



Chapter One Reading Assignment

SANDRA LUNG'AH
SCT212-0072/2021

Introduction

Over the past 65 years, computer technology has made significant progress, primarily due to advancements in technology and design innovations. The microprocessor emerged in the late 1970s, allowing for 35% annual growth in performance. This growth, combined with cost advantages, led to a growing percentage of the computer business being based on microprocessors. Two significant changes in the computer marketplace made it easier to commercially develop new architectures, such as RISC (Reduced Instruction Set Computer) architectures in the early 1980s. These machines focused on exploitation of instruction level parallelism and the use of caches. Personal computers and workstations emerged in the 1980s with the availability of the microprocessor.

Performance Growth Trends

Before the mid-1980s, performance grew at 25% per year, mainly driven by technology improvements.

From the mid-1980s to early 2000s, architectural and organizational advancements accelerated growth to 52% per year.

Since 2003, performance gains slowed to 22% per year due to power limitations and diminishing returns from instruction-level parallelism.

Impact on Computing:

The exponential performance improvement enabled the emergence of personal computers, workstations, smartphones, and cloud computing.

Microprocessors replaced minicomputers and became dominant across all computing levels.

This led to a renaissance in computer design, emphasizing both architectural innovation and efficiency.

Shift in Software Development:

The dramatic performance boost allowed programmers to prioritize productivity over performance, leading to the rise of managed languages (Java, C#) and scripting languages (Python, Ruby).

Software development moved towards Software as a Service (SaaS) and cloud computing.

End of the Hardware Renaissance:

After 2003, single-processor performance improvements declined due to power and parallelism limits.

The shift moved from instruction-level parallelism (ILP) to data-level parallelism (DLP), thread-level parallelism (TLP), and request-level parallelism (RLP).

Future progress relies on explicit parallel computing rather than traditional performance scaling.

Classes of computers

Personal Mobile Device (PMD)

Personal mobile device (PMD) is the term we apply to a collection of wireless devices with multimedia user interfaces such as cell phones, tablet computers, and so on. Applications on PMDs are often Web-based and media-oriented. Other key characteristics in many PMD applications are the need to minimize memory and the need to use energy efficiently.

Desktop Computing

the desktop market tends to be driven to optimize price-performance. This combination of performance (measured primarily in terms of compute performance and graphics performance) and price of a system is what matters most to customers in this market, and hence to computer designers. As a result, the newest, highest-performance microprocessors and cost-reduced microprocessors often appear first in desktop systems.

Servers

the role of servers grew to provide larger-scale and more reliable file and computing services. Such servers have become the backbone of large-scale enterprise computing, replacing the traditional mainframe. For servers, different characteristics are important ie: First, availability is critical. Failure of such server systems is far more catastrophic than failure of a single desktop, since these servers must operate seven days a week, 24 hours a day.

Classes of computers cont'

A second key feature of server systems is scalability. Server systems often grow in response to an increasing demand for the services they support or an increase in functional requirements. Thus, the ability to scale up the computing capacity, the memory, the storage, and the I/O bandwidth of a server is crucial.

Finally, servers are designed for efficient throughput. That is, the overall performance of the server, in terms of transactions per minute or Web pages served per second, is what is crucial. Responsiveness to an individual request remains important, but overall efficiency and cost-effectiveness, as determined by how many requests can be handled in a unit time, are the key metrics for most servers.

Clusters/Warehouse-Scale Computers

The growth of Software as a Service (SaaS) for applications like search, social networking, video sharing, multiplayer games, online shopping, and so on has led to the growth of a class of computers called clusters. Clusters are collections of desktop computers or servers connected by local area networks to act as a single larger computer. Each node runs its own operating system, and nodes communicate using a networking protocol. The largest of the clusters are called warehouse-scale computers (WSCs), in that they are designed so that tens of thousands of servers can act as one.

Embedded Computers

Embedded computers are found in everyday machines; microwaves, washing machines, most printers, most networking switches, and all cars contain simple embedded microprocessors.

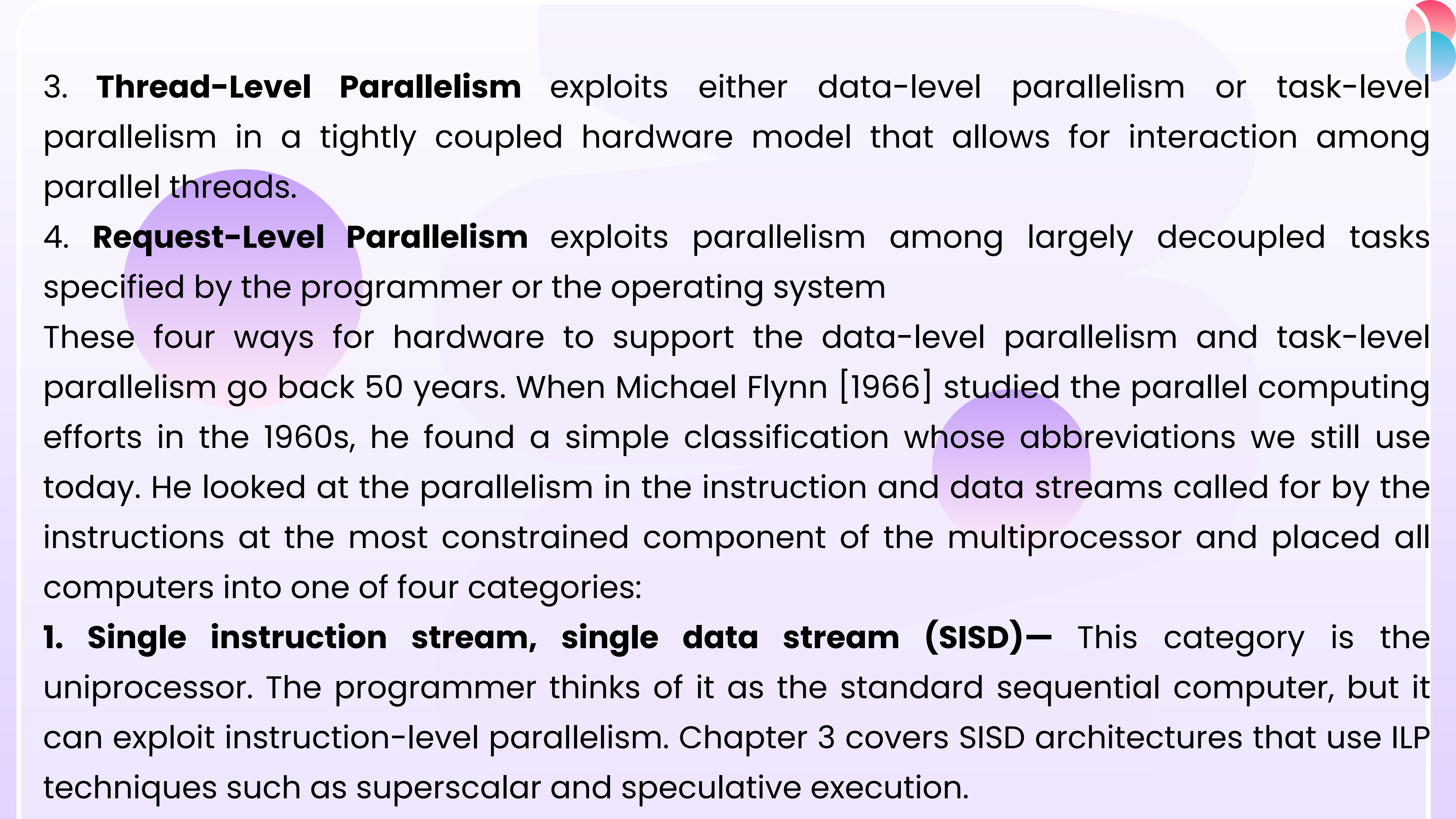
Classes of Parallelism and Parallel Architectures

Parallelism at multiple levels is now the driving force of computer design across all four classes of computers, with energy and cost being the primary constraints. There are basically two kinds of parallelism in applications:

1. **Data-Level Parallelism** (DLP) arises because there are many data items that can be operated on at the same time.
2. **Task-Level Parallelism** (TLP) arises because tasks of work are created that can operate independently and largely in parallel.

