دانشگاه کردستان
University of Kurdistan
زانکۆی کوردستان

Department of Computer Engineering
University of Kurdistan

Computer Architecture

Introduction

By: Dr. Alireza Abdollahpouri

Course Info

Course Textbooks

- D. A. Patterson and J. L. Hennessy, Computer Organization and Design, 5th Edition: The Hardware/Software Interface, Morgan Kaufman, 5th Ed.
- M. Mano, Computer System Architecture, Prentice-Hall, 3rd Ed., 1993.

Instructor

Dr. Alireza Abdollahpouri

Email: abdollahpouri@gmail.com

Course Info

- **Grading Policy**
  - **Homework** **15%**
  - **Midterm** **35%**
  - **Final** **45%**
  - **Class participation** **5%**
- **Web Page**
  - http://prof.uok.ac.ir/abdollahpouri/ComArch.html

Course Info

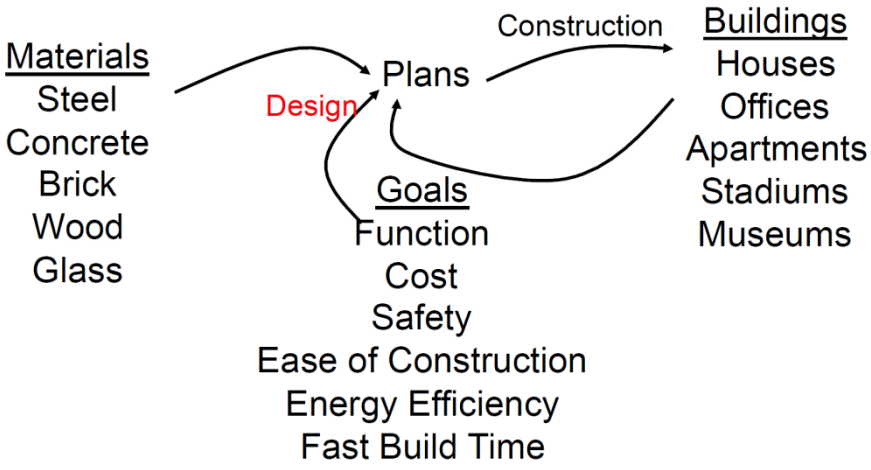
Topics covered

- Introduction, basic computer organization
- Register Transfer Language (RTL)
- Instruction Set Architecture (ISA)
- Computer Arithmetic
- MIPS ISA and assembly language
- MIPS (single cycle and multi-cycle)
- Pipelining
- Memory Systems
- I/O

What is Computer Architecture?

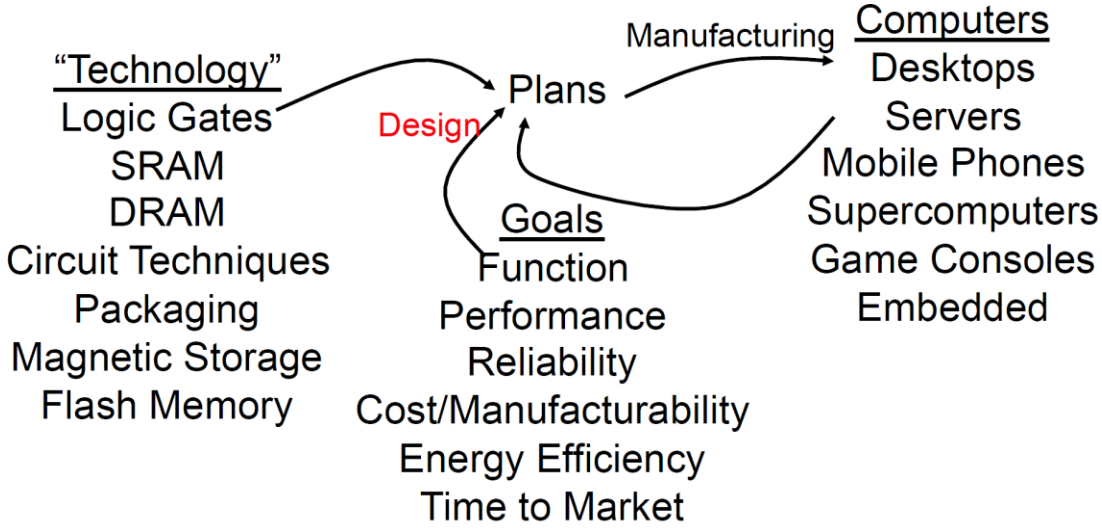
Computer Architecture is the science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.

An analogy to architecture of buildings…



The role of a building architect:

The role of a Computer architect:

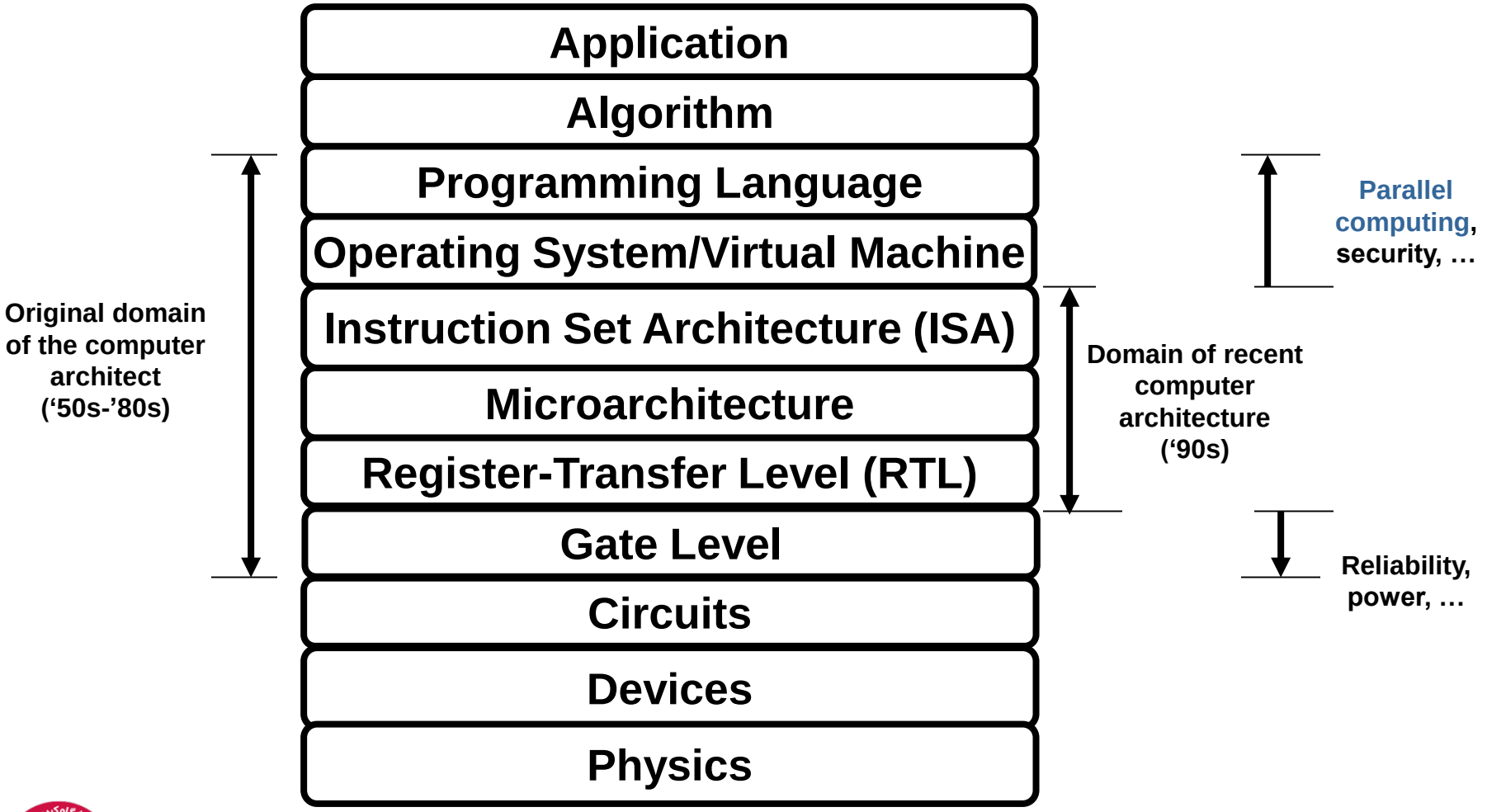


Some motivation to learn computer architecture

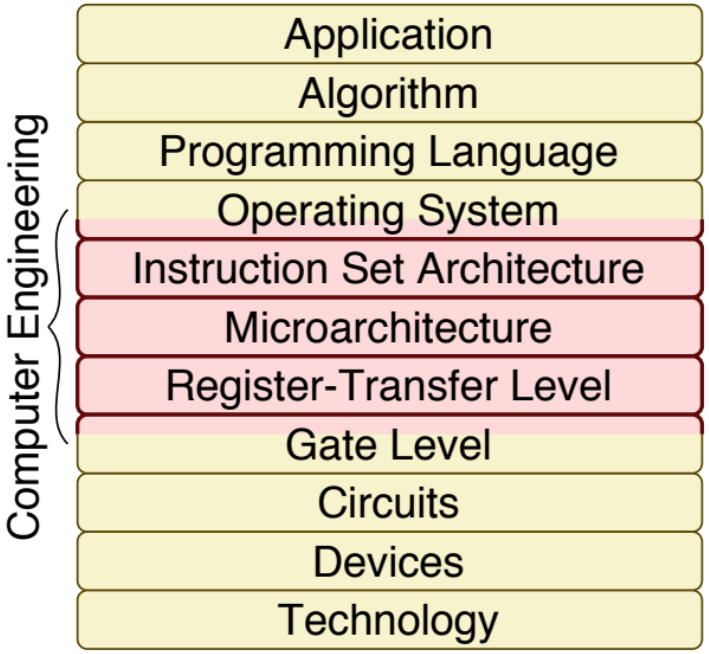
Here are some specific examples of how knowledge of computer architecture can be beneficial:

- A **software developer** can use their knowledge of computer architecture to optimize their code for performance. This can lead to faster and more efficient programs.
- A **computer engineer** can use their knowledge of computer architecture to design new computer systems that are more powerful and reliable.
- A **cybersecurity expert** can use their knowledge of computer architecture to identify and exploit vulnerabilities in computer systems.
- A **data scientist** can use their knowledge of computer architecture to design and optimize algorithms for large-scale data processing.

Abstraction Layers in Modern Systems



The Computer Systems Stack



Sort an array of numbers

2,6,3,8,4,5 -> 2,3,4,5,6,8

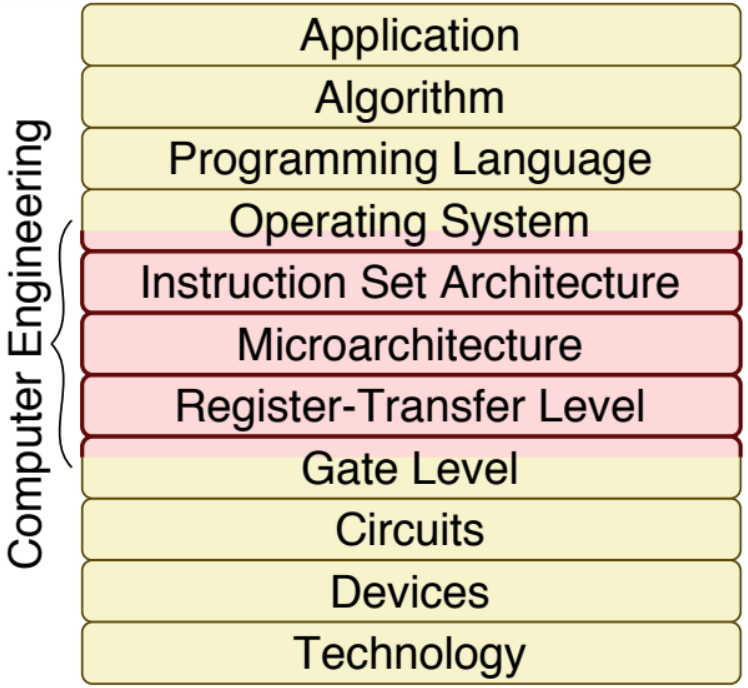
Insertion sort algorithm

1. Find minimum number in input array
2. Move minimum number into output array
3. Repeat steps 1 and 2 until finished

C implementation of insertion sort

```
void isort( int b[], int a[], int n ) {
  for ( int idx, k = 0; k < n; k++ ) {
    int min = 100;
    for ( int i = 0; i < n; i++ ) {
      if ( a[i] < min ) {
        min = a[i];
        idx = i;
      }
    }
    b[k]   = min;
    a[idx] = 100;
  }
}
```

