**TASK 3 PIPELINE PROCESSOR DESIGN**

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.

**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](https://www.geeksforgeeks.org/advanced-risc-machine-arm-processor/) 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](https://www.geeksforgeeks.org/what-are-the-functions-of-a-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](https://www.geeksforgeeks.org/difference-between-alu-and-cu/) 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](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.

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