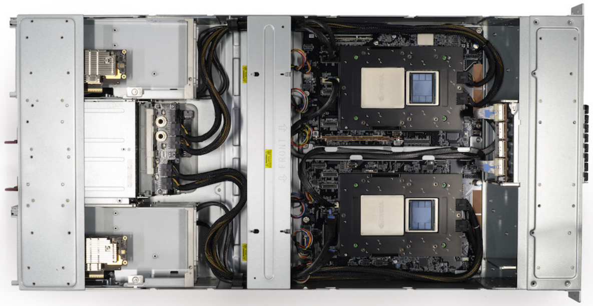
- Many processors exploit all three
  - Best performance/watt achieved with each in moderation rather than one/two to the extreme
- ILP
  - Typically 2 to 6-way superscalar pipeline
  - Performance improvement tapers off with wider pipelines while power may increase significantly
- TLP
  - Support for multiple threads may require small amount of additional hardware over single threaded SS pipeline
  - May improve hardware *efficiency* compared to SS alone
- DLP
  - Many applications (AI, graphics, video and audio processing, etc.) make this worthwhile

Ongoing Trends in Computer Systems

Which Computer is Faster, and By How Much?

Cray 1 Supercomputer
(1975)

vs.



NVIDIA GH200 NVL2 Server
(2025)

Iron law of Processor Performance

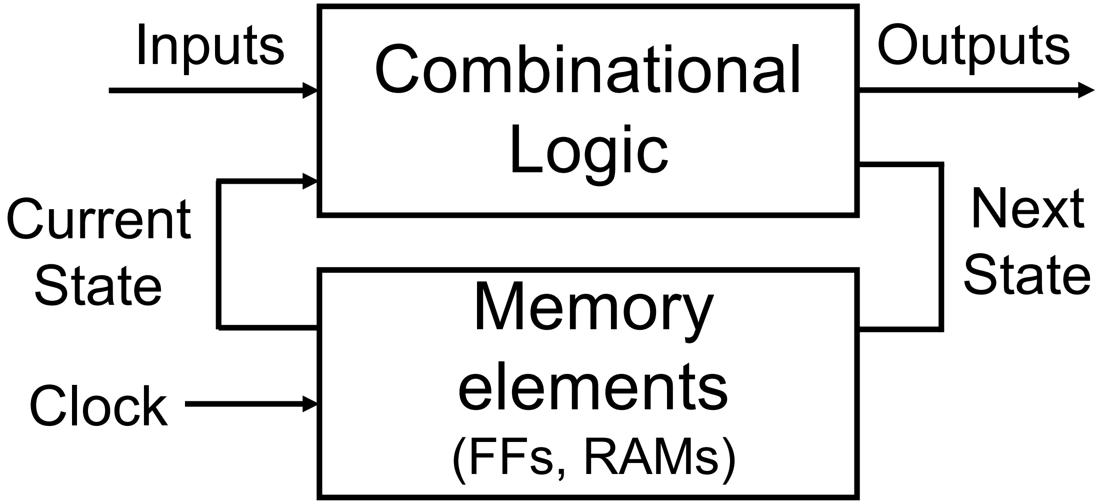
CPU Execution Time = I x CPI x CT

number of instructions in the program

average number of cycles per instruction

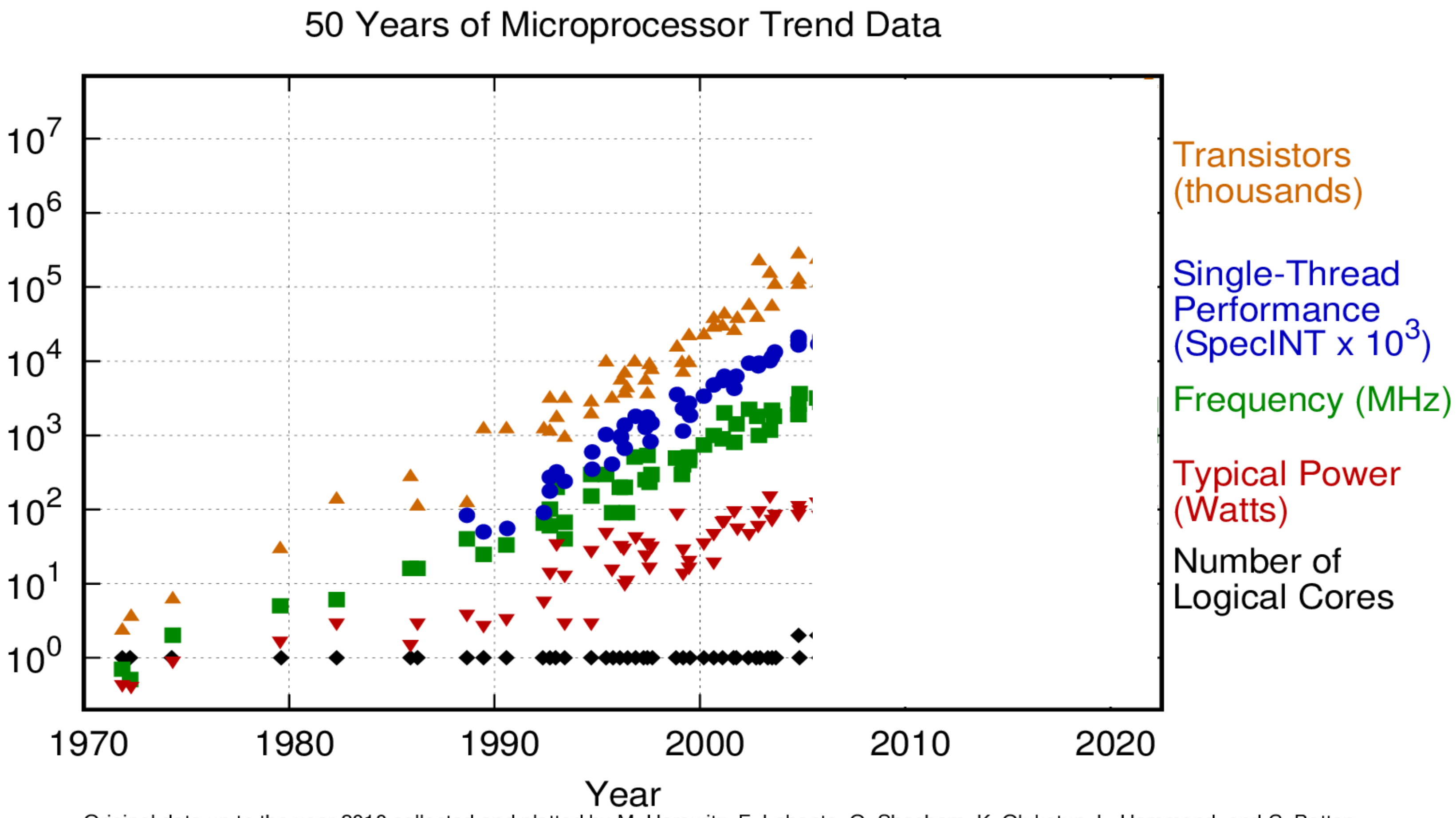
clock cycle time (1/frequency)

