Lecture 1:
Basic Concepts and Computer Evolution

Organization & Architecture

It is key to distinguish the differences between *computer architecture* and *computer organization*.

Computer Architecture It refers to attributes of the system that are visible to the programmer, or more specifically, the attributes of the system that have direct impact on the logical execution of a program. It is often interchangeable with the term *instruction set architecture* (ISA), which defines the instructions formats, instruction *operation codes* (opcodes), registers, memory and an algorithm for controlling instruction execution.

Architectural attributes include the instruction set, the number of bits used to represent various data types, I/O mechanisms,and techniques for addressing memory.

Computer Organization It refers to the operational units and their interconnections that realize the architectural specifications. Organizational attributes include hardware details transparent to the programmer such as control signals, interfaces between the computer and peripherals, and the memory technology used.

For example. it is an arcitectural design issue to determine whether a computer will have a multiplication instruction; whereas, it is an organizational issue to determine if the instruction will by implemented by a special multiply unit or by a mechanism that makes repeated use of the addition unit of the system.

Furthermore, architecture would relate to the software; whereas, organization would relate to computer model.

Structures & Functions

A computer is a hierarchical system; that is, it is a set interrated subsystems such that each subsystem is heirarchical in structure until the lowest elementary subsystems. Heirarchical system can be designed as separate individual parts; hence, a single subsystem can be designed at a time independently. Each subsystem consists of components and their interrelationships. The behavior of each subsystem depends only on a simplified, abstracted characterization of the next lower level subsystem. Ultimately, the designer is concerned with the *structure* and *function* of the subsystem where the structure is the way in which the components are interrated and the function is the operation of each individual component as part of the structure.

When designing a hierarchical system, one can choose the bottom-up approach or the top-down approach. For the bottom-up approach, the design starts at the elementary components, and then, build up to the complete system. Whereas, the top-down approach starts with the top view, and then, decompose the system to its subparts. The top-down approach is the clearest and most effective method that is usually used for computer design.

There are essentially only four basic functions that a computer can perform

Data Processing: Data may take a wide variety of forms, and the range of processing requirements is broad. However, we shall see that there are only a few fundamental methods or types of data processing.

Data Storage: Even if the computer is processing data on the fly (i.e., data come in and get processed, and the result go out immediately), the computer must temporarily store at least those pieces of data that are being worked on at any given moment. Thus, there is at least a short-term data storage function. Equally important, the computer performs a long-term data storage function. Files of data are stored on the computer for subsequent retrieval and update.

Data Movement: The computer's operating environment consists of devices that serve as either sources or destinations of data. Whne data are received from or delivered to a device that is directly connected to

the computer, the process is known as input-output (I/0), and the device is referred to as a *peripheral*. When data are moved over longer distances, to or from a remote device, the process is known as *data communication*.

Control: Within the computer. a control unit manages the computer's resources and orchestrates the performance of its functional parts in response to instructions.

There are four main structural components of a traditional single-processor computer

Central Processing Unit (CPU): Controls the operation of the computer and performs its data processing functions, often simply referred to as *processor*.

Main Memory: Stores data.

I/O: Moves data between the computer and its external environment.

System interconnection: Some mechanism that provides for communication among CPU, main memory and I/O. A common example of system interconnection is by means of a *system bus*, consisting of a number of conducting wire to which all the other components attach.

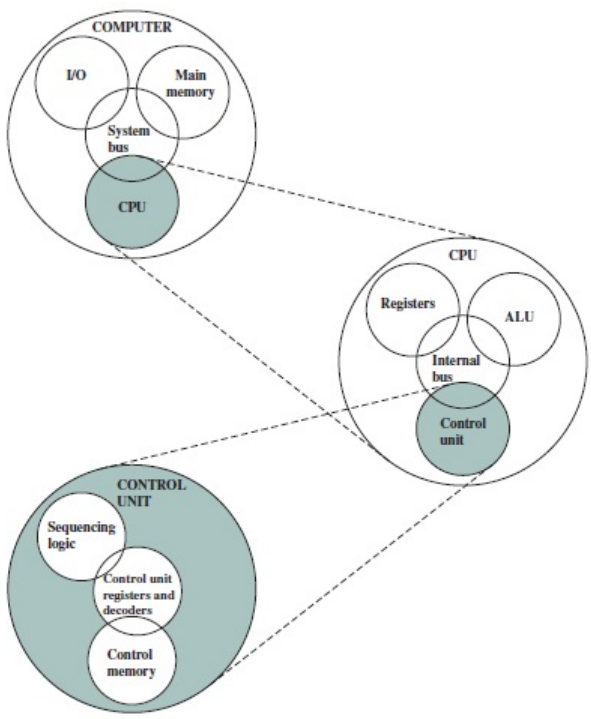

Figure 1.1 The Computer: Top-Level Structure

There are four major structural components of the CPU

Control Unit: Controls the operation of the CPU and hence the computer.

