ECE 485/585
Computer Organization and Design

Lecture 1: Introduction
Fall 2022

Won-Jae Yi, Ph.D.

Department of Electrical and Computer Engineering
Illinois Institute of Technology

Topics

- Introduction to Computers
- Computer Performance Measures
- MIPS Instruction Set
- ALU Design
- Datapath Design
- Control Unit design
- Exceptions
- Pipelined Datapath Design
- Hazards
- Cache
- Memory
- Introduction to Parallel Processors

RISC Vs CISC

Reduced Instruction Set Computer

- Windows CE
- Sony PS 1, 2, PSP
- Nintendo 64
- Cisco Router

MIPS
ARM
RISC-V

Intel
AMD.
MC68K

Introduction

- This course is all about how computers operate
- What do we mean by a computer??
  - Different types: desktop, server, embedded system (e.g., smartphone)
  - Different usage: automobiles, graphics, finance, shopping…
  - Different manufacturers: Intel, AMD, Apple, Microsoft, HP, Samsung…
  - Different underlying technologies and different costs
- Best way to learn
  - Focus on specific instance and learn how it works
  - While learning general principles and historical perspectives

What will you learn?

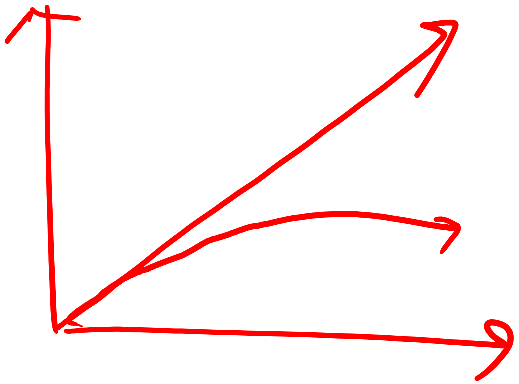
- How programs written in a high-level language are executed in hardware
- Interface between software and hardware
- What determines the performance of a program and how can a programmer improve it?
- How can a hardware engineer improve the performance?

Why learn this topic?

- You want to call yourself a “computer engineer”
- You want to build hardware or software people use (need performance)
- You need to make a purchasing decision or offer “expert” advice

Computer Revolution

- Progress in computer technology
  - Underpinned by Moore's Law
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- Computers are pervasive



Classes of Computers

Edge Computing

Vs.

- Personal computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- Server computers ← Cloud Computing
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized

INTERNET

Classes of Computers

- Supercomputers
  - High-end scientific and engineering calculations
  - Highest capability but represent a small fraction of the overall computer market
- Embedded systems/computers/devices
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

PostPC Era

- Personal Mobile Device (PMD)
  - Battery operated
  - Connects to the Internet
  - Hundreds of dollars
  - Smartphones, tablets, smart watches
- 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, Google, Microsoft

app, data, runtime, middleware, O/S

Platform

app. data

Infrastructure.

PaaS, IaaS

Understanding Performance

- Algorithm
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation

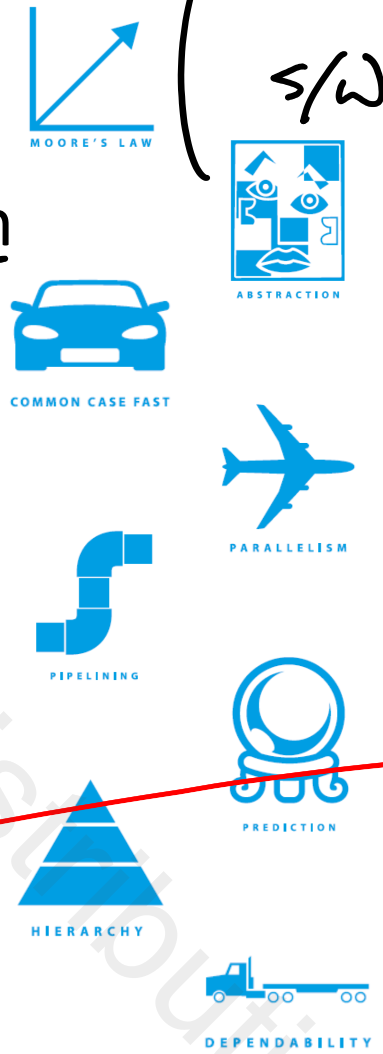
ch2, 3

- Processor and memory system
  - Determine how fast instructions are executed
- I/O system (including OS)
  - Determines how fast I/O operations are executed

