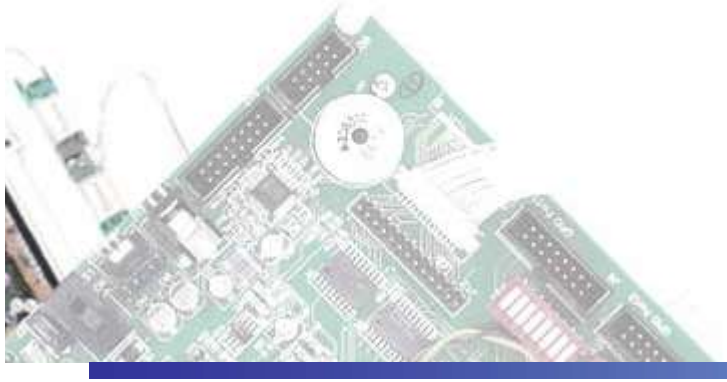
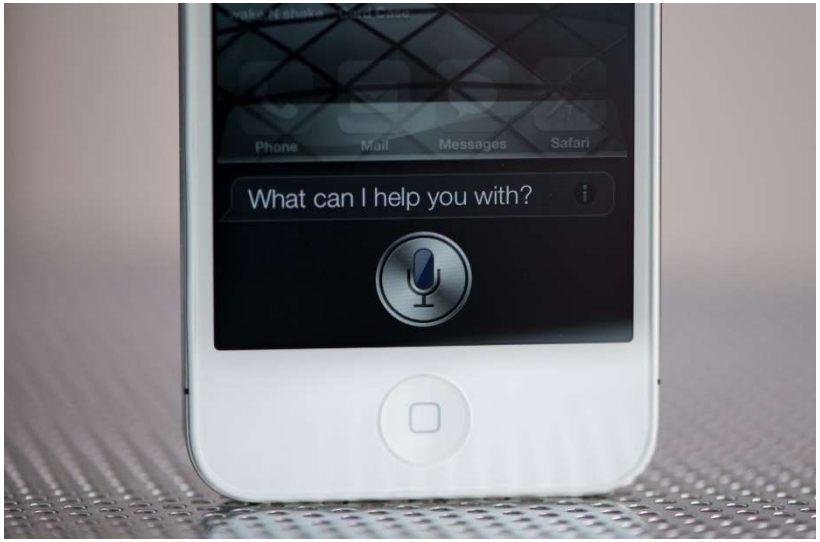
Computer Architecture

Chapter 1

Computer Abstractions & Technology



Why Computer Organization



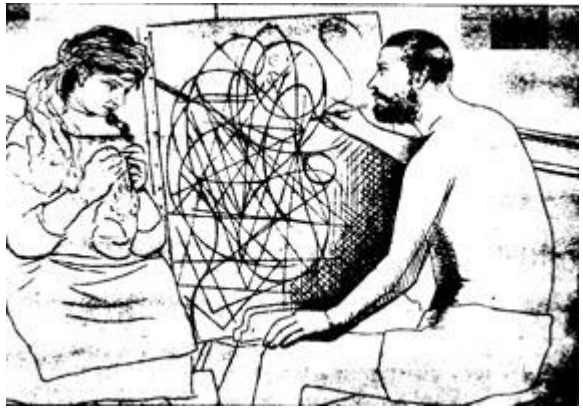
Why Computer Organization

- Embarrassing if you are a BS in CS/CE and can't make sense of the following terms: DRAM, pipelining, cache hierarchies, I/O, virtual memory, …
- Embarrassing if you are a BS in CS/CE and can't decide which processor to buy: 3 GHz P4 or 2.5 GHz Athlon (helps us reason about performance/power), …
- Obvious first step for chip designers, compiler/OS writers
- Will knowledge of the hardware help you write better programs?

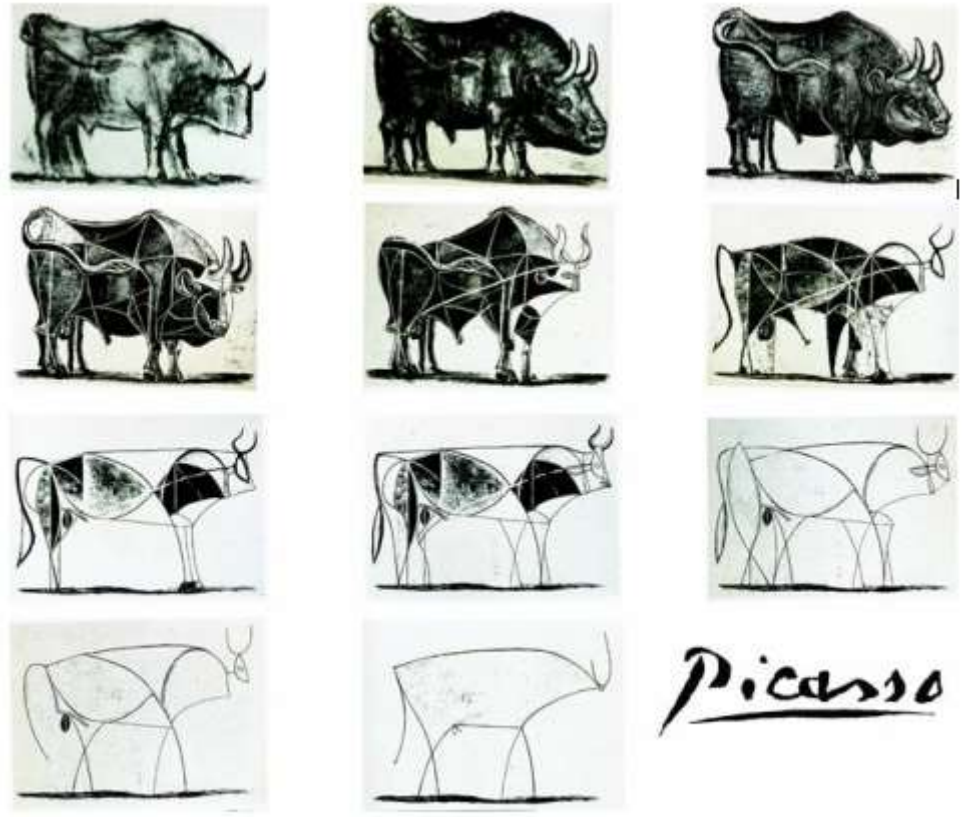
Must a Programmer Care About Hardware?

- Must know how to reason about program performance and energy
- Memory management: if we understand how/where data is placed, we can help ensure that relevant data is nearby
- Thread management: if we understand how threads interact, we can write smarter multi-threaded programs
- Why do we care about multi-threaded programs?

Abstraction(추상化) (1)



<화가와 모델>, 파블로 피카소 작, 1932



Pablo Picasso, Bull (plates I - XI) 1945

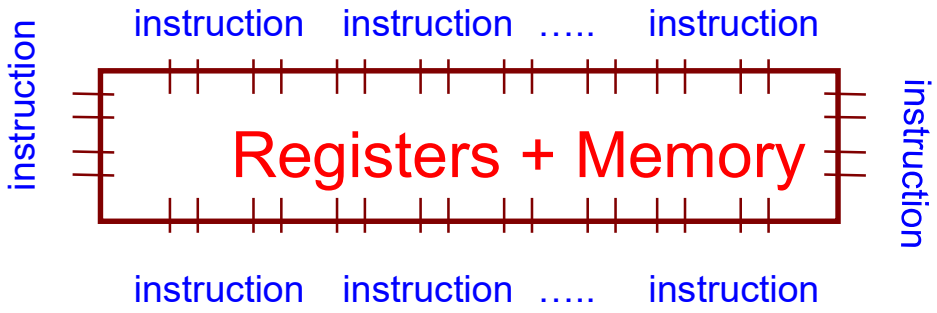
Abstraction (2)

- (Before)

