CS1: Computer Architecture

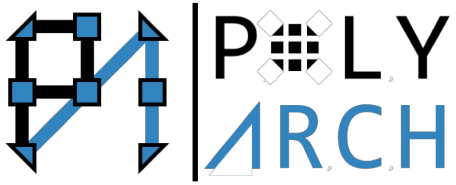
(is Von Neumann friend or foe?)

Tony Nowatzki

11/30/2018

About ME

- Graduated from Wisconsin 2016
- Joined UCLA January 2017
  - End of my second year
- Lead PolyArch Resesarch Group
  - 3 Wonderful Stuents
  - Design next-generation processors



- What do I teach?
- Fall: CS251a: Advanced Architectures
  (10 minute version today)
- Winter: CS33: Computer Organization
  (architecture + OS + low-level programming)
- Spring: CS259: Architectures for Machine Learning

In this talk

- What is architecture?
- Why is it broken?

  (some kinds of computers are hard to make faster)
- What can we do to fix it?

What is architecture?

- Hardware organization?
- Circuit design?
- Building chips?
- Something else?

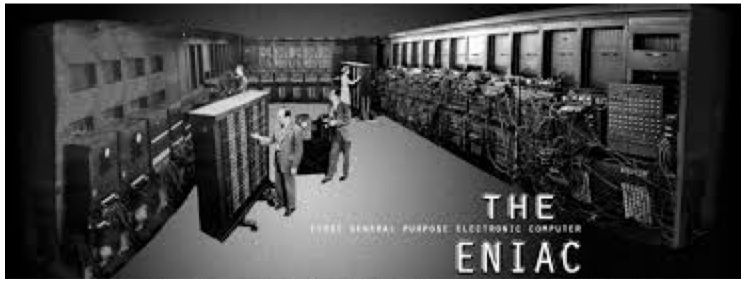
- Fun fact: You can have a computer without having an architecture!

Computers Pre-1964

- Each Computer was New
  - Implemented machine (has mass) → hardware
  - Instructions for hardware (no mass) → software

- Software Lagged Hardware
  - Each new machine design was different
  - Software needed to be rewritten in assembly/machine language

- Unimaginable today
  - Going forward: Need to separate HW interface from implementation



ENIAC: First architecture, kindof

Software World

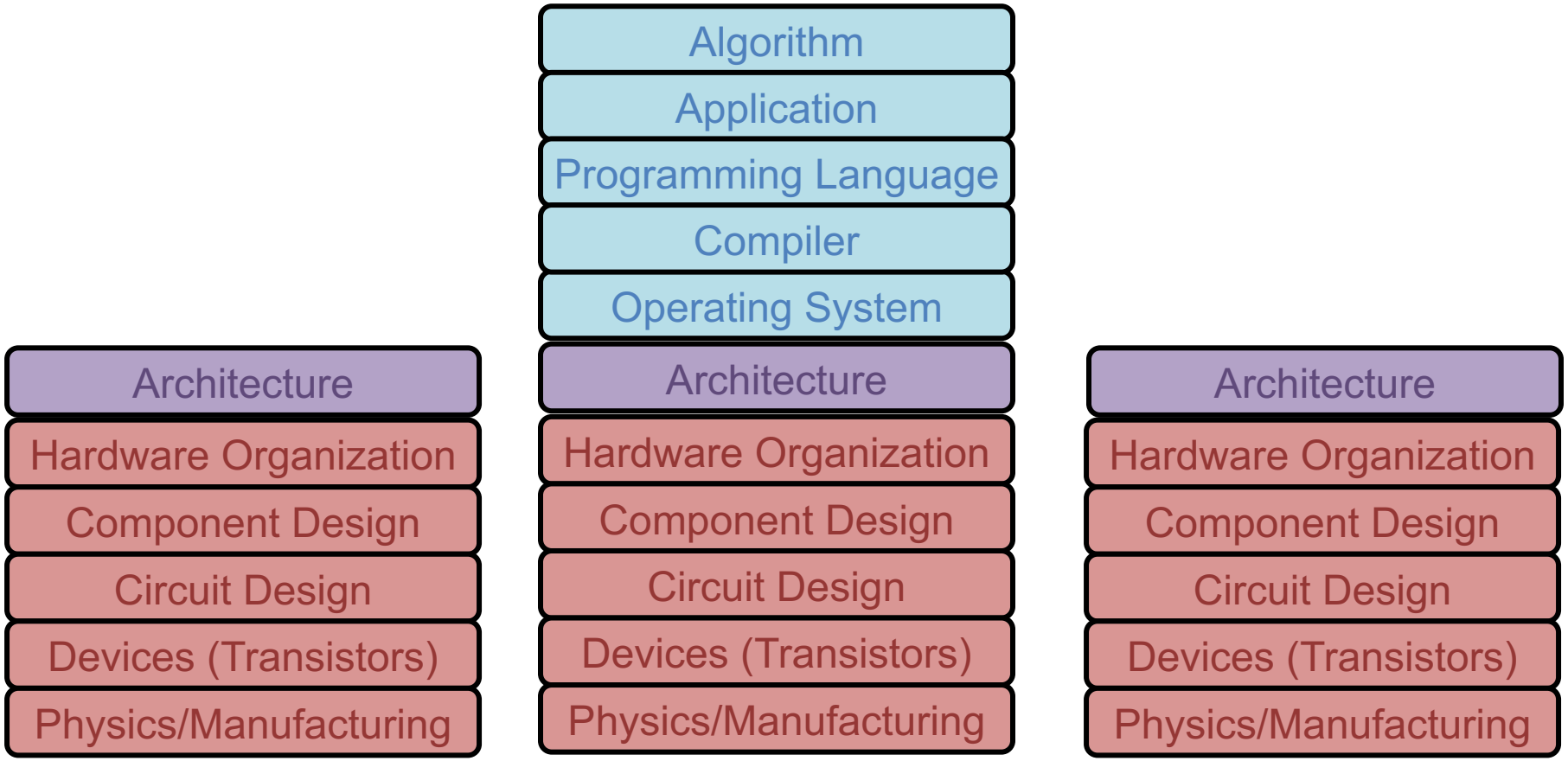
Algorithm	Algorithm	Algorithm
Application	Application	Application
Programming Language	Programming Language	Programming Language
Compiler	Compiler	Compiler
Operating System	Operating System	Operating System

Machine 1 | Machine 2 | Machine 3

Hardware Organization	Hardware Organization	Hardware Organization
Component Design	Component Design	Component Design
Circuit Design	Circuit Design	Circuit Design
Devices (Transistors)	Devices (Transistors)	Devices (Transistors)
Physics/Manufacturing	Physics/Manufacturing	Physics/Manufacturing

Hardware World

Architecture



Machine 1

Better for big-data workloads

Machine 2

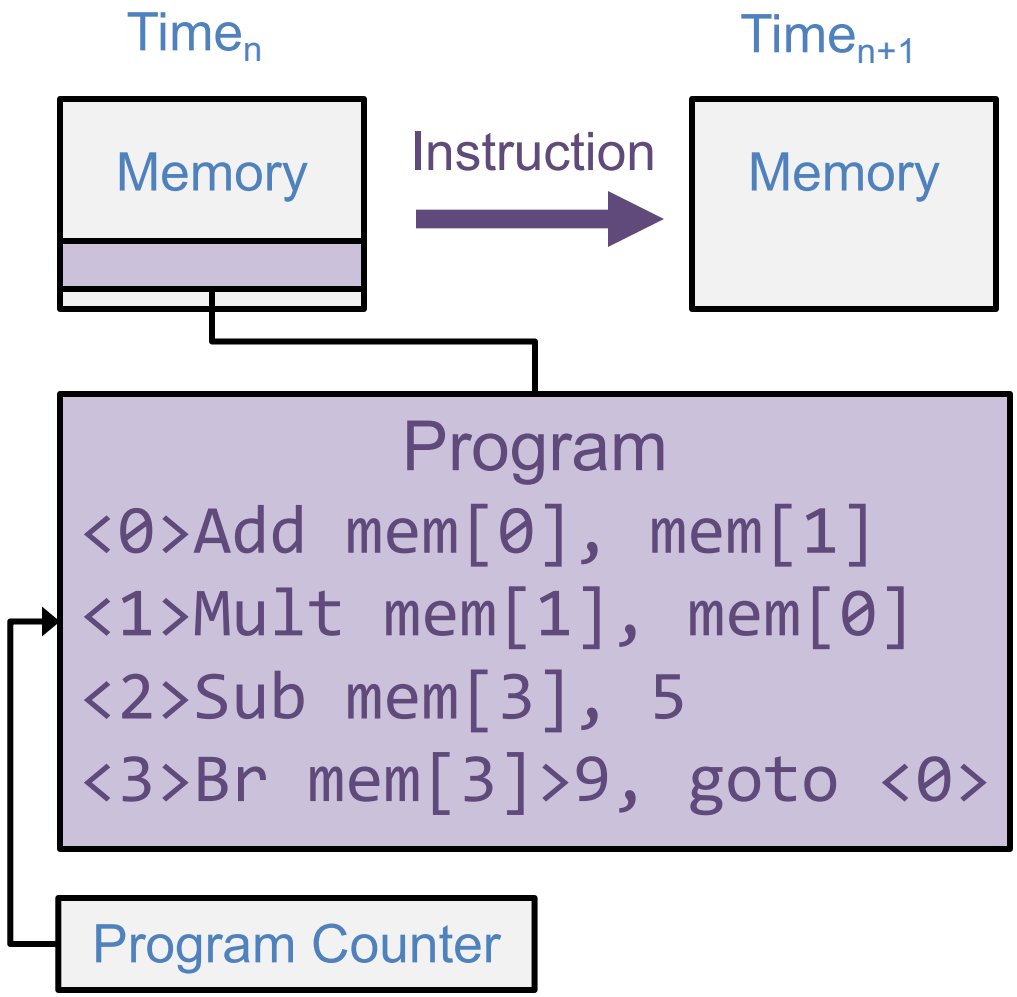
Better for desktop/gaming

Machine 3

Better mobile applications

What should be in an architecture?

