COMP 4611
DESIGN AND ANALYSIS OF COMPUTER ARCHITECTURES

http://course.cse.ust.hk/comp4611

Dr. GU Lin
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

Administrative Details

Instructor:

Dr. GU, Lin (the class of Tues., Thurs.)
Office #: 3562 (Lifts 25/26, 27/28)
Email: *lingu@cse.ust.hk*
Phone: 2358 6991
Office hours: Tuesdays: 16:30pm - 18:00pm
(or by appointments).

Teaching Assistants:

Arafet Ben Makhlouf, Zhiqiang Ma, Ke Hong

2

Administrative Details

Textbook

John L. Hennessy and David A. Patterson. ***Computer Architecture: A Quantitative Approach.*** Morgan Kaufman Publishers, 5th Edition, 2011. ISBN: 9780123838728.

Reference Book

David A. Patterson and John L. Hennessy. ***Computer Organization and Design: The Hardware/Software Interface****, 4th Edition*. Morgan Kaufmann Publishers, 2011. ISBN: 0123747503.

William Stallings. ***Computer Organization and Architecture: Designing for Performance****, 8th Edition*. Prentice Hall. ISBN: 0136073735.

Resources

Course web site: **http://course.cse.ust.hk/comp4611**
Class mailing lists: **comp4611@cse.ust.hk, csta4611@cse.ust.hk (for TAs)**
Information about lecture slides, office hours on course web site

3

Administrative Details

Tutorial sessions

Starts next week –
T1: Arafet Makhlouf
T2: Zhiqiang Ma
T3: Ke Hong

Schedule on course homepage

Grading Scheme

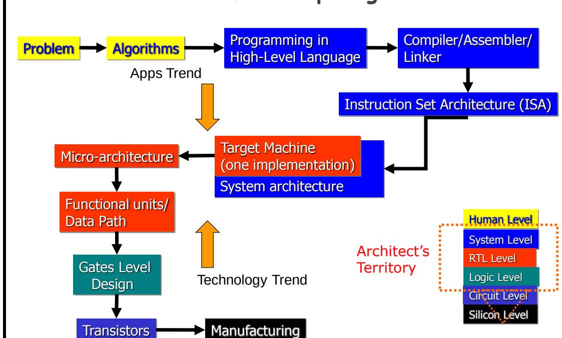
Homework: 30%.
Project: 15%.
Midterm: 20%.
Final Exam: 35%.

4

Project

- The project will be based on simulation of computer architectures.
  - Uses SimpleScalar as the simulator
  - Requires C programming
- The organization:
  - Group-based, up to 3 persons per group
  - Two tutorials cover materials related to the project

Breakdown of a Computing Problem



6

Course Description and Goal

What will COMP 4611 give me?

- Understanding of the ***internal design details*** of modern computers, their evolution, and trade-offs present at the hardware/software boundary.
- Understanding of the ***interaction*** and ***design*** of the various components at hardware level (processor, memory, I/O) and the software level (operating system, compiler, instruction sets).
- ***Intellectual preparation*** for dealing with a host of system design challenges.

7

Course Description and Goal (cont'd)

What have been making computers faster and cheaper? And how to continue the innovations?

- ***50 years of non-stop innovation – a technological miracle!***

Compare computers and automobiles

- ***The development of computers from the 1960's to the 2010's***

	Capacity (memory)	Speed (CPU)	Price
IBM7030 (Stretch)1961 	128KB	1.2 MIPS	US$13,500,000
i7 Desktop 2010 	2GB (typical)	2000+ MIPS	US$1,000

Course Description and Goal (cont'd)

Compare to computers and automobiles (cont'd)

- ***What would a car be like if they had innovated like a computer?***

	Capacity (passengers)	Speed (KM/hour)	Price
A good car in the 1960's	5	100	US$5,000
A car in the 2010's had it developed like computers	81920	166666	US$0.29

This class tells you how computer scientists and engineers created such a technological miracle!

Course Description and Goal (cont'd)

To understand the ***design techniques, machine structures, technology factors and evaluation methods*** that will determine the form of computers in the 21st Century

Technology

Programming Languages

Applications

Computer Architecture:
- Instruction Set Design
- Organization
- Hardware

Operating Systems

Measurement & Evaluation

History

10

Course Description and Goal (cont'd)

Will I use the knowledge gained in this subject in my profession?

Remember

- ***Few*** people design entire computers or entire instruction sets

But

- ***Many*** computer engineers design computer components
- Any ***successful*** computer engineer/architect needs to understand, in detail, all components of computers – in order to design any successful piece of hardware or software.

11

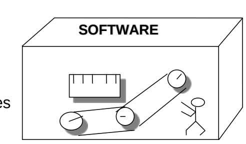
Computer Architecture in General

When we construct a building numerous practical considerations need to be taken into account:

- Available materials
- Worker skills
- Budget
- Space

Similarly, Computer Architecture is about working within constraints:

- What will the market buy?
- Cost/Performance
- Tradeoffs in materials and processes



12

Computer Architecture

Related to computer architecture –

- **Instruction set architecture (ISA):** The actual programmer-visible instruction set. Serves as the boundary between the software and hardware.

  e.g., mov r2, r1 (y = x;)
- **Implementation of a machine:**
  - **Organization:** Includes the high-level aspects of a computer's design such as: the memory system, the bus structure, the internal CPU unit which includes implementations of arithmetic, logic, branching, and data transfer operations.
  - **Hardware:** Refers to the specifics of the machine such as detailed logic design and packaging technology.

13

Three Computing Classes Today

- **Desktop/Laptop Computers**
  - Personal computer and workstation: US$500 – 5K
  - Optimized for price-performance
- **Servers**
  - Web server, file sever, computing sever: US$2K - 5M
  - Optimized for: availability, scalability, and throughput
- **Embedded Computers**
  - Fastest growing and the most diverse space: US$1 - 10K
    - Microwaves, washing machines, palmtops, cell phones, etc.
  - Optimizations: price, power, specialized performance

14

Three Computing Classes Today

Feature	Desktop	Server	Embedded
Price of the system	\$500-\$5K	\$2K-\$5M e.g., Web server, file sever, computing sever	\$1-\$100K (including network routers at high end) e.g. Microwaves, washing machines, palmtops, cell phones, network processors
Price of the processor	\$50-\$500	\$200-\$10K	\$0.01 - \$100
Sold per year	250M	6M	500M (only 32-bit and 64-bit)
Critical system design issues	Price-performance, graphics performance	Throughput, availability, scalability	Price, power consumption, application-specific performance

Desktops/Laptops (Personal Computers)

- Largest market in dollar terms
- Spans low-end (<\$500) to high-end (≈\$5K) systems
- Optimize price-performance
  - Performance measured in the number of calculations and graphic operations
  - Price is what matters to customers
- Arena where the newest, highest-performance and cost-reduced microprocessors appear
- Reasonably well characterized in terms of applications and benchmarking
- What will a PC of 2015 do?
- What will a PC of 2020 do?

16

Servers

- Provide more reliable file and computing services (Web servers)
- Key requirements
  - Availability – effectively provide service 24x7 (Yahoo!, Google, eBay)
  - Reliability
  - Scalability – server systems grow over time, so the ability to scale up the computing capacity is crucial
  - Performance – transactions per minute
- Related category: clusters / supercomputers
  - Question: how many server-class computers are there at Google?