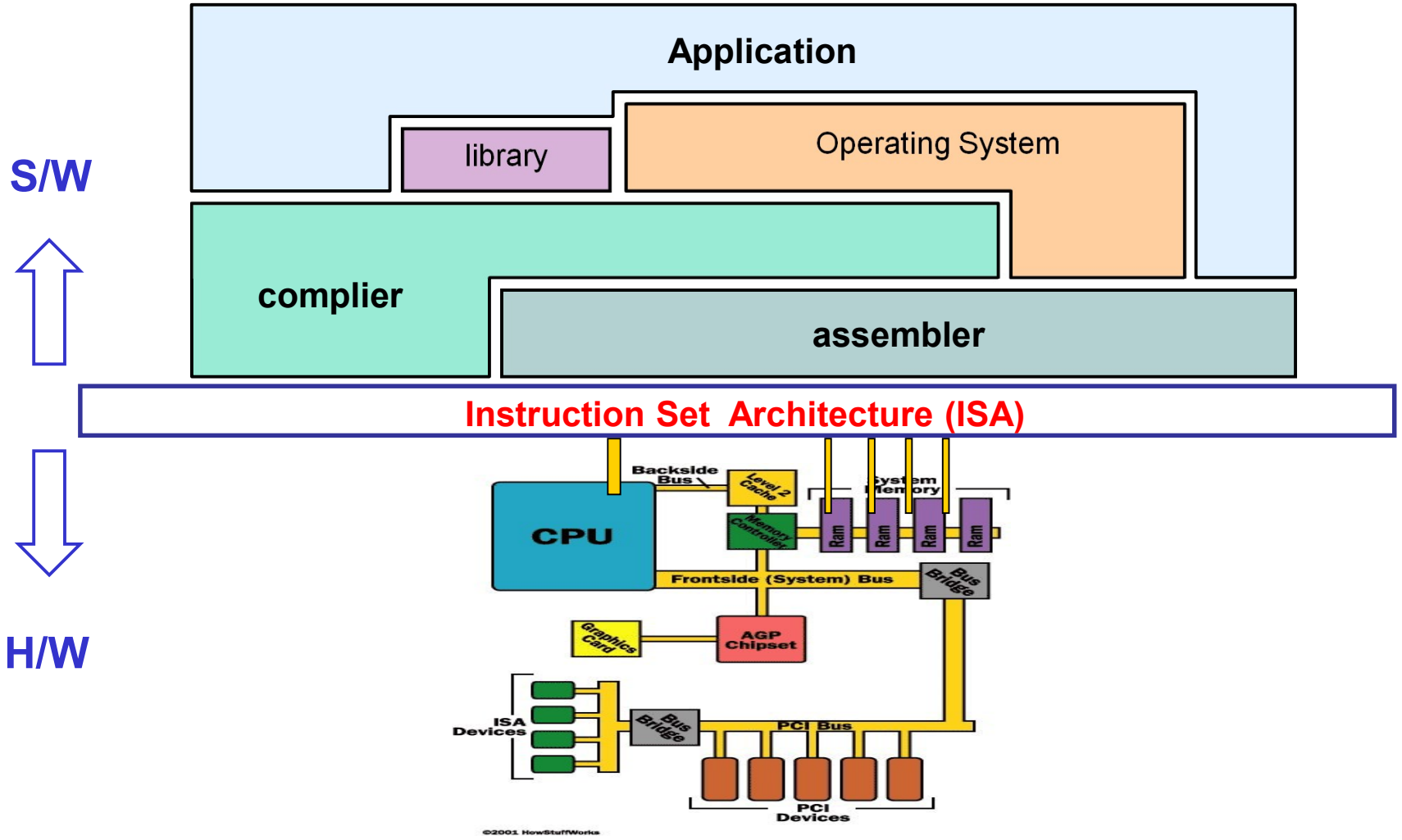
(After)

Abstraction (3)

- Abstraction in Computer Architecture
  - hide lower-level details of computer systems (시스템의 자세한 사항들을 숨김으로)
  - in order to facilitate design of sophisticated systems (복잡한 시스템 설계를 쉽게)
- Instruction set architecture(ISA) (=**Architecture**)
  - Interface between hardware and lower-level software
  - programmer가 보는(직접 다루는) computer system의 속성(형태)
    - conceptual structure (state) and functional behavior (operation)
    - as distinct from the organization of the data flows and controls, the logic design, and the physical implementation



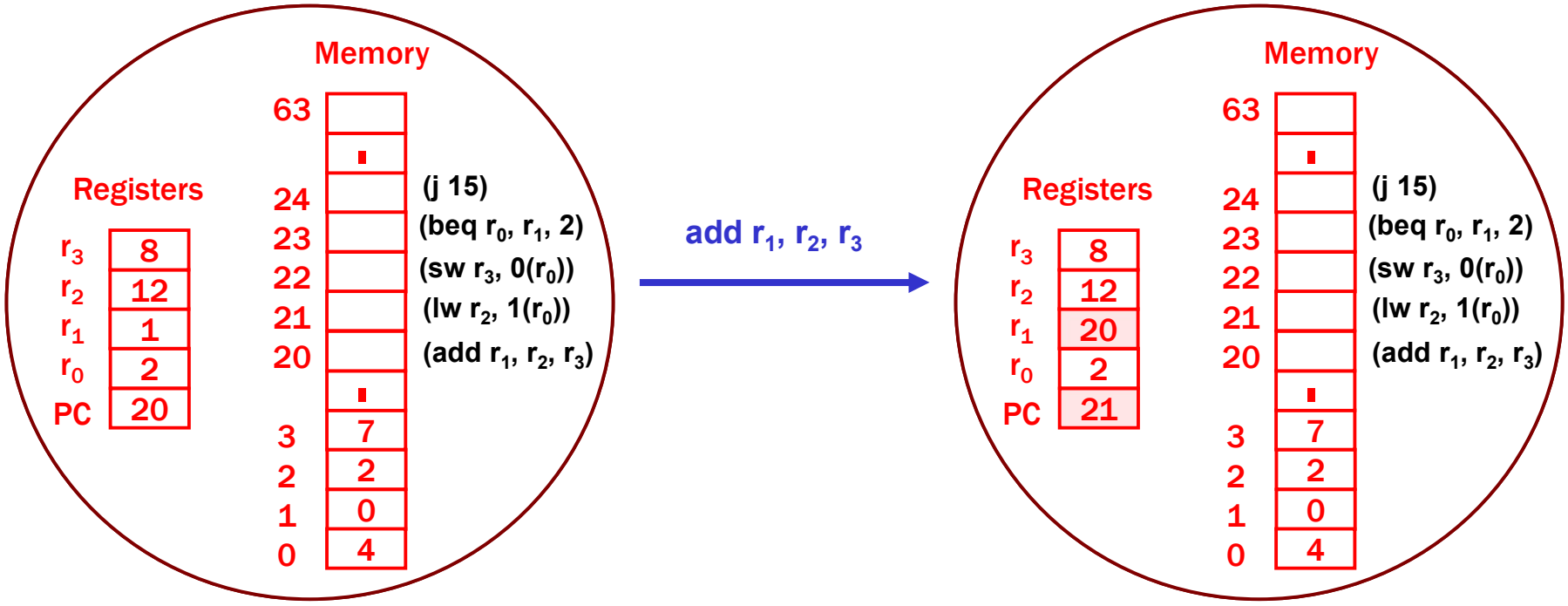
Instruction Set Architecture (ISA)



ISA Example (1)

