Chapter 1

Computer Abstractions and Technology

The Computer Revolution

- Progress in computer technology
  - Underpinned by **Moore's Law**
  - Moore's law predicted that this trend will continue into the foreseeable future.
    - At least it has been the case for about 50 years

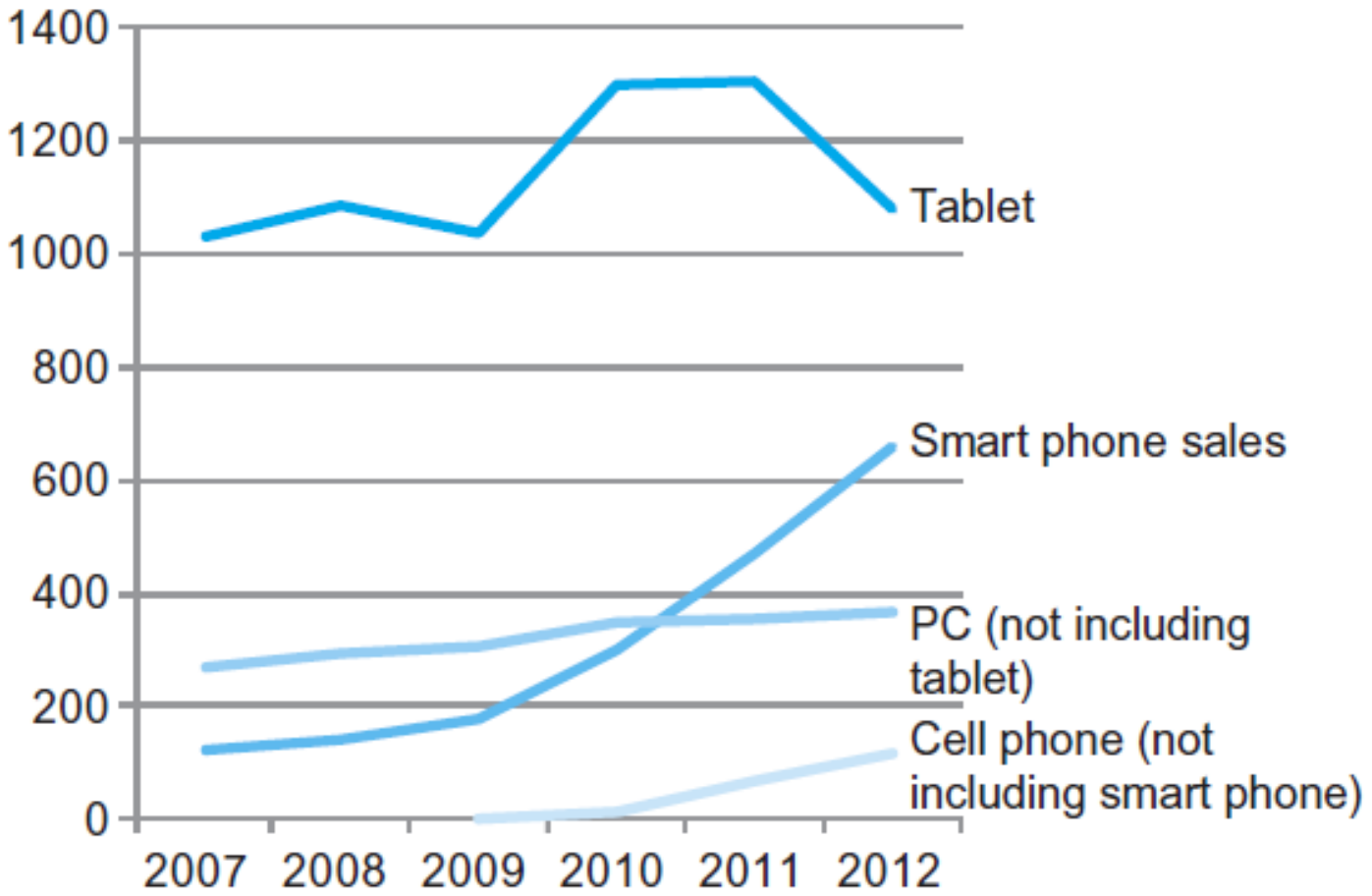
The Computer Revolution

- This progress has made novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- Computers are pervasive

