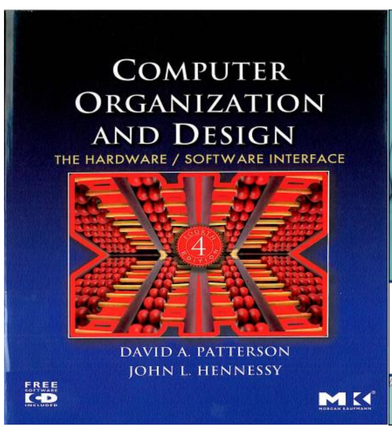
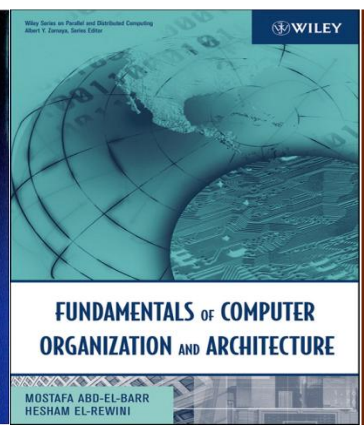
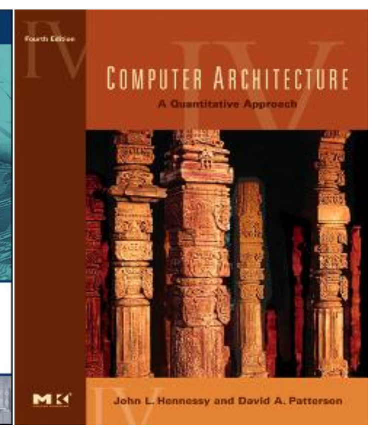
Computer Architecture

Lecture 0: Introduction





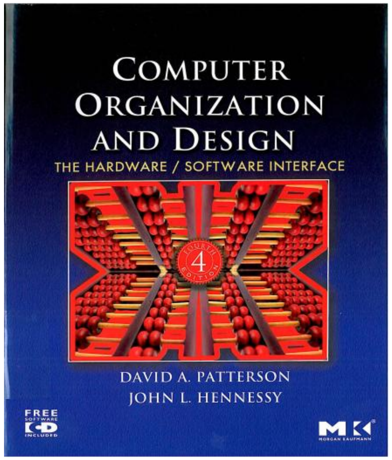


Nguyen Minh Son, Ph.D

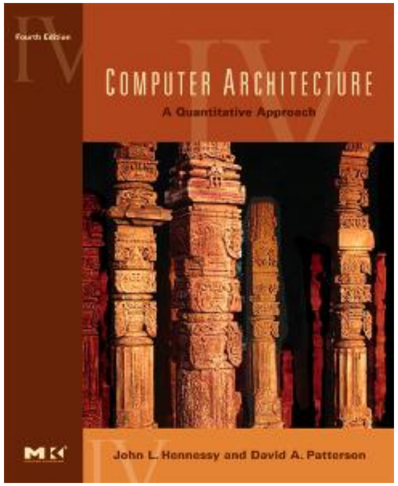
UIT-HCM

Course Information

- Instructor
  - Dr. Nguyen Minh Son (sonnm@uit.edu.vn)
- Office: Faculty of CE, Building A-327
- Prerequisite: *Computer Architecture* or the equivalent
- Meeting time: Fri1-3 periods, C-101
- Textbook
  - **David A. Patterson and John L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface* (4th edition), Morgan Kaufmann, 2008.**
- Other teaching materials
  - Some reference books available in class meetings and course web
  - Slides & Lectures
- Coordinator: …



Required



Reading references

Objectives – To Learn

Recent trends of architectural features in high-performance computer systems

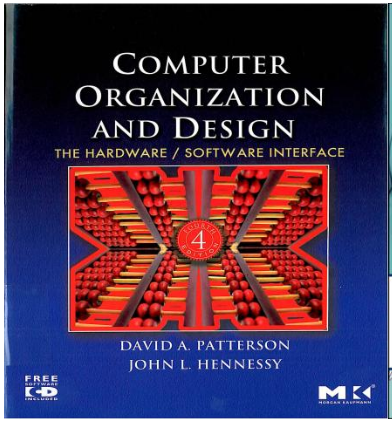
- Week 1: Computer Abstractions and Technology
- Week 2,3: Instructions - Language of the Computer
- Week 4,5: Arithmetic for Computers
- Week 6,7: Assessing and Understanding Performance
- Week 8: *Mid-Term Exam*
- Week 9,10,11: The processor – Datapath and Control
- Week 12,13,14: Pipelining - Datapath and Control
- Week 15: Presentation – Term-Projects
  - Optional: Large and Fast Exploiting Memory Hierarchy
- Week 16: *Final Exam*

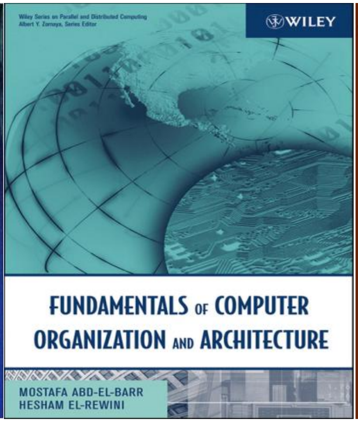
Course evaluation

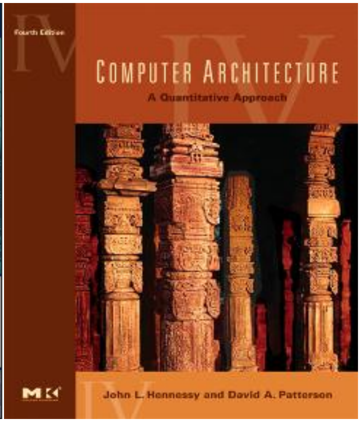
- Grading policy:
  - Quizzes (attendance): 10%
  - Midterm exam: 30% (Multiple choice and Essay)
  - Final exam: 60% (Multiple choice and Essay)
- No cheating ?!