- Ingredients:
  - **Memory:** a place to put values (state, variables, etc.)
  - **Instructions:** moves from one state to the next
  - **Program:** set of instructions (lets put it in memory)
  - **Execution model:** When do we execute each instruction?
- Von Neuman Execution:
  - Most common model today
  - Instructions are executed sequentially, **defined by a "program counter"**
  - Branch instruction (br)



Von Neuman Architectures: Instruction Set Architecture (ISA)

How does software use the ISA?

func.c

```
int func(unsigned n) {
  int ret=0;
  for(int i=0; i<n; ++i){
    if(i & 1) {
      ret+=i;
    }
  }
  return ret;
}
```

gcc compiler

a.out (binary)

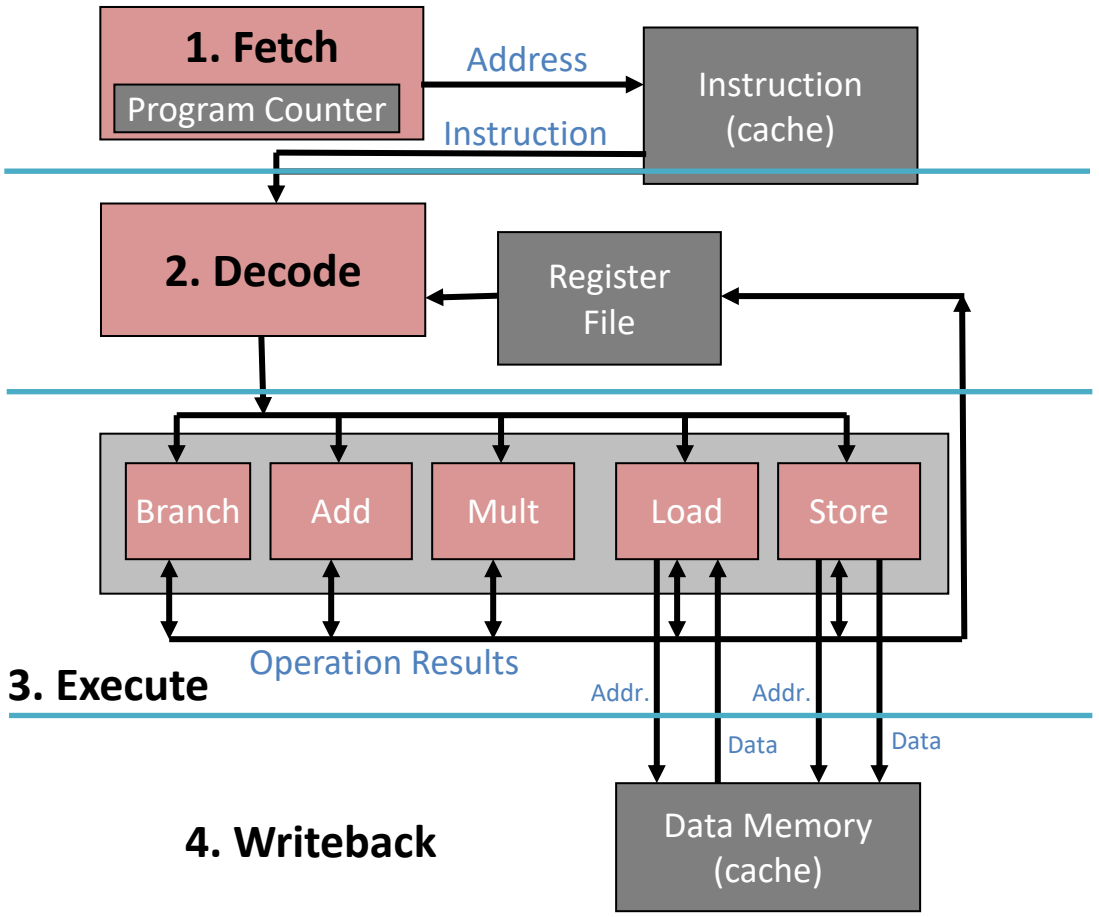
objdump

func() in x86 ISA

```
<0>:   mov    $0x0,%eax
<5>:   mov    $0x0,%edx
<10>:  br     <22>
<12>:  test   $0x1,%dl
<15>:  br     <19>
<17>:  add    %edx,%eax
<19>:  add    $0x1,%edx
<22>:  cmp    %edi,%edx
<24>:  br     <12>
<26>:  repz retq
```

How does hardware use the ISA?

- Steps to executing an instruction:
- **Fetch** – Grab instruction from memory
- **Decode** – Interpret instruction
- **Execute** – Perform Computation
- **Writeback** – Update State



Semi-realistic Diagram of CPU

Summary

1. ISA abstract hardware to make software stack simpler

2. Von Neumann ISA
   - Used by every CPU you own
   - Program: Set of instructions
   - Execution model: Sequential execution