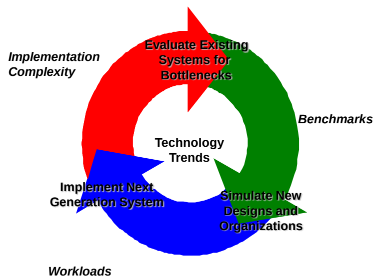
17

Embedded Computers

- Fastest growing portion of the market
- Computers as parts of other devices where their presence is not obviously visible
  - E.g., home appliances, printers, smart cards, cell phones, palmtops, set-top boxes, gaming consoles, network routers
- Wide range of processing power and cost
  - ≈\$0.1 (8-bit, 16-bit processors), \$10 (32-bit capable to execute 50M instructions per second), ≈\$100-\$200 (high-end video gaming consoles and network switches)
- Requirements
  - Real-time performance requirement (e.g., time to process a video frame is limited)
  - Minimize memory requirements
  - Low power
- SOCs (System-on-a-chip) combine processor cores and application-specific circuitry, DSP processors, network processors, …

18

The Task of a Computer Designer



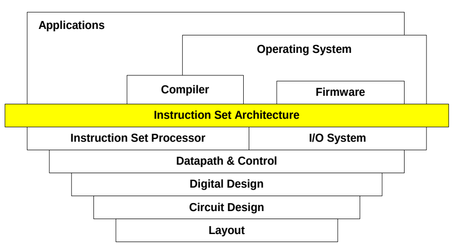
19

Job Description of a Computer Architect

- Make trade-off of performance, complexity, effectiveness, power, technology, cost, etc.
- Understand application requirements
  - General purpose Desktop (Intel Pentium class, AMD Athlon)
  - Game and multimedia (PS3: STI's Cell+Nvidia, Wii, Xbox 360)
  - Embedded and real-time (ARM, MIPS, XScale)
  - Online transactional processing (OLTP), data warehouse servers (Sun Fire T2000 (UltraSparc T1), IBM POWER (p690), Google Cluster)
  - Scientific (finite element analysis, protein folding, weather forecasts, defense related (IBM BlueGene, Cray T3D/T3E, IBM SP2))
  - Sometimes, there is no boundary …
- New emphases
  - Power Efficiency, Availability, Reliability, Security

20

Levels of Abstraction



S/W and H/W consists of hierarchical layers of abstraction, each hides details of lower layers from the above layer

The instruction set arch. abstracts the H/W and S/W interface and allows many implementation of varying cost and performance to run the same S/W

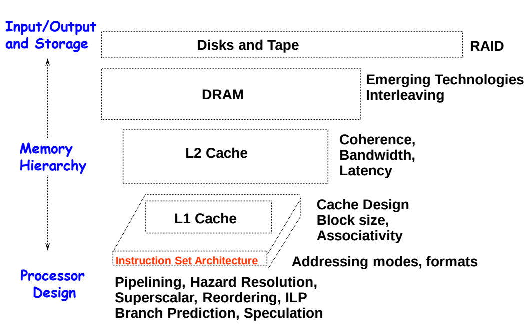
21

Topics to be covered in this class

- Fundamentals of Computer Architecture
- Instruction Set Architecture
- Pipelining & Instruction Level Parallelism
- Memory Hierarchy
- Input/Output and Storage Area Networks
- Multi-cores and Multiprocessors

22

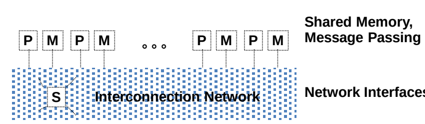
Computer Architecture Topics



23

Computer Architecture Topics

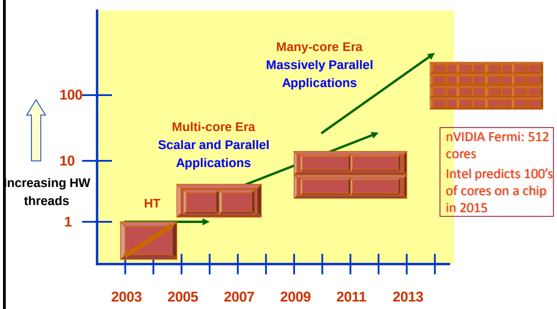
Multi-cores, Multiprocessors Networks and Interconnections



Topologies, Routing, Bandwidth, Latency, Reliability

24

Multiprocessing within a chip: Many-Core



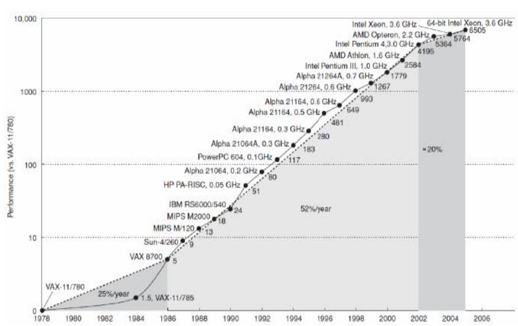
25

Trends in Computer Architectures

- Computer technology has been advancing at an alarming rate
  - ✓ You can buy a computer today that is more powerful than a supercomputer in the 1980s for 1/1000 the price.
- These advances can be attributed to advances in ***technology*** as well as advances in ***computer design***
  - – Advances in technology (e.g., microelectronics, VLSI, packaging, etc) have been fairly steady
  - – Advances in computer design (e.g., ISA, Cache, RAID, ILP, Multi-Cores, etc.) have a much bigger impact (***This is the focus of this class***).

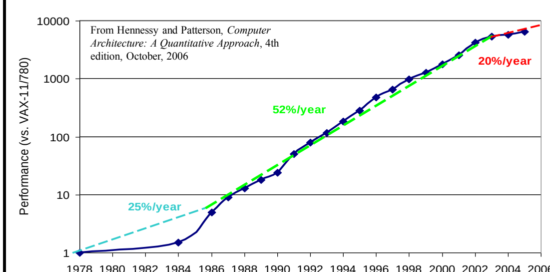
26

Processor Performance



27

Growth in processor performance



- VAX : 25%/year 1978 to 1986
- RISC + x86: 52%/year 1986 to 2002
- RISC + x86: 20%/year 2002 to present

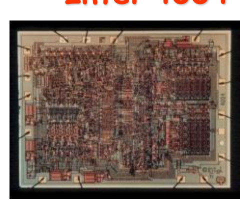
28

Trends in Technology

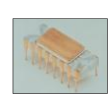
- Processor technology followed closely **Moore's Law**
  "*Transistor density of chips doubles every 1.5-2.0 years*"
- As a consequence of Moore's Law:
  - – Processor speed doubles every 1.5-2.0 years
  - – DRAM size doubles every 1.5-2.0 years
  - – Etc.
- These constitute a **target** that the computer industry aims for.
- Will Moore's Law continue?

