Lec-2

10 May 2021 15:05

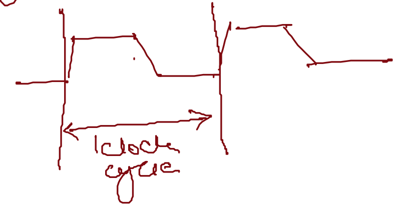
Principle of locality :
Systems that exhibit strong *locality of reference* are great candidates for performance optimization through the use of techniques such as the caching, prefetching for memory and advanced branch predictors at the pipelining stage of a processor core.

In computer science, **locality of reference**, also known as the **principle of locality**,[1] is the tendency of a processor to access the same set of memory locations repetitively over a short period of time.[2] There are two basic types of reference locality – temporal and spatial locality.

- Temporal locality refers to the **reuse of specific data and/or resources within a relatively small time** duration
- Spatial locality (also termed *data locality*[3]) refers to the **use of data elements within relatively close storage locations**.
  - Sequential locality, a special case of spatial locality, occurs when data elements are arranged and accessed linearly, such a s traversing the elements in a one-dimensional array.

clock cycle : A clock cycle is an electronic pulse. A CPU performs 1 unit of or 1 part of an instruction in 1 CPU cycle

clock cycle of a CPU = measured in Hertz (frequency) => No of cycles per second



1 MHz = 10^6 clockcycle / sec

1 GHz = 10^9 clock cycle / sec

Eg: a CPU is 2.5 GHz ie 2.5×10^9 cc/sec

1 Byte = 8 bit

1 kB = 2^{10} Byte

1 MB = 2^{10} kB = $2^{10} \times 2^{10}$ Byte

1 GB = 2^{10} MB = $2^{10} \times 2^{10} \times 2^{10}$ Byte

1 kB = 2^{10} Byte

1 MB = 2^{20} Byte

1 GB = 2^{30} Byte

1 TB = 2^{40} Byte

b = bit

B = Byte

8 x bit = Byte

1 micron = 1 millionth of a meter 10^{-6} m

Circuits of a computer chip are measured in the length of microns

★ Execution time

total time to complete a task

Throughput / Bandwidth

no of task completed / unit time

$$\boxed{\text{Performance} = 1/\text{Execution time}}$$

$\text{Perf}_x > \text{Perf}_y$ then $\text{Exectime}_y < \text{Exectime}_x$;

Quantitative performance analysis

x = n times faster than y

n Exe x

$\boxed{x = n \text{ times faster than } y}$

then $\frac{\text{Perf } x}{\text{Perf } y} = n$ ie $\frac{\text{exe time } y}{\text{exe time } x} = n$ ∴ $\text{Exe}_y = n\,\text{Exe}_x$

Cpu Execution time of a Program = Cpu clock cycle time × No of clock cycle required for a Program

| ie CPU capacity × no of clock cycles |

$\boxed{\text{clock rate} = \frac{1}{\text{clock cycle time}}}$

Eg: If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B

$\frac{\text{Exe}_y}{\text{Exe}_x} = \frac{15}{10} = 1.5$

$\boxed{\frac{\text{Perf } A}{\text{Perf } B} = \frac{\text{Exe } B}{\text{Exe } A} = n}$ ← for Comparison

∴ A = 1.5 times faster

Our favorite program runs in 10 seconds on computer A, which has a 2 GHz clock. We are trying to help a computer designer build a computer, B, which will run this program in 6 seconds. The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program. What clock rate should we tell the designer to target?

$\text{clock cycle time} = \frac{1}{\text{clock rate}}$

∴ Cpu execution time for a program = $\frac{\text{no of cpu clock cycle for a program}}{\text{clock rate}}$

⇒ ∴ $\text{CPU time}_A = \frac{N}{2 \times 10^9}$ N = no of cc required by program

$N = 10\cancel{s} \times 2 \times 10^9 \frac{\text{cycles}}{\cancel{\text{sec}}}$ $N = 2 \times 10^{10}$

for CPU B B will run in 6s but it will take more no of cpu cycles to complete same program

cpu rate B > CPU Rate A

Exec time A > Exe time B

Also Even if no of clock cycle of Prog in A < no of clock cycle of same Prog in B.

So B can complete even more no of cycles in comparatively less time.

$\text{CPU time}_B = \frac{N \times 1.2}{R}$ ∴ $6s = \frac{2 \times 10^{10} \times 1.2}{\text{Rate}(B)}$

$$CPU\ time_B = \frac{N \times 1.2}{Crate\ B}$$

$\therefore\ 6s = \frac{2 \times 10 \ldots}{Rate(B)}$

$$Clock\ Rate\ B = \frac{24 \times 10^{10}}{6}$$

$$0.4 \times 10^{10} = 4 \times 10^{9}$$

$= 4\ GHz$

Our favorite program runs in 10 seconds on computer A, which has a 2 GHz clock. We are trying to help a computer designer build a computer, B, which will run this program in 6 seconds. The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program. What clock rate should we tell the designer to target?

