CS2410
Computer Architecture

Dept. of Computer Science
University of Pittsburgh

Xulong Tang
http://xzt102.github.io/

Course Administration

- **Instructor**: Xulong Tang (tax6@pitt.edu)
- **Lecture**: Monday 11:00am – 12:15pm
  Wednesday 11:00am – 12:15pm
- **Location**: SENSQ 5313
- **Logistic**: Virtual and remote lectures till 27th Jan 2022
  In person lectures after 27th Jan 2022
  Remote participation after 27th Jan has to be approved by the instructor
- **Recording**: Lectures are synchronized. No recording.

The course administration is subject to change based on the University policies (e.g., COVID policies)

Course Administration

- **Office hour**: SENSQ 6115
  Mon 12:20pm – 1:20pm
  Wed 12:20pm – 1:20pm
  Zoom office hour till Jan 27th 2022. zoom office hour by appointment afterwards.
- **TA**: Weizheng Xu
  Office Hours: **TBD**
- **URL**: Canvas
- **Text**: *Computer Architecture: A Quantitative Approach; John Hennessy and David Patterson. Sixth Edition - Morgan & Kaufmann.*

  Computer Organization and Design - The Hardware/Software Interface By David Patterson and John Hennessy Fifth Edition - Morgan & Kaufmann.
- **Slides**: PDF on Canvas after each lecture
- **ACK**: Profs. Rami Melhem(Pitt), Mahmut Kandemir (PSU)

Grading Information

- **Grade determinants**
  - **1 midterm exams (Mar 2nd) 20%**
  - **Final exam (Apr 20th comprehensive) 25%**
  - **1 Programming Project 20%**
  - **4 homeworks 20%**
    - » **To be submitted on Canvas by 23:59 on the due date. No late assignments will be accepted.**
  - **Reading Assignments 10%**
  - **Attendance 5%**
- **In person, in class, close note, synchronized** exams.
- Let the instructor and the TA know about exam conflicts ASAP
- Grades will be posted on Canvas
  - Must submit email request for change of grade after discussions with the TA (Projects/homeworks) or instructor (Reading/Exams)
- Always send emails via Canvas
- Attending the class is very important
  - I hope to see all students in the class room.
  - Remote (zoom) participation is only allowed with reasons approved by the instructor.
  - No recording.

Academic Integrity & Other Policies

- Please read the syllabus in Canvas
- Any apparent cases of cheating and collaboration on exams, project, or assignments will be treated as academic dishonesty.

5

Academic Integrity

- Students in this course will be expected to comply with the University of Pittsburgh's Policy on Academic Integrity. Any student suspected of violating this obligation for any reason during the semester will be required to participate in the procedural process, initiated at the instructor level, as outlined in the University Guidelines on Academic Integrity. This may include, but is not limited to, the confiscation of the examination of any individual suspected of violating University Policy. Furthermore, no student may bring any unauthorized materials to an exam, including dictionaries and programmable calculators.
- To learn more about Academic Integrity, visit the Academic Integrity Guide for an overview of the topic. For hands-on practice, complete the Understanding and Avoiding Plagiarism tutorial.

6

Covid-19 Resources at Pitt

- https://www.coronavirus.pitt.edu/
- **The Health Rules**
  - Wear your face covering during class.
  - Wash your hands thoroughly and often.
  - If you need to cough or sneeze, do so in a disposable tissue or your bended elbow.
  - If soap and water are not available, use hand sanitizer that's at least 70% alcohol.
  - Frequently clean high-touch surfaces within your area, like your desk and office doorknob using the supplied materials.
  - And more from https://www.coronavirus.pitt.edu/
  - Covid testing program https://patient.questdiagnostics.com/pitt-COVID-testing

