BASIC FUNCTIONS AND OPERATIONAL UNITS

Kevin Skadron
Department of Computer Science, University of Virginia, USA

Keywords: Computers, computer systems, components, hierarchy, memory, mass storage, peripherals, motherboard, microprocessor, CPU, pipeline, functional units, execution units.

Contents

Summary

A computer system is best viewed as a hierarchy of components, ranging from peripherals like display monitors at the outermost level to elements at the innermost level like caches and registers that comprise the processor core. Detailed descriptions of these components and their interfaces will be left for later sections. This section describes the components that exist at each level and their basic purpose and function, and gives an overview of their relationship.

Components of a system, their functionality and their relation will be discussed here. Memory, mass storage devices, peripherals, and their relation with the CPU will be discussed here as well.

1. Introduction

A computer system consists of a wide variety of components. They are best viewed as comprising a hierarchy of levels. At the outermost level are the devices that permit the system to interface with the user. These peripherals are how users interact with the computer and how they store data, and include familiar components like the keyboard, monitor, and disk drive. At the innermost level are the functional units that comprise the processing core itself. These include units like registers and arithmetic units. This article describes the components that exist at each level and their basic purpose and function.

At the outermost level, we can regard a computer system as being made up of the core system and its peripherals. The core system can be thought of as the motherboard—a printed circuit board that contains the processor, the memory, the memory bus (or system bus) that connects these, and the I/O bus that is connected to the system bus. Peripherals are connected to the system over the I/O bus. Within the processor exist computing modules or stages, like the fetch unit, the execution unit, and so on. And within each of these are individual functional units, like branch predictors, arithmetic-logic units (ALUs), and registers. A high-level schematic of a computer system appears in Figure 1.

The purpose of this article is to explain the components that make up each level of this hierarchy and their relationship.

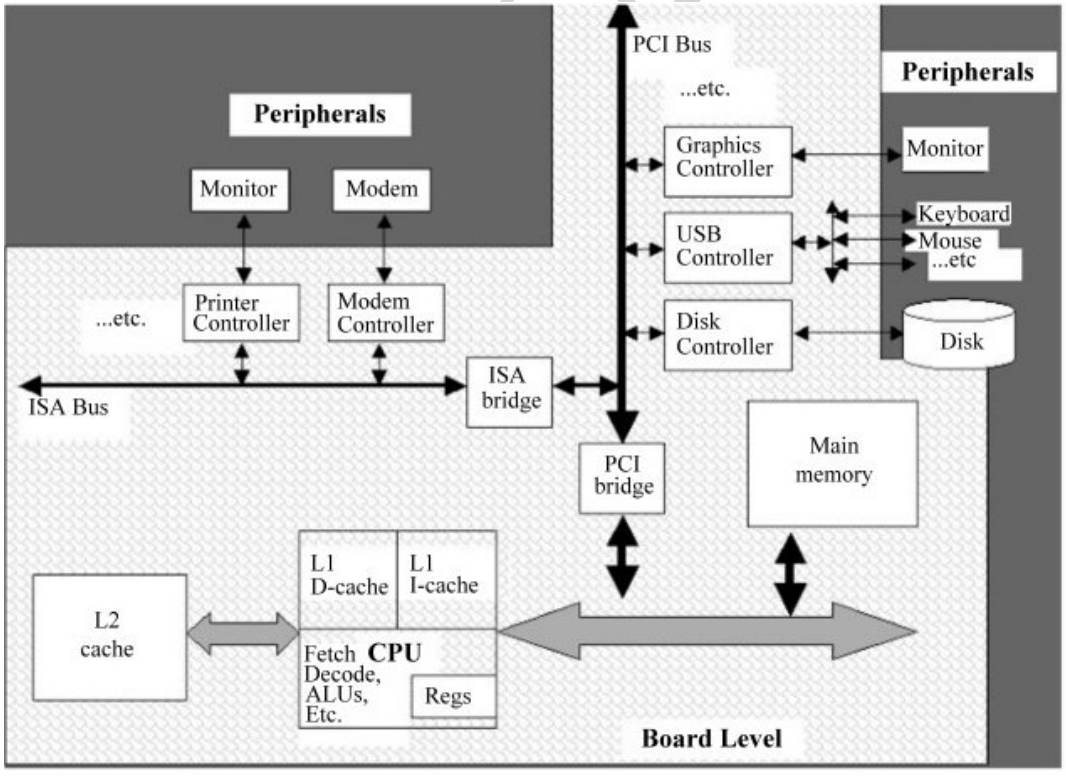

Figure 1: Schematic of a computer system showing its different levels and some common components.

2. Peripherals

Peripherals exist to convey data to and from the processor or to store data. Examples include keyboards, monitors, mice, disk drives, CD-ROM drives, etc. They are connected via a controller to the computer system via an I/O bus, which is in turn connected to the memory bus. This organization has already been depicted in Figure 1. The purpose of the controller is to interpret the electrical signals produced by the peripheral into the format required by the I/O bus, and manage the interaction with the bus. The I/O bus is separate and a client of the memory bus, connected by something called a bridge. This arrangement places less load on the memory bus and allows it to run faster, improving communication between the CPU and memory and thereby boosting the CPU's performance. To further improve CPU performance, most I/O takes place as direct memory access or DMA. With DMA, the I/O takes place directly between memory and the peripheral, without involving the CPU. This lets the CPU work on the main computation and avoid being distracted by the I/O (unless the CPU needs to access memory, in which case it may stall until memory becomes available). When the I/O transfer is complete, an interrupt is generated to notify the processor. This forces the current program to stop and invokes an interrupt handler in the operating system. This handler then takes any necessary special actions, such as checking for errors or allowing a program waiting for data to resume. When the handler completes, the OS allows one of the ready and waiting programs (perhaps but not necessarily the one that was interrupted) to resume execution.

Note that low-cost systems can mount I/O devices directly on the system bus, thereby dispensing with one of the two busses. This has two negative consequences: first, more devices must share the single bus; and second, such centralized buses are usually slow in order to better accommodate the I/O devices, and this in turn slows down communication between the CPU and memory.

Different I/O devices actually expect different bus standards. This sometimes means that the system contains additional busses in order to accommodate these different devices. For example, we might have an ISA as a client of the PCI bus, which is in turn a client of the memory bus.

ISA stands for "Industry Standard Bus" and is the original IBM PC bus. EISA is a subsequent, backward-compatible improvement for higher speed, and stands for "Extended ISA". Most computer systems today use the PCI (Peripheral Component Interconnect) bus developed by Intel, which is faster yet. These systems sometimes still contain an ISA bus to permit the use of older ISA devices, and many systems are now adding support for USB (Universal Serial Bus) devices, a recent standard meant to ease the use of low-speed I/O devices like keyboards, mice, scanners, etc. USB devices can be plugged into the computer as it runs, without opening the computer or requiring a reboot.

No matter how many busses the system contains, two or more devices may try to access the bus simultaneously. Arbitration is therefore required to serialize the bus accesses. Otherwise simultaneous accesses would interfere with each other and corrupt the data being sent on the bus. I/O devices that cannot be stopped are given priority to avoid loss

of data. But many controllers can buffer some I/O data—today's disk drives can buffer many sectors of data, keyboard controllers can remember some number of keystrokes, and so forth.

Before we survey some of the most common peripherals, it is important to note that the use of all these peripherals requires involvement by the operating system. It is the OS that contains the device drivers that understand how to convert program commands into I/O commands suited to each specific device, and how to manage input so that each application receives exactly the input that should receive. These device drivers are invoked by both program commands and by interrupt handlers.

2.1 Keyboards

Keyboards come in many varieties, representing different alphabets and with different auxiliary keys, but their operation is similar. In all cases, their purpose is to accept text input from the user. In most systems, pressing a key places a value representing the key into a register in the keyboard controller, and generates an interrupt. The CPU then reads that value—on Western keyboards with the Latin alphabet, a value between 1 and 102. Releasing the key generates a second interrupt. The distinction between press and release is necessary to implement SHIFT, CTRL, and ALT sequences.

This may sound like a lot of interrupts, but humans type incredibly slowly by a computer's standards. 1000 characters per second (i.e., ~2000 interrupts/sec.) is a fast typing rate, but this is actually similar to the rate at which the CPU performs context switches among different programs—CPUs execute billions of instructions per second.

Keyboards are typically ISA devices.

2.2 Monitors

Monitors are similar to televisions. They are traditionally implemented using a cathode ray tube (CRT), but today some flat-panel monitors use alternative technologies like LCDs (liquid crystal displays). Most laptops use LCD displays. In either case, the display is viewed as a matrix of pixels, and pixels are turned on or off (for monochrome displays) or set to different colors to display desired images, which may be text, windows, etc.

A video controller controls the display. The pixel values to be displayed are written into a buffer which is then used to control the CRT or LCD. This buffers consists of Video RAM or VRAM, which is a special kind of RAM memory designed specifically for use in video controllers. CRTs use a gun that shoots electrons at a phosphorescent coating on the back of the monitor's screen. An electrical grid within the CRT controls whether the electron beam is actually allowed to strike the phosphor. Color displays use three electron guns, one each for red, green, and blue, which together can be used to produce any desired color. LCDs are based on crystals that flow like a liquid. Their orientation can be controlled to determine whether light can pass through the display or not. The display itself consists of two glass plates with the liquid crystals sealed between them. The display is illuminated from behind, and an electrical grid controls the orientation of

the crystal corresponding to each pixel. Color is implemented by adding filters that separate the while backlight into red, green, and blue components that can be controlled independently.

Monitors are typically on the PCI bus when the system provides PCI support.

2.3 Mice and Other Pointing Devices

Many user interfaces today consist of many simultaneously visible windows or objects, and require a pointer to control where the user's input should actually occur. This is most frequently controlled by a mouse, a device roughly the size of a human hand that is placed flat on the work surface and rolled back and forth across it. As the mouse is moved back and forth across the desk, these motions translate into similar motions of the pointer (e.g., moving the mouse leftwards across the desk causes the pointer to move leftwards across the screen). Pressing or clicking buttons on the mouse allows the user to indicate various actions at the location where the pointer currently points.

Different mice have various numbers of buttons, and various means of tracking the user's motions. The basic techniques for tracking motion are mechanical and optical. A mechanical mouse uses a typically uses a rubber ball that is attached to orthogonally-placed wheels inside the mouse. As the mouse moves, the ball moves, and the wheels detect motion in the horizontal and vertical directions. For example, if the horizontal wheel moves while the vertical wheel does not, this corresponds to purely horizontal motion, and if both wheels move, this corresponds motion on some sort of diagonal. An optical mouse, on the other hand, shines a light onto a special mouse pad whose surface is marked with a grid. A photodector, also on the bottom of the mouse can then detect the mouse's motion.

Other pointing devices operate on broadly similar principles. Examples include track balls and joysticks. Some systems have touch-sensitive screens that allow the user to point directly on the screen with a finger or with a stylus.

Mice are typically on the ISA bus.

-
-
-

Bibliography

John L. Hennessy and David A. Patterson. (1996). *Computer Architecture: A Quantitative Approach*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann. [This is a graduate-level textbook describing

instruction-set and microarchitectural implementation and the interactions between them. It includes a detailed discussion of pipelining, branch prediction, caching, and virtual memory.]

Mark D. Hill, Norman P. Jouppi, and Gurindar S. Sohi, editors. (2000). *Readings in Computer Architecture.* San Francisco, CA, USA: Morgan Kaufmann. [This is an anthology of both historical and recent research papers in the area of computer architecture. The papers encompass all the levels discussed above.]

Kevin Skadron, Pritpal S. Ahuja, Margaret Martonosi, and Douglas W. Clark. (1999). Branch Prediction, Instruction-Window Size, and Cache Size: Performance Tradeoffs and Simulation Techniques. *IEEE Transactions on Computers*, 48(11):1260-81. [This paper explores how the interactions between different functional units—especially the instruction-window, the caches, and the branch predictor—and their configurations affect performance in some non-obvious ways. This has consequences for both design and modeling of computer systems.]

Andrew S. Tanenbaum. (1999). *Structured Computer Organization*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall. [This is an introductory textbook discussing the different levels of computer systems, their relation, and their implementation. It includes an overview of peripherals and bus standards, as well as a discussion of the role of the operating system and the compiler.]

William A. Wulf and Sally A. McKee. (1995). Hitting the Memory Wall: Implications of the Obvious. *Computer Architecture News*, 23(1):20-24. [This paper discusses the growing problem of the "memory wall" and its implications for computer design.]