Advanced Systems Lab

Spring 2021

Lecture: Architecture/Microarchitecture and Intel Core

Instructor: Markus Püschel, Ce Zhang

TA: Joao Rivera, several more

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Organization

Research project: Deadline ***March 12th***

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

Today

Architecture/Microarchitecture: *What is the difference?*

In detail: Intel Haswell and Sandybridge

Crucial microarchitectural parameters

Peak performance

Operational intensity

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*
- *ia*
- *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
```

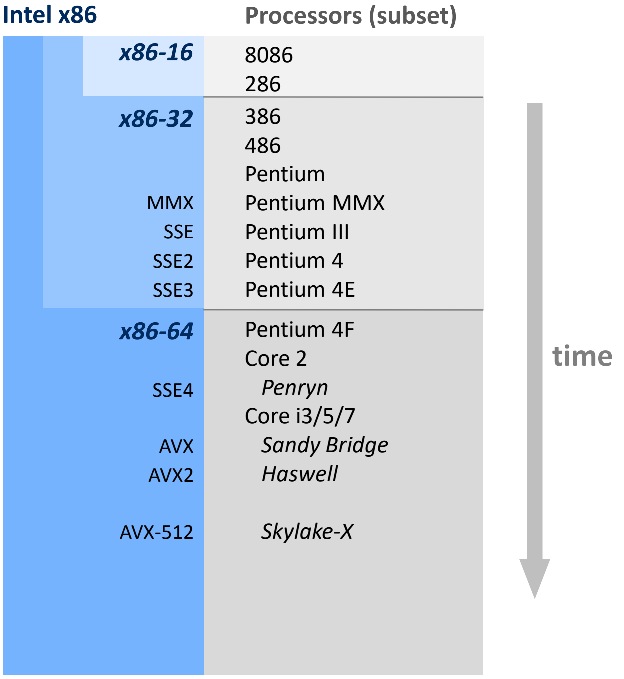
MMX:
Multimedia extension

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.

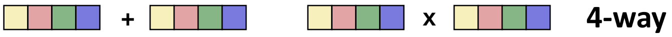


Intel x86	Processors (subset)
x86-16	8086
	286
x86-32	386
	486
	Pentium
