Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.

Seoul National University

Spring 2019

CPU Performance

Performance Issues

- Measure, analyze, report, and summarize
- Make intelligent choices
- See through the marketing hype
- Key to understanding underlying organizational motivation

- Questions
  - Why is some hardware better than others for different programs?
  - What factors of system performance are hardware related? (e.g., Do we need a new machine or a new operating system?)
  - How does the machine's instruction set affect performance?

Relative Performance

- Define

$$Performance = 1/Execution\ Time$$

- "X is *n* times faster than Y"

$$\frac{Performance_X}{Performance_Y} = \frac{Execution\ time_Y}{Execution\ time_X} = n$$

- Example: time taken to run a program
  - 10s on machine A, 15s on machine B
  - Execution $Time_B$ / Execution $Time_A$ = 15s / 10s = 1.5
  - Machine A is 1.5 times faster than machine B

CPU (Execution) Time

- "Iron law of CPU performance"

$$CPU\ Time = \frac{Seconds}{Program} = \frac{Cycles}{Program} \times \frac{Seconds}{Cycle}$$

$$= \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}$$

$$= Instruction\ Count \times CPI \times Clock\ Cycle\ Time$$

$$= \frac{Instruction\ Count \times CPI}{Clock\ Rate}$$

CPI

- (Average) Cycles Per Instruction

$$CPI = \frac{Clock\ Cycles}{Instruction\ Count} = \sum_{i=1}^{n} \left(CPI_i \times \frac{Instruction\ Count_i}{Instruction\ Count} \right)$$

- Example: CPI = 0.43 x 1 + 0.21 x 2 + 0.12 x 2 + 0.24 x 2 = 1.57

Instruction Class	Frequency	CPI_i
ALU operations	43%	1
Loads	21%	2
Stores	12%	2
Branches	24%	2

CPU Performance Factors

$$CPU\ Time = Instruction\ Count \times CPI \times Clock\ Cycle\ Time$$

	Instruction Count	CPI	Clock Cycle
Algorithm	○	△	
Programming language	○	○	
Compiler	○	○	
ISA	○	○	○
Microarchitecture		○	○
Technology			○

CPI for PIPE

- CPI ≃ 1.0
  - Fetch instruction each clock cycle
  - Effectively process new instruction almost every cycle
    (Although each individual instruction has latency of 5 cycles)
- CPI > 1.0
  - Sometimes must stall or cancel branches
- Computing CPI
  - *C* clock cycles
  - *I* instructions executed to completion
  - *B* bubbles injected (*C* = *I* + *B*)

$$CPI = C / I = (I + B) / I = 1.0 + B / I$$

Average penalty due to bubbles

Performance Penalties due to Bubbles

LP: Penalty due to load/use hazard stalling (LP = 0.25*0.20*1 = 0.05)	Typical value
Fraction of instructions that are loads	0.25
Fraction of load instructions requiring stall	0.20
Number of bubbles injected each time	1
MP: Penalty due to mispredicted branches (MP = 0.20*0.40*2 = 0.16)	
Fraction of instructions that are conditional jumps	0.20
Fraction of conditional jumps mispredicted	0.40
Number of bubbles injected each time	2
RP: Penalty due to ret instructions (RP = 0.02*3 = 0.06)	
Fraction of instructions that are returns	0.02
Number of bubbles injected each time	3

- Net effect of penalties: 0.05 + 0.16 + 0.06 = 0.27 (CPI = 1.27)

SPEC CPU Benchmark

- SPEC (Standard Performance Evaluation Corporation)
  - A non-profit organization that aims to "produce, establish, maintain and endorse a standardized set" of performance benchmarks for computers
  - CPU, Power, HPC (High-Performance Computing), Web servers, Java, Storage, …
  - http://www.spec.org

- SPEC CPU benchmark
  - An industry-standardized, CPU-intensive benchmark suite, stressing a system's processor, memory subsystem and compiler
    - Companies have agreed on a set of real program and inputs
    - Valuable indicator of performance (and compiler technology)
  - CPU89 → CPU92 → CPU95 → CPU2000 → CPU2006 → CPU2017