Computer Architecture

Lecture 1: Computer Technology





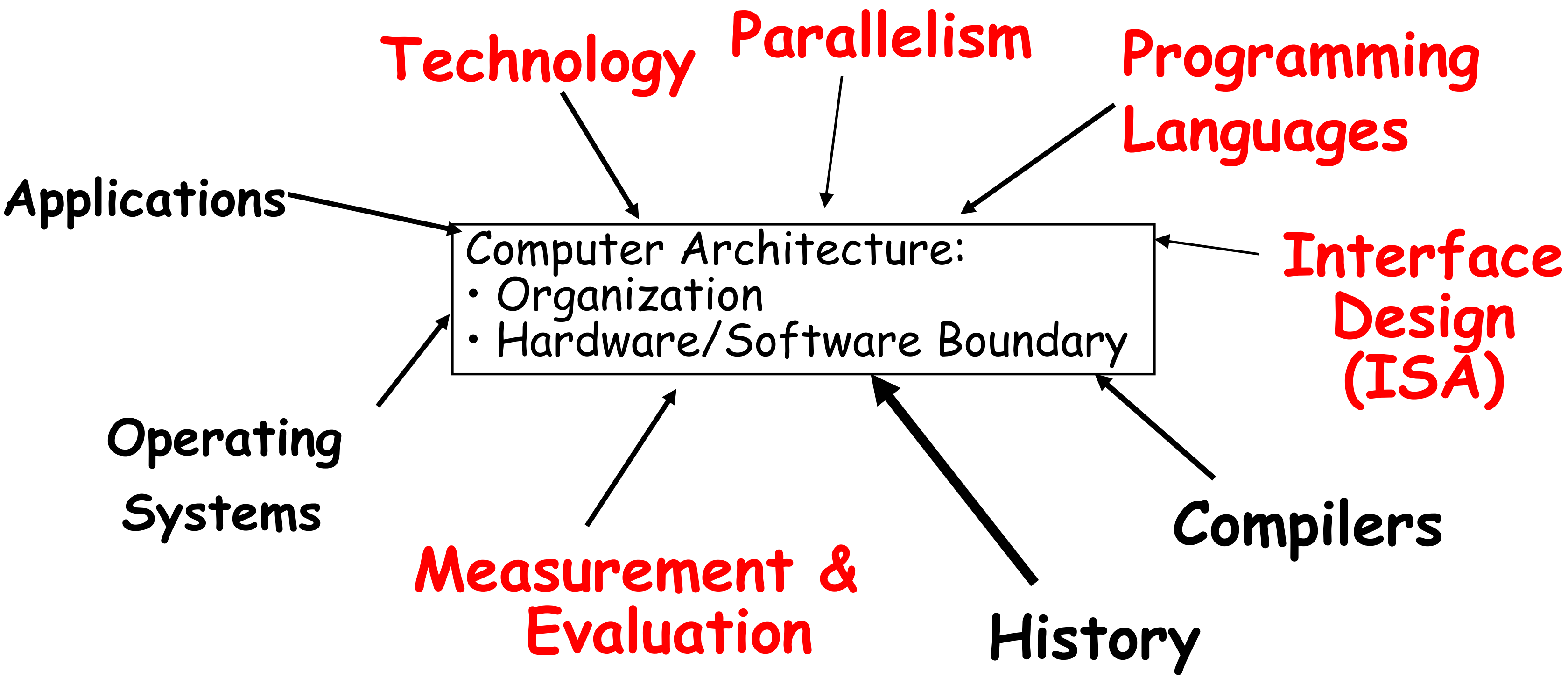


Nguyen Minh Son, Ph.D

UIT-HCM

Course Focus

- Understanding machine structures, technology factors, evaluation methods that will determine the form of computers in 21st Century



In Your Course

- Mix of lecture vs. discussion
  - Depends on how well reading is done before class
- Goal is to learn how to know a computer system
  - Learn fundamental computer organization: machanism, sequential and parallel computing.
  - Learn how to evaluate and measure the performance of computer system.

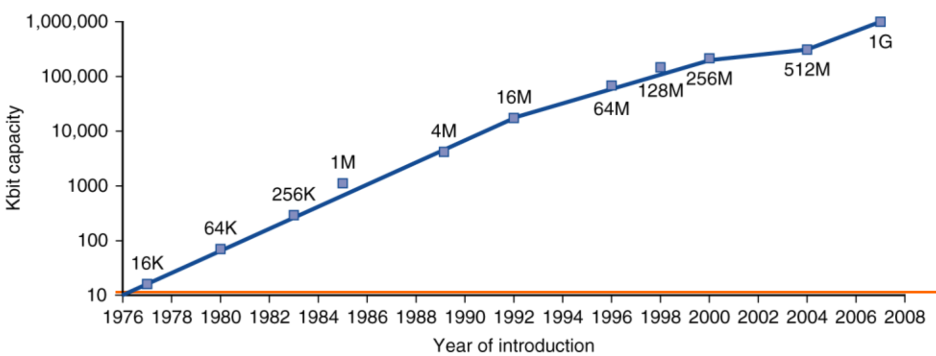
Today outline

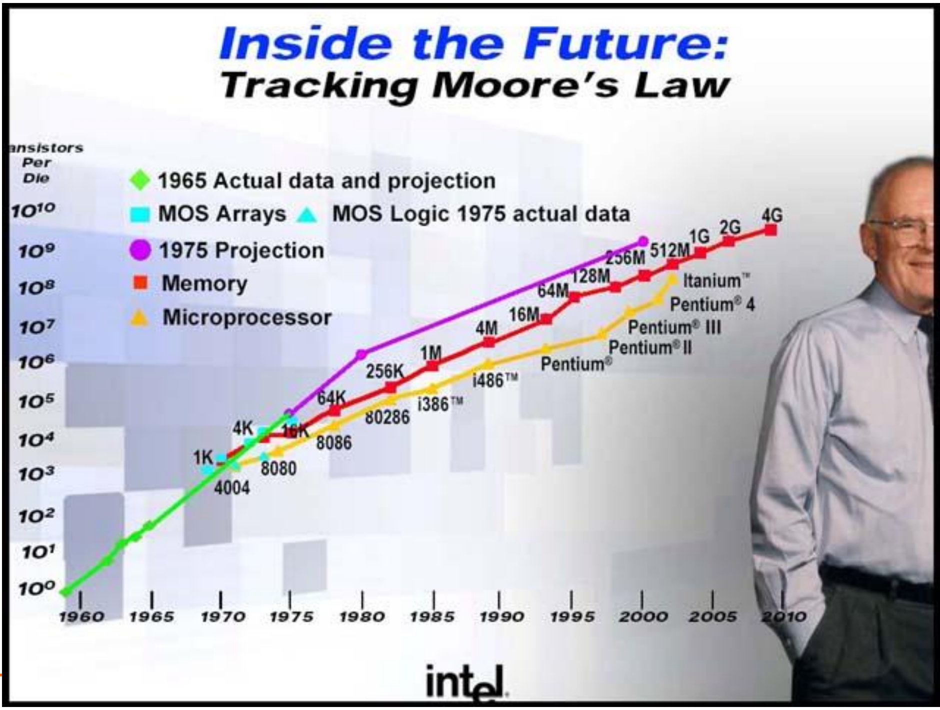
- Computer Revolution
  - CPU technology
  - Memory development
- Market generation
  - Applications
- Overview of Computer System
  - Layers
  - Application and Technology trends
  - Problem solver
- Manufacturing Ics
- Computer architecture

