Advanced Systems Lab

Spring 2022

Lecture: Architecture/Microarchitecture and Intel Core

Instructor: Markus Püschel, Ce Zhang

TA: Joao Rivera, several more

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Organization

Research project: Deadline *March 11th*

Finding team: fastcode-forum@lists.inf.ethz.ch

Today

Architecture/Microarchitecture: *What is the difference?*

In detail: Intel Skylake

Crucial microarchitectural parameters

Peak performance

Operational intensity

Brief: Apple M1 processor

Definitions

Architecture (also instruction set architecture = ISA): The parts of a processor design that one needs to understand to write assembly code

Examples: instruction set specification, registers

Counterexamples: cache sizes and core frequency

Example ISAs

- *x86*
- *MIPS*
- *POWER*
- *SPARC*
- *ARM*

Some assembly code

```
ipf:
    xorps   %xmm1, %xmm1
    xorl    %ecx, %ecx
    jmp     .L8
.L10:
    movslq  %ecx,%rax
    incl    %ecx
    movss (%rsi,%rax,4), %xmm0
    mulss (%rdi,%rax,4), %xmm0
    addss   %xmm0, %xmm1
.L8:
    cmpl    %edx, %ecx
    jl      .L10
    movaps  %xmm1, %xmm0
    ret
```

MMX:
Multimedia extension

SSE:
Streaming SIMD extension

AVX:
Advanced vector extensions

Backward compatible:
Old binary code (≥ 8086) runs on newer processors.

New code to run on old processors?
Depends on compiler flags.

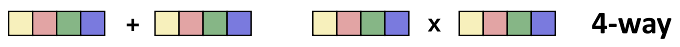
Intel x86		Processors (subset)
x86-16		8086
		286
x86-32		386
		486
		Pentium
	MMX	Pentium MMX
	SSE	Pentium III
	SSE2	Pentium 4
	SSE3	Pentium 4E
x86-64		Pentium 4F
		Core 2
	SSE4	*Penryn*
		Core i3/5/7
	AVX	*Sandy Bridge*
	AVX2	*Haswell*
	AVX-512	*Skylake-X*

time

ISA SIMD (Single Instruction Multiple Data) Vector Extensions

What is it?

- *Extension of the ISA. Data types and instructions for the parallel computation on short (length 2–8) vectors of integers or floats*



- *Names: MMX, SSE, SSE2, …, AVX, …*

Why do they exist?

- ***Useful:*** *Many applications have the necessary fine-grain parallelism*
  Then: speedup by a factor close to vector length
- ***Doable:*** *Chip designers have enough transistors to play with; easy to build with replication*

We will have an extra lecture on vector instructions

- *What are the problems?*
- *How to use them efficiently*

FMA = Fused Multiply-Add

$x = x + y \cdot z$

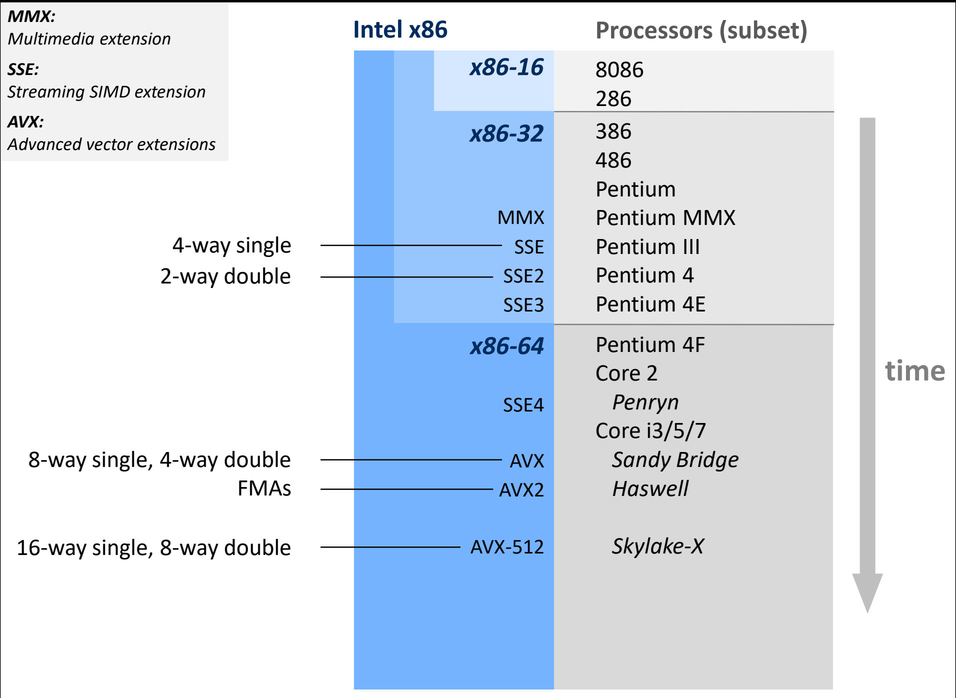
Done as one operation, i.e., involves only one rounding step

