# COMPUTER ARCHITECTURE: CSC 303

## INTRODUCTION

In spite of the variety and pace of change in the computer field, certain fundamental concepts apply consistently throughout. The application of these concepts depends on the current state of technology and the price/performance objectives of the designer. Many computer manufacturers offer a family of computer models, all with the same architecture but with differences in organization. In a class of computers called microcomputers, the relationship between the architecture and organization is very close. Changes in technology not only influenced organization but also result in the introduction of more powerful and more complex architecture. However, because a computer organization must be designed to implement a particular architectural specification, a thorough treatment of organization requires a detailed examination of architecture as well.

## COMPUTER ORGANIZATION AND ARCHITECTURE

In computer engineering, computer architecture is the conceptual design and fundamental operational structure of a computer system. It is a blueprint and functional description of requirements and design implementations for the various parts of a computer focusing largely on the way by which the central processing unit (CPU) performs internally and accesses addresses in memory.

The term "computer organization" refers to the operational units of the computer and their interconnection. The main hardware unit are the CPU, Memory, I/O units etc, the hardware details such as the various computational unit, control signals, interface between peripherals and the memory technology are included in the computer organization.

Computer architecture comprises at least three main subcategories:

- **Instruction Set Architecture (ISA)**: Is the abstract image of a computing system that is seen by a machine language (or assembly language) programmer, including the instruction set, memory address modes, processor registers, address and data formats.

- **Microarchitecture:** Also known as computer organization describes the data paths, data processing elements and data storage elements. It also describes how to implement the ISA.
- **System Design:** Which includes all of the other hardware components within a computing system such as:

✓ System interconnects such as computer buses and switches

✓ Memory controllers and hierarchies

✓ CPU off-load mechanisms such as direct memory access (DMA) issues like multi-processing.

Once both ISA and microarchitecture have been specified, the actual device needs to be designed into hardware and this design process is often called implementation. Implementation is usually not considered architectural definition, but rather hardware design engineering.

## STRUCTURAL AND FUNCTIONAL COMPOENT OF A COMPUTER

- **Structure:** The way in which the components are interrelated.
- **Function:** The operation of each individual component as part of the structure.

The structure and function of a computer system describes the basic functions that a computer can perform. In general terms, there are only four:

- Data processing
- Data storage
- Data movement
- Control

Computer systems must be able to **process data**, this data may take a wide variety of forms, and the range of processing requirement are broad. It is also essential that a computer **store data**, even if it is processing the data on the fly (i.e data come in and get processed and the results go out immediately) for subsequent retrieval and update. Similarly, computers must be able to **move data** between itself an outside world using devices that serve as either sources or destinations of data. When data are received from or delivered to a device that is directly connected to the computer, the process is known as input- output (I/O) 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 communications. Finally, there must be control of these three functions. Ultimately, this control is exercised by the individuals who provide the computer with instructions. Within the computer a **control unit** manages the computer resources and coordinates the performance of its functional parts in response to those instructions.

There are four main structural components of a computer system:

- **Central Processing Unit (CPU)**: Controls operations and performs 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 Interconnections**: Some mechanism that provides 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 wires to which all the other components attach. However, the most interesting and complex component is the CPU, as shown in figure 1.

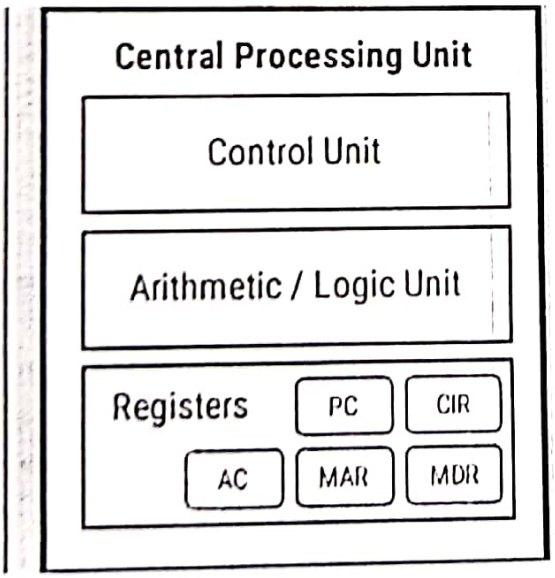



**Figure 1: The CPU**

CPU major structural components are as follows:

- **Control Unit**: The control unit controls and monitors communications between the hardware attached to the computer. It controls the input and output of data, checks

that signals have been delivered successfully and make sure that data goes to the correct place at the correct time.

- **Arithmetic Logic Unit (ALU):** The ALU is where the CPU performs the arithmetic and logic operations. Every task that a computer carries out is completed here, e.g typing into a word processor involves adding binary digits to the file and then calculating which pixels on the screen should change so that you can see the characters. The ALU's operations fall into two parts:

  I. The arithmetic part, which deals with calculations, e.g 1 + 2 = 3

  II. The logic part, which deals with any logical comparisons, e.g 2>1

- **Registers:** Registers are a type of computer memory used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU. The registers used by the CPU are often termed as processor registers. Some mostly used registers are Accumulator (AC), Program Counter (PC), Memory Address Register (MAR), Memory Data Register (MDR) and Current Instruction Register (CIR).

i. **Accumulator Register (AC):** The accumulator register is located inside the ALU, It is used during arithmetic & logical operations of ALU.

ii. **Program Counter (PC):** Program counter register is also known as Instruction Pointer (IP) register. This register stores the address of the next instruction to be fetched for execution. When the instruction is fetched, the value of IP is incremented. Thus this register always points or holds the address of next instruction to be fetched.

iii. **Memory Address Register (MAR):** This register holds the address of memory where CPU wants to read or write data. When CPU wants to store some data in the memory or reads the data from the memory, it places the address of the required memory location in the MAR.

iv. **Memory Data Register (MDR):** MDR is the register of a control unit that contains the data to be stored in the computer storage (e.g. RAM) or the data after a fetch from the computer storage. It acts like a buffer and holds anything that is copied from the memory ready for the processor to use it.

v. **Current Instruction Register (CIR):** The Instruction Register (IR) or Current Instruction Register (CIR) is the part of a CPU's control unit that holds the instruction currently being executed or decoded. In simple processors, each instruction to be executed is loaded into the instruction register, which holds it while it is decoded, prepared and ultimately executed.