29

Intel 4004 Die Photo



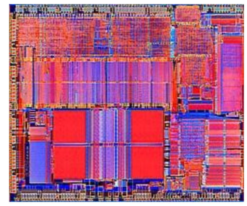
- Introduced in 1970
  - – First microprocessor
- 2,250 transistors
- 12 mm^2
- 108 KHz



Intel 8086 Die Scan

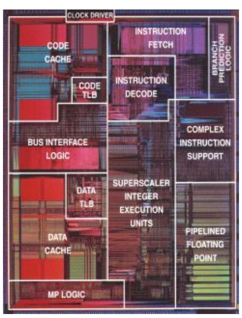
- Introduced in 1979
  - Basic architecture of the IA32 PC
- 29,000 transistors
- 33 mm^2
- 5 MHz

Intel 80486 Die Scan



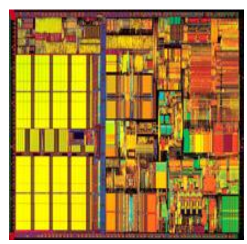
- Introduced in 1989
  - 1st pipelined implementation of IA32
- 1,200,000 transistors
- 81 mm^2
- 25 MHz

Pentium Die Photo



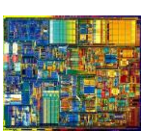
- Introduced in 1993
  - 1st superscalar implementation of IA32
- 3,100,000 transistors
- 296 mm^2
- 60 MHz

Pentium III

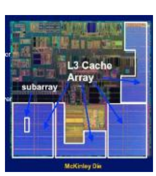


- Introduced in 1999
- 9,500,000 transistors
- 125 mm^2
- 450 MHz

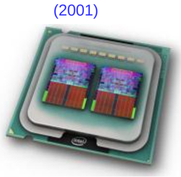
Pentium IV and Duo



Intel P4 – 55M tr (2001)



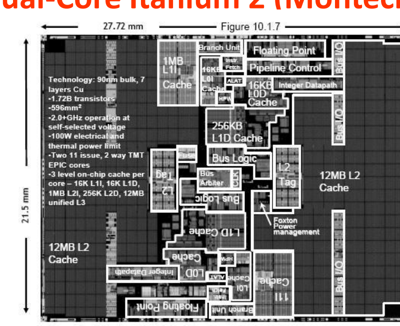
Intel Itanium – 221M tr. (2001)



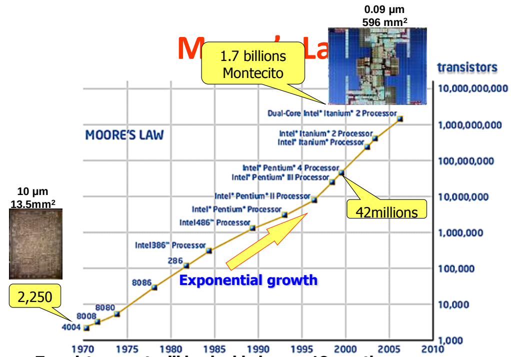
Intel Core 2 Extreme Quad-core 2x291M tr. (2006)

35

Dual-Core Itanium 2 (Montecito)

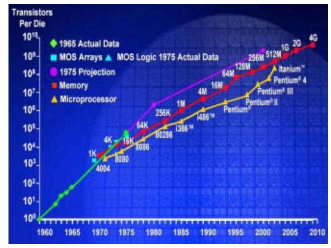


36



Transistor count will be doubled every 18 months
— Gordon Moore, Intel co-founder

Memory Capacity (Single Chip DRAM)

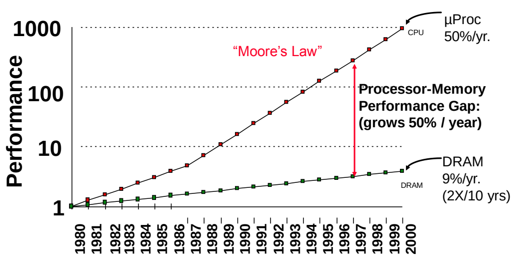


year	size(Mb)	cyc time
1980	0.0625	250 ns
1983	0.25	220 ns
1986	1	190 ns
1989	4	165 ns
1992	16	145 ns
1996	64	120 ns
2000	256	100 ns
2007	2G	52 ns

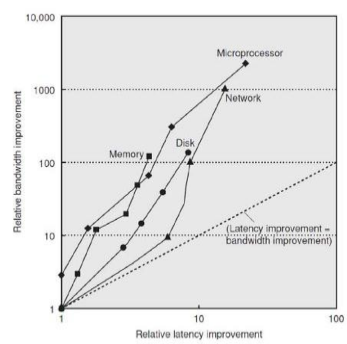
Moore's Law for Memory: Transistor capacity increases by 4x every 3 years

MOORE's LAW

Processor-DRAM Memory Gap (latency)



Latency vs. Performance (Throughput)

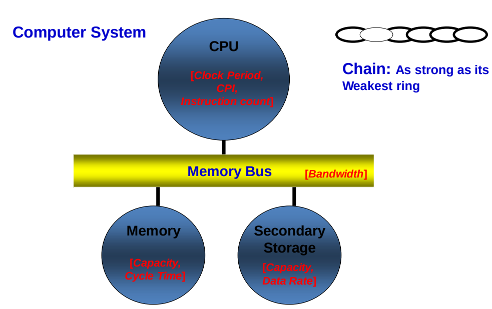


Technology Trends

	Capacity	Speed (latency)
Logic	2x in 3 years	2x in 3 years
DRAM	4x in 3 years	2x in 10 years
Disk	4x in 3 years	2x in 10 years

- Speed increases of memory and I/O have not kept pace with processor speed increases.
- This phenomenon is extremely important in numerous processing/computing devices

Processor-Memory Gap: We need a balanced Computer System



Cost and Trends in Cost

- *Cost is an important factor in the design of any computer system (except may be supercomputers)*
- *Cost changes over time*
  - *The learning curve and advances in technology lowers the manufacturing costs (**Yield**: the percentage of manufactured devices that survives the testing procedure).*
  - *High volume products lowers manufacturing costs (doubling the volume decreases cost by around 10%)*
    - *More rapid progress on the learning curve*
    - *Increases purchasing and manufacturing efficiency*
    - *Spreads development costs across more units*
  - *Commodity products decreases cost as well*
    - *Price is driven toward cost*
    - *Cost is driven down*

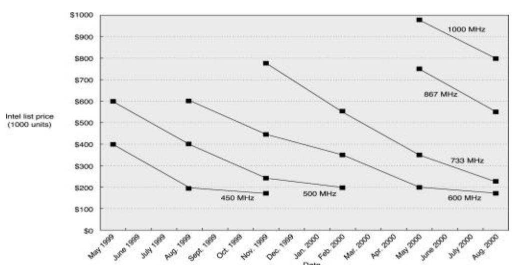
43

Cost, Price, and Their Trends

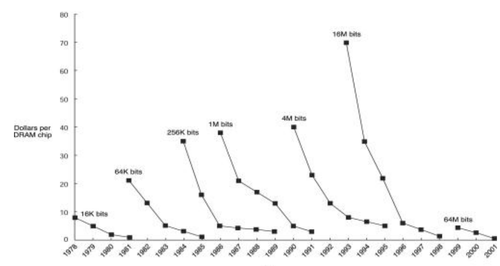
- Price – what you sell a good for
- Cost – what you spent to produce it
- Understanding cost
  - Learning curve principle – manufacturing costs decrease over time (even without major improvements in implementation technology)
    - Best measured by change in yield – the percentage of manufactured devices that survives the testing procedure
  - Volume (number of products manufactured)
    - doubling the volume decreases cost by around 10%)
    - decreases the time needed to get down the learning curve
    - decreases cost since it increases purchasing and manufacturing efficiency
  - Commodities – products sold by multiple vendors in large volumes which are essentially identical
    - Competition among suppliers lower cost

44

Processor Prices

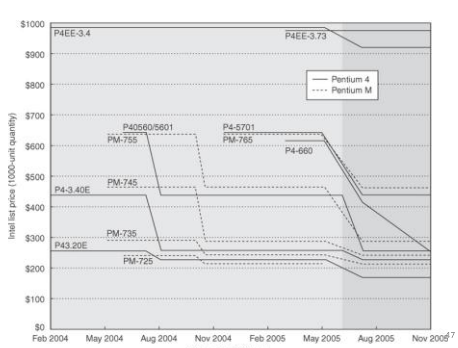


Memory Prices



46

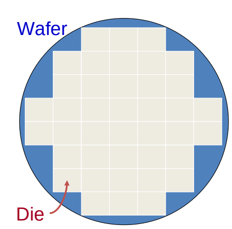
Trends in Cost: The Price of Pentium4 and PentiumM