x86

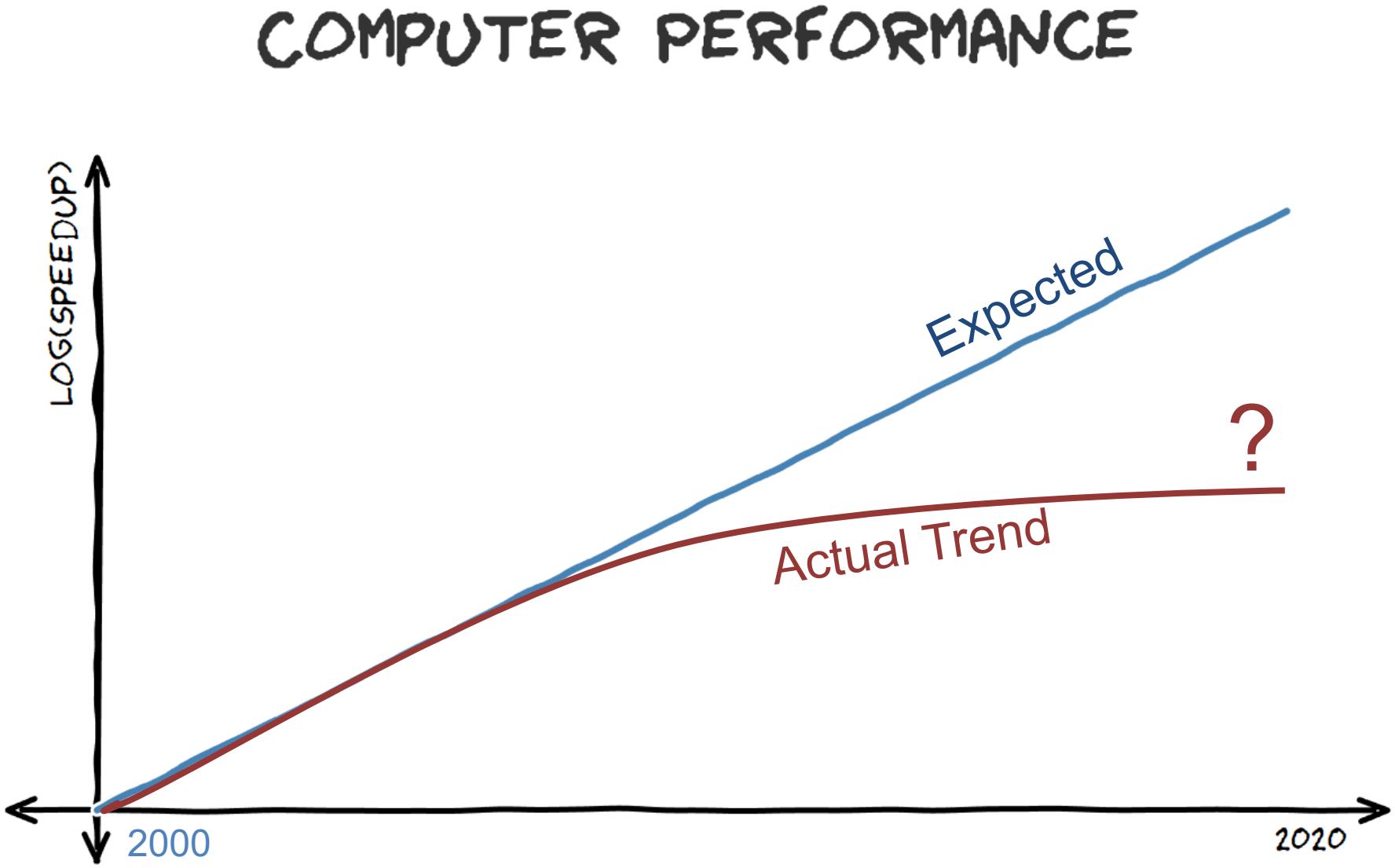
Destop, Server

ARM

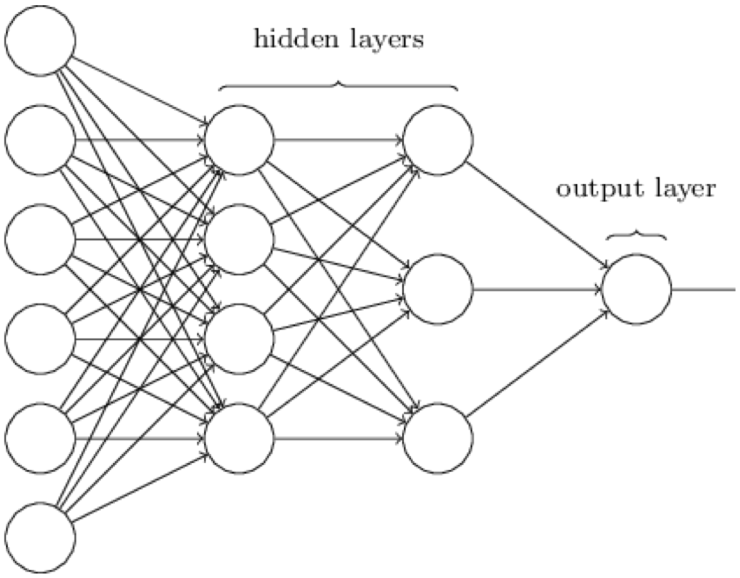
Mobile, Server

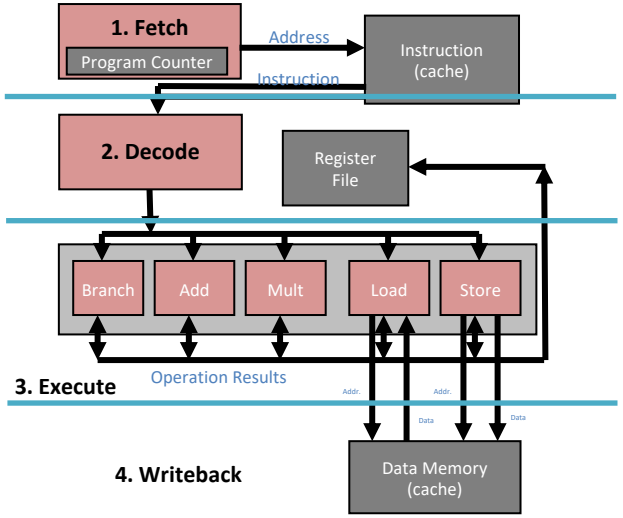
3. You now know how a computer works
   - Processing pipeline: Fetch/Decode/Execute/Writeback

Problem: CPUs getting harder to improve



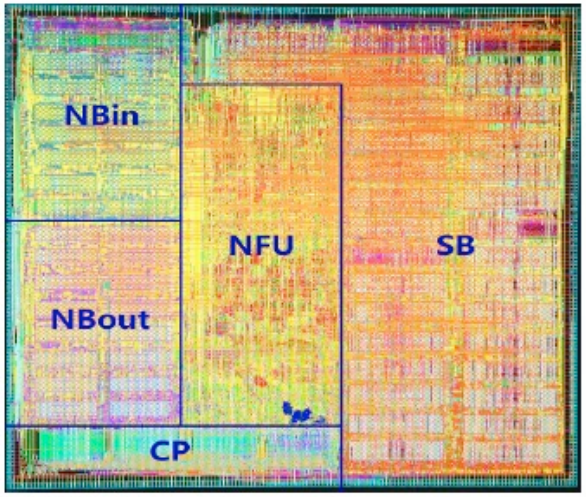
A tale of two computing paradigms…



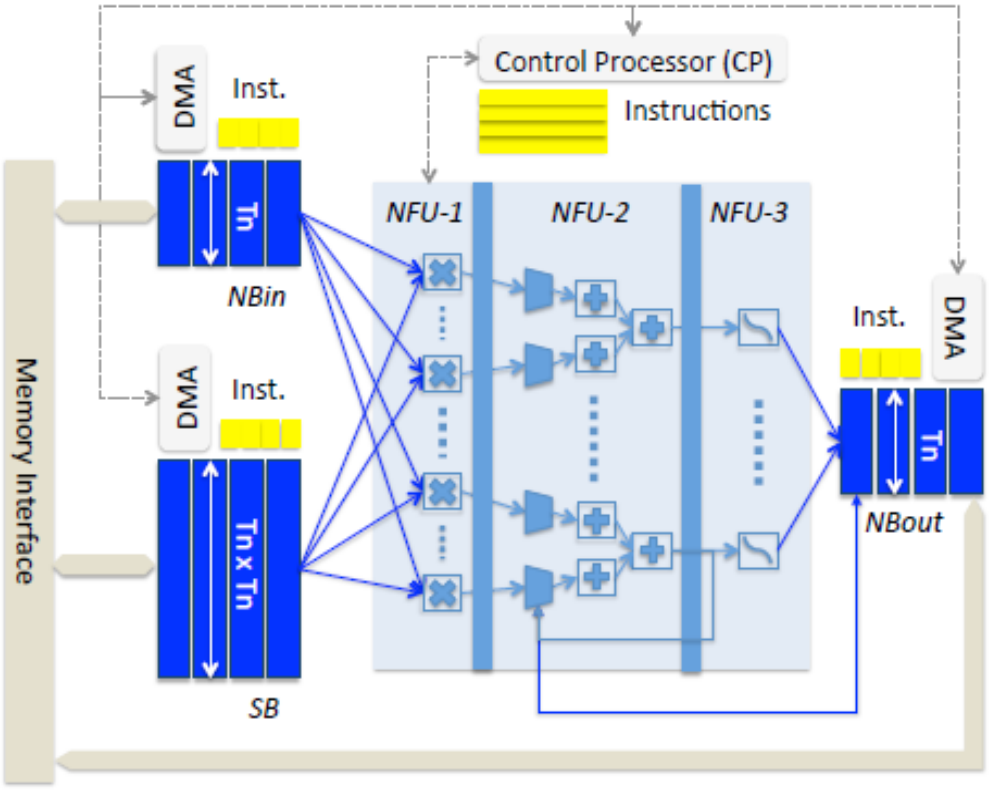


Deep Learning **General Purpose CPU**

Deep Learning Accelerator (2014)



DianNao



120x faster than traditional CPU,
20x Less Energy

Deep Learning Chips 2014 - 2018

- 2014:
  - DianNao – Simple SIMD Accelerator
  - **DaDianNao – Massive Neural Network on a Single Chip**
- 2015:
  - ShiDianNao – Extension to Computer Vision
  - **FPGA-Based-CNN – Reconfigurable Neural Net (@UCLA)**
  - Origami – Low Power Neural Network Accel.
- 2016:
  - Proteus -- Exploiting Numerical Precision Variability
  - NeuroCube -- 3D Memory + Neural Accelerators
  - Stripes -- Bit-Serial Deep Neural Network Computing
  - PuDianNao – Supports Multiple Mach. Learning Alg.
  - **ISAAC -- Analog Arithmetic Memristor-Based Design**
  - **EIE – Reduced Network Size by Order of Magnitude!**
  - **…**

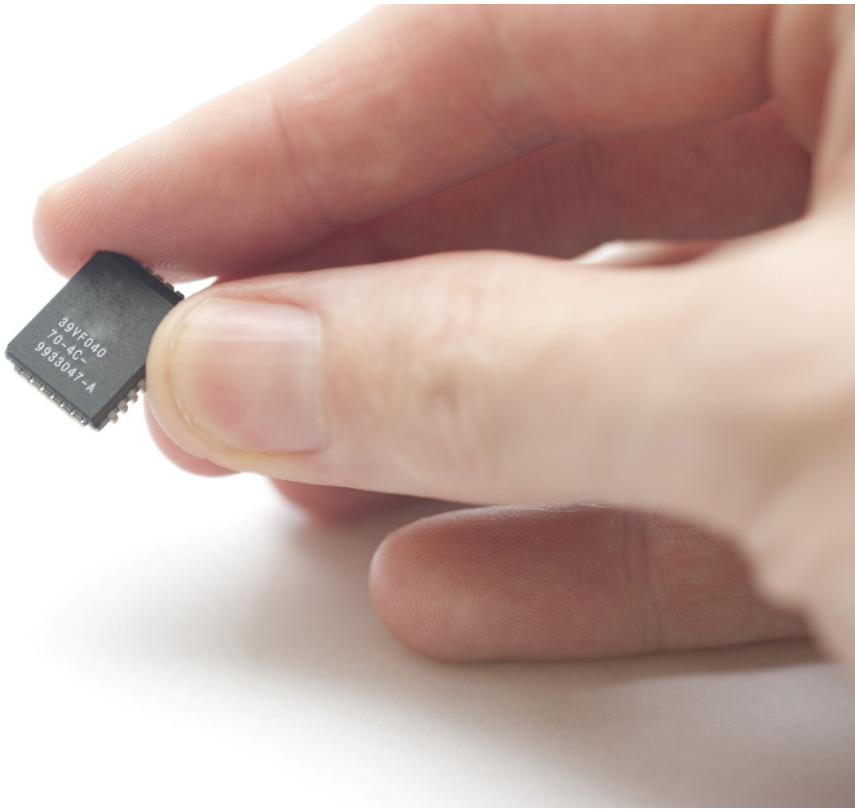
- 2017
  - **TPU – Tensor Processing Unit Released**
  - *SCALEDEEP: High-throughput*
  - ***SCNN: Compressed-sparse CNNs***
  - *Scalpel: Architecture aware NN pruning*
  - De sa et la.: Optimizing SGD Training
  - Park et al.: Scale-out techniques for NNs
  - Bit-pragmatic NN acceleration
  - **CirCNN: Frequency-domain arithmetic**
- 2018
  - Compressing DMA Engine: Leveraging Sparsity During Training
  - **In-situ AI: Incremental Deep Learning for IoT**
  - Reliability for Memristive Neural Network Accelerators
  - GAN-based Deep Learning Accelerators
  - …