ch4, 5, 6

8 Great Ideas

- 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



H/W

S/W

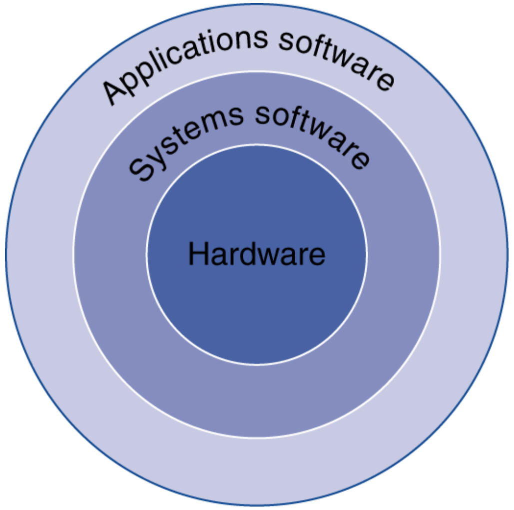
L1 L2 L3

Cache

RAM

HDD/SSD

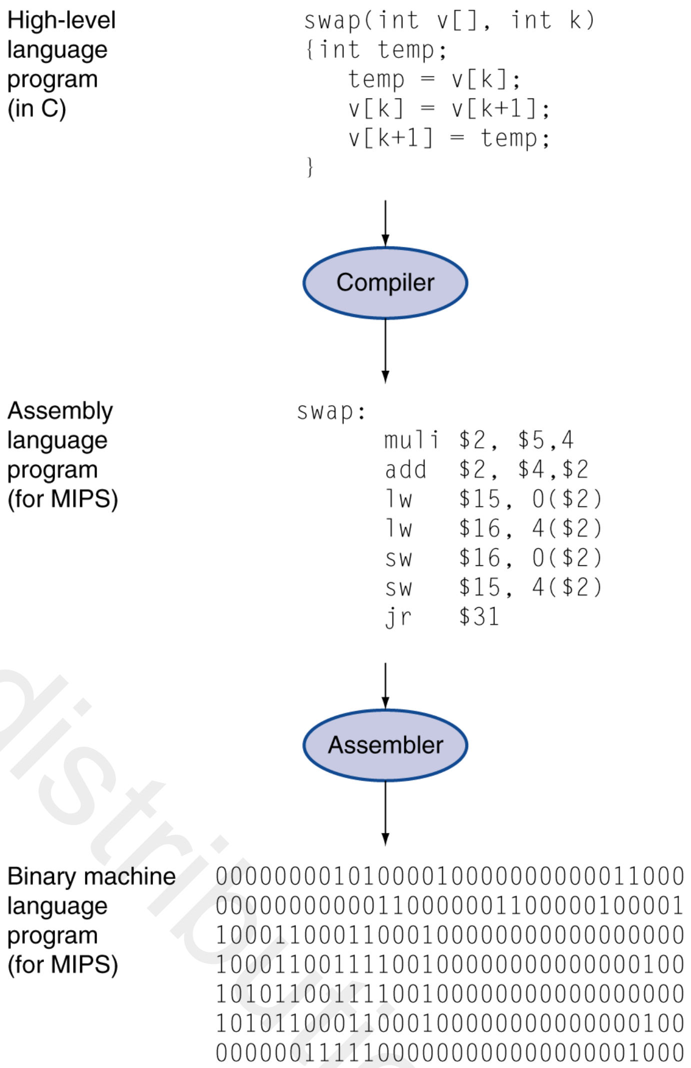
Below Your Program



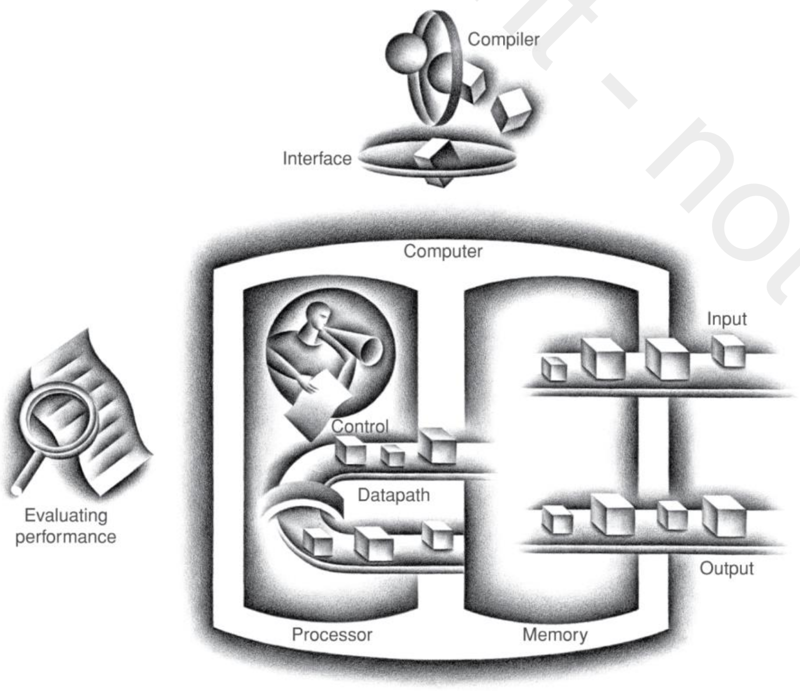
- Application software
  - Written in High-level Language (HLL)
- System software