The Computer Revolution

- Progress in computer technology
  - Underpinned by Moore's Law
  - Reduced cost,
  - Increased performance and capacity
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - WWW, Search Engines
- Computers are pervasive
  - Internet of Things

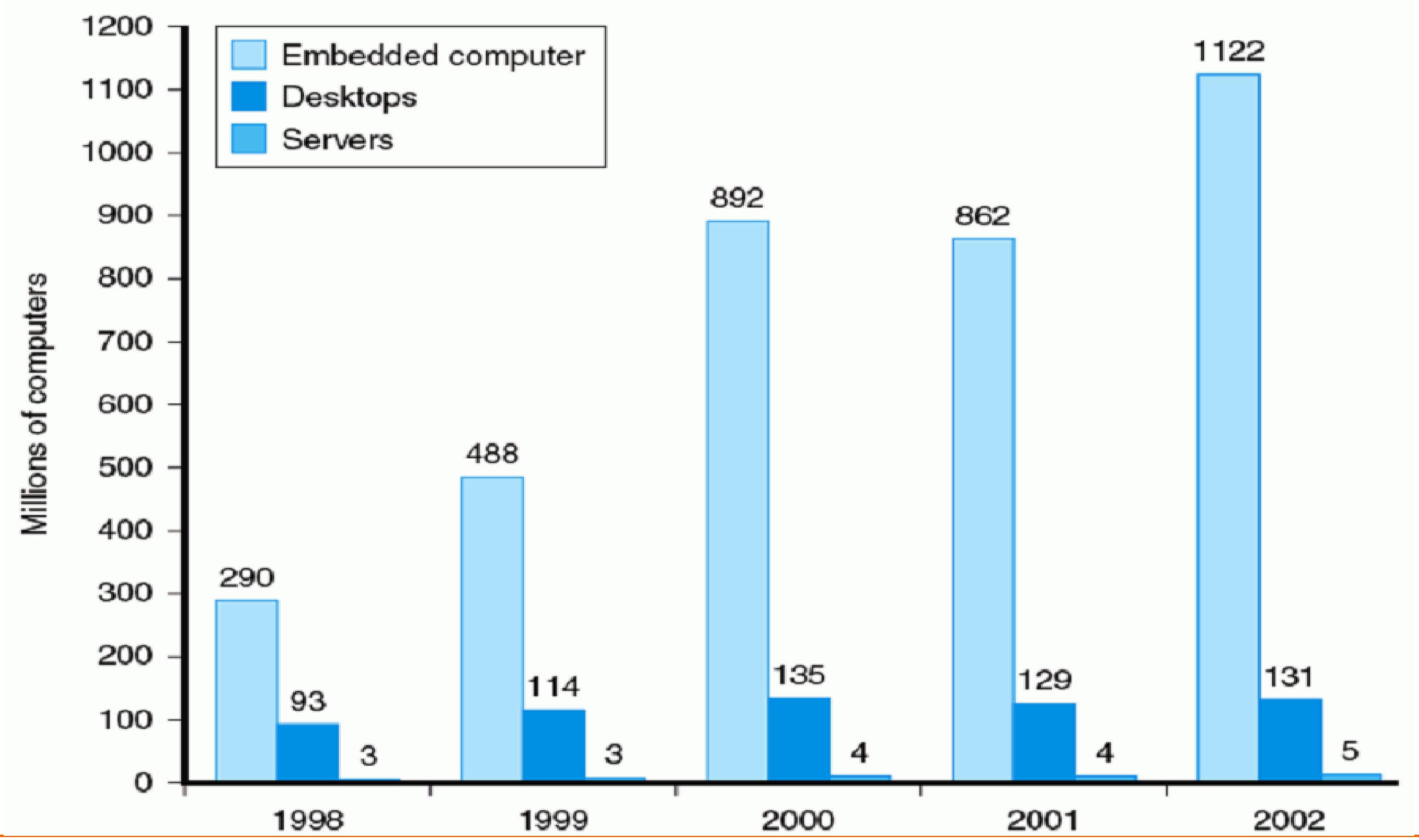
Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000