The Computer Systems Stack



Mac OS X, Windows, Linux

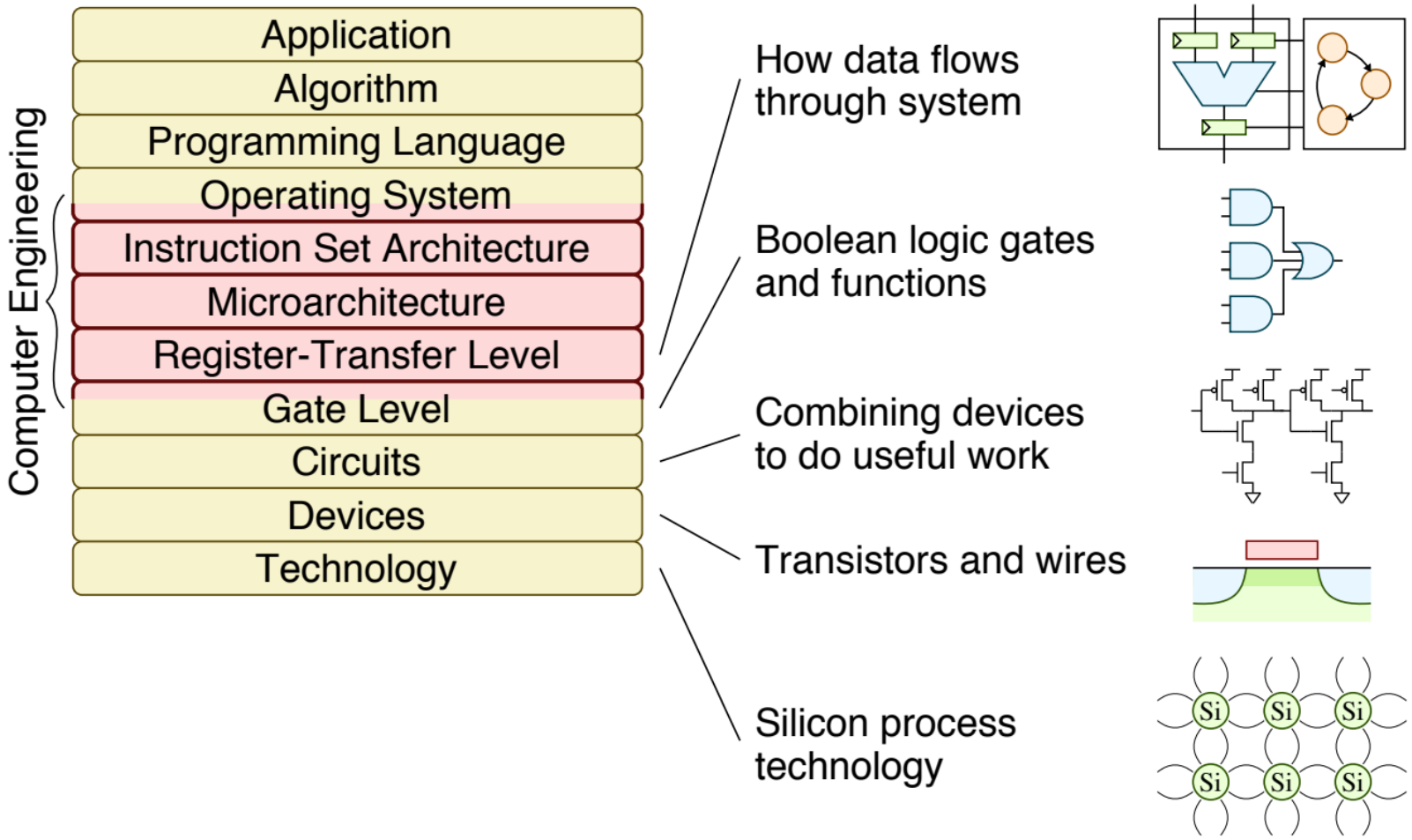
Handles low-level hardware management

MIPS32 Instruction Set

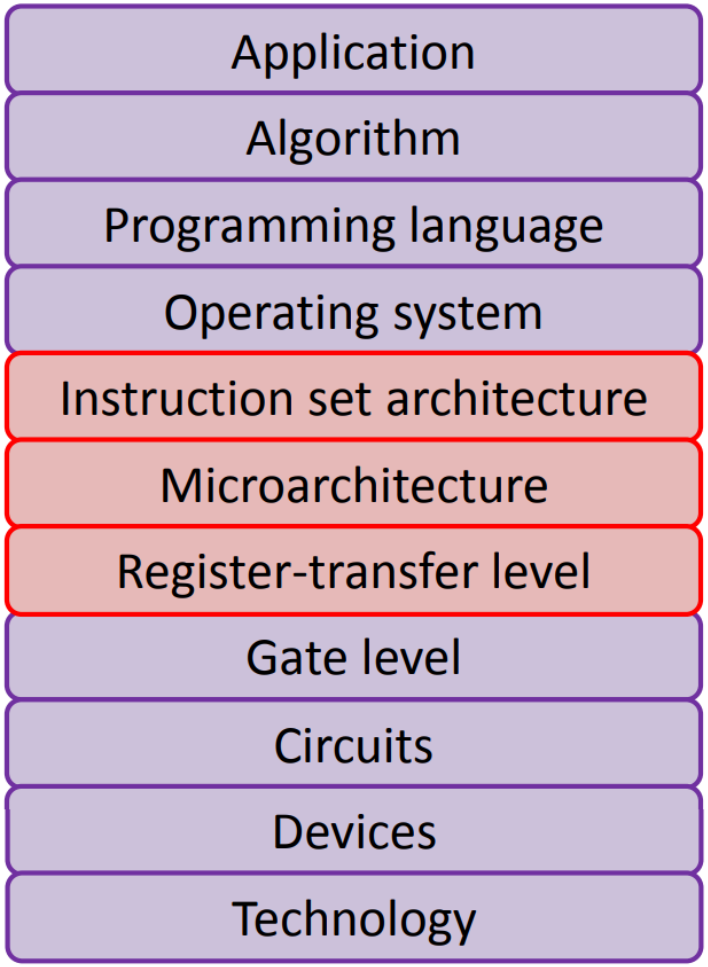
Instructions that machine executes

```
blez  $a2, done
move  $a7, $zero
li    $t4, 99
move  $a4, $a1
move  $v1, $zero
li    $a3, 99
lw    $a5, 0($a4)
addiu $a4, $a4, 4
slt   $a6, $a5, $a3
movn  $v0, $v1, $a6
addiu $v1, $v1, 1
movn  $a3, $a5, $a6
```

The Computer Systems Stack



Application requirements vs. technology constraints



Application requirements

- Suggest how to improve architecture
- Provide revenue to fund development

Computer architects provide feedback to guide application and technology research directions

Technology constraints

- Restrict what can be done efficiently
- New technologies make new arch possible

Three key trends in computer engineering

1. Growing diversity in application requirements motivate growing diversity in computing systems

2. Energy and power constraints motivate transition to multiple processors integrated onto a single chip.

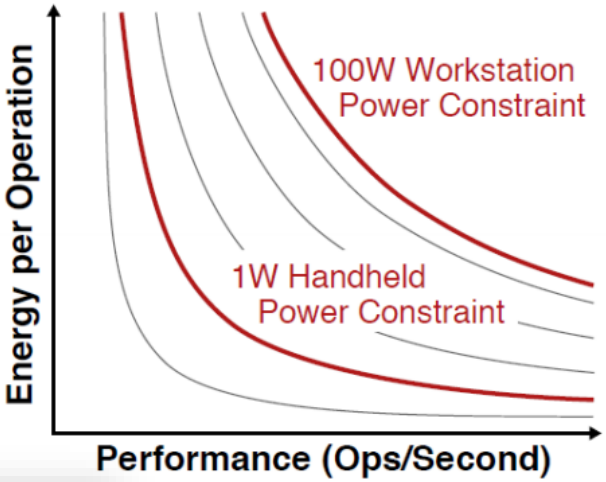
3. Technology scaling challenges motivate new emerging processor, memory, and network device technologies

Trend 1:Growing diversity in apps & systems

Trend 2: Energy/power constraints all modern systems

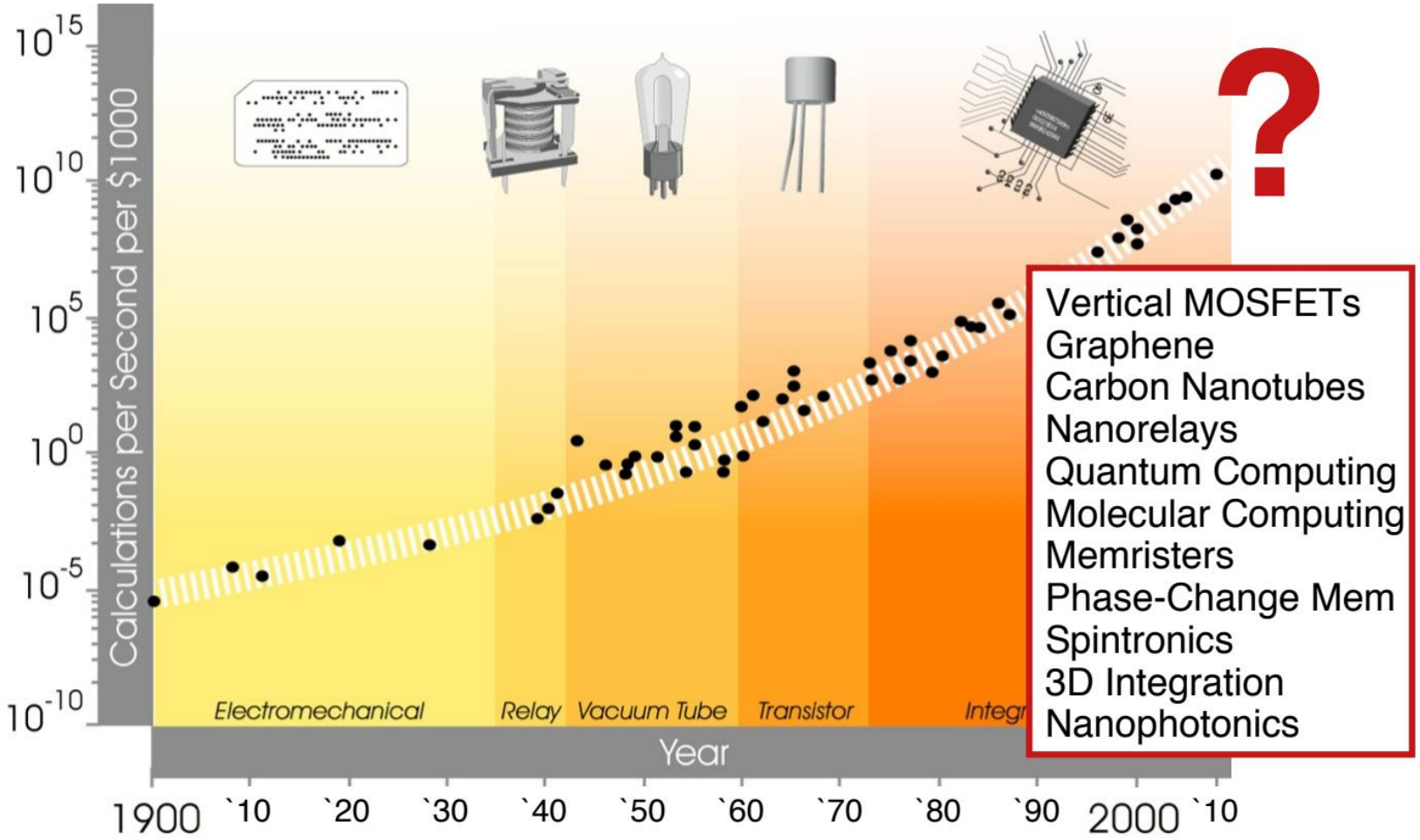
$$\text{Power} = \frac{\text{Energy}}{\text{Second}} = \frac{\text{Energy}}{\text{Ops}} \times \frac{\text{Ops}}{\text{Second}}$$

Power	Energy
Chip packaging	Battery life
Chip cooling	Electricity bill
System noise	Mobile device weight
Case temperature	
Data-center air conditioning	



Transition to multicore processors

Trend 3:Emerging device technologies



Outline

History

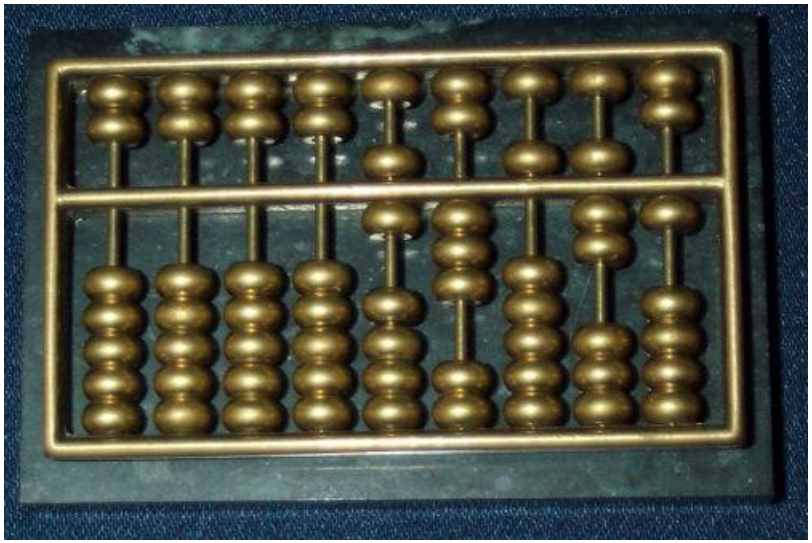
Integrated Circuit Design

Performance

Evolution of Digital Computers

- **First generation**
  Vacuum tube computers (1945~1953)
- **Second generation**
  Transistorized computers (1954~1965)
- **Third generation**
  Integrated circuit computers (1965~1980)
- **Fourth generation**
  Very large scale integrated (VLSI) computers (1980~2000)
- **Fifth generation**
  System-on-chip (SOC) computers (2000~)

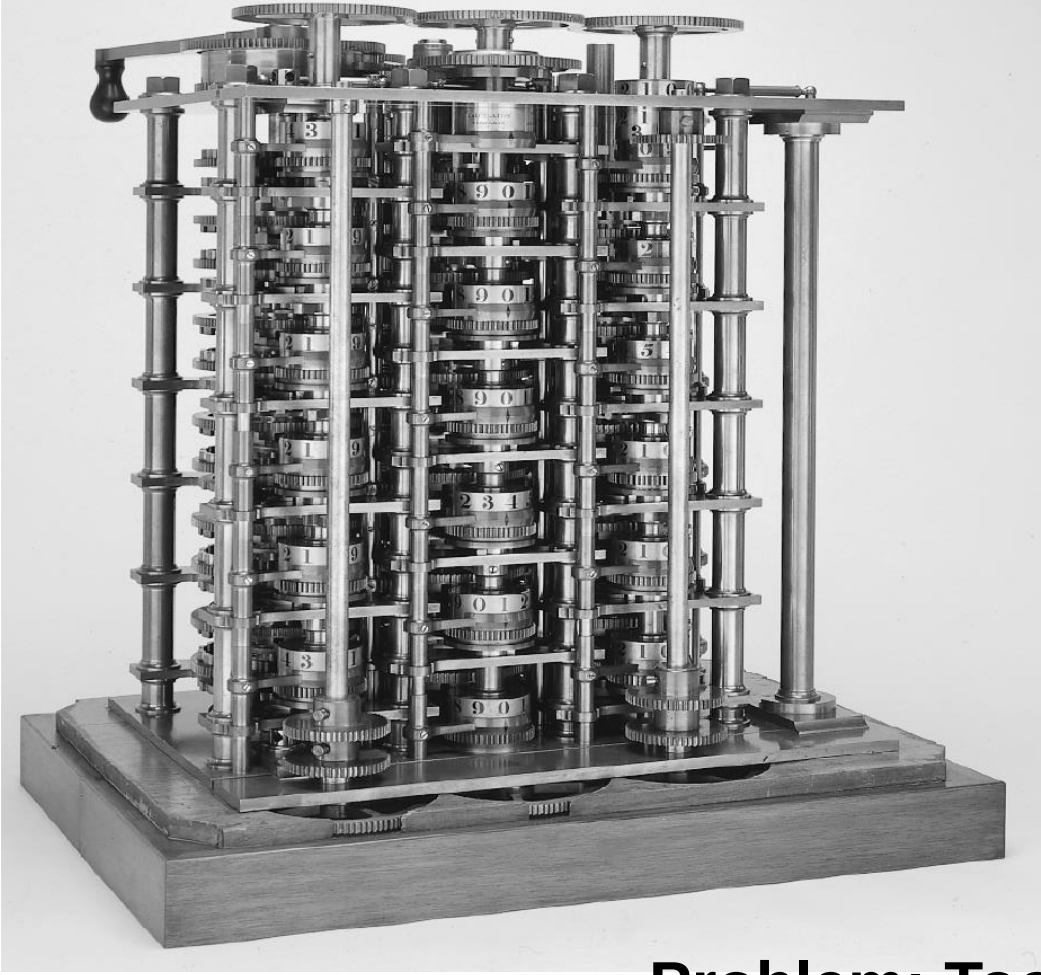
Grandfather of Today Computers



Abacus,
3000 BC (?)