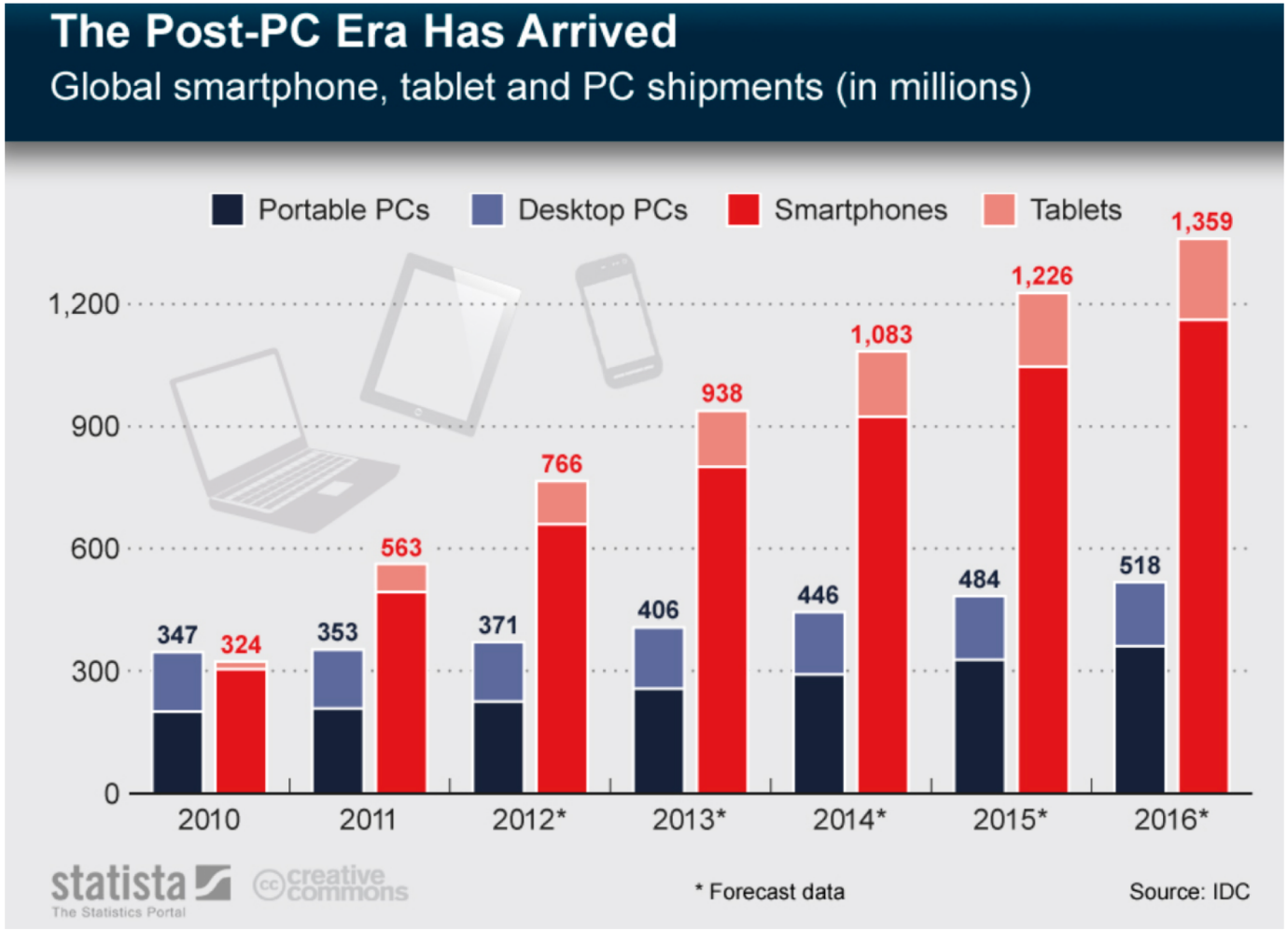
Classes of Computers

- Personal computers
- Server computers
- Supercomputers
- Embedded computers

The PostPC Era



The PostPC Era

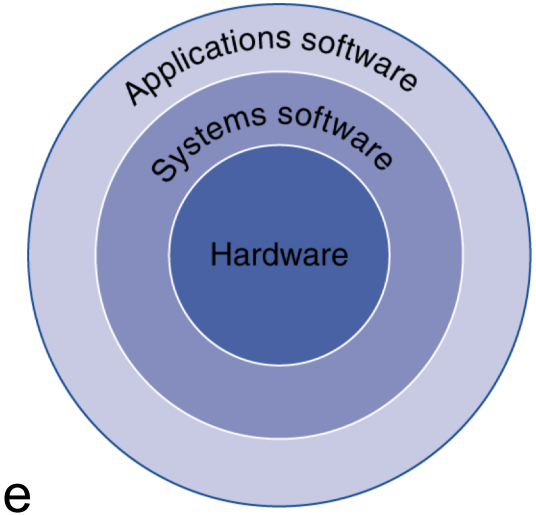


Understanding Performance

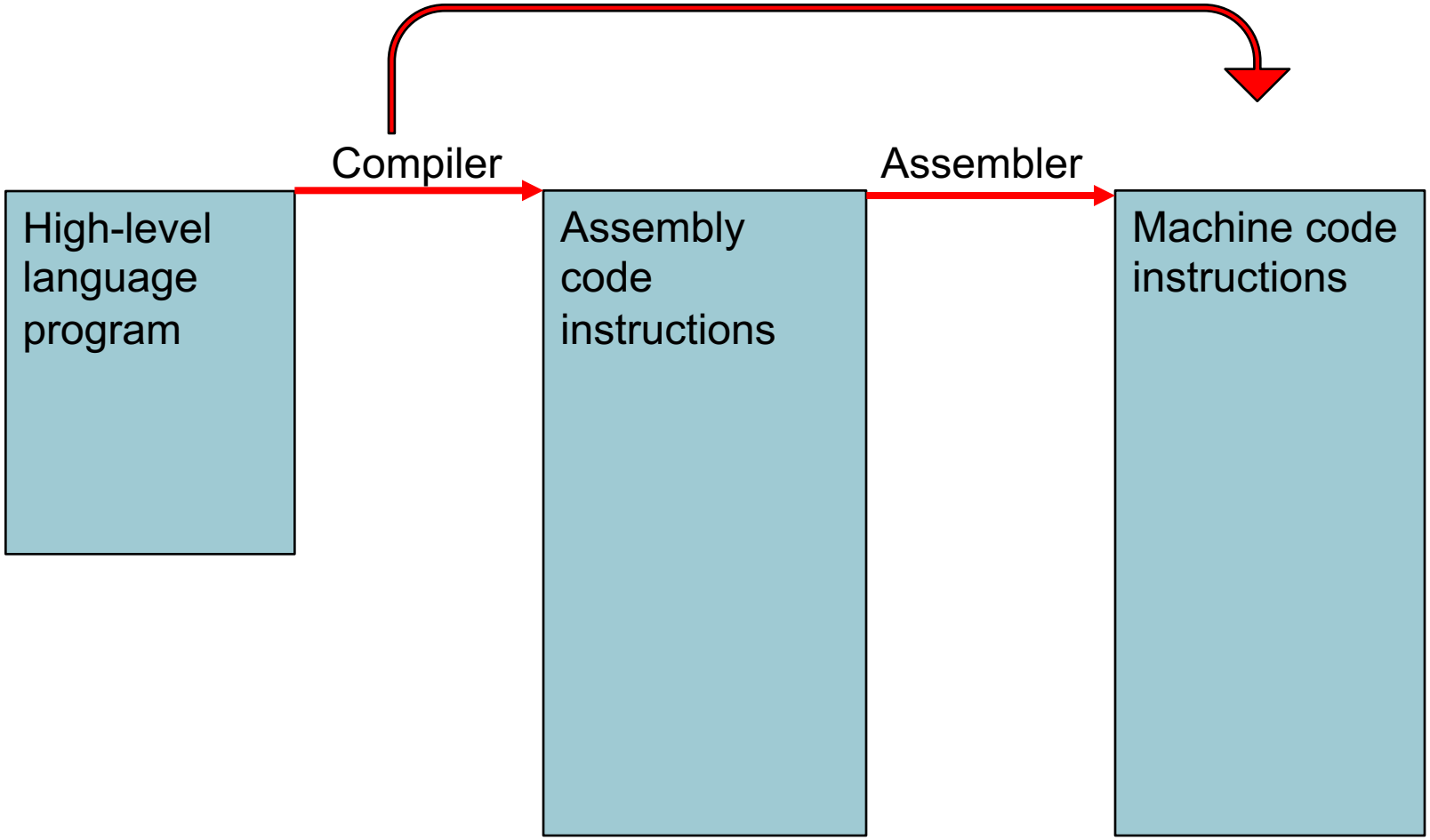
- What determines program performance?
  - And how can it be improved?
- Programs are translated into the machine language
  - And the hardware executes them
- So both HW and SW influence the performance
  - As well as the stuff interfacing them

Understanding Performance

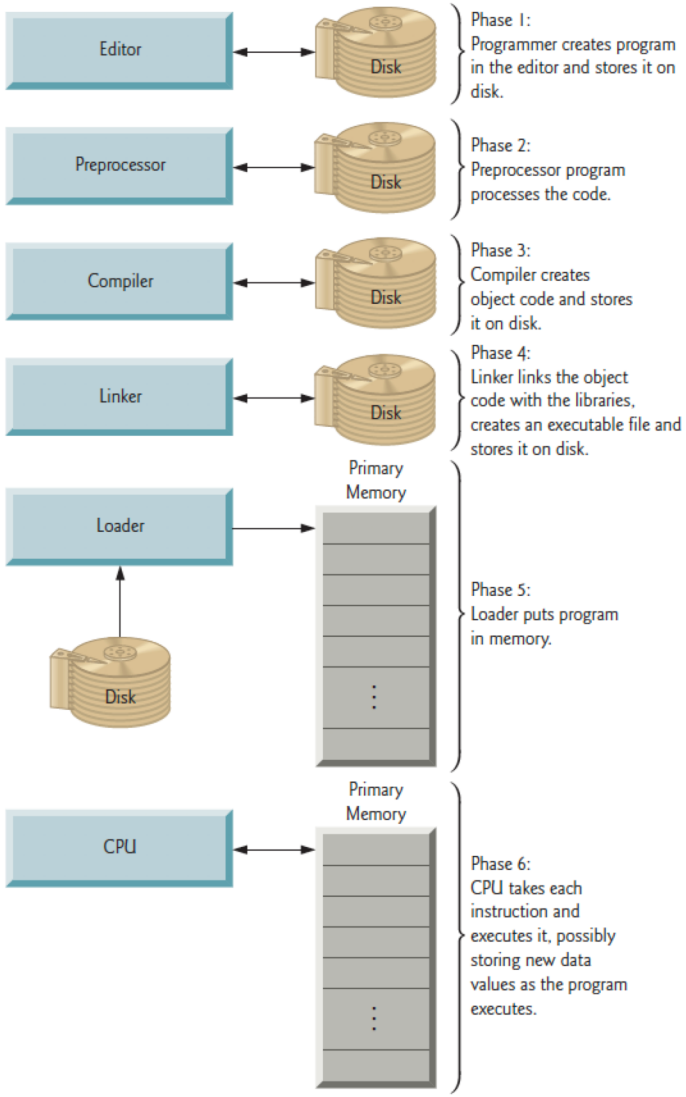
- Application software
  - Written in high-level language
- System software
  - Compiler: translates HLL code to machine code
    - Or HHL-> assembly -> machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources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



Understanding Performance



Understanding Performance



Understanding Performance

- Algorithm
- Programming language, compiler, architecture
- Processor and memory system
- I/O system (including OS)

Understanding Performance

Software

Program Requirements

↓

Algorithm

↓

Programming language used

↓

Operations to be executed

Compiler →

Hardware

Computer Architecture (ISA)