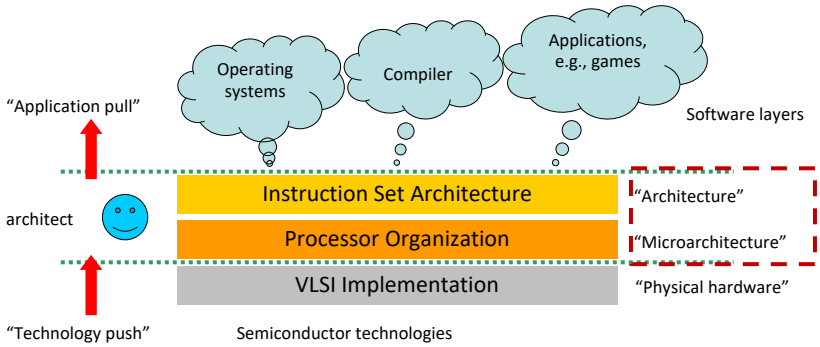
Course Schedule

- Based on the basics of computer architecture (CS1541)
  - In case you need review the basics:
    - » Computer Organization and Design - The Hardware/Software Interface By David Patterson and John Hennessy Fifth Edition - Morgan & Kaufmann.
- Schedule:
  - 2 weeks: review of the MIPS ISA and pipelined datapath design
  - 3 weeks: superscalar datapath design issues
  - 2 weeks: memory hierarchies and memory design issues
  - 3 weeks: multiprocessor/multicore design issues
  - 2 weeks: throughput processors and GPUs
  - Detailed schedule on syllabus.

Today's Class

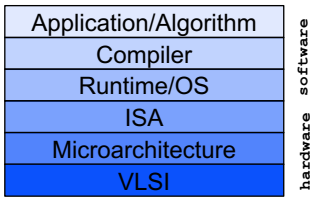
- Students on the waitlist will be enrolled automatically, after which they will be able to access Canvas.
- One quiz will be distributed
  - No credit and not mandatory
  - Background survey of computer architecture basics
  - Encourage all students to participate
  - Due this Friday(14th Jan)
- Office hours start 19th Jan.
- Lecture
  - What is computer architecture
  - Why we care about it
  - What is the trend (the past and the future)
  - How we measure performance

What is Computer Architecture? (§1.3)



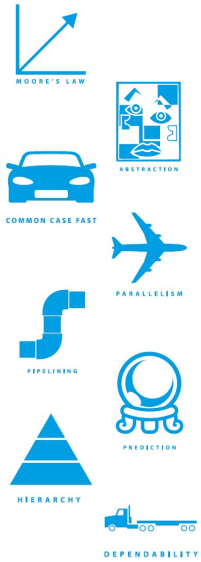
Models of Parallel executions:

- Instruction Level parallelism (ILP)
- Data-level parallelism (DLP)
- Thread-level parallelism (TLP)
- Request-level parallelism (RLP)
- Memory-level parallelism (MLP)



Eight Great Ideas in Computer Architecture

- Design for ***Moore's Law***
- Use ***abstraction*** to simplify design
- Make the ***common*** *case fast*
- Performance *via* ***parallelism***
- Performance *via* ***pipelining***
- Performance *via* ***prediction***
- *Hierarchy* of **memories**
- ***Dependability*** *via* redundancy



11

11

Flynn's Taxonomy

classic von Neumann

SISD **Single instruction stream** **Single data stream** `Single Core`	**(SIMD)** **Single instruction stream** **Multiple data stream** `GPUs, Vector machines`
MISD **Multiple instruction stream** **Single data stream** `TPUs`	**(MIMD)** **Multiple instruction stream** **Multiple data stream** `Multi-core`

Does it make sense?

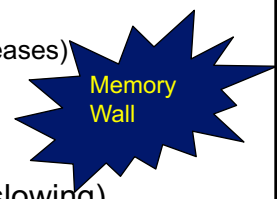
Yes, systolic array.