1642, add & sub, Blaise Pascal

The first Computer

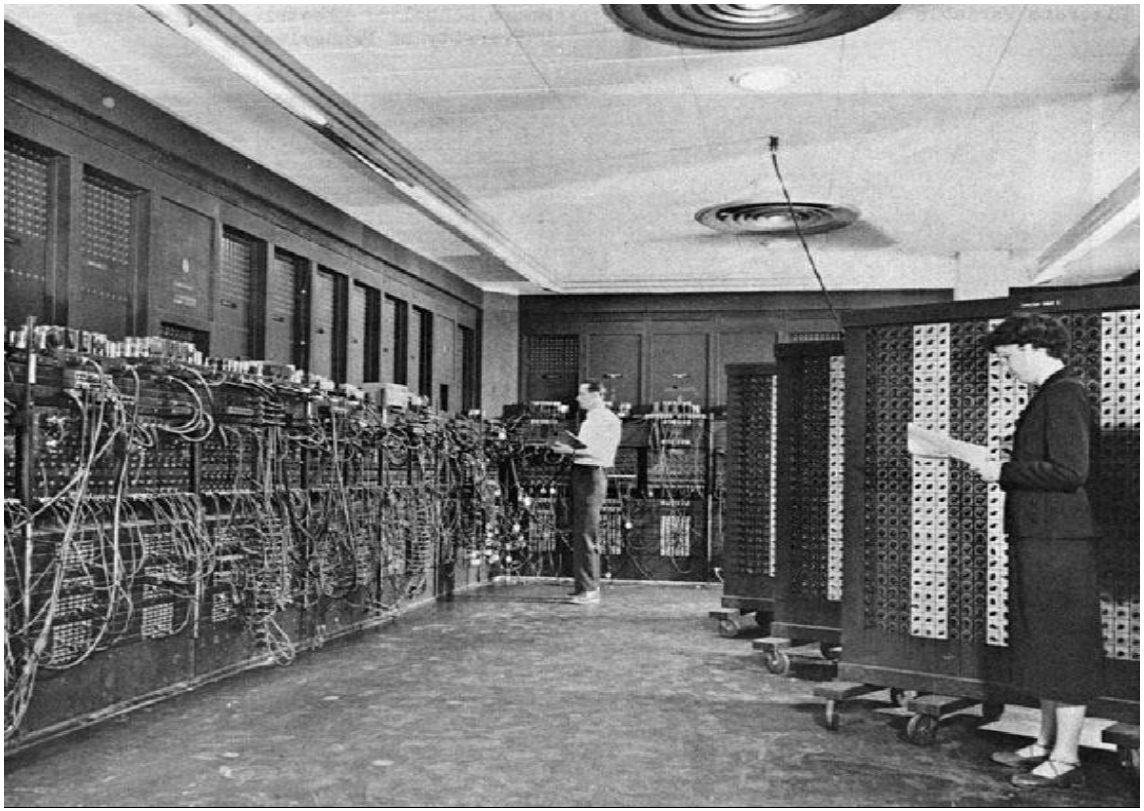


The Babbage Difference Engine (1822)
25,000 parts
cost: £17,470

Mechanical computing devices
Used decimal number system
Could perform basic arithmetic Operations

Problem: Too complex and expensive!

ENIAC - The first electronic computer (1946)

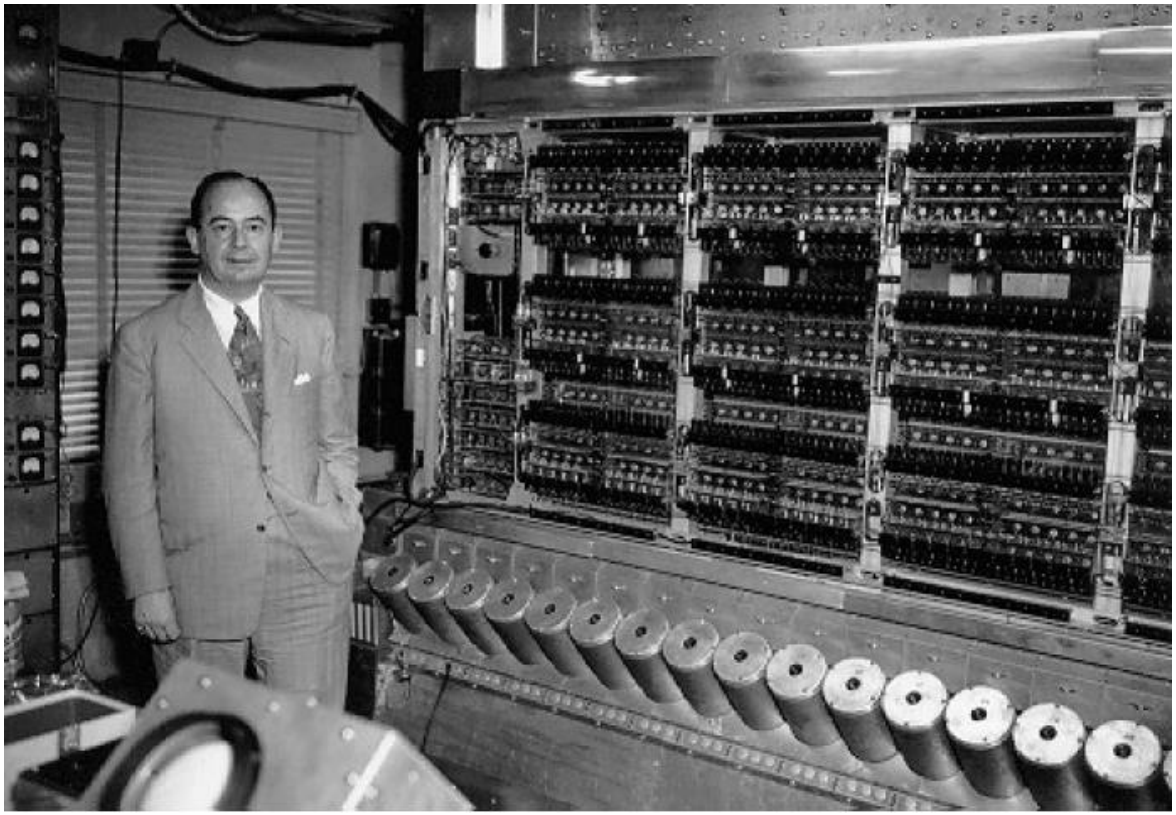


17,468 vacuum tubes
30 tons
63 m²
150 kW
5,000 simple addition
or subtraction operations

Problem: Reliability issues and excessive power consumption!

The IAS machine

Developed 1952 by
John von Neumann

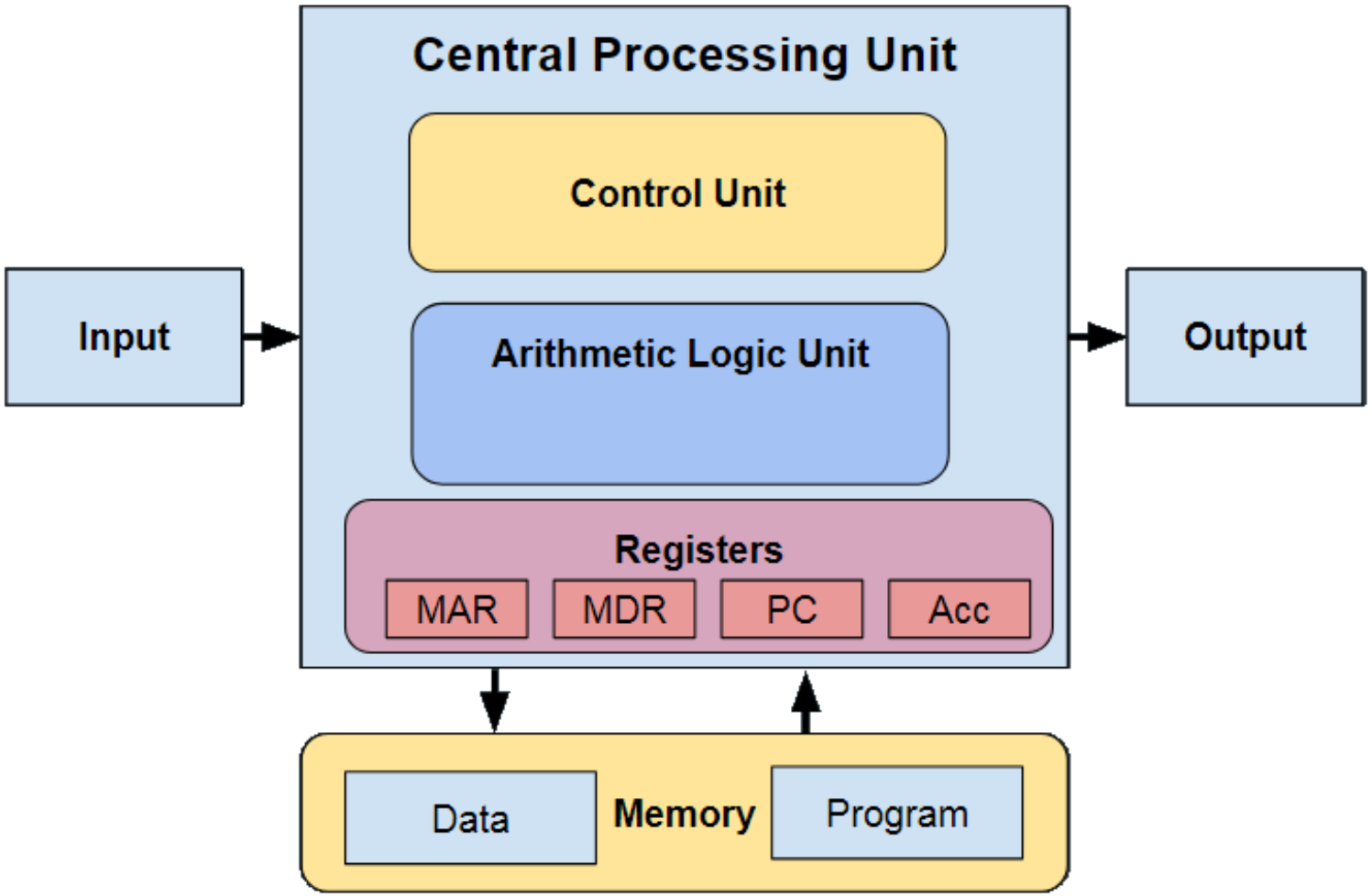


The Von-Neumann Architecture

stored-program concept

- General purpose machine
- Independent of applications
- Flexible & Programmable
- 4 main units
  - Control unit (Instruction counter)
  - Arithmetic unit (Accumulator)
  - Input/Output unit (Connection to the outside)
  - Main memory (to store data and instructions)
- Interconnected by simple buses

The Von-Neumann Architecture



The Von-Neumann Architecture

- Program is composed of a **sequence of instructions**
  - Read one after the other from main memory

- Program execution can be altered
  - Conditional or unconditional jumps
  - Change the current execution
  - Carried out by loading new value into PC register

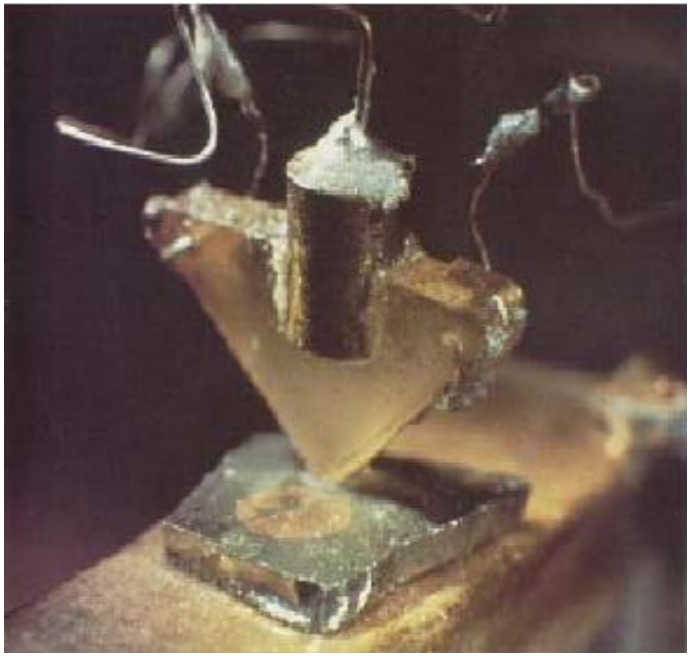
- Usage of binary numbers
  - Just two values allowed per digit: 0/1
  - Easy to implement: voltage yes or no

Von-Neumann Architecture – Today

- Still the dominant architecture in current systems
  - Used in all popular systems / chips
- Only minor modifications
  - Control und Arithmetic unit combined
  - New memory paths between memory and I/O
    Direct Memory Access (DMA)
- Additions to the concept
  - Multiple arithmetic units / Multiple CPUs
  - Parallel processing