↓

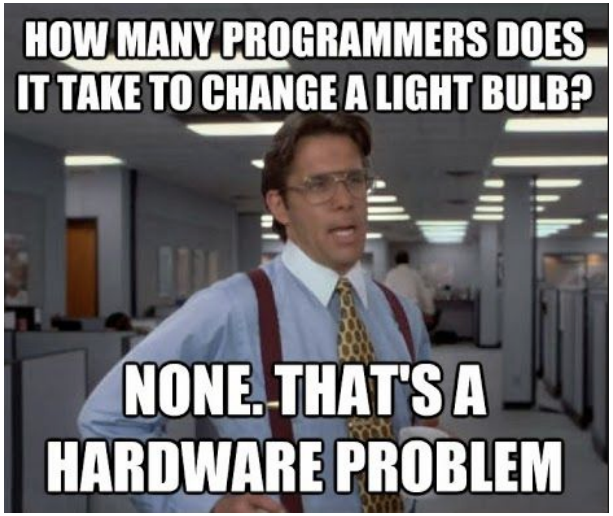
Total of machine Instructions

↓

Number of clock cycles to execute each instruction

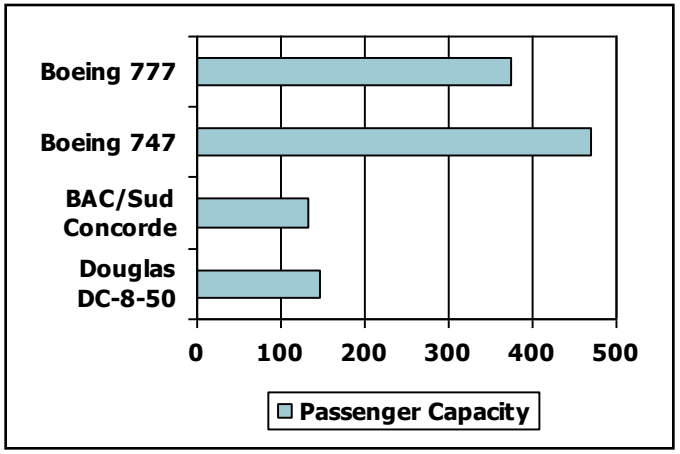
↓

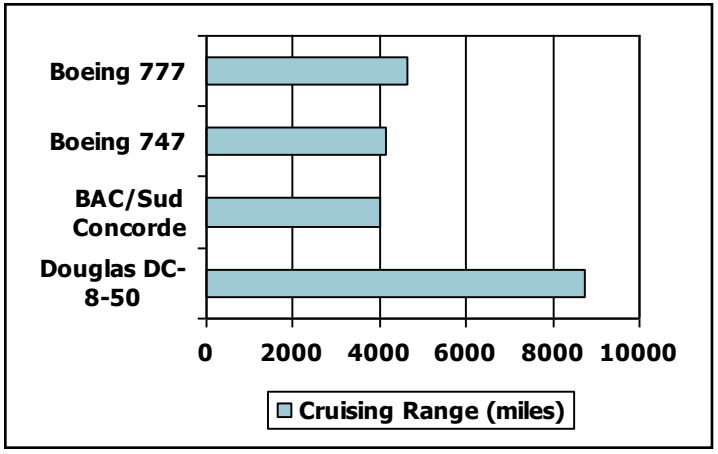
How fast and efficient are the memory and IO

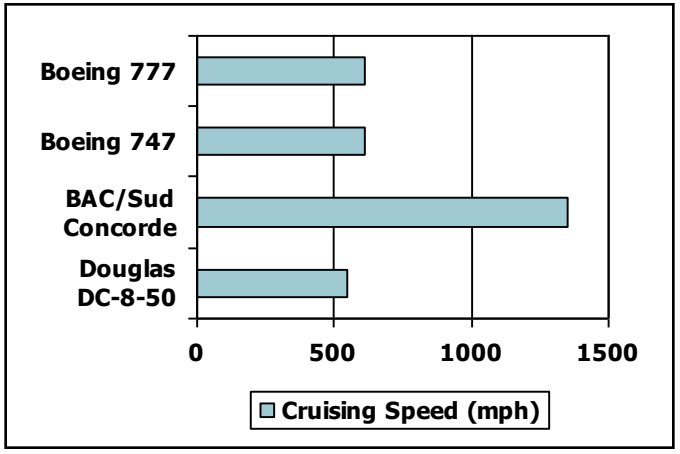


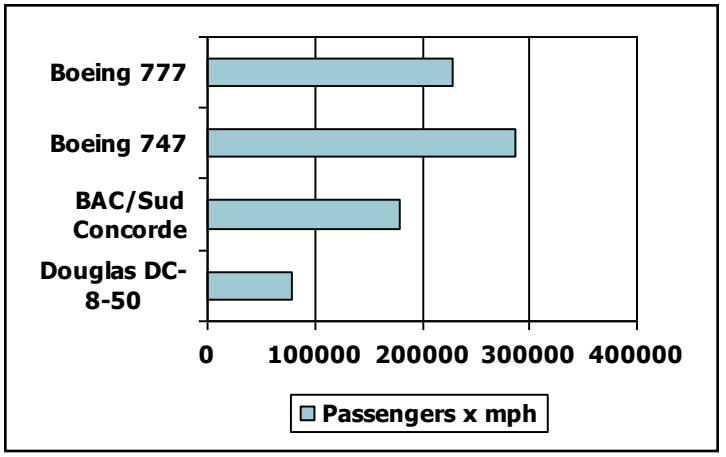
Defining Performance

- Which airplane has the best performance?