12

12

Trends in Technology(§1.4)

- Integrated circuit technology
  - Transistor density: +35%/year (feature size decreases)
  - Die size: +10-20%/year
  - Integration overall: +40-55%/year



- DRAM capacity: +25-40%/year (growth is slowing)
  (memory usage doubles every year)
- Flash capacity: +50-60%/year
  - 8-10X cheaper/bit than DRAM
- Magnetic disk technology: +40%/year
  - 8-10X cheaper/bit then Flash and 200-300X cheaper/bit than DRAM
- Clock rate stopped increasing

13

Bandwidth and Latency

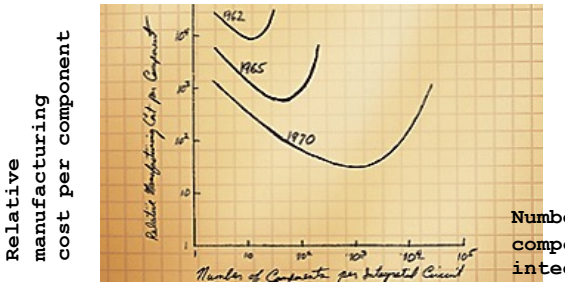
Latency lags bandwidth (in the last 40 years)

- Bandwidth or throughput
  - Total work done in a given time
  - 32,000 - 40,000X improvement for processors
  - 400 - 2400X improvement for memory and disks
- Latency or response time
  - Time between start and completion of an event
  - 50 - 90X improvement for processors
  - 8 - 9X improvement for memory and disks

14

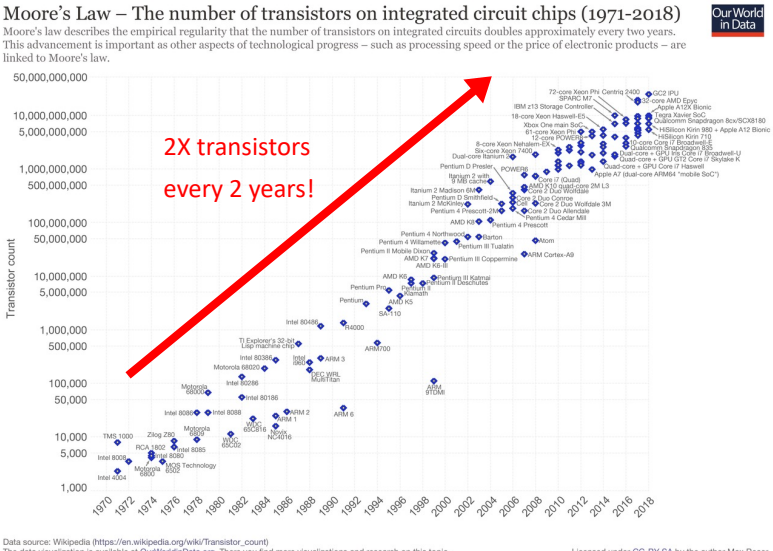
Transistors and wires

- Feature size
  - Minimum size of transistor or wire
  - 10 microns in 1971 to 22 nm in 2012 to16 nm in 2018 to 7nm and 2020 to 5nm
  - Transistor performance scales linearly
    - Wire delay per unit length does not improve with feature size!
  - Integration density scales quadratically



Moore's note (1965)

Moore's Law: 2X transistors / "2 years"



Dennard Scaling

- **In 1974, Robert H. Dennard observes that transistor dimensions could be scaled by -30% (0.7x) every technology generation.**
  - Their area reduces by 50%.
  - To keep the electric field constant, the voltage, V, is reduced by 30% (0.7x).
  - Circuit delays reduce by 30% (0.7x).
- **Therefore**
  - The 30% reduction in delay allows an increase in operating frequency, f, by about 40% (1.4x).
  - The 30% reduction in all distances and related 50% drop in areas lead to a decrease in capacitance, C, by 30%.
  - As a result, power consumption in turn decreases by 50%.
- **In a nutshell, in every technology generation, if the transistor density doubles (Moore's Law), the circuit becomes 40% faster, and power consumption (with twice the number of transistors) stays the same.**