Deep Learning 5 Year Ago

Deep Learning 2018

Industry Reality

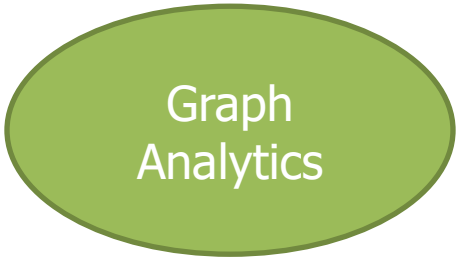
- Google -- TPU
- Intel – Nervana
- Baidu – SDA
- Microsoft – Brainwave
- NVIDIA GPU



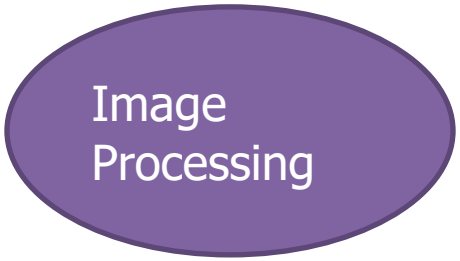
Startups too:

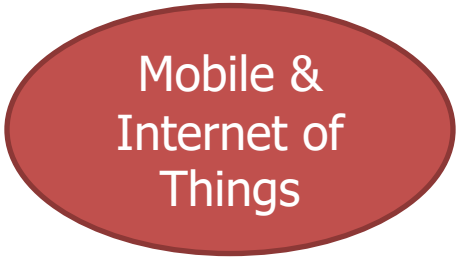
- Cerebras
- Deep Vision
- Graphcore
- Wave Computing
- Cambricon
- …

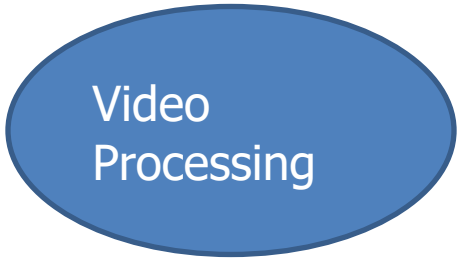
$



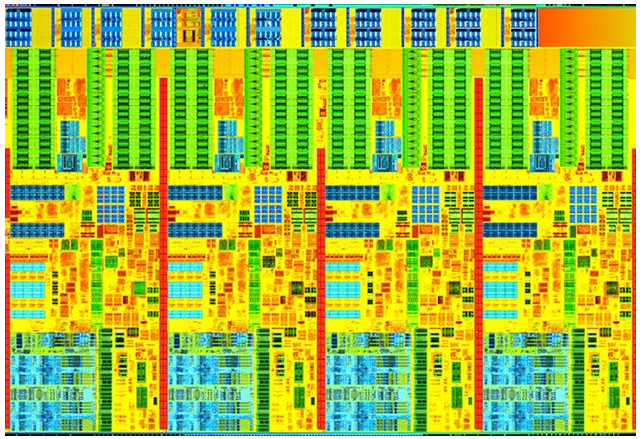
But what about the rest of computing???