Relative Performance

- Define Performance = 1/Execution Time
- Relative performance:

$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Execution time}_Y / \text{Execution time}_X = n$$

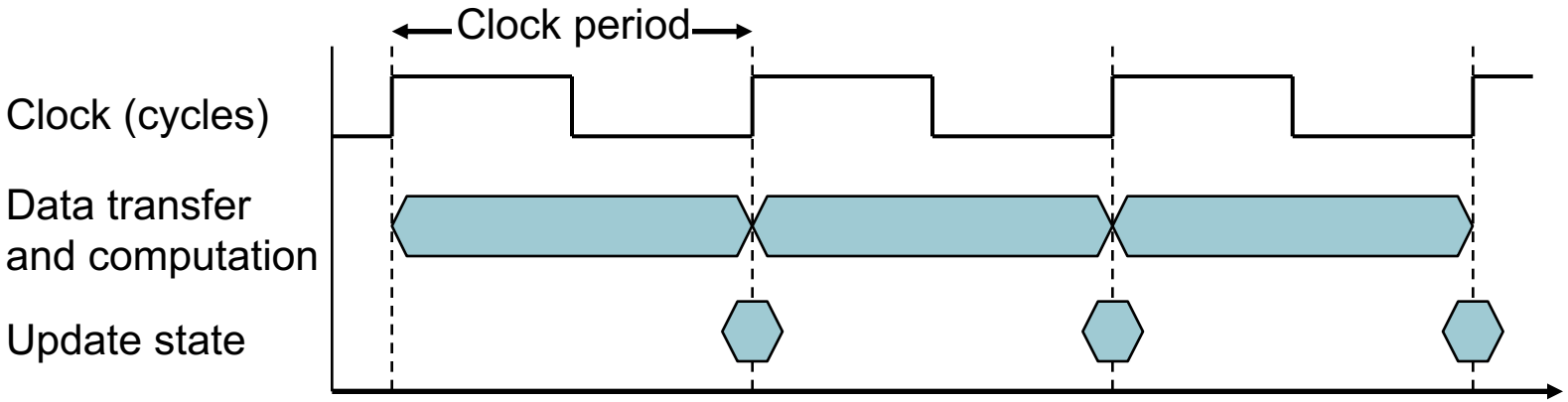
  - "X is n time faster than Y"
- Example: time taken to run a program
  - 10s on computer A, 15s on computer B
  - Execution Time_B / Execution Time_A = 15s / 10s = 1.5
    - So computer A is 1.5 times faster than computer B

Relative Performance - SPEC

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

CPU Clocking

- Operation of digital hardware 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 (#cycles/s)

Measuring Execution Time

- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - Determines CPU performance
- Different programs are affected differently by CPU and system performance

Relative Performance - SPEC

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

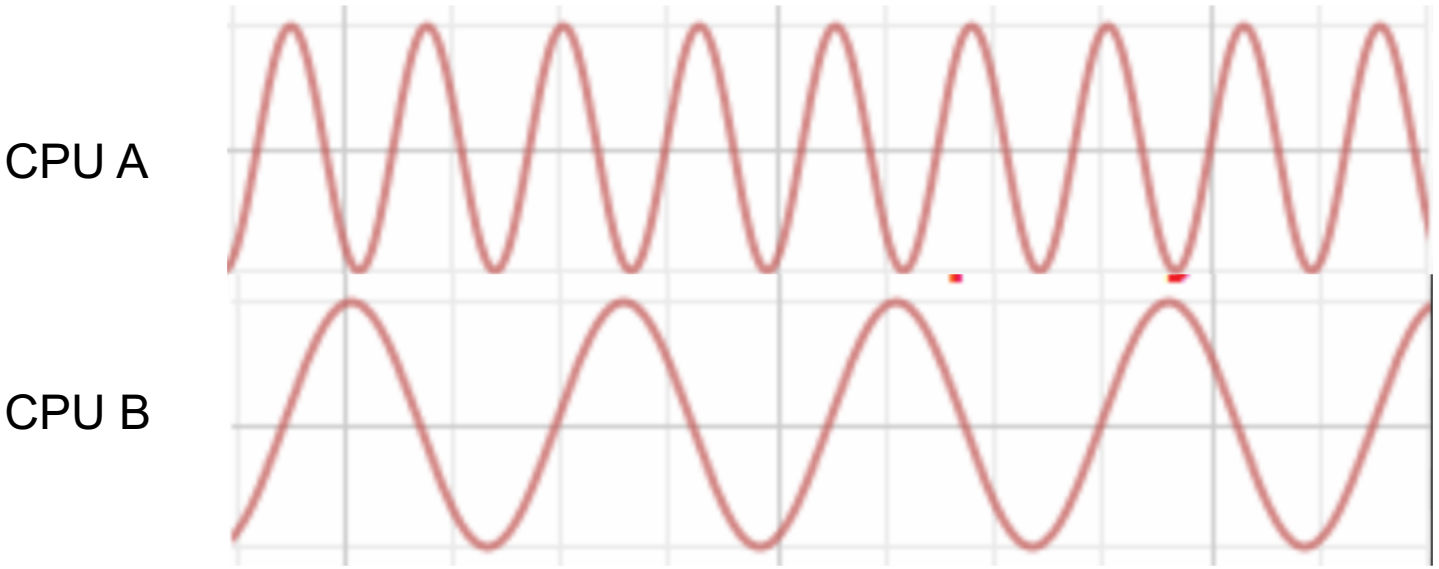
CPU Time

in CPU B

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

(Clock Period)

