Computer Science Dept. / 2nd year 2nd Course (2024-2025)
Computer Architecture
Dr.Haleema Essa

Parallel Processing and Pipelining

- ***Parallel processing*** is a term used to denote a large class of techniques that are used to provide simultaneous data processing tasks for the purpose of increasing the ***computational speed*** of a computer system.
- concurrent data processing to achieve ***faster execution***.

The purpose:

1. Speed up the computer processing capability
2. Increase the throughput

Throughput: the amount of processing that can be accomplished during a given interval of time.

The side effects (Disadvantages)

1. The amount of hardware increases
2. The cost of the system increases

Parallel processing / levels of complexity

- ***At the lower level***

Serial Shift register or parallel load registers

- ***At the higher level***

Multiplicity of functional units that performer identical or different operations simultaneously.

There are *two major factors* used to categorize such system:

1- Processing Units: processing units can communicate and interact with each other using either ***shared memory*** or ***message passing*** methods.

2- Interconnection Network: interconnection network for shared memory system can be classified as ***bus-based*** versus ***switch-based***.

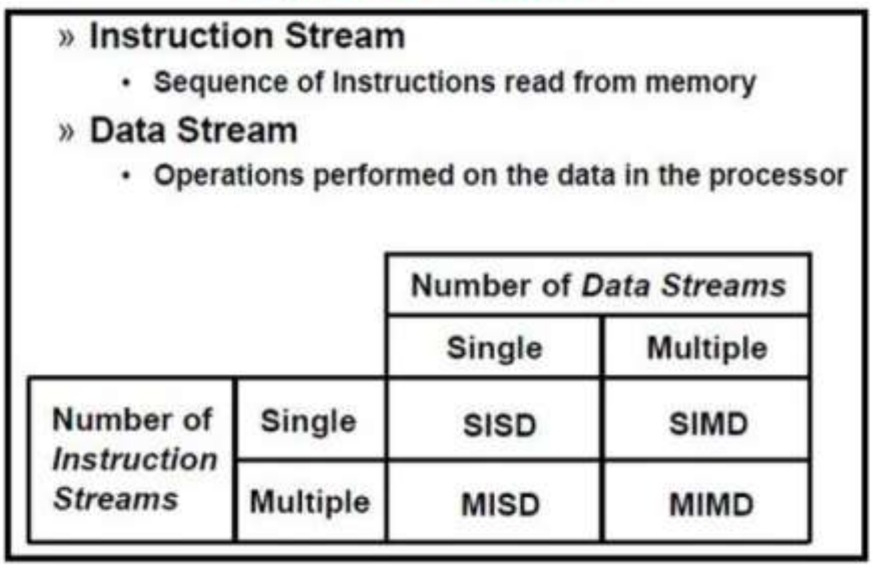
* In message passing systems, the inter connection network is divided into ***static*** and ***dynamic***.

1- Static Connection: It has a fixed topology that does not change while the programs are running.

2- Dynamic Connection: It has a changed topology as the program executes.

- Computer architecture can be classified into the following four distinct categories:

Flynn's classification



» Instruction Stream
- Sequence of Instructions read from memory

» Data Stream
- Operations performed on the data in the processor

		Number of *Data Streams*	
		Single	Multiple
Number of *Instruction Streams*	Single	SISD	SIMD
	Multiple	MISD	MIMD

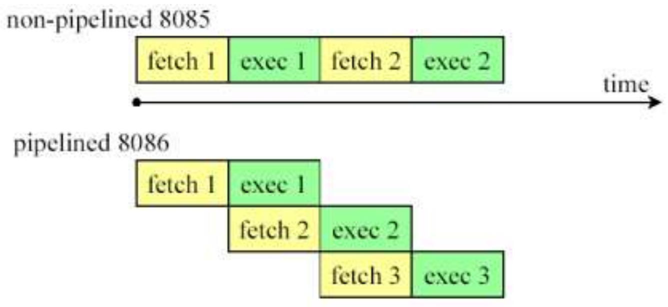
Pipelining: organizes the execution of the multiple instructions simultaneously. Pipelining improves the throughput of the system. In pipelining, the instruction is divided into the subtasks. Each subtask performs the dedicated task. An instruction in a process is divided into **5** subtasks likely,

Instruction Fetch	Instruction Decode	Operand Fetch	Instruction Execute	Operand Store

The instruction steps in process consist of:
1. In the first subtask, the instruction is fetched.
2. The fetched instruction is decoded in the second stage.
3. In the third stage, the operands of the instruction are fetched.