MMX	Pentium MMX
SSE	Pentium III
SSE2	Pentium 4
SSE3	Pentium 4E
x86-64	Pentium 4F
	Core 2
SSE4	*Penryn*
	Core i3/5/7
AVX	*Sandy Bridge*
AVX2	*Haswell*
AVX-512	*Skylake-X*

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

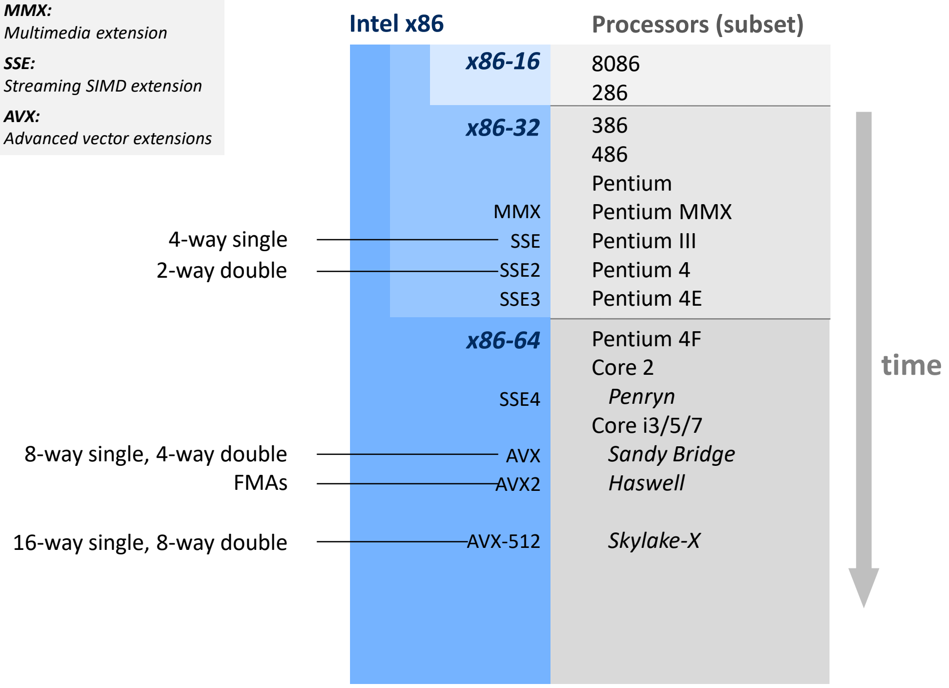
```
/* matrix multiplication; A, B, C are n x n matrices of doubles */
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < n; k++)
      C[i*n+j] += A[i*n+k]*B[k*n+j];
```

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

MMX:
Multimedia extension

SSE:
Streaming SIMD extension

AVX:
Advanced vector extensions



	Intel x86	Processors (subset)
	x86-16	8086
		286
	x86-32	386
		486
		Pentium
	MMX	Pentium MMX
4-way single	SSE	Pentium III
2-way double	SSE2	Pentium 4
	SSE3	Pentium 4E
	x86-64	Pentium 4F
		Core 2
	SSE4	*Penryn*
		Core i3/5/7
8-way single, 4-way double	AVX	*Sandy Bridge*
FMAs	AVX2	*Haswell*