Computer hardware in turn can exploit these two kinds of application parallelism in four major ways:

1. **Instruction-Level Parallelism** exploits data-level parallelism at modest levels with compiler help using ideas like pipelining and at medium levels using ideas like speculative execution.
2. **Vector Architectures and Graphic Processor Units** (GPUs) exploit data-level parallelism by applying a single instruction to a collection of data in parallel



3. **Thread-Level Parallelism** exploits either data-level parallelism or task-level parallelism in a tightly coupled hardware model that allows for interaction among parallel threads.

4. **Request-Level Parallelism** exploits parallelism among largely decoupled tasks specified by the programmer or the operating system

These four ways for hardware to support the data-level parallelism and task-level parallelism go back 50 years. When Michael Flynn [1966] studied the parallel computing efforts in the 1960s, he found a simple classification whose abbreviations we still use today. He looked at the parallelism in the instruction and data streams called for by the instructions at the most constrained component of the multiprocessor and placed all computers into one of four categories:

1. Single instruction stream, single data stream (SISD)— This category is the uniprocessor. The programmer thinks of it as the standard sequential computer, but it can exploit instruction-level parallelism. Chapter 3 covers SISD architectures that use ILP techniques such as superscalar and speculative execution.

2. **Single instruction stream, multiple data streams** (SIMD)—The same instruction is executed by multiple processors using different data streams. SIMD computers exploit data-level parallelism by applying the same operations to multiple items of data in parallel. Each processor has its own data memory (hence the MD of SIMD), but there is a single instruction memory and control processor, which fetches and dispatches instructions. Chapter 4 covers DLP and three different architectures that exploit it: vector architectures, multimedia extensions to standard instruction sets, and GPUs.

3. **Multiple instruction streams, single data stream** (MISD)—No commercial multiprocessor of this type has been built to date, but it rounds out this simple classification.

4. **Multiple instruction streams, multiple data streams** (MIMD)—Each processor fetches its own instructions and operates on its own data, and it targets task-level parallelism. In general, MIMD is more flexible than SIMD and thus more generally applicable, but it is inherently more expensive than SIMD. For example, MIMD computers can also exploit data-level parallelism, although the overhead is likely to be higher than would be seen in an SIMD computer. This overhead means that grain size must be sufficiently large to exploit the parallelism efficiently.

Defining Computer Architecture

Instruction Set Architecture: The Myopic View of Computer Architecture

We use the term instruction set architecture (ISA) to refer to the actual programmer visible instruction set in this book. The ISA serves as the boundary between the software and hardware.

1. **Class of ISA**—Nearly all ISAs today are classified as general-purpose register architectures, where the operands are either registers or memory locations.

2. **Memory addressing**—Virtually all desktop and server computers, including the 80x86, ARM, and MIPS, use byte addressing to access memory operands. Some architectures, like ARM and MIPS, require that objects must be aligned.

3. **Addressing modes**—In addition to specifying registers and constant operands, addressing modes specify the address of a memory object.

4. **Types and sizes of operands**—Like most ISAs, 80x86, ARM, and MIPS support operand sizes of 8-bit (ASCII character), 16-bit (Unicode character or half word), 32-bit (integer or word), 64-bit (double word or long integer), and IEEE 754 floating point in 32-bit (single precision) and 64-bit (double precision).

5. **Operations**—The general categories of operations are data transfer, arithmetic logical, control and floating point.

6. **Control flow instructions**—Virtually all ISAs, including these three, support conditional branches, unconditional jumps, procedure calls, and returns.

7. **Encoding an ISA**—There are two basic choices on encoding: fixed length and variable length. All ARM and MIPS instructions are 32 bits long, which simplifies instruction decoding.

Genuine Computer Architecture: Designing the Organization and Hardware to Meet Goals and Functional Requirements

The implementation of a computer has two components: organization and hardware. The term organization includes the high-level aspects of a computer's design, such as the memory system, the memory interconnect, and the design of the internal processor or CPU (central processing unit—where arithmetic, logic, branching, and data transfer are implemented). The term microarchitecture is also used instead of organization.

