

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

Last Updated : 13 Sep, 2024  
Pipelining is a technique used in modern processors to improve performance by executing multiple instructions simultaneously. It breaks down the execution of instructions into several stages, where each stage completes a part of the instruction. These stages can overlap, allowing the processor to work on different instructions at various stages of completion, similar to an assembly line in manufacturing.

In this article, you will get a detailed overview of Pipeline in Computer Organization and Architecture.

Table of Content

What is Pipelining?  
What is Throughout?  
What is Latenecy?  
Advantages of Pipelining  
Disadvantages of Pipelining  
What is Pipelining?  
Pipelining is an arrangement of the CPU’s hardware components to raise the CPU’s general performance. In a pipelined processor, procedures called ‘stages’ are accomplished in parallel, and the execution of more than one line of instruction occurs. Now let us look at a real-life example that should operate based on the pipelined operation concept. Consider a water bottle packaging plant. For this case, let there be 3 processes that a bottle should go through, ensing the bottle(I), Filling water in the bottle(F), Sealing the bottle(S).

It will be helpful for us to label these stages as stage 1, stage 2, and stage 3. Let each stage take 1 minute to complete its operation. Now, in a non-pipelined operation, a bottle is first inserted in the plant, and after 1 minute it is moved to stage 2 where water is filled. Now, in stage 1 nothing is happening. Likewise, when the bottle is in stage 3 both stage 1 and stage 2 are inactive. But in pipelined operation, when the bottle is in stage 2, the bottle in stage 1 can be reloaded. In the same way, during the bottle 3 there could be one bottle in the 1st and 2nd stage accordingly. Therefore at the end of stage 3, we receive a new bottle for every minute. Hence, the average time taken to manufacture 1 bottle is:

Therefore, the average time intervals of manufacturing each 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 the output of one stage is connected to the input of the 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 the RISC pipeline with their respective operations:

Stage 1 (Instruction Fetch): In this stage the CPU fetches the instructions from the address present in the memory location whose value is stored in the program counter.  
Stage 2 (Instruction Decode): In this stage, the instruction is decoded and register file is accessed to obtain the values of registers used in the instruction.  
Stage 3 (Instruction Execute): In this stage some of activities are done such as ALU operations.  
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, the execution time of ‘n’ instructions will be:

ETnon-pipeline = n \* k \* Tp

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

S = Performance of non-pipelined processor /  
 Performance of 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’ is 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 for Dependencies and Data Hazard and Set 3 for Types of pipeline and Stalling.

Performance of pipeline is measured using two main metrices as Throughput and latency.