Assumptions

- 8 bit ISA
- # of registers = 4 + PC (Program Counter)
- Memory size = 64B

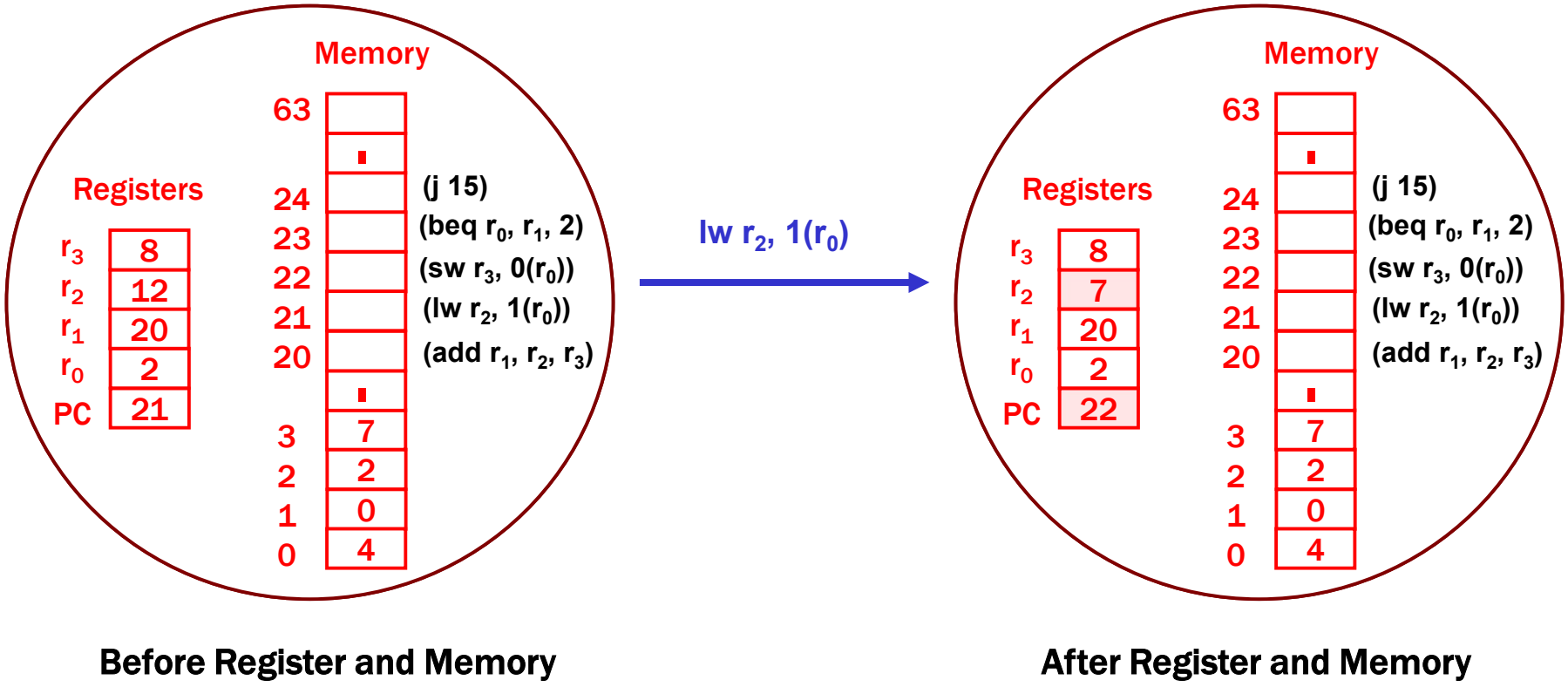


Before Register and Memory

After Register and Memory

ISA Example (2)

Assumptions
- 8 bit ISA
- # of registers = 4 + PC (Program Counter)
- Memory size = 64B

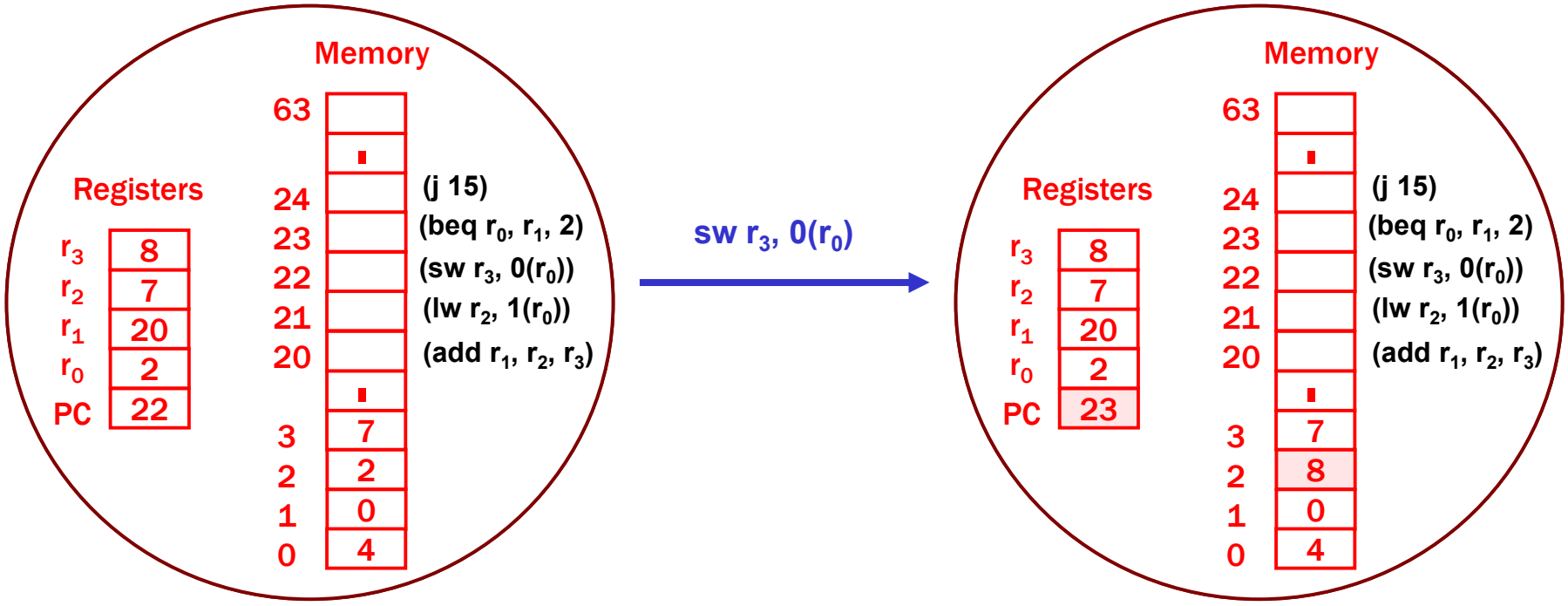


Before Register and Memory

After Register and Memory

ISA Example (3)

Assumptions
- 8 bit ISA
- # of registers = 4 + PC (Program Counter)
- Memory size = 64B

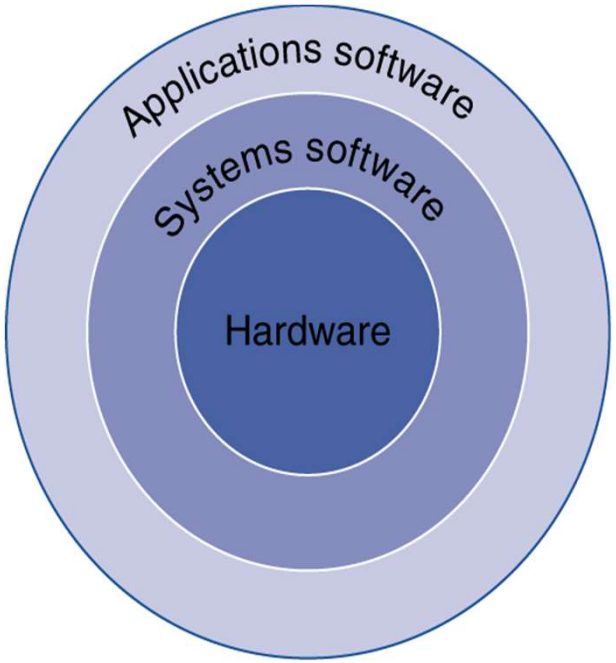


Before Register and Memory

After Register and Memory

Below Your Program

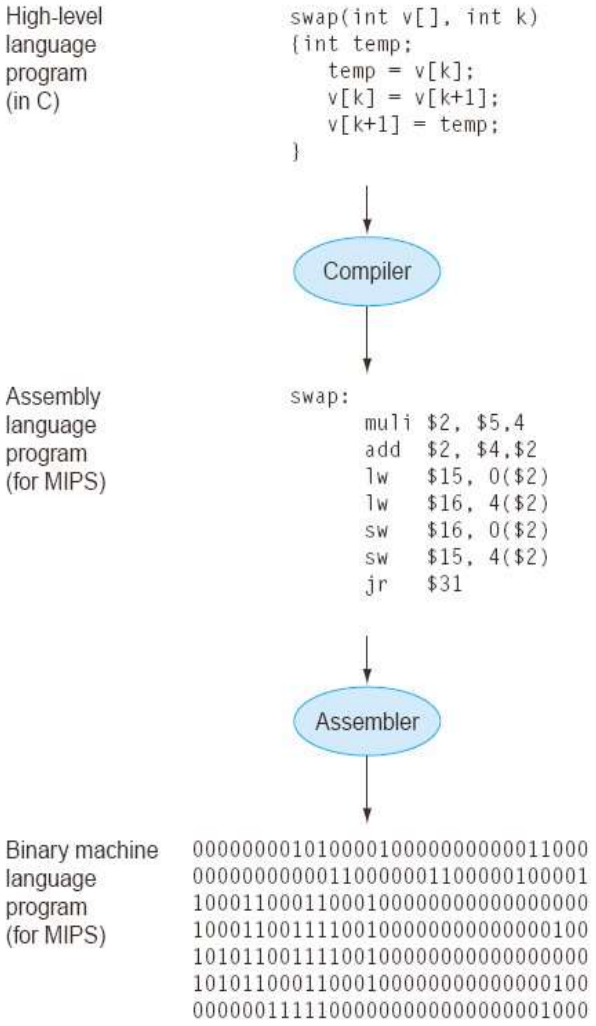
- Application software
  - Written in high-level language (HLL)
- System software
  - Compiler
    - translates HLL code to machine code (thru assembly code)
  - Operating System (system service code)
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources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