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

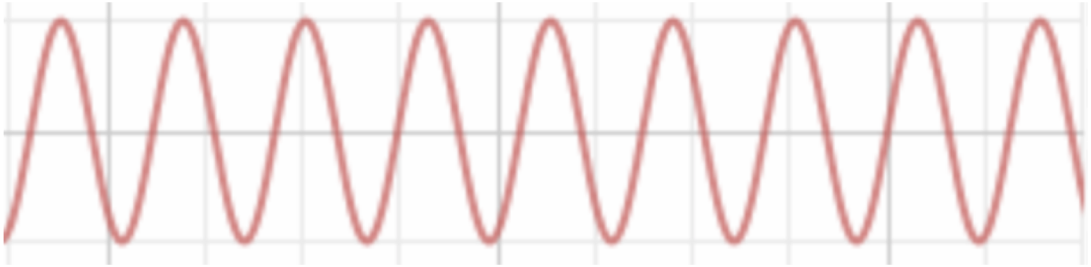


CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}} \quad \downarrow \uparrow$$

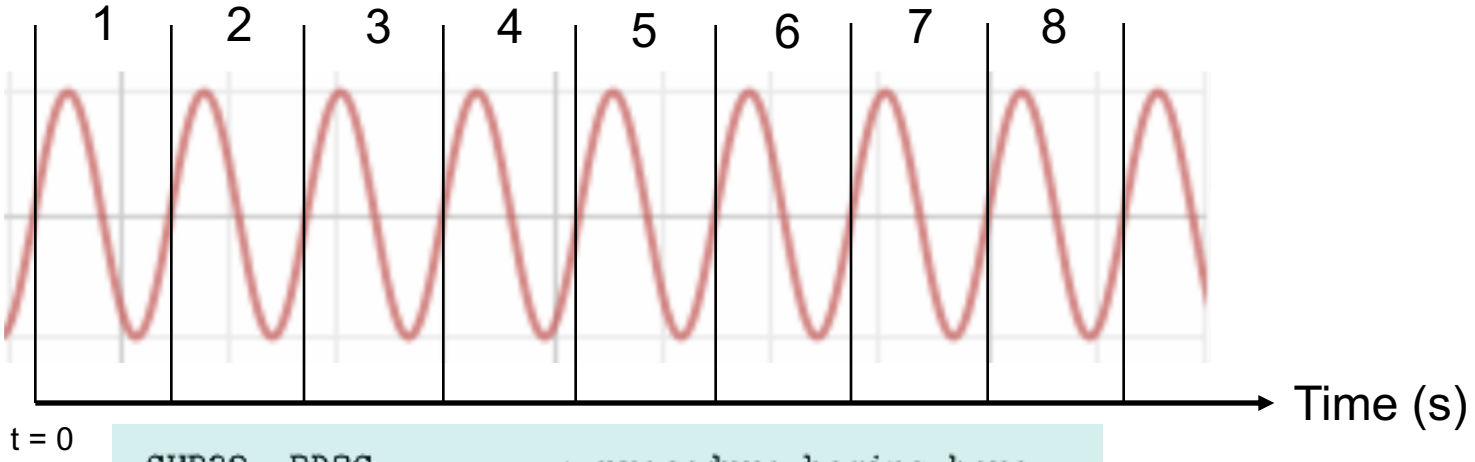
- How to lower CPU time to improve performance?
  1. Reduce number of clock cycles
     - In effect by reducing clock cycles per instruction
  2. Increasing clock rate (in effect, reducing clock cycle time or period)
  3. Or both 1 & 2 (best case)



Instruction Count and CPI

$$\text{CPU Time} = \boxed{\text{CPU Clock Cycles}} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$




```
SUB32   PROC            ; procedure begins here
        CMP   AX,97     ; compare AX to 97
        JL    DONE      ; if less, jump to DONE
        CMP   AX,122    ; compare AX to 122
        JG    DONE      ; if greater, jump to DONE
        SUB   AX,32     ; subtract 32 from AX
DONE:   RET             ; return to main program
SUB32   ENDP            ; procedure ends here
```

Q1) Assuming my CPU can execute every instruction in 1 cycle then what is the total CPU clock cycles here to execute the entire program? 8

Q2) Instruction Count (IC) = ?

Q3) CPI = ?

Instruction Count and CPI

Our original Total CPU Time definition

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

Clock cycles can be replaced with

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

Our new Total CPU Time definition in terms of IC & CPI

Instruction Count and CPI

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

```
SUB32   PROC            ; procedure begins here
        CMP   AX,97     ; compare AX to 97
        JL    DONE      ; if less, jump to DONE
        CMP   AX,122    ; compare AX to 122
        JG    DONE      ; if greater, jump to DONE
        SUB   AX,32     ; subtract 32 from AX
DONE:   RET             ; return to main program
SUB32   ENDP            ; procedure ends here
```

- Running on a 1GHz computer (Clock rate = 1GHz) and
- Average CPI = 2

CPU Time (seconds) = (8 instructions x 2 CPI) / 1000000000 Hz = 0.000000016 s

CPU Time (seconds) = 8 instructions x 2 CPI x 0.000000001 s = 0.000000016 s = 16 ns

Instruction Count and CPI

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program influenced by
  - Algorithm, programming language, ISA and compiler
- Cycles per instruction influenced by
  - CPU hardware
- What if different instructions have different CPI?
  - We always use the average CPI of the instruction set

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same program, same compiler, and same ISA
- Which is faster, and by how much?