## COMPUTER DESIGN

Basically, there are two computer design models which includes the contemporary computer design that are based on concepts developed by John Von Neumann at the Institute for Advanced Studies Princeton which is referred to as the Von Neumann architecture and the other design model is the Harvard architecture.

### 1. Von Neumann Architecture

The Von Neumann architecture is a computer design model that uses a processing unit and a single separate storage structure to hold both instructions and data as shown in Figure 2. It is named after mathematician and early computer scientist John Von Neumann. The Von Neumann design thus forms the basis of modern computing though it suffers from one obvious problem, all information (instructions and data) must flow back and forth between the processor and memory through a single channel, and this channel will have finite bandwidth. When this bandwidth is fully used the processor can go no faster. This performance limiting factor is called the **Von Neumann bottleneck.**

The key elements of Von Neumann architecture are:

- Data and instructions are both stored as binary digits
- Data and instructions are both stored in primary storage
- Instructions are fetched from memory one at a time and in order (serially)
- The processor decodes and executes an instruction, before cycling around to fetch the next instruction.
- The cycle continues until no further instruction.

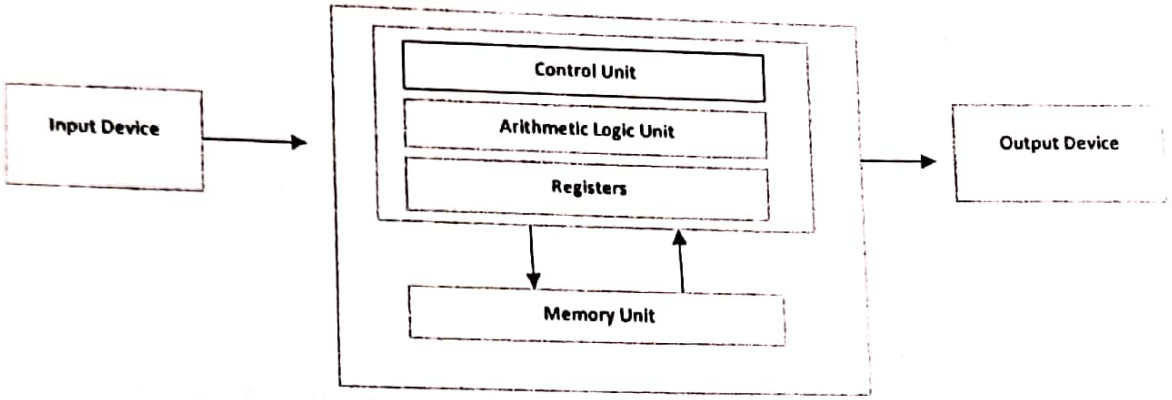



**Figure 2: Von Neumann Architecture**

## 2. Harvard Architecture

In early computer systems, machine instructions were stored on punch cards and data could be stored on another media such as magnetic tapes. This kept the instructions and data entirely separate from one another, known as the Harvard architecture. In modern computer systems this can be achieved by using a central processing unit with two separate memory units, one to store machine instructions and another to store data, which are connected by different buses.

In Harvard architecture, there is no need to make the two memories share characteristics. In particular, the word width, timing, implementation technology and memory address structure can differ. Instruction memory is often wider than data memory. In some systems, instructions can be stored in read-only memory while data memory generally requires read-write memory. In some systems, there is much more instruction memory than data memory so instruction addresses are much wider than data addresses.

A Harvard Architecture is still used for applications which run fixed programs, in areas such as digital signal processing, but not for general purpose computing. The advantage is the increased bandwidth available due to having separate communication channels for instructions and data. However, in a pure Harvard architecture, mechanisms must be provided to separately load the program to be executed into instruction memory and any data to be operated upon into data memory.

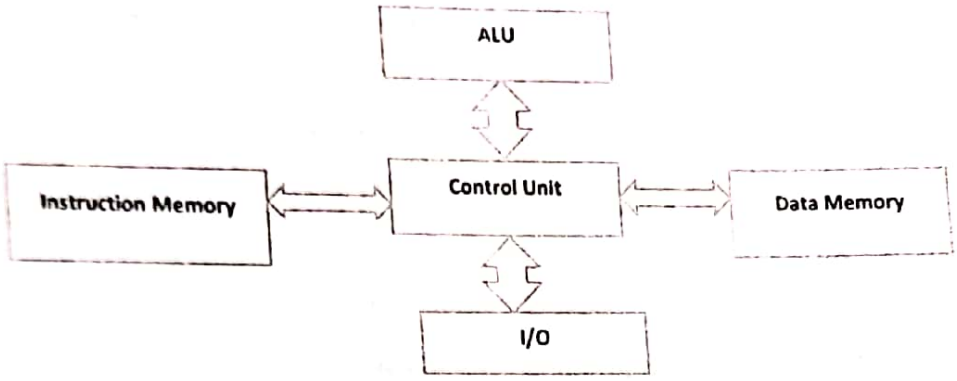



**Figure 3: Harvard Architecture**

## INSTRUCTION FORMATS

An instruction is a command given to the computer to perform specified operation on given data. Each instruction consists of two parts: an opcode (operation code) and an operand. The first part of an instruction, which specifies the operation to be performed, is known as opcode. The second part of an instruction called operand is the data on which computer performs the specified operation.