Benchmark Games

An embarrassed Intel Corp. acknowledged Friday that a bug in a software program known as a compiler had led the company to overstate the speed of its microprocessor chips on an industry benchmark by 10 percent. However, industry analysts said the coding error…was a sad commentary on a common industry practice of "cheating" on standardized performance tests…The error was pointed out to Intel two days ago by a competitor, Motorola …came in a test known as SPECint92…Intel acknowledged that it had "optimized" its compiler to improve its test scores. The company had also said that it did not like the practice but felt to compelled to make the optimizations because its competitors were doing the same thing…At the heart of Intel's problem is the practice of "tuning" compiler programs to recognize certain computing problems in the test and then substituting special handwritten pieces of code…

Saturday, January 6, 1996 New York Times

SPEC CPU2017

- SPECspeed suites
  - 10 integer benchmarks and 10 floating point benchmarks
  - Always run one copy of each benchmark
    - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine
    - Sun Microsystems' historical server: Sun Fire V490 with 2100MHz UltraSPARC-IV+ chips
  - Summarize as geometric mean of performance ratios:

$$\sqrt[n]{\prod_{i=1}^{n} Execution\ time\ ratio\ _i}$$

- SPECrate suites
  - 10 integer benchmarks and 13 floating point benchmarks
  - Run multiple concurrent copies of each benchmark (for throughput)

SPECspeed 2017 Suites

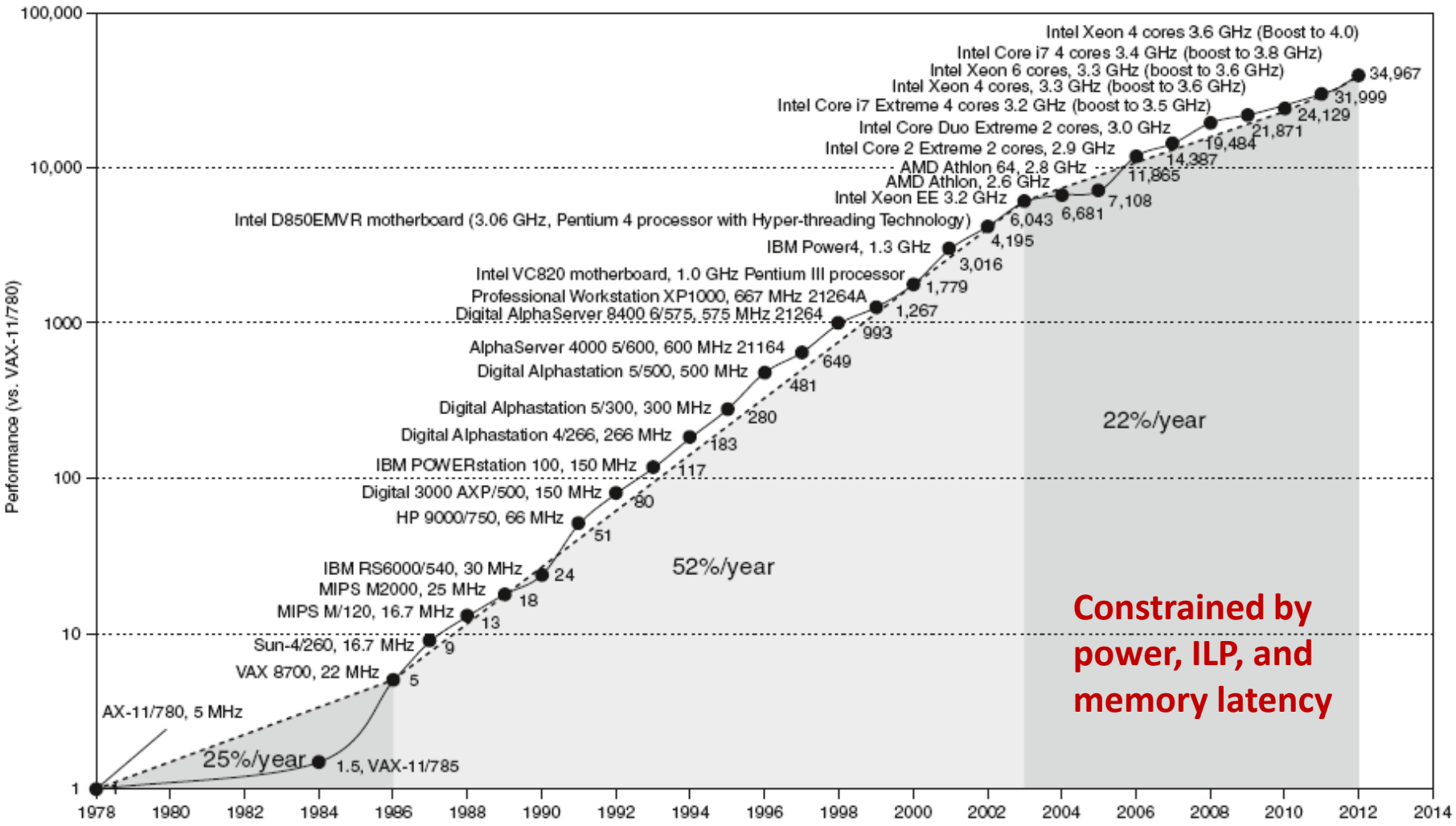
Integer Benchmarks (SPECspeed 2017 Integer)				Floating Point Benchmarks (SPECspeed 2017 Floating Point)			
Name	Lang.	KLOC	Application Area	Name	Lang.	KLOC	Application Area
perlbench	C	362	Perl interpreter	bwaves	Fortran	1	Explosion modeling
gcc	C	1304	GNU C compiler	cactuBSSN	C++, C, Fortran	257	Physics: relativity
mcf	C	3	Route planning	lbm	C	1	Fluid dynamics
omnetpp	C++	134	Discrete event simulation – computer network	wrf	Fortran, C	991	Weather forecasting
xalancbmk	C+	520	XML to HTML conversion via XSLT	cam4	Fortran, C	407	Atmosphere modeling
x264	C	96	Video compression	pop2	Fortran, C	338	Wide-scale ocean modeling
deepsjeng	C++	10	AI: alpha-beta tree search (Chess)	Imagick	C	259	Image manipulation
leela	C++	21	AI: Monte Carlo tree search (Go)	nab	C	24	Molecular dynamics
exchange2	Fortran	1	AI: Recursive solution generator (Sudoku)	fotonik3d	Fortran	14	Computational Electromagnetics
xz	C	33	General data compression	roms	Fortran	210	Regional ocean modeling