17

Source: https://en.wikipedia.org/wiki/Dennard_scaling

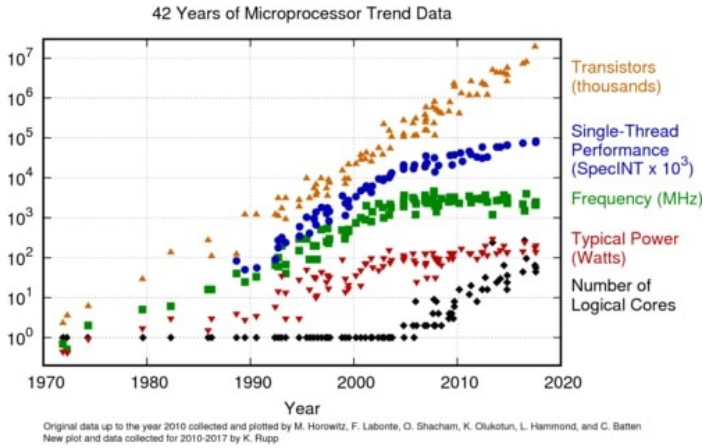
Technology Scaling Road Map (ITRS)

Year	2008	2010	2012	2014	2016	2018	2020
Feature size (nm)	45	32	22	14	10	7	5

- **Fun facts about 45nm transistors**
  - 30 million can fit on the head of a pin
  - You could fit more than 2,000 across the width of a human hair
  - If car prices had fallen at the same rate as the price of a single transistor has since 1968, a new car today would cost about 1 cent

18

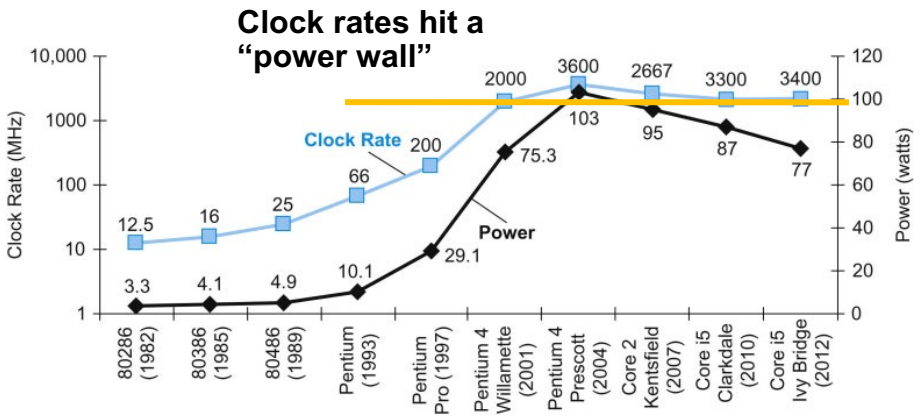
End of Moore's Law (and Dennard Scaling)



19

19

But What Happened to Clock Rates and Why?



- **In CMOS IC technology**

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

20

20

Reducing Power

- Suppose a new CPU has
  - 85% of the capacitive load of the previous generation
  - 15% voltage reduction, 15% slower clock

$$\frac{P_{new}}{P_{old}} = \frac{(C_{old} \times 0.85) \times (V_{old} \times 0.85)^2 \times (F_{old} \times 0.85)}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.52$$

- We have hit the power wall
  - We can't reduce the supply voltage much further (Dennard scaling is over), or the capacitive load
  - We can't remove more heat without new cooling technologies (e.g., liquid cooled)
- How can we increase the performance while lowering (or keeping the same) clock rate?

21

The Move to Multicore Processors

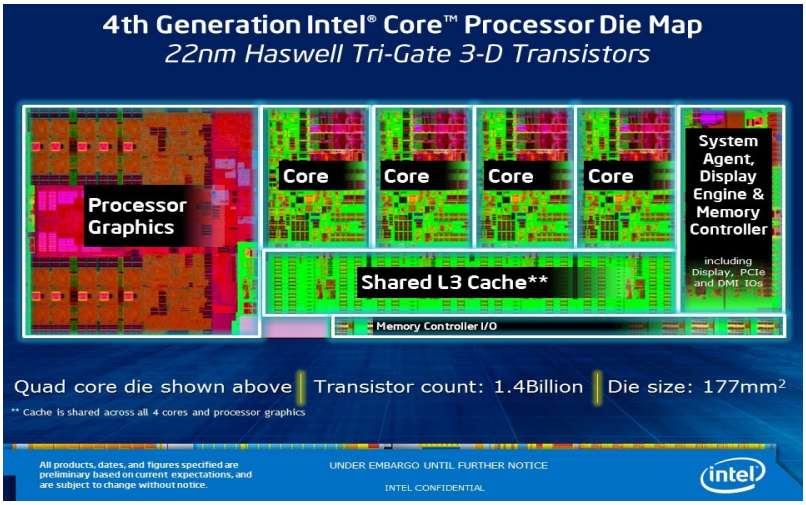
- The power challenge has forced a change in the design of microprocessors
- As of 2006 all server companies were shipping microprocessors with multiple cores per chip (processor)