Understanding the Question:

① there is a Comp A (2GHz) which run a Prog in 10 sec

② We need to Get this program run in 6 sec in another Computer B

③ We can do this by increasing the clock rate [rate B > rate A]

④ Designer tell that if we increase the clock rate, No of clock cycles will also incre by 1.2x ie (in CompB it will take more no of clock cycles for the program)

CPI : clock cycles per instruction (No of clock cycle per instruction)

No of CPU clock cycles [Required for a Program] = No of instructions of a Program × Average no of clock cycles per instruction

$\boxed{\text{CPU clock} \ \ldots \text{ction of } \times CPI}$

[CPI : Avg clock cycle per instruction]

Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program, and computer B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program. Which computer is faster for this program and by how much

what we know

A : CPI : 2.0 [Avg CC/Ins]
time : 250ps

B : CPI = 1.2
time = 500ps

CPU clock cycle A = # inst. × CPI

when we ... time ...

CPU clock cycle A = # instr. × CPI
= I × 2.0

CPU clock cycle B = # instructions × ...
= I × 1.2

Prog is same ∴ $I_A = I_B = I$ (no of instructions)

Question: Which one is faster.

Execution time = No of clock cycles × clock cycle time ✓

[Execution time = No of CPU clock cycles × $\frac{1}{\text{clock rate}}$] ✓

time A = I × 2.0 × ~~250~~ ps = 500 I
time B = I × 1.2 × 500 ps = 600 I

$\frac{\text{perf A}}{\text{B}} = \frac{\text{time B}}{\text{time A}} = \frac{6}{5} = 1.2$

CPU clock cycle = clock cycle time : time taken to cover one clock cycle pulse (time)

CPU clock speed = clock rate : No of cpu clock cycles in per unit time (speed)

Since distance in always fixed

∴ [time = $\frac{1}{\text{Speed}}$]

Eg: there is an [Program] which has Avg 2 clock cycle / instruction

Program 1 { I1, I2, I3, I4 }

∴ CPI = 2, no of instructions = I

∴ How many cpu clock cycle?

No of clock cycles = I × 2.

Also Execution time of a Program = No of clock cycles for a Program (call it N) × [Clock cycle time]

Per clock cycle time

CPI : Avg no of clock cycle for Program / Prog freq
I = No of instructions
N : No of clock cycle for the program

∴ Execution time (Program) = I × 2 × 250

... clock

I = No of ...
N : No of clock cycles for the program

∴ Execution time (Program) = I x 2 x 250

total no of instructions; Avg clock cycle per instruction; cpu clock cycle time

total No of clock cycles for program; time per clock cycle

total time for the program

Execution time for the Entire program = I x 2 x 2.5

Suppose a program (or a program task) takes 1 billion instructions to execute on a processor running at 2 GHz. Suppose also that 50% of the instructions execute in 3 clock cycles, 30% execute in 4 clock cycles, and 20% execute in 5 clock cycles. What is the execution time for the program or task?

From <https://www.d.umn.edu/~gshute/arch/performance-equation.xhtml>

$I = 10^9$. Clock rate = 2×10^9 clock cycle/sec. Clock cycle time = $\frac{1}{2\times10^9}$ = 0.5×10^{-9}

$I_A = 5\times10^8$ ∴ $CPI_A = 3$ ... No of CPU clock cycle A: 15×10^8

$I_B = 3\times10^8$ $CPI_B = 4$ ... B: 12×10^8

$I_C = 2\times10^8$ ∴ $CPI_C = 5$ ... C: 10×10^8

⇒ 1.5, 1.2, 1.0 ⇒ 3.7×10^9 clock cycles Required

∴ total time = No of clock cycles × Cpu clock time = $3.7\times10^9 \times 0.5\times10^{-9}$ = 1.85

⇒ 1.85 seconds

Amdahl's law (or **Amdahl's argument**[1]) is a formula which gives the theoretical speedup in latency of the execution of a task at fixed workload that can be expected of a system whose resources are improved usually parallelism.

Amdahl's law is often used in parallel computing to predict the theoretical speedup when using multiple processors. For example, if a program needs 20 hours to complete using a single thread, but a one-hour portion of the program cannot be parallelized, therefore only the remaining 19 hours (p = 0.95) of execution time can be parallelized, then regardless of how many threads are devoted to a parallelized execution of this program, the minimum execution time cannot be less than one hour

the theoretical speedup is limited to at most 20 times the single thread performance, $\left(\frac{1}{1-p} = 20\right)$ p = Parallel Part of Program

$\left(\frac{1}{1-0.95}\right) = \frac{1}{0.05} = 20$ ✓

when we speedup part of a program (only some instructions) then overall speedup can be calculated as:

$$\text{Speed up} = \frac{1}{(1-\text{frac}_{enh}) + \frac{\text{Frac}_{enh}}{\text{Speedup}_{enh}}}$$

Frac enh ← fraction of Program that is enhanced

Speedup enh ← Speedup that we achieved

on the enhancement

$$\text{Speedup} = \frac{\text{Performance After enhancement}}{\text{performance before enhancement}}$$

If 90% of a program is speeded up to run 10 times faster what is overall speedup ?

frac enh = 0.9   Speedup enh = 10

$$\text{Overall Speedup} = \frac{1}{(1-0.9)+\left(\frac{0.9}{10}\right)}$$

$$\Rightarrow \frac{1}{0.10 + .09}$$

$$\Rightarrow \frac{1}{0.19} = \underline{5.26}$$

$\boxed{\text{Speedup} = 5.26}$

Making 4/5th of a program run 20% faster

frac enh = 4/5 = 0.8 or 80%

Speedup enh = 20% ie 1.2

$$\text{Overall Speedup} = \frac{1}{(1-0.8)+\left(\frac{0.8}{1.2}\right)}$$

$$\Rightarrow \frac{1}{0.2+2/3} = \frac{1}{0.866}$$

$$\Rightarrow \underline{1.53}$$

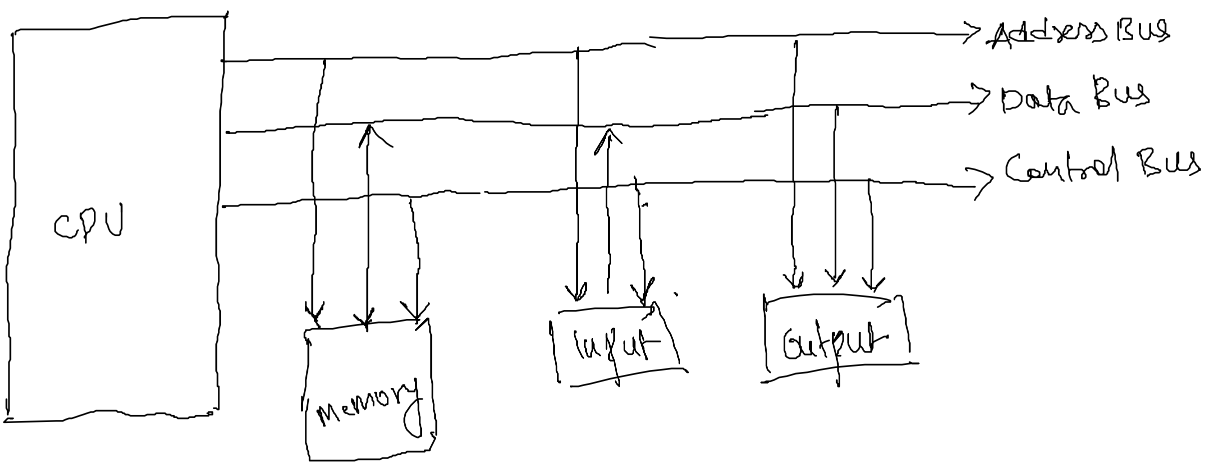
Memory Read instruction

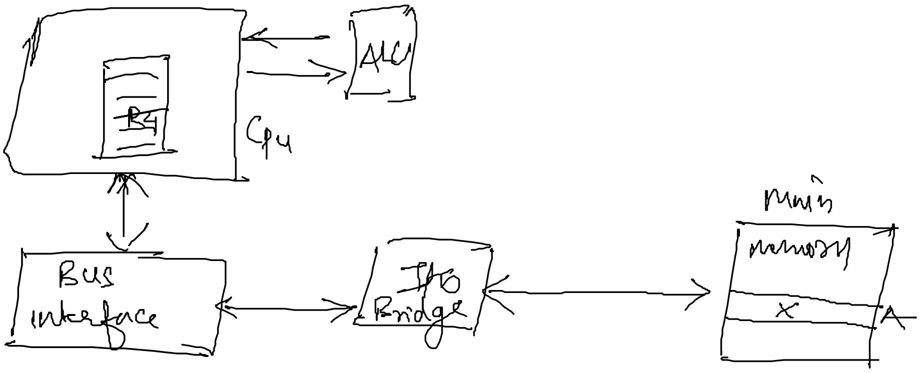
① CPU places ① Address A ② Read Control Signal } on memory bus   R4 ← [A]

on Register4 ⇐ load [contents of Address A]

② Main memory reads A from memory bus, retrieves word X and places it on the bus.

③ CPU reads the word from the bus, copies it to the Register R4





MOV [A] ← R4

Write operation

(1) CPU places Address A and WRITE control signal on the bus (control bus)

(2) Main memory then reads them and wait for the data word to arrive.

(3) CPU then places the data word Y on the bus.

(4) main memory then reads it from the data bus and stores it on Address A

Y →