Revisiting Synchronous Circuits



- **The processor functions as a large state machine**
  - **The changes in the state of the memory elements are synchronized by a clock signal**
  - **A faster clock enables more operations (instructions) per second**

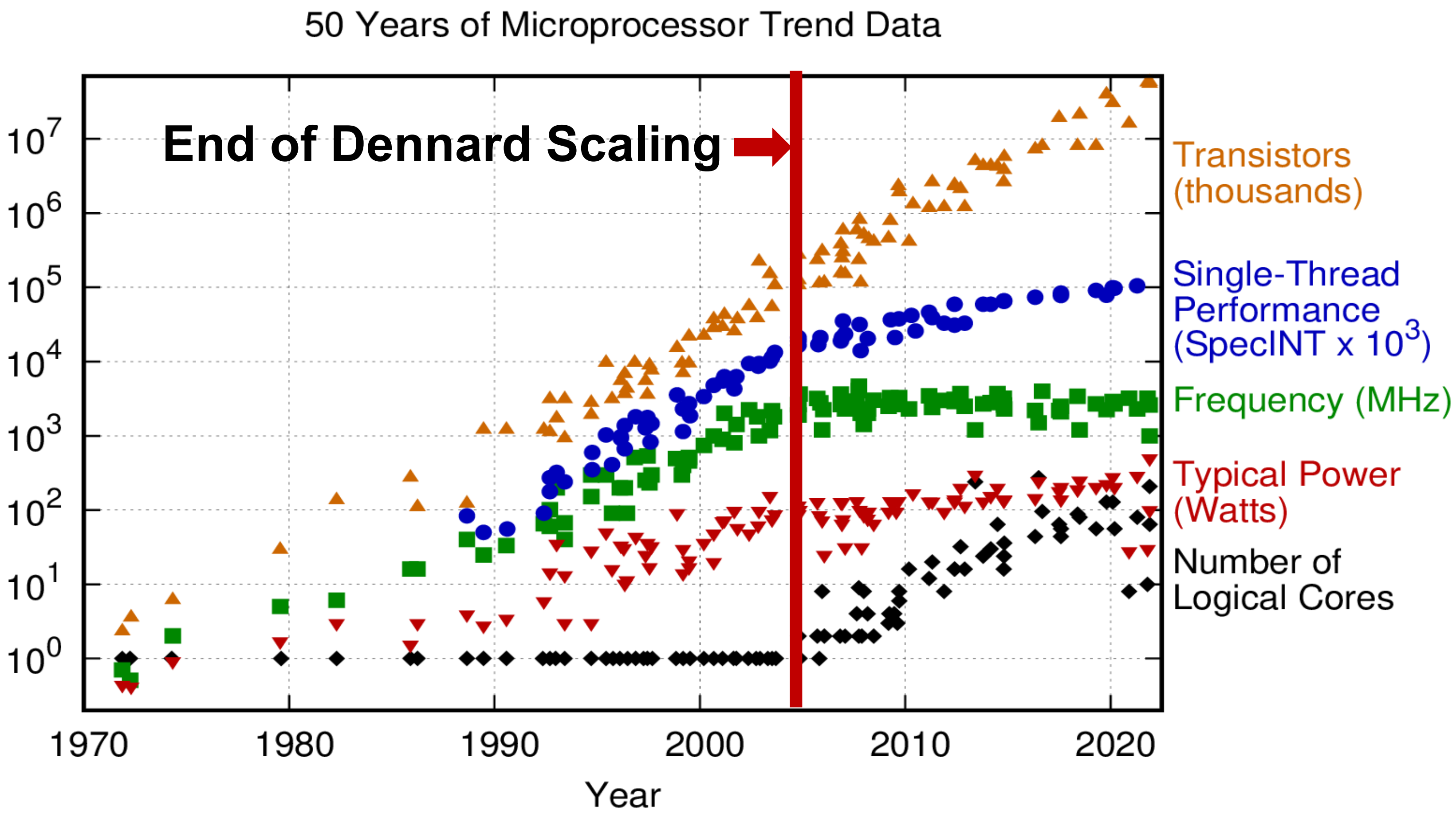
Microprocessor Scaling (pre-2005)