16-way single, 8-way double	AVX-512	*Skylake-X*

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)*

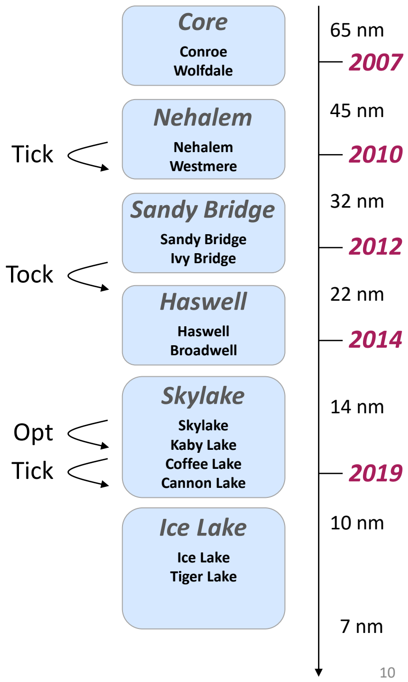
Intel's Tick-Tock Model

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)



Core: Conroe, Wolfdale — 65 nm — 2007

Tick

Nehalem: Nehalem, Westmere — 45 nm — 2010

Tock

Sandy Bridge: Sandy Bridge, Ivy Bridge — 32 nm — 2012

Haswell: Haswell, Broadwell — 22 nm — 2014

Opt

Tick

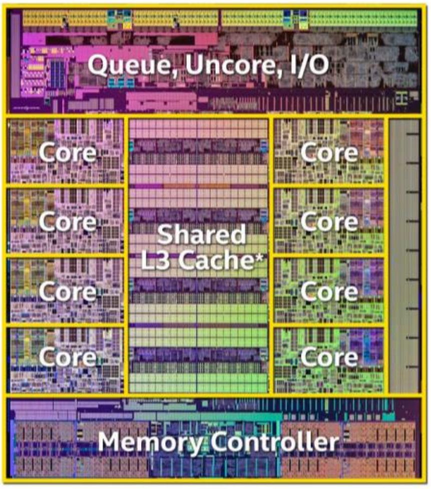
Skylake: Skylake, Kaby Lake, Coffee Lake, Cannon Lake — 14 nm — 2019

Ice Lake: Ice Lake, Tiger Lake — 10 nm

7 nm

Pictures: Intel

Intel Processors: Example Haswell

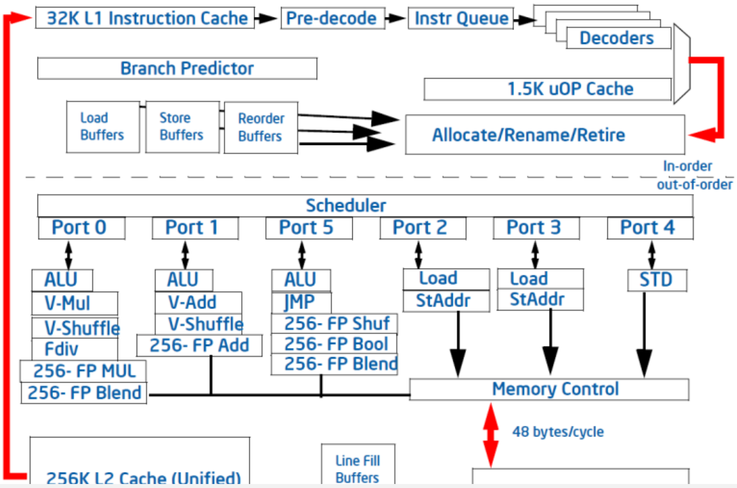


http://www.anandtech.com

Intel® Core™ i7-5960X Processor Extreme Edition
Transistor count: 2.6 Billion
Die size: 17.6mm x 20.2mm

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

