Know Your Personal Computer

1. Introduction to Computers

S K Ghoshal

Siddhartha Kumar Ghoshal works with whatever goes inside parallel computers. That includes hardware, system software, algorithms and applications. From his early childhood he has designed and built electronic gadgets. One of the most recent ones among them is a sixteen processor parallel computer with IBM PC motherboards.

This article describes in brief the basics of the organization of the control and processing unit, memory subsystem and peripherals of a computer.

Introduction

Computers are built using semiconductors and other electronic parts, magnetic media and electromechanical devices. Collectively these are called the *hardware* of the computer. Their organization is subdivided into Control and Processing Unit (CPU), Memory subsystems and peripherals. They come in all sizes and capabilities, ranging from supercomputers to pen-tops, but there is a basic unity in their organization. We will first discuss the organization of computers and follow with a discussion on peripherals and software.

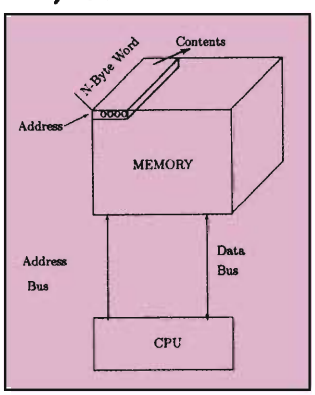
Computation and Memory

Just as humans use a base-10 arithmetic system because they have ten fingers on their two hands, computers use a base-2 arithmetic as digital hardware computes and stores information most reliably in one of two stable states, called OFF and ON. Alternatively they are called 0 and 1. *Bit* is an abbreviation of binary digit.

Using patterns of bits, computers store integers, rational numbers called *floating point numbers* which approximate real numbers, characters and many other data types.

Figure 1 is a block diagram of a computer. It consist of a CPU and memory.

Figure 1 Computer-memory subsystem.



About the series "Know Your Personal Computer"

The advent of personal computers in the early 1980s revolutionized the applications of computers. Computers which were expensive and somewhat daunting to the common man suddenly became affordable to many small organizations and individuals. An early decision by IBM (International Business Machines Corporation) to publicize the internal hardware structure and interface details enabled many manufacturers to design 'IBM clones' and price them competitively. Concurrently software companies, particularly Microsoft, designed an operating system called MS-DOS (Microsoft Disk Operating System) which allowed a novice to start using the computer. A host of compilers for popular programming languages (C, FORTRAN, PASCAL, COBOL etc.) appeared. Application programs for word processing, database design, accounting and numerous other areas were developed. These developments are of particular significance to India as computers became affordable to many colleges, schools and individuals. Almost all colleges and many schools now have computer centres with numerous personal computers. Students routinely use them to write programs. As a user one is usually curious to know what is 'inside the personal computer' and learn details not usually found in text books. The intention of this series is to explain in some detail the hardware and software of personal computers. It will include articles on the CPU system, memory system, peripheral systems, PC interfacing, basic input-output system, PC operating systems, PC networking, multimedia and some recent developments related to PCs.

The topics are chosen such that a science graduate will have a reasonably good knowledge of computer basics, applications of computers and future trends if she or he diligently reads the articles in the series. It will assist the teacher in supplementing information found in text books with issues of practical consequence in using personal computers and may even be useful in troubleshooting personal computers.

The series is written by S K Ghoshal who has many years of experience in designing systems using PCs. Readers are welcome to send suggestions on the articles and queries about personal computers.

V Rajaraman

A CPU can operate on these data types (called *operands*) and produce results specific to these types. The type of the operation performed is called the *op-code*. An op-code combined together with the addresses of the operand is called an *instruction*. An instruction is also represented as a bit pattern.

Just as humans use a base-10 arithmetic system because they have ten fingers, computers use a base-2 arithmetic.