Better accuracy than sequence of mult and add

Natural pattern in many algorithms

```
/* matrix multiplication; A, B, C are n x n matrices of doubles */
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < n; k++)
      C[i*n+j] += A[i*n+k]*B[k*n+j];
```

Exists only recently in Intel processors *(Why?)*



Definitions

Microarchitecture: Implementation of the architecture

Examples: Caches, cache structure, CPU frequency, details of the virtual memory system

Examples
- *Intel processors (Wikipedia)*
- *AMD processors (Wikipedia)*

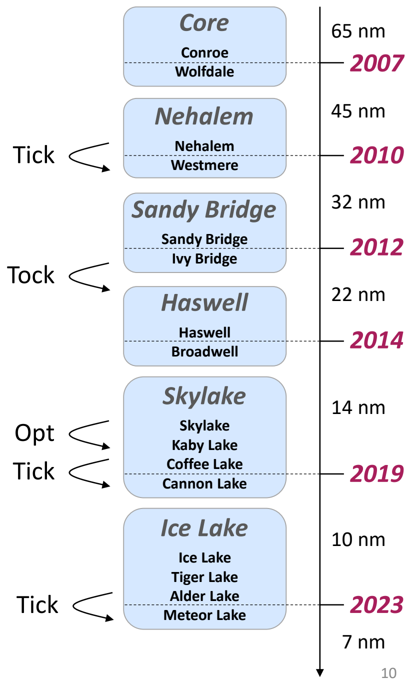
Intel's Tick-Tock Model

Tick: Shrink of process technology

Tock: New microarchitecture

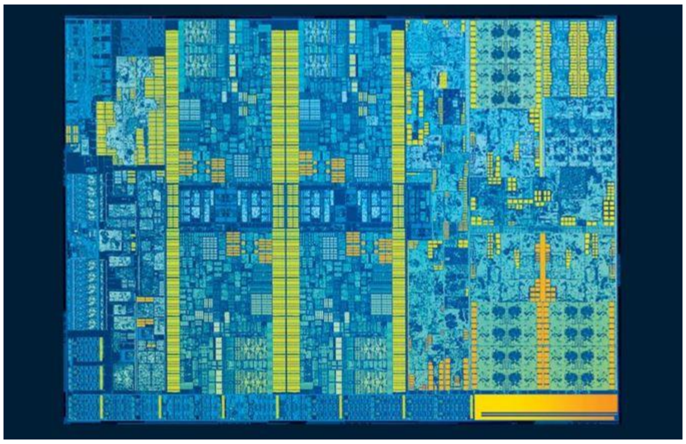
2016: Tick-tock model got discontinued
Now:
process (tick)
architecture (tock)
optimization (opt)

Example: Core and successors
Shown: Intel's microarchitecture code names (server/mobile may be different)