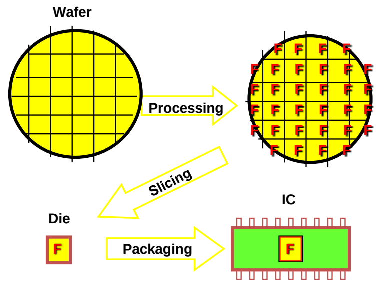
47

Integrated Circuit Costs

- Each copy of the integrated circuit appears in a die
- Multiple dies are placed on each wafer
- After fabrication, the individual dies are separated, tested, and packaged

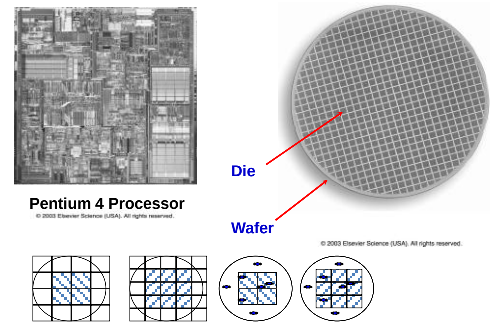


Wafer, Die, IC



49

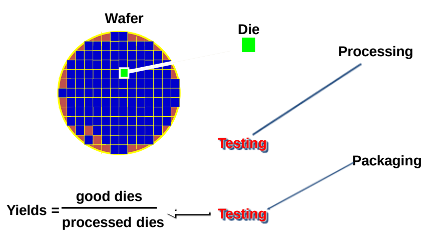
Integrated Circuit Costs



50

Integrated Circuit Costs

$$\text{IC Cost} = \frac{\text{Die Cost} + \text{Testing cost} + \text{Packaging Cost}}{\text{Final Test Yield}}$$



51

Integrated Circuit Costs

$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

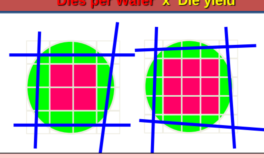
$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} \times \text{Die yield}}$$

52

Integrated Circuit Costs

$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} \times \text{Die yield}}$$



$$\text{Dies per Wafer} = \frac{\pi\,(\text{Wafer\_diameter}/2)^2}{\text{Die Area}} - \frac{\pi\,(\text{Wafer\_diameter})}{(2 * \text{Die Area})^{\frac{1}{2}}}$$

53

Example

- Find the number of dies per 20-cm wafer for a die that is 1.0 cm on a side and a die that is 1.5cm on a side

Answer

$$\text{Dies per Wafer} = \frac{\pi\,(\text{Wafer\_diameter}/2)^2}{\text{Die Area}} - \frac{\pi\,(\text{Wafer\_diameter})}{(2 * \text{Die Area})^{\frac{1}{2}}}$$

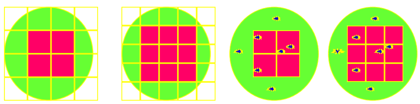
- 270 dies
- 107 dies

54

Integrated Circuit Costs

$$\text{Die Yield} = \text{Wafer Yield} * \left(1 + \frac{\text{Defects per unit area} * \text{Die\_Area}}{\alpha}\right)^{-\alpha}$$

Where α is a parameter inversely proportional to the number of mask Levels, which is a measure of the manufacturing complexity.
For today's CMOS process, good estimate is $\alpha = 3.0\text{-}4.0$



55

Integrated Circuits Costs

Die Cost goes roughly with (die area)^4

example :

defect density : 0.8 per cm^2
$\alpha = 3.0$

case 1: 1 cm x 1 cm
die yield = $(1+(0.8\text{x}1)/3)^{-3}$ = 0.49

case 2: 1.5 cm x 1.5 cm
die yield = $(1+(0.8\text{x}2.25)/3)^{-3}$ = 0.24

20-cm-diameter wafer with 3-4 metal layers : $3500

case 1 : 132 good 1-cm^2 dies, $27
case 2 : 25 good 2.25-cm^2 dies, $140

56

Other Costs

$$\text{Die Test Cost} = \frac{\text{Test equipment Cost} * \text{Ave. Test Time}}{\text{Die Yield}}$$

Packaging Cost: depends on pins, heat dissipation, beauty, ...

Chip	Die cost	Package			Test & assembly cost	Total
		pins	type	cost		
486DX2	$12	168	PGA	$11	$12	$35
Power PC 601	$53	304	QFP	$3	$21	$77
HP PA 7100	$73	504	PGA	$35	$16	$124
DEC Alpha	$149	431	PGA	$30	$23	$202
Super SPARC	$272	293	PGA	$20	$34	$326
Pentium	$417	273	PGA	$19	$37	$473

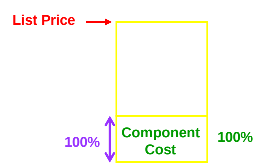
QFP: Quad Flat Package
PGA: Pin Grid Array
BGA: Ball Grid Array

57

Cost/Price

What is Relationship of Cost to Price?

Component Costs

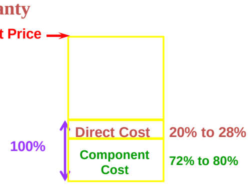


58

Cost/Price

What is Relationship of Cost to Price?

- Component Costs
- Direct Costs (add 25% to 40% to component cost) Recurring costs: labor, purchasing, scrap, warranty



59

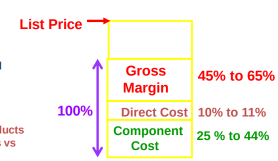
Cost/Price

What is Relationship of Cost to Price?

- Component Costs
- Direct Costs (add 25% to 40%) recurring costs: labor, purchasing, scrap, warranty
- Gross Margin (add 82% to 186%) nonrecurring costs: R&D, marketing, sales, equipment maintenance, rental, financing cost, pretax profits, taxes

PC's -- Lower gross margin
- Lower R&D expense
- Lower sales cost
  Mail order, Phone order, retail store...
- Higher competition
  Lower profit, volume sale,...

Gross margin varies depending on the products
High performance large systems vs
Lower end machines

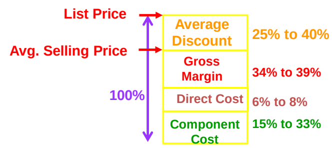


60

Cost/Price

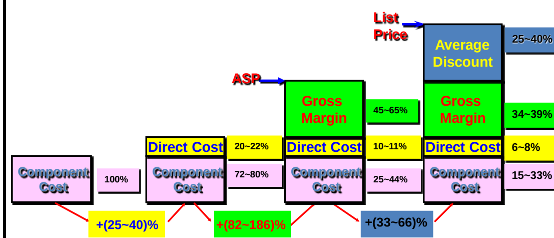
What is Relationship of Cost to Price?

- Component Costs
- Direct Costs (add 25% to 40%) recurring costs: labor, purchasing, scrap, warranty
- **Gross Margin (add 82% to 186%) nonrecurring costs:**
  R&D, marketing, sales,equipment maintenance, rental, financing cost, pretax profits, taxes
- **Average Discount to get List Price (add 33% to 66%): volume discounts and/or retailer markup**



Cost/Price

What is Relationship of Cost to Price?