Product	AMD Opteron X	Intel i7 Haswell	IBM Power 7+
Release date	2013	2013	2012
Technology	28nm bulk	22nm FFET	32nm SOI
Cores/Clock	4/2.0 GHz	4/3.5GHz	8/4.4 GHz
Power (TDP)	22W	84W	~120 W

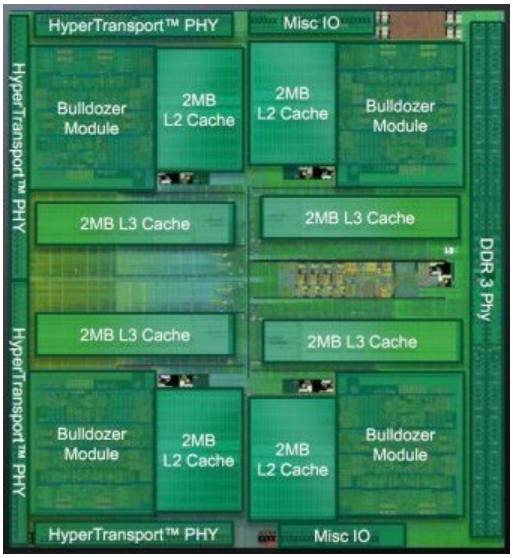
- Plan of record was to double the number of cores per chip per generation (about every two years)

22

Intel Haswell



AMD Opteron (Bulldozer)

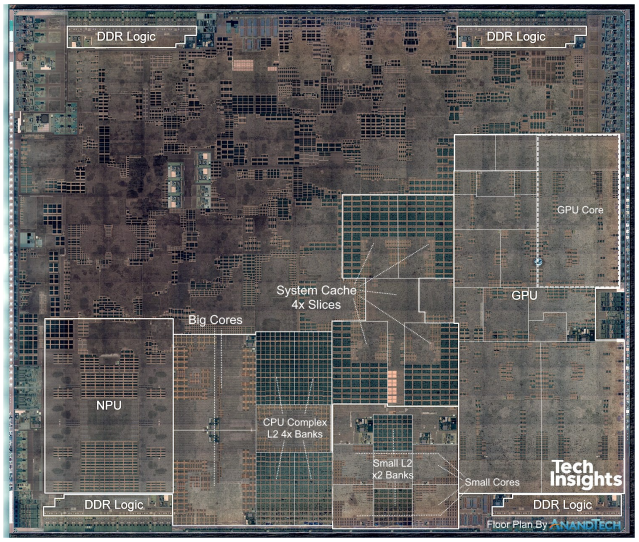


Apple A6 (iPhone 5)

25

25

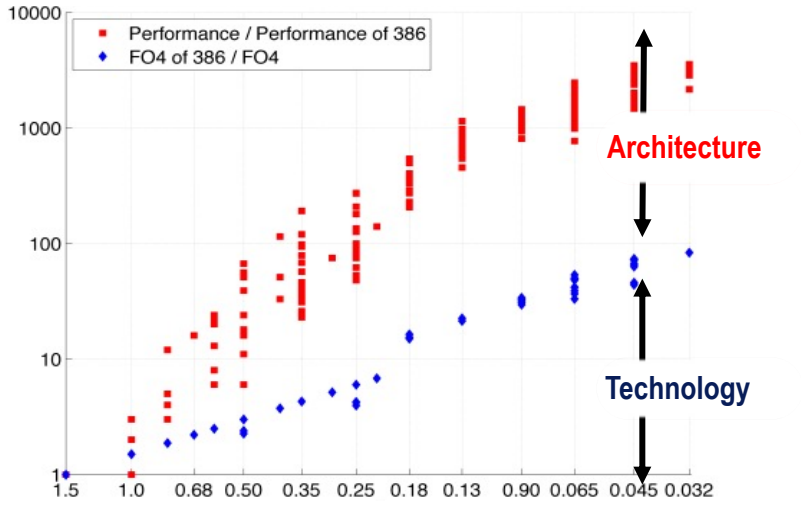
Apple A12



26

26

How much benefit from whom?