Computer perform task on the basis of instruction provided. An instruction in computer comprises of groups called fields. These field contains different information as for computers everything is in 0 and 1 so each field has different significance on the basis of which a CPU decide what so perform. The most common fields are:

- Operation field which specifies the operation to be performed like addition.
- Address field which contain the location of operand, i.e., register or memory location.
- Mode field which specifies how operand is to be founded.

An instruction is of various length depending upon the number of addresses it contain. Generally, CPU organization are of three types on the basis of number of address fields:

1. Single Accumulator organization
2. General register organization
3. Stack organization

We know that a machine instruction has an opcode and zero or more operands. Architectures are differentiated from one another by the number of bits allowed per instruction (16, 32, and 64 are the most common), by the number of operands allowed per instruction, and by the types of instructions and data each can process. More specifically, instruction sets are differentiated by the following features:

- Operand storage in the CPU (data can be stored in a stack structure or in registers)
- Number of explicit operands per instruction (zero, one, two, and three being the most common)
- Operand location (instructions can be classified as register-to-register, register-to-memory or memory-to-memory, which simply refer to the combinations of operands allowed per instruction)
- Operations (including not only types of operations but also which instructions can access memory and which cannot)
- Type and size of operands (operands can be addresses, numbers, or even characters)

An instruction format consisted of two fields: The first one containing the operation code and the other containing the address(es). As additional features of address modification became available, it was necessary to add special bit positions in the instruction word to specify functions such as indirect addressing, use of index registers, use of base registers in relative addressing, etc. Still other bits were sometimes used to allow for reference to parts of a data word; this was usually as fractions of the word, as character positions, more recently as byte position.

As registers became common, distinct operation codes were used to refer to register locations; these locations could be specified in many fewer bits than normal addresses, and variable-length instruction formats were developed.

In first organization operation is done involving a special register called accumulator. In second on multiple registers are used for the computation purpose. In third organization the work on stack basis operation due to which it does not contain any address field. It is not necessary that only a single organization is applied, a blend of various organization is mostly what we see generally. On this basis, number of address instructions are classified as:

## THREE ADDRESS INSTRUCTION

Computers with three address instruction formats can use each address field to specify either a processor or a memory operand. Program created are much short in size but number of bits per instruction increase. These instructions make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more information but each micro operation (changing content of register, loading address in address bus etc.) will be performed in one cycle only.

| Opcode | Destination Address | Source Address | Source Address | Mode |
|---|---|---|---|---|

The program in assembly language that evaluate the expression: X = (A+B) * (C+D)

R1, R2 are register

M [ ] is any memory location

ADD R1, A, B R1 = M [A] + M [B]

ADD R2, C, D R2 = M [C] + M [D]

MUL X, R1, R2 M [X] = R1 * R2

The advantage of the three address formats is that it results in short program when evaluating arithmetic expression. The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

## TWO ADDRESS INSTRUCTION

Two address instructions are the most common in commercial computers. Two address can be specified in the instruction. Here again each address field can specify either a process register on a memory word. The destination address can also contain operand.

| Opcode | Destination Address | Source Address | Mode |
|---|---|---|---|

The program to evaluate expression: X = (A+B) * (C+D) is as follows:

R1, R2 are registers

M [ ] is any memory location

```
MOV   R1, A   R1 = M [A]
ADD   R1, B   R1 = R1 + M [B]
MOV   R2, C   R2 = M [C]
ADD   R2, D   R2 = R2 + M [D]
MUL   R1, R2  R1 = R1 * R2
MOV   X, R1   M [X] = R1
```

**ONE ADDRESS INSTRUCTIONS**

One address instruction uses an implied ACCUMULATOR (AC) register for all data manipulation. One operand is in accumulator and other is in register or memory location. Implied means that the CPU already know that one operand is in accumulator so there is no need to specify it. For multiplication and division there is a need for a second register.

| Opcode | Operand/Address of operand | Mode |
|---|---|---|

The program to evaluate expression: X = (A+B) * (C+D) is

AC is accumulator

M [ ] is any memory location

M [T] is temporary location

```
LOAD   A   AC = M [A]
ADD    B   AC = AC + M [B]
STORE  T   M [T] = AC
```

All operations are done between AC register and a memory operand. It is the address of a temporary memory location required for storing the intermediate result.

```
LOAD   C   AC = M [C]
ADD    D   AC = AC + M [D]
MUL    T   AC = AC * M [T]
```

STORE X M [X] = AC

**ZERO ADDRESS INSTRUCTION**

A stack organized does not use an address field for the instruction ADD and MUL. The PUSH & POP instruction however need an address field to specify the operand that communicates with the stack. To evaluate an expression first it is converted to revere Polish Notation i.e. Post Fix Notation.

Expression: X = (A+B) * (C+D)

Post fixed: X = AB+CD+*

Top means top of stack

M [X] is any memory location

| | | |
|---|---|---|
| PUSH | A | TOP = A |
| PUSH | B | TOP = B |
| ADD | | TOP = A+B |
| PUSH | C | TOP = C |
| PUSH | D | TOP = D |
| ADD | | TOP = C+D |
| MUL | | TOP = (C+D) * (A+B) |
| POP | X | M [X] = TOP |

## INSTRUCTION SET DESIGN

One of the most interesting and most analyzed aspect of computer design is instruction set, as it affect so many aspect of the computer system. The instruction defines any of the functions performed by the processor and thus has significant effect on the implementation of the process. The instruction set is the programmer's means of controlling the processor. Thus, programmer requirements must be considered in designing the instruction set. The most important of this fundamental design issues include the following:

- **Operation repertoire:** How many and which operations to provide and how complex operations should be.