A collection of instructions is called a computer program, or code. They are kept in memory, which is a collection of memory cells

Self Modifying Code

This idea of interchangeability of instruction and data was one of the most important ideas of von Neumann. In the early days of computers this allowed one to write self modifying programs. So the same instruction could be executed repeatedly each time modifying its bit pattern to give it a different meaning or to make it access operands from different memory locations. In those days memory was scarce. So programs had to manage with tiny amounts of memory (compared to today's standards) and do a lot of work by the same standards. Thus, without a self-modifying code many programs which later became the first working prototype of the many application programs of today, would never have been written. However it is difficult to make self modifying programs work correctly and even more difficult to document them. So their use is discouraged.

organized in a highly regular fashion. Eight bits are grouped together and called a *byte*. Each byte-wide memory cell has a unique address and its contents can be accessed by referring to it by the address. The memory is connected to the CPU, using a large number of data and address lines, called a *bus*. There is an *address bus* which is unidirectional and conveys the addresses generated by the CPU to the memory. There is also a bidirectional *data bus* which allows memory to exchange data with the CPU.

The width of the data bus is usually a multiple of eight. Memory is thus often organized in bytes as a whole. Each byte has a unique address. A byte is the minimum unit of information exchanged between the CPU and the memory. Each data type needs an integral number of bytes to be represented.

When data is brought inside the CPU for manipulation, it is kept in memory cells which are located in the CPU and called *registers*. Registers are also as wide as the integral multiple of bytes.

Each instruction also needs an integral number of bytes to be stored. The CPU fetches, decodes and executes them one by one. The CPU has an internal register called the *program counter (PC)* to index into the code memory and point to the next instruction to be executed. There is no way to distinguish between an instruction and a data-type and this fundamental limitation of today's computer organisation delays the development of correct programs.

The main memory of a computer can be read and written and data which is to be manipulated is stored in it. The code that is translated from highlevel language is also kept in the main memory. Code memory should not be over written although some programs called *self modifying code* do this.

When digital hardware is switched on, it is uninitialized. Each one-bit memory cell of the system can be in a state 0 or 1, which is decided randomly as it depends on so many physical parameters

beyond the analysis and control of the system designer. Thus before hardware can be used, it must be initialized.

That is done by executing the code from a *read only memory* (ROM) which cannot be altered by the CPU and which retains the information even when the computer is switched off. ROMs also contain many data objects that are needed to initialize the computer before it can be used. Programs and data stored in a ROM are collectively referred to as the *firmware* of the computer. In order to make the CPU begin its execution by fetching code from the ROM, a signal called RESET is applied to the CPU from outside. In most systems it is generated when the system is switched on. Optionally there is a reset button.

Main memory is made of two types of memory cells: static RAM (SRAM) and dynamic RAM (DRAM). Static RAMs consume more power per bit but are faster.

A given computer system has both of the above types of memory. A small amount of SRAM and a large amount of DRAM comprise the main memory system. Programs display a phenomenon called coherence which can be used effectively to design the memory systems of computers. There are two types of coherence:

- *Temporal Coherence* which means that if a given memory location is accessed now, there is a fair chance that it will be accessed soon again.

- *Spatial Coherence* which means that if a given memory location is accessed, it may well be that its neighboring locations will be accessed next.

Thus it helps to keep frequently accessed blocks of memory in fast SRAMs, which are backed up by slower but larger DRAMs. This principle is called *caching*. The smaller, faster memory at the higher level (i.e., closer to the CPU) is called *cache* memory

Locality of Reference

There is a phenomenon called locality of reference which programs display when they are executed. In fact 90% of the time they access 10% of the total memory locations they use if one takes an average over a large number of programs. Both temporal and spatial coherence are responsible for this. Temporal coherence occurs because programs often execute in tight loops, waiting for an iteration to converge or for some other event to take place. Spatial coherence arises because programmers often place data objects of the same type (for example the elements of a matrix) in consecutive memory addresses and perform similar transformations on them, one by one.

The memory is connected to the CPU, using a large number of data and address lines, called a bus.

What is a 'Cache'?

The dictionary meaning for cache is "a hiding place for food and stores left behind (e.g. by explorers) for future use". In computers however, it denotes a small but fast memory which acts as a temporary storage or as a buffer between two levels of memory.

whereas the larger and slower memory at the lower level, further away from the CPU is called *main memory*.

Note that this usage is relative as there are usually many levels in a hierarchical organization of a memory system and what is cache memory at a given level may itself be main memory at the next upper level. Caching is used at all levels in a memory subsystem design. The aim is to provide a memory that appears as fast as the cells in the highest level and as large as the capacity of the lowest level. For programs that are coherent, this aim is realized to a large extent. Between the SRAM and DRAMs of the memory system, the caching functions are performed *in hardware,* that is, by designing and physically implementing digital circuits using semiconductor integrated circuits. These circuits are called *cache controllers.*

Caching is *transparent* to the program that is running on the computer or the programmer who wrote the program. The program does not need to be written in a special way to accommodate caching. Nor does the programmer even have to be aware of it. Whether at a given point of time during the execution of the program (called *runtime*) a given data object or instruction is there in the main memory or the cache memory is decided based on the behaviour of the program in the immediate past and other ground realities prevailing at that instant.

Caching is used at all levels in a memory subsystem design. The aim is to provide a memory that appears as fast as the cells in the highest level and as large as the capacity of the lowest level.

Secondary Memory

Main memory is too small to hold all the data and code that is required. So *secondary memory,* which is magnetic in nature, is used to supplement main memory. Secondary memory is slower, but is cheaper per byte and is much larger in size, than the main memory. It can be used in two ways:

- *Virtual memory* where the magnetic disk holds the code and data objects which are not needed at the very moment of runtime. However, the objects are placed in such an address space that they can be addressed by the CPU (this address

space is called the *virtual address space* because it need not always be populated by *physical* memory) directly at any time. They are kept in disks and loaded into the main memory when needed. To make room for them, some other blocks of memory which have not been used for sometime are stored in the disk.

- *Archival storage* where objects are removed from the CPUs address space. They are brought back by a time consuming process requiring human intervention.

For implementing virtual memory, random access magnetic storage media, typically disks, are employed. This is because the CPU may potentially require any item in its virtual address space at any runtime. The physical location of the object referred in the virtual address space should not affect its access time. So magnetic memory devices such as floppy and hard disks, whose arms can seek any desired track are used to implement virtual memory. Such devices are called *swap devices* because they swap data objects and code pages to and from the main memory. Of course floppies are much slower as swap devices and in most cases it is impractical to use them due to their small storage capacities. In a hard disk, the magnetic material is coated on mechanically inflexible (hence the name) circular aluminum platters which rotate at very high speeds and the read/write heads float above them at a very small height. Hard disks can record data at very high densities and access them very fast. In floppy disks, a flexible substrate (hence the name) coated with a ferromagnetic material is squeezed by a pair of heads. Floppy disks are cheap and can be removed from the drives.

For archival storage and retrieval of data and programs, one can use sequential storage devices. An example is magnetic tapes. They can be accessed only from the beginning to the end and an object which is placed deep inside the medium can be accessed only after all other objects preceding it have been accessed and/or skipped. For backing up and restoring contents of memory, this is adequate. Tapes cost very little per bit of storage. Their capacities too are enormous.

What is a 'Transparent Mechanism'?

We call something transparent, if light passes through it. Glass is transparent. So one does not see glass itself. One sees what is behind the glass. If the means of making something happen is not visible, that mechanism is called *transparent* in computer parlance.

In a hard disk, the magnetic material is coated on mechanically inflexible (hence the name) circular aluminum platters which rotate at very high speeds and the read/write heads float above them at a very small height. Hard disks can record data at very high densities and access them very fast.

There is no way to distinguish between an instruction and a data-type and this fundamental limitation of today's computer organisation delays the development of correct programs.