Arithmetic & Logic Unit (ALU): Performs the computer's data processing function.

Registers: Provides storage internal to the CPU.

CPU interconnection: Some mechanism that provides for communication among the control unit, ALU and registers.

Contemporary computers generally have multiple processors. If a computer consists of multiple processors on a single chip, the computer is called a multicore computer, and each processing unit (consisting of a control unit, ALU, registers, and perhaps *cache*) is called a *core*. The general functional elements of a core are:

Instruction Logic: This includes the tasks involved in fetching instructions, and decoding each instruction to determine the instruction to determine the instruction operation and the memory locations of any operands.

Arithmetic & Logic Unit (ALU): Performs the operation specified by an instruction.

Load/Store Logic: Manages the transfer of data to and from main memory via cache.

As stated the core also contains *caches* which are multiple layers of memory between the processor and main memory. Cache memory is smaller and faster than main memory and is used to speed up memory access. Typically, multiple levels of cache are used to improve performance of a multicore computer. The closer the cache is to the core, the faster and smaller it is than the cache that are further away from the core.

A principle components of a typical multicore computer starts with a *printed circuit board* (PCB) which is a rigid, flat board that holds and interconnects chips and oher electronic components. The board is made of layers, typically two to ten, that interconnect components via copper pathways that are etched into the board. The main printed circuit board in a computer is called a system board or *motherboard*. Likewise, there are smaller boards called *expansion boards* that can be plugged into the slots of the motherboard. Another essential component of the motherboard is its *chips*. A chip is a single piece of semiconducting material, typically silicon, upon which electronic circuits and logic gates are fabricated. Furthermore, the motherboard contains a slot or socket for the processor chip, memory chips, I/O controller chips and other key computer components.

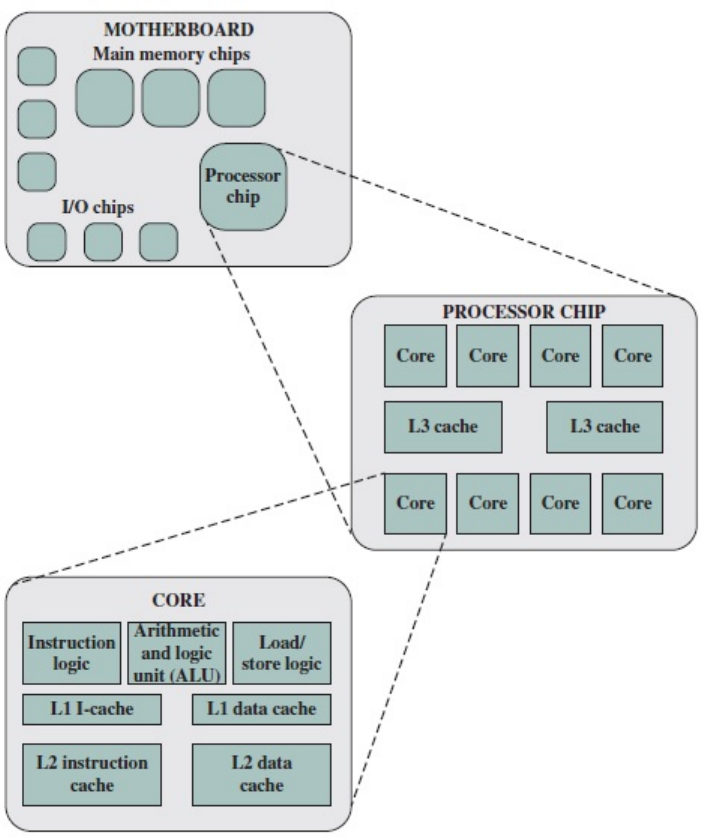

Figure 1.2 Simplified View of Major Elements of a Multicore Computer

A Brief History of Computers

The first generation computer used vacuum tubes for digital logic and memory. One of the most famous first generation computer was known as the IAS computer. It is the prototype of all subsequent general purpose computers. It was the first computer to implement the *stored-program concept* which was credited to be established to the mathematician John von Neumann.