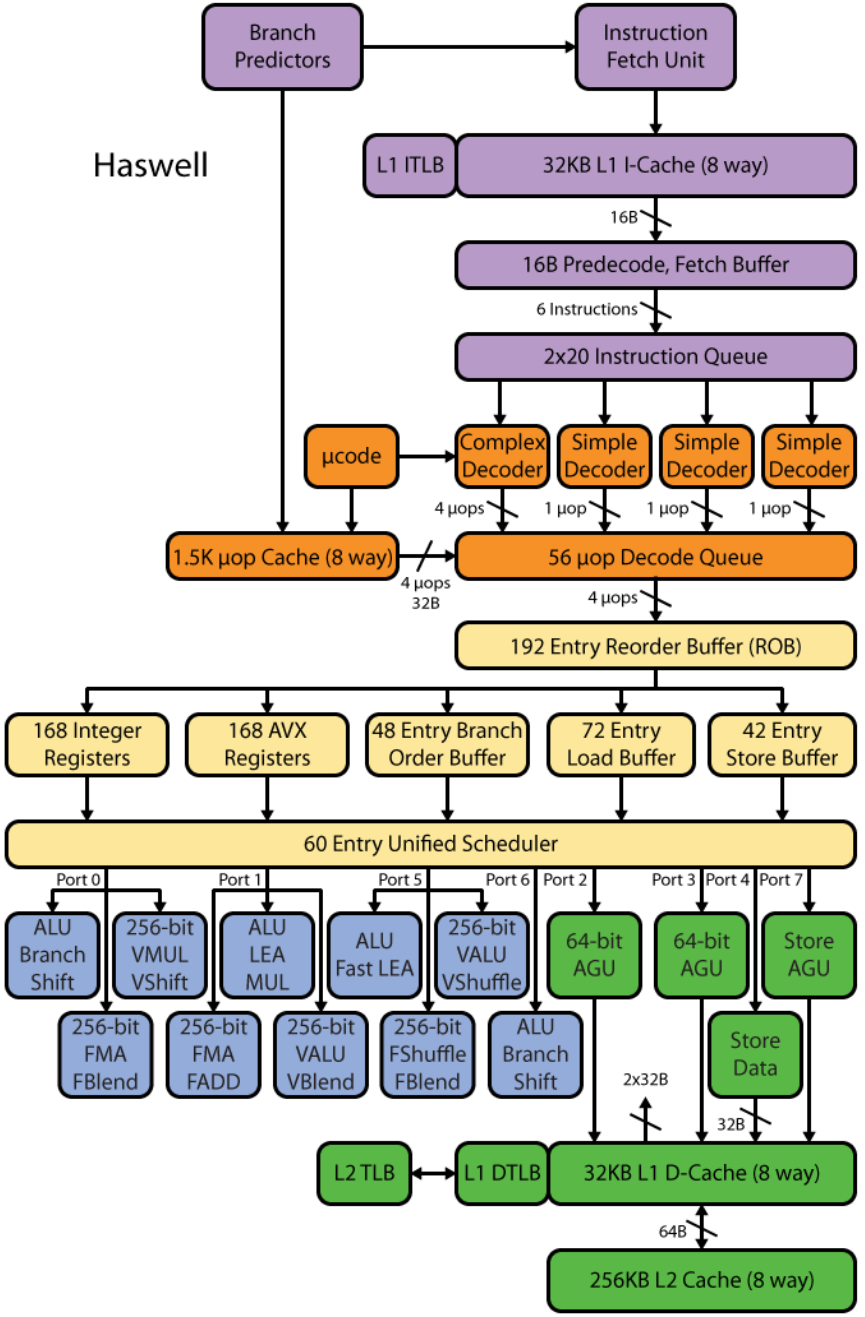




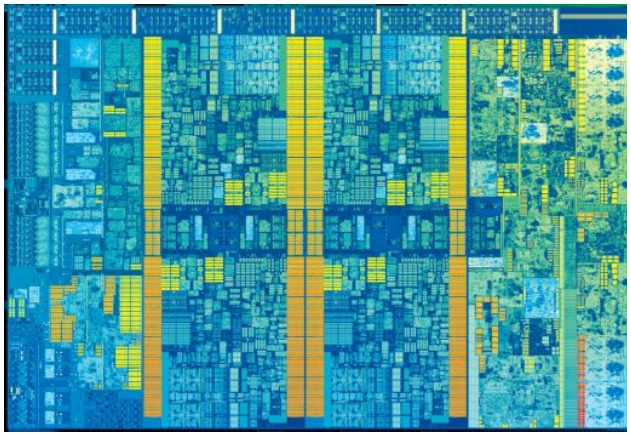
General Purpose 5 Years Ago 2013



Intel Haswell



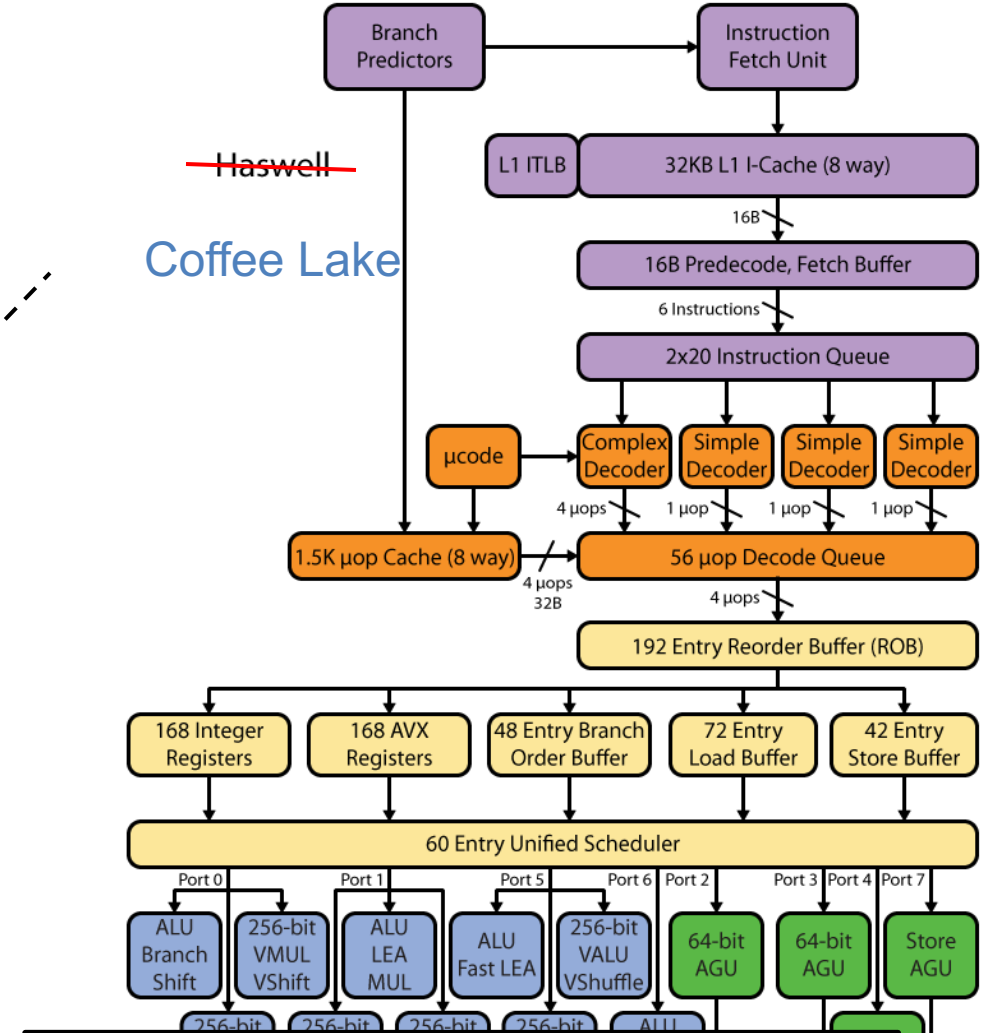
General Purpose (2018)



Intel Coffee Lake
~15% Speedup

~~Haswell~~

Coffee Lake



Now with more Spectre and Meltdown!

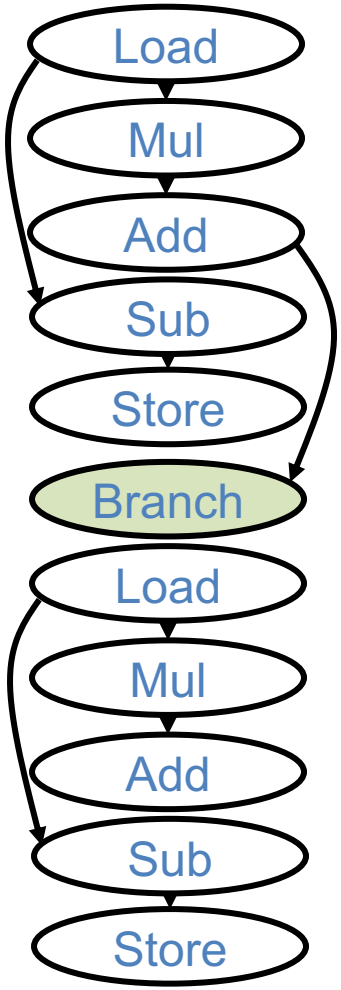
Paradox:

- General purpose processors **stagnating**
- Machine learning processors **thriving**

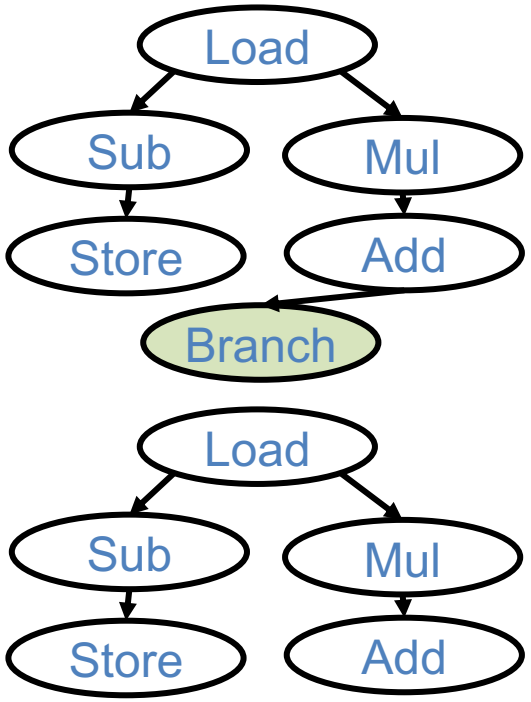
- Three key reasons:
  - General purpose processor design is inefficient
    - Bold Claim: because they are Von Neumann
  - No longer technology free ride
  - Lack Codesign Architecture

How do GPPs get high performance?

Von Neumann Order

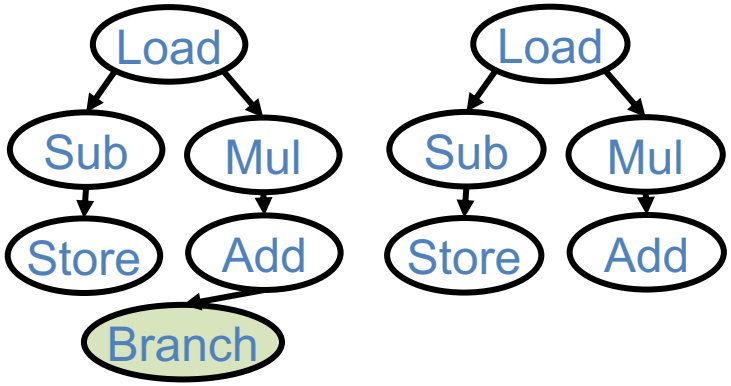


Out-of-order



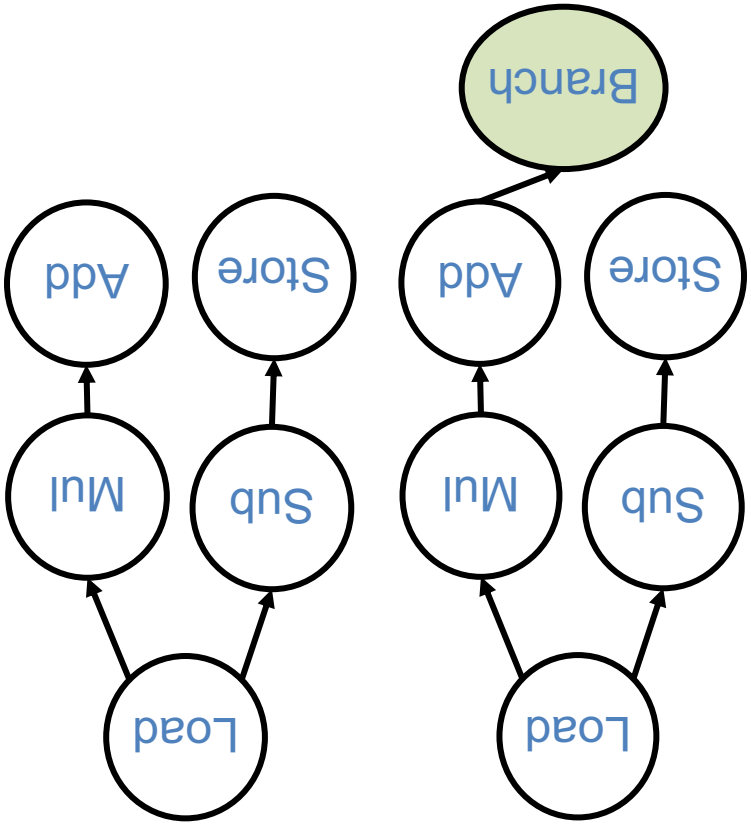
Reordered the instructions to increase parallelism!