Trends in Power in ICs

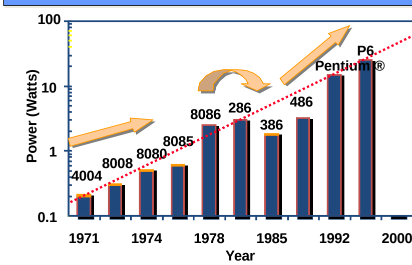
Power becomes a first-class architectural design constraint

- Power Issues
  - How to bring it in and distribute around the chip? (many pins just for power supply and ground, interconnection layers for distribution)
  - How to remove the heat (dissipated power)
- Why worry about power?
  - Battery life in portable and mobile platforms
  - Power consumption in desktops, server farms
    - Cooling costs, packaging costs, reliability, timing
    - Power density: 30 W/cm2 in Alpha 21364 (3x of typical hot plate)
  - Environment?
    - IT consumes a significant amount of energy
  - And performance!
    - Power and heating is limiting designers' ability to improve performance of processors!

Unless you are huntry – http://www.youtube.com/watch?v=zrg8nJ0bsUk

Why worry about power? -- Power Dissipation

Lead microprocessors power continues to increase



Power delivery and dissipation will be prohibitive

Source: Borkar, De Intel®

Performance Evaluation of Computers

Metrics for Performance

- The hardware performance is one major factor for the success of a computer system.

How to measure performance?

- A computer user is typically interested in reducing the ***response time (execution time)*** - the time between the start and completion of an event.
- A computer center manager is interested in increasing the ***throughput*** - the total amount of work done in a period of time.

An Example

Plane	DC to Paris [hour]	Top Speed [mph]	Passengers	Throughput [p/h]
Boeing 747	6.5	610	470	72 (=470/6.5)
Concorde	3	1350	132	44 (=132/3)

- Which has higher performance?
  - Time to deliver 1 passenger?
    - Concord is 6.5/3 = 2.2 times faster (120%)
  - Time to deliver 400 passengers?
    - Boeing is 72/44 = 1.6 times faster (60%)

67

Definition of Performance

- We are most interested in response time
- Performance [things/sec]

$$Performance(x) = \frac{1}{Execution\_time(x)}$$

- "X is n times faster than Y"

$$n = \frac{Execution\_time(y)}{Execution\_time(x)} = \frac{Performance(x)}{Performance(y)}$$

- As faster means both increased performance and decreased execution time, to reduce confusion will use "improve performance" or "improve execution time"

68

Computer Performance Evaluation: Cycles Per Instruction (CPI) – CPU Performance

- Sometimes, instead of using response time, we use ***CPU time*** to measure performance.
- ***CPU time*** can also be divided into ***user CPU time*** (program) and ***system CPU time*** (OS).

- The CPU time performance is probably the most accurate and fair measure of performance

69

Unix Times

- Unix time command report:

  90.7u 12.9s 2:39 65%

  - Which means
    - User CPU time is 90.7 seconds
    - System CPU time is 12.9 seconds
    - Elapsed time is 2 minutes and 39 seconds
    - Percentage of elapsed time that is CPU time is:

$$\frac{90.7 + 12.9}{159} = 0.65$$

70

Cycles Per Instruction (CPI) – CPU Performance

- Most computers run synchronously utilizing a CPU clock running at a constant clock rate:

  where: Clock rate = 1 / clock cycle

- A computer machine instruction is comprised of a number of elementary or micro-operations which vary in number and complexity depending on the instruction and the exact CPU organization and implementation.
  - **A micro-operation is an elementary hardware operation that can be performed during one clock cycle.**
  - **This corresponds to one micro-instruction in microprogrammed CPUs.**
  - **Examples: register operations: shift, load, clear, increment, ALU operations: add, subtract, etc.**
- Thus a single machine instruction may take one (or less than one) or more cycles to complete termed as the ***Cycles Per Instruction (CPI).***

71

CPU Performance Equation

CPU time = CPU clock cycles for a program X Clock cycle time

or:

CPU time = CPU clock cycles for a program / clock rate

CPI (clock cycles per instruction):

CPI = CPU clock cycles for a program / I

where I is the dynamic instruction count.

72

CPU Execution Time: The CPU Equation

- Instruction count (I): A program is comprised of a number of instructions
  - Measured in: instructions/program
- CPI: the average number of cycles for an instruction to be complete.
  - Measured in: cycles/instruction
- Cycle time (C): CPU has a fixed clock cycle time. C = 1/clock rate
  - Measured in: seconds/cycle
- CPU execution time is the product of the above three parameters as follows:

CPU Time = I x CPI x C

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

73

CPU Execution Time

For a given program and machine:

CPI = Total program execution cycles / Instructions count

CPU clock cycles = Instruction count x CPI

CPU execution time

= CPU clock cycles x Clock cycle
= Instruction count x CPI x Clock cycle
= I x CPI x C

74

CPU Execution Time: Example

- A Program is running on a specific machine with the following parameters:
  - Total instruction count: 10,000,000 instructions
  - Average CPI for the program: 2.5 cycles/instruction.
  - CPU clock rate: 200 MHz.
- What is the execution time for this program:

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

CPU time = Instruction count x CPI x Clock cycle
= 10,000,000 x 2.5 x 1 / clock rate
= 10,000,000 x 2.5 x $5\text{x}10^{-9}$
= .125 seconds

75

Factors Affecting CPU Performance

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

	Instruction Count I	CPI	Clock Cycle C
Program	X	X	
Compiler	X	X	
Instruction Set Architecture (ISA)	X	X	
Organization		X	X
Technology			X

76

Performance Comparison: Example

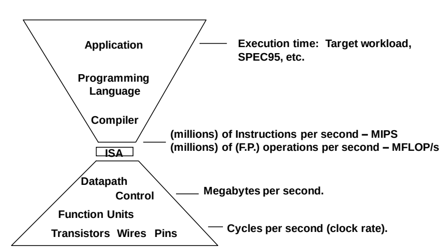
- Using the same program with these changes:
  - A new compiler used: New instruction count 9,500,000
    New CPI: 3.0
  - Faster CPU implementation: New clock rate = 300 MHZ
- What is the speedup with the changes?

$$\text{Speedup} = \frac{\text{Old Execution Time}}{\text{New Execution Time}} = \frac{I_{old} \times CPI_{old} \times \text{Clock cycle}_{old}}{I_{new} \times CPI_{new} \times \text{Clock Cycle}_{new}}$$

Speedup = $(10{,}000{,}000 \times 2.5 \times 5\text{x}10^{-9})$ / $(9{,}500{,}000 \times 3 \times 3.33\text{x}10^{-9})$
= .125 / .095 = 1.32

77

Metrics of Computer Performance



Each metric has a purpose, and each can be misused.

78

Choosing Programs To Evaluate Performance

Levels of programs or benchmarks that could be used to evaluate performance:

- **Actual Target Workload: Full applications that run on the target machine.**
- **Real Full Program-based Benchmarks:**
  - Select a specific mix or suite of programs that are typical of targeted applications or workload (e.g SPEC95, SPEC CPU2000).
- **Small "Kernel" Benchmarks:**
  - Key computationally-intensive pieces extracted from real programs.
    - Examples: Matrix factorization, FFT, tree search, etc.
- **Microbenchmarks:**
  - Small, specially written programs to isolate a specific aspect of performance characteristics: Processing: integer, floating point, local memory, input/output, etc.

Types of Benchmarks

Pros		Cons
• Representative	Actual Target Workload	• Very specific. • Non-portable. • Complex: Difficult to run, or measure.
• Portable. • Widely used. • Measurements useful in reality.	Full Application Benchmarks	• Less representative than actual workload.
• Easy to run, early in the design cycle.	Small "Kernel" Benchmarks	• Easy to "fool" by designing hardware to run them well.
• Identify peak performance and potential bottlenecks.	Microbenchmarks	• Peak performance results may be a long way from real application performance

SPEC: Standard Performance Evaluation Corporation

The most popular and industry-standard set of CPU benchmarks.

