Title: Scientific Computing and Computational Science

Date: Jun 19, 2017 05:15 PM

URL: http://pirsa.org/17060067

Abstract:

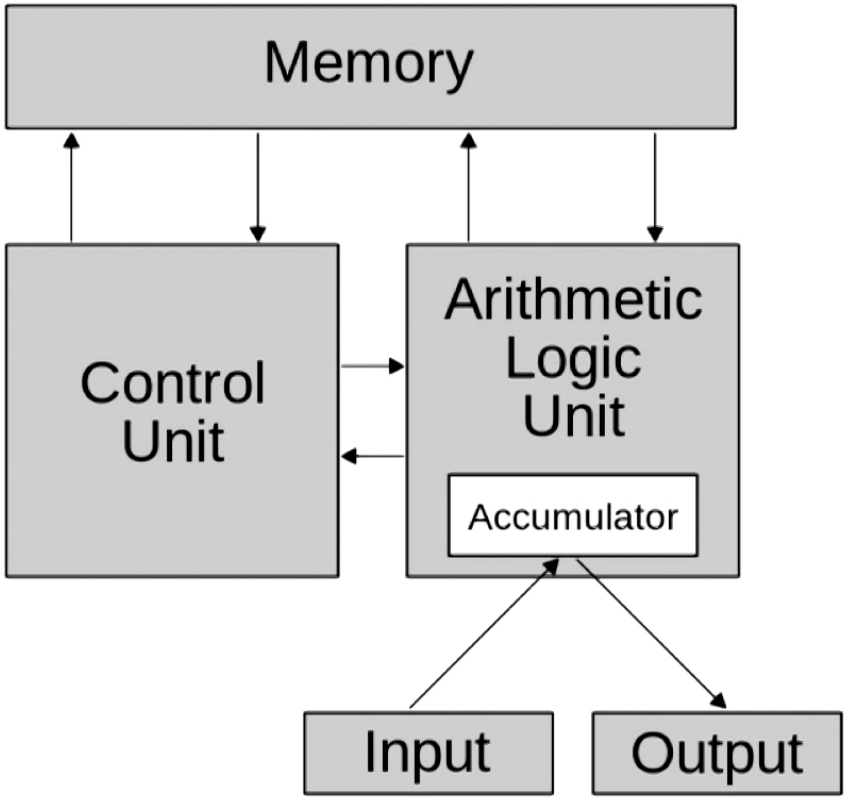
Scientific Computing and Computational Science

Erik Schnetter, Perimeter Institute

Making Quantum Gravity Computable, 2017-05-19

- What is a Computer?
- How do I program it?

Von Neumann Architecture



Sequential Execution:

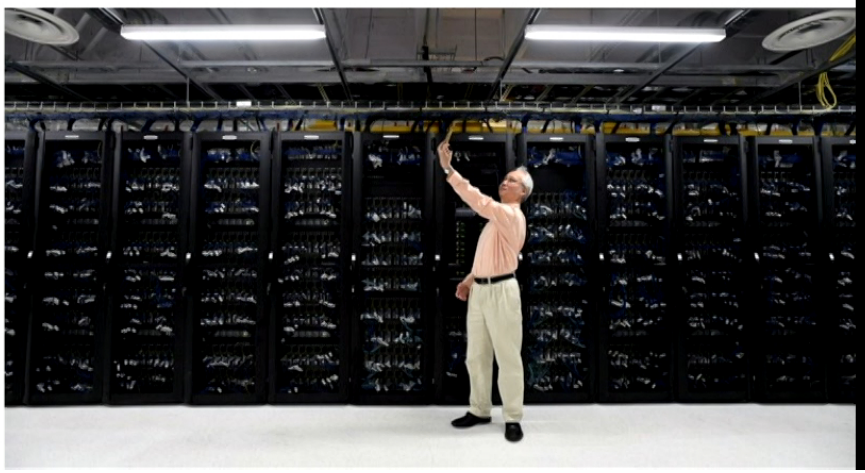
x := a + b

1. read instruction
2. read a
3. read b
4. add
5. write x

[Wikipedia]

