Fundamentals of Computer Architecture

Madiba Hudson-Quansah

Contents

Chapter 1

Terminology

1.1 Computer Hardware

Definition 1.1.1: Hardware

The physical components of a computer system that can be seen and touched.

- Central Processing Unit (CPU) - Instruction sets and execution.
- Memory - RAM / ROM, caching mechanisms and memory hierarchy.
- Input / Output (IO)- Peripherals, buses and device controllers.
- Motherboards and Chipsets - Physical layout and connectivity of components.

1.2 Key Characteristics of Hardware

- Physical Components - Physical parts of a computer system.
- Electronic Circuits - Electrical circuits that perform functions such as processing data, storing information and facilitating communication between other components.
- Peripheral Devices - Devices that are connected to the computer system to provide additional functionality.
- Physical Infrastructure - Servers, routers, switches.
- Assembly of Components - The physical assembly of components to form a functional computer system.
- Firmware - Software that is embedded in hardware components for low level control of the specific component.
- Mechanical Parts - Physical parts of a computer system that are not electronic.

1.3 Operating System Fundamentals

Definition 1.3.1: Process

An instance of a program running on a computer system. A process differs from a process by having its own memory space and CPU time, i.e. A program becomes a process when it is loaded into memory and executed by the CPU.

- Process Management - Processing handling - Scheduling, Multitasking, Threads.
- Memory Management - Virtual Memory, Paging, Segmentation.
- File Systems (FS) - How data is stored, accessed, and organized on ROM.

- I/O Management - Managing data transfer between peripherals and the CPU.
- Security - Authentication, Authorization, Encryption, and Firewalls.

1.3.1 Execution of Programs

Program execution from source files goes through the following steps:

1. Compilation - Source code is compiled into machine code or an intermediate language.
2. Linking - Libraries and dependencies are linked to the executable file.
3. Loading - The executable file is loaded into memory by a loader software and executed by the CPU.
4. CPU Time - The CPU scheduler allocates CPU time to the process for execution.
5. Process Running - The process is executed by the CPU and performs the required operations.
6. Process Termination - The process is terminated and the resources are released.

1.4 System Architecture

- Von Neumann Architecture - CPU, Memory, IO, and Bus.
- Harvard Architecture - Separate memory for data and instructions.
- System Buses - Communication between different components of the system, PCI, USB.
- Interrupts and Handling - Managing interrupts from hardware devices for processing.

1.4.1 Von Neumann Architecture

Definition 1.4.1: Von Neumann Architecture

A computer architecture that is based on the concept of a stored-program computer. The Von Neumann architecture consists of a CPU, Memory, IO, and a Bus. The CPU fetches instructions from memory, decodes them, and executes them and then fetches the data the instructions operate on from the same memory usually at another memory address/location. The Von Neumann architecture is used in most modern computers.

1.4.2 Harvard Architecture

Definition 1.4.2: Harvard Architecture

A computer architecture that has separate memory for data and instructions. The Harvard architecture allows the CPU to fetch data and instructions simultaneously, which can improve performance. The Harvard architecture is used in some embedded systems and microcontrollers.

1.5 Components of a Computer System

Definition 1.5.1: System

A collection of components that work together to perform complex computational tasks, manage resources, or provide specific services. These components are interconnected and interdependent on each other.

- CPU - Data Path and Control Unit

  Data Path - Arithmetic and Logic Unit (ALU), Registers, and Cache.

  Control Unit - Instruction Fetch, Decode, and Execute. Controls the flow of electrons / data to perform operations in the CPU.

- Memory - Speed Size trade-off. The faster the memory the smaller the size. Memory hierarchy.
    1. Registers - Fastest memory, used to store data that is currently being processed.
    2. Cache - Faster than RAM, used to store frequently accessed data. L1, L2, L3.
    3. RAM - Random Access Memory, used to store data that is currently being processed.
    4. ROM - Read Only Memory, used to store firmware and boot instructions.
        - SSD - Solid State Drive, faster than HDD, used to store data.
        - HDD - Hard Disk Drive, slower than SSD, used to store data.
        - Optical Drives - CD, DVD, Blu-ray, used to store data.
        - Magnetic Tapes - Used for long term storage of data / Archival.
- Input / Output

1.6 Assembly Language

Definition 1.6.1: Assembly

A low-level programming language that is used to write programs that are executed by the CPU. Assembly language is specific to the CPU architecture and provides a way to directly control the hardware components of the system. Maps mnemonics to machine code instructions. Example `ADD`, `MOV`, `JUMP`

Assembly language is less productive than high-level languages but provides more control over the hardware components of the system. Assembly language programs are translated into machine code by an assembler and executed by the CPU.

1.7 General Terminology

Definition 1.7.1: RISC

Reduced Instruction Set Computer. A computer architecture that uses a small set of simple instructions that can be executed quickly. RISC architectures are designed to be efficient and fast by simplifying the instruction set and reducing the complexity of the CPU.

Definition 1.7.2: ARM

Advanced RISC Machine. A family of RISC architectures that are widely used in embedded systems, smartphones, tablets, and other devices. ARM processors are known for their low power consumption and high performance.

Theorem 1.7.1 Moore's Law

The number of transistors on a microchip doubles approximately every two years, resulting in an exponential increase in computing power. Moore's Law has been a driving force behind the rapid advancement of computer technology.

Theorem 1.7.2 Amdahl's Law

Amdahl's Law is a formula that describes the theoretical speedup of a program when running on multiple processors. The formula states that the speedup of a program is limited by the fraction of the program that can be parallelized. Amdahl's Law is used to analyse the performance of parallel computing systems.

$$S(n) = \frac{1}{(1-P) + \frac{P}{n}}$$

Where $S(n)$ is the speedup of the program running on n processors, P is the fraction of the program that can be parallelized.

Definition 1.7.3: Dennard Scaling

Dennard scaling is a principle that states that as transistors get smaller, their power density remains constant. This principle has allowed for the continued increase

Improvements in processor performance has slowed down due to the following factors

- Transistors no longer getting much better because of the slowing of Moore's Law 1.7 and the end of Dennard Scaling 1.7.
- The unchanging power budgets for microprocessors.
- The replacement of the single processor with several energy-efficient processors.
- The limits to multiprocessing to achieve Amdahl's Law 1.7.

Chapter 2

Classes of Computers

Rapid advancements in computer technology have led to the development of different classes of computers that are optimized for specific tasks and applications. These classes of computers can be broadly categorized into the following categories:

- Internet of Things / Embedded Computers
- Personal Mobile Devices
- Desktop Computing
- Servers
- Clusters/Warehouse Scale Computers

Feature	**Personal Mobile Device**	**Desktop**	**Server**	**Clusters**	**IOT**
Price of System	$100 - $1000	$300 - $2500	$5000 - $10,000,000	$100,000 - $200,000,000	$10 - $100,000
Price of Microprocessor	$10 - $100	$50 - $500	$200 - $2000	$50 - $250	$0.01 - $100
Critical System Design Issues	Cost, energy, media performance, responsiveness	price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, Throughput, energy proportionality	Price, energy, application-specific performance

Table 2.1: Summary of Classes of Computers

2.1 Internet of Things / Embedded Computers

Definition 2.1.1: Embedded Computer

A computer system that is designed to perform a specific task or function. Embedded computers are used in a wide range of applications, including consumer electronics, industrial automation, and automotive systems. Embedded computers are typically small, low-power devices that are optimized for a specific task.

Definition 2.1.2: Internet of Things (IOT)

Embedded computers that are connected to the internet, typically wirelessly, collecting useful data about their environs and interacting with the physical world. Some examples of IOT devices include smart thermostats, smart watches, smart cars, and smart homes.

Embedded computers have the widest spread of processing power and cost. They include 8-bit to 32-bit processors that may cost a penny, and high end 64-bit processors for cars and network switches that cost $100. Price is a key factor in the design of computers for embedded systems.

2.2 Personal Mobile Devices (PMD)

Definition 2.2.1: Personal Mobile Device

A small, portable computing device that is designed to be used on the go. Personal mobile devices include smartphones, tablets, and wearable devices. These devices are optimized for mobility, battery life, and connectivity.

Cost is also a key factor in the design of PMDs, with energy efficiency and media performance also being critical. Applications of PMDs are often web-based and media oriented. Processors in PMDs are often also considered to embedded computers because of their low power consumption and small size but are separated due to their ability to run externally developed software and share many features with desktop computers.

Responsiveness and predictability are key characteristics for media-applications, often requiring real-time performance.

Definition 2.2.2: Real Time Performance

A segment of an application that has an absolute maximum execution time. For example, in playing a video on a PMD, the time to process each frame is limited since the processor must accept and process the next frame shortly.

In other applications another requirement arises where the average time for a particular task is constrained as well as the number of instances when some maximum time is exceeded. This is known as **Soft Real Time Performance**.