- SPECmarks, 1989:
  - 10 programs yielding a single number ("SPECmarks").
- SPEC92, 1992:
  - SPECInt92 (6 integer programs) and SPECfp92 (14 floating point programs).
- SPEC95, 1995:
  - SPECint95 (8 integer programs):
    - go, m88ksim, gcc, compress, li, ijpeg, perl, vortex
  - SPECfp95 (10 floating-point intensive programs):
    - tomcatv, swim, su2cor, hydro2d, mgrid, applu, turb3d, apsi, fppp, wave5
  - Performance relative to a Sun SuperSpark I (50 MHz) which is given a score of SPECint95 = SPECfp95 = 1

SPEC: Standard Performance Evaluation Corporation

- SPEC CPU2000, 1999:
  - CINT2000 (11 integer programs). CFP2000 (14 floating-point intensive programs)
  - Performance relative to a Sun Ultra5_10 (300 MHz) which is given a score of SPECint2000 = SPECfp2000 = 100
- SPEC CPU2006:
  - CINT2006 (12 integer programs). CFP2006 (17 floating-point intensive programs)
  - Performance relative to a Sun SPARC Enterprise M8000 which is given a score of SPECint2006 = 11.3 SPECfp2006 = 12.4

SPEC CPU2006 Programs

CINT2006 (Integer)

Benchmark	Language	Descriptions
400.Perlbench	C	Programming Language
401.bzip2	C	Compression
403.Gcc	C	C Compiler
429.mcf	C	Combinatorial Optimization
445.gobmk	C	Artificial Intelligence: Go
456.Hmmer	C	Search Gene Sequence
458.sjeng	C	Artificial Intelligence: chess
462.libquantum	C	Physics / Quantum Computing
464.h264ref	C	Video Compression
471.omnetpp	C++	Discrete Event Simulation
473.astar	C++	Path-finding Algorithms
483.xalancbmk	C++	XML Processing

Source: http://www.spec.org/osg/cpu2006/CINT2006/

SPEC CPU2006 Programs

CFP2006 (Floating Point)

Benchmark	Language	Descriptions
410.Bwaves	Fortran	Fluid Dynamics
416.Gamess	Fortran	Quantum Chemistry
433.Milc	C	Physics / Quantum Chromodynamics
434.Zeusmp	Fortran	Physics / CFD
435.Gromacs	C, Fortran	Biochemistry / Molecular Dynamics
436.cactusADM	C, Fortran	Physics / General
437.leslie3d	Fortran	Fluid Dynamics
444.Namd	C++	Biology / Molecular Dynamics
447.dealII	C++	Finite Element Analysis
450.Soplex	C++	Linear Programming, Optimization
453.Povray	C++	Image Ray-tracing
454.Calculix	C, Fortran	Structural Mechanics
459.GemsFDTD	Fortran	Computational Electromagnetics
465.Tonto	Fortran	Quantum Chemistry
470.Lbm	C	Fluid Dynamics
481.Wrf	C, Fortran	Weather
482.sphinx3	C	Speech

Source: http://www.spec.org/osg/cpu2006/CFP2006/

Top 20 SPEC CPU2006 Results (As of August 2007)

Top 20 SPECint2006 / Top 20 SPECfp2006

#	MHz	Processor	int peak	int base	MHz	Processor	fp peak	fp base
1	3000	Core 2 Duo E6850	22.6	20.2	4700	POWER6	22.4	17.8
2	4700	POWER6	21.6	17.8	3000	Core 2 Duo E6850	19.3	18.7
3	3000	Xeon 5160	21.0	17.9	1600	Dual-Core Itanium 2 9050	18.1	17.3
4	3000	Xeon X5365	20.8	18.9	1600	Dual-Core Itanium 2 9040	17.8	17.0
5	2666	Core 2 Duo E6750	20.5	18.3	2666	Core 2 Duo E6750	17.7	17.1
6	2667	Core 2 Duo E6700	20.0	17.9	3000	Xeon 5160	17.7	17.1
7	2667	Core 2 Quad Q6700	19.7	17.6	3000	Opteron 2222	17.4	16.0
8	2666	Xeon X5355	19.1	17.3	2667	Core 2 Duo E6700	16.9	16.3
9	2666	Xeon 5150	19.1	17.3	2800	Opteron 2220	16.7	13.3
10	2666	Xeon X5355	18.9	17.2	3000	Xeon 5160	16.6	16.1
11	2667	Xeon X5355	18.6	16.8	2667	Xeon X5355	16.6	16.1
12	2933	Core 2	18.5	17.8	2667	Core 2 Quad Q6700	16.6	16.1
13	2400	Core 2 Quad Q6600	18.5	16.5	2666	Xeon X5355	16.6	16.1
14	2600	Core 2 Duo X7800	18.3	16.4	2933	Core 2 Extreme X6800	16.2	16.0
15	2667	Xeon 5150	17.6	16.6	2400	Core 2 Quad Q6600	16.0	15.4
16	2400	Core 2 Duo T7700	17.6	16.6	1400	Dual-Core Itanium 2 9020	15.9	15.2
17	2333	Xeon E5345	17.5	15.9	2667	Xeon 5150	15.9	15.5
18	2333	Xeon 5148	17.4	15.9	2333	Xeon E5345	15.4	14.9
19	2333	Xeon 5140	17.4	15.7	2600	Opteron 2218	15.4	12.5
20	2660	Xeon X5355	17.4	15.7	2400	Xeon X3220	15.3	15.1

Source: http://www.spec.org/cpu2006/results/cint2006.html

85

Top SPEC CPU2006 Results (CINT2006)

Hardware Vendor	System	Result	Base	# Cores	Published
Sun Microsystems	Sun Fire X2250 (Intel Xeon X5272 3.4GHz)	28.5	23.6	4	Sep-08
SGI	SGI Altix XE 250 (Intel Xeon X5272 3.4GHz)	28.4	23.8	4	Jun-08
Hewlett-Packard Company	ProLiant DL160 G5 (3.4 GHz, Intel Xeon X5272)	28.4	23.6	4	Jul-08
Dell Inc.	PowerEdge M600 (Intel Xeon X5460, 3.16 GHz)	27.9	24.3	8	Aug-08
Dell Inc.	PowerEdge 2900 III (Intel Xeon X5460, 3.16 GHz)	27.7	24.2	8	Aug-08

Top SPEC CPU2006 Results (Cfp2006)

Hardware Vendor	System	# Cores	Result	Base	Published
Fujitsu Limited	Fujitsu SPARC Enterprise M8000	64	28.8	25	Aug-08
Sun Microsystems	Sun SPARC Enterprise M8000	64	28.8	25	Aug-08
Hewlett-Packard Company	ProLiant DL160 G5 (3.4 GHz, Intel Xeon X5272)	4	25.3	21.8	Jul-08
SGI	SGI Altix XE 250 (Intel Xeon X5272 3.4GHz)	4	25.3	21.9	Jun-08
Sun Microsystems	Sun Fire X2250 (Intel Xeon X5272 3.4GHz)	4	25.1	21.4	Sep-08
IBM Corporation	IBM Power 595 (5.0 GHz, 1 core)	1	24.9	20.1	Apr-08

LINPACK: Benchmark for Supercomputers

LINPACK solves a dense system of linear equations – "This performance does not reflect the *overall performance* of a given system, as no single number ever can. It does, however, reflect the *performance of a dedicated system for solving a dense system of linear equations.*" – http://www.top500.org/project/linpack