CINT2006 for Intel Core i7-920

- 4 cores @ 2.66GHz

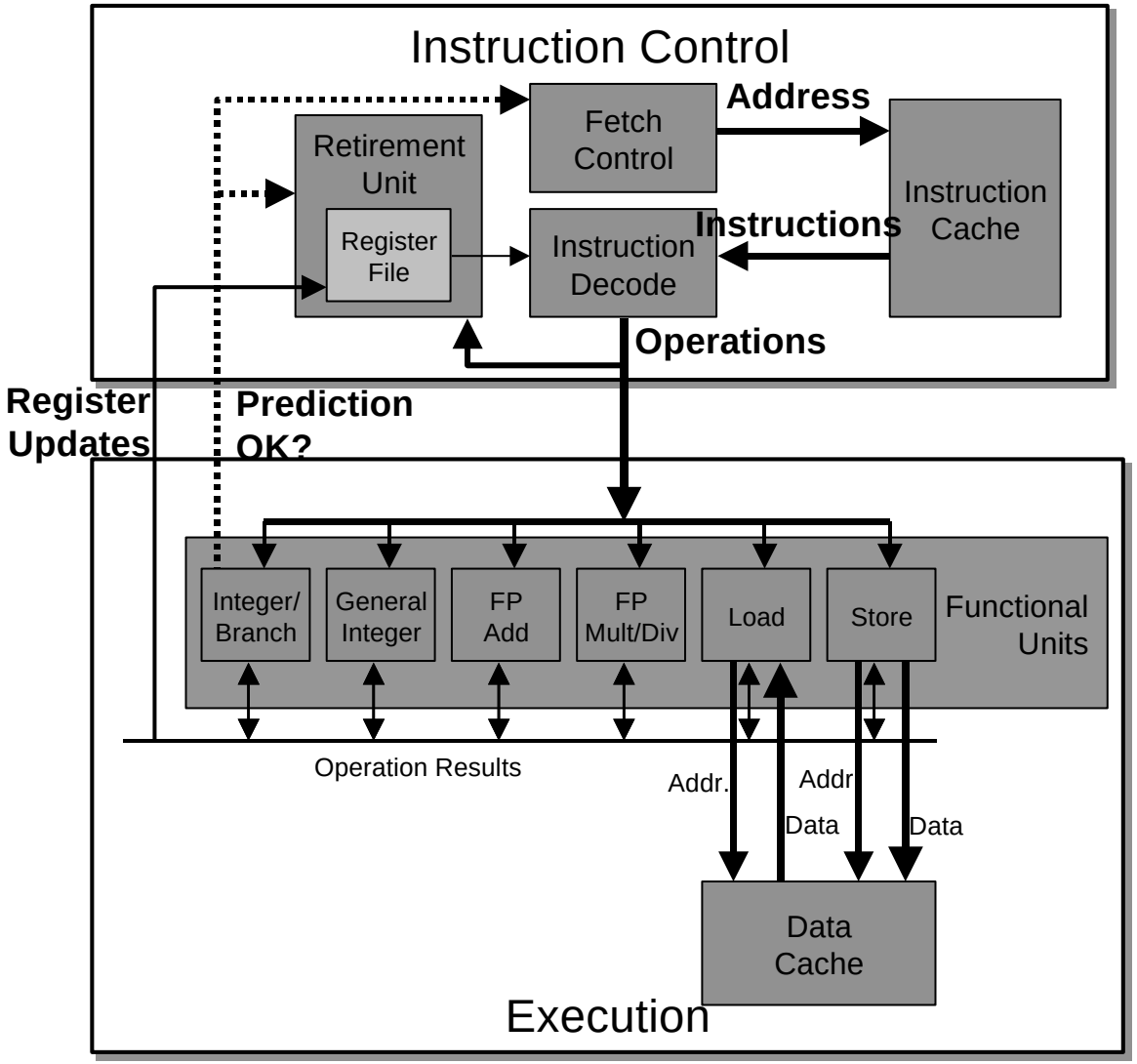
Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

