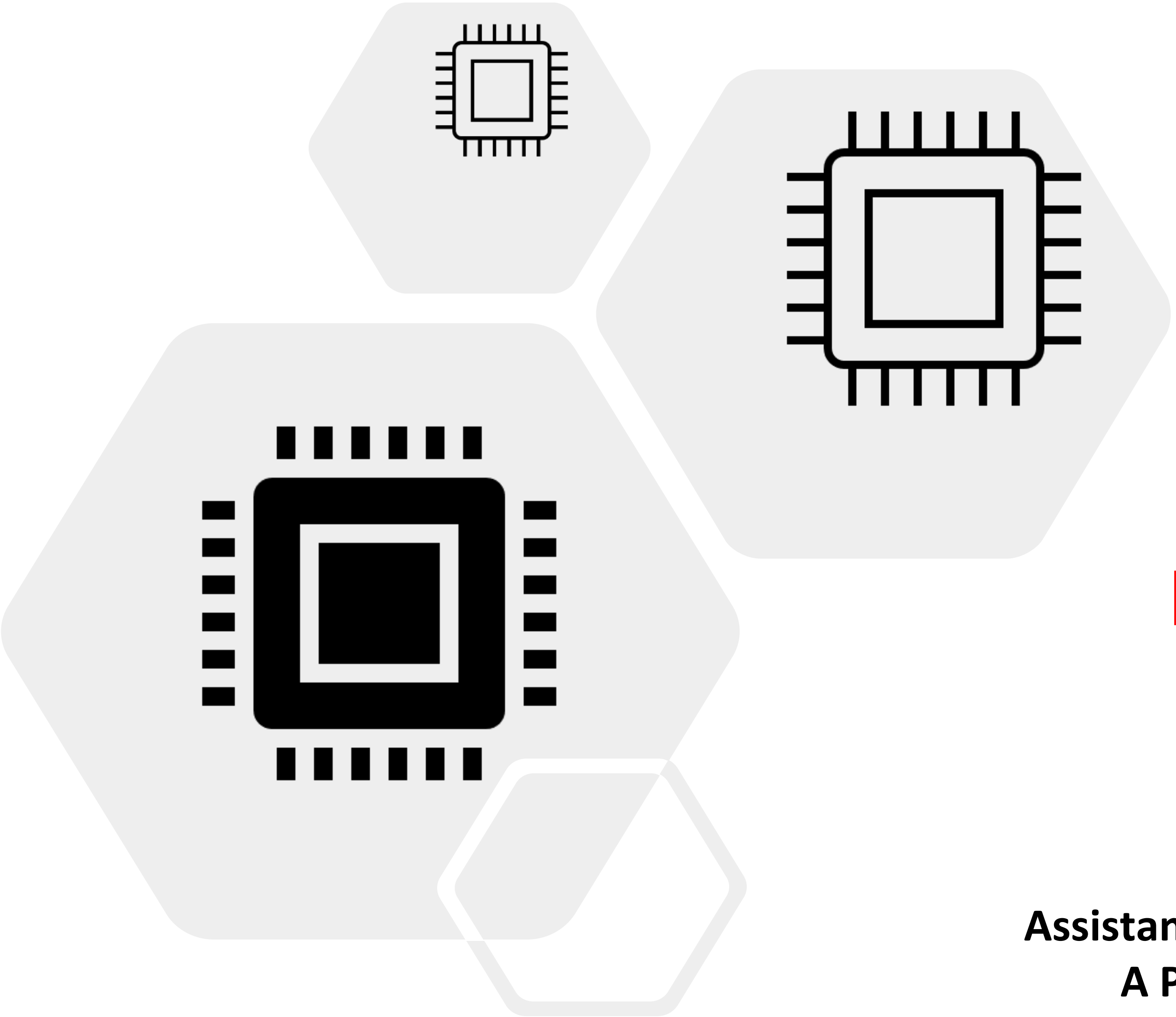
# High Performance Computing



## Introduction

## Levels of Parallelism


**Shafaque Fatma Syed**
**Assistant Professor - Dept. of Computer Engineering**
**A P Shah Institute of Technology, Mumbai**

# Topics to be discussed

- Levels of  Parallelism

# Levels of Parallelism

## Bit-level Parallelism:

- This form of parallelism is based on doubling the processor word size. Increased bit-level parallelism means faster execution of arithmetic operations for large numbers.
- For example, an 8-bit processor needs two cycles to execute a 16-bit addition while a 16-bit processor just one cycle.
- This level of parallelism seems to have come to an end with the introduction of 64-bit processors.