Invention of the Transistor

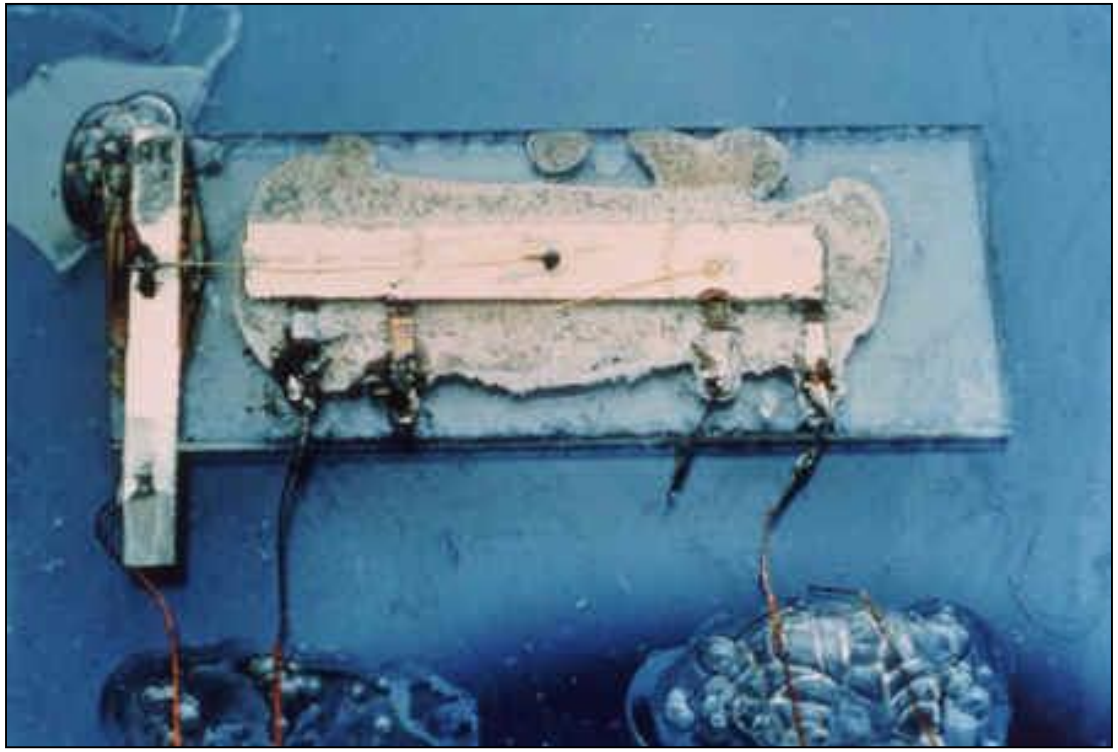
Vacuum tubes invented in 1904 by Fleming
Large, expensive, power-hungry, unreliable

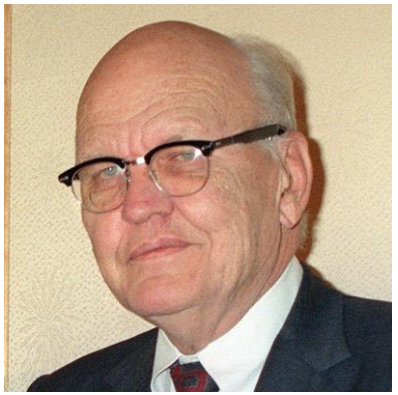


Invention of the bipolar transistor (BJT) 1947
Shockley, Bardeen, Brattain – Bell Labs

Integrated Circuit (IC)

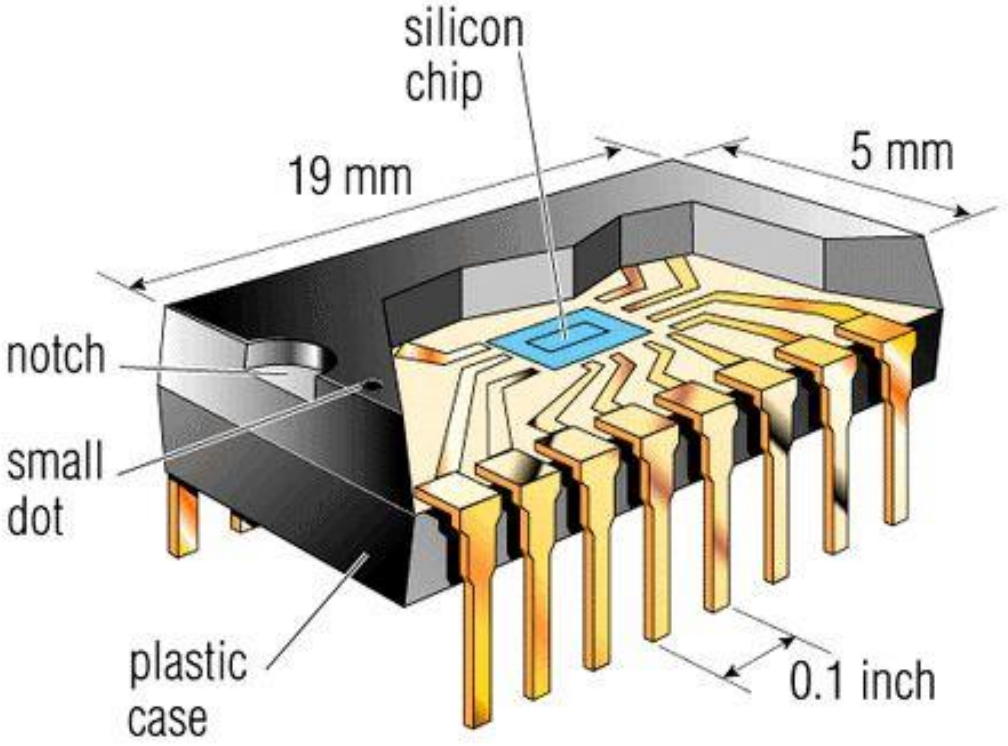
Integrated Circuit (IC)



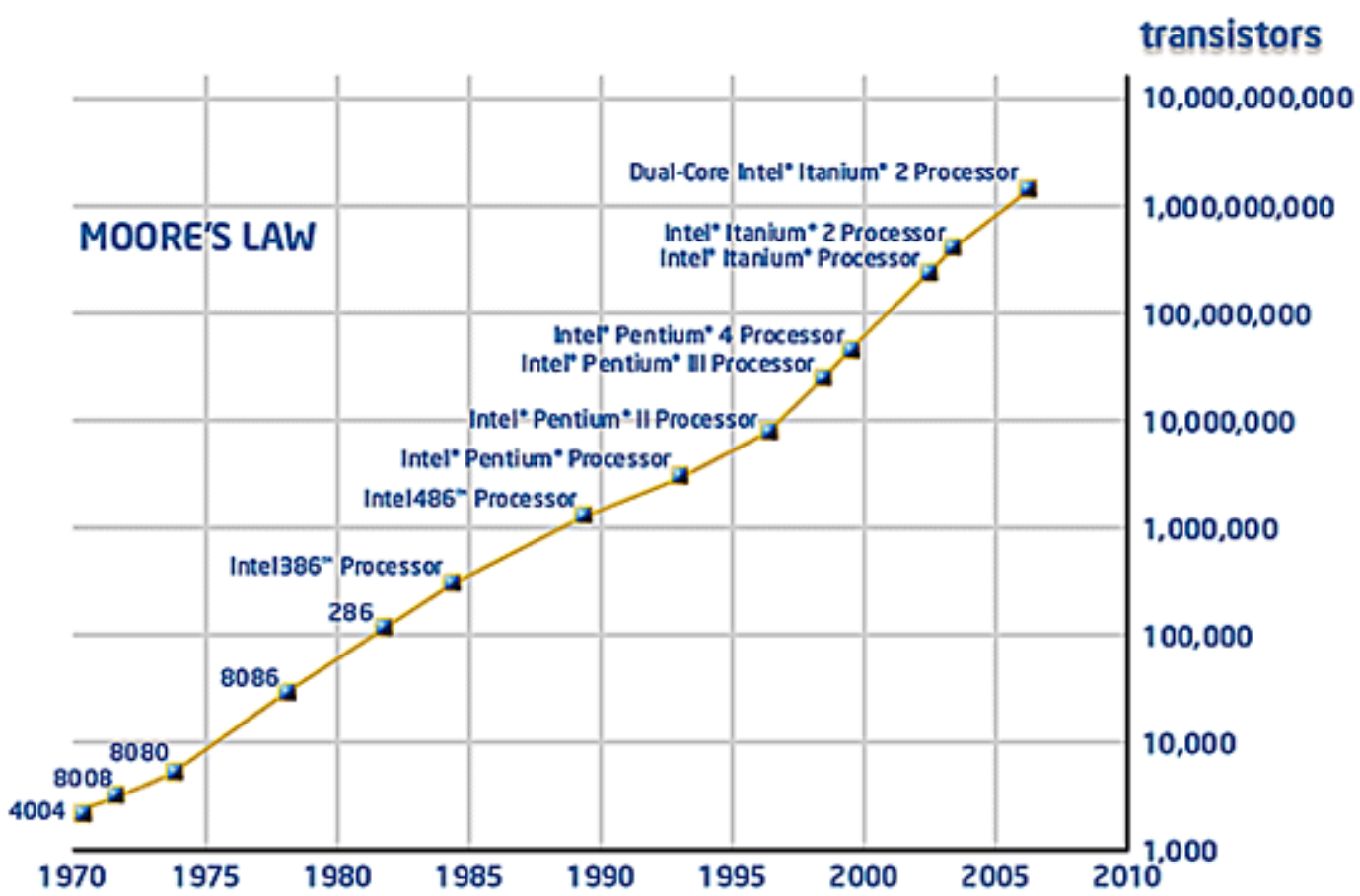


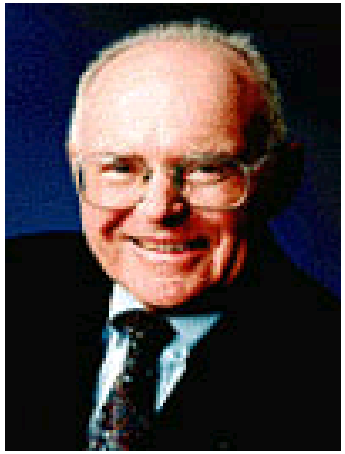
First integrated circuit (germanium), 1958
Jack S. Kilby, Texas Instruments

Integrated Circuit (IC)



Moore's Law





In 1965, Gordon Moore noted that the number of transistors on a chip doubled every 18 to 24 months.

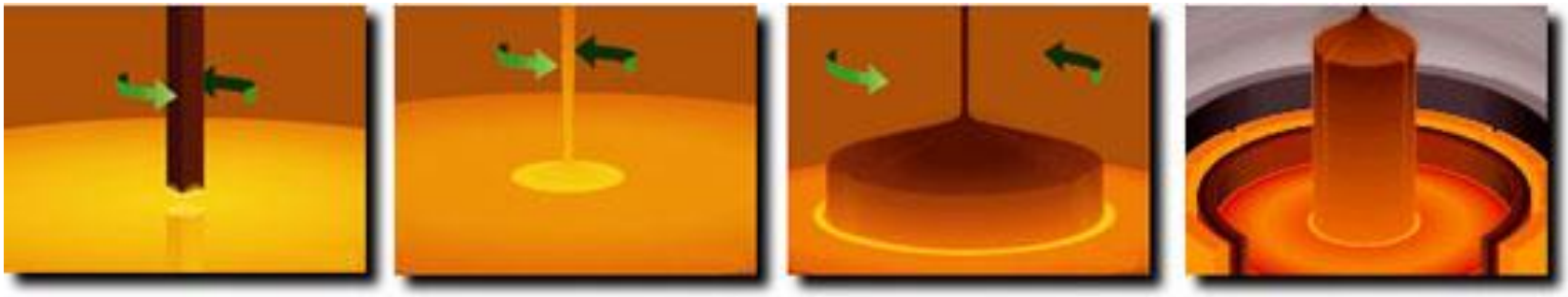
Outline

History

Integrated Circuit Design

Performance

Silicon Ingot growth



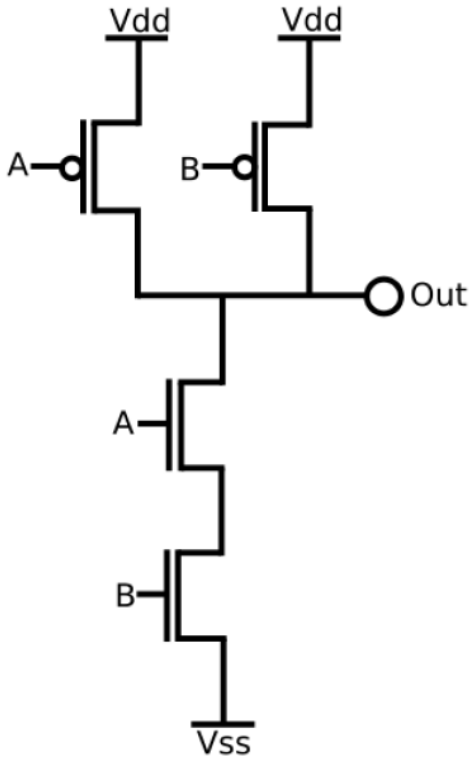
- Czochralski Process is a Technique in Making Single-Crystal Silicon
- A Solid Seed Crystal is Rotated and Slowly Extracted from a Pool of Molten Si
- Requires Careful Control to Give Crystals Desired Purity and Dimensions

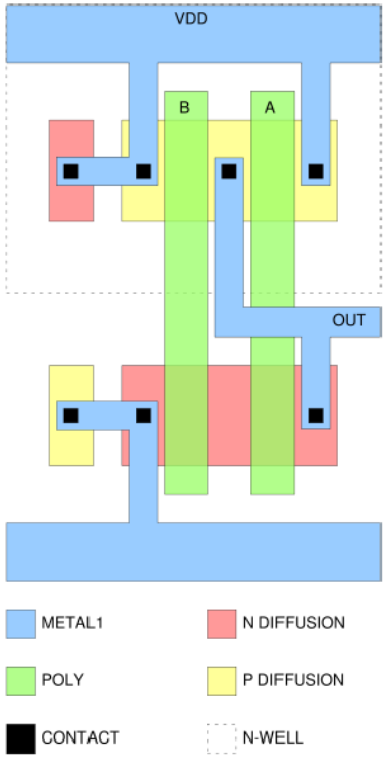
Silicon Ingot

- The Silicon Cylinder is Known as an **Ingot**
- Typical Ingot is About 1 or 2 Meters in Length
- Can be Sliced into Hundreds of Smaller Circular Pieces Called **Wafers**
- Each Wafer Yields Hundreds or Thousands of Integrated Circuits