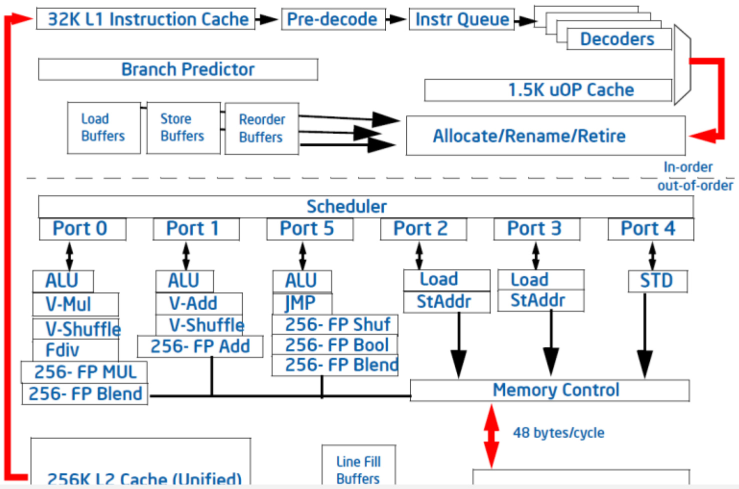
Pictures: Intel

Intel Processors: Example Skylake


http://www.anandtech.com

Detailed information about Intel processors

Microarchitecture: The View of the Computer Architect



we take the software developer's view …

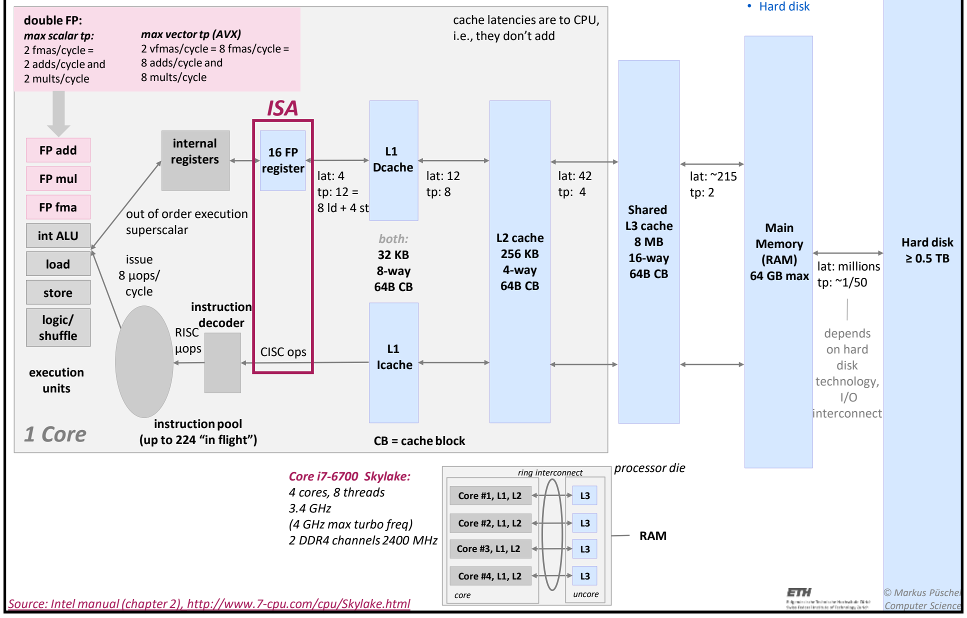
Source: Intel Architectures Optimization Reference Manual

Distribute microarchitecture abstraction

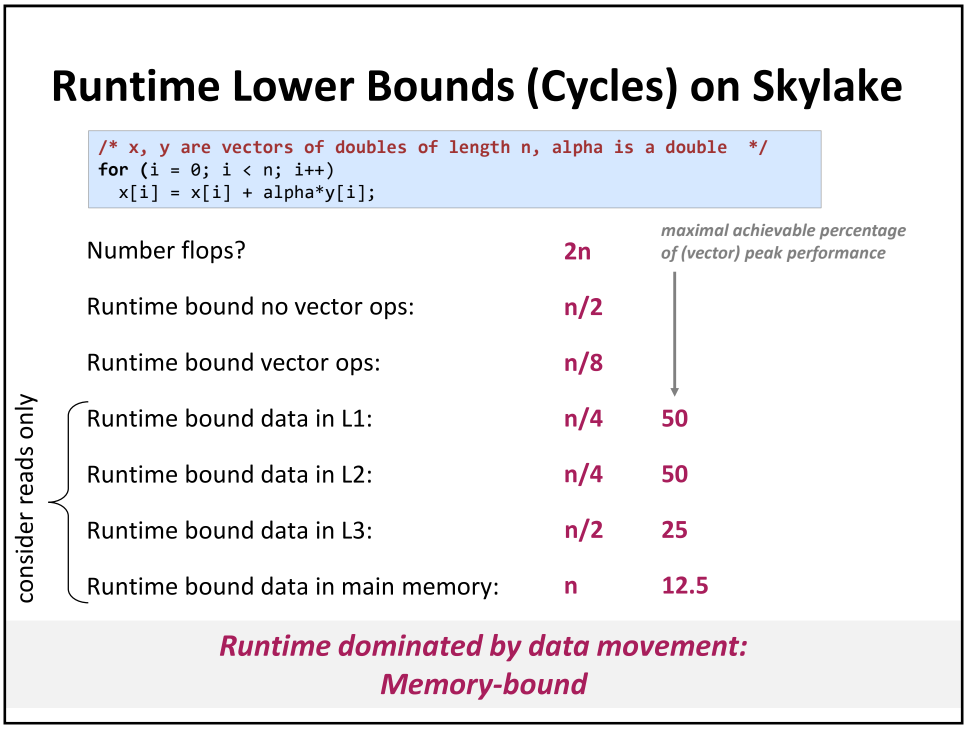
Abstracted Microarchitecture: Example Core i7 Skylake (2015)