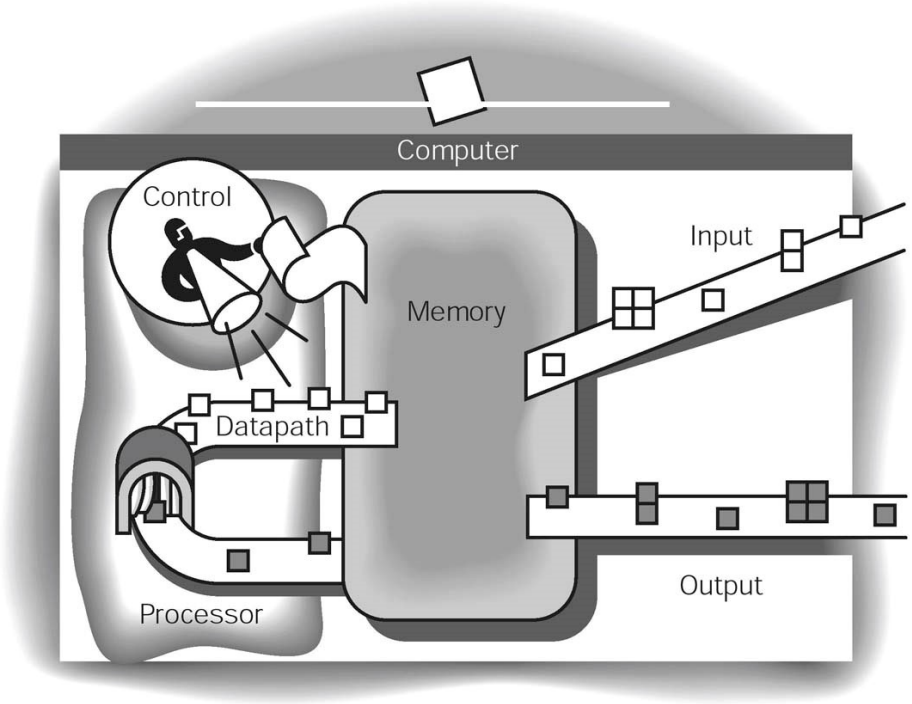
Levels of Program Code

- High-level language such as C
  - Level of abstraction closer to problem domain (문제 영역에 더 가까운 표현)
  - Provides for productivity and portability (생산성과 이식성)
- Assembly language
  - Textual representation of machine instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data which machine can understand

* Everything is represented in binary digits in a computer.



Computer System Inside



- Classic 5 Components
  - Processor = Datapath + Control
  - Memory
  - Input & Output
- Input/output includes
  - User-interface devices
    - display, keyboard, mouse
  - Storage devices
    - HDD, ODD, Flash
  - Network adapters
    - for communicating with other computers

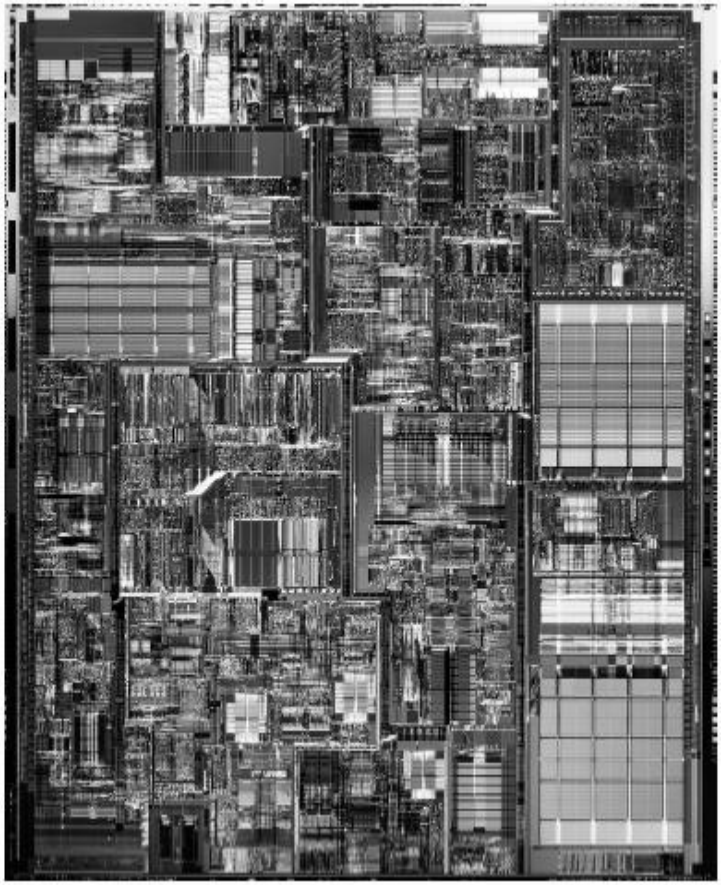
Inside the Processor (CPU)

- Datapath
  - performs operations on data
  - ALU, registers, internal buses
- Control
  - sequences(controls) the datapath, memory, and bus operations...

cf. cache memory

- called on-chip cache or level-1 cache
- small fast SRAM memory for immediate access to data

Processor Inside (Pentium4)



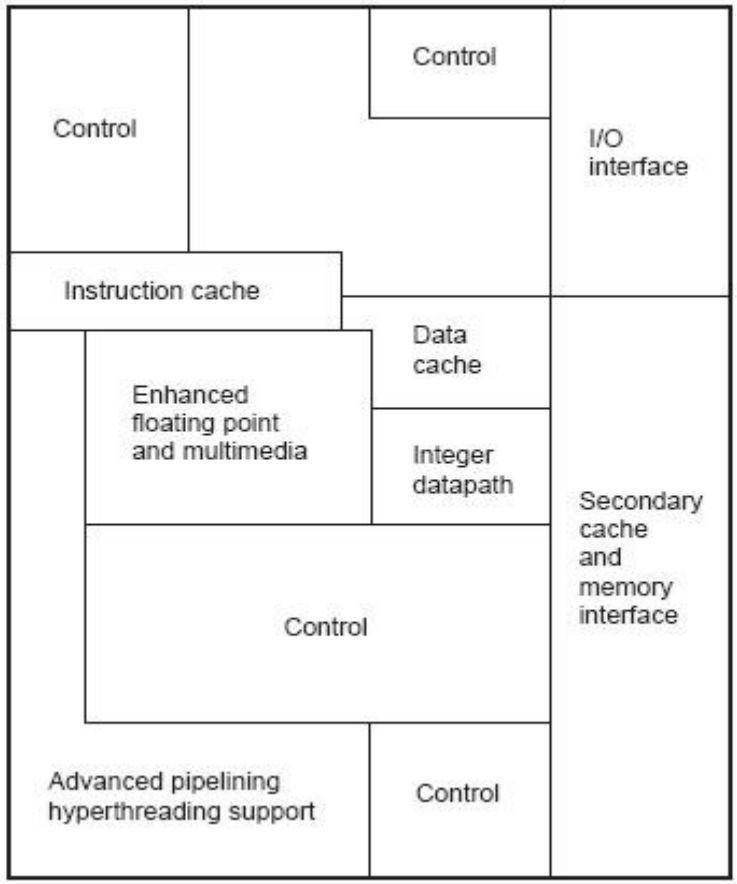
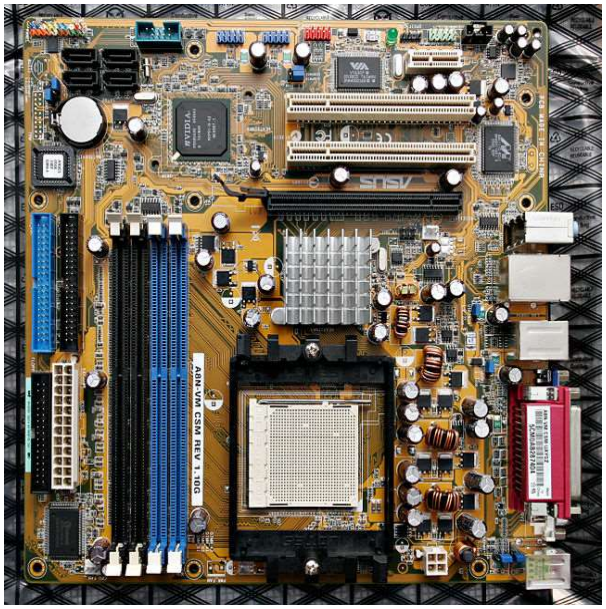
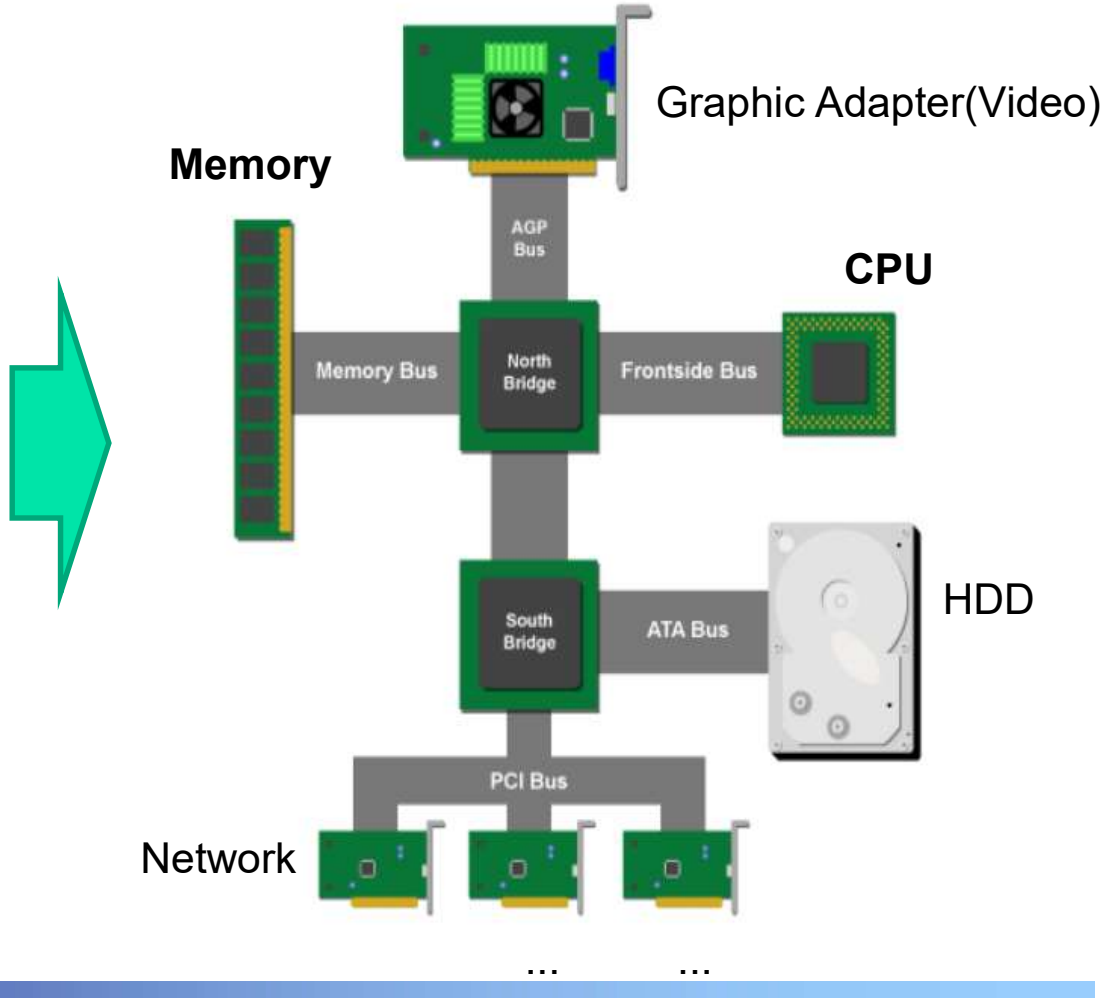