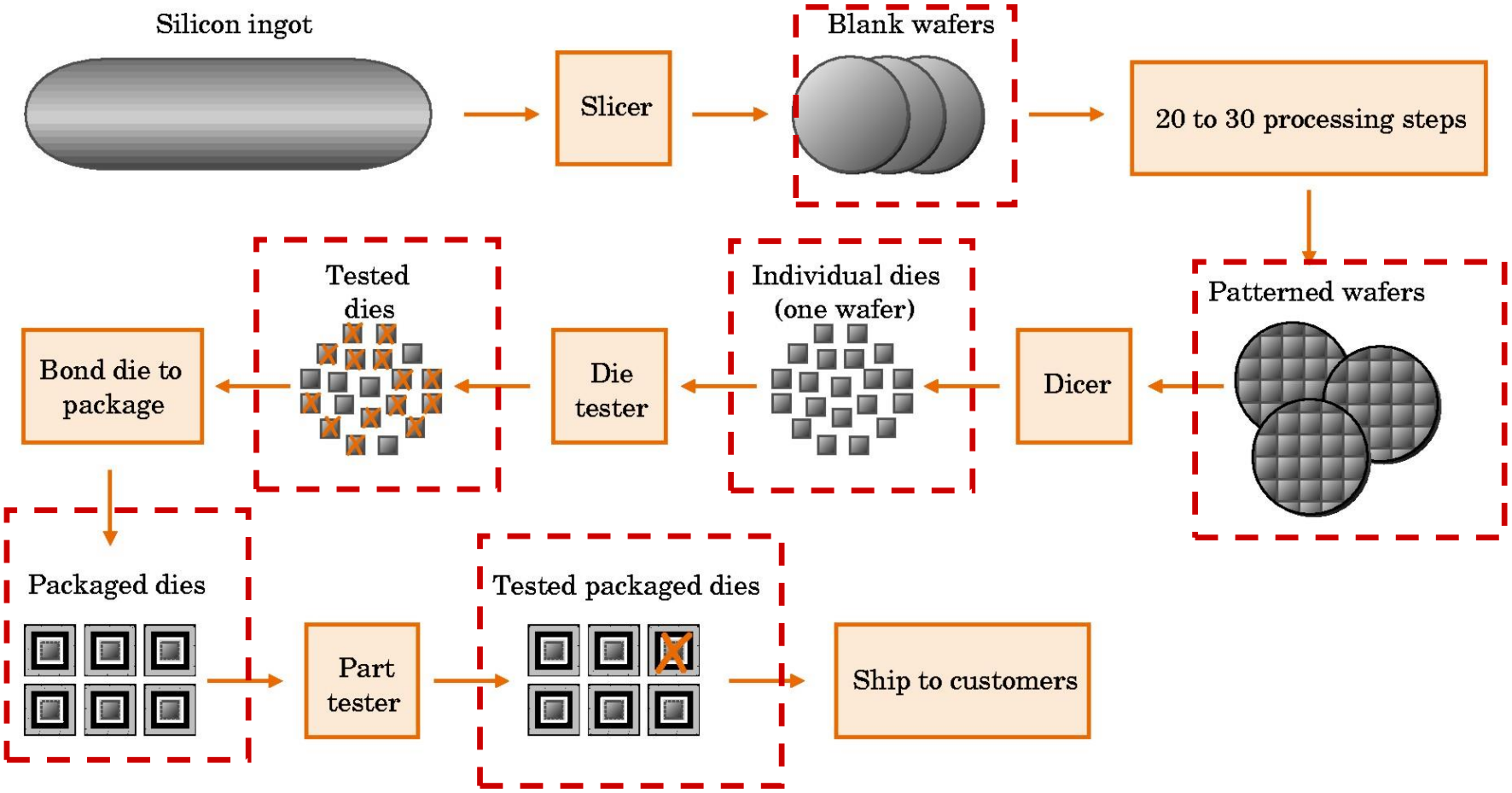
CMOS NAND Gate

- NAND logic built with CMOS technology

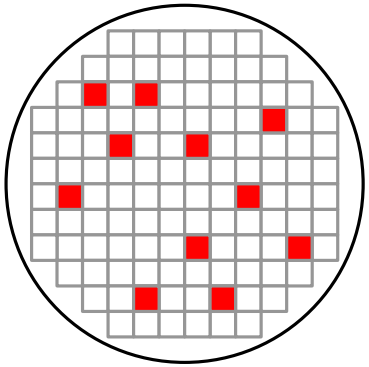




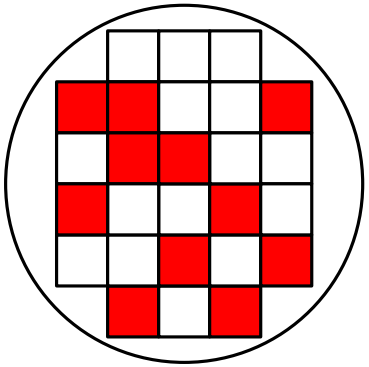
Chip Manufacturing Process



Effect of Die Size on Yield



120 dies, 109 good

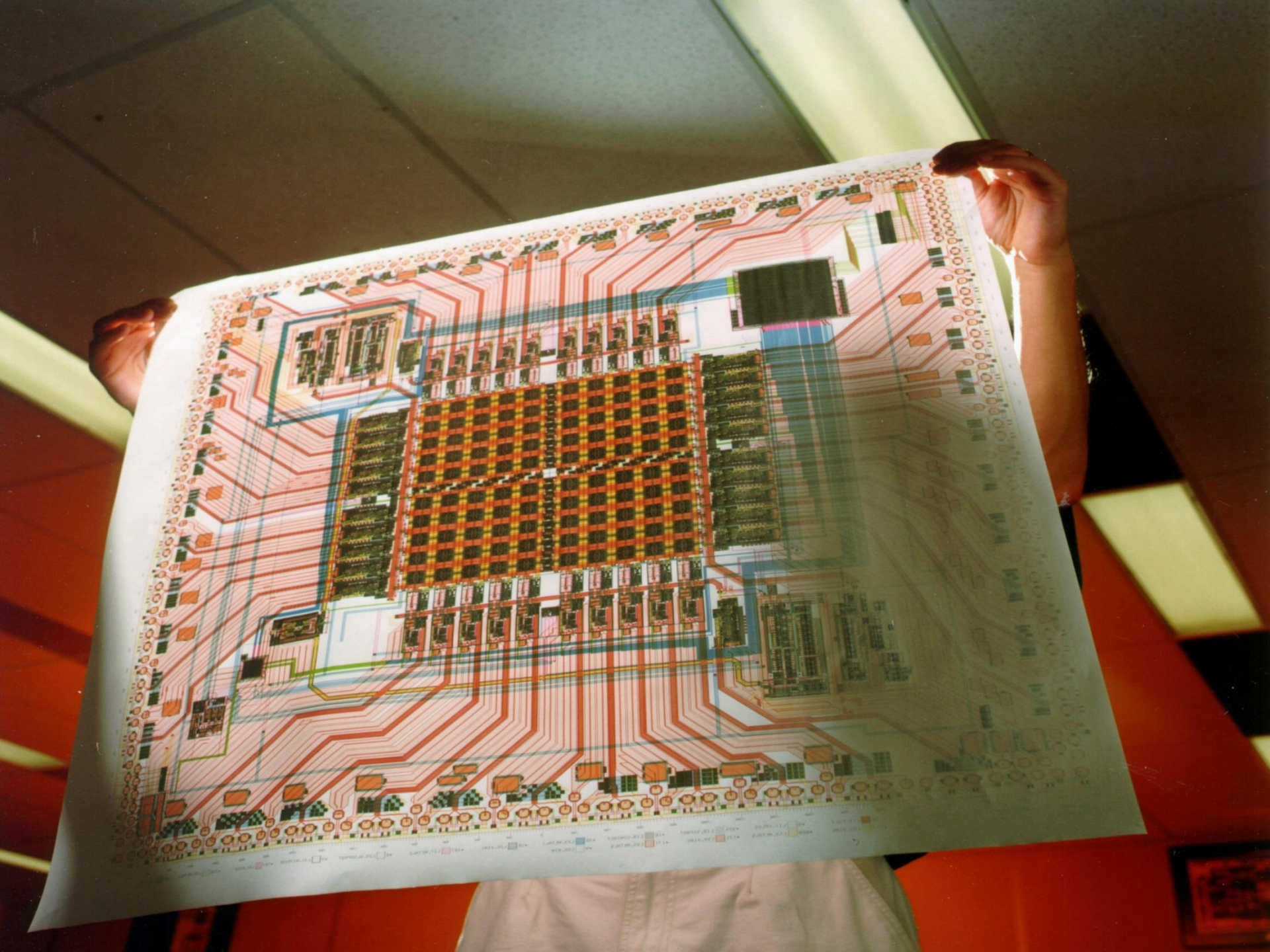


26 dies, 15 good

Visualizing the dramatic decrease in yield with larger dies.

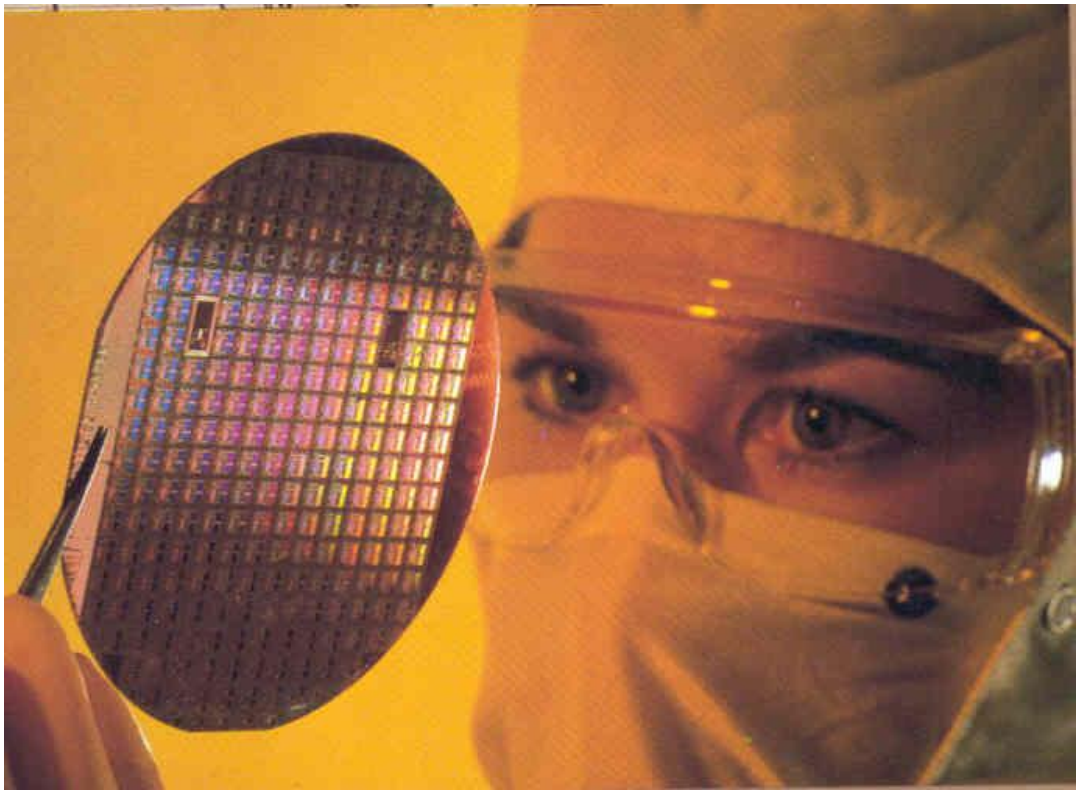
Die yield =$_{def}$ (number of good dies) / (total number of dies)

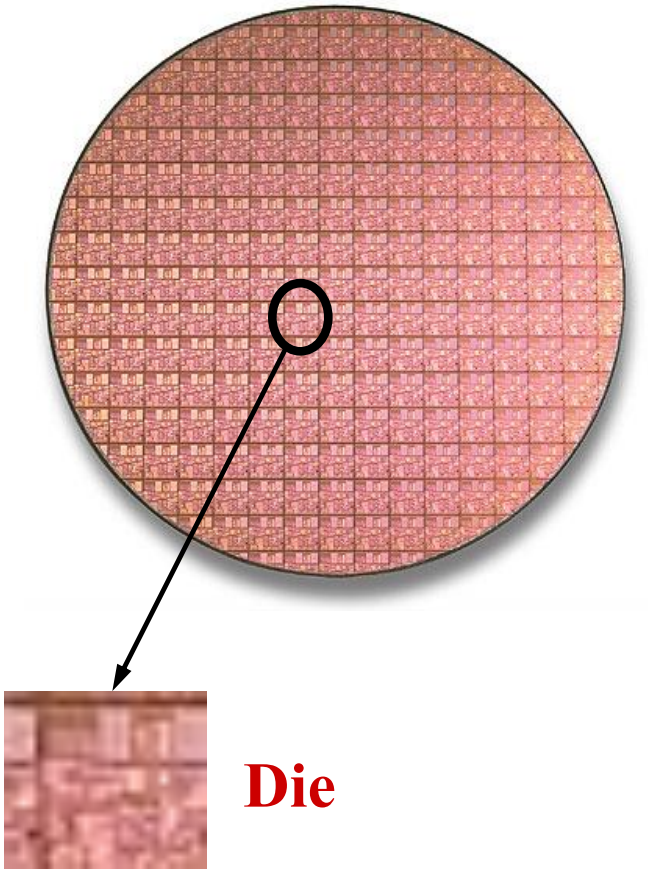
Die cost = (cost of wafer) / (total number of dies × die yield)
= (cost of wafer) × (die area / wafer area) / (die yield)



