Ch:1

Various Computing Architectures

(1) Personal Computers (PCs) - Desktops/Laptops
for individual use, low cost, graphics display, keyboard, mouse

(2) Servers - Much larger computers accessed via network, runs larger programs for multiple users

Greater computing, storage, and input/output capacity
More emphasis on dependability
Application → File Storage, Web Serving, Business Appl.

(3) Supercomputers - Several processors, 1 TB = 2^{40}, several terabytes of memory

Applications → High-end scientific applications
e.g. Weather Forecasting
Protein structure prediction

(4) Embedded Computers → computer inside a device for running specific applications

e.g. - µP inside a car
- computer in a TV set
- Network of processors for controlling airplane
- Often have stringent requirements on cost or power
- Lower tolerance to failure

Post PC Era

(1) PC → Personal Mobile Device (PMD): e.g. Smart Phone, Tablet PC
Battery operated, Wireless connectivity

(2) Server → Cloud Computing: Warehouse Scale Computers (WSC)
100,000 Servers
(Amazon, Google)

Some Key Questions

(1) Programs written in high-level language
↓
Language of the hardware
↓
Hardware Executes the resulting program by hardware

(2) What techniques can be used to improve performance?

(3) ——— to improve energy efficiency?

(4) Why multicore μPs?
↓
(multiple processors in a single IC)

Eight Great Ideas in Computer Arch.

(1) Moore's Law :-

- Ic resources double every 18-24 months.
- Computer architects must anticipate where the technology can reach when the design finishes (as designs can take years).

(2) Use Abstractions to Simplify Design :-

- Lower level details are hidden to offer a simpler model at higher levels

(3) Make the Common Case Fast :-

- Common case is simpler than the rare case
- A judicious strategy is to make the common case fast rather than optimize the rare case

(4) Performance via Parallelism :-

- Enhance performance by performing operations in parallel

e.g. Switch from unicore to multicore processors

(5) Performance via Pipelining

- A particular pattern of parallelism among instructions

e.g. One can increase the depth of the pipeline to overlap more instructions

(6) Performance via Prediction

- Compiler or processor guesses the outcome of an instruction to remove it as a dependence in executing other instructions

e.g. We might speculate on the outcome of a branch, so that instructions after the branch could be executed earlier

(7) Hierarchy of Memories :—

Memory (conflicting requirements resolved by Hierarchy)
- Fast :— Memory speed shapes performance
- Large :— Capacity limits size of problems
- Cheap :— Cost determines cost of computer

Cache
- Fastest - Smallest - Most Expensive/bit — TOP
- Slowest - Largest - Cheapest/bit — BOTTOM

(8) Dependability via Redundancy :—

Redundant components can take over when a failure occurs and also help detect failures. In this way computers can be made dependable

Five classic components of a Computer

Input : Through which a computer is fed information e.g. keyboard, microphone

Output : Through which a computer conveys the result of computation e.g. display, speaker

Memory : Programs are kept when they are running and also data needed by the running programs

Built from Dynamic Random Access Memory chip

(Random Access → Memory access takes same amount of time independent of what portion is being read) (*)

Datapath : The component of the processor that performs arithmetic operations

Control : The component of the processor that commands the datapath, memory and program I/O devices according to the instructions

Processor / Central Processor Unit (CPU)

(*) **Cache** : A small, fast memory that acts as a buffer for a slower, larger memory

SRAM technology : Faster, less dense and more expensive than DRAM

Primary / Main Memory - Memory inside the computer
Volatile → useful only when computer has power
(access time ~ ms)

(DRAMs)

Secondary / Nonvolatile memory - Used to store programs and data between runs

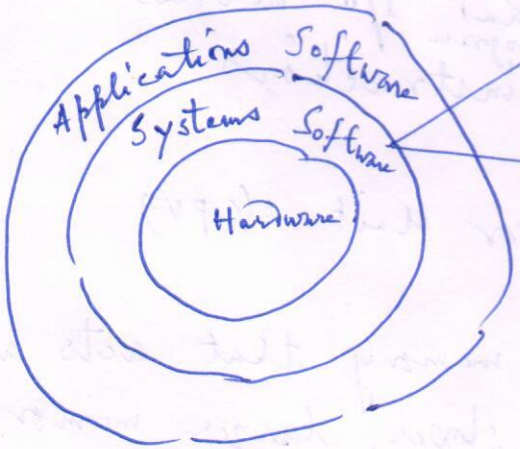
Magnetic disks / Hard Disk
- Rotating mechanical devices having larger access time and lower cost (~ ms

Flash memory
- Semiconductor memory cheaper and slower than DRAM but costlier and faster than magnetic d
(access time ~ μs)

~~Backlogues~~

Program and Instructions

Use abstractions



OS : Supervising program that manages the resources of a computer for proper handling the progra

Compilers : A program that translates high language statemen into assembly language statem

High-level language program (in C)

↓ COMPILER

Assembly language program (in MIPS)

↓ ASSEMBLER

Binary machine language program

Instruction Set Architecture / Architecture

An abstract interface between the hardware and the lowest-level software which would enable a machine language program to work properly.

Application Binary Interface

Basic instruction set + operating system interface

↓

for application programmers

Technologies for building processors and memory

Transistors (On/off switch)

↓

ICs (~ 100s of Transistors)

↓

VLSI (~ Millions of Transistors)

Performance

Q: What do we mean when we say a computer has a better performance than another

For a desktop computer, reduce response time / execution time

↓

→ time between start and completion of a task

For a datacenter, several servers running, increase throughput / bandwidth

↓

Total amount of work done in a given time

$$\text{Performance}_X = \frac{1}{\text{Execution Time}_X}$$

X is n times faster than Y

$$\Rightarrow \frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

Ex.0 If computer A runs a program in 10 sec and computer B runs the same program in 15 sec., then perf. ratio $= \frac{15}{10} = 1.5$

$\Rightarrow$ A is 1.5 times faster than B

CPU Execution Time / CPU Time

The actual time CPU spends for computing a specific tas

User CPU Time

CPU time spent in the program

(CPU Performance)

System CPU Time

CPU time spent in th OS performing tasks on behalf of the program

(System Performance

CPU execution time for a program = CPU clock cycles for a program × Clock Cycle T

(Clock Period : Length of each clock cy
Clock Cycle : Time for one clock period

$$= \frac{\text{CPU clock cycles}}{\text{Clock Rate}}$$

Ex.1

Given

Computer A has 2 GHz clock → A specific program runs in 10 sec.

Required

In a computer B, the program should run in 6 sec. In order to achieve this goal, computer B requires 1.2 times as many clock cycles as computer A for this program.

Design

Find the clock rate of B.

Solution :—

10

$$\text{CPU time}_A = \frac{\text{CPU clock cycles}_A}{\text{Clock rate}_A} \quad \text{—(1)}$$

$$10 \text{ sec} = \frac{\text{CPU clock cycles}_A}{2 \text{ GHz}}$$

$$\Rightarrow \text{CPU clock cycles}_A = 10 \text{ sec.} \times (2 \times 10^9) \frac{\text{cyc}}{\text{se}}$$

$$= 20 \times 10^9 \text{ cycles} \quad \text{—(2)}$$

Similarly

$$\text{CPU time}_B = \frac{\text{CPU clock cycles}_B}{\text{clock rate}_B} \quad \text{—(3)}$$

$$6 \text{ sec} = \frac{1.2 \times \text{CPU clock cycles}_A}{\text{clock rate}_B}$$

$$\Rightarrow \text{clock rate}_B = \frac{1.2 \times 20 \times 10^9}{6} = 4 \text{ GHz}$$

So, to run the program in 6 sec., B must have twice the clock rate of A.

Instruction Performance

CPU clock cycles = Instructions for a program × Average clock cycles per instruction (CPI) for a program or part of it

Background

The computer performance should depend on the no. of instructions for the program has

compiler generate instructions which a computer ~~had~~ to execute to run the program

Ex 2 Two different implementations of the same instruction set architecture

Computer A

Clock Cycle → 250 ps

CPI → 2.0

Computer B

Clock Cycle → 500 ps

CPI → 1.2

Which computer is faster and by how much?

Soln Let both the computers execute the same number of instructions I.

$$\text{CPU clock Cycles}_A = I \times 2.0 \quad \text{—(1)}$$

$$\text{CPU Clock Cycles}_B = I \times 1.2 \quad \text{—(2)}$$

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

$$= I \times 2.0 \times 250 \quad \text{—(3)}$$

$$= 500 \times I \text{ ps}$$

Similarly,

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

$$= I \times 1.2 \times 500 \quad \text{——(4)}$$

$$= 600 \times I \text{ ps}$$

$$\frac{\text{CPU Performance}_A}{\text{CPU Performance}_B} = \frac{\text{CPU Execution Time}_B}{\text{CPU Execution Time}_A} = \frac{600 \times I}{500 \times I}$$

$$= 1.2$$

So, computer A is 1.2 times faster than computer B.

CPU Performance Equation

(no. of instructions executed by progra

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

$$= \frac{IC \times CPI}{\text{Clock Rate}}$$

Observation

Three different factors affect CPU time

IC ↑ ⇒ CPU time ↑ ⇒ CPU Perf. ↓

CPI ↑ ⇒ CPU time ↑ ⇒ CPU Perf. ↓

Clock Rate ↑ ⇒ CPU time ↓ ⇒ CPU Perf.

CPU time = Instruction Count × Clock Cycle Time × Cycles per instruction

So $\text{CPU time} = \left(\sum_{i=1}^{n} IC_i \times CPI_i\right) \times \text{Clock-Cycle Time}$ (# instr. = n)

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

Note

1. CPU time is equally dependent on these three fac
2. Basic technologies influencing these factors are interdependent.

* Clock Cycle Time — Hardware technology and organization
* CPI — Organization and instruction set architecture
* Instruction Count — Instruction set architecture and compiler technology

Potential performance improvement techniques primarily improve one component and slightly impair the other two

Ex.3 Consider two different implementations of the same instruct set architecture

Instructions Based on CPI Divided into 4 classes

	CPI	CPI
A	1	2
B	2	2
C	3	2
D	3	2

Clock Rate: 2.5 GHz 3 GHz

Given a program with dynamic instruction count 1X1

A: 10%, B: 20%, C: 50%, D: 20%

Which implementation is faster? Global CPI for each sy
Clock Cycles ----

Soln

$$\text{CPU time}_I = \left(\sum_{i=1}^{4} IC_{i_I} \times CPI_{i_I}\right) \times \frac{1}{\text{Clock Rate}_I}$$

$$= \frac{10^6 \times \left((0.1\times1)+(0.2\times2)+(0.5\times3)+(0.2\times\right.}{2.5\times10^9}$$

$$= 1.04 \times 10^{-3}\ s$$

$$\text{Clock Cycle}_I = CPI_I \times IC_I = 2.6\times10^6$$

Similarly

$$\text{CPU time}_{II} = 0.17 \times 10^{-3}\ s$$

$$\text{Performance} \propto \frac{1}{\text{CPU time}}$$

$\Rightarrow$ P-II is faster than P.I.

Global CPI

$$\text{CPU time}_I = IC_I \times CPI_I \times \frac{1}{\text{Clock Rate}_I}$$

$$\Rightarrow CPI_I = \frac{\text{CPU time}_I \times \text{Clock Rate}_I}{IC_I}$$

$$= \frac{1.04\times10^{-3} \times 2.5 \times 10^9}{10^6}$$

$$= 2.6$$

Similarly $CPI_{II} = 2.01$

$$\text{Clock Cycle}_{II} = CPI_{II} \times IC_{II} = 2.01\times10^6$$

CPU Time = IC × CPI × Clock Cycle Time

↓ Measured by running a program

IC: ↓ Measured by Simulator architecture or Software tools

Clock Cycle Time: ↓ Published as part of documentation

Depends on a wide variety of factors like memory system and processor structure (varies by application, also by implementations with same instruction set)

Understanding Program Performance

1. Algorithm → IC*, CPI** → * No. of source program instructions, hence machine processor instructions
   ** slower or faster

2. Programming Language → IC*, CPI** → * statements translated, processor instructions
   ** Heavy support for data abstraction requires indirect calls

3. Compiler → IC*, CPI** → * Source language instructions translated into machine instructions

4. Instruction Set Architecture → IC, CPI, Clock Rate → Affects instructions needed for a program, cost in cycles of each instruction, overall clock rate

An alternative performance measure :—

$$MIPS = \frac{\text{Instruction Count}}{\text{Execution Time} \times 10^6}$$

↓ Million Instructions per Second

(instruction execution rate)

* Faster computers have higher MIPS rating

Problems using MIPS

(1) Computers with different instruction sets cannot be compared using MIPS. (IC will vary)

(2) MIPS varies between programs on the same computer

$$MIPS = \frac{Ic}{\frac{Ic \times CPI}{\text{Clock Rate}} \times 10^6} = \frac{\text{Clock Rate}}{CPI \times 10^6}$$

(System Perf. Eval. C ↑ SPEC CPU2

So, if CPI varies e.g. by a factor of 5 for on an Intel Core i7 computer, MIPS would also vary

(3) If a new program executes more instructions but each instruction is faster, MIPS can vary independently from performance

	A	B
IC	10 billion	8 billion
Clock Rate	4 GHz	4 GHz
CPI	1.0	1.1

Seems faster →

$$MIPS_A = \frac{4 \times 10^9}{1 \times 10^6} = 4 \times 10^3$$

$$MIPS_B = \frac{4 \times 10^9}{1.1 \times 10^6} = 3.64 \times 10^3$$

$$CPU\ Time_A = \frac{(10 \times 10^9) \times (1)}{(4 \times 10^9)} = 2.5\ s$$

Faster → $CPU\ Time_B = (8 \times 10^9) \times (1.1) / 4 \times 10^9 = 2.2\ s$

15
17

$$\text{Clock Cycle}_I = CPI_I \times IC_I$$
$$= 2.6 \times 10^6 = 26 \times 10^5$$

$$\text{Clock Cycle}_{II} = CPI_{II} \times IC_{II}$$
$$= 2.01 \times 10^6$$
$$= 20.1 \times 10^5$$

Amdahl's Law

Performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used

$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

$$= \frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$$

$$\text{Execution time}_{new} = \text{Execution time}_{old} \times \left((1 - \text{Fraction}_{enhanced}^{(\alpha)}) + \frac{\text{Fraction}_{enhanced}^{(\alpha)}}{\text{speedup}_{enhanced}^{(s)}} \right)$$

$$\text{Speedup}_{overall} = \frac{\text{Execution time}_{old}}{\text{Execution time}_{new}} = \frac{1}{(1-\alpha) + \frac{\alpha}{s}}$$

Notes :-

(1) Fraction Enhanced : Fraction of the computation time in the original machine that can be converted to take advantage of the enhancement

e.g. Suppose a program takes 60s to complete. 20s of the execution time can make use of the enhancement

$\Rightarrow$ Fraction$_{enhanced}$ $= \frac{1}{3}$ (always < 1).

Ex 4 Enhancement to the processor of a server system used for Web serving. New CPU is 10 times faster on computation in Web serving application than the original processor. Original CPU is busy with computation 40% of the time and is waiting for I/O 60% of the time. Overall speedup = ?

Soln.

Fraction$_{enhanced}$ $= 0.4$

Speedup$_{enhanced}$ $= 10$

$$\text{Speedup}_{overall} = \frac{1}{(1-0.4) + \frac{0.4}{10}} = 1.56.$$

Notes :- (contd.)

(2) Speedup$_{enhanced}$: Improvement gained by the enhanced execution mode; how much faster the task would run if the enhanced mode were used for the entire program

$$= \frac{\text{Time of the original mode}}{\text{Time of the enhanced mode}} >$$

(3) Law of diminishing returns - incremental improvement in speedup gained by an additional improvement in the performance of just a portion of the computation diminishes as improvements are added.

Ex.5 Suppose 90% of a calculation can be parallelized. Find the maximum speedup that can be achieved with 10, 100 and 1000 processors

$\text{Fraction}_{\text{enhanced}} = 0.9$ (since 90% of a calculation can be parallelized)

With 10 processors :-

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

$$= \frac{1}{(1-0.9) + \frac{0.9}{10}} \quad (\because 10 \text{ p}$$

$$= 5.26$$

With 100 processors :-

$$\text{Speedup}_{\text{overall}} = \frac{1}{(1-0.9) + \frac{0.9}{100}} = 9.17$$

With 1000 processors :-

$$\text{Speedup}_{\text{overall}} = \frac{1}{(1-0.9) + \frac{0.9}{1000}} = 9.91$$

(4) Helps to compare designs

(4) Helps to compare design alternatives.

Ex. 6 FPSQR is responsible for 20% of the execution time of a critical graphics benchmark.

Proposal 1 :- Enhance FPSQR hardware to achieve a speedup of 10

Proposal 2 :- Make FP operations, which contribute upto 50% of the execution time, 1.8 times faster.

Compare the two proposals.

Soln:

Proposal 1

$$\text{Fraction}_{\text{enhanced}} = 0.2$$

$$\text{Speedup}_{\text{enhanced}} = 10$$

$$\text{speedup}_{\text{overall}} = \frac{1}{(1-0.2) + \frac{0.2}{10}} = 1.22$$

Proposal 2

$$\text{Fraction}_{\text{enhanced}} = 0.5$$

$$\text{Speedup}_{\text{enhanced}} = 1.8$$

$$\text{speedup}_{\text{overall}} = \frac{1}{(1-0.5) + \frac{0.5}{1.8}} = 1.29$$

So, proposal 2 is better.

Ex. A JAVA application runs in 15s on a desktop PC. New JAVA compiler requires only 0.6 as many instructions but increases CPI by 1.1. How fast the application would run in new compiler

Let in original PC, instr. count be IC_1 and clock rate be CR_1 — CPI be CPI_1

$$CPU\ Time_1 = \frac{IC_1 \times CPI_1}{CR_1} \Rightarrow 15 = \frac{IC_1 \times CPI}{CR_1}$$

$$CPU\ Time_2 = \frac{IC_2 \times CPI_2}{CR_2} = \frac{0.6\, IC_1 \times 1.1\, CPI}{CR_1}$$

$$\Rightarrow CPU\ Time_2 = 0.6 \times 1.1 \times \left(\frac{IC_1 \times CPI_1}{CR_1}\right)$$

$$= 0.6 \times 1.1 \times 15$$

$$= 9.9\ s$$

Turbo Mode :- Intel Core i7 processor
Temporarily increase clock rate by about 10% until the chip gets too warm

IPC :- CPI can be fractional e.g. some processor can fetch and execute multiple instructions per clock cycle (IPC)

If $IPC = 2$, $CPI = \frac{1}{IPC} = 0.5$

Power consumption (metric Joules)

IC fabrication by CMOS

↓

Dynamic Energy (primary source of energy consumption)
(energy consumed when transistors switch state from off to on and vice-versa)

Energy for single transistor ∝ ½ × Capacitive Load × Voltage^2

↳ # transistors connected to an o/p
technology determines cap. of (farads) wires and transistors

↓

(0 → 1 → 0 or 1 → 0 → 1)

Power required per transistor

Power ∝ ½ × Capacitive Load × Voltage^2 × Frequency switched

(fn. of fan out and technology) (fn. of clock rate)

↓

cap. of wires/trans.

Ex. New processor : 85% of capacitive load
15% voltage reduction results in a
15% shrink in freq.

$$\frac{\text{Power}_{new}}{\text{Power}_{old}} = \frac{(0.85\,CL)(0.85\,V)^2(0.85\,FS)}{CL \times V^2 \times FS} = 0.52$$

Note

1. Static energy consumption in CMOS : Leakage current even when the transistor is off.

2. Power for ICs → Fed in to chip and distributed in chip
→ Dissipated as heat and must be removed

From Uniprocessors to Multiprocessors

Processors ≡ Cores

Microprocessors ≡ Multicore Microprocessors

e.g. Quadcore microprocessor ≡ Chip with four processors/co

Decreasing response time of a single program running on the single processor has reached a limit.

Alternative :—

Use multiple processors to improve response time
rewrite codes

Difficulties for writing explicit parallel programs

1. Performance programming → Correct Fast

2. Programmer needs to divide an application more-or-less uniformly to each processor & also

3. Overhead of scheduling and coordination should be less

Some examples of parallelism :—

1. Subword Parallelism / Parallelism and Computer Arithmetic / Data Level Paralle
   - Computing on elements in parallel such as during adding or multiplying two vectors

2. Instruction Level Parallelism / Pipelining :—
   - Runs programs faster by overlapping the execution of instructions

3. Parallelism and Memory Hierarchy:
Cache Coherence : All processors use the same address space so that any processor can read or write any data

Parallelism and Memory Hierarchy:

4. Redundant Array of Inexpensive Disks:
(RAID) : An organization of disks that uses an array of small and inexpensive disks so as to increase both performance and reliability

Accelerate I/O Perf.
(Increase read heads by replacing a few large disks by large no. of such disks)

Graphics Processing Unit (GPU):

- A processor optimized for 2D and 3D graphics, video, visual computing and display.

- Uniform and scalable array of processors, large amount of floating point processing power

GPU computing via a parallel programming language and API

Compute Unified Device Architecture : A scalable parallel programming model and language based on C/C++.

- GPU unifies graphics and computing

Ex,

	CPI	Single Processor
Arithmetic	1	2.56×10^9
L/S	12	1.28×10^9
B	5	256×10^6

Each processor has 2 GHz clock freq.

Parallelized to multiple cores – 1, 2, 4, 8

No. of A, L/S divided by $0.7 \times p$ (per process

– – – B remains same

Find Total execution time on 1, 2, 4, 8 processors and show rel. sp.ups for 2, 4, 8

Soln. $$\text{Exec. Time} = \frac{\text{Clock Cycle}}{\text{Clock Rate}}$$

$$\text{Clock cycles} = CPI_A \times \#A + CPI_{L/S} \times \#L/S + CPI_B \times \#B$$

$$= 19.2 \times 10^9$$

$$\text{Exec. Time}_{\text{single Proc.}} = \frac{19.2 \times 10^9}{2 \times 10^9} = 9.6\text{s}$$

$$\text{Exec. Time}_{\text{2-Proc.}} = \frac{\frac{1 \times 2.56 \times 10^9}{0.7 \times 2} + \frac{12 \times 1.28 \times 10^9}{0.7 \times 2} + \frac{5 \times 256 \times 10^6}{1}}{2 \times 10^9}$$

$$= 7.04\text{ s}$$

$$\text{Rel. speed up} = \frac{7.04}{9.6} = 0.73$$

Total exec. tim = $9.6 + 7.04 + 3.84 + 2.24 = 22.72\text{s}$

4 Proc. — 3.84s, 0.4

8 Proc. — 2.24s, 0.23

b) CPI of the A instr. doubled, what would be the impact on the execution time?