Figure 1.9

Close-up of PC Motherboard

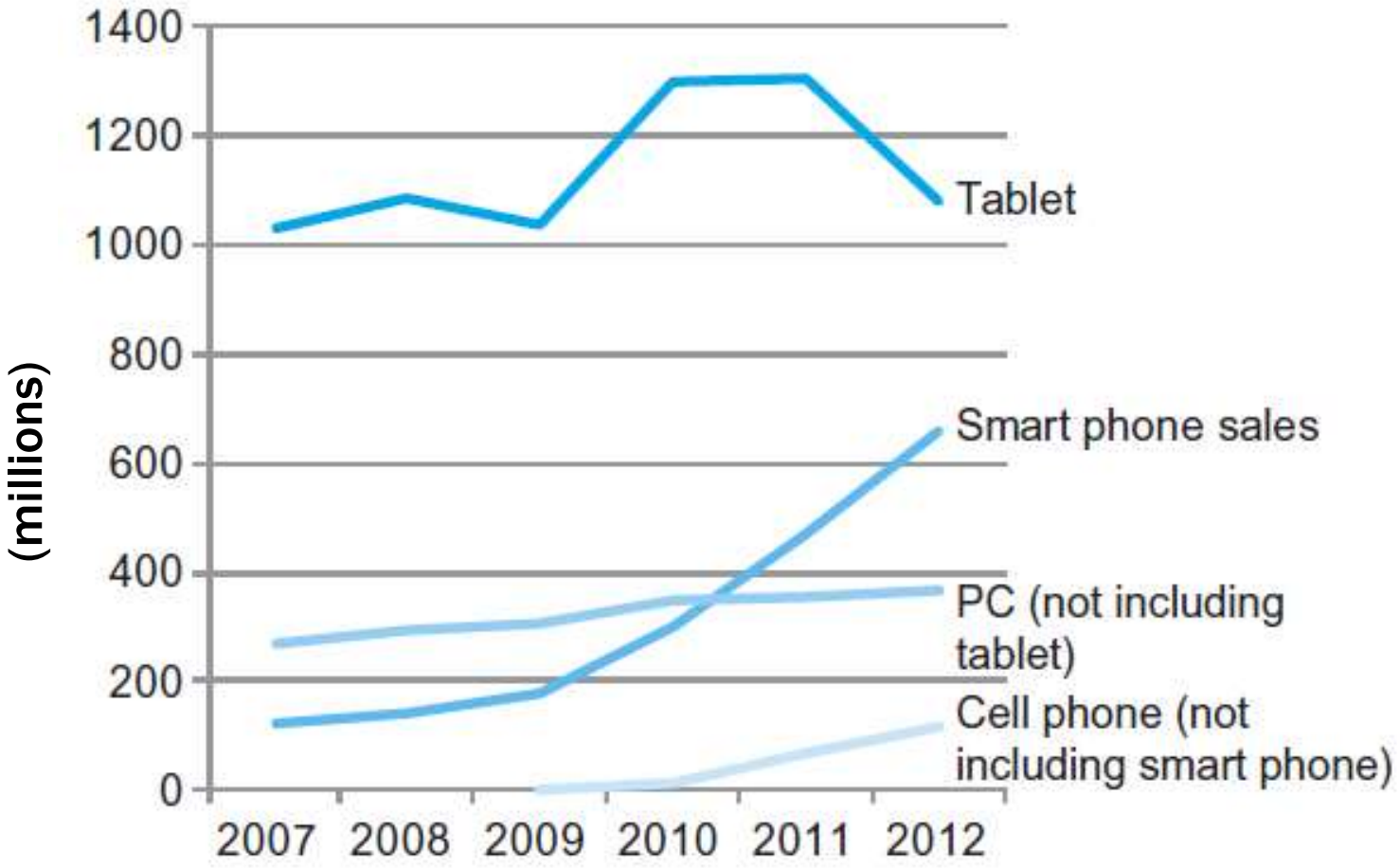




Classes of Computers

- **Personal computers**
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- **Server computers**
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
- **Supercomputers**
  - High-end scientific and engineering calculations
  - Highest capability but represent a small fraction of the overall computer market
- **Embedded computers**
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