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory,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

Levels of Program Code

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data



Components of a Computer

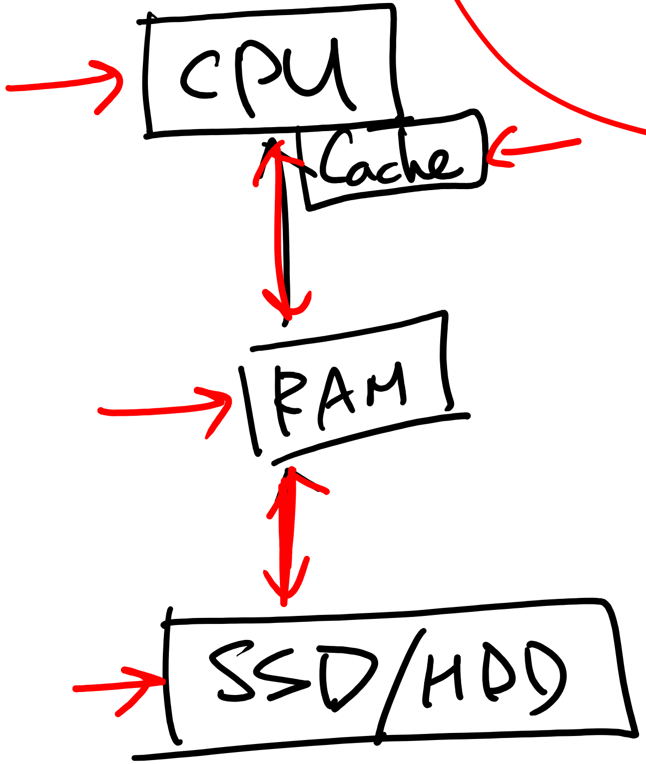


- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, SSD, Magnetic tapes
  - Network adapters
    - For communicating with other computers (machines or human)