Danowitz et al., CACM 04/2012, Figure 1

27

The Trend

- End of Moore's Law and Dennard Scaling since 2004
  - Memory wall
    - » The growing disparity of speed between CPU and memory outside the CPU chip
  - Power wall
    - » The trend of consuming exponentially increasing power (and

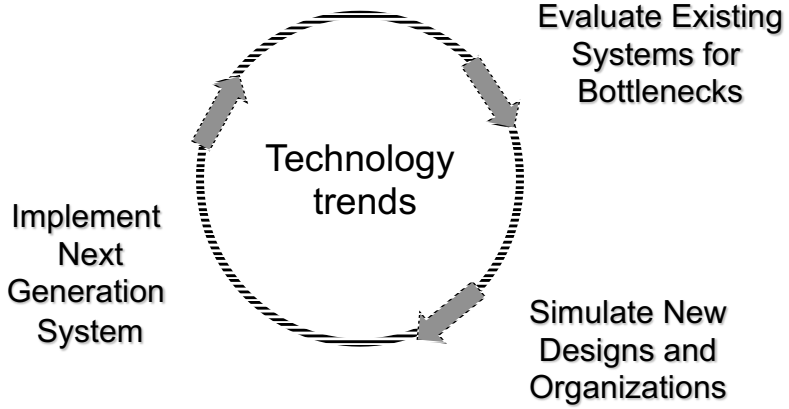
Computer architecture designs and innovations are the key to continue providing the application performances and speedups.

- Heterogeneous multi-core processors
  - NPU, Tensor core, Ray tracing core, etc
- Domain specific processors
  - AI and computer vision
  - Autonomous driving

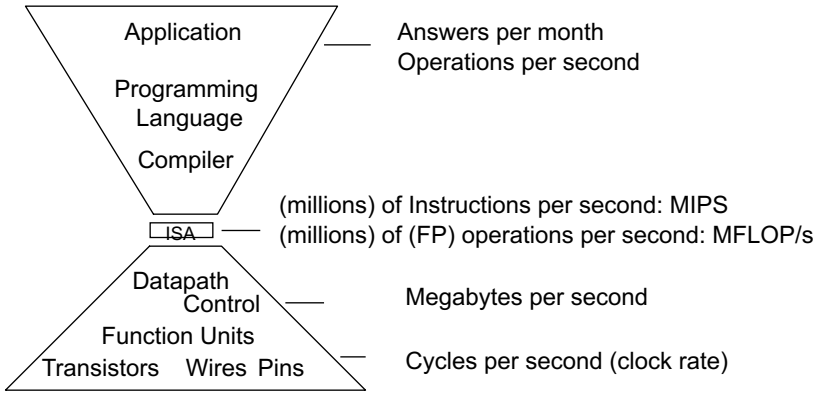
28

Source: https://en.wikipedia.org/wiki/Multi-core_processor
https://en.wikipedia.org/wiki/Random-access_memory

Computer Engineering Methodology



Performance (§1.8)



Measuring Performance

- **Time to run the task (latency)**
  - Execution time, response time, CPU time, …
- **Tasks per day, hour, week, sec, ns, …**
  - Throughput, bandwidth

Performance measurement Tools

- Hardware prototypes : Cost, delay, area, power estimation
- Simulation (many levels, ISA, RT, Gate, Circuit, …)
- Benchmarks (Kernels, toy programs, synthetic), Traces, Mixes
- Analytical modeling and Queuing Theory

35

Relative Performance

- To maximize performance, need to minimize execution time

$$\text{performance}_X = 1 / \text{execution\_time}_X$$

If computer X is n times faster than computer Y, then

$$\frac{\text{performance}_X}{\text{performance}_Y} = \frac{\text{execution\_time}_Y}{\text{execution\_time}_X} = n$$

- Decreasing response time almost always improves throughput

36

Relative Performance Example

- If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?