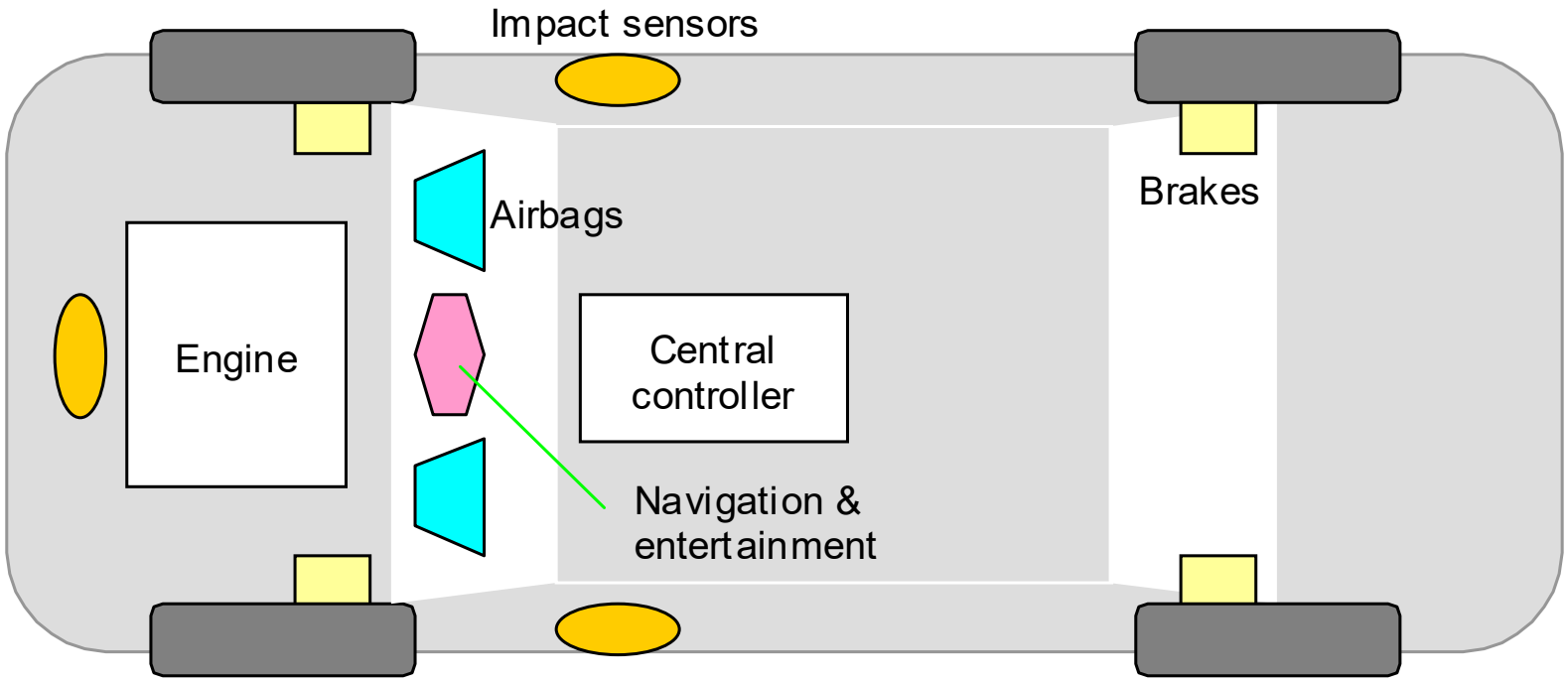
The PostPC Era (1)



The PostPC Era (2)

- Personal Mobile Device (PMD)
  - Battery operated
  - Connects to the Internet
  - Hundreds of dollars
  - Smart phones, tablets, electronic glasses
- Cloud computing
  - Warehouse Scale Computers (WSC)
  - Software as a Service (SaaS)
  - Portion of software run on a PMD and a portion run in the Cloud
  - Amazon and Google

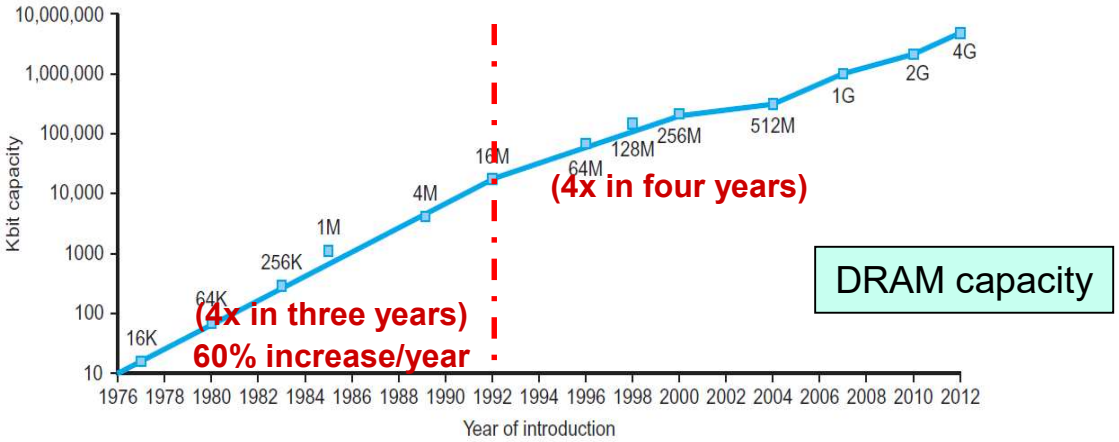
Embedded Processing Example



Technology Trends (1)

1965	1977	1998	2005
			
IBM System 360/50	DEC VAX 11/780	Apple iMac	Pentium4
0.15 MIPS 64 KB $1M $6.6M per MIPS $16M per MB	1 MIPS(peak) 0.5 MIPS(estimated) 1 MB $200K $200K to $400 per MIPS $200K per MB	700 MIPS(peak) 427 MIPS(estimated) 32 MB $1229(September 1998) $1.75 to $2.90 per MIPS $38 per MB	~15000 MIPS(peak) ~6000 MIPS(estimated) 512 MB < $1000 $0.07 to $0.17 per MIPS < $2 per MB

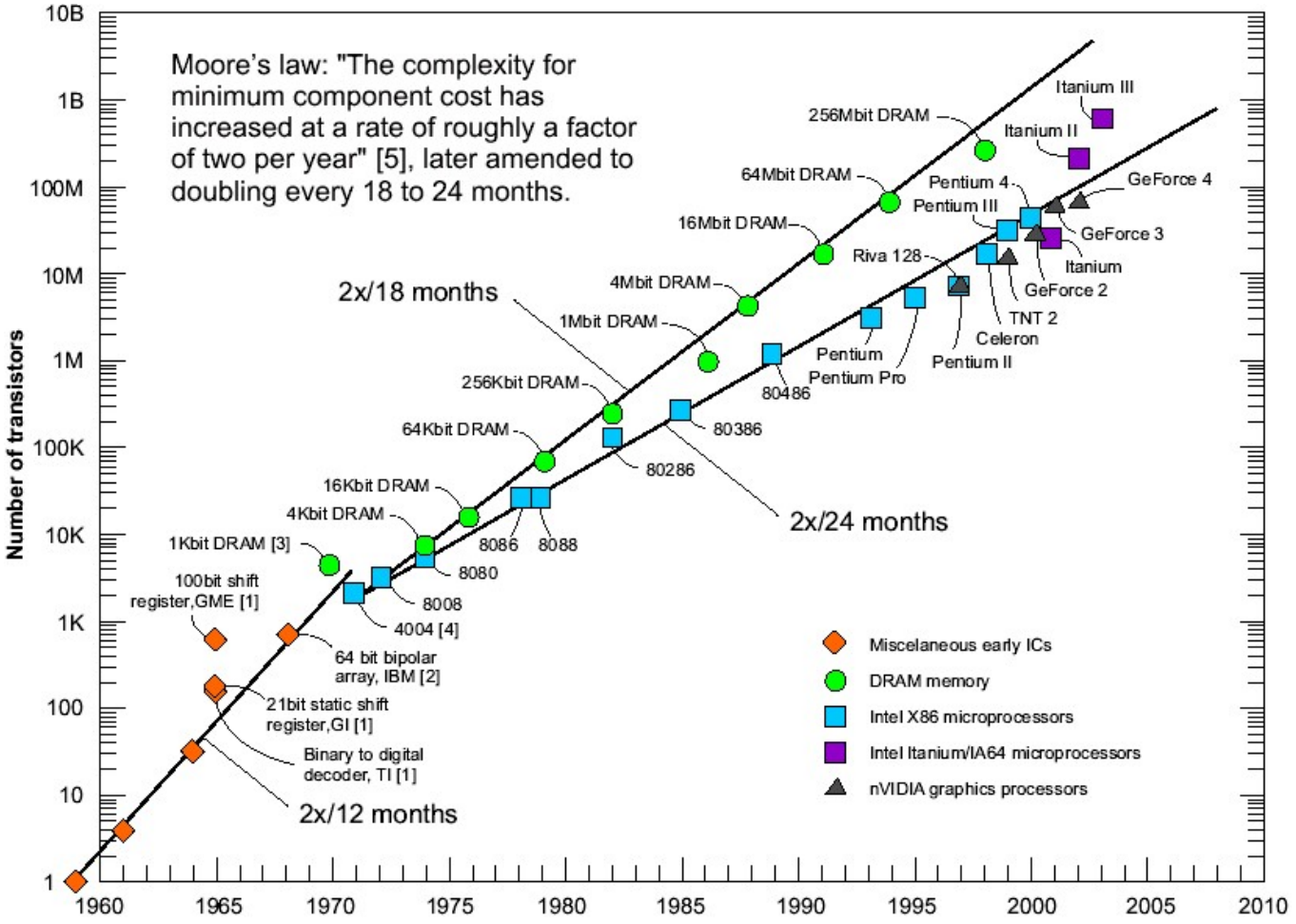
Technology Trends (2)



year	size	cycle time
1980	64 Kbits	250 ns
1983	256 Kbits	220 ns
1986	1 Mbits	190 ns
1989	4 Mbits	165 ns
1992	16 Mbits	145 ns
1996	64 Mbits	125 ns
2000	256 Mbits	100 ns