Rank	Computer
1	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM
2	K computer, SPA... VIIIfx 2.0GHz, Tofu interconnect Fujitsu
3	**Mira** - BlueGene/Q... 16C 1.60GHz, Custom IBM
4	**SuperM...** FDR IBM
5	**Tianhe-**
6	**Jaguar** NVIDIA
7	**Fermi** -
8	**JuQUE**
9	**Curie t** Bull
10	**Nebulae** - Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050 Dawning

List	Rank	Total Cores	Rmax (TFlops)	Rpeak (TFlops)	Power (kW)
06/2012	1	1572864	16324.8	20132.7	7890.00
11/2011	17	65536	690.2	838.9	340.50

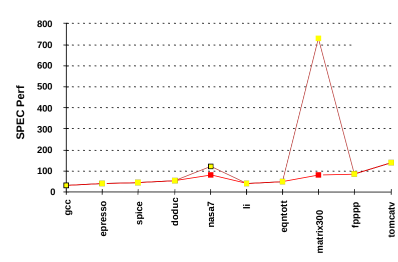
88

Performance Evaluation Using Benchmarks

- "For better or worse, benchmarks shape a field"
- Good products created when we have:
  - Good benchmarks
  - Good ways to summarize performance
- Given sales depend in big part on performance relative to competition, there is big investment in improving products as reported by performance summary
- If benchmarks inadequate, then choose between improving product for real programs vs. improving product to get more sales;
  Sales almost always wins!

89

How to Summarize Performance



90

Comparing and Summarizing Performance

	Computer A	Computer B	Computer C
P1(secs)	1	10	20
P2(secs)	1,000	100	20
Total time(secs)	1,001	110	40

For program P1, A is 10 times faster than B,
For program P2, B is 10 times faster than A,
and so on...

The relative performance of computers is unclear with Total Execution Times

91

Summary Measure

Arithmetic Mean $\frac{1}{n} \sum_{i=1}^{n} \text{Execution Time}_i$

Good, if programs are run equally in the workload

92

Arithmetic Mean

- The arithmetic mean can be misleading if the data are skewed or scattered.
  - Consider the execution times given in the table below. The performance differences are hidden by the simple average.

Program	System A Execution Time	System B Execution Time	System C Execution Time
v	50	100	500
w	200	400	600
x	250	500	500
y	400	800	800
z	5000	4100	3500
Average	1180	1180	1180

93

Unequal Job Mix

Relative Performance

- Weighted Execution Time
  - Weighted Arithmetic Mean $\sum_{i=1}^{n} \text{Weight}_i \times \text{Execution Time}_i$

- Normalized Execution Time *to a reference machine*
  - Arithmetic Mean
  - Geometric Mean

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution Time Ratio}_i}$$

Normalized to the reference machine

94

Weighted Arithmetic Mean

$\text{WAM(i)} = \sum_{j=1}^{n} \text{W(i)}_j \times \text{Time}_j$

	A	B	C	W(1)	W(2)	W(3)
P1 (secs)	1.00	10.00	20.00	0.50	0.909	0.999
P2(secs)	1,000.00	100.00	20.00	0.50	0.091	0.001
WAM(1)	500.50	55.00	20.00			
WAM(2)	91.91	18.19	20.00			
WAM(3)	2.00	10.09	20.00			

1.0 x 0.5 + 1,000 x 0.5

95

Normalized Execution Time

$$\text{Geometric Mean} = \sqrt[n]{\prod_{I=1}^{n} \text{Execution time ratio}_i}$$

	A	B	C
P1	1.00	10.00	20.00
P2	1,000.00	100.00	20.00

	Normalized to A			Normalized to B			Normalized to C		
	A	B	C	A	B	C	A	B	C
P1	1.0	10.0	20.0	0.1	1.0	2.0	0.05	0.5	1.0
P2	1.0	0.1	0.02	10.0	1.0	0.2	50.0	5.0	1.0
Arithmetic mean	1.0	5.05	10.01	5.05	1.0	1.1	25.03	2.75	1.0
Geometric mean	1.0	1.0	0.63	1.0	1.0	0.63	1.58	1.58	1.0

96

Disadvantages of Arithmetic Mean

Performance varies depending on the reference machine

	Normalized to A			Normalized to B			Normalized to C		
	A	B	C	A	B	C	A	B	C
P1	1.0	10.0	20.0	0.1	1.0	2.0	0.05	0.5	1.0
P2	1.0	0.1	0.02	10.0	1.0	0.2	50.0	5.0	1.0
Arithmetic mean	1.0	5.05	10.01	5.05	1.0	1.1	25.03	2.75	1.0

B is 5 times slower than A

A is 5 times slower than B

C is slowest

C is fastest

97

The Pros and Cons Of Geometric Means

- Independent of running times of the individual programs
- Independent of the reference machines
- Do not predict execution time
  - the performance of A and B is the same : only true when P1 ran 100 times for every occurrence of P2

	Computer A	Computer B	Computer C
P1(secs)	1	10	20
P2(secs)	1,000	100	20
Total time(secs)	1,001	110	40

1(P1) x 100 + 1000(P2) x 1
= 10(P1) x 100 + 100(P2) x 1

	Normalized to A			Normalized to B			Normalized to C		
	A	B	C	A	B	C	A	B	C
P1	1.0	10.0	20.0	0.1	1.0	2.0	0.05	0.5	1.0
P2	1.0	0.1	0.02	10.0	1.0	0.2	50.0	5.0	1.0
Geometric mean	1.0	1.0	0.63	1.0	1.0	0.63	1.58	1.58	1.0

98

Geometric Mean

- The real usefulness of the normalized geometric mean is that no matter which system is used as a reference, the ratio of the geometric means is consistent.
- This is to say that the ratio of the geometric means for System A to System B, System B to System C, and System A to System C is the same no matter which machine is the reference machine.

99

Geometric Mean

- The results that we got when using System B and System C as reference machines are given below.
- We find that 1.6733/1 = 2.4258/1.4497.

System A Execution Time	Execution Time Normalized to B	System B Execution Time	Execution Time Normalized to B	System C Execution Time	Execution Time Normalized to B
Geometric Mean	1.6733		1		0.6898

System A Execution Time	Execution Time Normalized to C	System B Execution Time	Execution Time Normalized to C	System C Execution Time	Execution Time Normalized to C
Geometric Mean	2.4258		1.4497		1

100

Geometric Mean

- The inherent problem with using the geometric mean to demonstrate machine performance is that all execution times contribute equally to the result.
- So shortening the execution time of a small program by 10% has the same effect as shortening the execution time of a large program by 10%.
  - Shorter programs are generally easier to optimize, but in the real world, we want to shorten the execution time of longer programs.
- Also, if the geometric mean is not proportionate, a system giving a geometric mean 50% smaller than another is not necessarily twice as fast!

101

Measure Computer Performance: MIPS (Million Instructions Per Second)

- For a specific program running on a specific computer, MIPS is a measure of millions of instructions executed per second:

MIPS = Instruction count / (Execution Time x 10^6)
= Instruction count / (CPU clocks x Cycle time x 10^6)
= (Instruction count x Clock rate) / (Instruction count x CPI x 10^6)
= Clock rate / (CPI x 10^6)

- Shorter execution time usually means faster MIPS rating.

102

Computer Performance Measures : MIPS (Million Instructions Per Second)

- **M**eaningless **I**ndicator of **P**rocessor **S**peed
- Problems:
  - No account for instruction set used.
    - Cannot be used to compare computers with different instruction sets.
  - Program-dependent: A single machine does not have a single MIPS rating.
  - A higher MIPS rating in some cases may not mean higher performance or better execution time. i.e. due to compiler design variations.