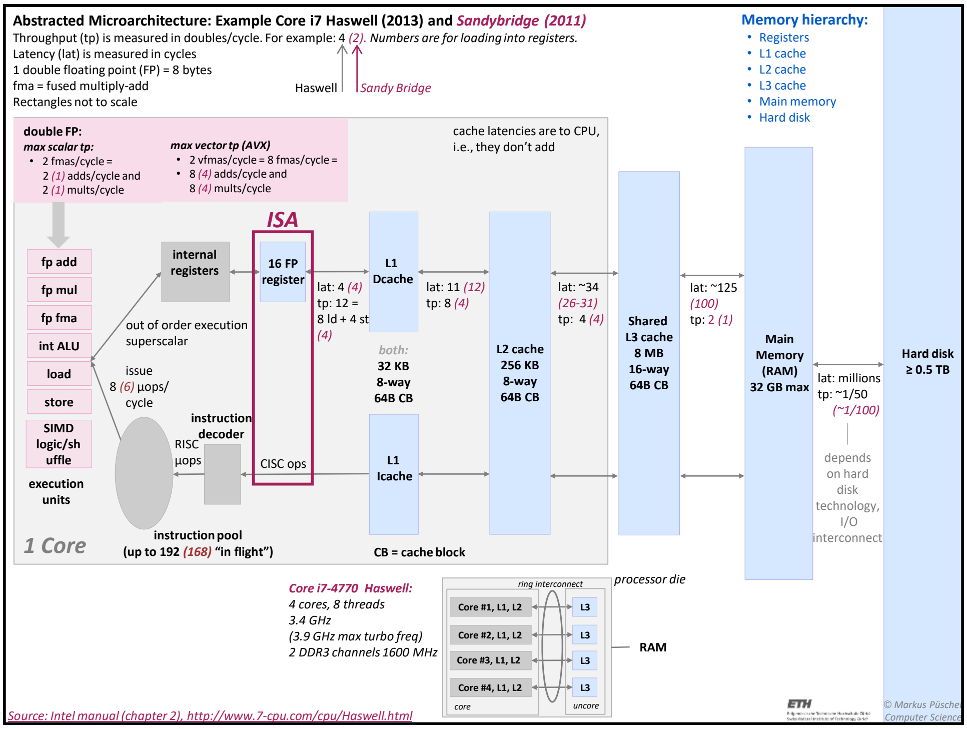
Abstracted Microarchitecture: Example Core i7 Haswell (2013) and *Sandybridge (2011)*

Throughput (tp) is measured in doubles/cycle. For example: 4 *(2). 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

Haswell *Sandy Bridge*

Memory hierarchy:

- Registers
- L1 cache
- L2 cache
- L3 cache
- Main memory
- Hard disk



double FP:

max scalar tp:

- 2 fmas/cycle = 2 *(1)* adds/cycle and 2 *(1)* mults/cycle

max vector tp (AVX)

- 2 vfmas/cycle = 8 fmas/cycle =
- 8 *(4)* adds/cycle and 8 *(4)* mults/cycle

cache latencies are to CPU, i.e., they don't add

ISA

fp add
fp mul
fp fma
int ALU
load
store
SIMD logic/shuffle

execution units

internal registers

16 FP register

out of order execution superscalar

issue 8 *(6)* μops/ cycle

instruction decoder

RISC μops

CISC ops

instruction pool (up to 192 *(168)* "in flight")