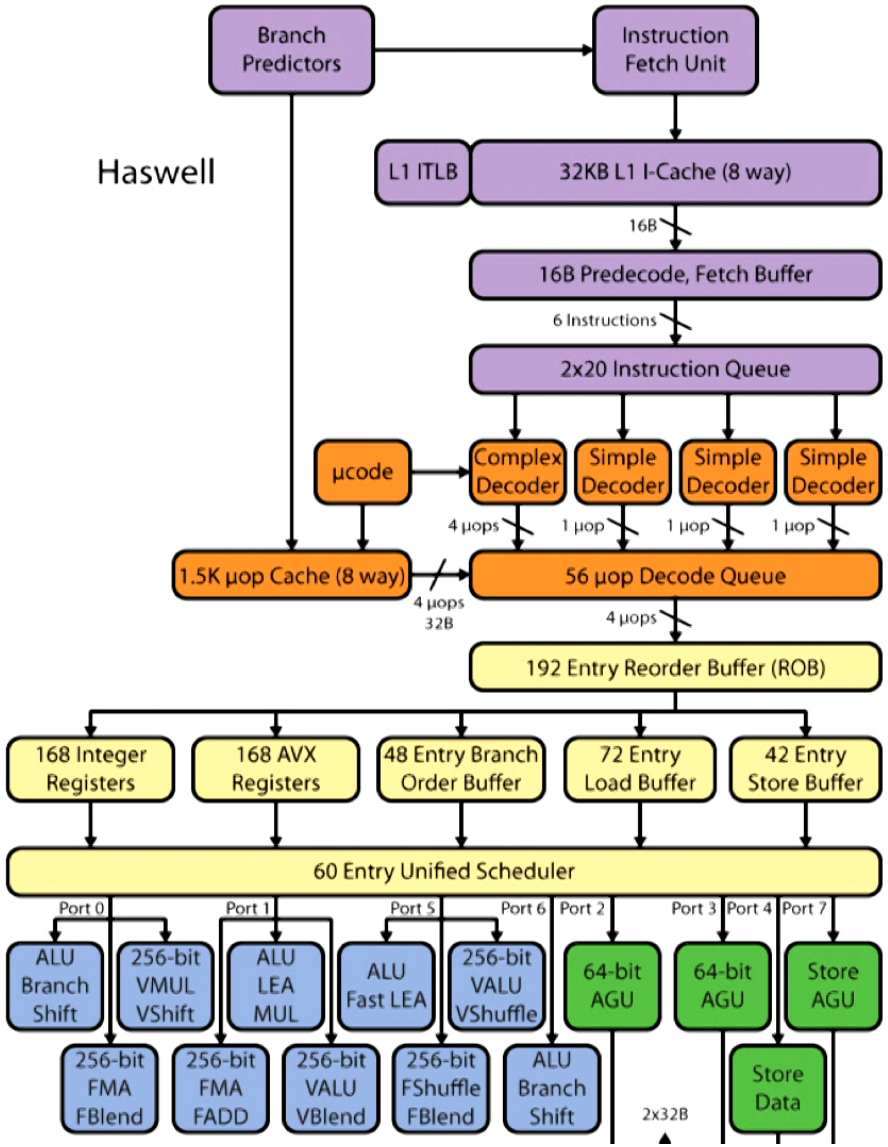
C64 (1984)

Graham (UW, 2017) [The Record]