In 1946, von Neumann and his colleagues began the design of the IAS computer, which was completed in 1952, at the Princeton Institute for Advance Studies. The IAS computer consisted of

- A *main memory*, which stores both data and instructions.
- An *arithmetic and logic unit (ALU)* capable of operating on binary data.
- A *control unit*, which interprets the instructions in memory and causes them to be executed.
- *Input-output (I/O)* equipment operated by the control unit

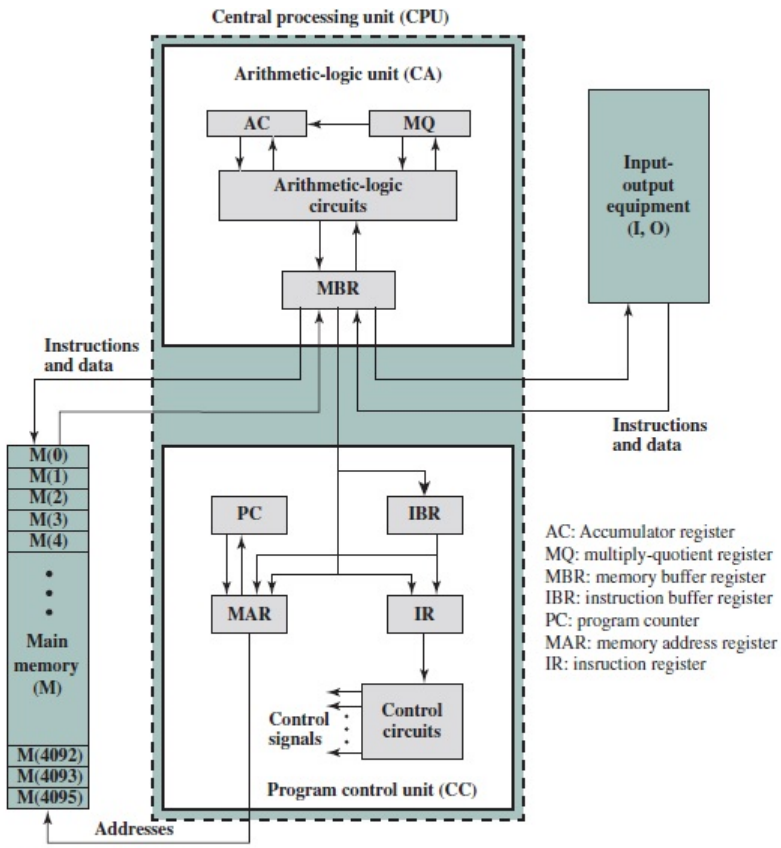

Figure 1.6 IAS Structure

The structure was outlined in von Neumann's earlier proposal. Some excerpts to recap are

> 2.2 **First:** Since the device is primarily a computer, it will have to perform the elementary operations of arithmetic most frequently. These are addition, subtraction, multiplication, and division. It is therefore reasonable that it should contain specialized organs for just these operations.
>
> It must be observed, however, that while this principle as such is probably sound, the specific way in which it is realized requires close scruntiny. At any rate a *central arithmetical* part of the device will probably have to exist, and this constitutes *the first specific part:* CA.
>
> 2.3 **Second:** The logical control of the device, that is, the proper sequencing of its operations, can be most efficiently carried out by a central control organ. If the device is to be *elastic*, that is, as

nearly as possible *all purpose*, then a distinction must be made between the specific instuctions given for and defining a particular problem, and the general control organs that see to it that these instructions- no matter what they are - are carried out. The former must be stored in some way; the latter are represented by definite operating parts of the device. By the *central control* we mean this latter function only, and the organs that perform it form *the second specific part:* CC.

2.4 **Third:** Any device that is to carry out long and complicated sequences of operations (specifically of calculations) must have a considerable memory...

The instructions which govern a complicated problem may constitute considerable maerial, particularly so if the code is circumstantial (which it is in most arrangements). This material must be remembered.

At any rate, the total *memory* constitutes *the third specific part of the device:* M.

2.6 The three specific parts CA, CC (together C), and M correspond to the *associative* neurons in the human nervous system. It remains to discuss the equivalents of the]emphsensory or *afferent* and the *motor* or *efferent* neurons. These are the *input* and *output* organs of the device.

The device must be endowed with the ability to maintain input and output (sensory and motor) contact with some specific medium of this type. The medium will be called the *outside recording medium of the device:* R.