1 Core

lat: 4 *(4)*
tp: 12 = 8 ld + 4 st *(4)*

L1 Dcache

L1 Icache

both: **32 KB 8-way 64B CB**

lat: 11 *(12)*
tp: 8 *(4)*

L2 cache 256 KB 8-way 64B CB

lat: ~34 *(26-31)*
tp: 4 *(4)*

Shared L3 cache 8 MB 16-way 64B CB

lat: ~125 *(100)*
tp: 2 *(1)*

Main Memory (RAM) 32 GB max

lat: millions
tp: ~1/50 *(~1/100)*

depends on hard disk technology, I/O interconnect

Hard disk ≥ 0.5 TB

CB = cache block

Core i7-4770 Haswell:
4 cores, 8 threads
3.4 GHz
(3.9 GHz max turbo freq)
2 DDR3 channels 1600 MHz

ring interconnect

Core #1, L1, L2 — **L3**
Core #2, L1, L2 — **L3**
Core #3, L1, L2 — **L3**
Core #4, L1, L2 — **L3**

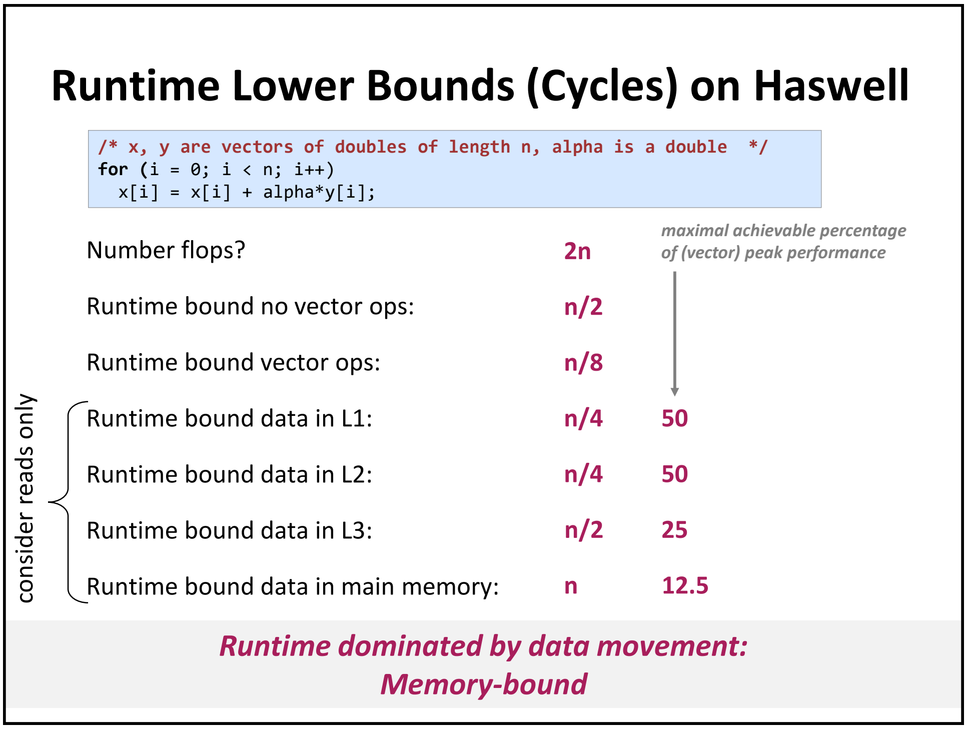
core *uncore*

processor die

RAM

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

© Markus Püschel Computer Science



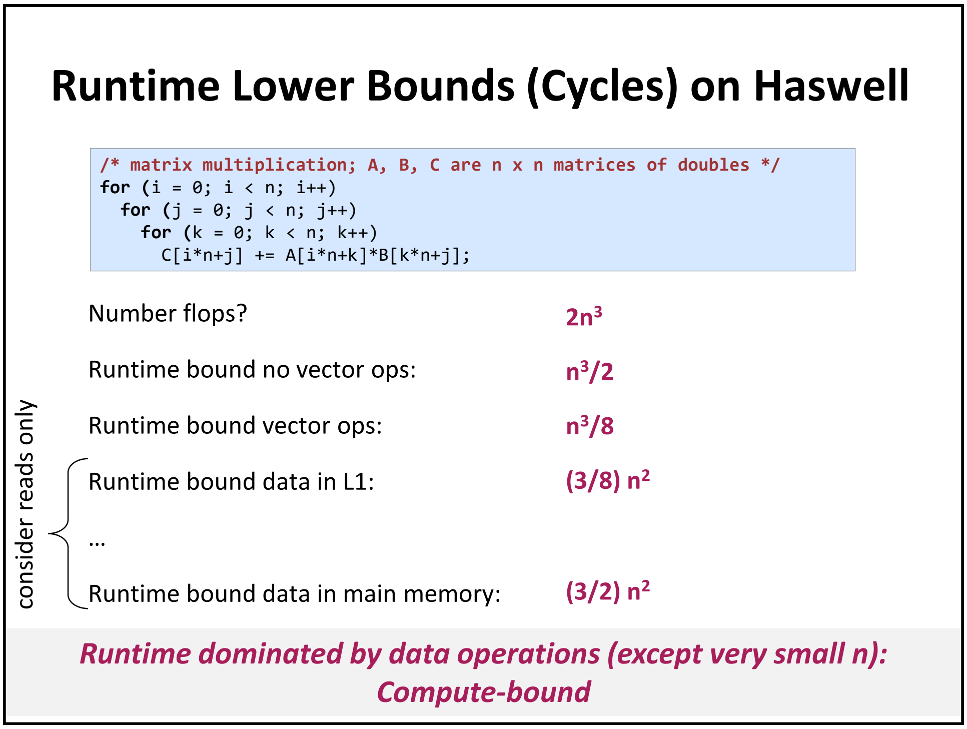
Runtime Lower Bounds (Cycles) on Haswell