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same program, same compiler, and same ISA
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster…

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

…by this much

Average CPI

- If different instruction classes take different numbers of cycles

```
SUB32   PROC          ; procedure begins here
        CMP   AX,97   ; compare AX to 97
        JL    DONE    ; if less, jump to DONE
        CMP   AX,122  ; compare AX to 122
        JG    DONE    ; if greater, jump to DONE
        SUB   AX,32   ; subtract 32 from AX
DONE:   RET           ; return to main program
SUB32   ENDP          ; procedure ends here
```

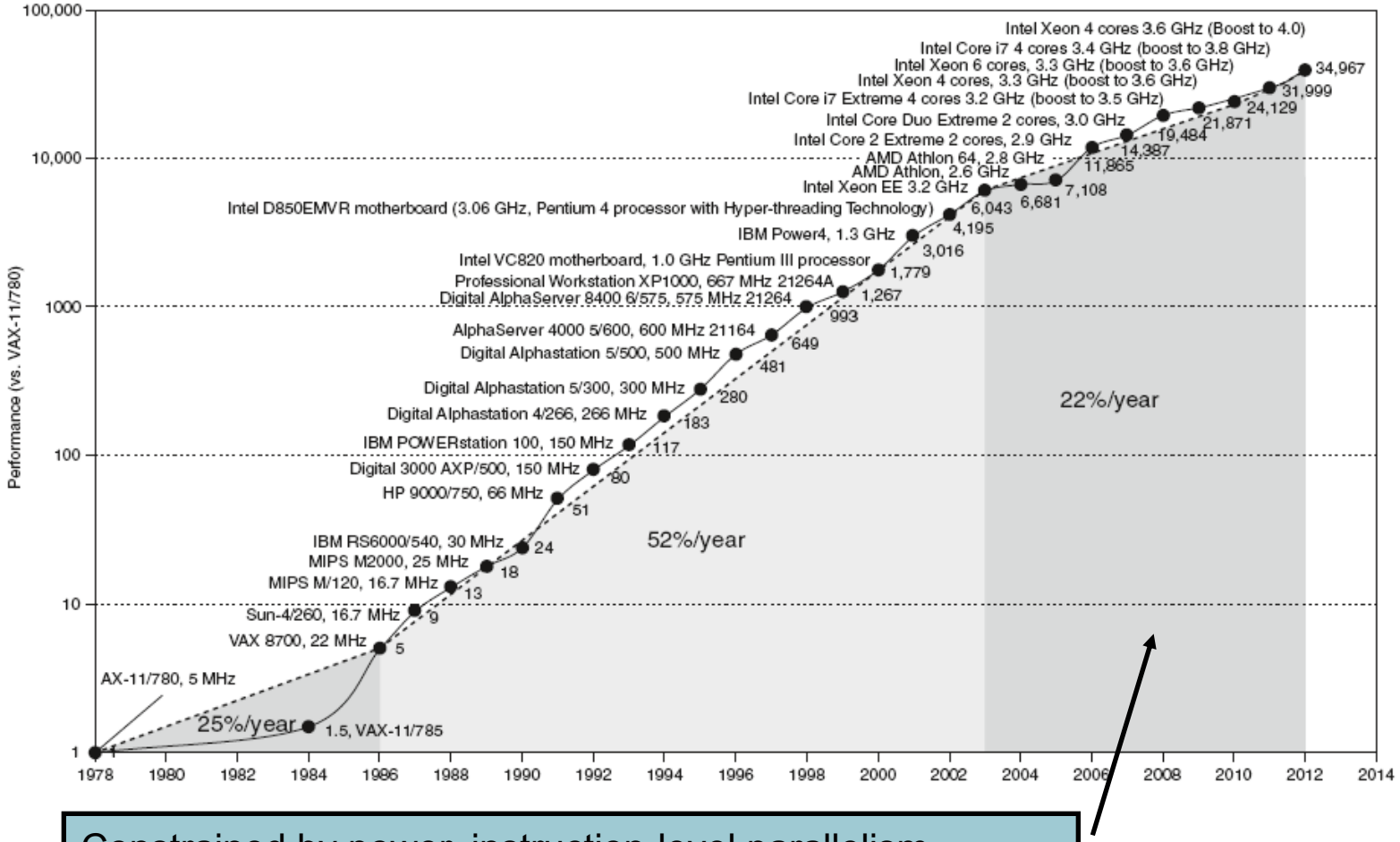
Performance Summary

The BIG Picture

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

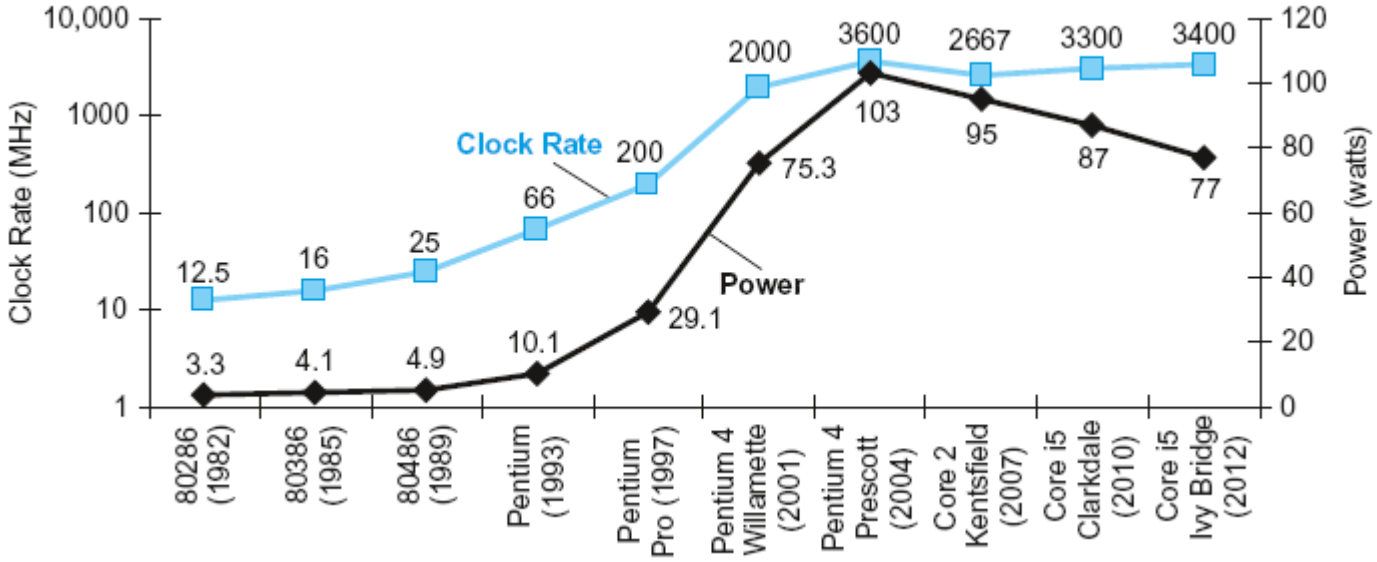
- Performance depends on
  - Algorithm: affects IC, CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - ISA: affects IC, CPI, T_c