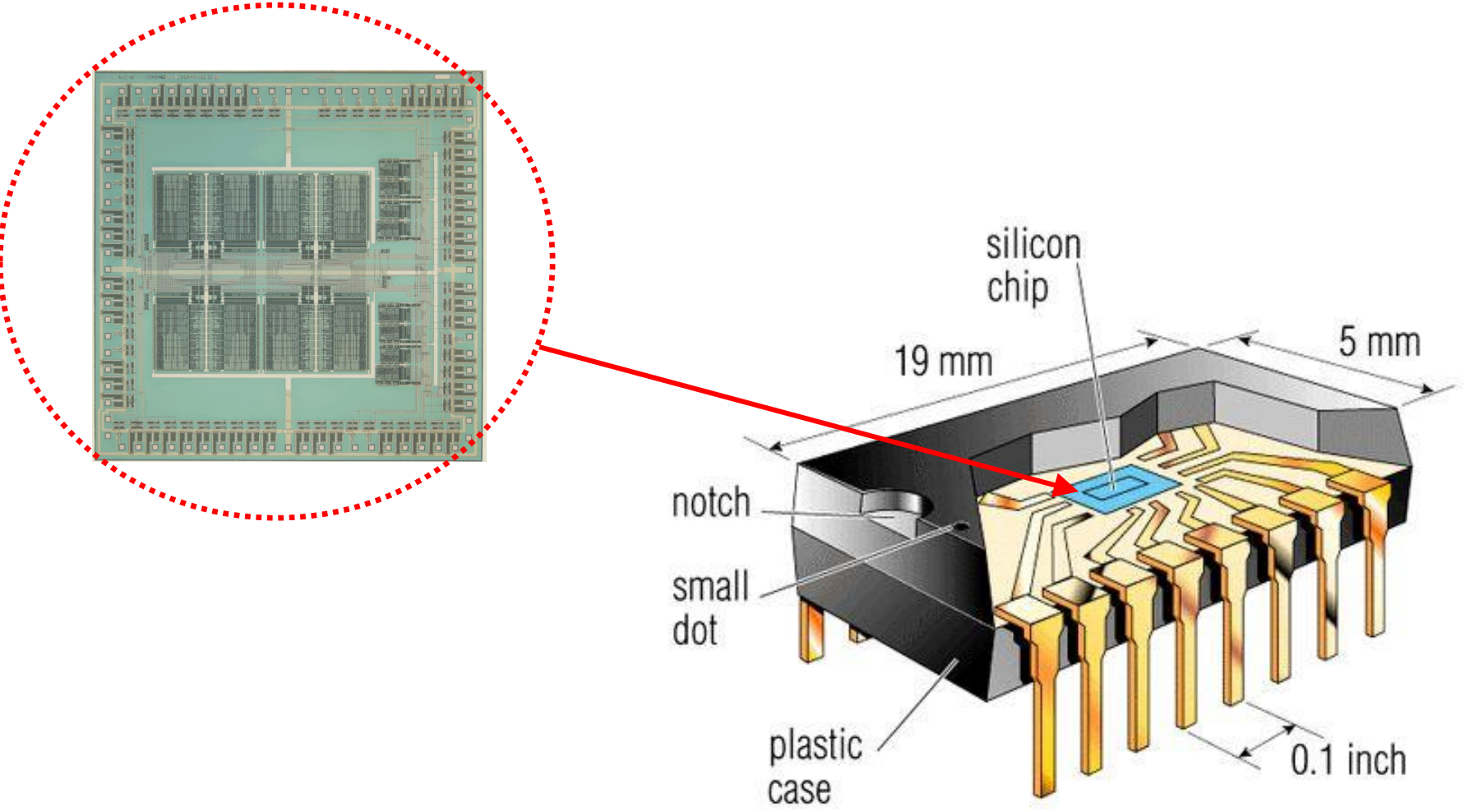
Clean Room

Wafer



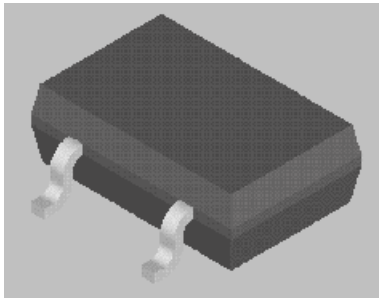


Die

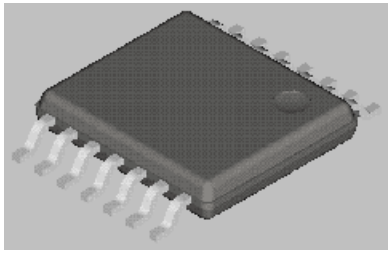


Package Types

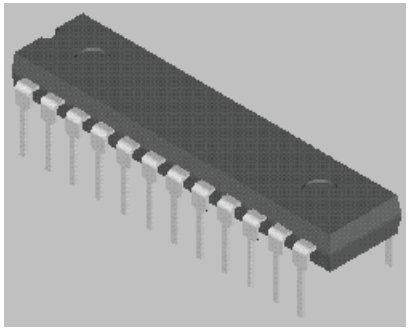
- Small Outline Transistor (SOT)



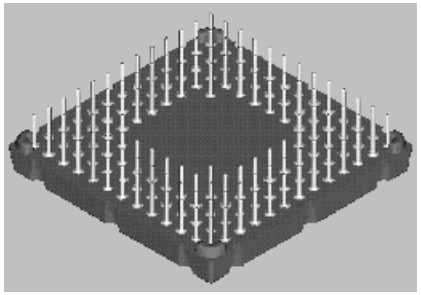
- **Small Outline Package (SOP)**

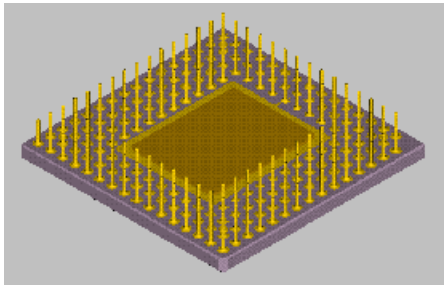


- **Dual-In-Line Package (DIP)**

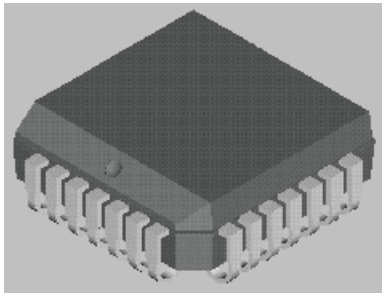


- **Plastic/Ceramic Pin Grid Array (PPGA/CPGA)**





- **Plastic Leaded Chip Carrier (PLCC)**

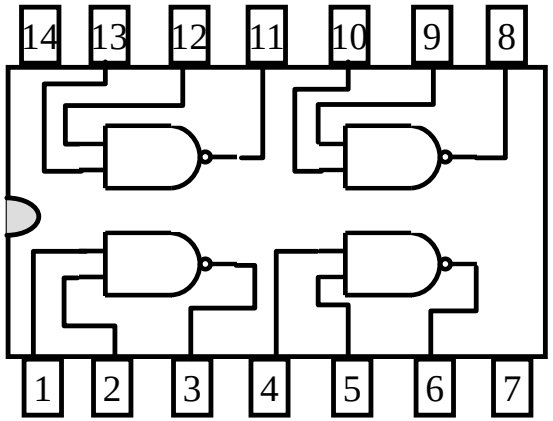


Levels of Integration

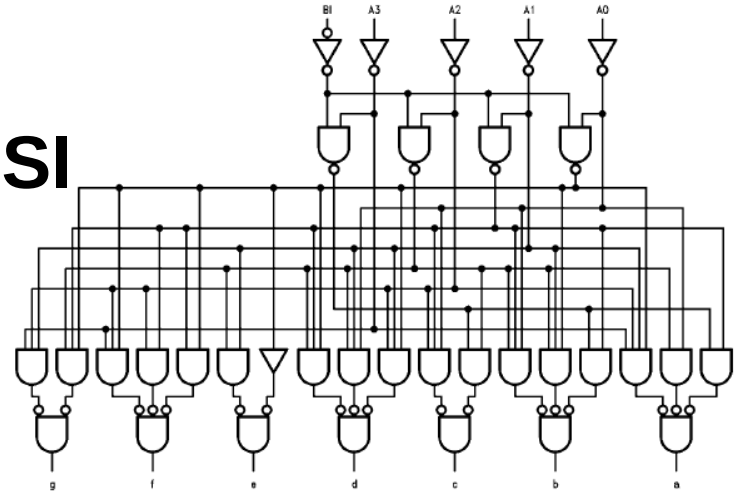
Small-Scale Integration	SSI	<100	1963
Medium-Scale Integration	MSI	100-300	1970
Large-Scale Integration	LSI	300 - 30000	1975
Very Large-Scale Integration	VLSI	30000 - 1million	1980
Ultra-Large Scale Integration	ULSI	>1million	1990
Giga Scale Integration	GSI	>1billion	2010

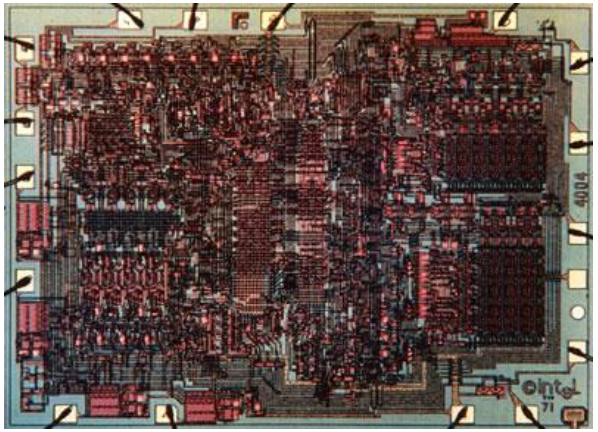
Levels of Integration

SSI



MSI

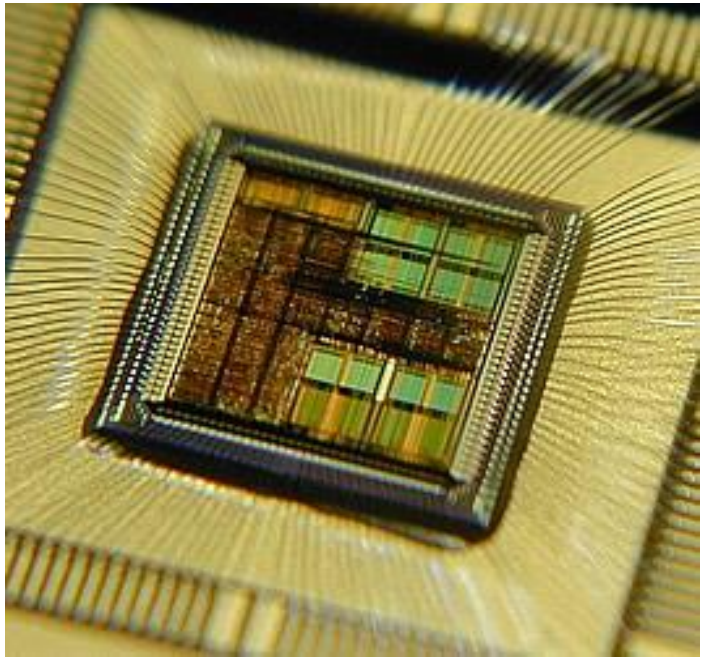




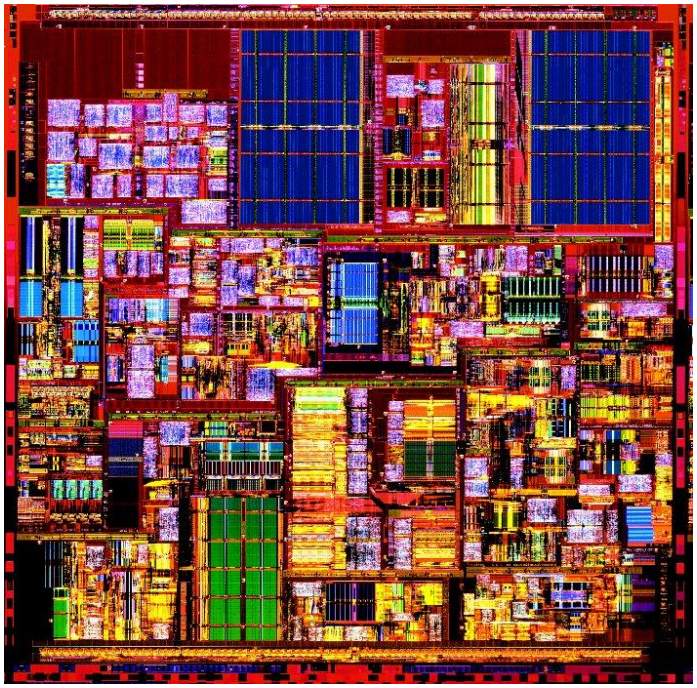
LSI

Intel 4004
~2300 transistors

Levels of Integration



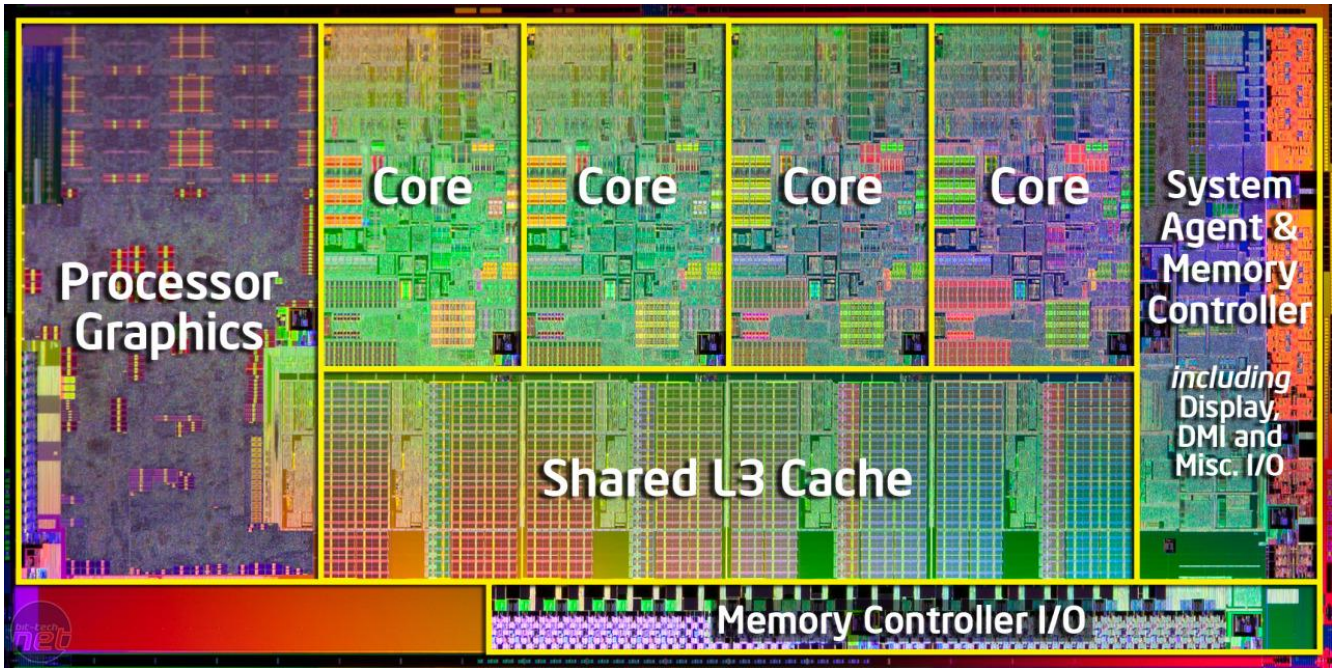
VLSI



ULSI

Intel Pentium 4
55 million Transistors

Levels of Integration



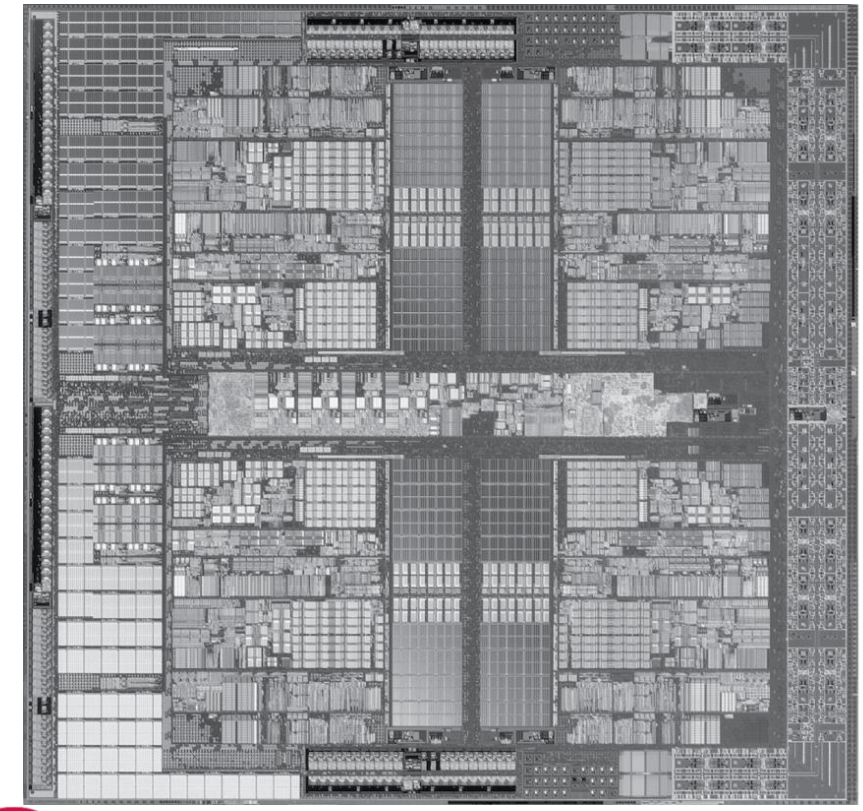
GSI **intel sandy-bridge (32 nm technology)**

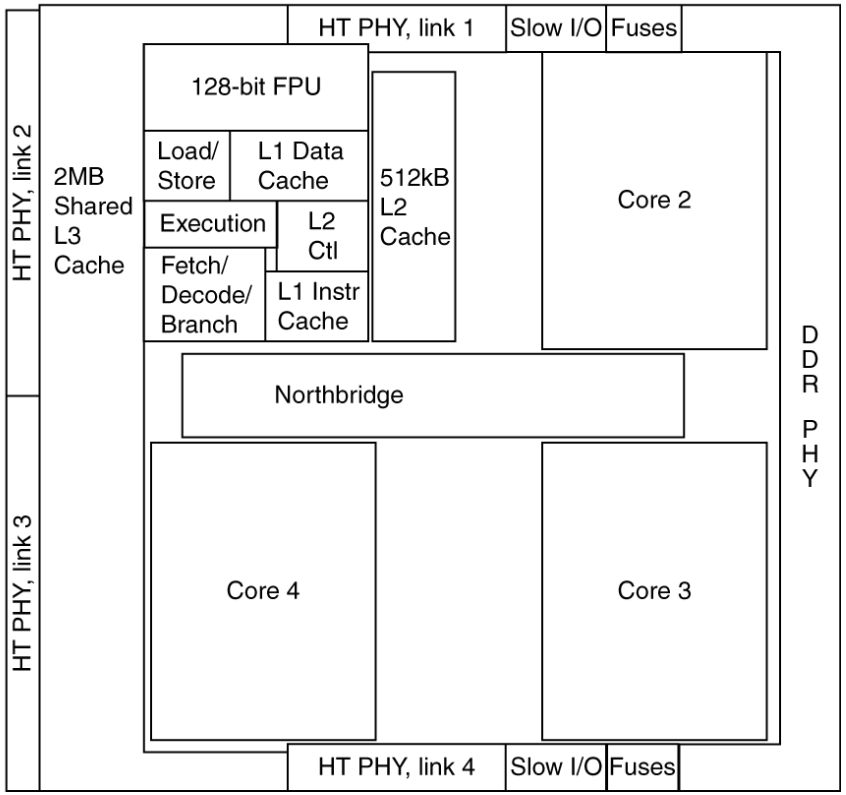
(A sheet of paper is about 100,000 nanometers thick.
A human hair measures roughly 50,000 to 100,000 nanometers in diameter)

Inside a Multicore Processor Chip

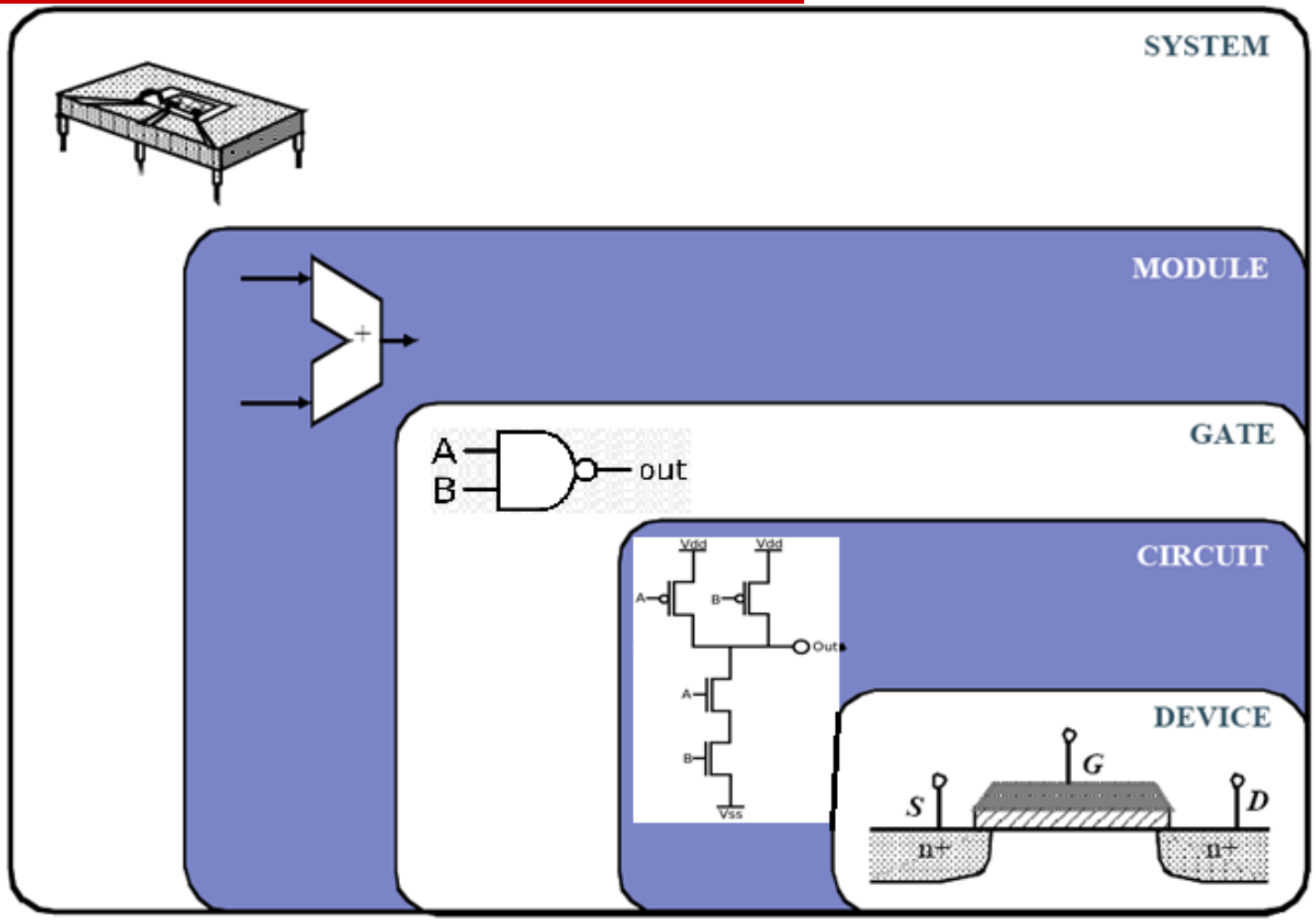
AMD Barcelona: 4 Processor **C**ores

3 Levels of Caches





Design Abstraction Levels



ICs In Human Life

Cell Phones

Digital Cameras

Hearing aids

Automotive

Biomedical

Computers

Outline

History

Integrated Circuit Design

Performance

Performance

Performance of Aircraft: An Analogy

Aircraft	Passengers	Range (km)	Speed (km/h)	Price ($M)
Airbus A310	250	8 300	895	120
Boeing 747	470	6 700	980	200
Boeing 767	250	12 300	885	120
Boeing 777	375	7 450	980	180
Concorde	130	6 400	2 200	350
DC-8-50	145	14 000	875	80

Speed of sound ≈ 1220 km / h

Different Views of Performance

Performance from the viewpoint of a passenger: Speed

Note, however, that flight time is but one part of total travel time. Also, if the travel distance exceeds the range of a faster plane, a slower plane may be better due to not needing a refueling stop

Performance from the viewpoint of an airline: Throughput

Measured in passenger-km per hour (relevant if ticket price were proportional to distance traveled, which in reality it is not)

Airbus A310	250 × 895 = 0.224 M passenger-km/hr
Boeing 747	470 × 980 = 0.461 M passenger-km/hr
Boeing 767	250 × 885 = 0.221 M passenger-km/hr
Boeing 777	375 × 980 = 0.368 M passenger-km/hr
Concorde	130 × 2200 = 0.286 M passenger-km/hr
DC-8-50	145 × 875 = 0.127 M passenger-km/hr

Performance from the viewpoint of FAA: Safety

CPU Performance and Speedup

Performance = 1 / CPU execution time

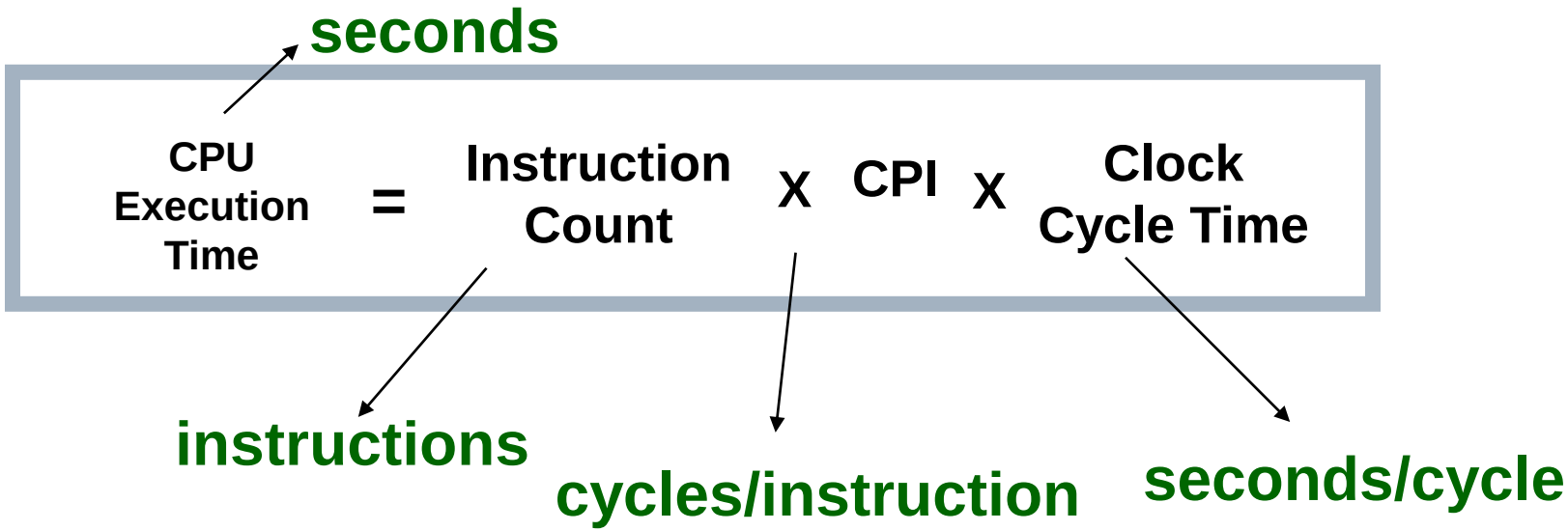
(Performance of M_1) / (Performance of M_2) = Speedup of M_1 over M_2
= (Execution time of M_2) / (Execution time M_1)

Terminology: M_1 is *x* times as fast as M_2 (e.g., 1.5 times as fast)
M_1 is 100(*x* – 1)% faster than M_2 (e.g., 50% faster)

CPU time = Instructions × (Cycles per instruction) × (Secs per cycle)
= Instructions × CPI / (Clock rate)

Instruction count, CPI, and clock rate are not completely independent, so improving one by a given factor may not lead to overall execution time improvement by the same factor.

CPU Execution Time



- Improve performance => reduce execution time
  - Reduce instruction count (ISA, Programmer, Compiler)
  - Reduce cycles per instruction (ISA, Machine designer)
  - Reduce clock cycle time (Hardware designer, Physicist)

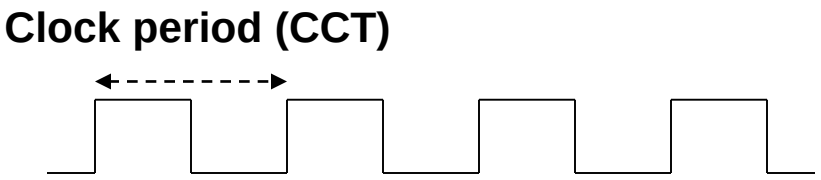
Elaboration on the CPU Time Formula

CPU time = IC × CPI × CCT = IC × CPI / (Clock rate)

Instruction count: Number of instructions executed, not number of instructions in our program (**dynamic** count)

CPI (average): Is calculated based on the **dynamic** instruction mix and knowledge of how many clock cycles are needed to execute various instructions (or instruction classes)

Clock rate: 1 GHz = 10^9 cycles / s (cycle time 10^{-9} s = 1 ns)
200 MHz = 200 × 10^6 cycles / s (cycle time = 5 ns)



Dynamic Instruction Count

How many instructions are executed in this program fragment?