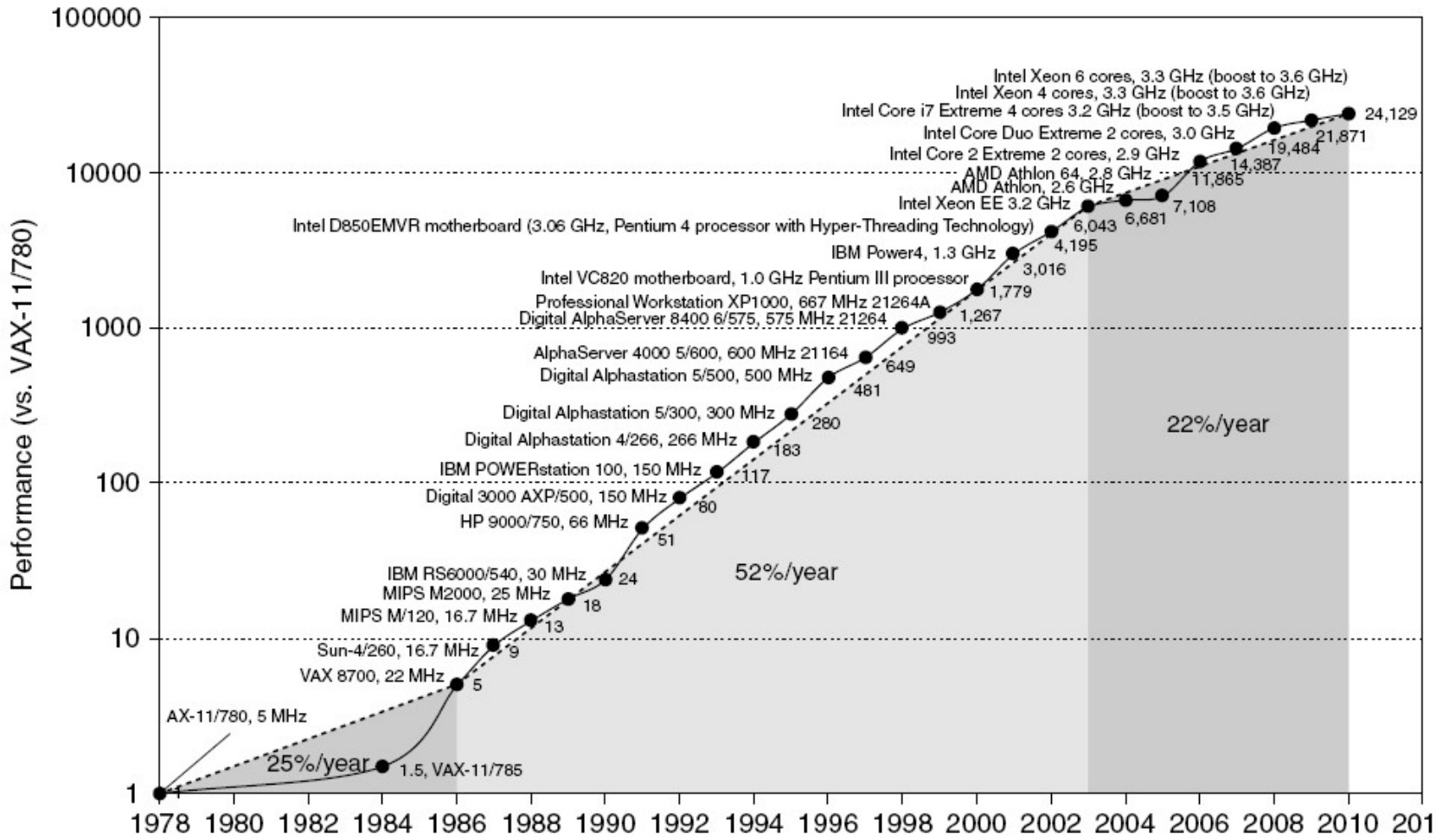
Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2013	Ultra large scale IC	250,000,000,000

Transistors Per Die Trends

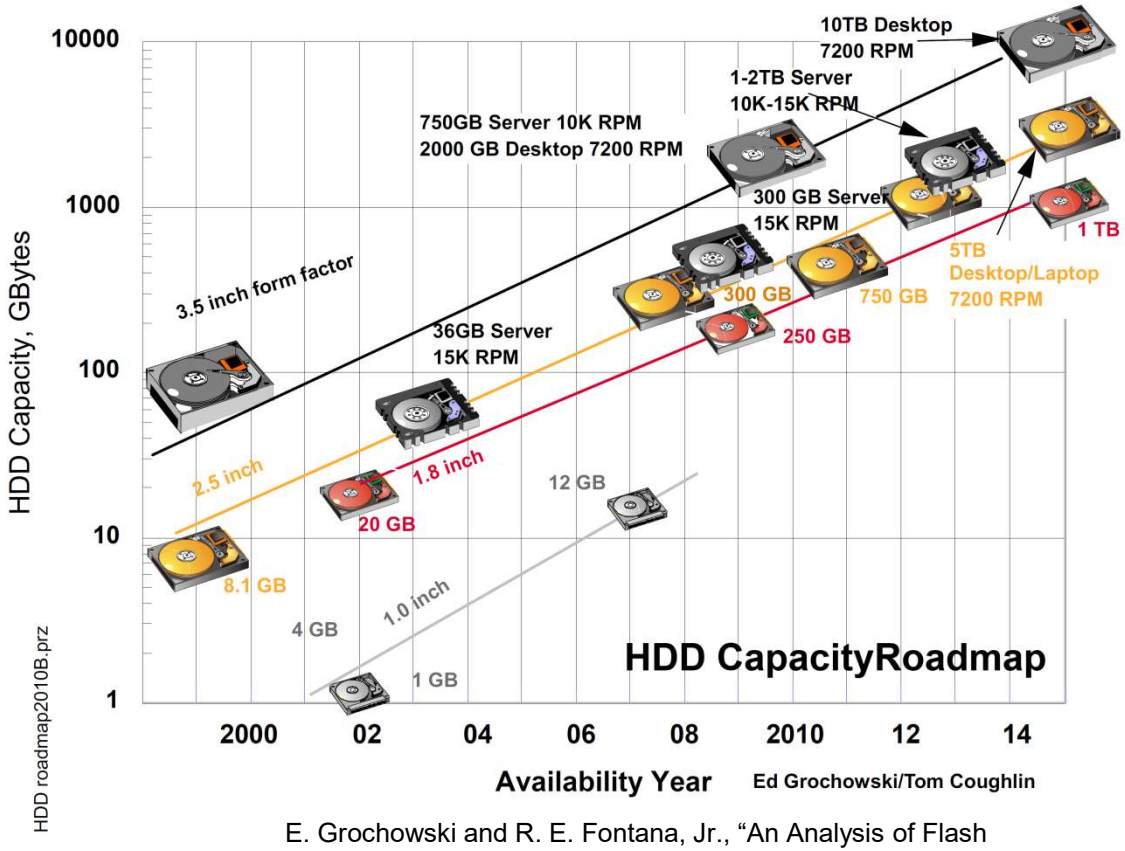


Source: www.icknowledge.com

Processor Performance Trends



Hard Disk Drive (HDD) Technology Trends

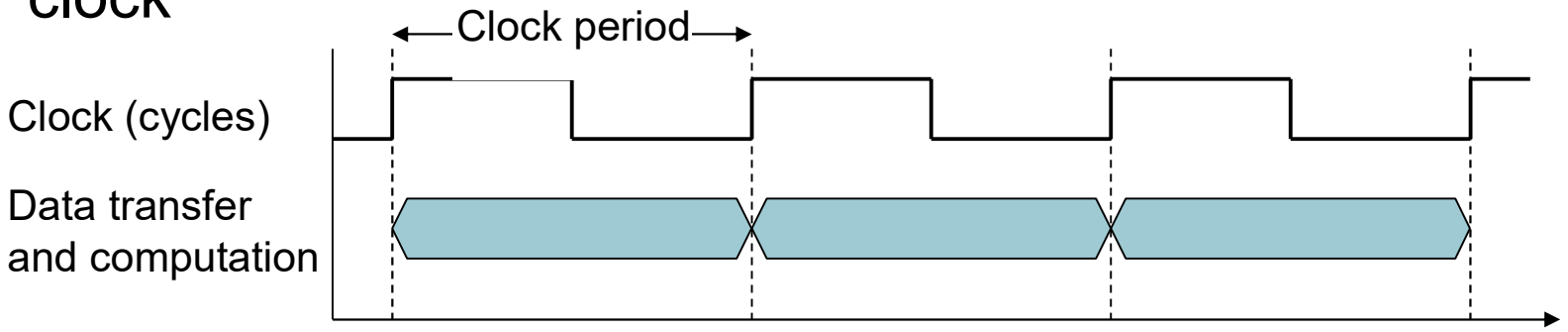


E. Grochowski and R. E. Fontana, Jr., "An Analysis of Flash and HDD Technology Trends," Flash Memory Summit 2011.

Disk density: 1.50x - 1.60x per year (4x in three years)

CPU Clocking

- Digital hardware operations governed by a constant-rate clock



  - Clock period: duration of a clock cycle
    - e.g., 250ps = 0.25ns = 250×10^{-12}s
  - Clock frequency (rate): cycles per second
    - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Performance and Its Factors

- CPU execution time

  = CPU clock cycles x clock cycle time

  = CPU clock cycles / clock rate

- Example:
  - Same instruction sets
  - Computer A : 4 GHz, 10 seconds
  - Computer B : ? GHz, 6 second
  - B requires 1.2 times as many clock cycles as A.