- **Data types:** The various types of data upon which operations are perform.
- **Instruction format:** Instruction length, number of assesses size of various fields and so on.
- **Registers:** Number of processor registers that can be referenced by instructions and their use.
- **Addressing:** The mode or modes by which the address of an operand is specified.

These issues are highly interrelated and must be considered together in designing an instruction set.

**ELEMENTS OF A MACHINE INSTRUCTION**

1. **Operation code:** Specifies the operation to be performed (e.g, ADD, I/O). The operation is specified by a binary code, known as the operation code or opcode.
2. **Source and destination operand reference:** This operation specify the input and output locations for the operation.
3. **Results operands reference:** The operation may produce a result
4. **Next instruction reference:** This tells the processor where to fetch the next instruction after the execution of this instruction is complete. The address of the next instruction to be fetched could be either a real address or a virtual address, depending on the architecture. Generally, the distinction is transparent to the instruction set architecture. In most cases, the next instruction to be fetched immediately follows the current instruction.

**INSTRUCTION REPRESENTATION**

Within the computer system, each instruction is represented by a sequence of bits. The instruction is divided into fields, corresponding to the constituent's elements of the instruction. Opcodes are represented by abbreviation called mnemonics that indicate the operation. Common examples include:

ADD add

SUB SUBTRACT

MUL multiply

DIV divide

LOAD Load data form memory

STOR Store data to memory

Operands are also represented symbolically, for example the instruction ADD, R, Y means add the value contained in data location Y to the contents of register R. In this example Y refers to the address of a location in memory and R refers to a particular register. Note that the operation is performed on the contents of a location not on its address: Thus, it is possible to write a machine language program in symbolic form.

X= 413

Y= 414

A simple program would accept this symbolic input, convert opcodes and operand references to binary form, and construct binary machine instructions. However symbolic machine language remains a useful tool for describing machine instructions, and we will use it for that purpose. Let's assume that the variables X and Y corresponds to location 413 and 414 respectively. If we assume a simple set of machine instruction, this operation could be accomplished with three instruction.

1. Load a register with the content of memory location 413.
2. Add the contents of memory location 414 to the register.
3. Store the contents of the register in memory location 413.

## CLASSIFICATION OF INSTRUCTION SET

The design of instruction set influences the design of microprocessor and as a consequence, the microprocessor architecture are of two types:

i. Reduced Instruction Set Computers (RISC)

ii. Complex Instruction Set Computers (CISC).

**Reduced Instruction Set Computers (RISC):** A computer with few instructions and simple construction is called reduced instruction set computer or RISC. RISC architecture is simple and efficient. The major characteristics of RISC architecture are:

1. Relatively few instructions.
2. Relatively few addressing modes.
3. Memory access limited to load and store instructions.

4. All operations are done within the registers of the CPU.
5. Fixed-length and easily-decoded instruction format.
6. Single cycle instruction execution.
7. Hardwired and micro programmed control.

**Complex Instruction Set Computers (CISC):** A computer with large number of instructions is called complex instruction set computer or CISC. Complex instruction set computer is mostly used in scientific computing applications requiring lots of floating point arithmetic. The major characteristics of CISC architecture are:

1. A large number of instructions - typically from 100 to 250 instructions.
2. Some instructions that perform specialized tasks and are used infrequently.
3. A large variety of addressing modes - typically 5 to 20 different modes.
4. Variable-length instruction formats
5. Instructions that manipulate operands in memory.

Because of the limits of early computer technology, most computers were by necessity RISC machines, since most of the software were written in assembly language, there was a drive to build instruction sets of greater sophistication and complexity. These new CISC instruction sets made assembly language programming easier, but they also made it difficult to build high-speed computer hardware.

CISC instructions were harder to decode and since CISC instructions involved long and complex operation sequences, they incurred a major cost by requiring more complicated logic to implement. Second, such instructions were also difficult to interrupt or abort if an exception occurred. Finally, such instructions usually carried many data dependencies that made it more difficult to support advanced architectural techniques. By returning to a RISC design, much faster computers can be built.

## INSTRUCTION FETCH AND EXECUTE

Instruction processing consists of two steps: The processor reads *(fetches)* instructions from memory one at a time and *executes* each instruction. Program execution consists of repeating

the process of instruction fetch and instruction execution. The instruction execution may involve several operations and depends on the nature of the instruction. The processing required for a single instruction is called an *instruction cycle*. Using the simplified two-step description given previously, the instruction cycle is illustrated in Figure 4.