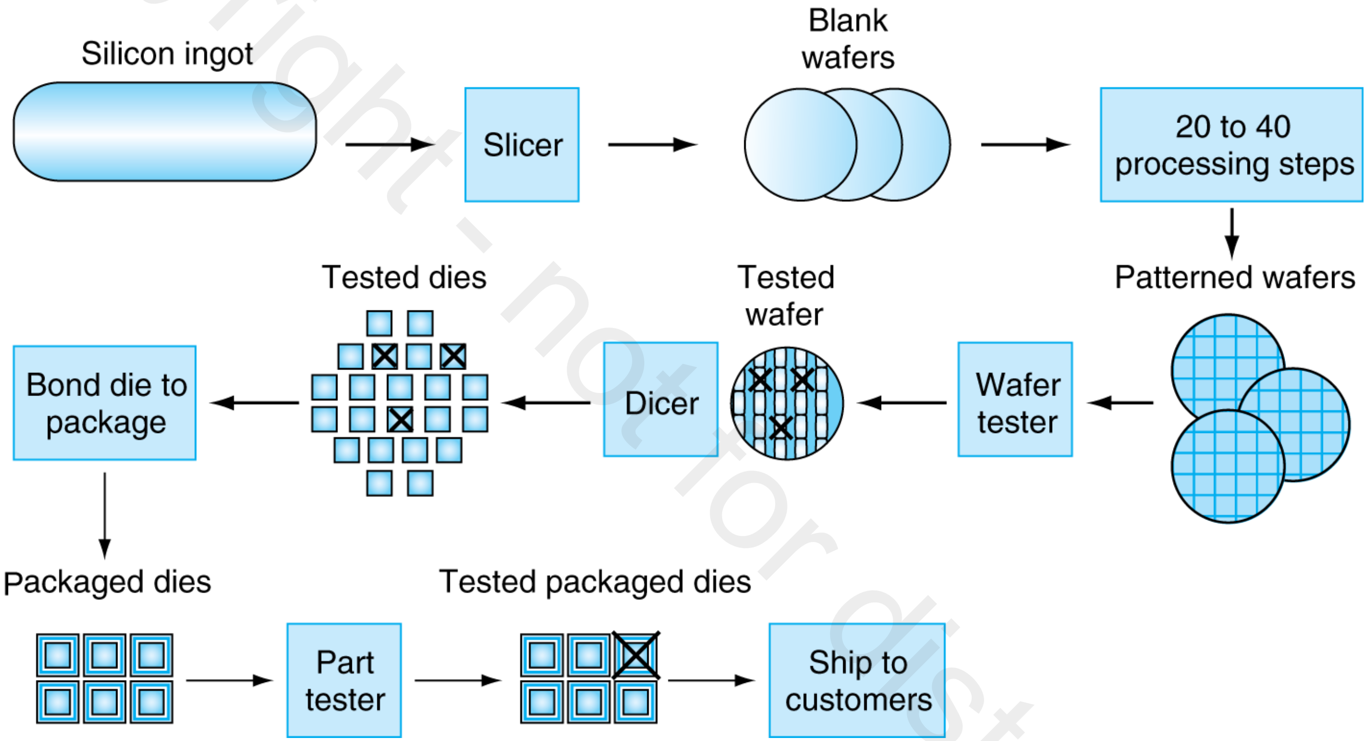
Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
  - Small fast SRAM memory for immediate access to data

frequently used data is stored close to ALU.