[Answer]

$\text{CPU time}_A = \text{CPU clock cycles}_A / \text{clock rate}_A$

$10 \text{ seconds} = \text{CPU clock cycles}_A / (4 \times 10^9 \text{ cycles/sec})$

$\text{CPU clock cycles}_A = 10 \text{ sec.} \times 4 \times 10^9 \text{ cycles/sec}$

$= 40 \times 10^9 \text{ cycles}$

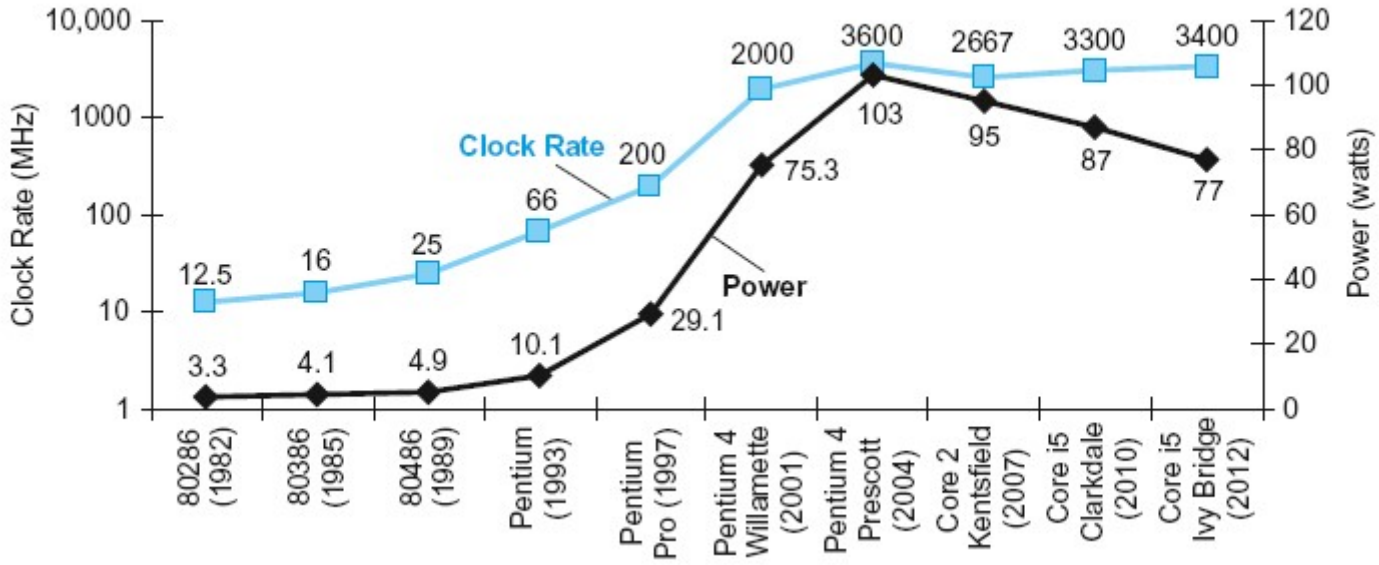
$\text{CPU time}_B = \text{CPU clock cycles}_B / \text{clock rate}_B$

$= 1.2 \times \text{CPU clock cycles}_A / \text{clock rate}_B$

$6 \text{ seconds} = 1.2 \times 40 \times 10^9 \text{ cycles} / \text{clock rate}_B$

$\therefore \text{clock rate}_B = 1.2 \times 40 \times 10^9 \text{ cycles} / 6 \text{ seconds} = 8 \text{ GHz}$

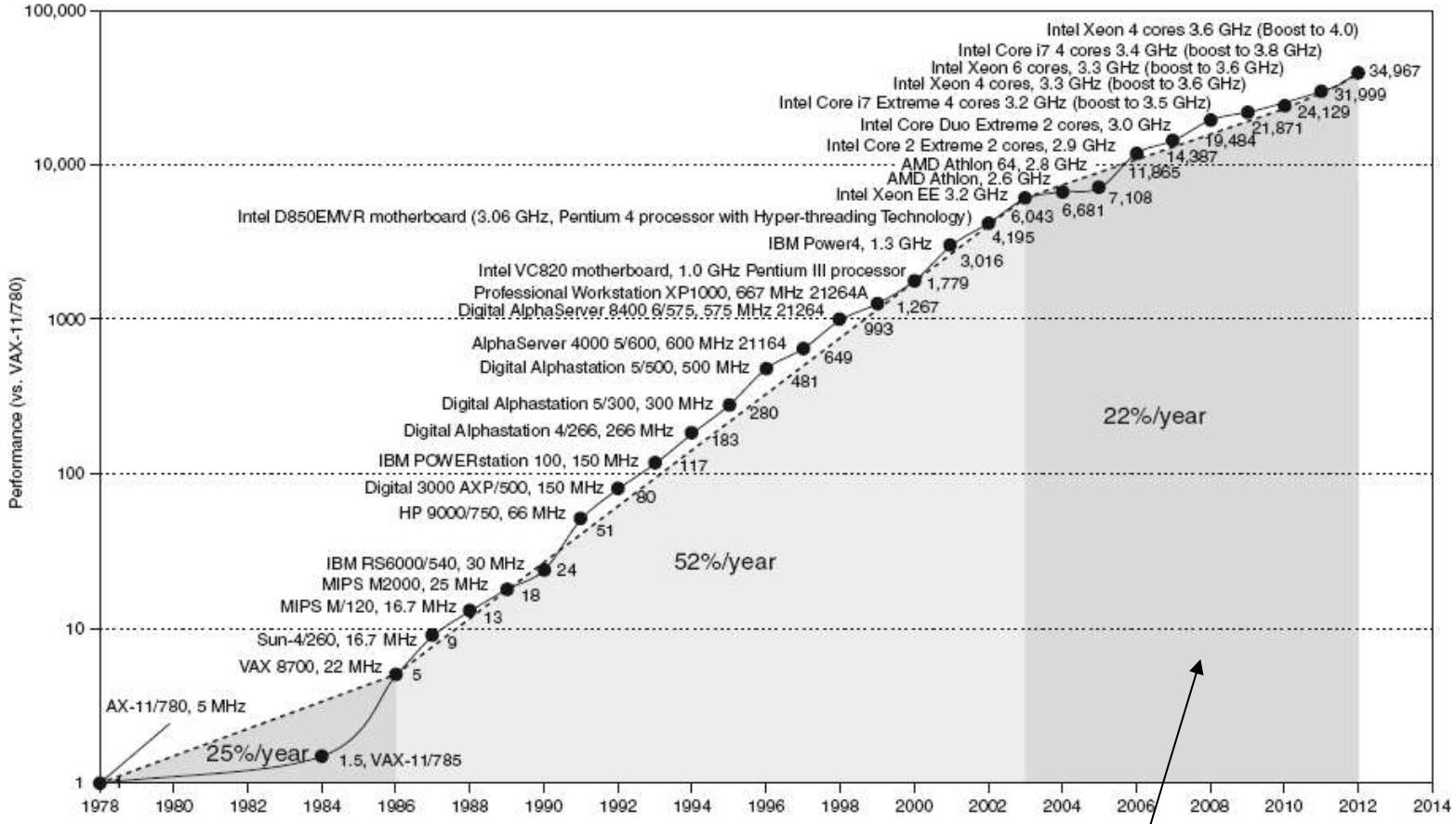
Power Trends



- In CMOS IC technology

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

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
- Requires explicitly parallel programming
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization

For Good Performance

- Algorithm
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation
- Processor and memory system
  - Determine how fast instructions are executed
- I/O system (including OS)
  - Determines how fast I/O operations are executed

Eight Great Ideas for Performance

- 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