2.7 **Fourth:** The device must have organs to transfer information from R into its specific parts C and M. These organs form its *input*, the *fourth specific part:* I. It will be seen that it is beast to make all transfers from R (by I) into M and never directly from C.

2.8 **Fifth:** The device must have organs to transfer from its specific parts C and M into R. These organs form its *output*, *the fifth specific part:* O. It will be seen that it is again best to make all transfers from M(by O) into R, and never directly from C.

The memory of the IAS computer consisted of $4,096(2^{12})$ storage locations called *words*, which were each 40 *binary digits* (bits). Both data and instructions are stored in memory. Numbers are represented in binary form, and each instruction is a binary code. Each number is represented by a sign bit and a 39-bit value. While each instruction is 20-bits with a 8-bit *opcode*, which specify the operation to be performed, and a 12-bit address designating one of the words in memory.

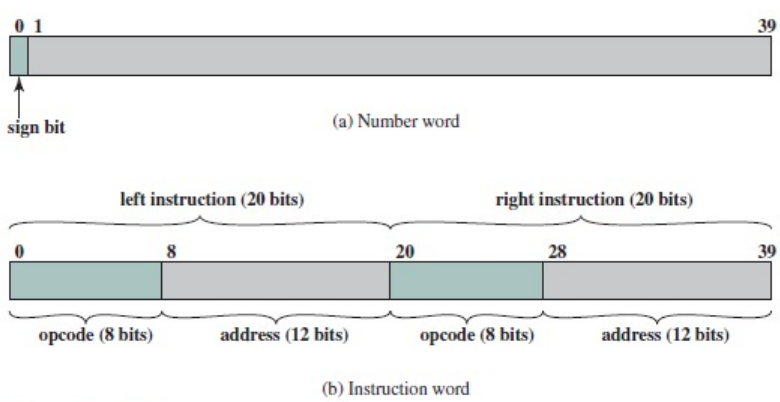

Figure 1.7 IAS Memory Formats

The control unit operates the IAS by fetching instructions from memory and executing them one at a time. It along with the ALU have storage locations called *registers* that are defined as

Memory Buffer Register (MBR): Contains a word to be stored in memory or sent to the I/O unit, or is used to receive a word from memory or from the I/O unit.

Memory Address Register (MAR): Specifies the address in memory of the word to be written from or read into the MBR.

Instruction Register (IR): Contains the 8-bit opcode instruction being executed.

Instruction Buffer Register (IBR): Employed to hold temporarily the right-hand instruction from a word in memory.

Program Counter (PC): Contains the address of the next instruction pair to be fetched from memory.

Accumulator (AC) and multiplier quotient (MQ): Employed to hold temporarily operands and results of ALU operations. The most significant bits of a result are stored in AC while the least significant bits are stored in MQ.

The IAS operates by repetitively performing an instruction cycle. Each instruction cycle consists of two subcycles: a fetch cycle and an execute cycle. During the fetch cycle, the opcode of the next instruction is loaded into the IR and the address portion is loaded into the MAR. This instruction may be taken from the IBR, or it can be obtained from memory by loading a word into the MBR, and the, down to the IBR, IR and MAR. The process is indirect because the operations are controlled by electronic circuitry and uses data paths.Once the opcode is in the IR, the ontrol circuitry interprets the opcode and executes the instruction by sending out the appropiate control signals to cause data to be moved or an operation to be performed by the ALU.