We know that A is n times faster than B if

$$\frac{\text{performance}_A}{\text{performance}_B} = \frac{\text{execution\_time}_B}{\text{execution\_time}_A} = n$$

The performance ratio is $\frac{15}{10} = 1.5$

So, A is said to be 1.5 times faster than B (or A is 50% faster than B!)

37

How to Summarize Performance

- Arithmetic mean (weighted arithmetic mean)
  - ex: tracks execution time: $\sum_{i=1}^{n} \frac{T_i}{n}$ or $\sum_{i=1}^{n} W_i * T_i$
- Harmonic mean (weighted harmonic mean) of rates
  - ex: track MFLOPS: $\frac{n}{\sum_{i=1}^{n} \frac{1}{Rate_i}}$
- Normalized execution time is handy for scaling performance (e.g., X times faster than Pentium 4)
- Geometric mean ==> $\sqrt[n]{\prod_{i=1}^{n} execution\_ratio_i}$

  where the execution ratio is relative to a reference machine

38

Performance Evaluation

- Good products created when we have:
  - Good benchmarks
  - Good ways to summarize performance
- For better or worse, benchmarks shape a field.
- Execution time and power are the main measure of computer performance!
- Reproducibility is important (should provide details of experiments)

Benchmarks

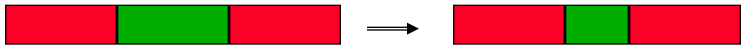
- SPEC2017: Standard Performance Evaluation Corporation
- PARSEC: Princeton Application Repository for Shared-Memory Computers
- MediaBench: Multimedia and embedded applications
- Transaction processing- TPC-C, SPECjbb
- Embedded Microprocessor Benchmark Consortium – EEMBC: Networking, telecom, digital cameras, cellular phones, ...
- Stanford parallel benchmarks: For parallel architecture and shared memory multiprocessors
- NAS: For massively parallel processor systems
- Rodinia: for GPU applications
- AI/DNN: MLperf, PMLB, MLbench

Amdahl's Law (§ 1.9)

Speedup due to some enhancement E:

$$Speedup_{overall} = \frac{ExTime_{withoutE}}{ExTime_{withE}} = \frac{Performance_{withE}}{Performance_{withoutE}}$$

Suppose that enhancement E accelerates a fraction of the task by a factor *S*, and the remainder of the task is unaffected



$$\frac{ExTime_{withE}}{ExTime_{withoutE}} = \left(1 - fraction_{enhanced}\right) + \frac{fraction_{enhanced}}{S}$$

Example: Floating point instructions can be improved to run 2X; but only 10% of actual instructions are FP. What is the overall speedup?

(1-10%) + 10%/2 = 95%. Speedup = 1/.95

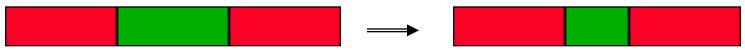
41

Amdahl's Law (§ 1.9)

Speedup due to some enhancement E:

$$Speedup_{overall} = \frac{ExTime_{withoutE}}{ExTime_{withE}} = \frac{Performance_{withE}}{Performance_{withoutE}}$$

Suppose that enhancement E accelerates a fraction of the task by a factor *S*, and the remainder of the task is unaffected



$$\frac{ExTime_{withE}}{ExTime_{withoutE}} = \left(1 - fraction_{enhanced}\right) + \frac{fraction_{enhanced}}{S}$$

Example: How much faster must the multiplier be to get a 2x performance improvement overall if multiples account for 20 seconds of the 100 second run time?

42

Computing CPU time

Average Cycles per Instruction (*CPI*) = $\sum_{j=1}^{n} CPI_j * F_j$

Where CPI_j is the number of cycles needed to execute instructions of type *j*, and F_j is the percentage (fraction) of instructions that are of type *j*.