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

Power Trends



- In CMOS IC technology

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

× 30 | 5V → 1V | × 1000

Reducing Power

- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

- The power wall
  - We can't reduce voltage further
  - We can't remove more heat
- Parallel processing has come to help.

Multiprocessors

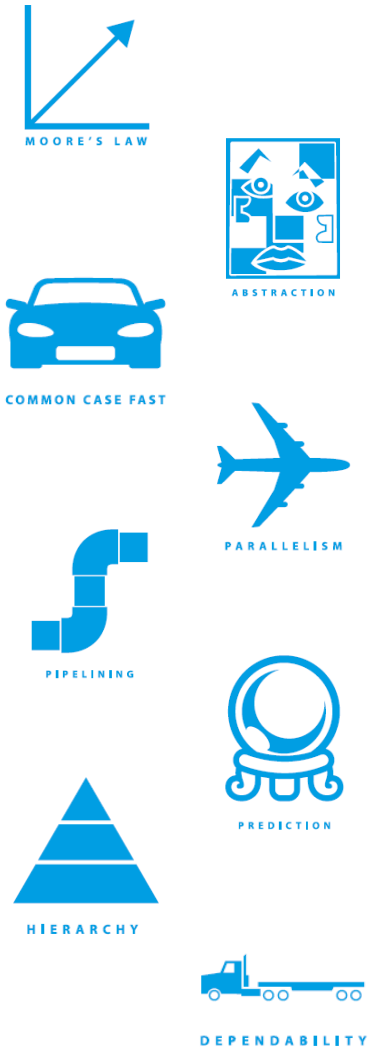
- How hardware designers improve performance
  - What is parallel processing?

Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
- Requires explicitly parallel programming performed by the programmer
  - In contrast with instruction level parallelism (or pipelining performed by the CPU)
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization

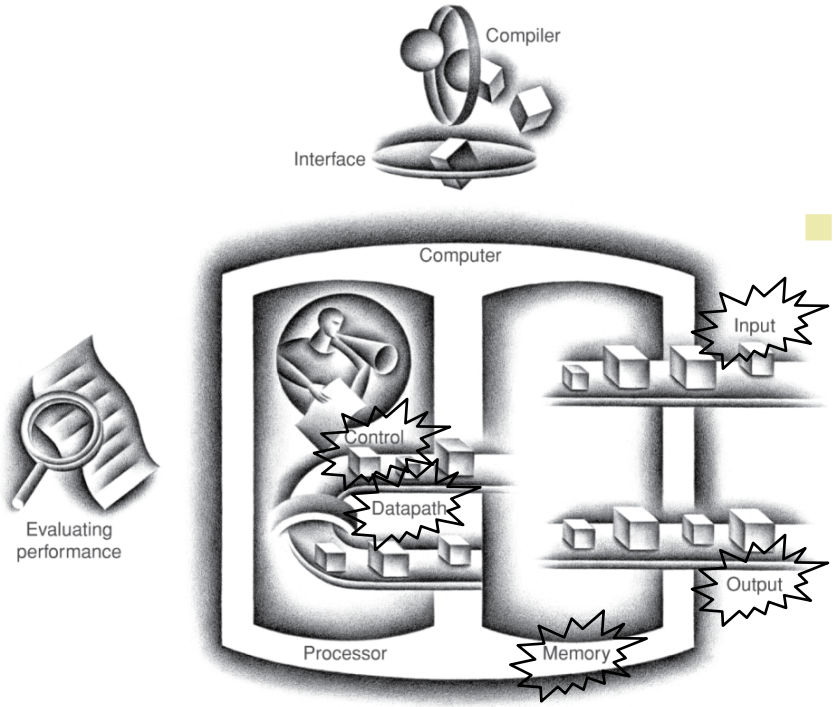
Eight Great Ideas

1. Design for ***Moore's Law***
2. Use ***abstraction*** to simplify design
3. Make the ***common case fast***
4. Performance *via* ***parallelism***
5. Performance *via* ***pipelining***
6. Performance *via* ***prediction***
7. ***Hierarchy*** of memories
8. ***Dependability*** *via* redundancy



Components of a Computer

The BIG Picture



- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

Concluding Remarks

- Cost/performance is improving
  - Due to underlying technology development
- Instruction set architecture acts as the hardware/software interface
  - A great influencer on the performance
- CPU Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance