Advanced Computer Architectures
Theory

Christian Rossi

Academic Year 2023-2024

Abstract

The course topics are:

- Review of basic computer architecture: the RISC approach and pipelining, the memory hierarchy.
- Basic performance evaluation metrics of computer architectures.
- Techniques for performance optimization: processor and memory.
- Instruction level parallelism: static and dynamic scheduling; superscalar architectures: principles and problems; VLIW (Very Long Instruction Word) architectures, examples of architecture families.
- Thread-level parallelism.
- Multiprocessors and multicore systems: taxonomy, topologies, communication management, memory management, cache coherency protocols, example of architectures.
- Stream processors and vector processors; Graphic Processors, GP-GPUs, heterogeneous architectures.

Contents

CHAPTER 1

Introduction

1.1 Architectures' classification

In 1966, Michael Flynn introduced a taxonomy outlining the architecture of calculators. This classification divides architectures into four categories:

- *Single Instruction Single Data*: utilized by uniprocessor systems.
- *Multiple Instruction Single Data*: although theoretically possible, this architecture lacks practical configurations.
- *Single Instruction Multiple Data*: features a straightforward programming model with low overhead and high flexibility, commonly employed in custom integrated circuits.
- *Multiple Instruction Multiple Data*: known for its scalability and fault tolerance, this architecture is utilized by off-the-shelf microservices.

1.1.1 Single instruction single data

The traditional concept of computation involves writing software for serial execution, typically on a single computer with a lone Central Processing Unit (CPU). Tasks are divided into a sequence of discrete instructions executed sequentially, allowing only one instruction to be processed at any given moment. This arrangement is illustrated by the single instruction single data architecture.

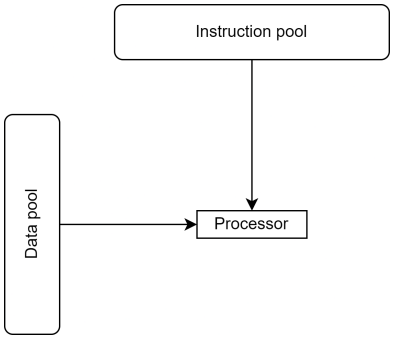


Figure 1.1: Single Instruction Single Data (SISD)

In a single instruction single data architecture:

- *Single instruction*: only one instruction is processed by the CPU in each clock cycle.
- *Single data*: only one data stream is utilized as input during each clock cycle.

Execution in this setup is deterministic, meaning the outcome is predictable and follows a defined sequence of steps. Single instruction single data architecture architectures represent the most prevalent type of computers.

1.1.2 Single instruction multiple data

In the single instruction multiple data architecture, the following characteristics apply:

- *Single instruction*: all processing units execute the same instruction simultaneously during each clock cycle.
- *Multiple data*: each processing unit can handle a different data element independently.

This architecture is particularly well-suited for specialized problems with a high level of regularity, such as graphics and image processing.

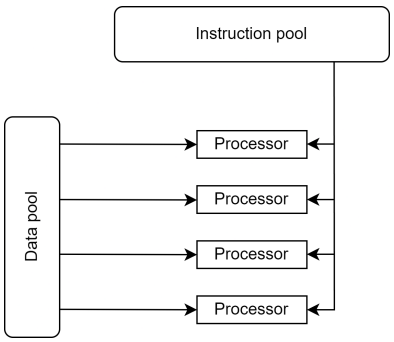


Figure 1.2: Single Instruction Multiple Data

1.1.3 Multiple instructions architectures

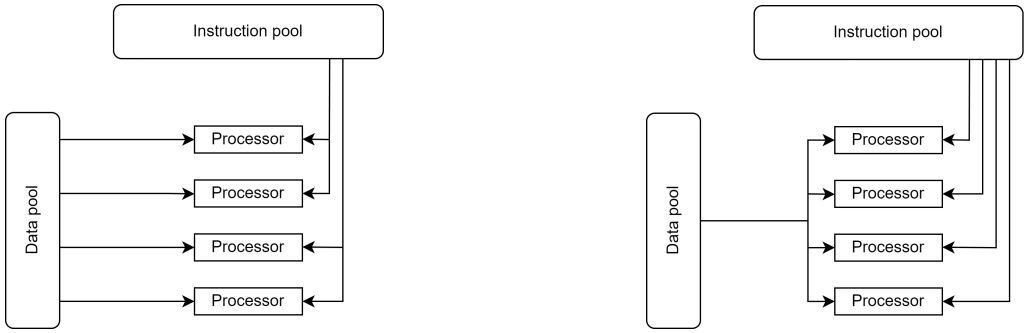
Hardware parallelism can be achieved through various methods:

- *Instruction-level parallelism*: this method harnesses data-level parallelism at different levels. Compiler techniques such as pipelining exploit modest-level parallelism, while speculation techniques operate at medium levels of parallelism.
- *Vector architectures and graphic processor units*: these architectures leverage data-level parallelism by executing a single instruction across a set of data elements simultaneously.
- *Thread-level parallelism*: this approach exploits either data-level or task-level parallelism within a closely interconnected hardware model that enables interaction among threads.
- *Request-level parallelism*: this method capitalizes on parallelism among largely independent tasks specified by either the programmer or the operating system.

Currently, the most common type of parallel computer features:

- *Multiple instruction*: each processor may execute a different instruction stream.
- *Multiple data*: each processor may operate with a distinct data stream.

Execution in parallel computing can occur synchronously or asynchronously, and it may be deterministic or non-deterministic.



(a) Multiple Instruction Multiple Data (b) Multiple Instruction Single Data

Figure 1.3: Possible architectures for hardware parallelism