Computer Organization and Architecture (CPE343)

Dr. Muhammad Naeem Awais
naeem.awais@cuilahore.edu.pk

CPU Performance

- Performance Matrices
- CPU Performance and Its Factors
- Instruction Performance
- Classic CPU Performance Equation
- Performance of Multicycle Datapath
- Performance of Pipelined Datapath

Performance metrics

- Execution time and throughput are usually linked
  - A faster processor will improve both
  - More processors will likely improve only throughput
- For some program running on machine X

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

- When we compare two computers: “X is n times faster than Y”

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

Relative Performance

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?

Relative Performance

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?

We know that A is *n* times as fast as B if

$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = n$$

Thus the performance ratio is

$$\frac{15}{10} = 1.5$$

and A is therefore 1.5 times as fast as B.

In the above example, we could also say that computer B is 1.5 times *slower than* computer A, since

$$\frac{\text{Performance}_A}{\text{Performance}_B} = 1.5$$

means that

$$\frac{\text{Performance}_A}{1.5} = \text{Performance}_B$$

CPU Performance and Its Factors

- Confining to CPU performance at this point is ***CPU Execution Time***.
- A simple formula relates the most basic metrics (clock cycles and clock cycle time) to CPU time:

$$\begin{array}{c}\text{CPU execution time}\\\text{for a program}\end{array} = \begin{array}{c}\text{CPU clock cycles}\\\text{for a program}\end{array} \times \text{Clock cycle time}$$

Alternatively, because clock rate and clock cycle time are inverses,

$$\begin{array}{c}\text{CPU execution time}\\\text{for a program}\end{array} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

Improving Performance

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?

Improving Performance

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?

Let's first find the number of clock cycles required for the program on A:

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

$$10 \text{ seconds} = \frac{\text{CPU clock cycles}_A}{2 \times 10^9 \dfrac{\text{cycles}}{\text{second}}}$$

$$\text{CPU clock cycles}_A = 10 \text{ seconds} \times 2 \times 10^9 \frac{\text{cycles}}{\text{second}} = 20 \times 10^9 \text{ cycles}$$

CPU time for B can be found using this equation:

$$\text{CPU time}_B = \frac{1.2 \times \text{CPU clock cycles}_A}{\text{Clock rate}_B}$$

$$6 \text{ seconds} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = \frac{0.2 \times 20 \times 10^9 \text{ cycles}}{\text{second}} = \frac{4 \times 10^9 \text{ cycles}}{\text{second}} = 4 \text{ GHz}$$

To run the program in 6 seconds, B must have twice the clock rate of A.

Instruction Performance

- The performance equations above did not include any reference to the number of instructions needed for the program.
- However, since the compiler clearly generated instructions to execute
- One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction.
- Therefore, the number of clock cycles required for a program can be written as

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \begin{array}{c}\text{Average clock cycles}\\ \text{per instruction}\end{array}$$

- The term clock cycles per instruction, which is the average number of clock cycles each instruction takes to execute, is often abbreviated as CPI.

Using the Performance Equation

Suppose we have two implementations of the same instruction set architecture. 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?

Using the Performance Equation

Suppose we have two implementations of the same instruction set architecture. 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?
We know that each computer executes the same number of instructions for the program; let's call this number *I*. First, find the number of processor clock cycles for each computer:

$$\text{CPU clock cycles}_A = I \times 2.0$$
$$\text{CPU clock cycles}_B = I \times 1.2$$

Now we can compute the CPU time for each computer:

$$\begin{aligned}\text{CPU time}_A &= \text{CPU clock cycles}_A \times \text{Clock cycle time} \\ &= I \times 2.0 \times 250\text{ ps} = 500 \times I\text{ ps}\end{aligned}$$

Likewise, for B:

$$\text{CPU time}_B = I \times 1.2 \times 500\text{ ps} = 600 \times I\text{ ps}$$

Clearly, computer A is faster. The amount faster is given by the ratio of the execution times:

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

We can conclude that computer A is 1.2 times as fast as computer B for this program.

The Classic CPU Performance Equation

- We can now write this basic performance equation in terms of instruction count (the number of instructions executed by the program), CPI, and clock cycle time:

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

- or, since the clock rate is the inverse of clock cycle time:

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

Comparing Code Segments

A compiler designer is trying to decide between two code sequences for a particular computer. The hardware designers have supplied the following facts:

	CPI for each instruction class		
	A	B	C
CPI	1	2	3

For a particular high-level language statement, the compiler writer is considering two code sequences that require the following instruction counts:

	Instruction counts for each instruction class		
Code sequence	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

Comparing Code Segments

A compiler designer is trying to decide between two code sequences for a particular computer. The hardware designers have supplied the following facts:

	CPI for each instruction class		
	A	B	C
CPI	1	2	3

For a particular high-level language statement, the compiler writer is considering two code sequences that require the following instruction counts:

	Instruction counts for each instruction class		
Code sequence	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

Sequence 1 executes 2 + 1 + 2 = 5 instructions. Sequence 2 executes 4 + 1 + 1 = 6 instructions. Therefore, sequence 1 executes fewer instructions.

We can use the equation for CPU clock cycles based on instruction count and CPI to find the total number of clock cycles for each sequence:

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{C}_i)$$

This yields

$$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ cycles}$$

$$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ cycles}$$

So code sequence 2 is faster, even though it executes one extra instruction. Since code sequence 2 takes fewer overall clock cycles but has more instructions, it must have a lower CPI. The CPI values can be computed by

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}}$$

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2.0$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

Example

- Consider a processor operates at a frequency of 1 GHz and is running a code that executes 1 billion instructions from beginning to end. The instruction mix of this code is 37% branches, 15% loads, 10% stores, and rest are ALU. The average CPI is 1.3 for branches, 0.7 for loads, 2 for stores, and 5 for ALU instructions. ***Compute*** the total execution time for this code on this processor.