Uniprocessor Performance

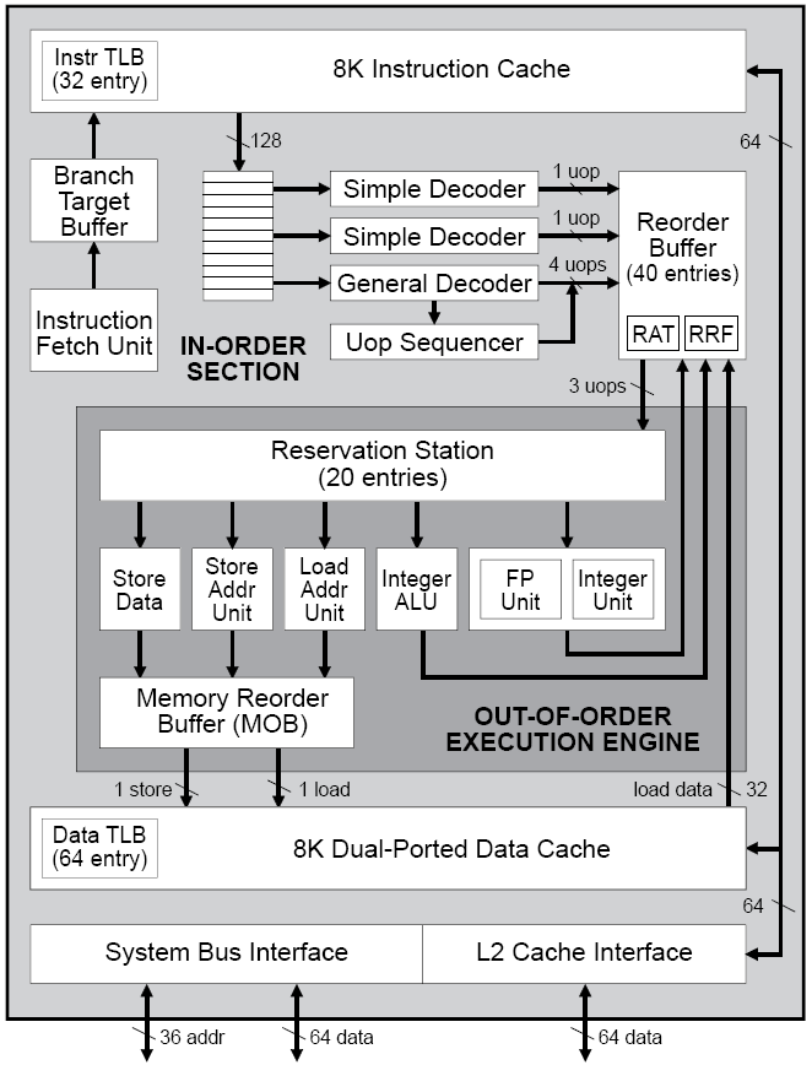


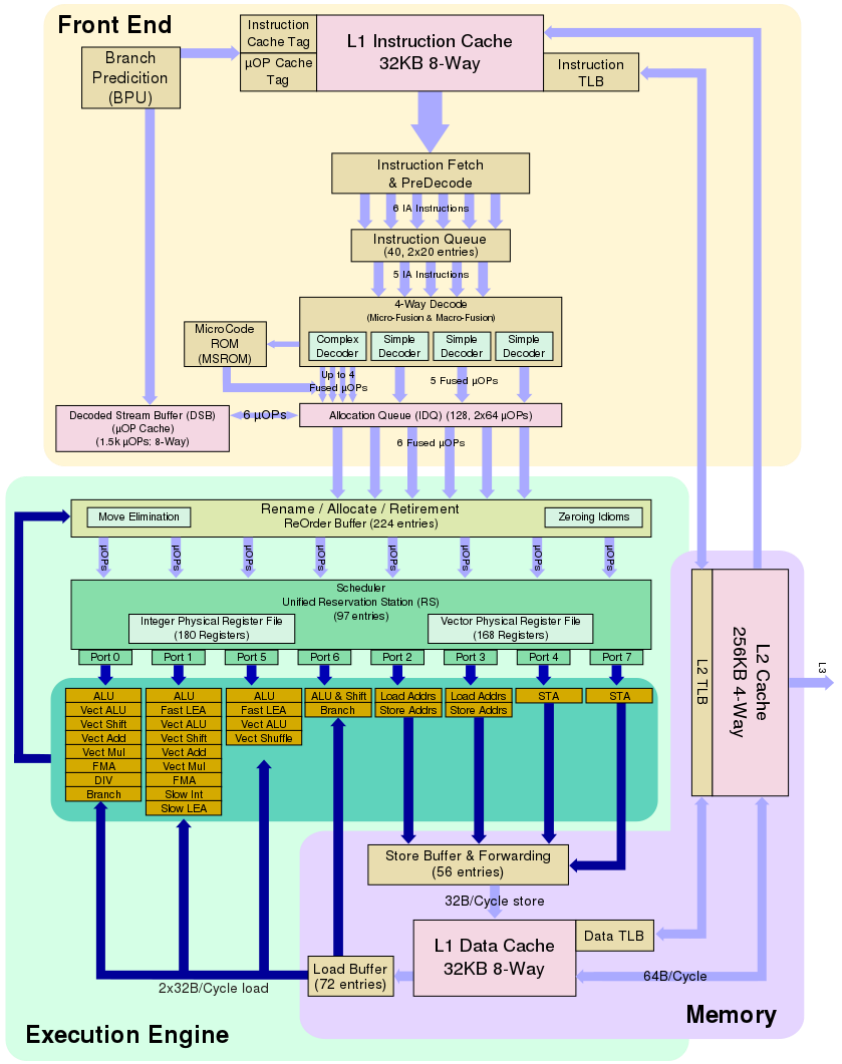
Modern CPU Design

Modern CPU Architecture



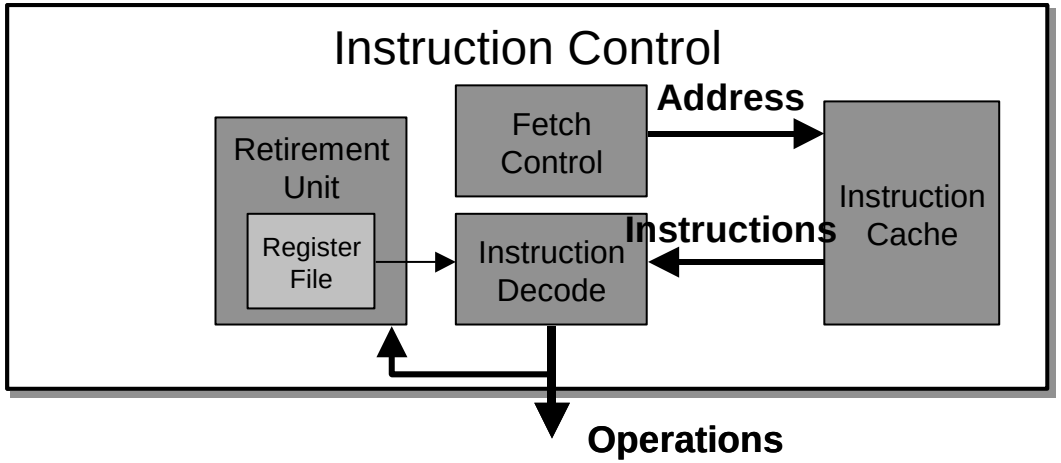
Intel P6 (1995) vs. Skylake (2015)





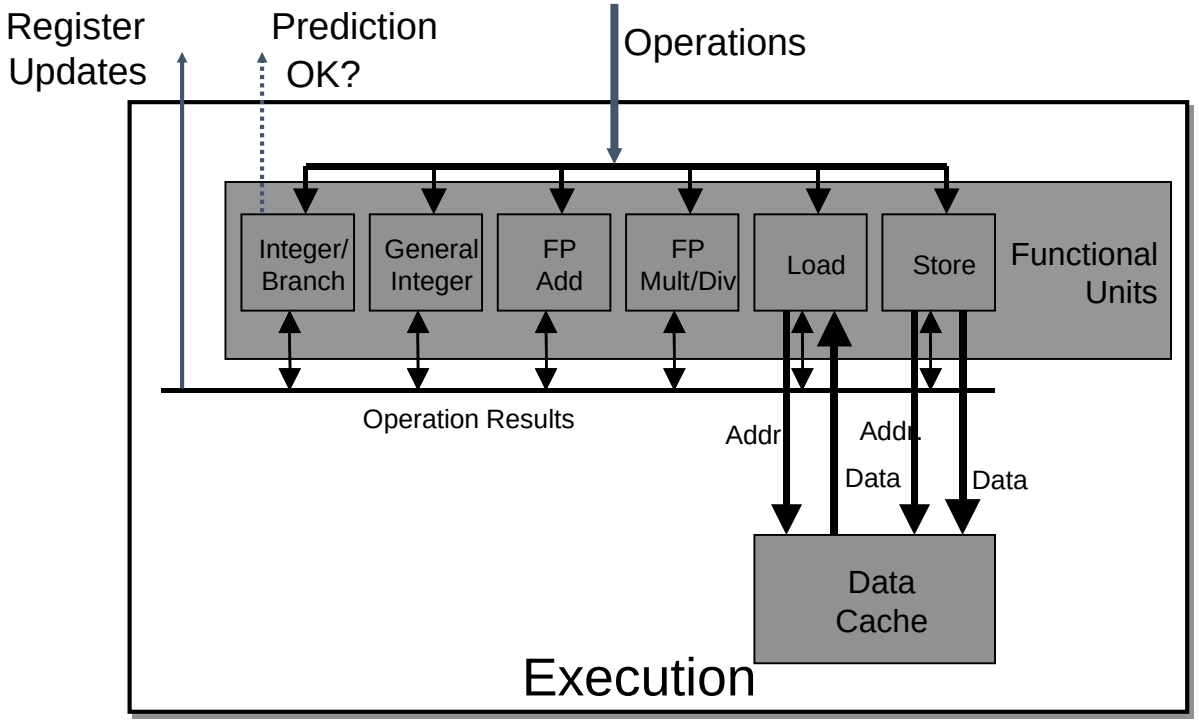
Instruction Control

- Grabs instruction bytes from memory
  - Based on current PC + Predicted targets for predicted branches
  - Hardware dynamically guesses whether branches taken/not taken and (possibly) branch target
- Translates instructions into Operations
  - Primitive steps required to perform instruction
  - Typical instruction requires 1-3 operations
- Converts register reference into Tags
  - Abstract identifier linking destination of one operation with sources of later operations