Compiler Variations, MIPS, Performance: An Example

- For the machine with instruction classes:

Instruction class	CPI
A	1
B	2
C	3

- For a given program two compilers produced the following instruction counts:

	Instruction counts (in millions) for each instruction class		
Code from:	A	B	C
Compiler 1	5	1	1
Compiler 2	10	1	1

- The machine is assumed to run at a clock rate of 100 MHz

Compiler Variations, MIPS, Performance: An Example (Continued)

MIPS = Clock rate / (CPI x 10^6) = 100 MHz / (CPI x 10^6)

CPI = CPU execution cycles / Instructions count

$$CPU\ clock\ cycles = \sum_{i=1}^{n} (CPI_i \times C_i)$$

CPU time = Instruction count x CPI / Clock rate

- For compiler 1:
  - CPI_1 = (5 x 1 + 1 x 2 + 1 x 3) / (5 + 1 + 1) = 10 / 7 = 1.43
  - MIP_1 = 100 / (1.428 x 10^6) = 70.0
  - CPU $time_1$ = ((5 + 1 + 1) x 10^6 x 1.43) / (100 x 10^6) = 0.10 seconds
- For compiler 2:
  - CPI_2 = (10 x 1 + 1 x 2 + 1 x 3) / (10 + 1 + 1) = 15 / 12 = 1.25
  - MIP_2 = 100 / (1.25 x 10^6) = 80.0
  - CPU $time_2$ = ((10 + 1 + 1) x 10^6 x 1.25) / (100 x 10^6) = 0.15 seconds

Computer Performance Measures : MFOLPS (Million FLOating-Point Operations Per Second)

- A floating-point operation is an addition, subtraction, multiplication, or division operation applied to numbers represented by a single or double precision floating-point representation.
- MFLOPS, for a specific program running on a specific computer, is a measure of millions of floating point-operation (megaflops) per second:

MFLOPS = Number of floating-point operations / (Execution time x 10^6)

Computer Performance Measures : MFOLPS (Million FLOating-Point Operations Per Second)

- A better comparison measure between different machines than MIPS.
- Program-dependent: Different programs have different percentages of floating-point operations present. i.e compilers have no such operations and yield a MFLOPS rating of zero.
- Dependent on the type of floating-point operations present in the program.

Quantitative Principles of Computer Design

Amdahl's Law:

By Gene Amdahl, architect of IBM/360

The performance gain from improving some portion of a computer is calculated by:

$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement}}{\text{Performance for the entire task without using the enhancement}}$$

$$\text{or Speedup} = \frac{\text{Execution time without the enhancement}}{\text{Execution time for entire task using the enhancement}}$$

Performance Enhancement Calculations: Amdahl's Law

- The performance enhancement possible due to a given design improvement is limited by the amount that the improved feature is used
- Amdahl's Law: Suppose that enhancement E accelerates a fraction F of the execution time by a factor S and the remainder of the time is unaffected then:

$$\text{Speedup(E)} = \frac{1}{(1 - F) + F/S}$$

Performance Enhancement Calculations: Amdahl's Law

Performance improvement or speedup due to enhancement E:

$$\text{Speedup(E)} = \frac{\text{Execution Time without E}}{\text{Execution Time with E}} = \frac{\text{Performance with E}}{\text{Performance without E}}$$

E accelerates a fraction F of the execution time by a factor S, then:

Execution Time with E = ((1-F) + F/S) X Execution Time without E

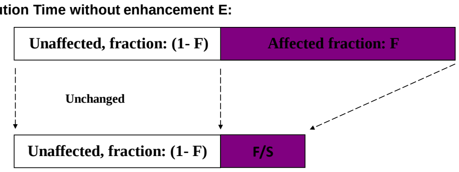
Hence speedup is given by:

$$\text{Speedup(E)} = \frac{\text{Execution Time without E}}{((1 - F) + F/S) \times \text{Execution Time without E}} = \frac{1}{(1 - F) + F/S}$$

Pictorial Depiction of Amdahl's Law

Enhancement E accelerates fraction F of execution time by a factor of S

Before:
Execution Time without enhancement E:



After:
Execution Time with enhancement E:

$$\text{Speedup(E)} = \frac{\text{Execution Time without enhancement E}}{\text{Execution Time with enhancement E}} = \frac{1}{(1 - F) + F/S}$$

Performance Enhancement Example

- For the RISC machine with the following instruction mix:

Op	Freq	Cycles	CPI(i)	% Time
ALU	50%	1	.5	23%
Load	20%	5	1.0	45%
Store	10%	3	.3	14%
Branch	20%	2	.4	18%

CPI = 2.2

Performance Enhancement Example

- If a CPU design enhancement improves the CPI of load instructions from 5 to 2, what is the resulting performance improvement from this enhancement:

Fraction enhanced = F = 45% or .45

Unaffected fraction = 100% - 45% = 55% or .55

Factor of enhancement = 5/2 = 2.5

Using Amdahl's Law:

$$\text{Speedup(E)} = \frac{1}{(1 - F) + F/S} = \frac{1}{.55 + .45/2.5} = 1.37$$

An Alternative Solution Using CPU Equation

- If a CPU design enhancement improves the CPI of load instructions from 5 to 2, what is the resulting performance improvement from this enhancement:

Old CPI = 2.2

New CPI = .5 x 1 + .2 x 2 + .1 x 3 + .2 x 2 = 1.6

$$\text{Speedup(E)} = \frac{\text{Original Execution Time}}{\text{New Execution Time}} = \frac{\cancel{\text{Instruction count}} \times \text{old CPI} \times \cancel{\text{clock cycle}}}{\cancel{\text{Instruction count}} \times \text{new CPI} \times \cancel{\text{clock cycle}}}$$

$$= \frac{\text{old CPI}}{\text{new CPI}} = \frac{2.2}{1.6} = 1.37$$

Which is the same speedup obtained from Amdahl's Law in the first solution.

Performance Enhancement Example

- A program runs in 100 seconds on a machine with multiply operations responsible for 80 seconds of this time. By how much must the speed of multiplication be improved to make the program four times faster?

$$\text{Desired speedup} = 4 = \frac{100}{\text{Execution Time with enhancement}}$$

→ Execution time with enhancement = 25 seconds

25 seconds = (100 - 80 seconds) + 80 seconds / n
25 seconds = 20 seconds + 80 seconds / n

→ 5 = 80 seconds / n

→ n = 80/5 = 16

Hence multiplication should be 16 times faster to get a speedup of 4.

115

Performance Enhancement Example

- For the previous example with a program running in 100 seconds on a machine with multiply operations responsible for 80 seconds of this time. By how much must the speed of multiplication be improved to make the program five times faster?

$$\text{Desired speedup} = 5 = \frac{100}{\text{Execution Time with enhancement}}$$

→ Execution time with enhancement = 20 seconds

20 seconds = (100 - 80 seconds) + 80 seconds / n
20 seconds = 20 seconds + 80 seconds / n

→ 0 = 80 seconds / n

No amount of multiplication speed improvement can achieve this.

116

Another Amdahl's Law Example

- New CPU 10X faster
- I/O-bound server, so 60% time waiting for I/O

$$\text{Speedup}_{\text{overall}} = \frac{1}{\left(1 - \text{Fraction}_{\text{enhanced}}\right) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$= \frac{1}{\left(1 - 0.4\right) + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$$

- Apparently, it's human nature to be attracted by 10X faster, vs. keeping in perspective it's just 1.6X faster