```
/* 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 Haswell

```
/* 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

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

$W(n)$: #flops (input size n)

$Q(n)$: #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 $Q_{read}(n) \leq Q(n)$ and thus $I_{read}(n) \geq I(n)$

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: $I(n) \leq I_{read}(n)$* **$= W(n)/Q_{read}(n) \leq 1/8$**

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.

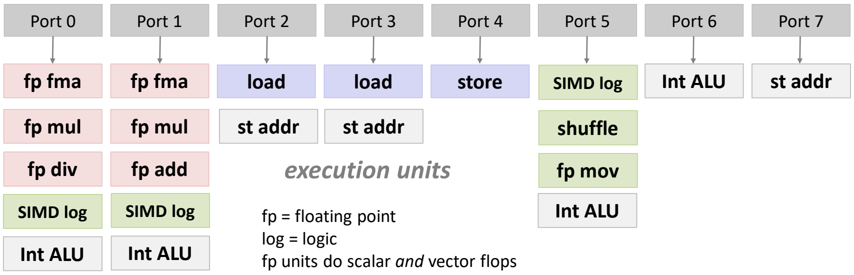
Benefit: Superscalar processors can take advantage of *instruction level parallelism (ILP)* that many programs have

Most CPUs since about 1998 are superscalar

Intel: since Pentium Pro

Simple embedded processors are usually not superscalar

Mapping of execution units to ports



Port 0: fp fma, fp mul, fp div, SIMD log, Int ALU

Port 1: fp fma, fp mul, fp add, SIMD log, Int ALU

Port 2: load, st addr

Port 3: load, st addr

Port 4: store

Port 5: SIMD log, shuffle, fp mov, Int ALU

Port 6: Int ALU

Port 7: st addr

execution units

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	5	2	0.5
mul	5	2	0.5
add	3	1	1
div (scalar) div (4-way)	14-20 25-35	1/13 1/27	13 27

- Every port can issue one instruction/cycle
- Gap = 1/throughput
- ***Intel calls gap 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 2-6. Dispatch Port and Execution Stacks of the Haswell Microarchitecture, Figure 2-2. CPU Core Pipeline Functionality of the Haswell 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)

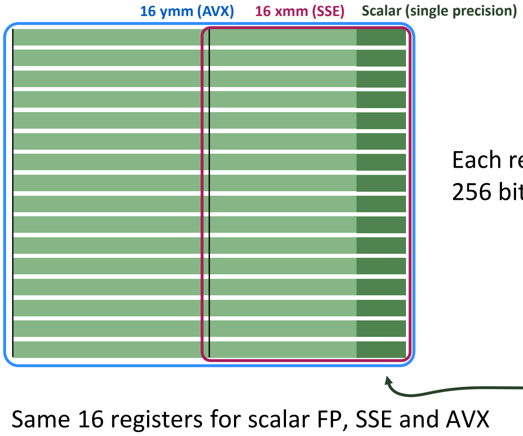
Execution units behind the same port cannot start an operation in the same cycle

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



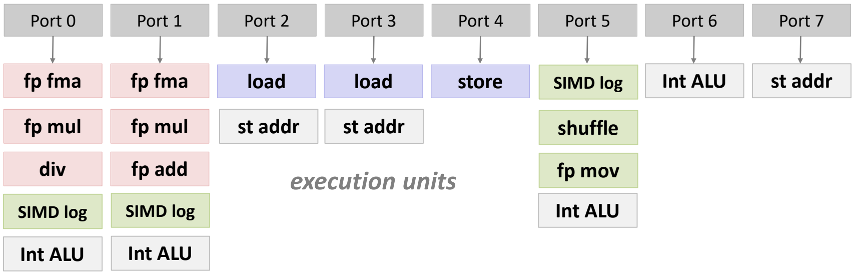
Each register:
256 bits = 4 doubles = 8 singles

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

© Markus Püschel
Computer Science
ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



Port 0	Port 1	Port 2	Port 3	Port 4	Port 5	Port 6	Port 7
fp fma	fp fma	load	load	store	SIMD log	Int ALU	st addr
fp mul	fp mul	st addr	st addr		shuffle		
div	fp add				fp mov		
SIMD log	SIMD log				Int ALU		
Int ALU	Int ALU						

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*
A function with n adds and n/2 mults in the C code	*n/2*

Comments on Intel Haswell 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: 1 add/cycle (scalar 1 flop/cycle, SIMD AVX 4 flops/cycle) FMA in port 0 can be used for add, but longer latency*

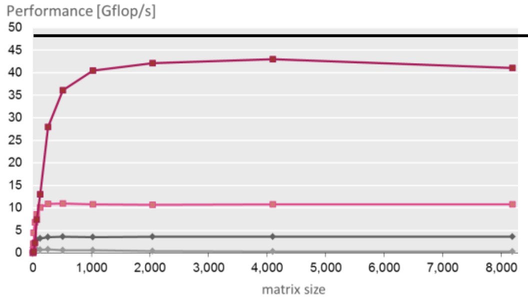
L1 bandwidth: two 32-byte loads *and* one 32-byte store per cycle

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

Matrix-Matrix Multiplication (MMM) on 2 x Core 2 Duo 3 GHz (Sandy Bridge)



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

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*