Caching is employed between the swap device and main memory. Frequently referred objects are retained in the main memory. Objects not in use are put back into the swap device. This is one of the memory management chores that an operating system of the computer performs routinely. Users and developers of computer programs need not be aware of these functions.

Thus objects go up and down the levels of memory hierarchy (see *Figure 2*). An object which is frequently used will eventually find itself in the CPU registers provided it is small enough to fit there. On the other hand, objects not in use will go further and further away from CPU until they are backed up in a tape and removed from the computing system.

Peripherals

These are attached to the computers. They extend the computers' capabilities to function like printing on paper, exchanging information with other computers, accepting input from human beings, displaying text or graphic images and the like. Peripherals, by themselves cannot do anything. They always need to be

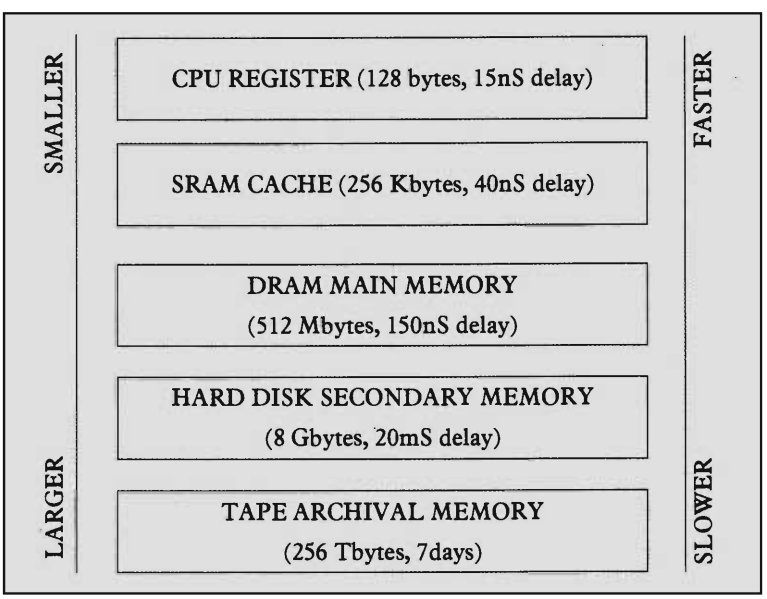


Figure 2 Levels of memory hierarchy, typical site and speed at each level is given as of 1995. These typical numbers change every year.

attached to a computer, usually referred to as the *host computer*. Peripherals too can have their own memory, but that is beyond the virtual address space of the host computer. Often they have their own built-in CPUs which perform special functions within that peripheral device. In such cases, the microcontroller chip containing the CPU is called an *embedded controller*.

Peripherals extend the computers' capabilities to function: they help in printing on paper, exchanging information with other computers, accepting input from human beings and displaying text or graphic images.

Software

The set of programs that run on a computer is collectively called its software. There are two types of software:

- *System software* which manage resources like the CPU and the main memory, extend the capabilities of the hardware and otherwise help users and application programs use the computer.
- *Application software* which run on the computer and have some end use.

Examples of system software are operating systems, different utilities and high-level language compilers.

There are many types of applications of computers. Numeric graphic, database and symbolic computing applications are some of them. Certain computers are specially suited for certain applications, and they keep evolving towards performing better and being more usable in these applications until they end up being embedded controllers in a special purpose equipment. Others are designed, built, sold and used as "general purpose computers" which can run applications from a wide range of fields but do not perform optimally in any of these roles. We will focus our attention, in future articles on general purpose computers.

We will learn more about CPUs, memory subsystems, system software and application software in the subsequent articles with special reference to personal computers. If you have any reactions to this article or the subsequent ones, please write to me.

Address for correspondence
S K Ghoshal,
Supercomputer Education and Research Centre,
Indian Institute of Science,
Bangalore 560 012, India.