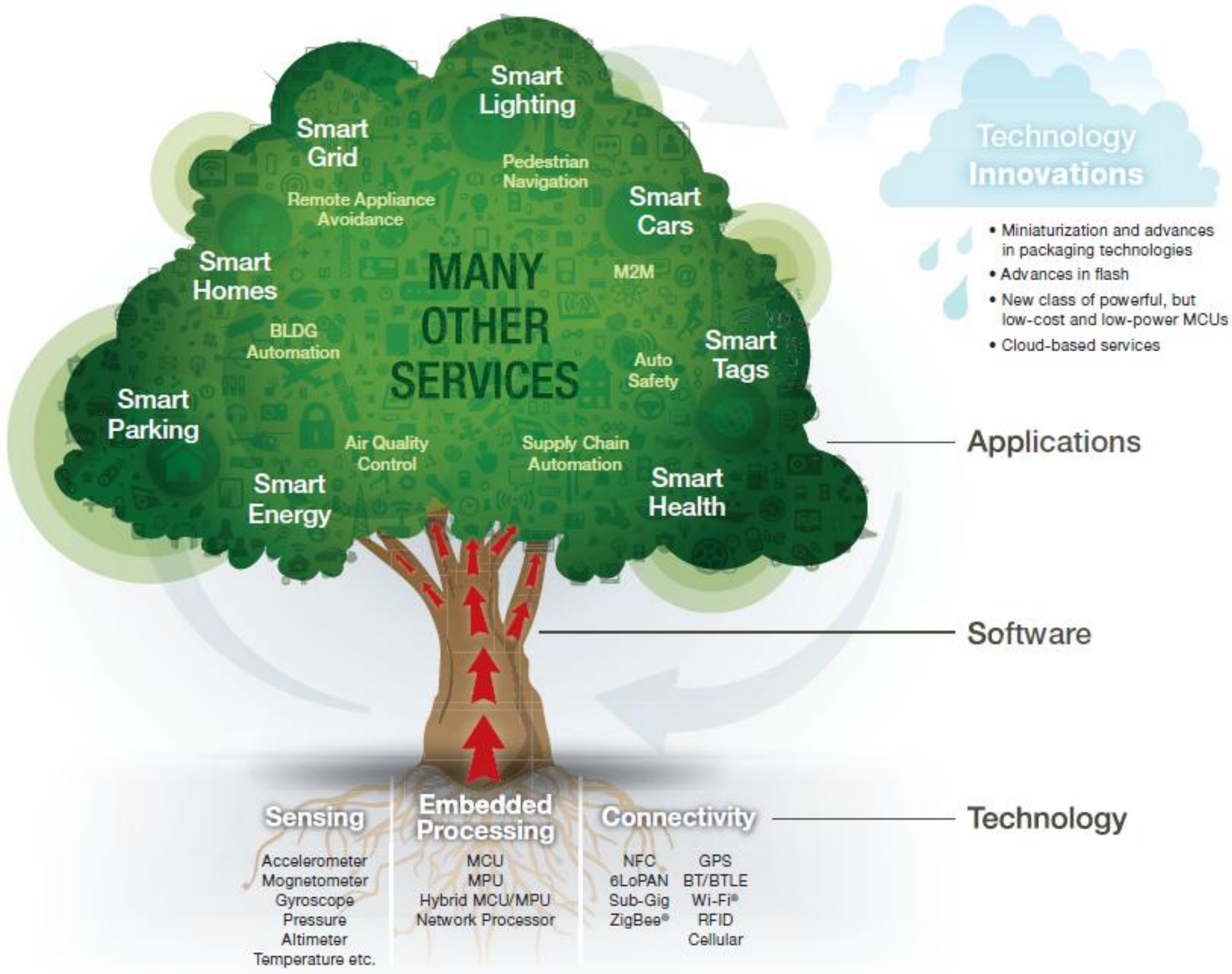


The Processor Market

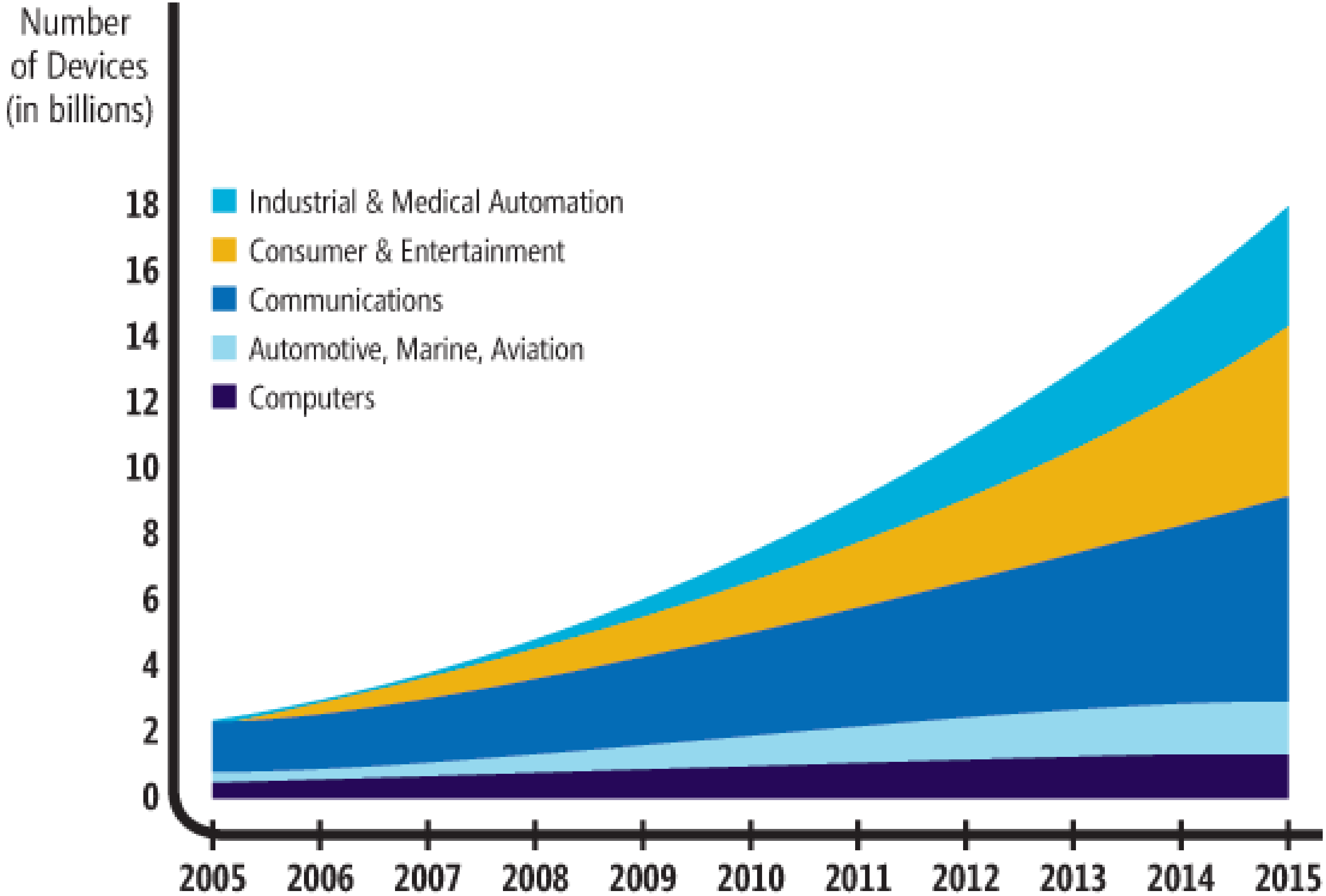
- Number of distinct processors sold between 1998 and 2002