Throughput (tp) is measured in doubles/cycle. For example: 4. *Numbers are for loading into registers.*
Latency (lat) is measured in cycles
1 double floating point (FP) = 8 bytes
fma = fused multiply-add
Rectangles not to scale

Memory hierarchy:
- Registers
- L1 cache
- L2 cache
- L3 cache
- Main memory
- Hard disk



Source: Intel manual (chapter 2), http://www.7-cpu.com/cpu/Skylake.html



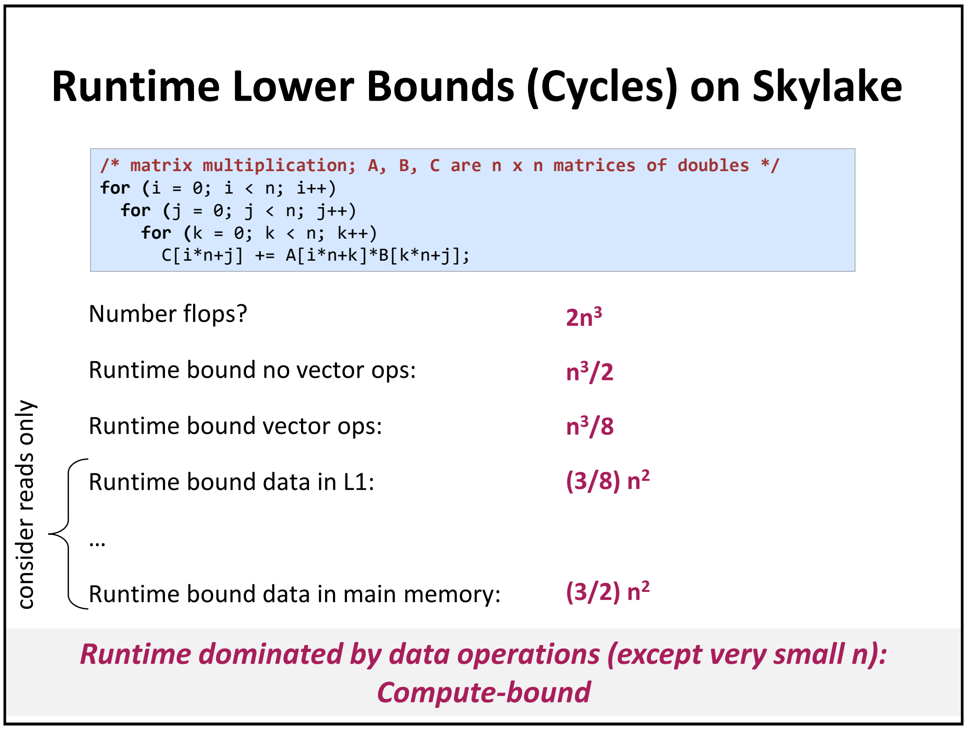
Runtime Lower Bounds (Cycles) on Skylake

```
/* x, y are vectors of doubles of length n, alpha is a double  */
for (i = 0; i < n; i++)
  x[i] = x[i] + alpha*y[i];
```

		maximal achievable percentage of (vector) peak performance
Number flops?	2n	
Runtime bound no vector ops:	n/2	
Runtime bound vector ops:	n/8	
Runtime bound data in L1:	n/4	50
Runtime bound data in L2:	n/4	50
Runtime bound data in L3:	n/2	25
Runtime bound data in main memory:	n	12.5

consider reads only

Runtime dominated by data movement:
Memory-bound



Runtime Lower Bounds (Cycles) on Skylake

```
/* matrix multiplication; A, B, C are n x n matrices of doubles */
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < n; k++)
      C[i*n+j] += A[i*n+k]*B[k*n+j];
```