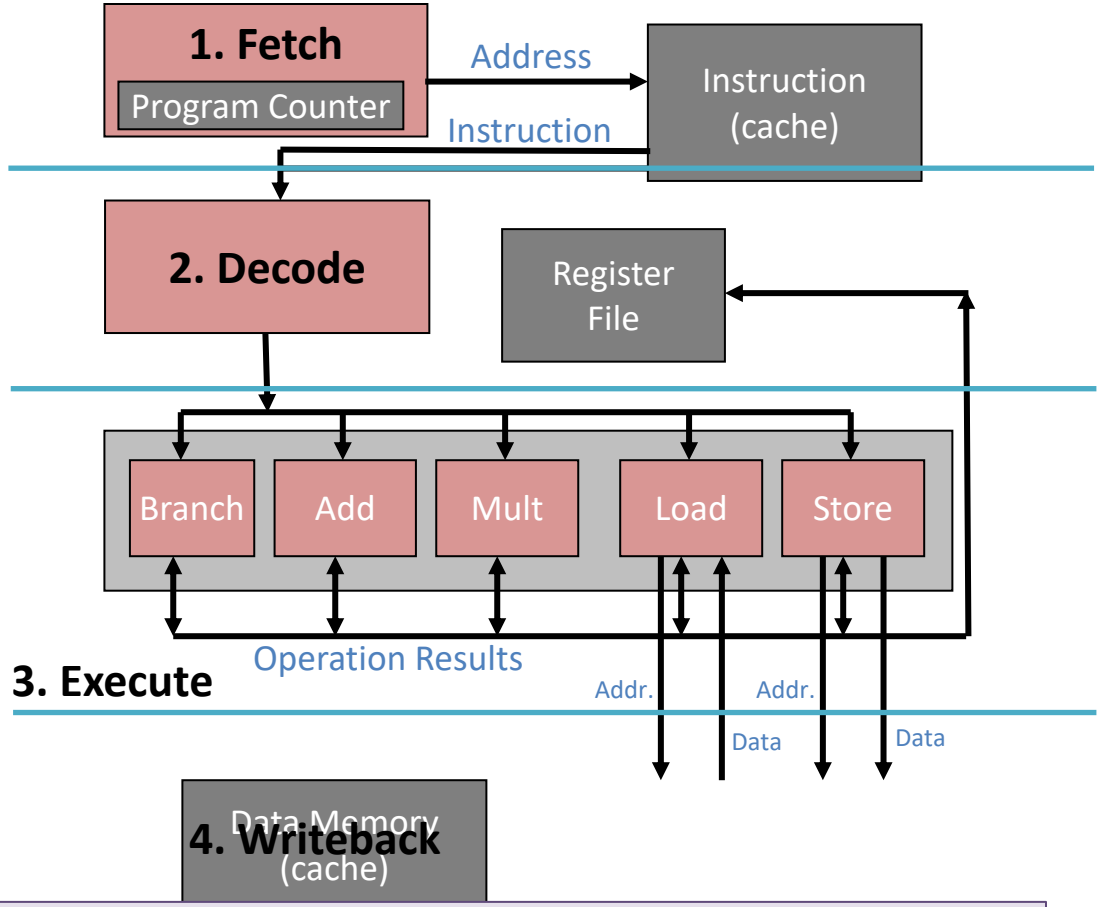
Out-of-order + Speculation



Pretend that we know control flow to get even more parallelism!

Out-of-order Speculative Processor

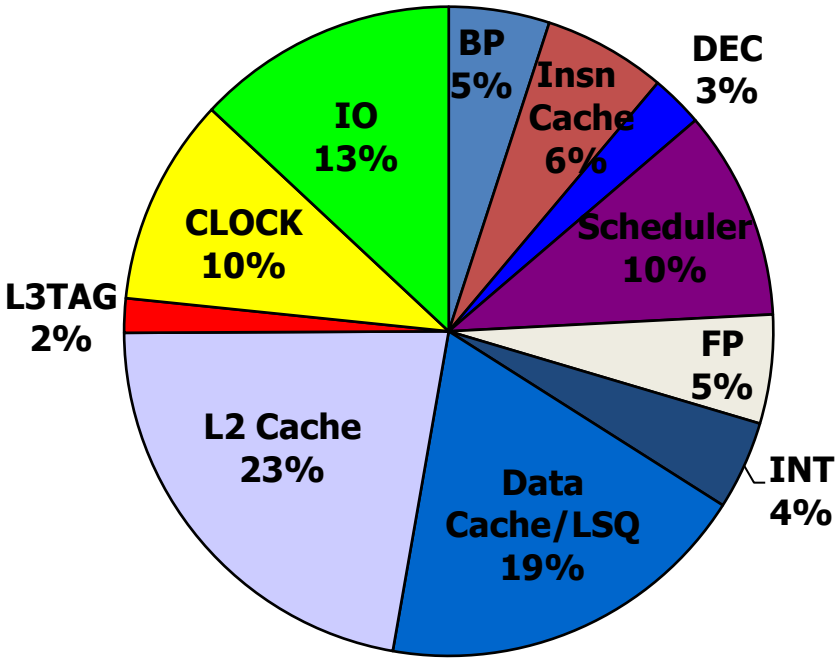


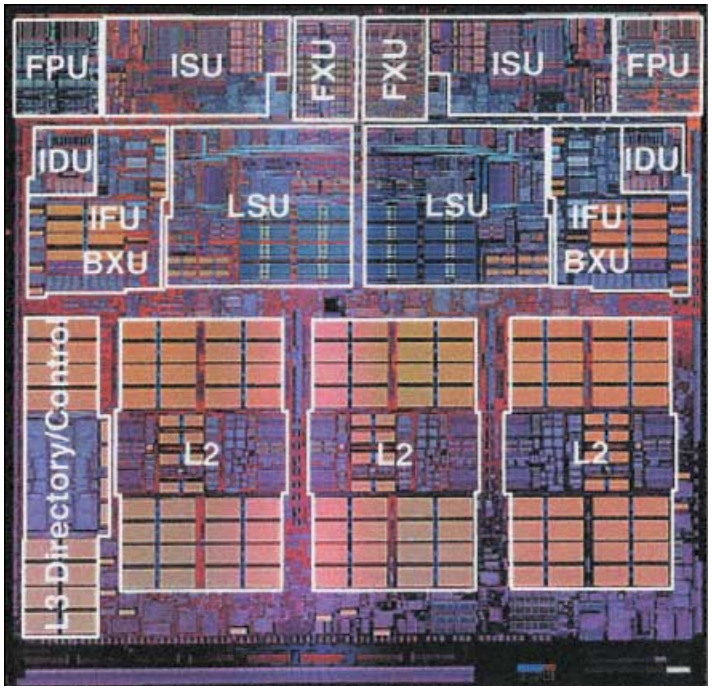


Unfortunate truth:
Maintaining the appearance of Von Neumann (sequential) while executing OOO is expensive in hardware.