Internet of Things

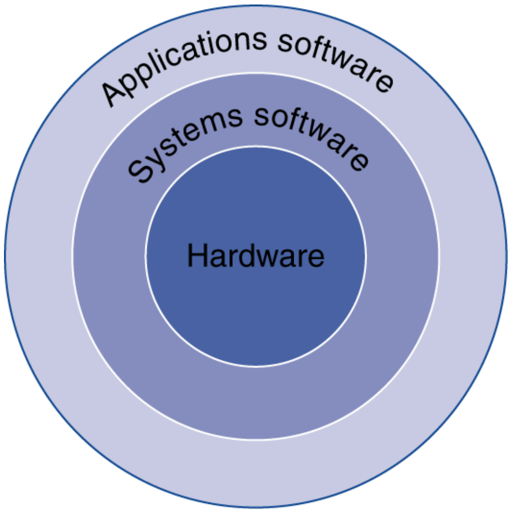


Internet of Things



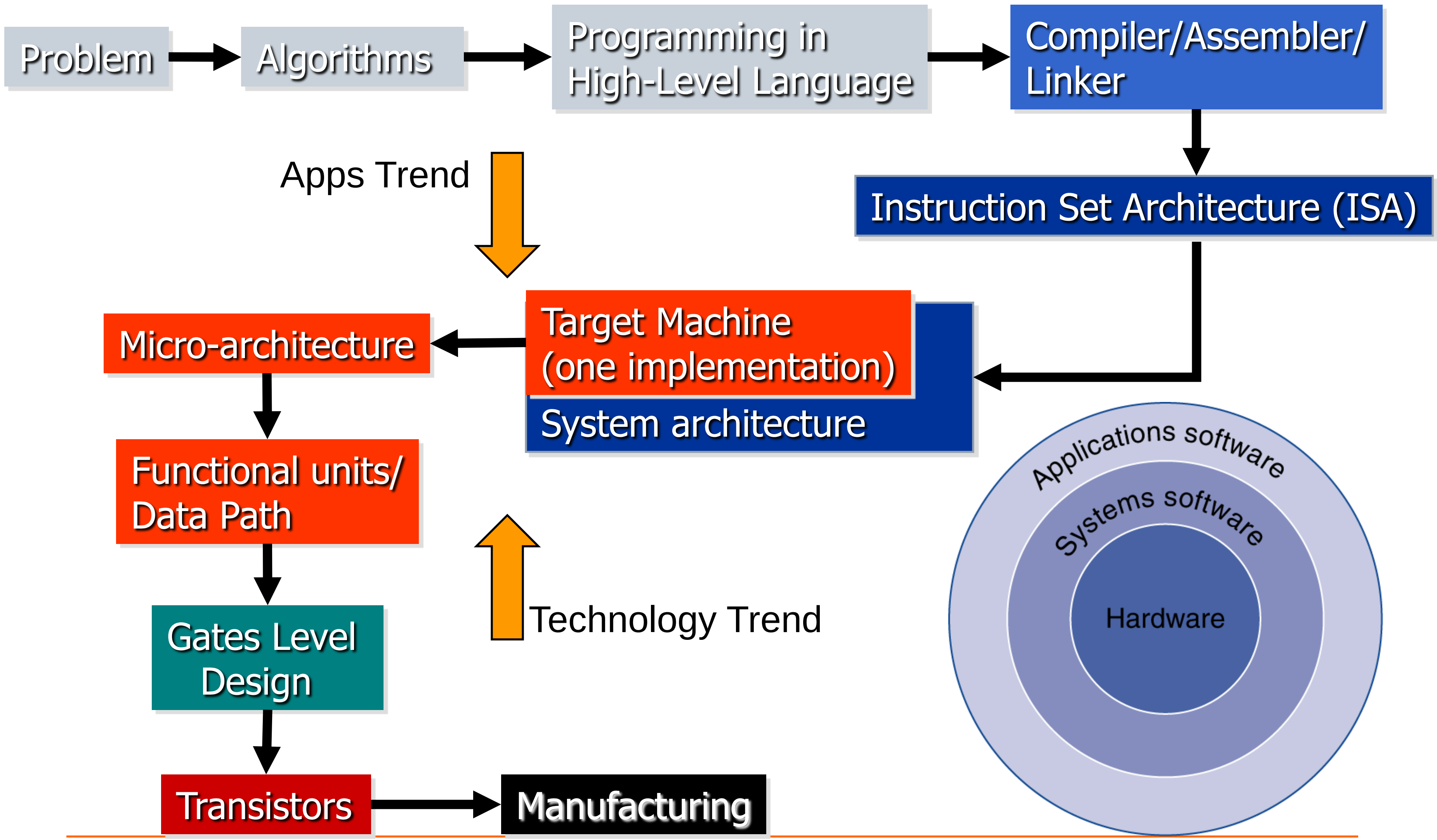
Source: John Gantz, The Embedded Internet, Methodology and Findings, IDC, January 2009

Computer System: Layers



- Application software
  - Written in high-level language
- System software
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers

Breakdown of a Computing Problem

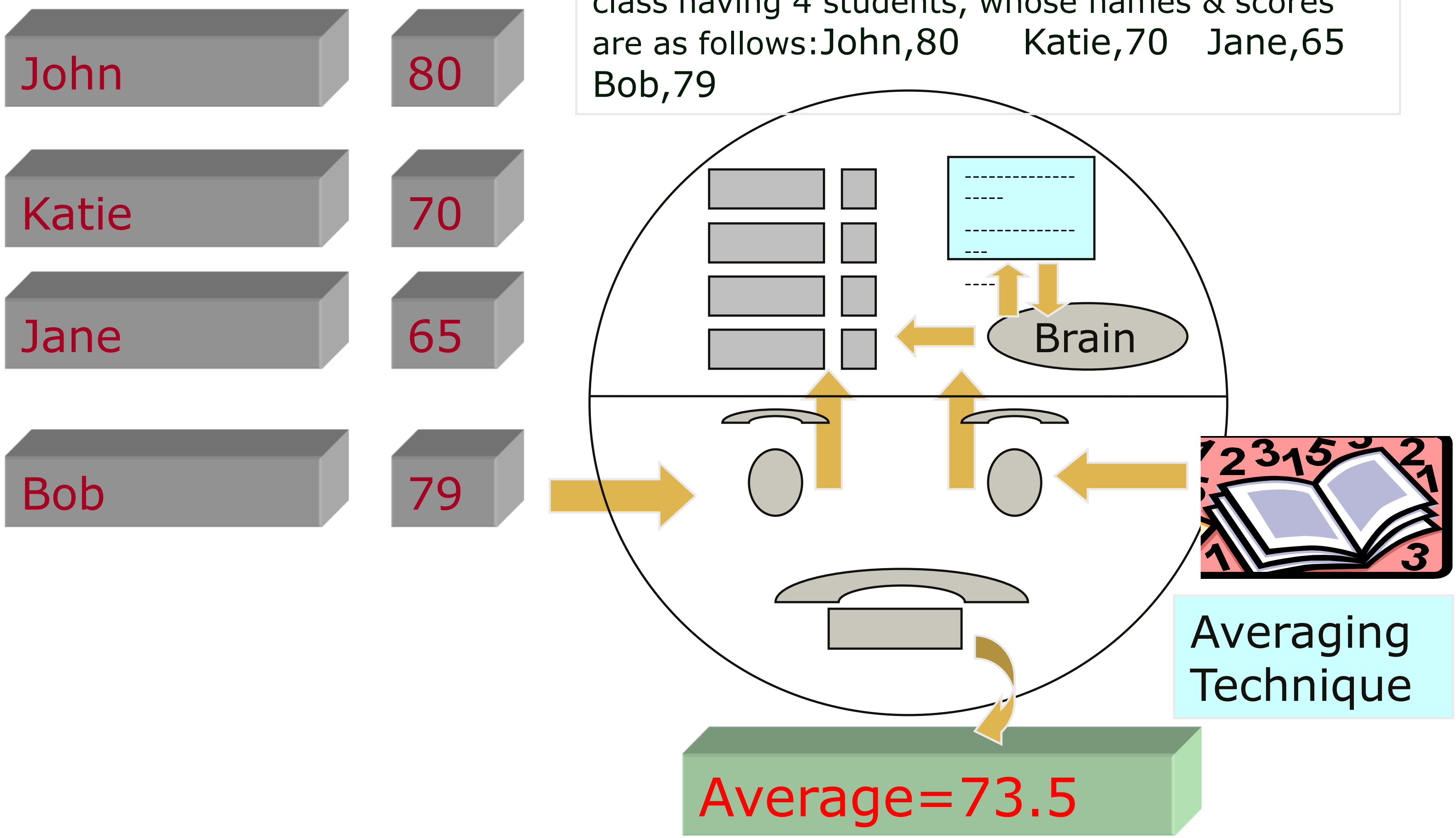


Computer as Problem Solver

- What is a **problem**?
  - A problem requires some unprocessed facts (**data**) converted into useful results (**information**)
  - For every problem, there exists a step-by-step method (**algorithm**) to do this conversion

Humans: A well known Problem Solver

What will the average score in "math exam" in a class having 4 students, whose names & scores are as follows: John,80 Katie,70 Jane,65 Bob,79



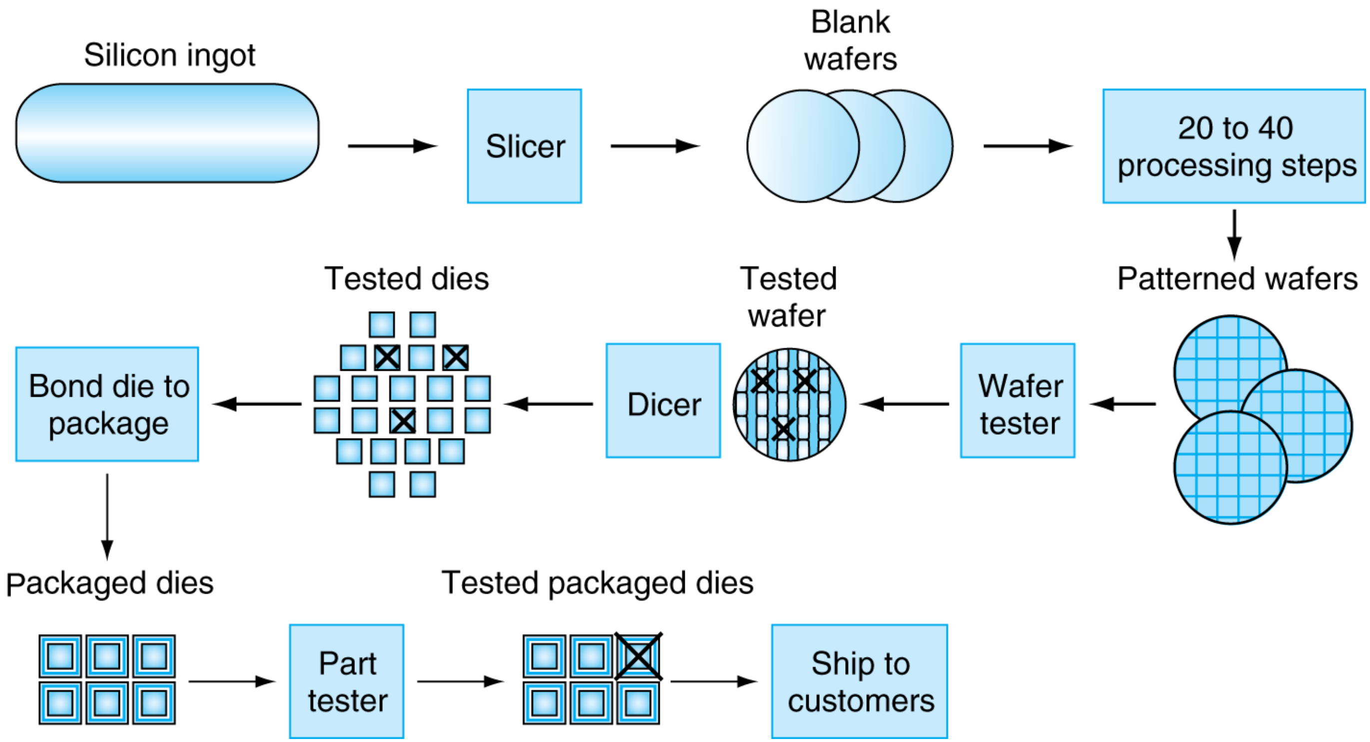
Computer: A more efficient Problem Solver

Average=73.5

John	80
Katie	70
Jane	65
Bob	79

Averaging Technique
(Computer Software)