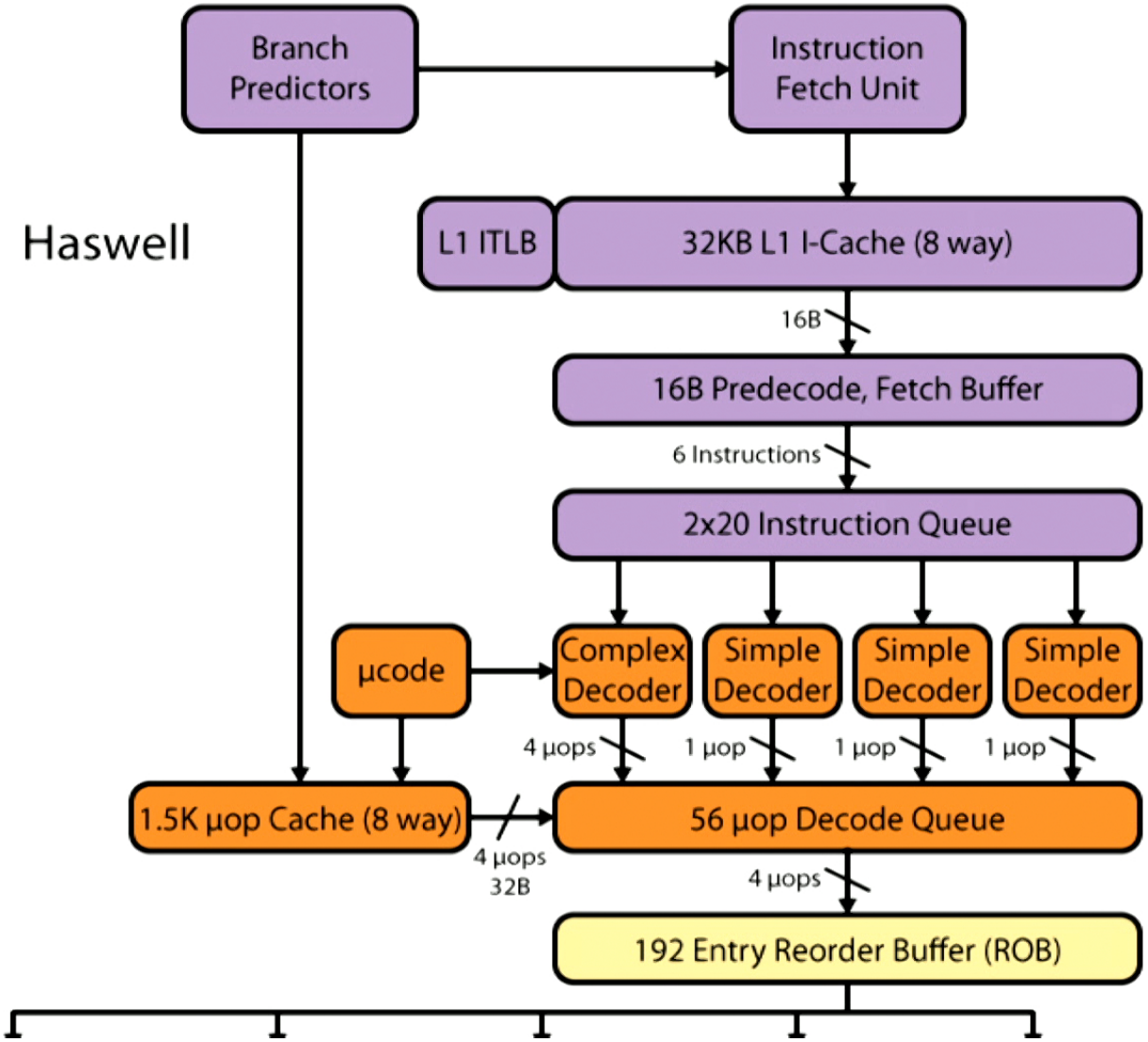
Algorithm Development

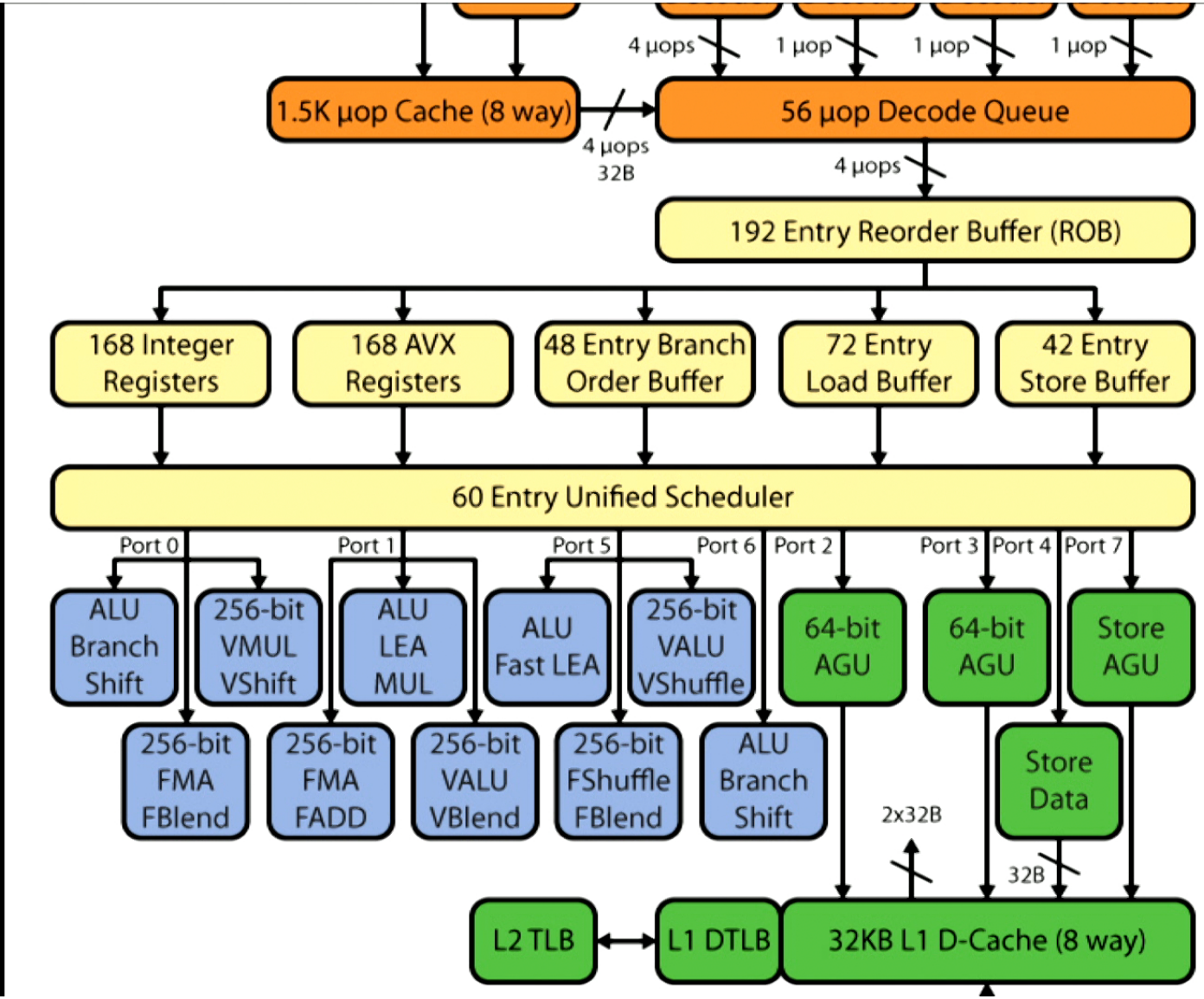
- Naïve idea:
  - Von Neumann (and others) developed a theoretical model of computing
  - Take an algorithm (i.e. constructive proof), map it to this model, implement it in a computer language
- In practice:
  - What is possible (and what is efficient) is not determined by an abstract model, but by current-day hardware technology
  - Good algorithms today look very different than e.g. 20 years ago



Modern Intel processor

[IDF 2012]


Haswell
Branch Predictors
Instruction Fetch Unit
L1 ITLB
32KB L1 I-Cache (8 way)
16B
16B Predecode, Fetch Buffer
6 Instructions
2x20 Instruction Queue
µcode
Complex Decoder
Simple Decoder
Simple Decoder
Simple Decoder
4 µops
1 µop
1 µop
1 µop
1.5K µop Cache (8 way)
4 µops
32B
56 µop Decode Queue
4 µops
192 Entry Reorder Buffer (ROB)