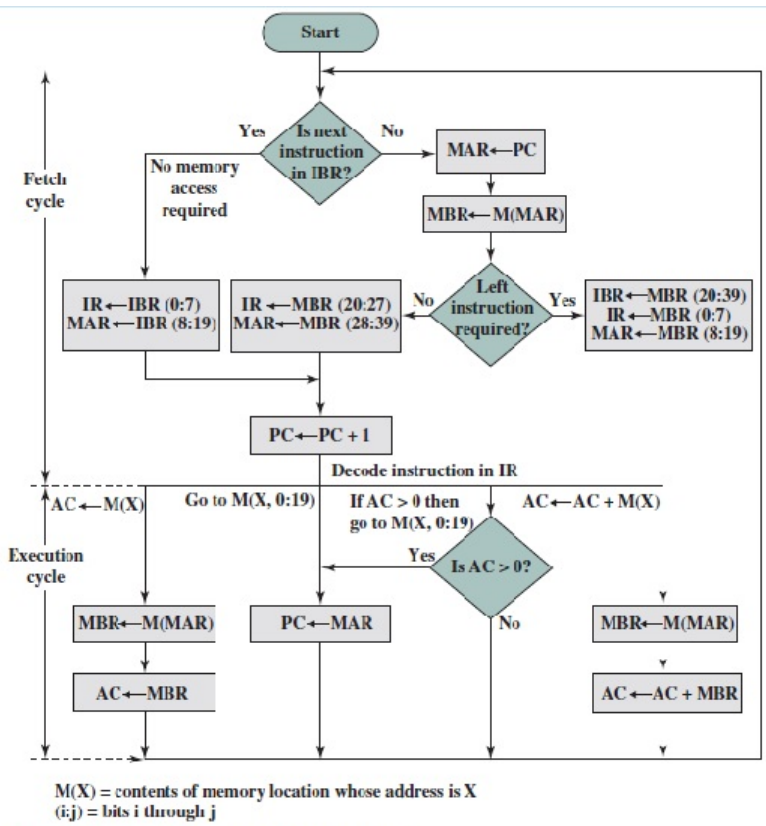

Figure 1.8 Partial Flowchart of IAS Operation

The IAS computer has a total of 21 instructions. The instructions are divided into five groups, which are

Data Transfer: Move data between memory and ALU registers or between two ALU registers.

Unconditional Branch: Normally, the control unit executes instructions in sequence from memory. This sequence can be changed by a branch instruction, which facilitates repetitive operations.

Conditional Branch: The branch can be made dependent on a condition, thus allowing decision points.

Arithmetic: Operations performed by the ALU.

Address Modify: Permits addresses to be computed in the ALU and then inserted into instructions stored in memory. This allows a program considable addressing flexibility.

Table 1.1 The IAS Instruction Set

Instruction Type	Opcode	Symbolic Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator AC
	00001001	LOAD MQ,M(X)	Transfer contents of memory location X to MQ
	00100001	STOR M(X)	Transfer contents of accumulator to memory location X
	00000001	LOAD M(X)	Transfer M(X) to the accumulator
	00000010	LOAD –M(X)	Transfer –M(X) to the accumulator
	00000011	LOAD |M(X)|	Transfer absolute value of M(X) to the accumulator
	00000100	LOAD –|M(X)|	Transfer –|M(X)| to the accumulator
Unconditional branch	00001101	JUMP M(X,0:19)	Take next instruction from left half of M(X)
	00001110	JUMP M(X,20:39)	Take next instruction from right half of M(X)
Conditional branch	00001111	JUMP + M(X,0:19)	If number in the accumulator is nonnegative, take next instruction from left half of M(X)
	00010000	JUMP + M(X,20:39)	If number in the accumulator is nonnegative, take next instruction from right half of M(X)
Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD |M(X)|	Add |M(X)| to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB |M(X)|	Subtract |M(X)| from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
	00001100	DIV M(X)	Divide AC by M(X); put the quotient in MQ and the remainder in AC
	00010100	LSH	Multiply accumulator by 2; that is, shift left one bit position
	00010101	RSH	Divide accumulator by 2; that is, shift right one position
Address modify	00010010	STOR M(X,8:19)	Replace left address field at M(X) by 12 rightmost bits of AC
	00010011	STOR M(X,28:39)	Replace right address field at M(X) by 12 rightmost bits of AC

The second generation of computers used *transistors* instead of vacuum tubes. A transistor is a solid-state device made of silicon. It was invented at Bell Labs in 1947, and in the 1950s, it became commercially available. The second generation also introduced more complex arithmetic and logic units and control units, the use of high-level programming languages, and the provision of *system software* with the computer. The system software provided the ability to load programs, move data to peripherals, and libraries to perform common computations, similar to modern operating systems.

A single, self-contained transistor is called a *discrete component*. Throughout the 1950s and early 1960s, electronic equipment was composed largely of discrete componentstransistors, resistors, capacitors, and so on. Discrete components were manufactured separately, packaged in their own containers, and soldered or wired together onto a Masonite-like circuit board. The perform was expensive and cumbersome. In 1958 came the achievement that revolutionized electronics and started the era of *microelectronics*: the invention of the *integrated circuit*.

As stated the basic functions of a digital computer are data storage, movement, processing, and control. To accomplish these functions, only two fundamental types of components are required: *gates* and *memory cells*. A gate is a device that implements a simple Boolean or logical function, and a memory cell is a device that can store 1 bit of data; that is, the device can be in one of two stable states at any time.