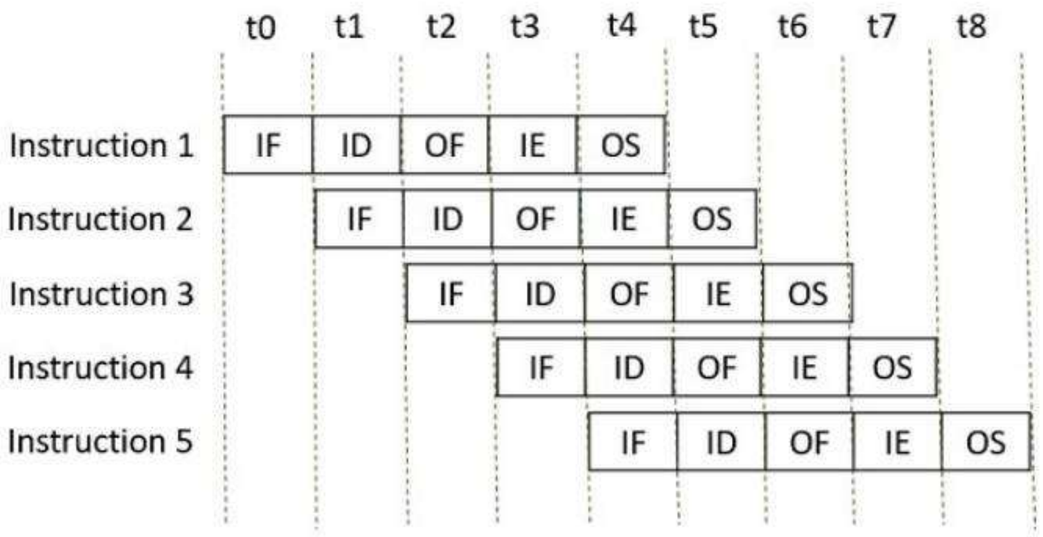
4. In the fourth, arithmetic and logical operation are performed on the operands to execute the instruction.
5. In the fifth stage, the result is stored in memory.

Pipelining: is a process that allows the CPU (Microprocessor) to fetch and execute instructions at the same time. Intel Co. implemented the concept of pipelining by splitting the internal architecture of the 8088/8086μP into two units that works simultaneously:



• Cycle time of the pipelining process

The figure below gives 5 instructions are pipelined. The first instruction gets completed in 5 clock cycle. After the completion of first instruction, in every new clock cycle, a new instruction completes its execution.


t0 t1 t2 t3 t4 t5 t6 t7 t8
Instruction 1: IF ID OF IE OS
Instruction 2: IF ID OF IE OS
Instruction 3: IF ID OF IE OS
Instruction 4: IF ID OF IE OS
Instruction 5: IF ID OF IE OS

Pipelining of 5 Instructions

Observe that when the Instruction fetch operation of the first instruction is completed in the next clock cycle the instruction fetch of second instruction gets started. This way the hardware never sits idle it is always busy in performing some or other operation. But, **NO** two instructions can **execute** their **same stage** at the **same clock cycle**.

Speed Calculation

- **K-segment** pipeline with a **clock cycle time (tp)** is used to execute **n** tasks.
- The first task **T1** requires a time equal to **(k*tp)** to complete its operation because we have **k** segments in the pipe.
- The remaining **n - 1** tasks emerge from the pipe at the rate of one task per clock cycle and they will be completed after a time equal to **(n - 1)*tp**.
- Therefore, to complete **n tasks** using a **k-segment** pipeline:

$$k + (n - 1) \text{ clock cycles}$$

For example, a system with **four** segments and **six** tasks.

The time required to complete all the operations is:

$$4 + (6-1) = 9 \text{ clock cycles}$$

NONPIPELINE UNIT

- Each task takes time equal to **tn**.
- The total time required for **n** tasks is

$$n * tn$$

PIPELINE UNIT

- To complete **n** tasks using a **k-segment** pipeline:

$$k + (n - 1) \text{ clock cycles}$$

- Total time is:

$$k + (n - 1) * tp$$

- SPEED UP (**S**):

$$S = n * tn / (k + n - 1) * tp$$

Example

- **4**-stage pipeline
- suboperation in each stage; **tp** = 20nS
- **100** tasks to be executed
- each task in a non-pipelined system require(**tn**) 80 ns

Pipelined System

(k + n - 1) * tp = (4 + 99) * 20 = 2060nS

Non-Pipelined System

n * tn = 100 * 80 = 8000nS

Speedup

S_k = 8000 / 2060 = 3.88

Types of Pipelining:

*Pipeline processors can be classified depending on their **functionality**.*

1. Arithmetic Pipelining

It is designed to perform high-speed floating-point addition, multiplication, and division. Here, the multiple arithmetic logic units are built in the system to perform the parallel arithmetic computation in various data format. ***Examples*** of the arithmetic pipelined processor are Star-100, TI-ASC, Cray-1, Cyber-205.

2. Instruction Pipelining

Here, the number of instruction are pipelined and the execution of current instruction is overlapped by the execution of the subsequent instruction. It is also called ***instruction look ahead***.

3. Processor Pipelining

Here, the processors are pipelined to process the same data stream. The data stream is processed by the first processor and the result is stored in the memory block. The result in the memory block is accessed by the second processor. The second processor reprocesses the result obtained by the first processor and the passes the refined result to the third processor and so on.

4. Unifunction Vs. Multifunction Pipelining

The pipeline performing the precise function every time is unifunctional pipeline. On the other hand, the pipeline performing multiple functions at a different time or multiple functions at the same time is multifunction pipeline.

Advantages

1. Pipelining improves the throughput of the system.
2. In every clock cycle, a new instruction finishes its execution.
3. Allow multiple instructions to be executed concurrently.

Disadvantages

1. Pipelining divides the instruction in 5 stages instruction fetch, instruction decode, operand fetch, instruction execution and operand store.
2. The pipeline allows the execution of multiple instructions concurrently with the limitation that no two instructions would be executed at the **same stage** in the **same clock cycle**.
3. All the stages must process at equal speed else the slowest stage would become the bottleneck.
4. Whenever a pipeline has to ***stall*** for any reason it is a ***pipeline hazard***.