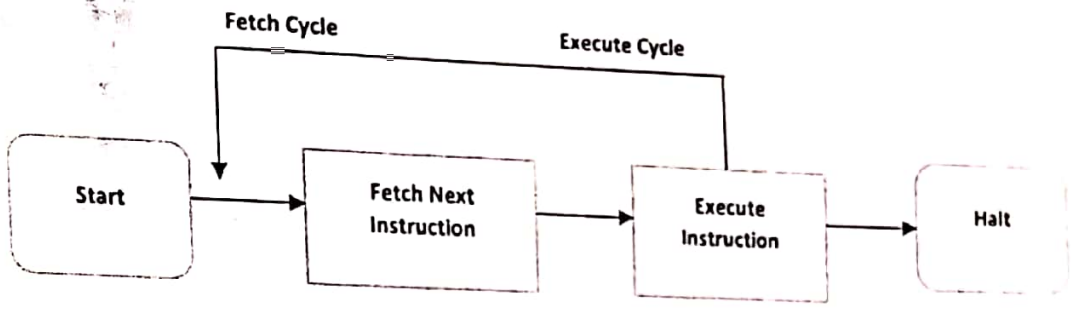



**Figure 4: Basic Instruction Cycle**

The two steps are referred to as the *fetch cycle* and the *execute cycle*. Program execution halts only if the machine is turned off, some sort of unrecoverable error occurs, or a program instruction that halts the computer is encountered. At the beginning of each instruction cycle, the processor fetches an instruction from memory. In a typical processor, a register called the program counter (PC) holds the address of the instruction to be fetched next. Unless told otherwise, the processor basic Instruction cycle always increments the PC after each instruction fetch so that it will fetch the next instruction in sequence (i.e., the instruction located at the next higher memory address).

For example, consider a computer in which each instruction occupies one 16-bit word of memory. Assume that the program counter is set to location 300. The processor will next fetch the instruction at location 300, on succeeding this cycle, it will fetch instructions from locations 301, 302, 303, and so on.

The fetched instruction is loaded into a register in the processor known as the instruction register (IR). The instruction contains bits that specify the action the processor is to take, the processor interprets the instruction and performs the required action. In general, these actions fall into four categories:

**Processor-memory**: Data may be transferred from processor to memory or from memory to processor.

**Processor-I/O**: Data may be transferred to or from a peripheral device or transferring between the processor and I/O module.

**Data processing:** The processor may perform some arithmetic or logic operation on data.

**Control:** An instruction may specify that the sequence of execution be altered. For example, the processor may fetch an instruction from location 149, which specifies that the next instruction be from location 182. The processor will remember this fact by setting the program counter to 182. Thus, on the next fetch cycle, the instruction will be fetched from location 182 rather than 150.

## COMPUTER ARITHMETIC

The Arithmetic and Logic Unit (ALU) is that part of the computer that actually performs arithmetic and logical operations on data. All of the other elements of the computer system- control unit, registers memory, I/0- are there mainly to bring into the ALU for it to process and then take the result back out.

- **Arithmetic and Logic Unit**

An ALU and all electronic components in the computers are based on the use of simple digital logic devices that can store binary digits and perform simple Boolean logic operations. Figure 5 indicates, in general terms, how the ALU is interconnected with the rest of the processor. Data are presented to the ALU in registers and the results of an operation are stored in registers. These registers are temporary storage locations within the processor that are connected by signal paths to the ALU. The ALU may also set flags as the result of an operation.

For example, an overflow flag is set to 1 if the result of a computation exceeds the length of the register into which it is to be stored. The flag values are also stored in registers within the processor. The control unit provides signals that control the operation of the ALU and the movement of the data into and out of the ALU.

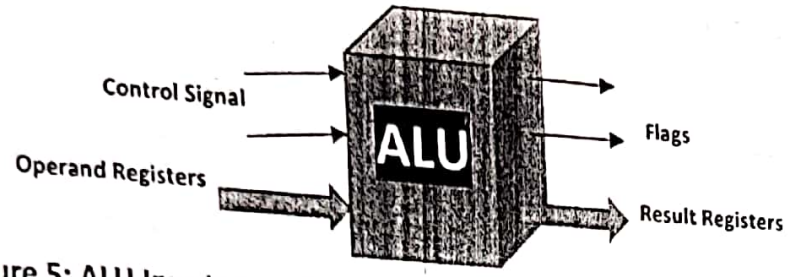



Figure 5: ALU Inputs and Outputs

- **Integer Representation**

In the binary number, arbitrary numbers can be represented with just the digits zero and one the minis sign and the period or radix point.

-1101.01012= -13.312510

Only binary digits (0 and 1) may be used to represent numbers. If we are limited to non-negative integers, the representation is straight forward. An 8 bit word can represent the numbers from 0 to 255.

00000000 = 0

00000001 = 1

00101001 = 41

10000000 = 128

11111111 = 255

- **Fixed-point representation**

Some representations are sometime referred to as fixed point. This is because the radix point (binary point) is fixed assumed to be to the right of the rightmost digit. The programmer can use the representation for binary fractions by scaling the numbers so that the binary poor implicitly positioned at some other location.

## COMPUTER MEMORY SYSTEM

A computer memory is just like a human brain, it stores data and instructions, where data is to be processed and instructions required for processing are stored. The memory is divided into large number of small parts called cells. Each location or cell has a unique address, which varies

from zero to memory size minus one. For example, if the computer has 64k words, then this memory unit has 64 * 1024 = 65536 memory locations. The address of these locations varies from 0 to 65535.

Memory is primarily of three types:

- Cache Memory
- Primary Memory/Main Memory
- Secondary Memory

**Cache Memory**

Cache memory is a very high-speed semiconductor memory which can speed up the CPU. It acts as a buffer between the CPU and the main memory. It is used to hold those parts of data and program which are most frequently used by the CPU. The parts of data and programs are transferred from the disk to cache memory by the operating system, from where the CPU can access them. Cache memory is faster than main memory, consumes less access time as compared to main memory. It stores programs that can be executed within a short period of time and stores data for temporary use.