## Instruction-level parallelism (ILP):

- This type of parallelism is trying to exploit the potential overlap between instructions in a computer program.
- Multiple instructions from the same instruction stream can be executed concurrently.
- Generated and managed by hardware (ex:superscalar) or by compiler (ex:VLIW).
- Limited in practice by data and control dependencies.

# Levels of Parallelism

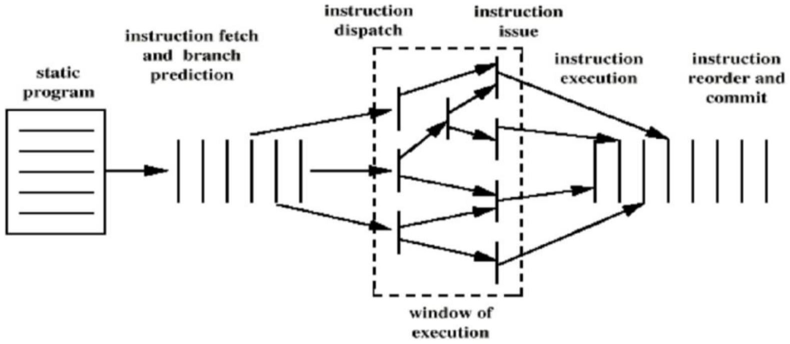
**Instruction-level parallelism (ILP):**
**Implemented on hardware level:**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| IF | ID | EX | MEM | WB | | | | |
| | IF | ID | EX | MEM | WB | | | |
| | | IF | ID | EX | MEM | WB | | |
| | | | IF | ID | EX | MEM | WB | |
| | | | | IF | ID | EX | MEM | WB |

**Instruction Pipelining** —**Execute different stages of different independent Instructions in the same cycle thus making full use of idle resources.**

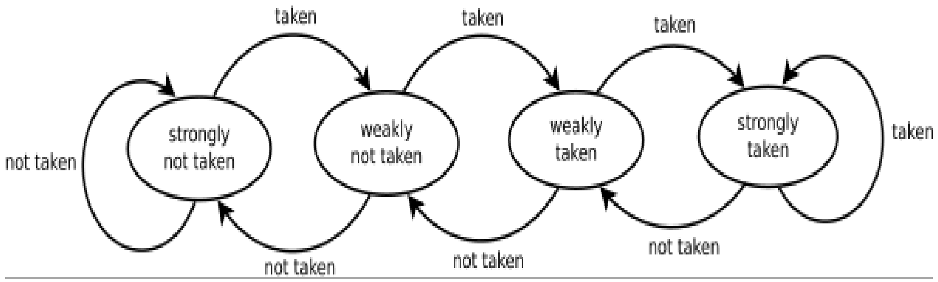
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| IF | ID | EX | MEM | WB | | | | |
| IF | ID | EX | MEM | WB | | | | |
| | IF | ID | EX | MEM | WB | | | |
| | IF | ID | EX | MEM | WB | | | |
| | | IF | ID | EX | MEM | WB | | |
| | | IF | ID | EX | MEM | WB | | |
| | | | IF | ID | EX | MEM | WB | |
| | | | IF | ID | EX | MEM | WB | |
| | | | | IF | ID | EX | MEM | WB |
| | | | | IF | ID | EX | MEM | WB |

**Superscalar Processors** — **Utilize multiple execution units in a single processor die, thus following instructions can be executed without waiting on complex preceding instructions to finish executing.**




**Out-of-order execution** — **Instructions not violating any data dependencies are executed when a unit is available even when preceding instructions are still executed.**

**Speculative execution / Branch prediction** — **The processor tries to avoid stalling while control instructions (branches — i.e. if or case commands) are still resolved by executing the most probable flow of the program.**