Example: Base Machine (Reg / Reg)

Op	Freq	Cycles	$CPI_j * F_j$	(% Time)
ALU	50%	1	.5	(33%)
Load	20%	2	.4	(27%)
Store	10%	2	.2	(13%)
Branch	20%	2	.4	(27%)
			1.5	

Typical Mix

Instructions Per Cycle (IPC) = 1 /*CPI*

43

$$CPU\ time = Cycle\ time * \sum_{j=1}^{n} (CPI_j \ \mathbf{x}\ I_j)$$

Where I_j is the number of instructions of type *j*, and

Cycle time is the inverse of the *clock rate*.

$$CPU\ time = \text{total \# of instructions} \ \mathbf{x}\ CPI \ \mathbf{x}\ Cycle\ time$$

Example: For some programs,

Machine A has a clock cycle time of 10 ns. and a CPI of 2.0
Machine B has a clock cycle time of 20 ns. and a CPI of 1.2

What machine is faster for this program, and by how much?

44

A Simple Performance Tradeoff Example

Op	Freq	CPI_i	Freq x CPI_i			
ALU	50%	1	.5	.5	.5	.25
Load	20%	5	1.0	.4	1.0	1.0
Store	10%	3	.3	.3	.3	.3
Branch	20%	2	.4	.4	.2	.4
Average (Effective) CPI			$\sum$ = 2.2	1.6	2.0	1.95

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?
  CPU time new = 1.6 x IC x CC so 2.2/1.6 means 37.5% faster
- How does this compare with using branch prediction to shave a cycle off the branch time?
  CPU time new = 2.0 x IC x CC so 2.2/2.0 means 10% faster
- What if two ALU instructions could be executed at once?
  CPU time new = 1.95 x IC x CC so 2.2/1.95 means 12.8% faster

46

Aspects of CPU Performance

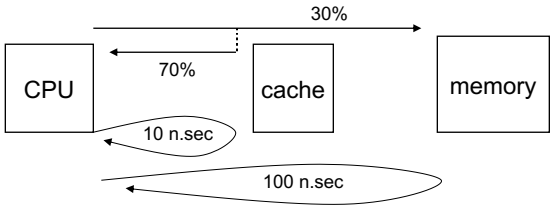
$$CPU\_time = \frac{Seconds}{program} = \frac{Instructions}{program} \times \frac{Cycles}{Instructions} \times \frac{Seconds}{Cycle}$$

	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X	X	
Inst. Set.	X	X	
Organization		X	X
Technology			X

47

Improving CPI using caches

An example:



What is the improvement (speedup) in memory access time?

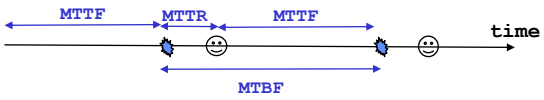
Caching works because of the principle of locality:

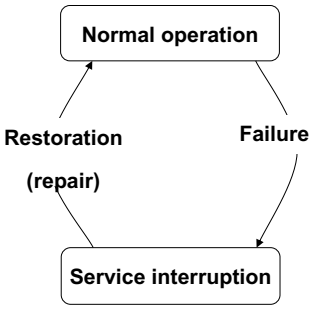
- Locality found in memory access instructions
  - Temporal locality: if an item is referenced, it will tend to be referenced again soon
  - Spatial locality: if an item is referenced, items whose addresses are close by tend to be referenced soon
- 90/10 locality rule
  - A program executes about 90% of its instructions in 10% of its code
- We will look at how this principle is exploited in various microarchitecture techniques

48

Dependability (§1.7)

- Fault: failure of a component
- Error: manifestation of a fault
- Faults may or may not lead to system failure





- **Reliability measure**: mean time to failure (MTTF)
- **Repair efficiency**: mean time to repair (MTTR)
- Mean time between failures
  MTBF = MTTF + MTTR
- Availability = MTTF / MTBF
- Improving Availability
  - Increase MTTF: fault avoidance, fault tolerance, fault forecasting
  - Reduce MTTR: improved tools and processes for diagnosis and repair

49