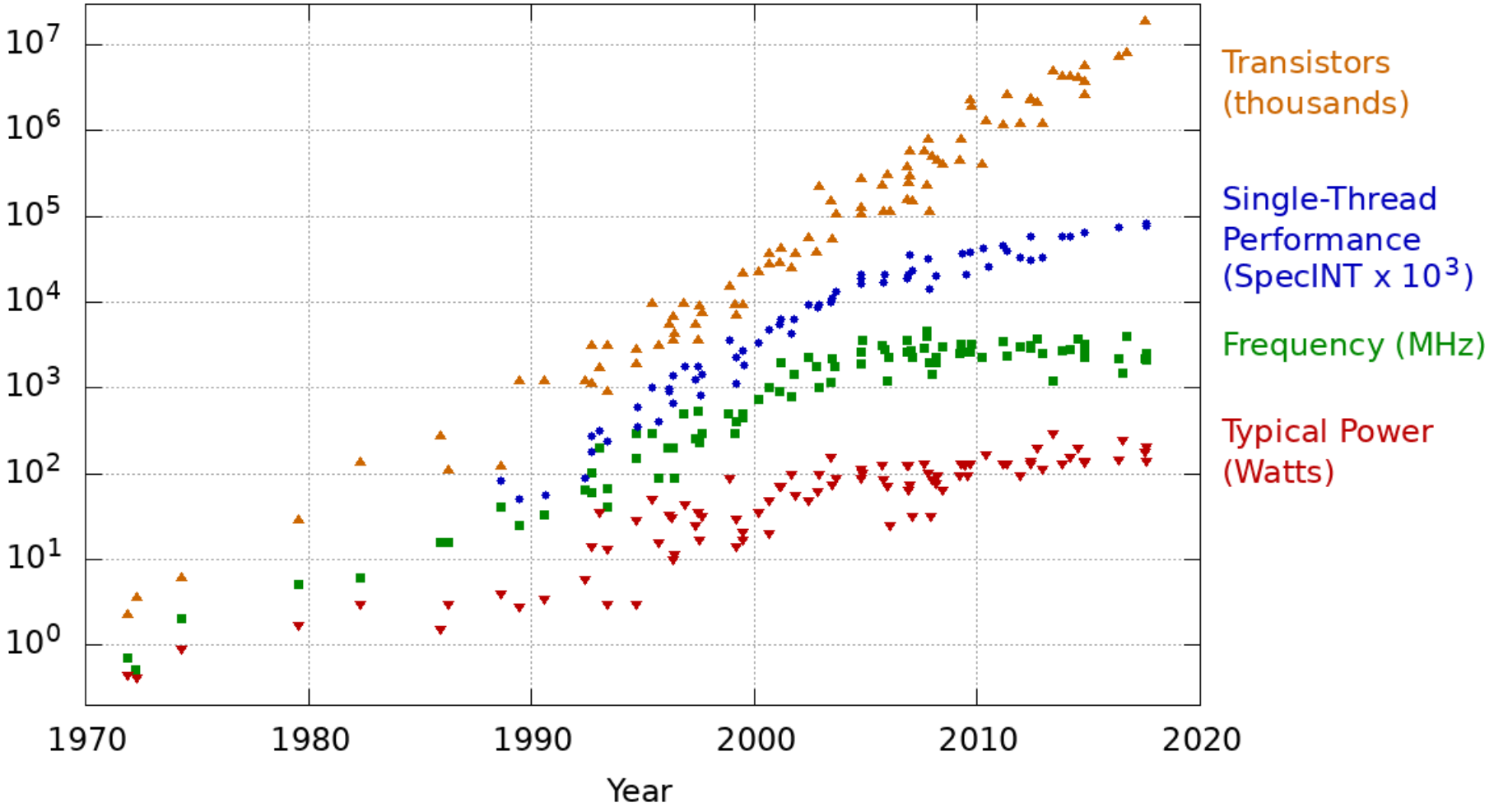
Processor Power Breakdown

- Power breakdown for IBM POWER4





5 Decades of Technology Trends

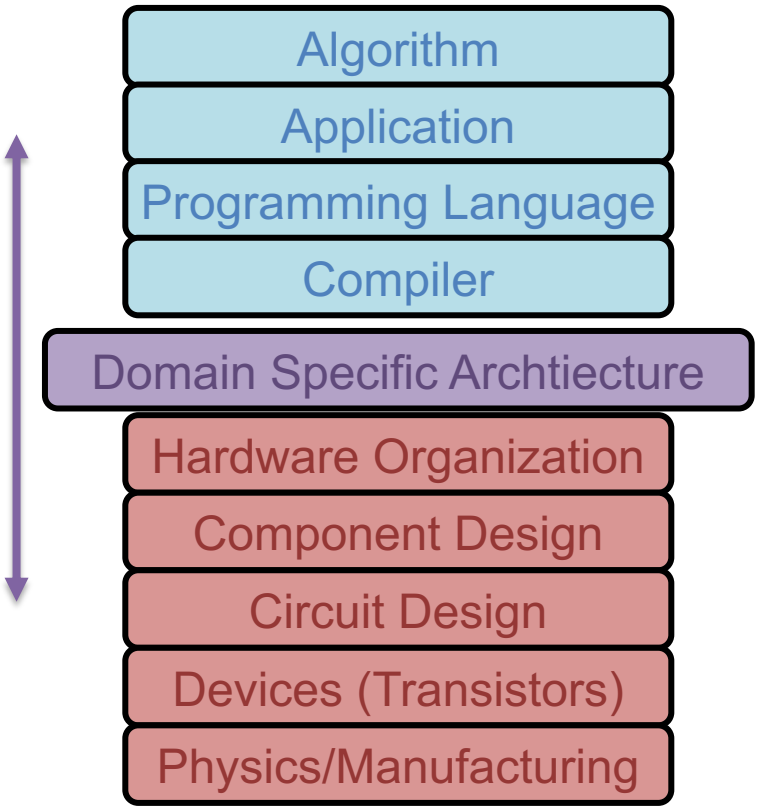


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten

New plot and data collected for 2010-2017 by K. Rupp

Learning from Domain-Specific

Co-optimize
for
Deep Learning



**Downside:
No longer
general
purpose**

Summary of Paradox:

- General purpose processors **stagnating**
- Machine learning processors **thriving**

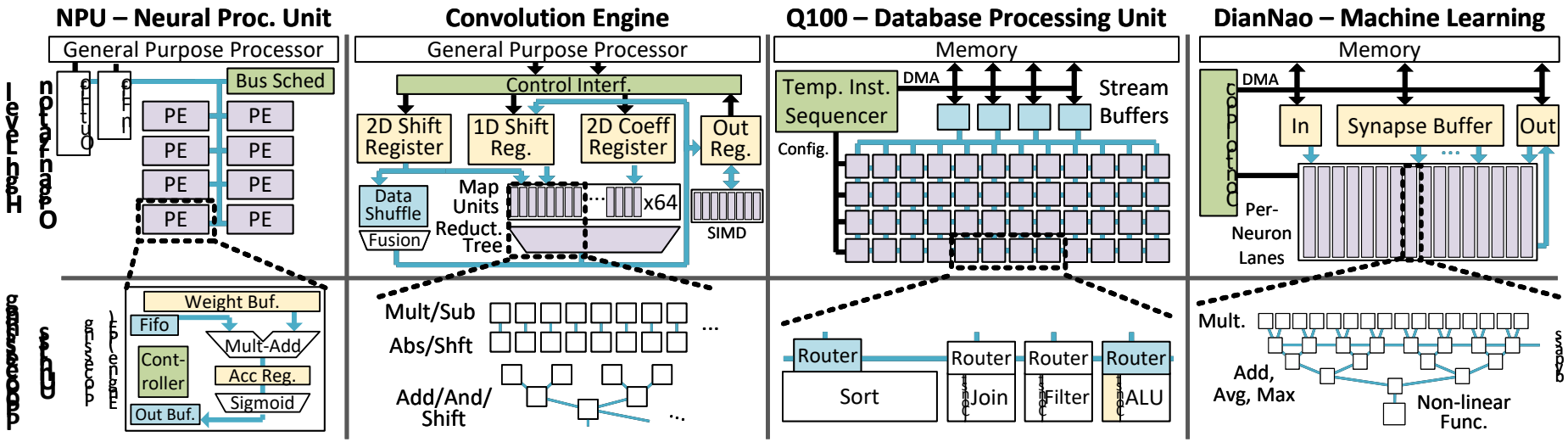
- Three key reasons:
  - General purpose processor design is inefficient
    - Bold Claim: because they are Von Neumann
  - No longer technology free ride
  - Lack Codesign Architecture

What can we do about it??

- Fundamental Architecture Challenge: How to get **performance** and **generality** without hurting **efficiency**

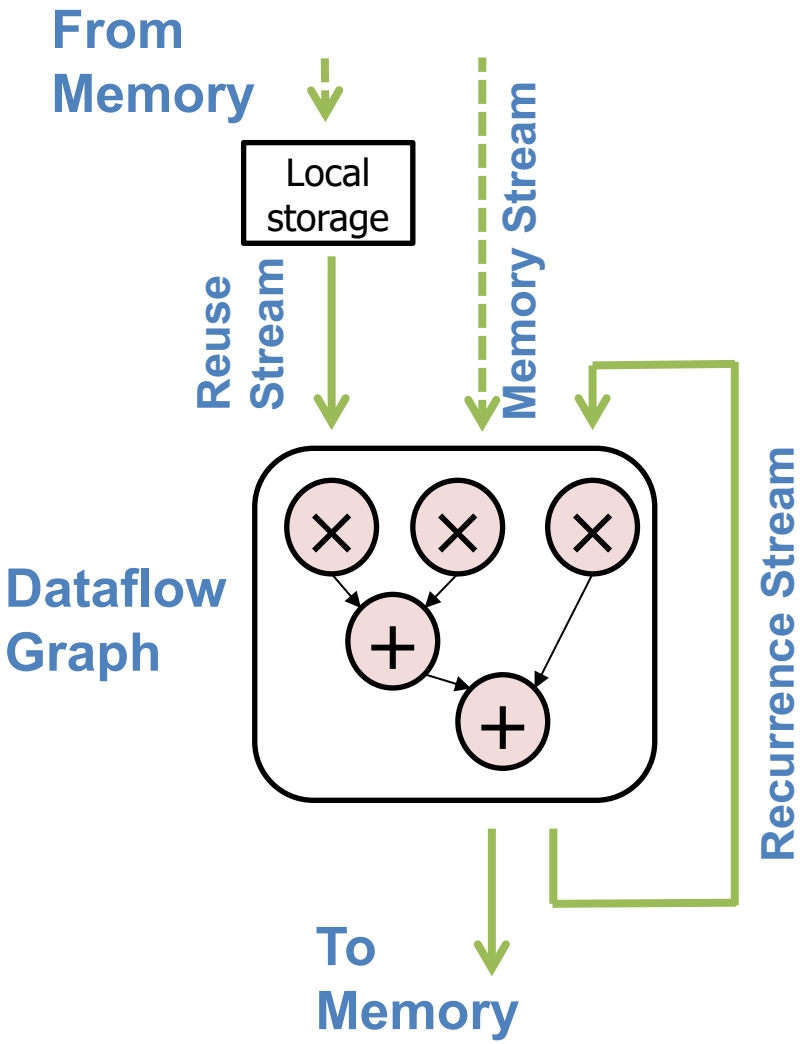
- Community has been focusing on increasing parallelism:
  - **Multicore Processors (thread-level parallelism)**
    - How to write efficient parallel programs?
    - How to create an efficient network for communication?
    - How to create an efficient storage hierarchy?
  - **General Purpose GPUs (data-level parallelism)**
    - How to design an efficient and general vector unit?
    - How to create automatic parallelizing compilers?
    - …
- **Our Approach: Figure out what domain-specific accelerators are doing right, and emulate them!**

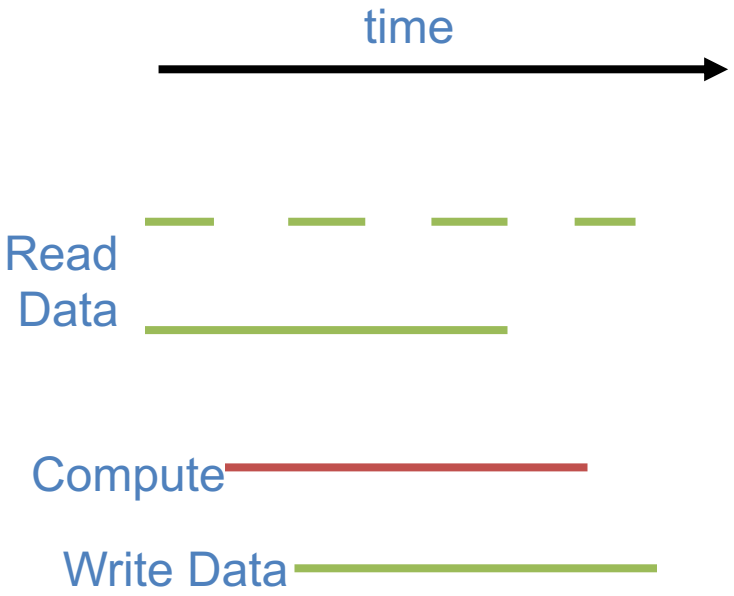
Find and distill common techniques across successful domain specific accelerators:



Parallelism Specialization (parallel PEs)	Data Reuse Specialization (scratchpads)	Immediate Data Use (direct. Comm.)	Alg. Simplicity (lightweight control unit)

Stream-Dataflow Execution Model





Fundamental Difference to Von Neumann: No overall sequential ordering.