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

- **Transistor counts per chip doubled roughly every 2 years, following Moore's Law**
- **Clock frequencies increased exponentially, enabled by Dennard scaling**

Microprocessor Scaling (post-2005)

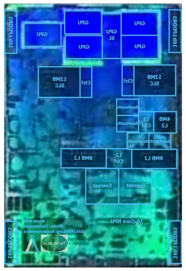


End of Dennard Scaling

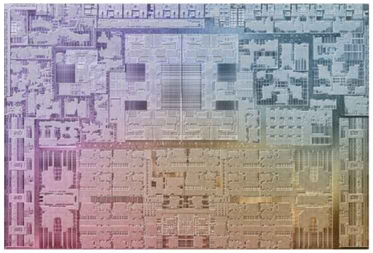
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

- **Transistor counts continue to scale**
- **Frequency scaling plateaued (end of Dennard scaling) => led to multicore & greater emphasis on energy efficiency**

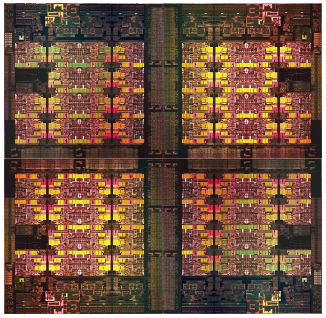
Era of Billion-Transistor Chips



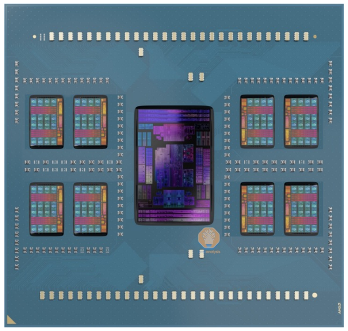
Apple A16
~16B transistors



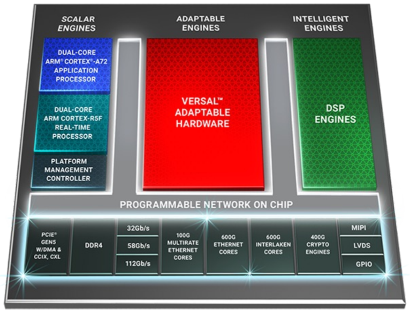
Apple M2 Pro
~40B transistors



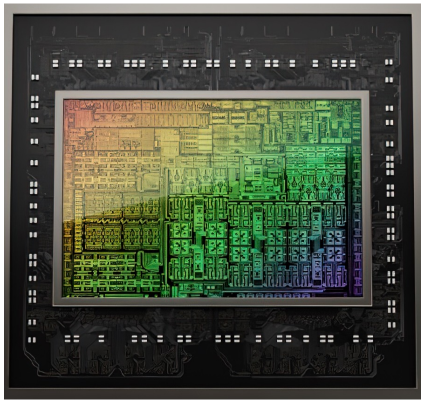
Intel Sapphire Rapids
(quad-chip module)
~48B transistors



AMD EPYC Bergamo
(9-chip module)
~82B transistors

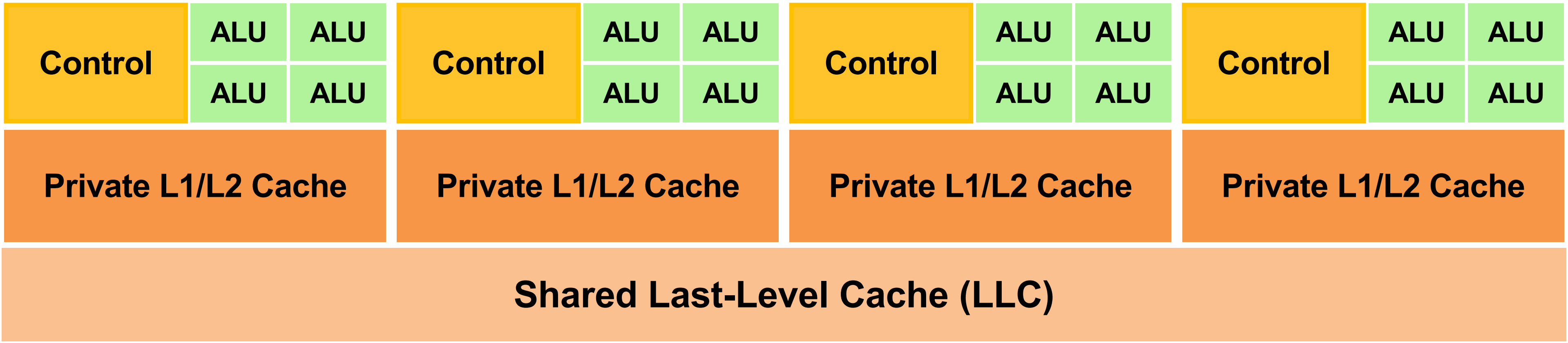

SCALAR ENGINES
ADAPTABLE ENGINES
INTELLIGENT ENGINES
DUAL-CORE ARM® CORTEX®-A72 APPLICATION PROCESSOR
DUAL-CORE ARM CORTEX-R5F REAL-TIME PROCESSOR
PLATFORM MANAGEMENT CONTROLLER
VERSAL™ ADAPTABLE HARDWARE
DSP ENGINES
PROGRAMMABLE NETWORK ON CHIP
PCIE® GEN5 W/DMA & CCIX, CXL
DDR4
32Gb/s
58Gb/s
112Gb/s
100G MULTIRATE ETHERNET CORES
600G ETHERNET CORES
600G INTERLAKEN CORES
400G CRYPTO ENGINES
MIPI
LVDS
GPIO

AMD (Xilinx) Versal Premium
~92B transistors



NVIDIA Blackwell B200
~208B transistors

Typical Multicore Architecture



Control | ALU ALU | ALU ALU

Control | ALU ALU | ALU ALU

Control | ALU ALU | ALU ALU

Control | ALU ALU | ALU ALU

Private L1/L2 Cache | Private L1/L2 Cache | Private L1/L2 Cache | Private L1/L2 Cache

Shared Last-Level Cache (LLC)

Do we expect a 4X speedup with 4 cores?

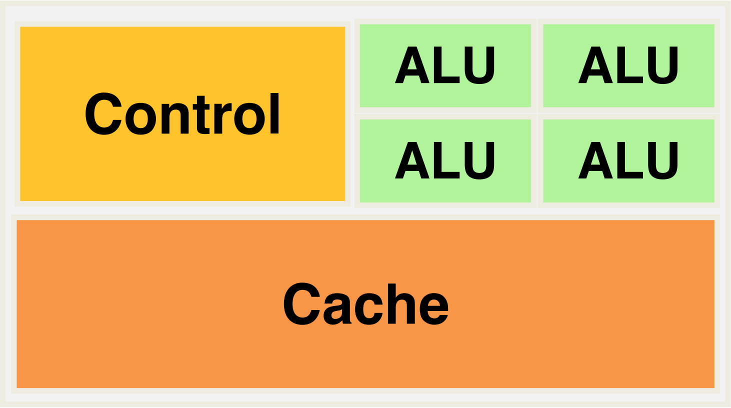
NO

- Per Amdahl’s Law, multicore speedup is limited by the serial part
- A multi-core processor typically runs at a lower frequency than a single big core due to power constraints

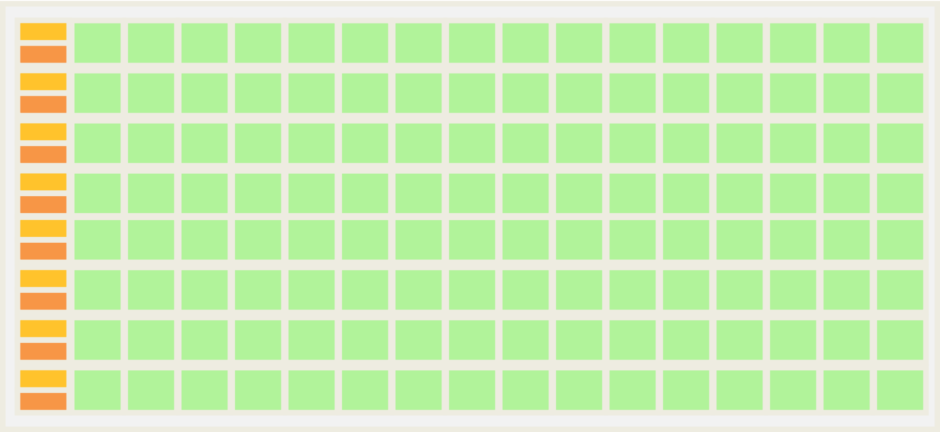
GPU Architecture

- GPU has thousands of cores to run many threads in parallel
  - Cores are simpler (compared to CPU)
  - No support of superscalar, OOO, speculative execution, etc.
  - ISA not backward compatible

- Optimized to increase throughput of running data-parallel applications
  - Initially targeting graphics code



CPU



GPU

Computing's Energy Problem

<<1W/chip

~1W/chip

~15W/chip

~50W/chip

~100W/chip

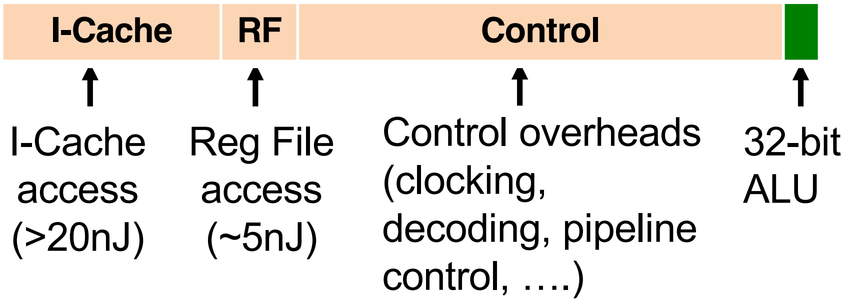
>100W/chip

$$Power = \frac{Energy}{Second} = \downarrow \boxed{\frac{Energy}{Op}} \times \boxed{\frac{Ops}{Second}} \uparrow$$

To increase performance (Ops/sec) in a power-constrained regime, energy per operation must decrease—in other words, energy efficiency (Ops/Joule) needs to improve!

Reducing Compute Energy Overhead

Energy breakdown of a typical instruction



I-Cache | RF | Control

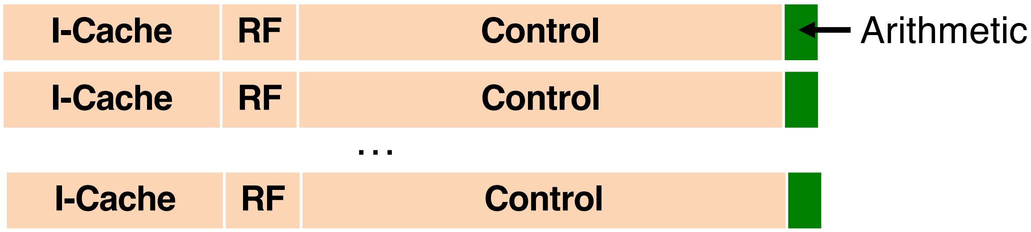
I-Cache access (>20nJ)

Reg File access (~5nJ)

Control overheads (clocking, decoding, pipeline control, ….)

32-bit ALU

A sequence of energy-inefficient instructions



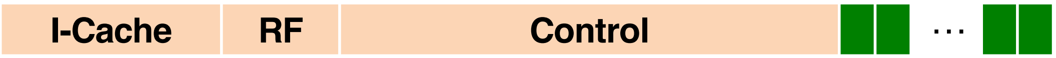
I-Cache | RF | Control ← Arithmetic

I-Cache | RF | Control

…

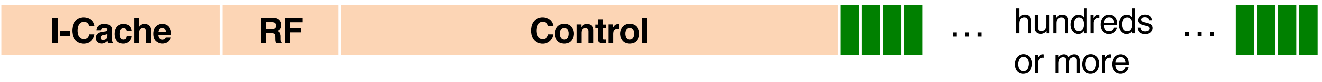
I-Cache | RF | Control

Single instruction multiple Data (SIMD): tens of operations per instruction



I-Cache | RF | Control …

Further specialization (what we achieve using accelerators)



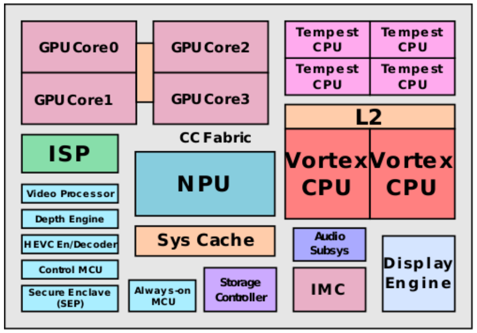
I-Cache | RF | Control … hundreds or more …

[Figure credit] Qadder, et al., Convolution Engine: Balancing Efficiency & Flexibility in Specialized Computing, ISCA'13.

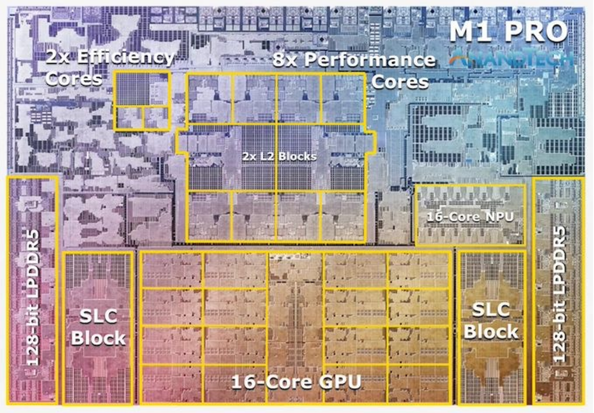
Era of Hardware Heterogeneity

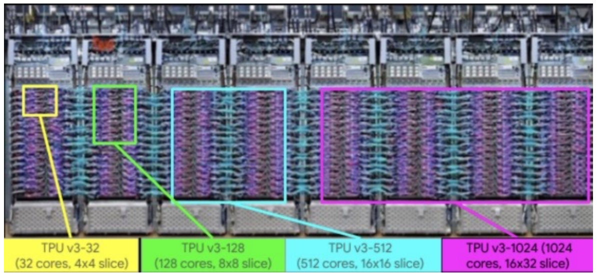
- Special-purpose accelerators are increasingly used to improve performance & energy efficiency in both cloud and edge/mobile environments

Apple 12 (iPhone X)

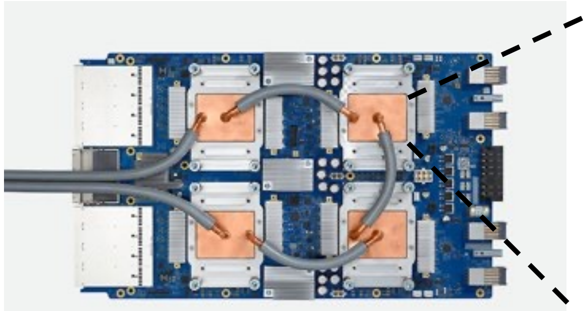


Apple M1 Pro

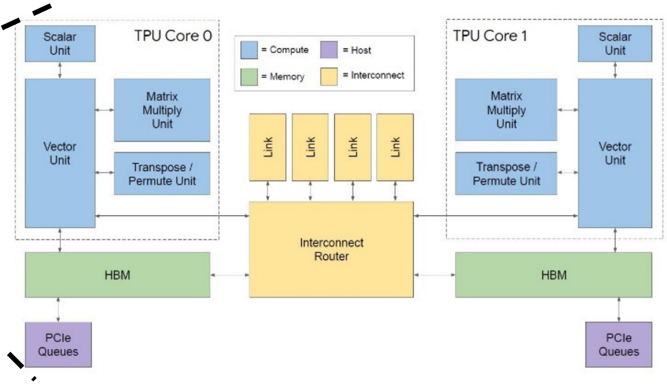




TPU v3-32 (32 cores, 4x4 slice)
TPU v3-128 (128 cores, 8x8 slice)
TPU v3-512 (512 cores, 16x16 slice)
TPU v3-1024 (1024 cores, 16x32 slice)

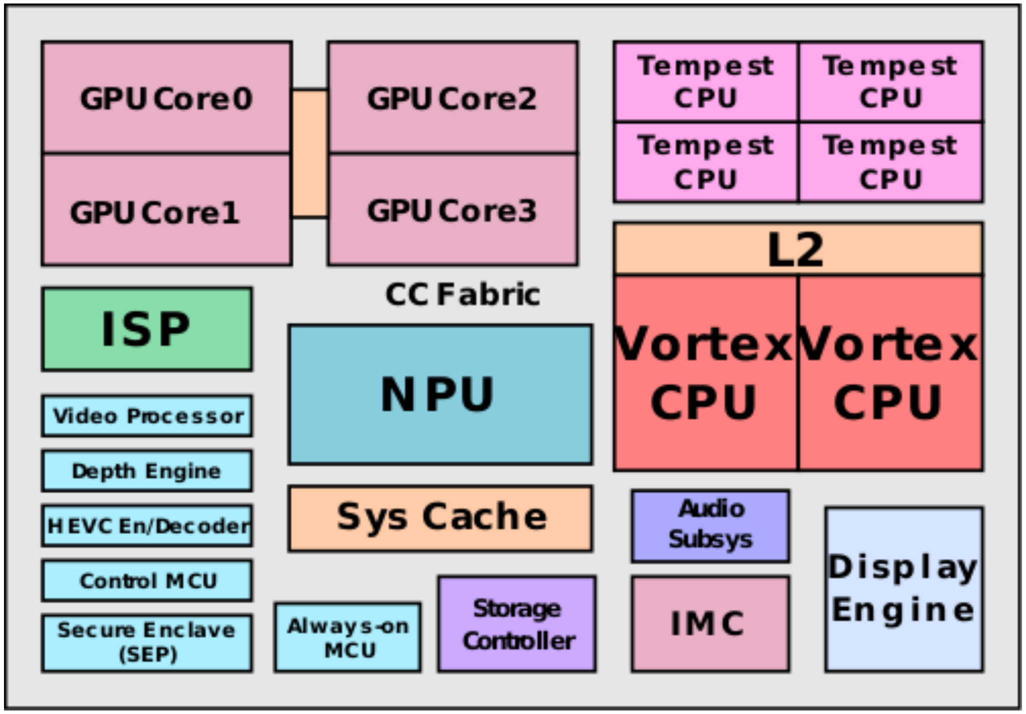


Google TPUv3



Hardware Specialization in Mobile Chips

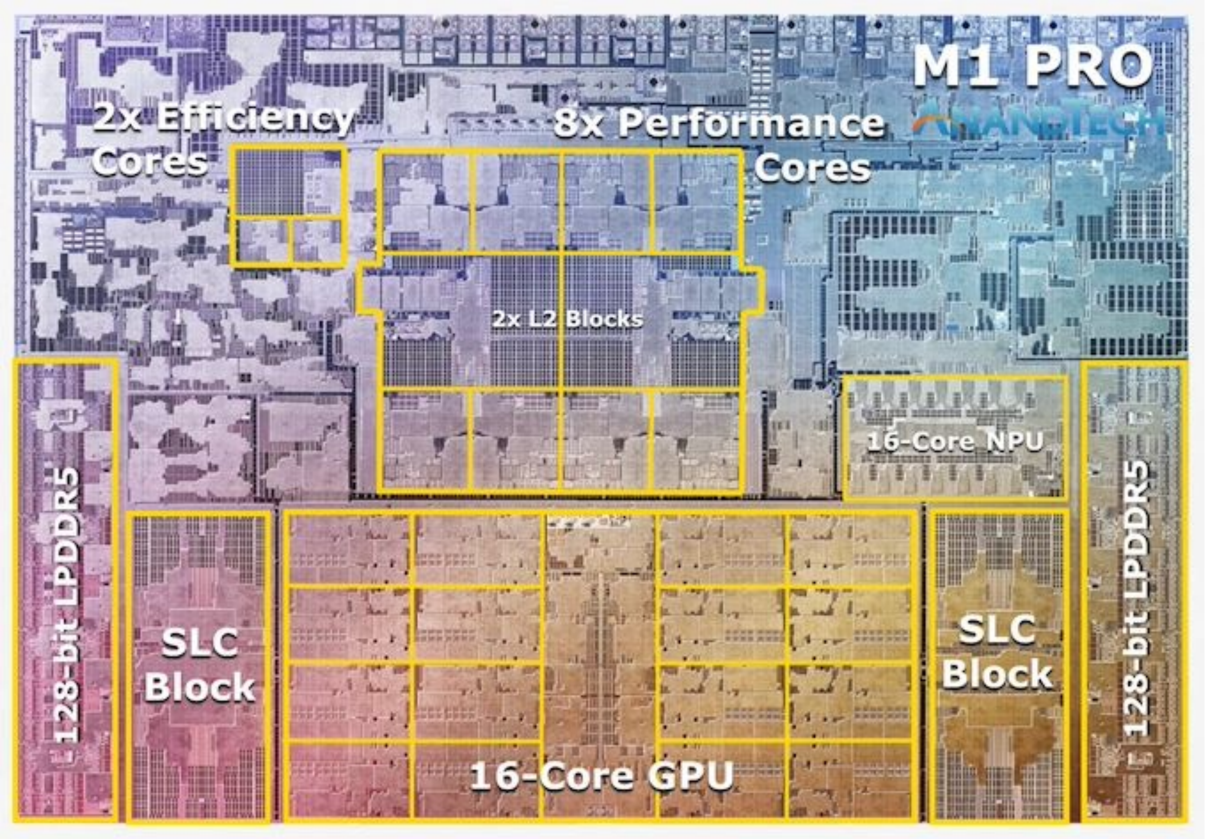
- Modern system-on-chips (SoCs) integrate a rich set of special-purpose hardware accelerators
  - Speed up critical tasks
  - Reduce power consumption and cost
  - Increase energy efficiency



Apple 12 (iPhone X)

HW Specialization in Laptop/Desktop Chips

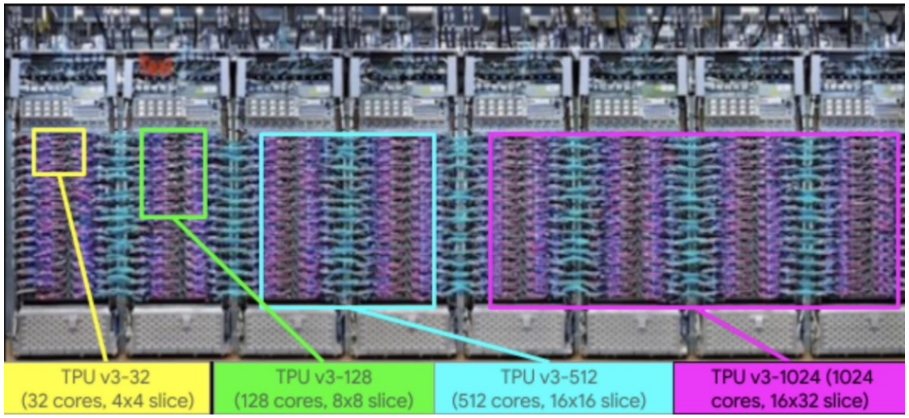
- Special-purpose hardware accelerators (e.g., GPUs, NPUs) improve performance and energy efficiency

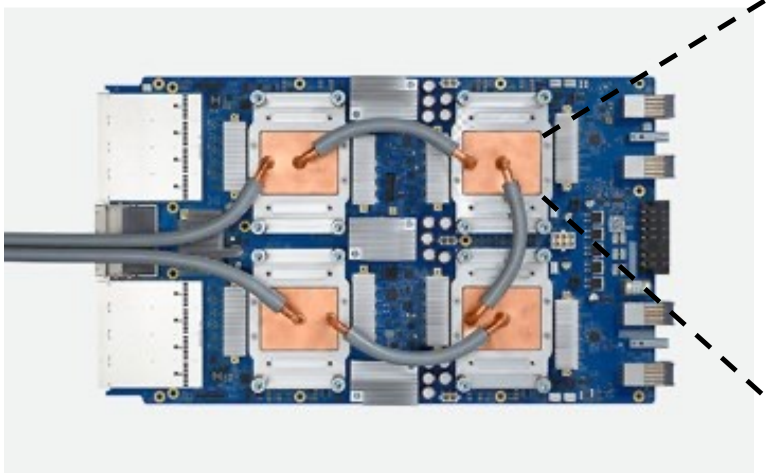


Apple M1 Pro SoC
(33.7B transistors)

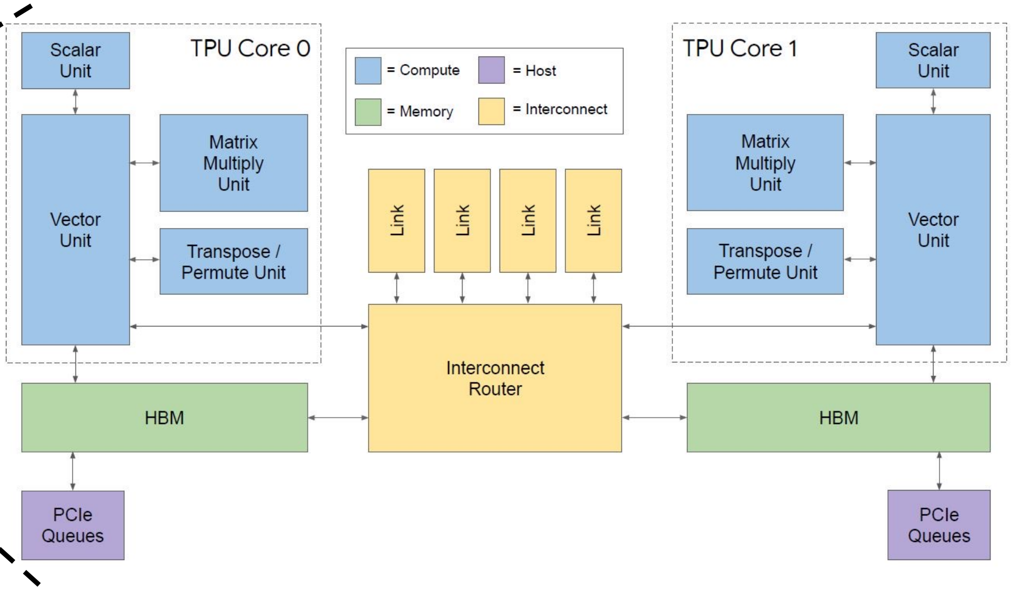
HW Specialization in Datacenter

- **ASIC- and FPGA-based accelerators are being deployed for a rich mix of compute-intensive applications in cloud datacenters**



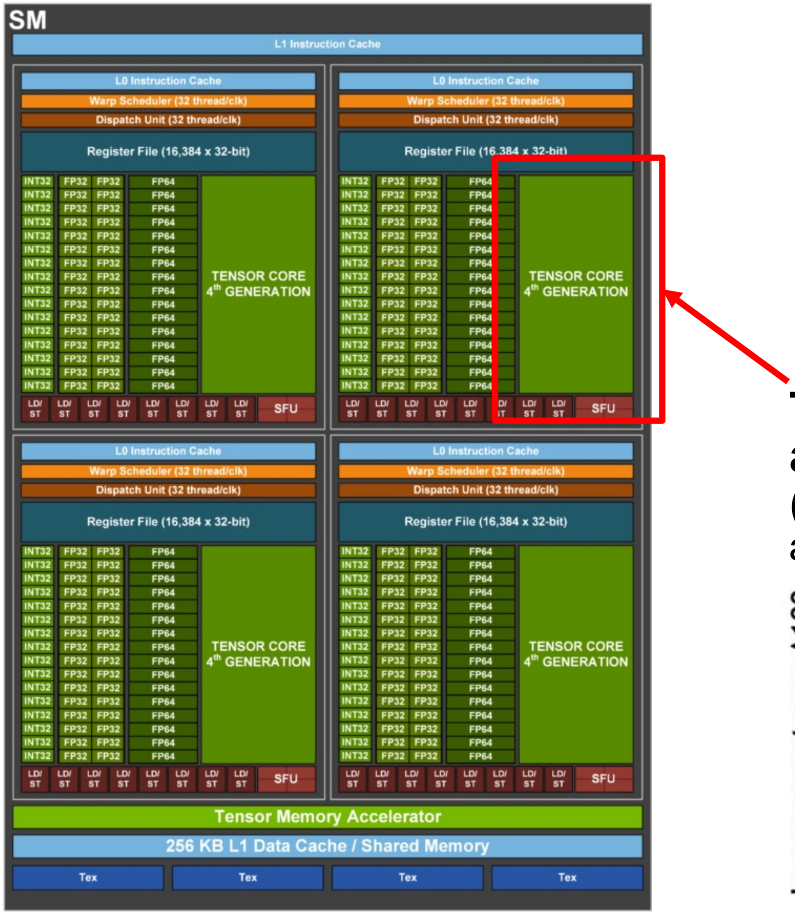


Google TPUv3 board

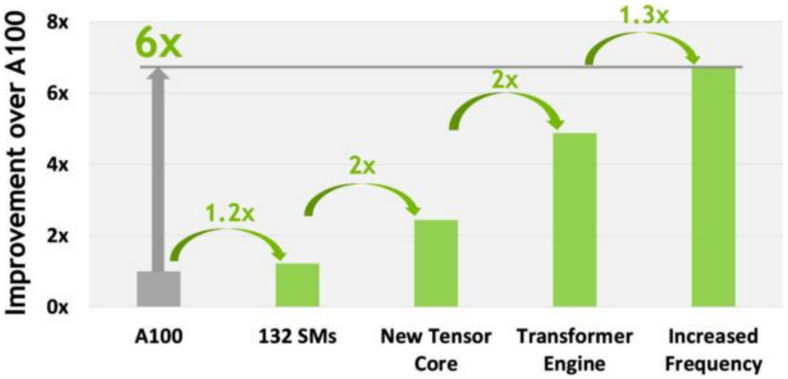


HW Specialization in GPUs

- Modern GPUs are increasingly specialized for AI workloads


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE 4th GENERATION
LD/ST
SFU
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

Tensor core in NVIDIA Hopper architecture
(WGMMA: Warp group matrix-multiply accumulation)

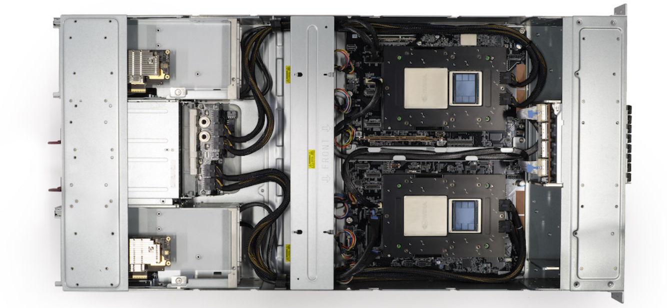


https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/
https://resources.nvidia.com/en-us-tensor-core/gtc22-whitepaper-hopper

The Incredible Advancements of Computer Hardware

Cray 1 Supercomputer
(1975)

160 Megaflops
115 kW



NVIDIA GH200 NVL2 Server
(2025)

8 PetaFLOPS
900-2000W

FLOPS stands for Floating Point Operations Per Second

Credit: slide adapted from Jonathan Ragan-Kelley's PLDI 2024 keynote

turing lecture

DOI:10.1145/3282307

Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.

BY JOHN L. HENNESSY AND DAVID A. PATTERSON

A New Golden Age for Computer Architecture

WE BEGAN OUR Turing Lecture June 4, 2018[11] with a review of computer architecture since the 1960s. In addition to that review, here, we highlight current challenges and identify future opportunities, projecting another golden age for the field of computer architecture in the next decade, much like the 1980s when we did the research that led to our award, delivering gains in cost, energy, and security, as well as performance.

> *"Those who cannot remember the past are condemned to repeat it."* —George Santayana, 1905

Software talks to hardware through a vocabulary called an instruction set architecture (ISA). By the early 1960s, IBM had four incompatible lines of computers, each with its own ISA, software stack, I/O system, and market niche—targeting small business, large business, scientific, and real time, respectively. IBM engineers, including ACM A.M. Turing Award laureate Fred Brooks, Jr., thought they could create a single ISA that would efficiently unify all four of these ISA bases.

They needed a technical solution for how computers as inexpensive as

» key insights

- **Software advances can inspire architecture innovation.**
- **Elevating the hardware/software interface creates opportunities for architecture innovation.**
- **The marketplace ultimately settles architecture debates.**

John Hennessy and David Patterson. "A New Golden Age for Computer Architecture." *Communications of the ACM*, 2019.

Follow-on Courses

- ECE 2400 / ENGRD 2140: Computer Systems Programming
- ECE 3140 / CS 3420: Embedded Systems
- ECE 4750 / CS 4420: Computer Architecture
- ECE 4740: Digital VLSI Design
- CS 4410: Operating Systems
- CS 4120: Compilers

- Fill out 2300 course evaluation

Next Time

Final Exam