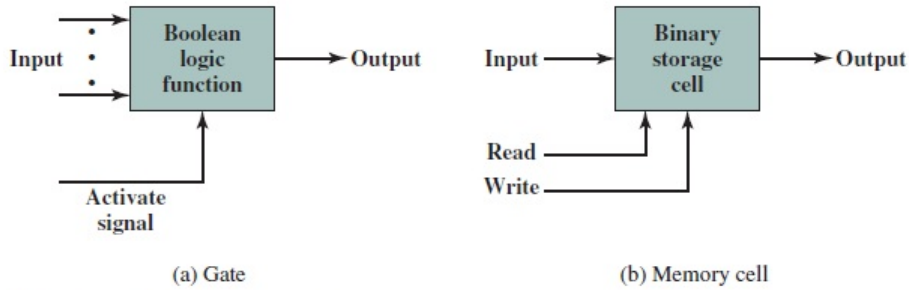

Figure 1.10 Fundamental Computer Elements

By interconnecting large numbers of these fundamental devices, we can construct a computer. We can relate this to our four basic functions as follows:

- Data storage: Provided by memory cells.
- Data processing: Provided by gates.
- Data movement: The paths among components are used to move data from memory to memory and from memory through gates to memory.
- Control: The paths among components can carry control signals. For example, a gate will have one or two data inputs plus a control signal input that activates the gate. When the control signal is ON, the gate performs its function on the data inputs and produces a data output. Conversely, when the control signal is OFF, the output line is null, such as the one produced by a high impedance state. Similarly, the memory cell will store the bit that is on its input lead when the WRITE control signal is ON and will place the bit that is in the cell on its output lead when the READ control signal is ON.

The integrated circuit exploits the fact that such components as transistors, resistors, and conductors can be fabricated from a semiconductor such as silicon. It is merely an extension of the solid-state art to fabricate an entire circuit in a tiny piece of silicon rather than assemble discrete components made from separate pieces of silicon into the same circuit. Hence, many transistors can be produced at the same time on a single wafer of silicon and they can be connected with a process of metallization to form circuits. Therefore, the wafer is broken up into chips with each chip consisting of many gates and/or memory cells plus a number of input and output attachment points. And these chips are then packaged in housing that protects them and provides pins for attachment to devices beyond the chips. These packages can then be interconnected on a printed circuit board to produce larger and more complex circuits.

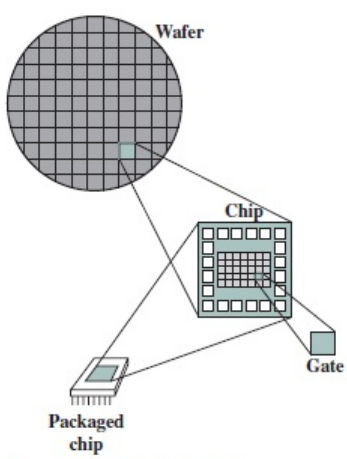

Figure 1.11 Relationship among Wafer, Chip, and Gate

These early integrated circuits are referred to as *small-scale integration* (SSI). As time when on, it was possible to pack more and more components in the same chip. This led to the famous Moores law, which was propounded by Gordon Moore, cofounder of Intel, in 1965. Moore observed that the number of transistors that could be put on a single chip was doubling every year, and correctly predicted that this pace would continue into the near future. The consequences of Moore's law are:

- The cost of a chip has remained virtually unchanged during this period of rapid growth in density. This means that the cost of computer logic and memory circuitry has fallen at a dramatic rate.
- Because logic and memory elements are placed closer together on more densely packed chips, the electrical path length is shortened, increasing operating speed.
- The computer becomes smaller, making it more convenient to place in a variety of environments.
- There is a reduction in power requirements.
- The interconnections on the integrated circuit are much more reliable than solder connections. With more circuitry on each chip, there are fewer interchip connections.

Later generations of computers used larger scale integrations that ultimately sped up the speed of computers. The following table show a timeline of the computer generations

Table 1.2 Computer Generations

Generation	Approximate Dates	Technology	Typical Speed (operations per second)
1	1946–1957	Vacuum tube	40,000
2	1957–1964	Transistor	200,000
3	1965–1971	Small- and medium-scale integration	1,000,000
4	1972–1977	Large scale integration	10,000,000
5	1978–1991	Very large scale integration	100,000,000
6	1991–	Ultra large scale integration	>1,000,000,000