Manufacturing ICs



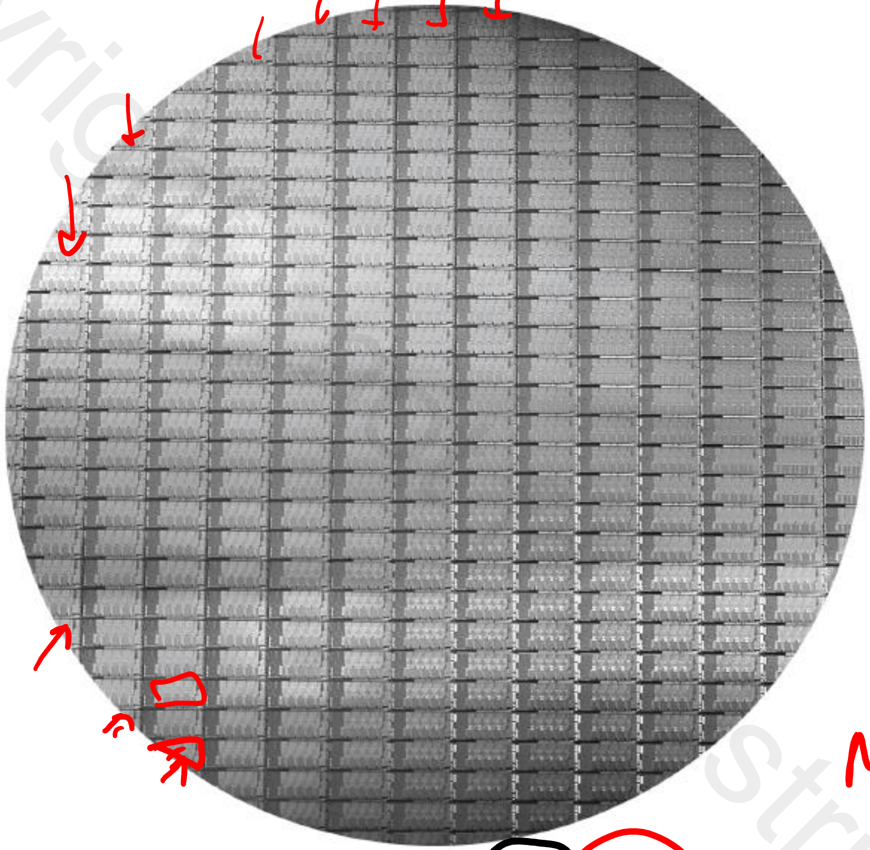
- Yield: proportion of working dies per wafer

https://youtu.be/Q5paWn7bFg4

Intel Core i7 Wafer

average half-pitch of a memory cell at this technology level

x nm tech.