Number flops?	$2n^3$
Runtime bound no vector ops:	$n^3/2$
Runtime bound vector ops:	$n^3/8$
Runtime bound data in L1:	$(3/8)\ n^2$
…	
Runtime bound data in main memory:	$(3/2)\ n^2$

consider reads only

Runtime dominated by data operations (except very small n):
Compute-bound

Operational Intensity

Definition: Given a program P, assume cold (empty) cache

$$\textit{Operational intensity: } I(n) = \frac{W(n)}{Q(n)}$$

#flops (input size n)

#bytes transferred cache ↔ memory (for input size n)

Lower bounds for Q(n) yield upper bounds for I(n)

Sometimes we only consider reads from memory $Q_{read}(n) \le Q(n)$ and thus $I_{read}(n) \ge I(n)$

Operational Intensity (Cold Cache)

```
/* x, y are vectors of doubles of length n, alpha is a double  */
for (i = 0; i < n; i++)
  x[i] = x[i] + alpha*y[i];
```

Operational intensity (reads only):

- *Flops: W(n)* **= 2n**
- *Memory transfers (doubles):* **≥ 2n (just from the reads)**
- *Reads (bytes): $Q_{read}(n)$* **≥ 16n**
- *Operational intensity: $I(n) \le I_{read}(n)$* **$= W(n)/Q_{read}(n) \le 1/8$**

Operational Intensity (Cold Cache)

```
/* matrix multiplication; A, B, C are n x n matrices of doubles */
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < n; k++)
      C[i*n+j] += A[i*n+k]*B[k*n+j];
```

Operational intensity (reads only):

▪ *Flops: $W(n)$*	**$= 2n^3$**
▪ *Memory transfers (doubles):*	**$\geq 3n^2$ (just from the reads)**
▪ *Reads (bytes): $Q_{read}(n)$*	**$\geq 24n^2$**
▪ *Operational intensity: $I(n) \leq I_{read}(n)$*	**$= W(n)/Q_{read}(n) \leq n/12$**

Compute/Memory Bound

A function/piece of code is:

- ***Compute bound*** *if it has high operational intensity*
- ***Memory bound*** *if it has low operational intensity*

A more exact definition depends on the given platform

More details later: Roofline model

Superscalar Processor

Definition: A superscalar processor can issue and execute *multiple instructions in one cycle*. The instructions are retrieved from a sequential instruction stream and are usually scheduled dynamically.

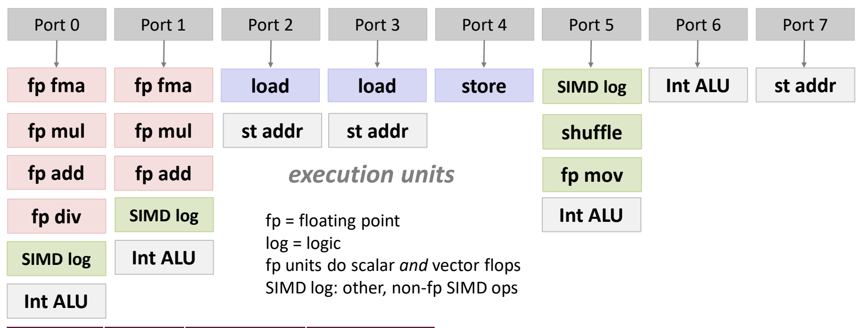
Benefit: Superscalar processors can take advantage of *instruction level parallelism (ILP)* that many programs have

Most CPUs since about 1998 are superscalar

Intel: since Pentium Pro

Simple embedded processors are usually not superscalar

Execution Units and Ports (Skylake)



fp = floating point
log = logic
fp units do scalar *and* vector flops
SIMD log: other, non-fp SIMD ops

Execution Unit (fp)	Latency [cycles]	Throughput [ops/cycle]	Gap [cycles/issue]
fma	4	2	0.5
mul	4	2	0.5
add	4	2	0.5
div (scalar) div (4-way)	14 14	1/4 1/8	4 8