Each "for" consists of two instructions: increment index, check exit condition

```
250 instructions
for i = 1, 100 do
20 instructions
    for j = 1, 100 do
    40 instructions
        for k = 1, 100 do
        10 instructions
        endfor
    endfor
endfor
```

Static count = 326

12,422,450 Instructions

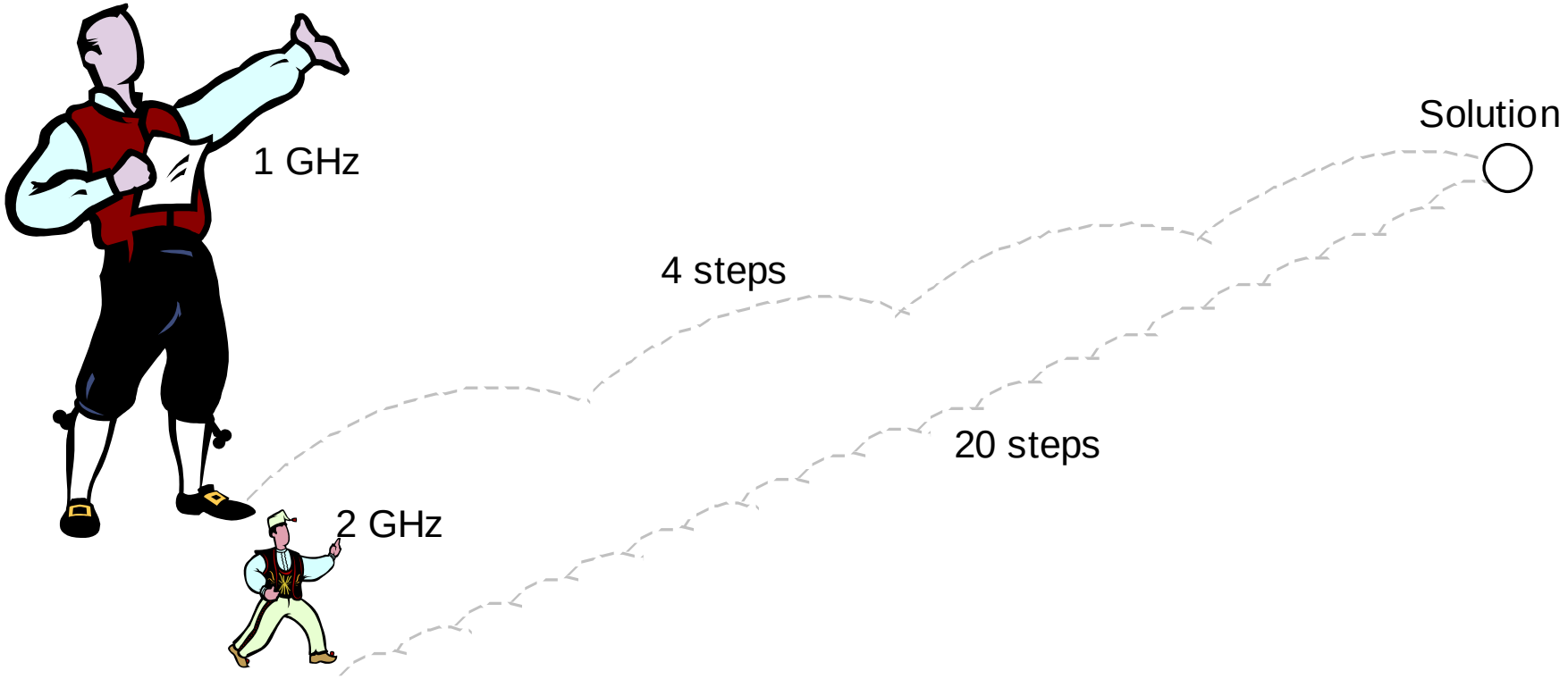
2 + 20 + 124,200 instructions
100 iterations
12,422,200 instructions in all

2 + 40 + 1200 instructions
100 iterations
124,200 instructions in all

2 + 10 instructions
100 iterations
1200 instructions in all

for i = 1, n
while x > 0

Faster Clock ≠ Shorter Running Time



Faster steps do not necessarily mean shorter travel time.

Effect of Instruction Mix on Performance

Consider two applications DC and RS and two machines M_1 and M_2:

Class	Data Comp.	Reactor Sim.	M_1's CPI	M_2's CPI
A: Ld/Str	25%	32%	4.0	3.8
B: Integer	32%	17%	1.5	2.5
C: Sh/Logic	16%	2%	1.2	1.2
D: Float	0%	34%	6.0	2.6
E: Branch	19%	9%	2.5	2.2
F: Other	8%	6%	2.0	2.3

Find the effective CPI for the two applications on both machines.

Solution

a. CPI of DC on M_1: $0.25 \times 4.0 + 0.32 \times 1.5 + 0.16 \times 1.2 + 0 \times 6.0 + 0.19 \times 2.5 + 0.08 \times 2.0 = 2.31$

DC on M_2: 2.54 RS on M_1: 3.94 RS on M_2: 2.89

MIPS (million instructions per second)

$$MIPS = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

Example

Code from	Instruction Counts (in billions) for each instruction set		
	A (1 CPI)	B (2 CPI)	C (3 CPI)
Compiler 1	5	1	1
Compiler 2	10	1	1

Clock rate = 4GHz A,B,C : Instruction Classes

- **Which code sequence will execute faster according to MIPS?**
- **According to execution time?**

Execution time & MIPS

$$\text{CPU clock cycles1} = (5 * 1 + 1 * 2 + 1 * 3) * 10^9 = 10 * 10^9$$

$$\text{CPU clock cycles2} = (10 * 1 + 1 * 2 + 1 * 3) * 10^9 = 15 * 10^9$$

$$\text{Execution time1} = \frac{10 * 10^9}{4 * 10^9} = 2.5 \text{ seconds}$$

$$\text{Execution time2} = \frac{15 * 10^9}{4 * 10^9} = 3.75 \text{ seconds}$$

Execution time & MIPS (2)

$$MIPS_1 = \frac{(5+1+1)\times 10^9}{2.5\,\text{seconsd}\times 10^6} = 2800$$

$$MIPS_2 = \frac{(10+1+1)\times 10^9}{3.75\times 10^6} = 3200$$

Comparing the Overall Performance

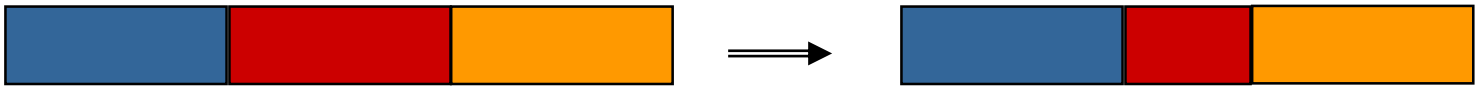
Measured or estimated execution times for three programs.

	Time on machine X	Time on machine Y	Speedup of Y over X	**Speedup of X over Y**
Program A	20	200	0.1	**10**
Program B	1000	100	10.0	**0.1**
Program C	1500	150	10.0	**0.1**
	Arithmetic mean		**6.7**	**3.4**
	Geometric mean		**2.15**	**0.46**

Performance Enhancement- Amdahl's Law

Speedup due to enhancement E:

$$\text{Speedup(E)} = \frac{\text{ExTime w/o E}}{\text{ExTime w/ E}} = \frac{\text{Performance w/ E}}{\text{Performance w/o E}}$$



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

Amdahl's Law

$$\text{ExTime}_{new} = \text{ExTime}_{old} \times \left[(1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}} \right]$$

$$\text{Speedup}_{overall} = \frac{\text{ExTime}_{old}}{\text{ExTime}_{new}} = \frac{1}{(1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}}}$$

Amdahl's Law

- Floating point instructions improved to run 2X; but only 15% of actual instructions are FP

$$ExTime_{new} = ExTime_{old} \times (0.85 + (0.15)/2) = 0.925 \times ExTime_{old}$$

$$Speedup_{overall} = \frac{1}{0.925} = 1.081$$

Amdahl's Law

Execution time OLD

Execution time New

Law of diminishing return:

Focus on the common case!

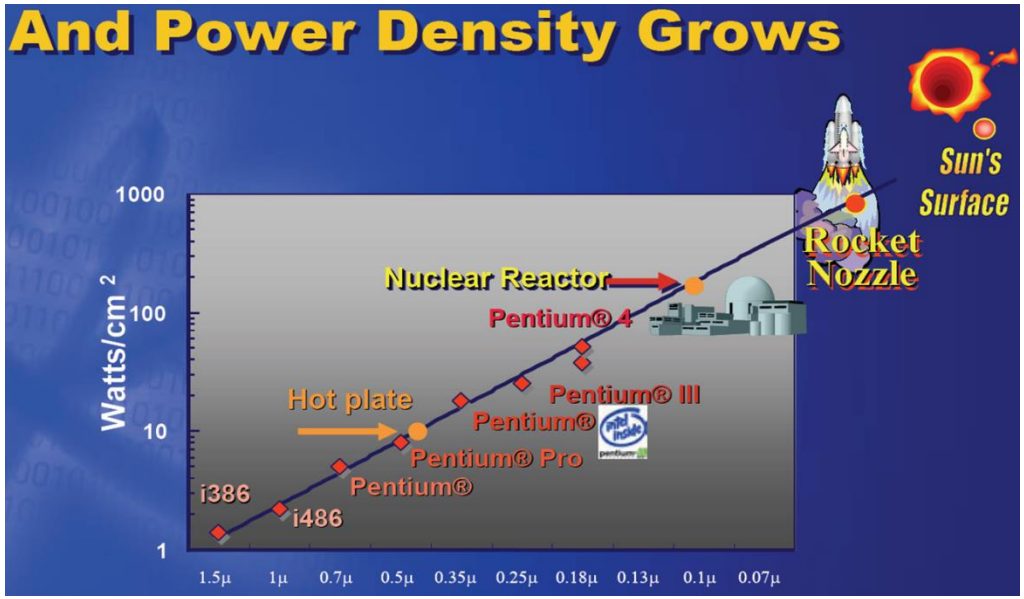
Another Key Metric: Power Dissipation

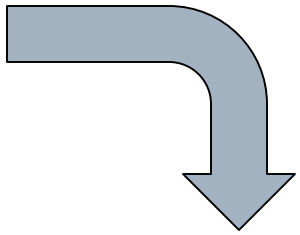
Example:

– If the voltage and frequency of a processing core are both reduced by 15% what would be the impact on dynamic power?

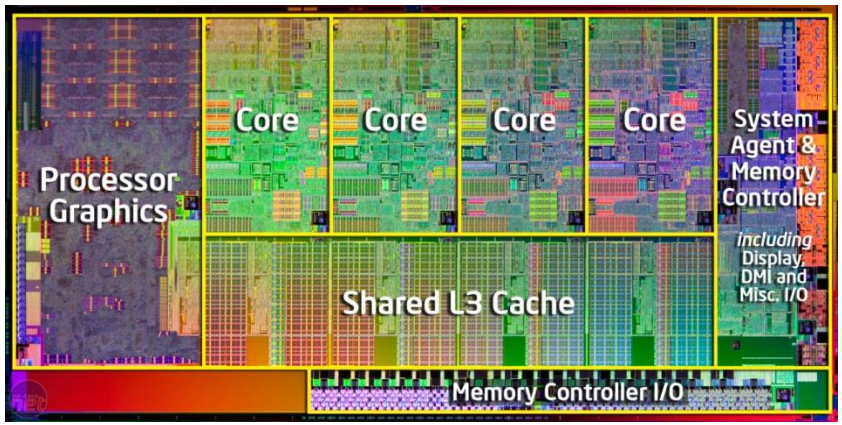
$$\text{Power Save} = \frac{P_{new}}{P_{old}} = \frac{C \times (V \times 0.85)^2 \times (F \times 0.85)}{C \times V^2 \times F} = 0.85^3 = 0.61$$

Power Dissipation





Use Multi-core CPUs



Which Programs

- Execution time of what program?
- Best case – your always run the same set of programs
  - Port them and time the whole workload
- In reality, use **benchmarks**
  - Programs chosen to measure performance
  - Predict performance of actual workload
  - Saves effort and money
  - Representative? Honest?

Benchmarks: SPEC2000

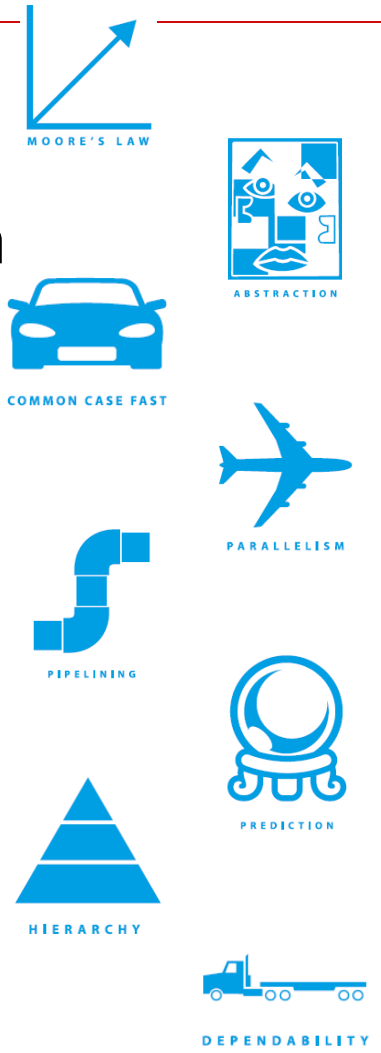
- System Performance Evaluation Cooperative
  - Formed in 80s to combat benchmarketing
  - SPEC89, SPEC92, SPEC95, now SPEC2000
- 12 integer and 14 floating-point programs
  - Sun Ultra-5 300MHz reference machine has score of 100
  - Report GM of ratios to reference machine

Benchmarks: SPEC 2000

12 Integer benchmarks (C and C++)		14 FP benchmarks (Fortran 77, 90, and C)	
Name	Description	Name	Description
gzip	Compression	wupwise	Quantum chromodynamics
vpr	FPGA placement and routing	swim	Shallow water model
gcc	GNU C compiler	mgrid	Multigrid solver in 3D potential field
mcf	Combinatorial optimization	applu	Partial differential equation
crafty	Chess program	mesa	Three-dimensional graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Neural networks image recognition
perlbmk	Perl application	equake	Seismic wave propagation simulation
gap	Group theory, interpreter	facerec	Image recognition of faces
vortex	Object-oriented database	ammp	Computational chemistry
bzip2	Compression	lucas	Primality testing
twolf	Place and route simulator	fma3d	Crash simulation using finite elements
		sixtrack	High-energy nuclear physics
		apsi	Meteorology: pollutant distribution

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




Questions