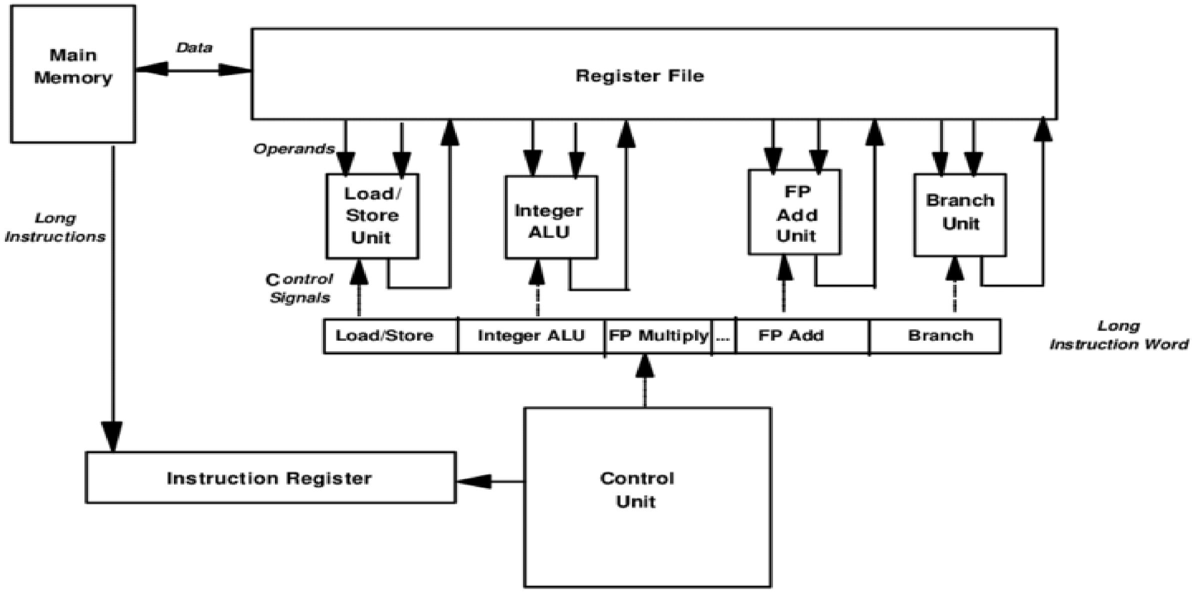


# Levels of Parallelism

**Instruction-level parallelism (ILP):**

**Realized on software level:**

**Using specialized compilers for *Very long instruction word* (VLIW) processors. VLIW processors have multiple execution units organized in multiple pipelines and require the compilers to prepare the parallel instruction streams to fully utilize their resources.**