Canonical Accelerator ISA & Execution Model?

Example Stream-Dataflow Program

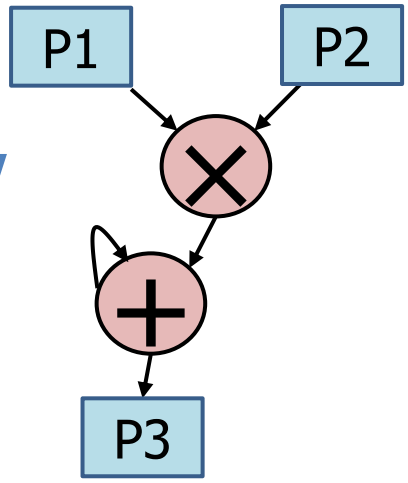
Original Program

```
for(int i = 0 to N) {
  c += a[i] * b[i];
}
```

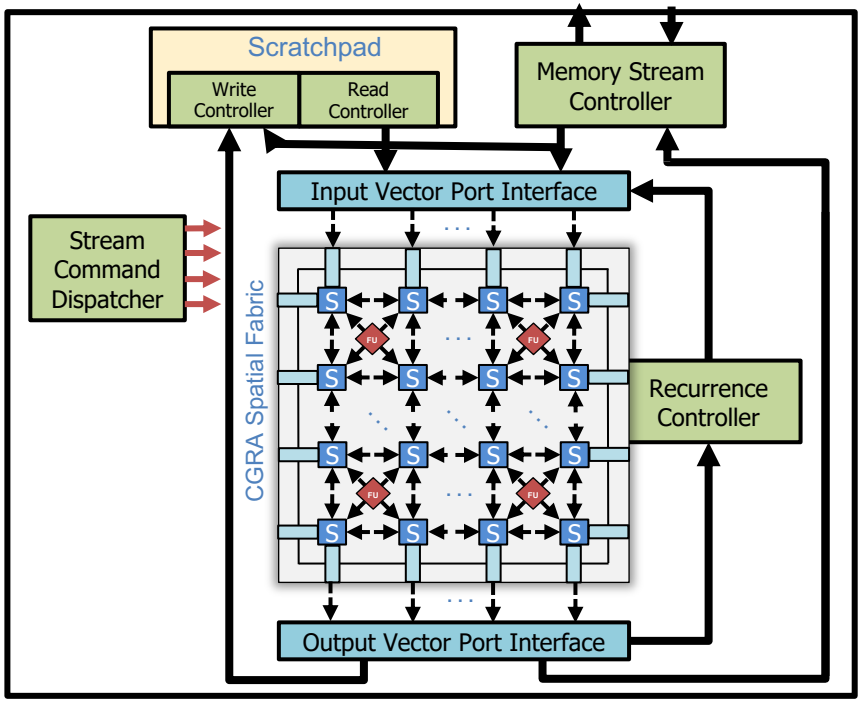
Stream-Code

```
SB_CONFIG(…)
Send a[i:i+N] -> P1
Send b[i:i+N] -> P2
Get P3 -> c
```

Dataflow Graph:



Stream Dataflow Processor



HPCA 2016,
ISCA 2017

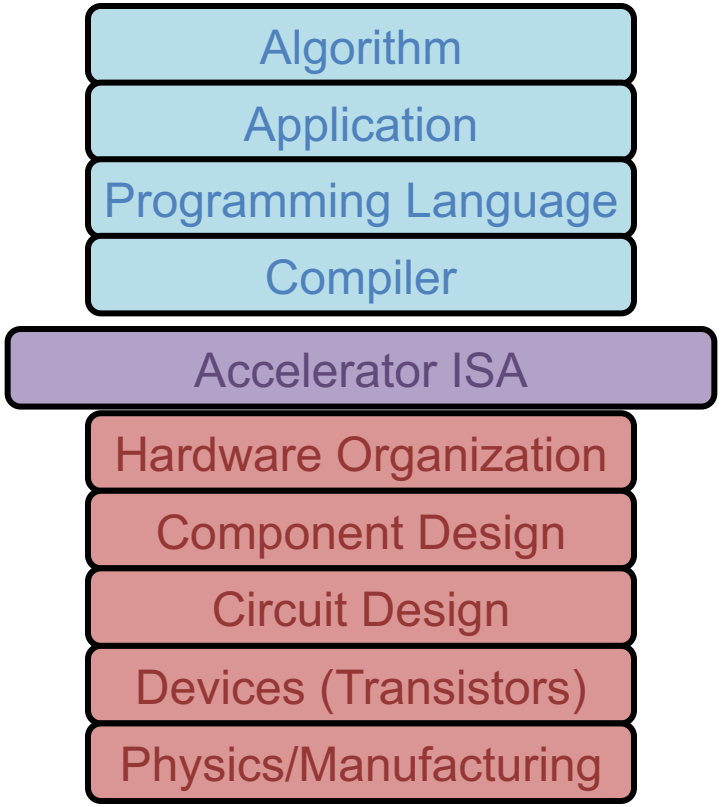
- **Dense Streaming:**
  - Workloads: Image proc, stream DB, deep neural
  - 10-100x speedup over CPU
- **5G Wireless:**
  - Workloads: Matrix factorization (qr,svd,cholesky)
  - 10x Faster than DSP architectures
- **Sparse data-processing:**
  - Workloads: sparse linear algebra, graph processing, irregular DB ops, GBDT

Upshot so far: Stream-dataflow ~= Domain Specific

Usually: usually within 2x power, 2x area of ASIC

Conclusions

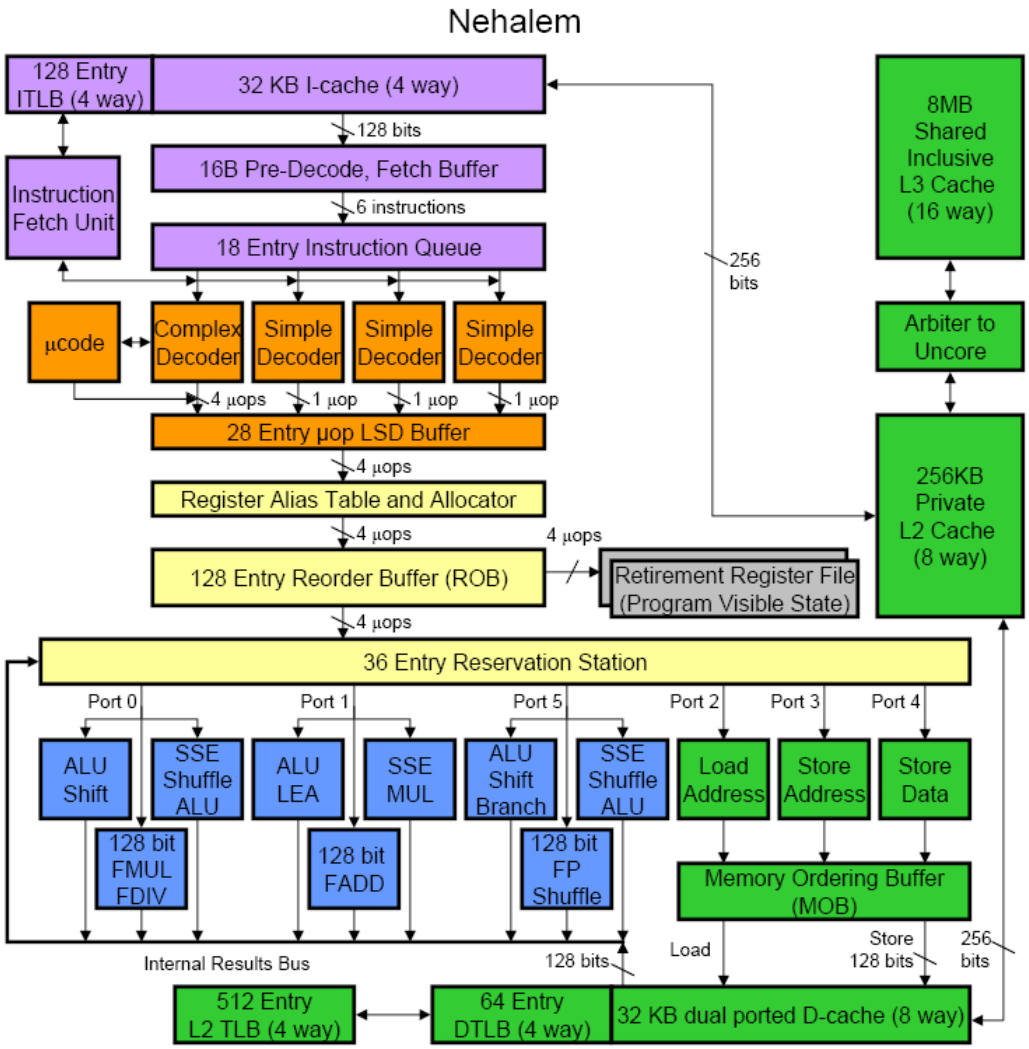
1. Traditional Von Neumann ISA and general purpose computers hard to improve...
2. Further co-design across devices, architecture and algorithms
3. Exciting time for architecture:
   - Industry/academics are finally willing to consider radically new architectures
   - Ample room and very large design space left to explore between general purpose and fully specialized



The end

- Questions?

General Purpose (2008)



Pentium 4 (2000) Was similar too111!!!