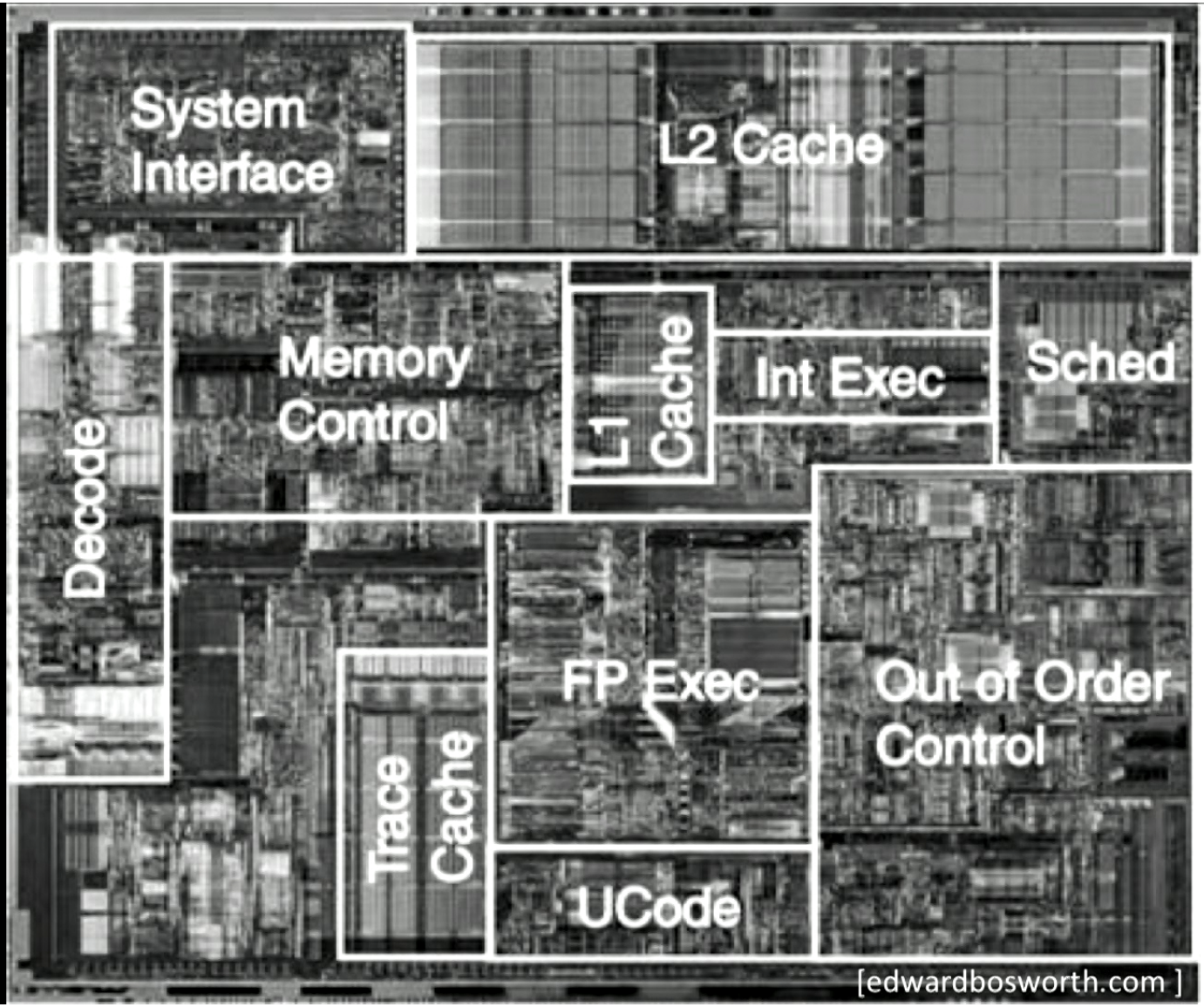

4 µops
1 µop
1 µop
1 µop
1.5K µop Cache (8 way)
4 µops
32B
56 µop Decode Queue
4 µops
192 Entry Reorder Buffer (ROB)
168 Integer Registers
168 AVX Registers
48 Entry Branch Order Buffer
72 Entry Load Buffer
42 Entry Store Buffer
60 Entry Unified Scheduler
Port 0
Port 1
Port 5
Port 6
Port 2
Port 3
Port 4
Port 7
ALU Branch Shift
256-bit VMUL VShift
ALU LEA MUL
ALU Fast LEA
256-bit VALU VShuffle
64-bit AGU
64-bit AGU
Store AGU
256-bit FMA FBlend
256-bit FMA FADD
256-bit VALU VBlend
256-bit FShuffle FBlend
ALU Branch Shift
Store Data
2x32B
32B
L2 TLB
L1 DTLB
32KB L1 D-Cache (8 way)