pitch

32nm

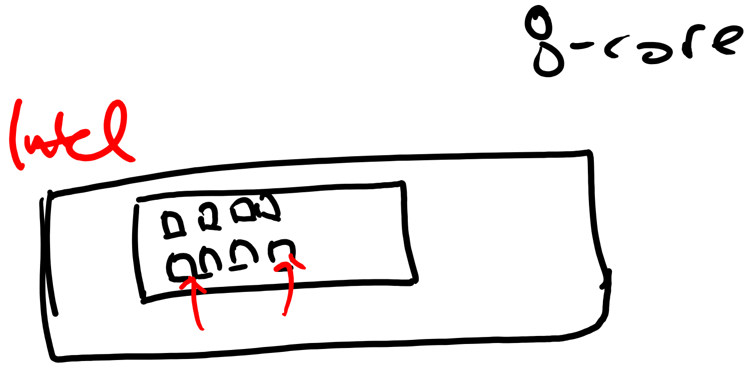
Metal 1 line

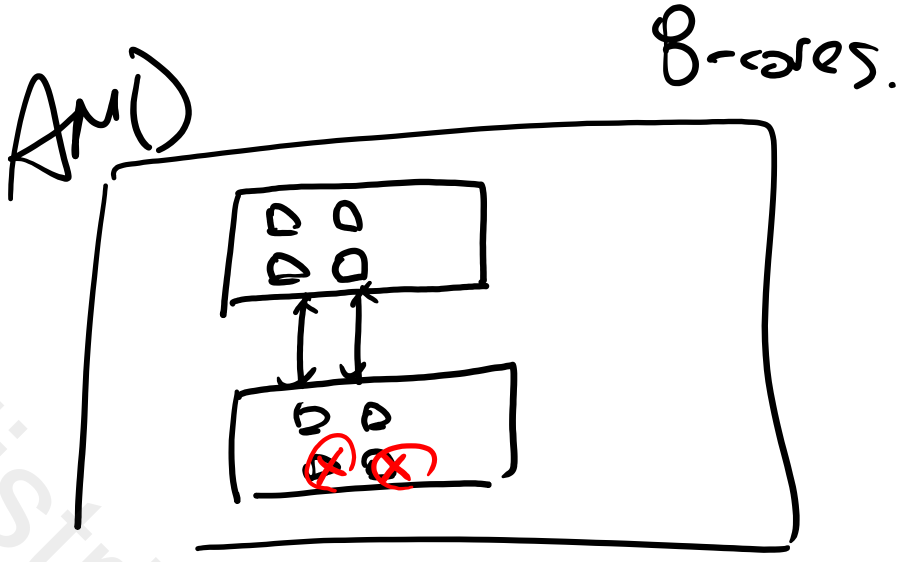
metal 1 line

- 300mm wafer, 280 chips, 32nm technology
- Each chip is 20.7 x 10.5 mm

Intel : one chip.

AMD : 4
6
8





6 core

Integrated Circuit Cost

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

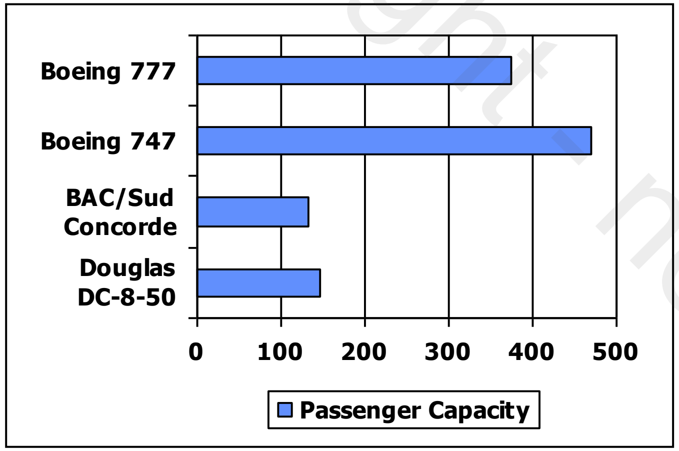
$$\text{Dies per wafer} \approx \text{Wafer area} / \text{Die area}$$

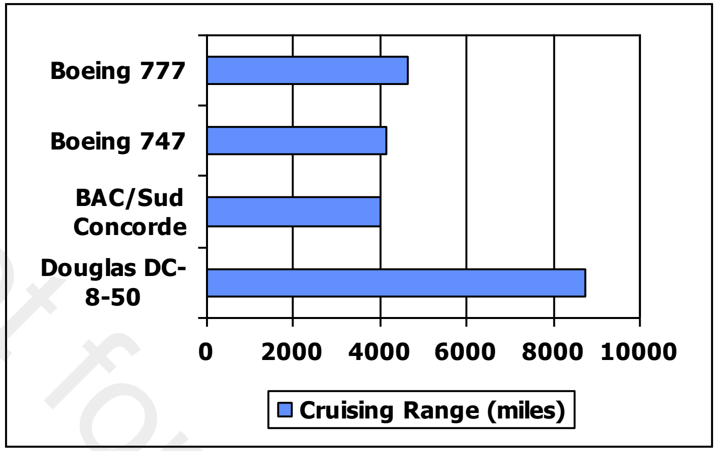
$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}$$