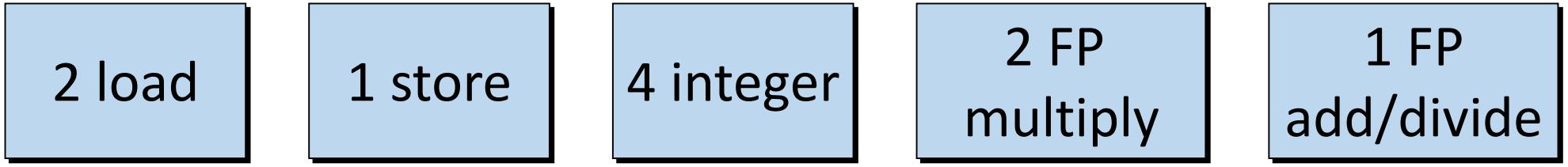
Execution Units

- Multiple functional units
  - Each can operate in independently
- Operations performed as soon as operands available
  - Not necessarily in program order
  - Within limits of functional units
- Control logic
  - Ensures behavior equivalent to sequential program execution



CPU Capabilities of Intel Haswell

- Multiple instructions can execute in parallel



- Some instructions take > 1 cycle, but can be pipelined

Instruction	Latency	Cycles/Issue
Load / Store	4	1
Integer Multiply	3	1
Integer Divide	3 – 30	3 – 30
Double/Single FP Multiply	5	1
Double/Single FP Add	3	1
Double/Single FP Divide	10 – 15	6 – 11

Haswell Operation

- Translates instructions dynamically into "μops"
  - ~ 118 bits wide
  - Holds operation, two sources, and destination

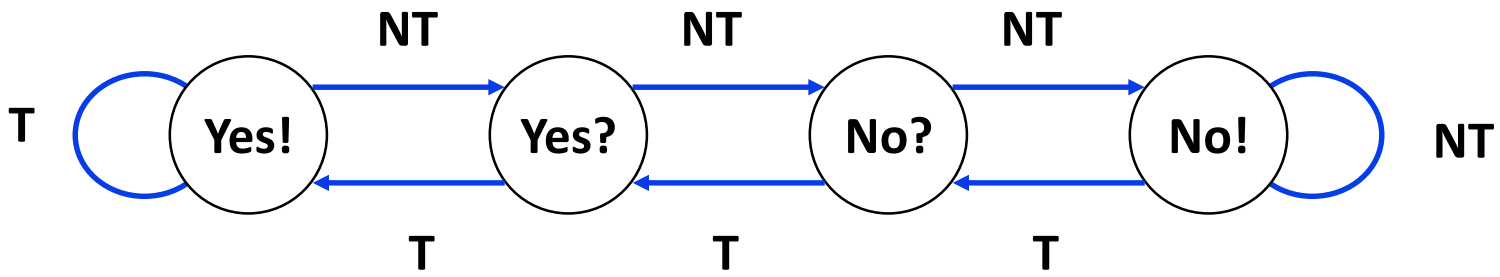
- Executes μops with "Out of Order" engine
  - μops executed when
    - Operands available
    - Functional unit available
  - Execution controlled by "Reservation Stations"
    - Keeps track of data dependencies between μops
    - Allocates resources

High-Performance Branch Prediction

- Critical to performance
  - Typically 11 – 15 cycle penalty for misprediction
- Branch Target Buffer (BTB)
  - 512 entries
  - 4 bits of history
  - Adaptive algorithm: Can recognize repeated patterns, e.g., alternating taken – not taken
- Handling BTB misses
  - Detect in ~cycle 6
  - Predict taken for negative offset, not taken for positive (Loops vs. conditionals)

Example Branch Prediction

- Branch history
  - Encode information about prior history of branch instructions
  - Predict whether or not branch will be taken



- State machine
  - Each time branch taken, transition to right
  - When not taken, transition to left
  - Predict branch taken when in state Yes! or Yes?

Processor Summary

- Design technique
  - Create uniform framework for all instructions
    - Want to share hardware among instructions
  - Connect standard logic blocks with bits of control logic
- Operation
  - State held in memories and clocked registers
  - Computation done by combinational logic
  - Clocking of registers/memories sufficient to control overall behavior
- Enhancing performance
  - Pipelining increases throughput and improves resource utilization
  - Must make sure to maintain ISA behavior