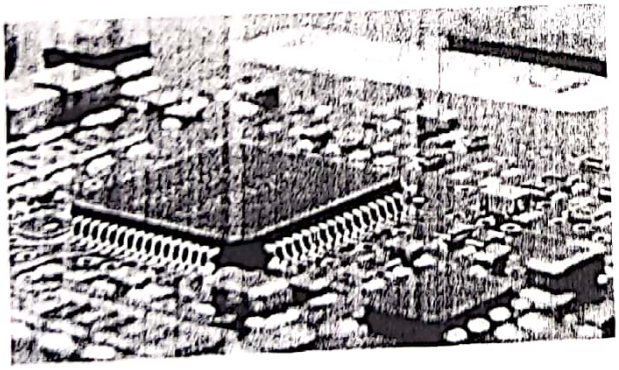


**Figure 6: Cache Memory**

**Primary Memory (Main Memory)**

Primary memory or main memory is a volatile memory that holds only those data and instructions on which the computer is currently working on. It has a limited capacity and data is lost when power is switched off and are generally made up of semiconductor device. These memories are

not as fast as registers. The data and instruction required to be processed resides in the main memory. It is divided into two subcategories RAM and ROM.


**Figure 7: RAM and ROM**

**Random Access Memory (RAM):** RAM is the internal memory of the CPU for storing data, program and program result. It is a read/write memory which stores data when the machine is working and as soon as the machine is switched off, data is erased. Access time in RAM is independent of the address, that is, each storage location inside the memory is as easy to reach as other locations and takes the same amount of time. RAM is of two types: Static RAM (SRAM) and dynamic RAM (DRAM).

**Static RAM (SRAM)**

The word **static** indicates that the memory retains its contents as long as power is being supplied. However, data is lost when the power gets down due to volatile nature. SRAM chips use a matrix of 6-transistors and no capacitors. Transistors do not require power to prevent leakage, so SRAM need not be refreshed on a regular basis. There is extra space in the matrix, hence SRAM uses more chips than DRAM for the same amount of storage space, making the manufacturing costs higher. SRAM is thus used as cache memory and has very fast access.

**Dynamic RAM (DRAM)**

DRAM, unlike SRAM must be continually refreshed in order to maintain the data. This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second. DRAM is used for most system memory as it is cheap and small. All DRAMs are made up of memory cells, which are composed of one capacitor and one transistor.

**...Only Memory**

The memory from which we can only read but cannot write on it. This type of memory is non-volatile and the information is stored permanently during manufacture. A ROM stores instructions that are required to start a computer. This operation is referred to as bootstrap. ROM chips are not only used in the computer but also in other electronic items like washing machine and microwave oven. The various types of ROMs and their characteristics are as follows.

**MROM (Masked ROM)**

The very first ROMs were hard-wired devices that contained a pre-programmed set of data or instructions. This kind of ROMs are known as masked ROMs, which are inexpensive.

**PROM (Programmable Read Only Memory)**

PROM is read-only memory that can be modified only once by a user. The user buys a blank PROM and enters the desired contents using a PROM program. Inside the PROM chip, there are small fuses which are burnt open during programming. It can be programmed only once and is not erasable.

**EPROM (Erasable and Programmable Read Only Memory)**

EPROM can be erased by exposing it to ultra-violet light for a duration of up to 40 minutes. Usually, an EPROM eraser achieves this function. During programming, an electrical charge is trapped in an insulated gate region. The charge is retained for more than 10 years because the charge has no leakage path. For erasing this charge, ultra-violet light is passed through a quartz crystal window (lid). This exposure to ultra-violet light dissipates the charge during normal use, the quartz lid is sealed with a sticker.

**EEPROM (Electrically Erasable and Programmable Read Only Memory)**

EEPROM is programmed and erased electrically, it can be erased and reprogrammed about ten thousand times Both erasing and programming take about 4 to 10 ms (millisecond) and any location can be selectively erase and programmed. Hence, the process of reprogramming is flexible but slow.

**Secondary Memory**

This type of memory is also known as external memory or non-volatile. It is slower than the main memory. Th are used for storing data/information permanently. CPU directly does not access these memories, instead t are accessed via input-output routines. The contents of secondary memories are first transferred to the m memory and then the CPU can access it. For example, disk, CD-ROM, DVD, etc.

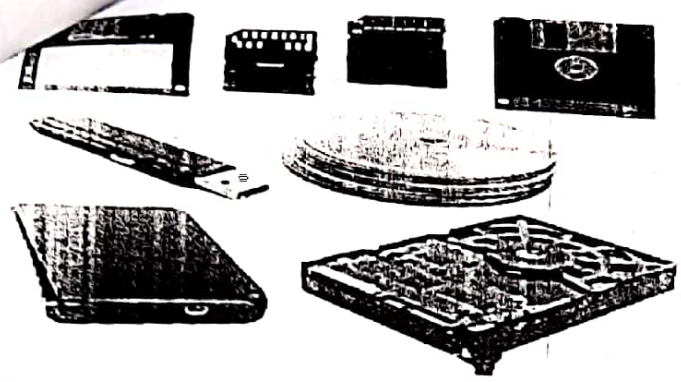


Figure 8: Secondary Memory

## CHARACTERISTICS OF MEMORY

The key characteristics of memory devices or memory system are as follows:

1. Location
2. Capacity
3. Unit of Transfer
4. Access Method
5. Performance
6. Physical type
7. Physical characteristics
8. Organization

**Location:** It deals with the location of the memory device in the computer system. There are three possible locations:

- **CPU:** This is often in the form of CPU registers and small amount of cache
- **Internal or main:** This is the main memory like RAM or ROM. The CPU can directly access the main memory.
- **External or Secondary:** It comprises of secondary storage devices like hard disks, magnetic tapes. The CPU doesn't access these devices directly. It uses device controllers to access secondary storage devices.