- Every port can issue one instruction/cycle
- Gap = 1/throughput
- ***Intel says gap for the throughput!***
- Same exec units for scalar and vector flops
- Same latency/throughput for scalar (one double) and AVX vector (four doubles) flops, except for div

Source: Intel manual (Table C-8. 256-bit AVX Instructions, Table 2-1. Dispatch Port and Execution Stacks of the Skylake Microarchitecture, Figure 2-1. CPU Core Pipeline Functionality of the Skylake Microarchitecture)

Notes on Previous Slide

The availability of more than one port makes the processor superscalar

Execution units behind different ports can start an operation in the same cycle (superscalar)

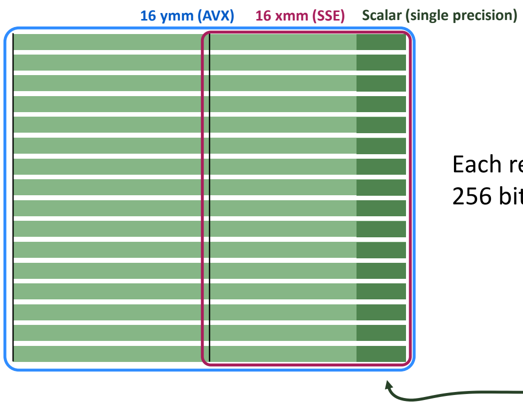
Execution units behind the same port cannot start an operation in the same cycle

Adds, Mults and FMAs have throughput of 2 because they have 2 units behind 2 different ports. Each of these units has a throughput of 1

An execution unit with throughput 1 is called fully pipelined

By default the compiler does not use FMAs for single adds or mults

Floating Point Registers

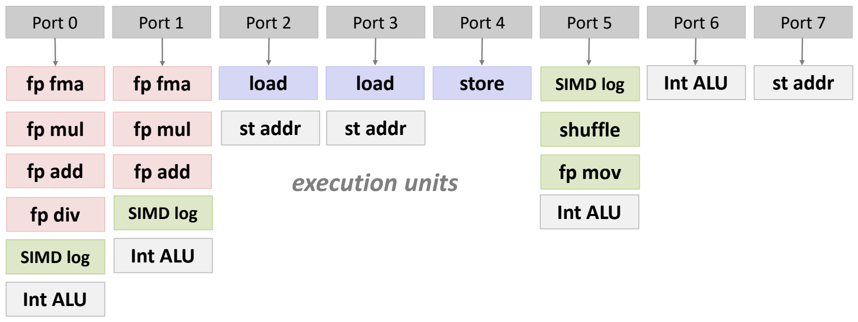


Each register:
256 bits = 4 doubles = 8 singles

Same 16 registers for scalar FP, SSE and AVX

Scalar (non-vector) single precision FP code uses the bottom eighth

Explains why throughput and latency is usually the same for vector and scalar operations



How many cycles are at least required (no vector ops)?

A function with n adds and n mults in the C code	*n/2*
A function with n add and n mult instructions in the assembly code	*n*
A function with n adds in the C code	*n/2*
A function with n add instructions in the assembly code	*n/2*
A function with n adds and n/2 mults in the C code	*n/2*

Comments on Intel Skylake Lake µarch

Peak performance 16 double precision flops/cycle (only reached if SIMD FMA)

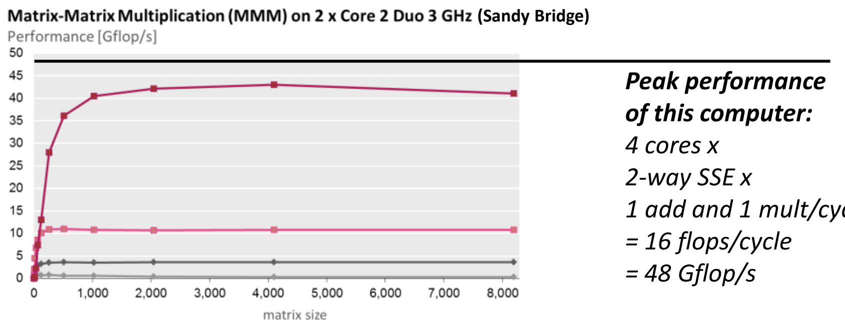
- *Peak performance mults: 2 mults/cycle (scalar 2 flops/cycle, SIMD AVX 8 flops/cycle)*
- *Peak performance adds: 2 adds/cycle (scalar 2 flop/cycle, SIMD AVX 8 flops/cycle)*

L1 bandwidth: two 32-byte loads *and* one 32-byte store per cycle

Shared L3 cache organized as multiple cache slices for better scalability with number of cores, thus access time is non-uniform

Shared L3 cache in a different clock domain (uncore)

Example: Peak Performance



Peak performance of this computer:
4 cores x
2-way SSE x
1 add and 1 mult/cycle
= 16 flops/cycle
= 48 Gflop/s

About M1 Processor

Release: November 2020
8 cores: 4 High-performance and 4 energy-efficient cores
ISA: ARMv8.4 (includes NEON 128-bit vector extension)
Microarchitecture: *Firestorm* (High-perf) and *Icestorm* (Energy-efficient)
Technology: 5nm

Successors (released in October 2021):

- *M1 Pro: Same as M1 but with 10 cores and twice L3 cache.*
- *M1 Max: Same as M1 Pro but with twice L3 cache (also bigger GPU).*

Cache	M1 (Firestorm)	M1 (Icestorm)
L1-I	192K	128K
L1-D	128K	64K
L2	12M Shared by four Firestorm cores	4M Shared by Icestorm cores
L3 (SLC)	16M, 24M (Pro) or 48M (Max). Shared by all cores and GPU.	

https://en.wikichip.org/wiki/apple/mx/m1
https://en.wikipedia.org/wiki/Apple_M1

Firestorm Microarchitecture

Integer ports:
1: alu + flags + branch + addr + msr/mrs nzcv + mrs
2: alu + flags + branch + addr + msr/mrs nzcv + ptrauth
3: alu + flags + mov-from-simd/fp?
4: alu + mov-from-simd/fp?
5: alu + mul + div
6: alu + mul + madd + crc + bfm/extr

Load and store ports:
7: store + amx
8: load/store + amx
9: load
10: load

FP/SIMD ports:
11: fp/simd
12: fp/simd
13: fp/simd + fcsel + to-gpr
14: fp/simd + fcsel + to-gpr + fcmp/e + fdiv + …

Instruction	Latency [cycles]	Gap [cycles/issue]
add	3	0.25
mul	4	0.25
div	10	1
load		0.33
store		0.5

Latency and gap of FP instructions in double precision. The numbers are the same for scalar and vector instructions.

This information is based on black-box reverse engineering.

https://dougallj.github.io/applecpu/firestorm.html

Icestorm Microarchitecture

Integer ports:
1: alu + br + mrs
2: alu + br + div + ptrauth
3: alu + mul + bfm + crc

Load and store ports:
4: load/store + amx
5: load

FP/SIMD ports:
6: fp/simd
7: fp/simd + fcsel + to-gpr + fcmp/e + fdiv + …

Instruction	Latency [cycles]	Gap [cycles/issue]
add	3	0.5
mul	4	0.5
div (scalar) div (2-way)	10 11	1 2
load		0.5
store		1

Latency and gap of FP instructions in double precision. The numbers are the same for scalar and vector instructions except for div.

This information is based on black-box reverse engineering.

https://dougallj.github.io/applecpu/icestorm.html

Apple’s Microarchitectures

Example: A series (used in iPhones and iPads)

Year	Microarchitecture	Technology
2013	Cyclone	28 nm
2014	Typhoon	20 nm
2015	Twister	16 nm
2016	Hurricane, Zephyr	10 nm
2017	Monsoon, Mistral	10 nm
2018	Vortex, Tempest	7 nm
2019	Lightning, Thunder	7 nm
2020	Firestorm, Icestorm	5 nm
2021	Avalanche, Blizzard	5 nm

Firestorm, Icestorm are the only ones currently used in M series processors (used for MacBook, iMacs and iPad Pro)

https://en.wikipedia.org/wiki/Apple_silicon

Summary

Architecture vs. microarchitecture

To optimize code one needs to understand a suitable abstraction of the microarchitecture and its key quantitative characteristics

- *Memory hierarchy with throughput and latency info*
- *Execution units with port, throughput, and latency info*

Operational intensity:

- *High = compute bound = runtime dominated by data operations*
- *Low = memory bound = runtime dominated by data movement*