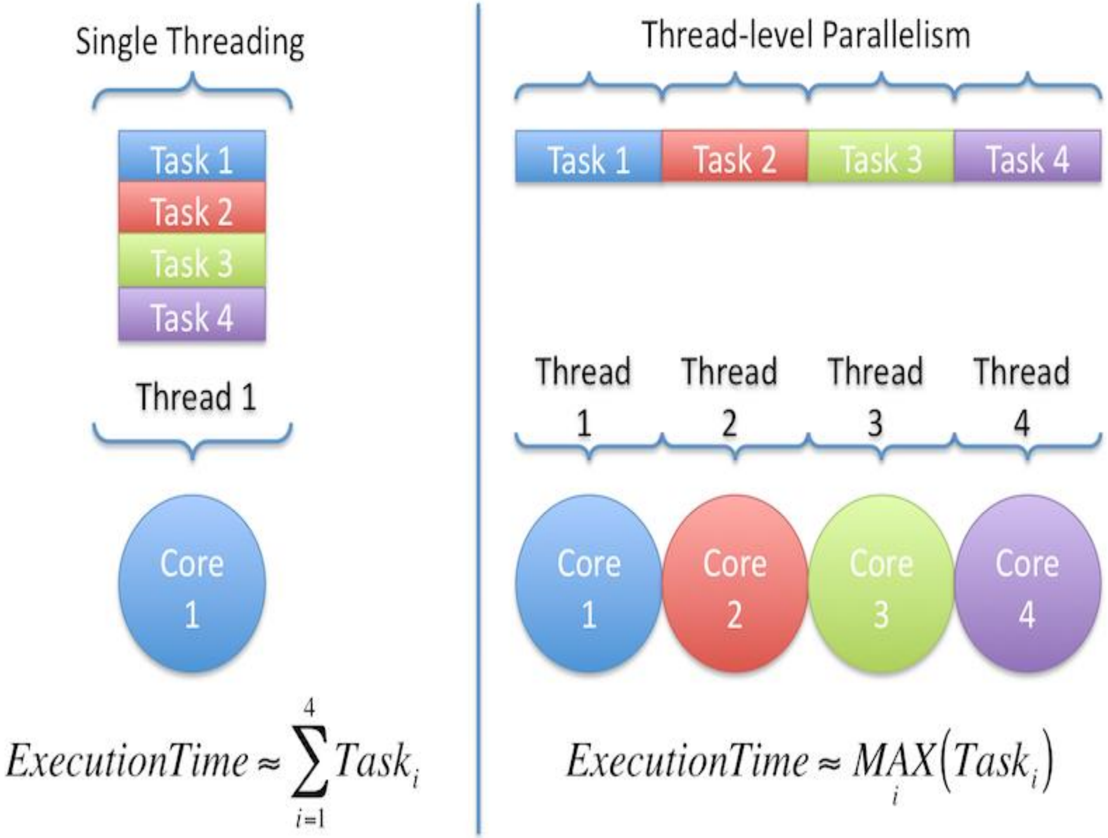



**Note: Most processors use a combination of the above ILP techniques. For example, Intel Pentium series used all of the above techniques to achieve higher performance with each product generation.**

# Levels of Parallelism

## Function/ Task / Thread-level parallelism:

- This high-level form of parallel computing is focusing on partitioning the application to be executed in distinct *tasks* or *threads,* that can be then executed simultaneously on different computation units.
- Threads can work on independent data fragments or even share data between them.
- Multiple threads or instruction sequences from the same application can be executed concurrently
- Generated by compiler and managed by compiler and hardware
- Limited in practice by communication/ synchronization overheads and by algorithm characteristics



# Levels of Parallelism

**Data parallelism:**

- **Instructions from a single stream operate concurrently on several data**
- **Limited by non-regular data manipulation patterns and by memory bandwidth**
- **Tries to partition the data to different available computation units instead.**
- **The cores execute the *same* task code over the data assigned to each.**
- **Data parallelism is the only available option for high-level parallelism in computer graphics because the graphic processor units (GPUs) are designed to execute each graphic processing task as fast as possible by partitioning each frame in regions.**
- **The task on command is then executed independently on each data region by their hundreds of processing units.**

# Levels of Parallelism

| Data Parallelisms | Task Parallelisms |
|---|---|
| 1. Same task are performed on different subsets of same data. | 1. Different task are performed on the same or different data. |
| 2. Synchronous computation is performed. | 2. Asynchronous computation is performed. |
| 3. As there is only one execution thread operating on all sets of data, so the speedup is more. | 3. As each processor will execute a different thread or process on the same or different set of data, so speedup is less. |
| 4. Amount of parallelization is proportional to the input size. | 4. Amount of parallelization is proportional to the number of independent tasks is performed. |
| 5. It is designed for optimum load balance on multiprocessor system. | 5. Here, load balancing depends upon on the e availability of the hardware and scheduling algorithms like static and dynamic scheduling. |

**Transaction-level parallelism:** **Multiple threads/processes from different transactions can be executed concurrently–Limited by concurrency overheads**

**Memory-level parallelism (MLP): is a term in computer architecture referring to the ability to have pending multiple memory operations, in particular cache misses or translation lookaside buffer(TLB) misses, at the same time.**

## You have completed this topic, you should be able to:

**Explain different levels of parallelism?**

# References Used


- **M. R. Bhujade, —Parallel Computing, 2nd edition, New Age International Publishers, 2009.**