Manufacturing ICs



- Yield: proportion of working dies per wafer

What Computer Architecture bring to Table

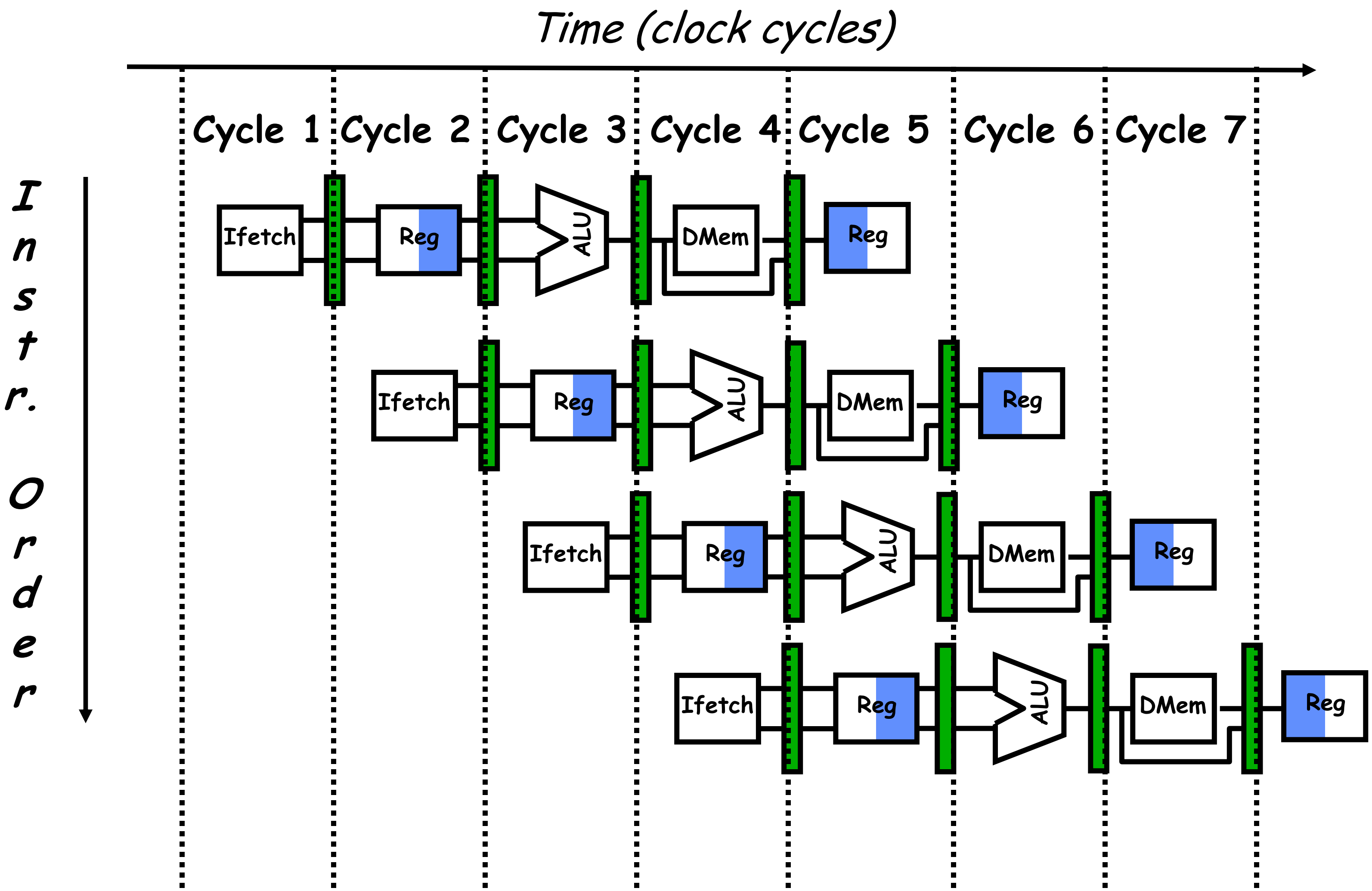
- Other fields often borrow ideas from architecture

- Quantitative Principles of Design
  1. Take Advantage of Parallelism
  2. Principle of Locality
  3. Focus on the Common Case
  4. Computer Performance: Amdahl's Law
  5. The Processor Performance Equation

1) Taking Advantage of Parallelism

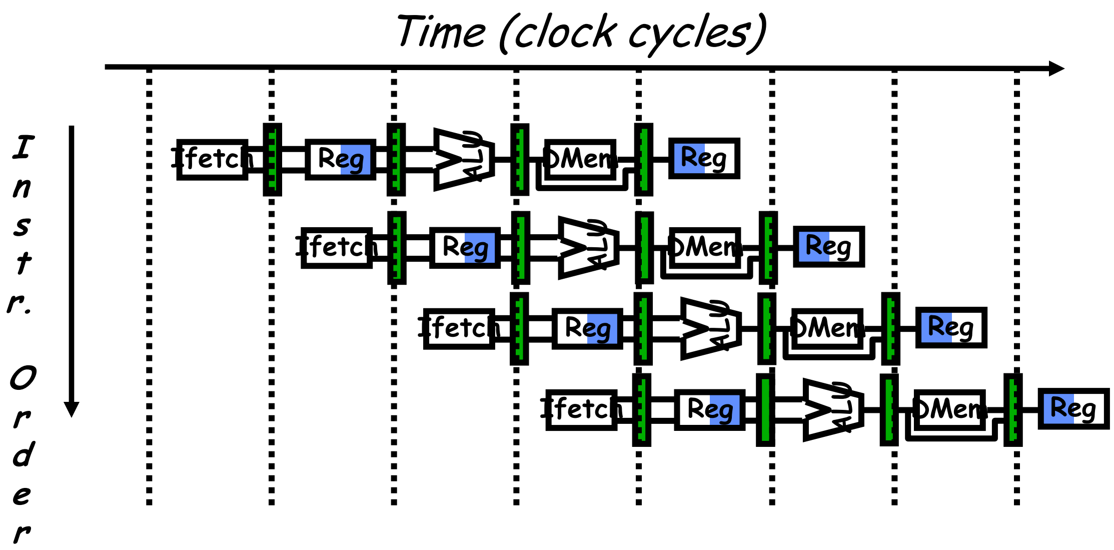
- Increasing throughput of server computer via multiple processors or multiple disks
- Detailed HW design
  - Carry lookahead adders uses parallelism to speed up computing sums from linear to logarithmic in number of bits per operand
  - Multiple memory banks searched in parallel in set-associative caches
- Pipelining: overlap instruction execution to reduce the total time to complete an instruction sequence
  - Not every instruction depends on immediate predecessor $\Rightarrow$ executing instructions completely/partially in parallel possible
  - Classic 5-stage pipeline:
    1) Instruction Fetch (Ifetch),
    2) Register Read (Reg),
    3) Execute (ALU),
    4) Data Memory Access (Dmem),
    5) Register Write (Reg)

Pipelined Instruction Execution



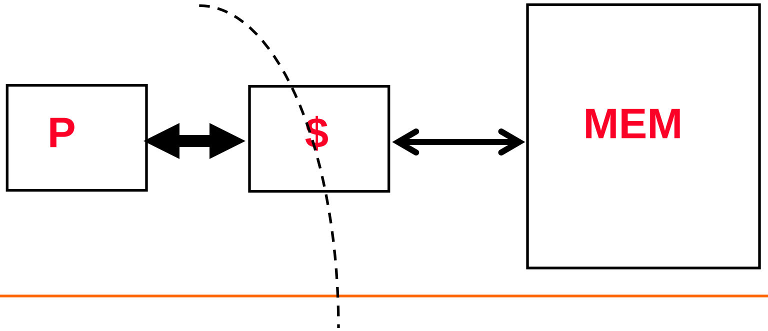
Limits to Pipelining

- **Hazards** prevent next instruction from executing during its designated clock cycle
  - Structural hazards: attempt to use the same hardware to do two different things at once
  - Data hazards: Instruction depends on result of prior instruction still in the pipeline
  - Control hazards: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps).