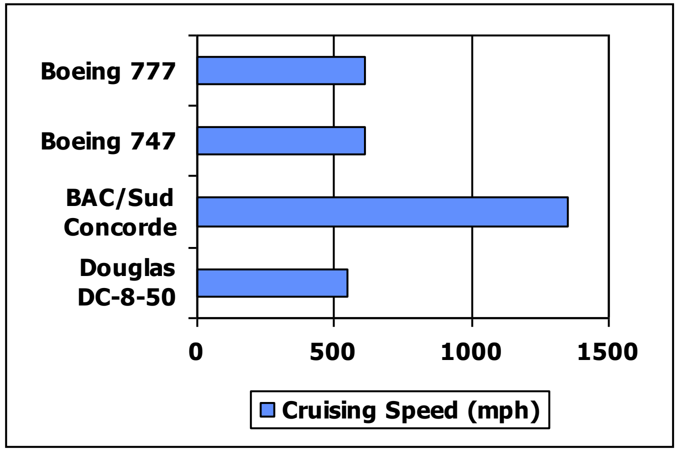
- Nonlinear relation to area and defect rate
  - Wafer cost and area are fixed
  - Defect rate determined by manufacturing process
  - Die area determined by architecture and circuit design

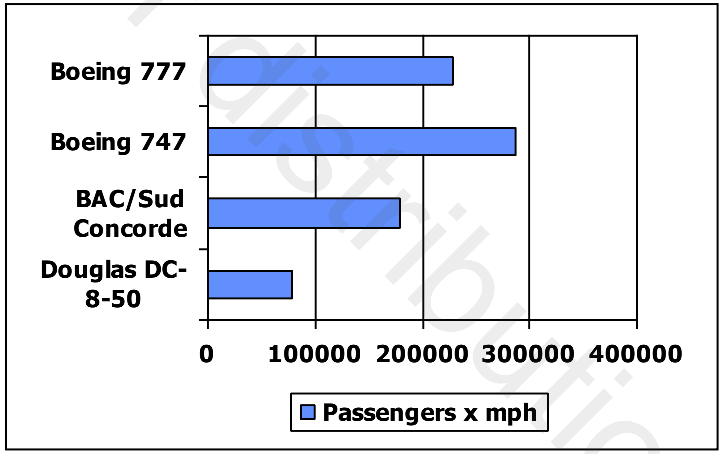
Defining Performance

- Which airplane has the best performance?









Performance Metrics

- Response time
- Throughput
- Relative performance
- Execution time
- CPU time
- Instruction count IC
- Clocks per Instruction (CPI)

Response Time and Throughput

· Processor times
· I/O times
· Memory times
· Propagation delays

"minimized"

- Response time → time difference between submission of a request until the response begins to be received.
  - How long it takes to do a task

"maximized"

- Throughput → number of processes completed per unit time.
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?
- We'll focus on response time for now…

Relative Performance

- Define Performance = 1/Execution Time → time spent by the job/task actively using processor resources
- "X is n time faster than Y"

$$\text{Performance}_X^{A}/\text{Performance}_Y^{B}$$
$$= \text{Execution time}_Y^{B}/\text{Execution time}_X^{A} = \boxed{n}$$

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time$_B$ / Execution Time$_A$ = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Measuring Execution Time

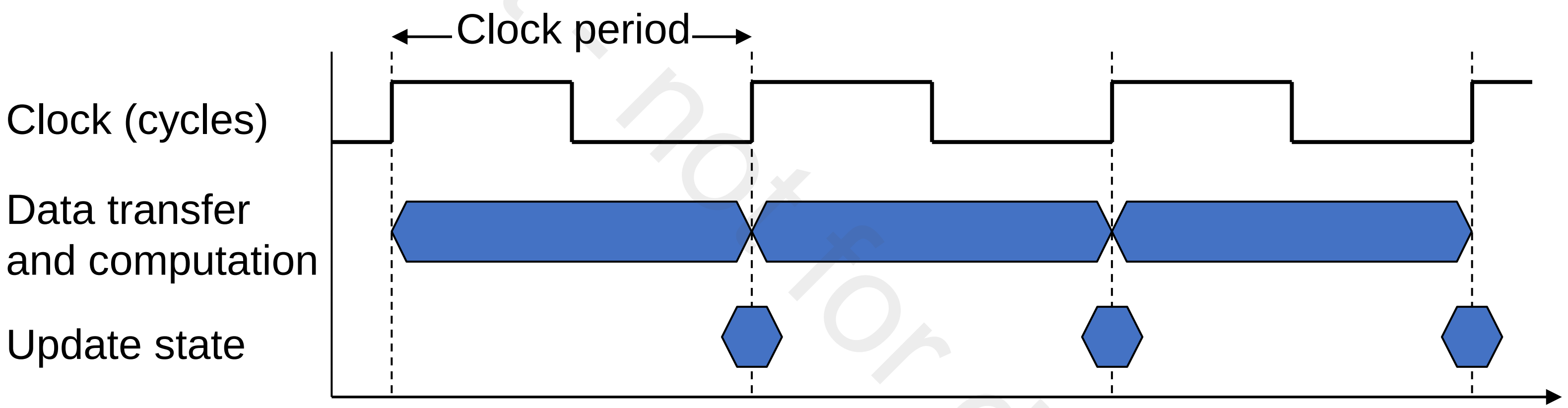
- Elapsed time = Total Execution time.
  - Total response time, including all aspects
    - **Processing, I/O, OS overhead, idle time**
  - Determines system performance
- CPU time → ONLY TIME SPENT AT CPU
  - Time spent processing a given job
    - **Does not count I/O time, other jobs' shares**
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance

Program

OS

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

$T = \frac{1}{f}$

$f = \frac{1}{T}$

CPU Time (CPU Exec. Time)

amount of time for one clock period

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

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

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10s \times 2\text{GHz} = \boxed{20 \times 10^9}$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4\text{GHz}$$

CPU time = $\frac{\text{clock cycle}}{\text{clock rate}}$

clock cycle$_A$ = CPU time$_A$ × clock rate$_A$

Instruction Count and CPI

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

CPI

clock cycle time × CPU

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

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

$$CPU_{time} = IC \times CPI \times T_c = \frac{IC \times CPI}{f_c}$$

- Instruction Count for a program
  - Determined by program, ISA(Instruction Set Architecture) and compiler
- Average cycles per instruction CPI_{avg}
  - Determined by CPU hardware
  - If different, instructions have different CPI
    - Average CPI affected by instruction mix

CPI Example

$\Rightarrow CPU_{time} = IC \times CPI \times T_c$

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

CPI avg

$$CPU\ Time_A = Instruction\ Count \times CPI_A \times Cycle\ Time_A$$

$$= I \times 2.0 \times 250ps = I \times 500ps$$

A is faster…

500 IC ps

$$CPU\ Time_B = Instruction\ Count \times CPI_B \times Cycle\ Time_B$$

$$= I \times 1.2 \times 500ps = I \times 600ps$$

600 IC ps

$$\frac{CPU\ Time_B}{CPU\ Time_A} = \frac{I \times 600ps}{I \times 500ps} = 1.2$$

…by this much

1.2

$\frac{\cancel{600 IC}}{\cancel{500 IC}}$

CPI in More Detail

CPI

- If different, instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

IC

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
  - Clock Cycles
    $= \underbrace{2\times1}_{A} + \underbrace{1\times2}_{B} + \underbrace{2\times3}_{C}$
    $= 10$
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    $= \underbrace{4\times1}_{A} + \underbrace{1\times2}_{B} + \underbrace{1\times3}_{C}$
    $= 9$
  - Avg. CPI = 9/6 = 1.5

Performance Summary

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

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, T_c

CISC → $\boxed{[(A \times B)+A]}$ = AML

RISC → MUL, ADD