**Capacity:** The capacity of any memory device is expressed in terms of word size and number of words

- **Word size:** Words are expressed in bytes (8 bits) and a word can mean any number of bytes. Commonly used word sizes are 1 byte (8 bits), 2bytes (16 bits) and 4 bytes (32 bits).

**Number of words:** This specifies the number of words available in the particular memory device. For example, if a memory device is given as 4K x 16. This means the device has a word size of 16 bits and a total of 4096(4K) words in memory.

**Unit of Transfer:** It is the maximum number of bits that can be read or written into the memory at a time. In case of main memory, it is mostly equal to word size. In case of external memory, unit of transfer is not limited to the word size; it is often larger and is referred to as blocks.

**Access Methods:** It is a fundamental characteristic of memory devices. It is the sequence or order in which memory can be accessed. There are three types of access methods:

- **Random Access:** If storage locations in a particular memory device can be accessed in any order and access time is independent of the memory location being accessed. Such memory devices are said to have a random-access mechanism. RAM IC's use this access method.
- **Serial Access:** If memory locations can be accessed only in a certain predetermined sequence, this access method is called serial access. Magnetic Tapes, CD-ROMs employs serial access methods.
- **Semi Random Access:** Memory devices such as magnetic hard disks use this access method were each track has a read/write head thus each track can be accessed randomly but access within each track is restricted to a serial access.

**Performance:** The performance of the memory system is determined using three parameters

- **Access Time:** In random access memories, it is the time taken by memory to complete the read/write operation from the instant that an address is sent to the memory. For non-random access memories, it is the time taken to position the read write head at the desired location. Access time is widely used to measure performance of memory devices.
- **Memory cycle time:** It is defined only for Random Access Memories and is the sum of the access time and the additional time required before the second access can commence.
- **Transfer rate:** It is defined as the rate at which data can be transferred into or out of a memory unit.

**Physical type:** Memory devices can be either semiconductor memory (like RAM) or magnetic surface memory (like Hard disks).

**Physical Characteristics:** Volatile/Non- Volatile: If a memory device continues hold data even if power is turned off then the memory device is non-volatile else it is volatile.

**Organization:** Erasable/Non-erasable: The memories in which data once programmed cannot be erased are called Non-erasable memories. Memory devices in which data in the memory can be erased is called erasable memory e.g. RAM (erasable), ROM (non-erasable)

## MEMORY HIERARCHY

In computer architecture the memory hierarchy is a concept used for storing & discussing performance issues in computer architectural design, algorithm predictions, and the lower level programming constructs such as involving locality of reference. The memory hierarchy in computer storage distinguishes each level in the hierarchy by response time. Since response time, complexity and capacity are related, the levels may also be distinguished by their performance and controlling technologies.

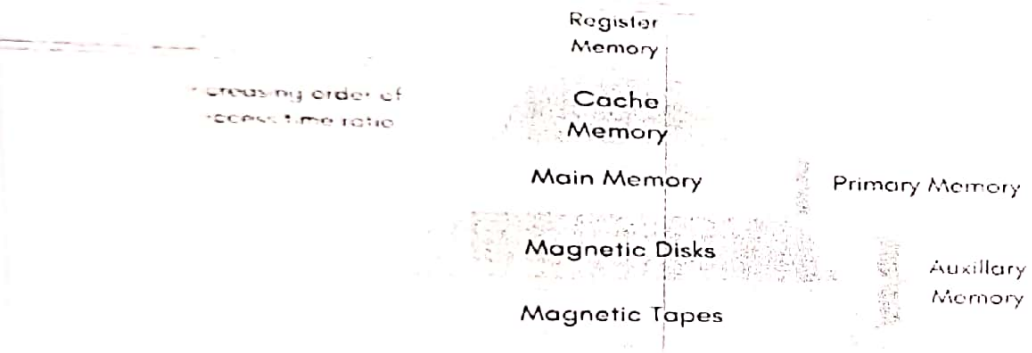



**Figure 9: Memory Hierarchy**

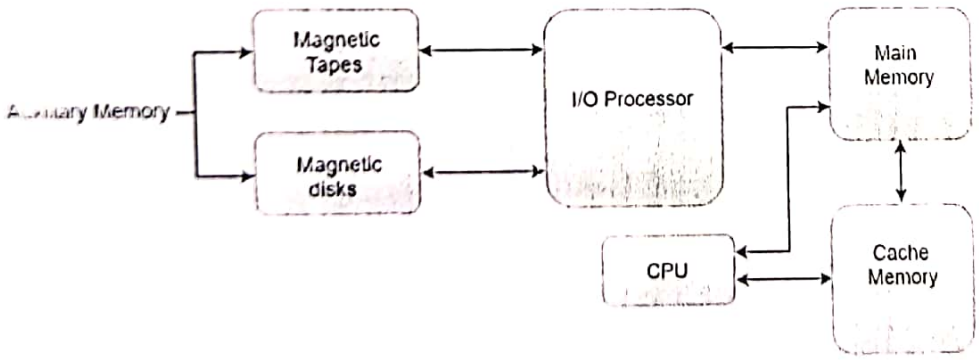



**Figure 10: Memory Hierarchy in Computer System**

The total memory capacity of a computer can be visualized by hierarchy of components. The memory hierarchy system consists of all storage devices contained in a computer system from the slow auxiliary memory to fast main memory and to smaller cache memory. Auxiliary memory access time is generally 1000 times that of the main memory, hence it is at the bottom of the hierarchy. The main memory occupies the central position

...use it is equipped to communicate directly with the CPU and with auxiliary memory devices through ...nput/output processor (I/O).