2) The Principle of Locality

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straight-line code, array access)
- Last 30 years, HW relied on locality for memory perf.



Level of the Memory Hierarchy

Capacity *Access Time* *Cost*			*Staging* *Xfer Unit*	
CPU Registers 100s Bytes 300 – 500 ps (0.3-0.5 ns)	**Registers**			**Upper Level**
		Instr. Operands	prog./compiler 1-8 bytes	faster ↑
L1 and L2 Cache 10s-100s K Bytes ~1 ns - ~10 ns $1000s/ GByte	**L1 Cache**			
		Blocks	cache cntl 32-64 bytes	
	L2 Cache			
		Blocks	cache cntl 64-128 bytes	
Main Memory G Bytes 80ns- 200ns ~ $100/ GByte	**Memory**			
		Pages	OS 4K-8K bytes	
Disk 10s T Bytes, 10 ms (10,000,000 ns) ~ $1 / GByte	**Disk**			
		Files	user/operator Mbytes	↓ Larger
Tape infinite sec-min ~$1 / GByte	**Tape**			**Lower Level**

3) Focus on the Common Case

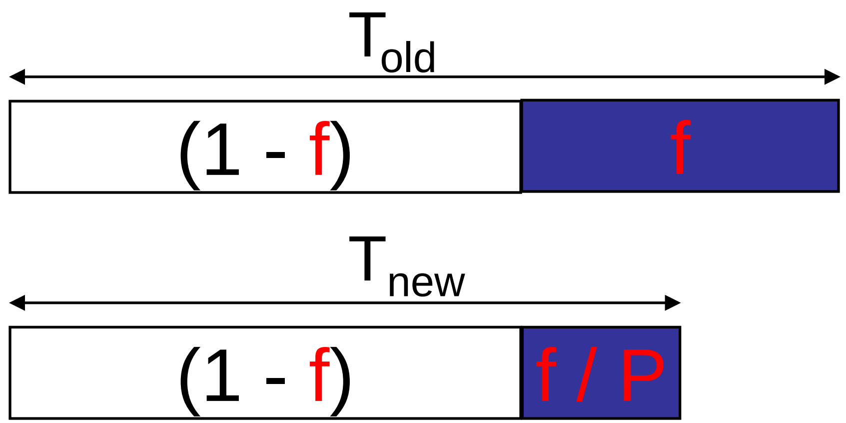
- **Common sense guides computer design**
  - Since its engineering, common sense is valuable
- **In making a design trade-off, favor the frequent case over the infrequent case**
  - E.g., Instruction fetch and decode unit used more frequently than multiplier, so optimize it 1st
  - E.g., If database server has 50 disks / processor, storage dependability dominates system dependability, so optimize it 1st
- **Frequent case is often simpler and can be done faster than the infrequent case**
  - E.g., overflow is rare when adding 2 numbers, so improve performance by optimizing more common case of no overflow
  - May slow down overflow, but overall performance improved by optimizing for the normal case
- **What is frequent case and how much performance improved by making case faster**

4) Computer Performance: Amdahl's Law

- Make the common case faster
- Speedup

  $= Perf_{new} / Perf_{old} = T_{old} / T_{new} = \dfrac{1}{(1-f)+\dfrac{f}{P}}$

- Performance improvement from using faster mode is limited by the fraction the faster mode can be applied.



5) Processor Performance Equation

CPI

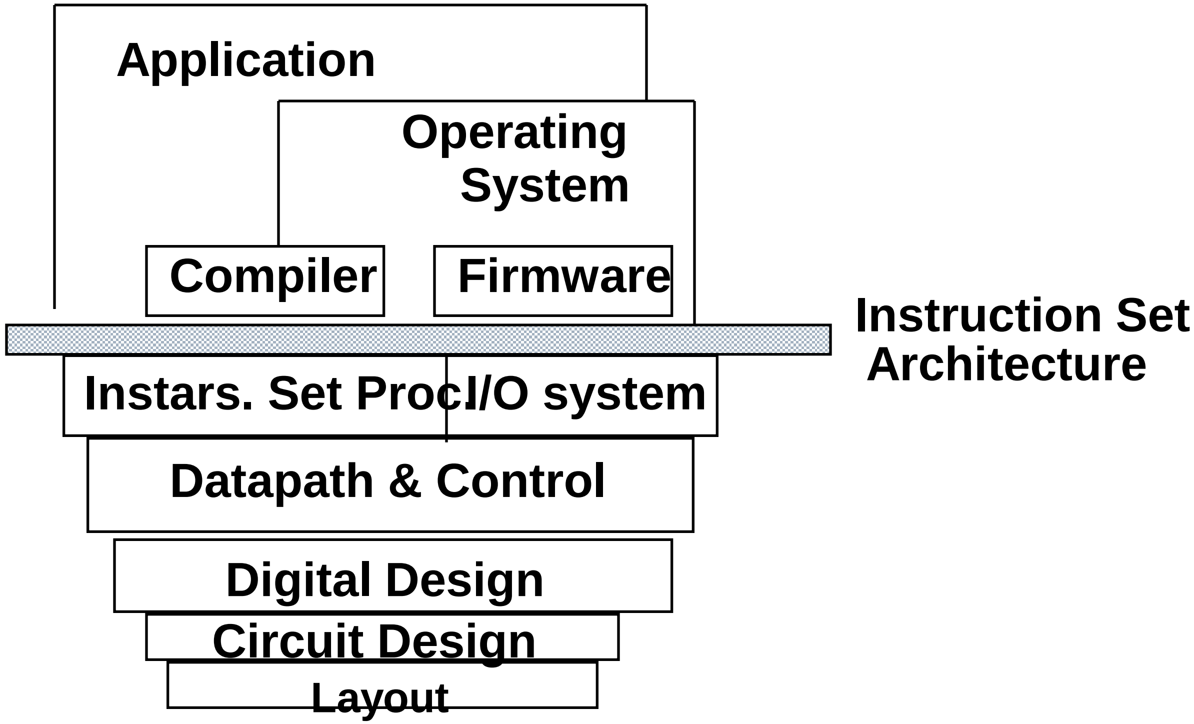
inst count Cycle time

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

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

Conclusion

- Advanced Computer Architecture =
  Computer Organization + Design & Performance



Enjoy !!!

Q&A