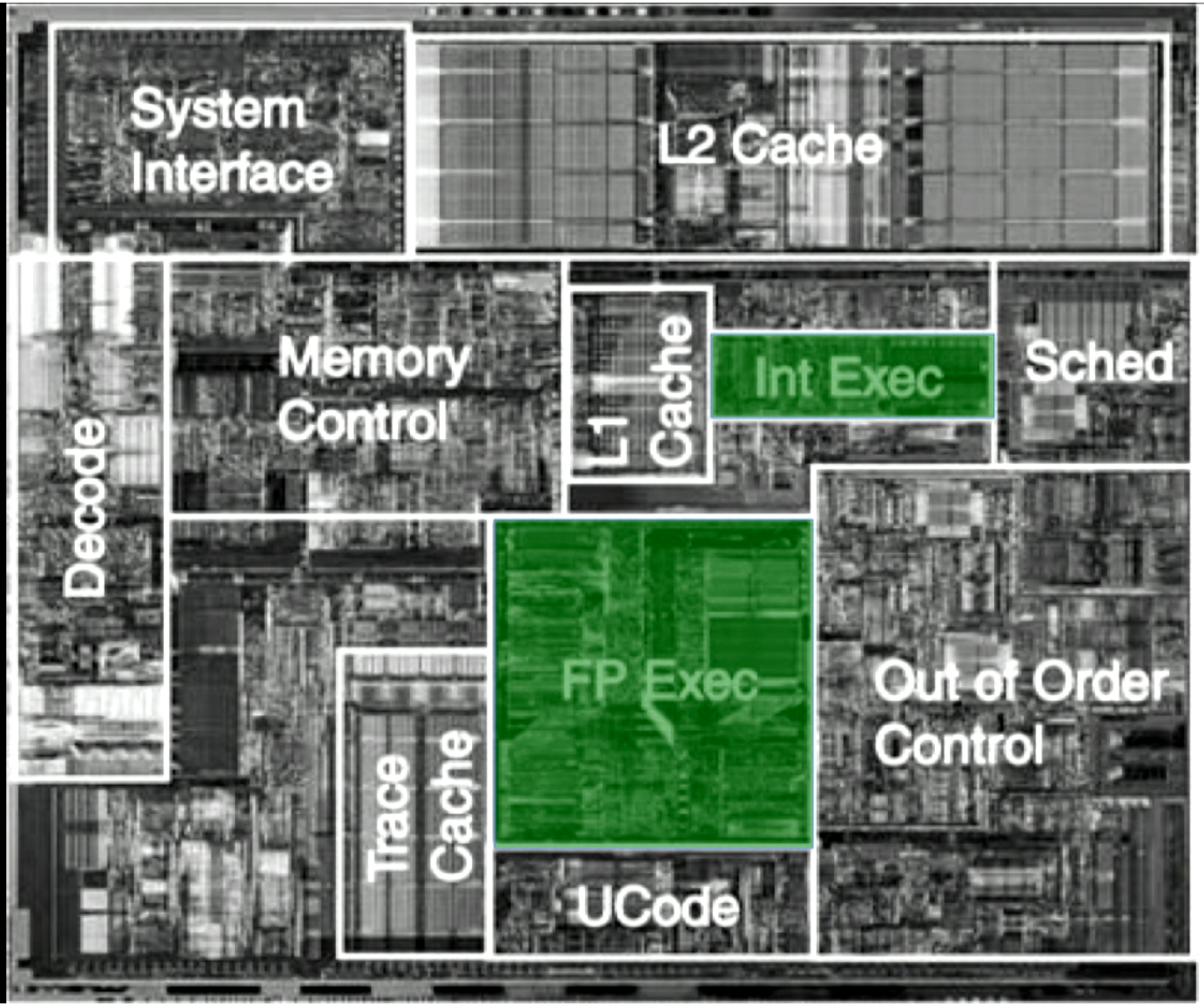
Sequential vs. Parallel

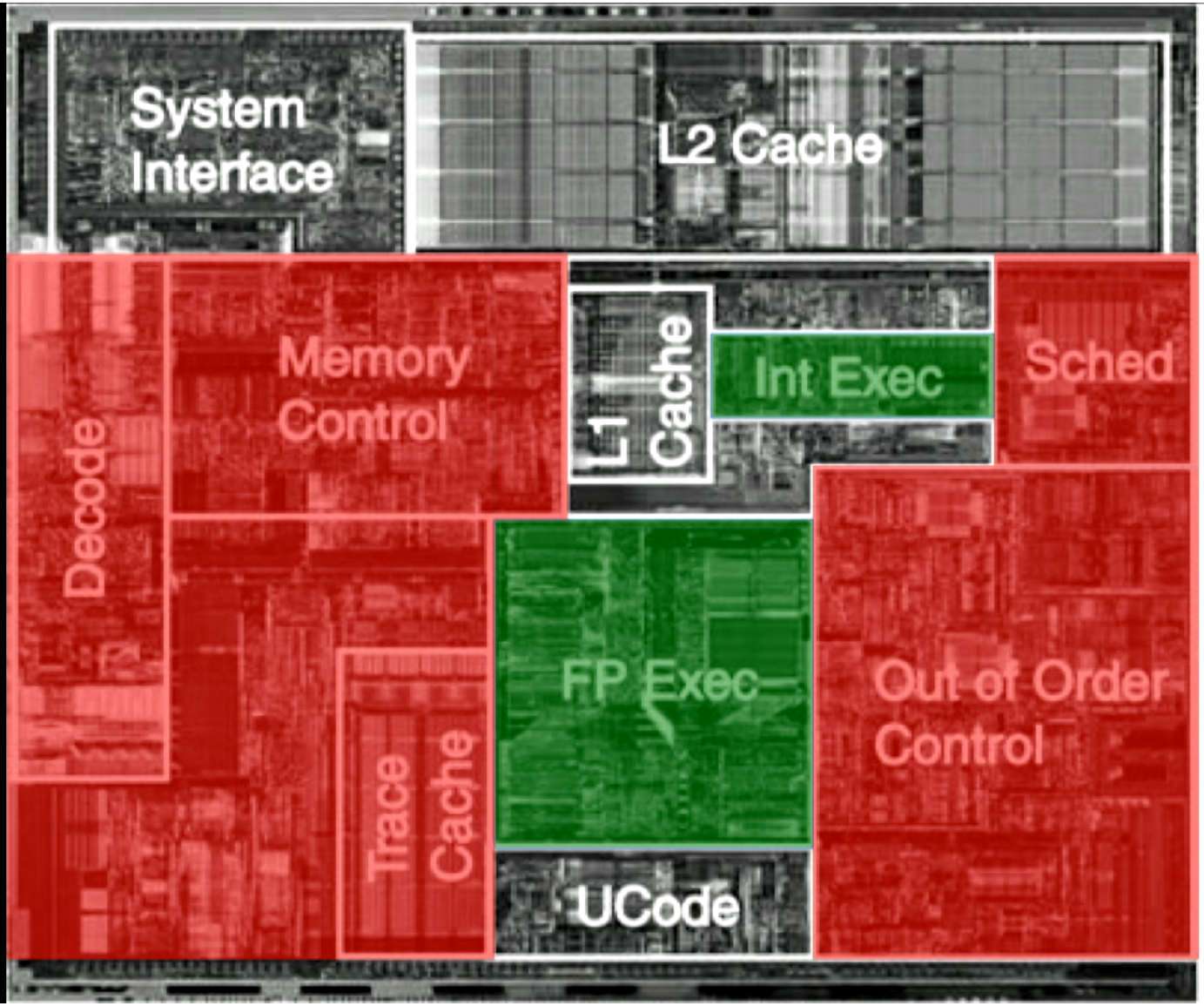
- Modern CPUs are highly parallel:
  - Time for one add operation: 10^{-9} sec
  - Peak performance: 10^{12} Flop/sec
- Efficient algorithms need to be parallel, not sequential
  - Operations need to be *local* and *independent*

