**Computer Organization and Architecture | Pipelining | Set 1 (Execution, Stages and Throughput)**

* Difficulty Level : [Easy](https://www.geeksforgeeks.org/easy/)
* Last Updated : 28 Jun, 2021

To improve the performance of a CPU we have two options:  
1) Improve the hardware by introducing faster circuits.  
2) Arrange the hardware such that more than one operation can be performed at the same time.

Since, there is a limit on the speed of hardware and the cost of faster circuits is quite high, we have to adopt the 2nd option.

Attention reader! Don’t stop learning now. Get hold of all the important CS Theory concepts for SDE interviews with the [**CS Theory Course**](https://practice.geeksforgeeks.org/courses/SDE-theory?vC=1) at a student-friendly price and become industry ready.

**Pipelining :** Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased. Simultaneous execution of more than one instruction takes place in a pipelined processor.

Let us see a real life example that works on the concept of pipelined operation. Consider a water bottle packaging plant. Let there be 3 stages that a bottle should pass through, Inserting the bottle(**I**), Filling water in the bottle(**F**), and Sealing the bottle(**S**). Let us consider these stages as stage 1, stage 2 and stage 3 respectively. Let each stage take 1 minute to complete its operation.  
Now, in a non pipelined operation, a bottle is first inserted in the plant, after 1 minute it is moved to stage 2 where water is filled. Now, in stage 1 nothing is happening. Similarly, when the bottle moves to stage 3, both stage 1 and stage 2 are idle. But in pipelined operation, when the bottle is in stage 2, another bottle can be loaded at stage 1. Similarly, when the bottle is in stage 3, there can be one bottle each in stage 1 and stage 2. So, after each minute, we get a new bottle at the end of stage 3. Hence, the average time taken to manufacture 1 bottle is :

**Without pipelining** = 9/3 minutes = 3m

I F S | | | | | |

| | | I F S | | |

| | | | | | I F S (9 minutes)

**With pipelining** = 5/3 minutes = 1.67m

I F S | |

| I F S |

| | I F S (5 minutes)

Thus, pipelined operation increases the efficiency of a system.

**Design of a basic pipeline**

* In a pipelined processor, a pipeline has two ends, the input end and the output end. Between these ends, there are multiple stages/segments such that output of one stage is connected to input of next stage and each stage performs a specific operation.
* Interface registers are used to hold the intermediate output between two stages. These interface registers are also called latch or buffer.
* All the stages in the pipeline along with the interface registers are controlled by a common clock.

**Execution in a pipelined processor**  
Execution sequence of instructions in a pipelined processor can be visualized using a space-time diagram. For example, consider a processor having 4 stages and let there be 2 instructions to be executed. We can visualize the execution sequence through the following space-time diagrams:

**Non overlapped execution:**

| Stage / Cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | I1 |  |  |  | I2 |  |  |  |
| S2 |  | I1 |  |  |  | I2 |  |  |
| S3 |  |  | I1 |  |  |  | I2 |  |
| S4 |  |  |  | I1 |  |  |  | I2 |

Total time = 8 Cycle

**Overlapped execution:**

| Stage / Cycle | 1 | 2 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- | --- |
| S1 | I1 | I2 |  |  |  |
| S2 |  | I1 | I2 |  |  |
| S3 |  |  | I1 | I2 |  |
| S4 |  |  |  | I1 | I2 |

Total time = 5 Cycle

**Pipeline Stages**

RISC processor has 5 stage instruction pipeline to execute all the instructions in the RISC instruction set. Following are the 5 stages of RISC pipeline with their respective operations:

* **Stage 1 (Instruction Fetch)**  
  In this stage the CPU reads instructions from the address in the memory whose value is present in the program counter.
* **Stage 2 (Instruction Decode)**  
  In this stage, instruction is decoded and the register file is accessed to get the values from the registers used in the instruction.
* **Stage 3 (Instruction Execute)**  
  In this stage, ALU operations are performed.
* **Stage 4 (Memory Access)**  
  In this stage, memory operands are read and written from/to the memory that is present in the instruction.
* **Stage 5 (Write Back)**  
  In this stage, computed/fetched value is written back to the register present in the instructions.

**Performance of a pipelined processor**  
Consider a ‘k’ segment pipeline with clock cycle time as ‘Tp’. Let there be ‘n’ tasks to be completed in the pipelined processor. Now, the first instruction is going to take ‘k’ cycles to come out of the pipeline but the other ‘n – 1’ instructions will take only ‘1’ cycle each, i.e, a total of ‘n – 1’ cycles. So, time taken to execute ‘n’ instructions in a pipelined processor:

ETpipeline = k + n – 1 cycles

= (k + n – 1) Tp

In the same case, for a non-pipelined processor, execution time of ‘n’ instructions will be:

ETnon-pipeline = n \* k \* Tp

So, speedup (S) of the pipelined processor over non-pipelined processor, when ‘n’ tasks are executed on the same processor is:

S = Performance of pipelined processor /

Performance of Non-pipelined processor

As the performance of a processor is inversely proportional to the execution time, we have,

S = ETnon-pipeline / ETpipeline

=> S = [n \* k \* Tp] / [(k + n – 1) \* Tp]

S = [n \* k] / [k + n – 1]

When the number of tasks ‘n’ are significantly larger than k, that is, n >> k

S = n \* k / n

S = k

where ‘k’ are the number of stages in the pipeline.

Also,**Efficiency** = Given speed up / Max speed up = S / Smax  
We know that, Smax = k

So, **Efficiency** = S / k

**Throughput** = Number of instructions / Total time to complete the instructions

So, **Throughput** = n / (k + n – 1) \* Tp

Note: The cycles per instruction (CPI) value of an ideal pipelined processor is 1

Please see [**Set 2**](https://www.geeksforgeeks.org/computer-organization-and-architecture-pipelining-set-2-dependencies-and-data-hazard/) for Dependencies and Data Hazard and [**Set 3**](https://www.geeksforgeeks.org/computer-organization-and-architecture-pipelining-set-3-types-and-stalling/) for Types of pipeline and Stalling.