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

Solution

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

$$\text{CPU Execution Time} = \left(\sum_{i=1}^{n} IC_i \times CPI_i\right) \times \text{Clock Cycle Time}$$

$$= (0.37 \times 1.3 + 0.15 \times 0.7 + 0.10 \times 2 + 0.38 \times 5) \times 1ns \times 1B$$

$$= (0.481 + 0.105 + 0.2 + 1.9) \times 1ns \times 1B$$

$$= 2.686 \times 1ns \times 1B$$

$$= 2.686 ns \times 1 \times 10^9$$

$$= 2.686 s$$

Example (Do it yourself)

- You are on the design team for a new processor. The clock of the processor runs at 200 MHz. The following table gives instruction frequencies for Benchmark B, as well as how many cycles the instructions take, for the different classes of instructions. For this problem, we assume that (unlike many of today's computers) the processor only executes one instruction at a time. ***Compute*** the total execution time for this code on this processor.

Instruction Type	Frequency	Cycles	Instruction Type
Loads & Stores	30%	6 cycles	Loads & Stores
Arithmetic Instructions	50%	4 cycles	Arithmetic Instructions
All Others	20%	3 cycles	All Others

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

Example (Do it yourself)

- 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?

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

Example (Do it yourself)

- Suppose the processor in the previous example is redesigned so that all instructions that initially executed in 5 cycles now execute in 4 cycles. Due to changes in the circuitry, the clock rate has to be decreased from 2.0 GHz to 1.9 GHz. No changes are made to the instruction set. What is the overall percentage improvement?

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

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

Example

Consider the unpipelined processor in the previous section. Assume that it has a 1 ns clock cycle and that it uses 4 cycles for ALU operations and branches and 5 cycles for memory operations. Assume that the relative frequencies of these operations are 40%, 20%, and 40%, respectively. Suppose that due to clock skew and setup, pipelining the processor adds 0.2 ns of overhead to the clock. Ignoring any latency impact, how much speedup in the instruction execution rate will we gain from a pipeline?

$$\text{Speedup from pipelining} = \frac{\text{Average instruction time unpipelined}}{\text{Average instruction time pipelined}}$$

The average instruction execution time on the unpipelined processor is

$$\text{Average instruction execution time} = \text{Clock cycle} \times \text{Average CPI}$$

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

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

Solution

The average instruction execution time on the unpipelined processor is

$$\begin{aligned}\text{Average instruction execution time} &= \text{Clock cycle} \times \text{Average CPI}\\ &= 1\ \text{ns} \times [(40\% + 20\%) \times 4 + 40\% \times 5]\\ &= 1\ \text{ns} \times 4.4\\ &= 4.4\ \text{ns}\end{aligned}$$

In the pipelined implementation, the clock must run at the speed of the slowest stage plus overhead, which will be 1 + 0.2 or 1.2 ns; this is the average instruction execution time. Thus, the speedup from pipelining is

$$\begin{aligned}\text{Speedup from pipelining} &= \frac{\text{Average instruction time unpipelined}}{\text{Average instruction time pipelined}}\\ &= \frac{4.4\ \text{ns}}{1.2\ \text{ns}} = 3.7\ \text{times}\end{aligned}$$

The 0.2 ns overhead essentially establishes a limit on the effectiveness of pipelining. If the overhead is not affected by changes in the clock cycle, Amdahl's law tells us that the overhead limits the speedup.

Example

The time delay of a 4-segment pipeline in different segments are t1=35 ns, t2 = 30 ns, t3 = 40 ns, t4=45 ns. The interface register delay time is t=5 ns. How long would it take to complete 100 instructions in the pipeline ? (Assume there is no dependency between the instructions).

Example

Consider a multicycle unpipelined architecture that operates at 1.5 GHz frequency and takes 5 clock cycles to complete all ALU operations, 4 clock cycles to complete branch and memory instructions. The frequencies of these operations are 50%, 35% and 15% respectively. Out of all ALU & memory instruction only 5% are load-ALU combinations. As an architect, you are assigned a task to evaluate the speedup of pipeline design over its unpipelined counterpart before making this a 5-stage pipelined architecture. The interfacing pipeline registers take extra 0.2 ns as a pipeline overhead. In this new design all load-ALU combinations cause a stall of 1 cycle, 10% of branch instructions cause a stall of 2 cycles and 20% memory instructions cause a stall of 30 cycles.

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

Pipeline CPI = Ideal pipeline CPI + Structural stalls + Data hazard stalls + Control stalls

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

Pipeline CPI = Ideal pipeline CPI + Structural stalls + Data hazard stalls + Control stalls

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

Instructi Executi Time$_{\text{unpipelined}}$ = CPI × cct

$= (5\times0.5 + 4\times0.35 + 4\times0.15)\times 0.667\,ns$

$= (2.5 + 1.4 + 0.6)\times 0.667 = 4.5\times 0.667\,ns$

$= 3\,ns$

Effective CPI$_{\text{pipelined}}$ = Base CPI + Stalls

$= 1. \quad + (30\times0.2\times0.15 + 2\times0.1\times0.35 + 1\times1\times0.05)$

$= 1 + (0.9 + 0.07 + 0.05)$

$= 1 + 1.02 = 2.02$

Instructi Executi Time$_{\text{pipelined}}$ = CPI × cct

due to the register pipeli overhead

$= 2.02 \times 0.667\,ns = 1.48\,ns + 0.2 = 1.50\,ns$

Speed up $= 3/1.50 = 2$ times