Causality

- Newtonian: Events have a unique global order
- (speed of light) times (0.33 ns) = 10 cm
  - Smaller than a laptop
- Relativistic: Events can have spacelike relation
  - "Statements can be reordered by the system"
  - aka "memory model", "cache coherency"


System Interface
L2 Cache
Memory Control
L1 Cache
Int Exec
Sched
Decode
FP Exec
Out of Order Control
Trace Cache
UCode
[edwardbosworth.com]


System Interface
L2 Cache
Memory Control
L1 Cache
Int Exec
Sched
Decode
FP Exec
Out of Order Control
Trace Cache
UCode


System Interface
L2 Cache
Decode
Memory Control
L1 Cache
Int Exec
Sched
FP Exec
Out of Order Control
Trace Cache
UCode

Causality

- Newtonian causality:
  - A convenient fiction supported by hardware and standard programming languages
- Relativistic causality:
  - Reality; significantly faster, but amazingly difficult to use correctly
  - See also accelerators, GPUs

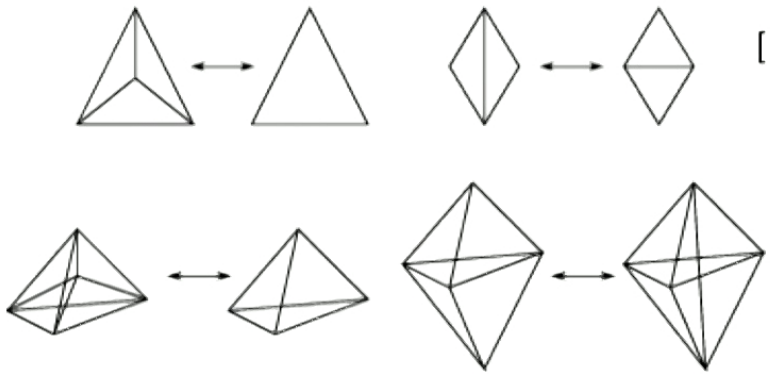
Energy Consumption

- Some well-known laws:
  - $Q = t \cdot I$ (parasitic capacity)
  - $U = R \cdot I$
  - $P = U \cdot I$
- Typical desktop CPU power and area:

  $50\ W / 200\ mm^2 = 2.5\ kW/m^2$
- Stovetop: $1.5\ kW / 750\ cm^2 = 0.2\ kW/m^2$

Supercomputer?

Are Pachner moves local? (Can they be parallelized?)



[Hellman, arXiv:1102.1688]

- Answer:
  - They are not local (only "almost local")
  - Need to form a monoid (or lattice?) for efficient parallelization

Object-Oriented Programming

- Current main-stream paradigm for large programs
- Objects:
  - Have a unique identity
  - Can model (classical) real-world items
  - Have a state that can change
- Problem:
  - Doesn't make sense for mathematical operations

Functional Programming

- "Functional" because there is an algebra of functions (e.g. composition)
- Usually:
  - Value semantics (no identities)
  - Referential transparency (immutability)
- Ideal for mathematical expressions, and for parallel programming
- (Surprisingly, can define object identities on top of this)

Functional Programming

- Dichotomy between mathematics and programming:
  1. Math is about eternal truths (there is no "time" in a proof)
  2. Programs execute sequentially
- How can one prove statements about programs?

- Functional programming:
  - Design programs to be order-independent
  - Remove distinction between data and functions

```
let x = 1
  sin(x)
end
```

```
let f = sin
   f(1)
end
```

- From juliacode.org:
  - high-level, high-performance dynamic programming language for technical computing
  - sophisticated compiler
  - distributed parallel execution
  - numerical accuracy
  - extensive mathematical function library
  - mature, best-of-breed open source C and Fortran libraries for linear algebra, random number generation, …
  - powerful browser-based graphical notebook interface

- Note: only few years old, still immature

Summary

- Causality in computers is relativistic
  - Newtonian causality is an expensive fiction
- Computers are highly parallel machines
  - Even laptops and cell phones
- Object-oriented programming doesn't help with mathematical modeling
  - Mathematical entities do not have an identity (functional programming!)