Trends in Technology

Integrated circuit logic technology—Transistor density increases by about 35% per year, quadrupling somewhat over four years. Increases in die size are less predictable and slower. The combined effect is a growth rate in transistor count on a chip of about 40% to 55% per year, or doubling every 18 to 24 months. This trend is popularly known as Moore's law.

Semiconductor DRAM (dynamic random-access memory)—Now that most DRAM chips are primarily shipped in DIMM modules, it is harder to track chip capacity, as DRAM manufacturers typically offer several capacity products at the same time to match DIMM capacity.

Semiconductor Flash (electrically erasable programmable read-only memory)—This non-volatile semiconductor memory is the standard storage device in PMDs, and its rapidly increasing popularity has fuelled its rapid growth rate in capacity.

Magnetic disk technology—Prior to 1990, density increased by about 30% per year, doubling in three years. It rose to 60% per year thereafter, and increased to 100% per year in 1996. Since 2004, it has dropped back to about 40% per year, or doubled every three years.

Network technology—Network performance depends both on the performance of switches and on the performance of the transmission system.

Performance Trends: Bandwidth over Latency

Bandwidth or throughput is the total amount of work done in a given time, such as megabytes per second for a disk transfer. In contrast, latency or response time is the time between the start and the completion of an event, such as milliseconds for a disk access.

Trends in Power and Energy in Integrated Circuits

Power and Energy: A Systems Perspective

From the viewpoint of a system designer, there are three primary concerns. First, what is the maximum power a processor ever requires? Meeting this demand can be important to ensuring correct operation. Second, what is the sustained power consumption? This metric is widely called the thermal design power (TDP), since it determines the cooling requirement. The third factor that designers and users need to consider is energy and energy efficiency.

Energy and Power within a Microprocessor

For CMOS chips, the traditional primary energy consumption has been in switching transistors, also called dynamic energy. The energy required per transistor is proportional to the product of the capacitive load driven by the transistor and the square of the voltage:
$$\text{Energy dynamic} \propto \text{Capacitive load} \times \text{Voltage}^2$$

Distributing the power, removing the heat, and preventing hot spots have become increasingly difficult challenges. Power is now the major constraint to using transistors; in the past, it was raw silicon area. Hence, modern microprocessors offer many techniques to try to improve energy efficiency despite flat clock rates and constant supply voltages: Do nothing well. Most microprocessors today turn off the clock of inactive modules to save energy and dynamic power.

Dynamic Voltage-Frequency Scaling (DVFS). Personal mobile devices, laptops, and even servers have periods of low activity where there is no need to operate at the highest clock frequency and voltages. Modern microprocessors typically offer a few clock frequencies and voltages in which to operate that use lower power and energy.

Design for typical case. Given that PMDs and laptops are often idle, memory and storage offer low power modes to save energy.

Overclocking. Intel started offering Turbo mode in 2008, where the chip decides that it is safe to run at a higher clock rate for a short time possibly on just a few cores until temperature starts to rise.

Finally, because the processor is just a portion of the whole energy cost of a system, it can make sense to use a faster, less energy-efficient processor to allow the rest of the system to go into a sleep mode. This strategy is known as race-to-halt.

Trends in Cost

The Impact of Time, Volume, and Commoditization

The cost of a manufactured computer component decreases over time even without major improvements in the basic implementation technology. The underlying principle that drives costs down is the learning curve—manufacturing costs decrease over time. The learning curve itself is best measured by change in yield—the percentage of manufactured devices that survives the testing procedure. Volume is a second key factor in determining cost. Increasing volumes affect cost in several ways. First, they decrease the time needed to get down the learning curve, which is partly proportional to the number of systems (or chips) manufactured. Second, volume decreases cost, since it increases purchasing and manufacturing efficiency.

Commodities are products that are sold by multiple vendors in large volumes and are essentially identical. Because many vendors ship virtually identical products, the market is highly competitive. Of course, this competition decreases the gap between cost and selling price, but it also decreases cost.