What is Throughout?  
It measure number of instruction completed per unit time.  
It represents overall processing speed of pipeline.  
Higher throughput indicate processing speed of pipeline.  
Calculated as, throughput= number of instruction executed/ execution time.  
It can be affected by pipeline length, clock frequency. efficiency of instruction execution and presence of pipeline hazards or stalls.  
What is Latenecy?  
It measure time taken for a single instruction to complete its execution.  
It represents delay or time it takes for an instruction to pass through pipeline stages.  
Lower latency indicates better performance .  
It is calculated as, Latency= Execution time/ Number of instruction executed.  
It in influenced by pipeline length, depth, clock cycle time, instruction dependencies and pipeline hazards.  
Advantages of Pipelining  
Increased Throughput: Pipelining enhance the throughput capacity of a CPU and enables a number of instruction to be processed at the same time at different stages. This leads to the improvement of the amount of instructions accomplished in a given period of time, thus improving the efficiency of the processor.  
Improved CPU Utilization: From superimposing of instructions, pipelining helps to ensure that different sections of the CPU are useful. This gives no time for idling of the various segments of the pipeline and optimally utilizes hardware resources.  
Higher Instruction Throughput: Pipelining occurring because when one particular instruction is in the execution stage it is possible for other instructions to be at varying stages of fetch, decode, execute, memory access, and write-back. In this manner there is concurrent processing going on and the CPU is able to process more number of instructions in a given time frame than in non pipelined processors.  
Better Performance for Repeated Tasks: Pipelining is particularly effective when all the tasks are accompanied by repetitive instructions, because the use of the pipeline shortens the amount of time each task takes to complete.  
Scalability: Pipelining is RSVP implemented in different types of processors hence it is scalable from simple CPU’s to an advanced multi-core processor.  
Disadvantages of Pipelining  
Pipeline Hazards: Pipelining may result to data hazards whereby instructions depends on other instructions; control hazards, which arise due to branch instructions; and structural hazards whereby there are inadequate hardware facilities. Some of these hazards may lead to delays hence tough strategies to manage them to ensure progress is made.  
Increased Complexity: Pipelining enhances the complexity of processor design as well as its application as compared to non-pipelined structures. Pipelining stages management, dealing with the risks and correct instruction sequence contribute to the design and control considerations.  
Stall Cycles: When risks are present, pipeline stalls or bubbles can be brought about, and this produces idle times in certain stages in the pipeline. These stalls can actually remove some of the cycles acquired by pipelining, thus reducing the latter’s efficiency.  
Instruction Latency: While pipelining increases the throughput of instructions the delay of each instruction may not necessarily be reduced. Every instruction must still go through all the pipeline stages and the time it takes for a single instruction to execute can neither reduce nor decrease significantly due to overheads.  
Hardware Overhead: It increases the complexity in designing the pipelining due to the presence of pipeline registers and the control logic used in managing the pipe stages and the data. This not only increases the cost of the wares but also forces integration of more complicated, and thus costly, hardware.  
Conclusion  
Pipelining is one of the most essential concepts and it improves CPU’s capability to process several instructions at the same time across various stages. It increases immensely the system’s throughput and overall efficiency by effectively determining the optimum use of hardware. On its own it enhances the processing speed but handling of pipeline hazards is critical for enhancing efficiency. It is thus crucial for any architect developing systems that will support HPC to have a war chest of efficient pipelining strategies that they can implement. can you summarize this and also higlight main keywords

Pipelining is a performance-enhancing technique in processors that allows simultaneous execution of multiple instructions by dividing them into sequential stages, similar to an assembly line. Here's a concise summary with **key concepts** highlighted:

**Key Summary**

1. **Pipelining Basics**
   * Breaks instruction execution into **stages** (e.g., fetch, decode, execute, memory access, write-back in a **5-stage RISC pipeline**).
   * Overlaps stages to maximize hardware utilization, reducing average instruction completion time.
   * Example: A bottling plant with 3 stages (inserting, filling, sealing) shows pipelining reduces average time per bottle from 3 minutes (non-pipelined) to ~1.67 minutes[[1]](#fn1).
2. **Performance Metrics**
   * **Throughput**: Instructions completed per unit time. Formula:where = stages, = instructions, = clock cycle time[[1]](#fn1).
   * **Latency**: Time for a single instruction to complete. Formula:
   * **Speedup**: Ratio of non-pipelined to pipelined execution time. For large :

[[1]](#fn1).

1. **Advantages**
   * **Increased throughput** via parallel stage processing.
   * **Improved CPU utilization** by minimizing idle hardware.
   * **Scalability** across processor designs (e.g., multi-core systems)[[1]](#fn1).
2. **Disadvantages**
   * **Pipeline hazards**: Data, control, and structural hazards cause stalls.
   * **Increased complexity** in design and hazard management.
   * **Hardware overhead** from pipeline registers and control logic[[1]](#fn1).

**Highlighted Keywords**

**Pipelining**, **stages**, **throughput**, **latency**, **speedup**, **RISC pipeline**, **hazards**, **stalls**, **clock cycle**, **hardware utilization**.

This technique is fundamental in modern processors but requires careful handling of hazards to maintain efficiency. For deeper insights, explore dependencies (Set 2) and pipeline types (Set 3)[[1]](#fn1).

⁂

1. <https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/48911144/88a38ddd-9113-43c1-bb4a-edf5ae605b64/paste.txt>