Kunal Sachdeva

4th Semester CSE

19115045

CSA TERM PAPER

PIPELINE CLOCKING

Introduction:

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.

Pipelining is a technique where multiple instructions are overlapped during execution. Pipeline is divided into stages and these stages are connected with one another to form a pipe like structure. Instructions enter from one end and exit from another end. Thus, pipelined operation increases the efficiency of a system.

Procedure:

- 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.

Pipeline clocking is cycles per instruction (aka clock cycles per instruction, clocks per instruction, or CPI) is one aspect of a processor's performance: the average number of clock cycles per instruction for a program or program fragment. Each stage requires one clock cycle and an instruction passes through the stages sequentially.

Without pipelining, in a multi-cycle processor, a new instruction is fetched in stage 1 only after the previous instruction finishes at stage 5, therefore the number of clock cycles it takes to execute an instruction is five (CPI = 5 > 1). In this case, the processor is said to be sub scalar.

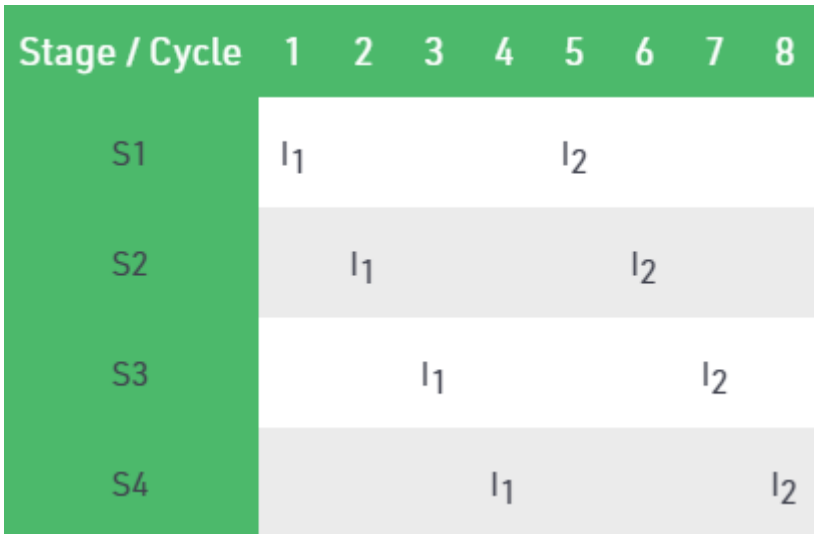
With pipelining, a new instruction is fetched every clock cycle, therefore, since one could theoretically have five instructions in the five pipeline stages at once (one instruction per stage), a different instruction would complete stage 5 in every clock cycle and on average the number of clock cycles it takes to execute an instruction

is 1 (CPI = 1). **19115045**

Execution in a pipelined processor:

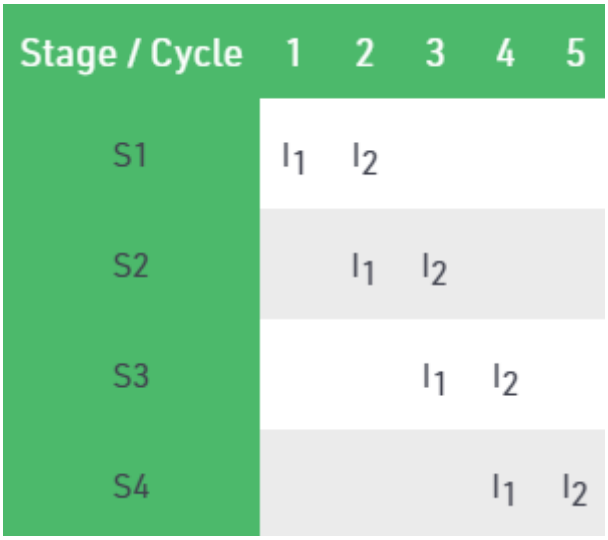
Execution sequence of instructions in a pipelined processor can be visualized using a space-time diagram. For example, consider a processor having 4 stages and let there be 2 instructions to be executed. We can visualize the execution sequence through the following space-time diagrams:

Non overlapped execution:



Total time = 8 Cycle

Overlapped execution:



Total time = 5 Cycle

Pipeline Stages: **19115045**

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.

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:

$$ET_{pipeline} = k + n - 1 \text{ cycles}$$

$$= (k + n - 1)\, Tp$$

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

$$ET_{non\text{-}pipeline} = n * k * Tp$$

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

$$S = ET_{non\text{-}pipeline} / ET_{pipeline}$$

$$S = [n * k * Tp] / [(k + n - 1) * Tp]$$

$$S = [n * k] / [k + n - 1]$$

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

$$S = n * k / n$$

$$S = k$$

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