2.3 Desktop Computing

Definition 2.3.1: Desktop Computer

A personal computer that is designed to be used on a desk or table. Desktop computers are typically larger and more powerful than personal mobile devices, with more storage, memory, and processing power. Desktop computers are used for a wide range of applications, including gaming, multimedia production, and office work.

The desktop market seeks to optimize price-performance chiefly, with energy and graphics performance also being critical.

Definition 2.3.2: Price-Performance

The combination of performance, measures in terms of compute performance and graphics performance, and the price of the computer system.

2.4 Servers

Definition 2.4.1: Server

A computer system that is designed to provide services or resources to other computers on a network. Servers are used for a wide range of applications, including web hosting, email, file storage, and database management. Servers are typically more powerful than desktop computers and are optimized for throughput, availability, and scalability.

The server market is chiefly characterized by the maximization of availability, scalability and throughput.

2.4.1 Availability

Definition 2.4.2: Availability

The percentage of time that a server is operational and available to provide services. Availability is a critical factor for servers that are used in mission-critical applications, such as e-commerce websites and financial systems.

2.4.2 Scalability

Definition 2.4.3: Scalability

The ability of a server system to grow in response to increasing demand for the services they support for an expansion in functional requirements.

The ability of a server to scale up computing capacity, memory, storage, and the I/O bandwidth is critical for servers that are used in applications that require high performance and reliability.

2.4.3 Throughput

Definition 2.4.4: Throughput

The overall performance of a server system, measured in terms of the number of requests it can handle per second or the amount of data it can process in a given time period.

2.5 Clusters / Warehouse Scale Computers (WSC)

Definition 2.5.1: Cluster

A collection of desktop computers or servers connected by local area networks to act as one unified computing resource. With each node, a separate computer system, running its own operating system and communicating via a network protocol.

Price-Performance is also critical to WSCs as they are so large with the majority of the cost of a warehouse associated with power and cooling the computers inside the warehouse. Availability is also curial for WSCs as the cost for downtime is very high.

2.5.1 WSCs vs Supercomputers

Supercomputers are designed to optimize floating-point performance and are used for scientific and engineering, running large communication-intensive batch programs that can run for weeks at a time. WSCs are designed to emphasize interactive applications, large-scale storage, dependability, and high internet bandwidth.

Chapter 3

Classes of Parallelism and Parallel Architectures

Definition 3.0.1: Parallelism

The ability to perform multiple tasks simultaneously. Parallelism can be achieved at different levels of a computer system, including instruction level, task level, and data level.

There are mainly two kinds of parallelism in applications:

Data-Level Parallelism (DLP) - The ability to perform the same operation on multiple data elements simultaneously.

Task-Level Parallelism (TLP) - The ability to perform tasks simultaneously and independently.

Computer hardware can be designed to exploit these two kinds of parallelism in four major ways:

Instruction Level Parallelism (ILP) - Exploits DLP with the use of compilers to optimize code to perform tasks like pipelining, and speculative execution.

Vector architectures, graphics processing units (GPUs) and multimedia instruction sets - Exploits DLP by applying a single instruction to a collection of data in parallel.

Thread-level parallelism - Exploits DLP or TLP in a hardware model that allows for interaction between parallel threads.

Request-Level parallelism - Exploits TLP among largely decoupled tasks specified by the programmer of operating system.

With Flynn's Taxonomy, all computers can be classified into categories based on the way they handle parallelism in the instruction and data streams. The four categories are:

Single Instruction, Single Data (SISD) - One instruction stream and one data stream are processed at a time. The uni-processor, or a single-core computer. Can perform ILP.

Single Instruction, Multiple Data (SIMD) - The same instruction is executed by multiple processors using different data steams. SIMD computers exploit DLP by applying the same operations to multiple items of data in parallel. Each processor has its own data memory, but there is a single instruction memory and control processor that fetches and dispatches instructions.

Multiple Instruction, Single Data (MISD) - Multiple processors execute different instructions on the same data stream. MISD computers are not common and are not used in practice.

Multiple Instruction, Multiple Data (MIMD) - Each processor fetches its own instructions and operates on its own data and targets TLP. MIMD computers are more flexible than SIMD computers and can be used for a wider range of applications, but is inherently more complex and expensive than SIMD.

This taxonomy is not mutually exclusive, and many computers can be classified into more than one category. For example, a GPU can be classified as both SIMD and MIMD.

Chapter 4

Defining Computer Architecture