When program not residing in main memory is needed by the CPU, they are brought in from auxiliary memory. Programs not currently needed in main memory are transferred into auxiliary memory to provide space in main memory for other programs that are currently in use. The cache memory is used to store program data which is currently being executed in the CPU. Approximate access time ratio between cache memory and main memory is about 1 to 7~10. Register is a very fast computer memory, used to store data/instruction in execution.

## Virtual Memory

Virtual memory is a memory management capability of an operating system (OS) that uses hardware and software to allow computers to compensate for physical memory shortages by temporarily transferring data from RAM to disk storage. Virtual address space is increased using active memory in RAM and inactive memory in hard disk drives to form contiguous addresses that hold both the application and its data. Virtual memory was developed at a time when physical memory installed RAM was expensive.

Computers have a finite amount of RAM, so memory can run out, especially when multiple programs run at the same time. A system using virtual memory uses a section of the hard drive to emulate RAM. With virtual memory, a system can load larger programs or multiple programs running at the same time, allowing each one to operate as if it has infinite memory and without having to purchase more RAM.

While copying virtual memory into physical memory, the OS divides memory into page files or swap files with a fixed number of addresses. Each page is stored on a disk and when the page is needed, the OS copies it from the disk to main memory and translates the virtual addresses into real addresses.

### TYPES OF VIRTUAL MEMORY

A computer's memory management unit (MMU) handles memory operations, including managing virtual memory. In most computers, the MMU hardware is integrated into the CPU. There are two ways in which virtual memory is handled: paged and segmented.

**PAGING:** Paging divides memory into sections or paging files, usually approximately 4 KB in size. When a computer uses up its RAM, pages not in use are transferred to the section of the hard drive designated for virtual memory using a swap file. A swap file is a space set aside on the hard drive as the virtual memory extensions of the computer's RAM. When the swap file is needed, it's sent back to RAM using a process called page swapping. This system ensures that computer's OS and applications don't run out of real memory.

paging process includes the use of page tables, which translate the virtual addresses that the OS and applications use into the physical addresses that the MMU uses. Entries in the page table indicate whether or not the page is in real memory. If the OS or a program doesn't find what it needs in RAM, then the MMU responds to the missing memory reference with a page fault exception to get the OS to move the page back to memory when it's needed. Once the page is in RAM, its virtual address appears in the page table.

**SEGMENTATION:** Segmentation is also used to manage virtual memory. This approach divides virtual memory into segments of different lengths and segments not in use in memory can be moved to virtual memory space on the hard drive. Segmented information or processes are tracked in a segment table, which shows if a segment is present in memory, whether it's been modified and what its physical address is.

Some virtual memory systems combine segmentation and paging. In this case, memory gets divided into frames or pages. The segments take up multiple pages and the virtual address includes both the segment number and the page number.

**HOW TO MANAGE IT**

Operating systems have default settings that determine the amount of hard drive space to allocate for virtual memory. That setting will work for most applications and processes, but there may be times when it's necessary to manually reset the amount of hard drive space allocated to virtual memory, such as with applications that depend on fast response times or when the computer has multiple Hard Disk Drives (HDDs).

When manually resetting virtual memory, the minimum and maximum amount of hard drive space to be used for virtual memory must be specified. Allocating too little HDD space for virtual memory can result in a computer running out of RAM. If a system continually needs more virtual memory space, it may be wise to consider adding RAM.

**PROS OF USING VIRTUAL MEMORY**

Among the primary benefits of virtual memory is its ability to handle twice as many addresses as main memory. It uses software to consume more memory by using the HDD as temporary storage while MMUs translate virtual memory addresses to physical addresses via the CPU. Programs use virtual addresses to store instructions and data; when a program is executed, the virtual addresses are converted into actual memory addresses.

Virtual memory also frees applications from managing shared memory and saves users from having to add memory modules when RAM space runs out. The use of virtual memory has its trade-offs, particularly with speed, it's generally better to have as much physical memory as possible so programs work directly from RAM or physical memory.

use of virtual memory slows a computer because data must be mapped between virtual and physical memory, which requires extra hardware support for address translations. In a virtualized computing environment, administrators can use virtual memory management techniques to allocate additional memory to a virtual machine (VM) that has run out of resources. Such virtualization management tactics can improve VM performance and management flexibility.

## DATA REPRESENTATION

Computers are classified according to functionality, physical size and purpose. Functionality computers could be analog, digital or hybrid. Digital computers process data that is in discrete form whereas analog computers process data that is continuous in nature. Hybrid computers on the other hand can process data that is both discrete and continuous. In digital computers, the user input is first converted and transmitted as electrical pulses that can be represented by two unique states ON and OFF. The ON state may be represented by a "1" and the off state by a "0". The sequence of ON'S and OFF'S forms the electrical signals that the computer can understand.

A digital signal rises suddenly to a peak voltage of +1 for some time then suddenly drops -1 level on the other hand, an analog signal rises to +1 and then drops to -1 in a continuous version. When a digital signal is to be sent over analog telephone lines e.g. e-mail, it has to be converted to analog signal. This is done by connecting a device called a **modem** to the digital computer. This process of converting a digital signal to an analog signal is known as **modulation**. On the receiving end, the incoming analog signal is converted back to digital form in a process known as **demodulation**.

Physical size computers could be mini, micro, mainframe or supercomputer. Minicomputers are midsized computers, in size and power mini computers lie between workstations and mainframes. It has distinction between small minicomputers and workstation, but in general, a minicomputer is a multiprocessing system capable of supporting from 4 to 200 users simultaneously. Microcomputer became the most common type of computer in the late 20th century. The term microcomputer was introduced with advent of system based on single chip microprocessor.

Mainframe computers was created to distinguish the traditional, large, institutional computer intended to service multiple users from the smaller, single user machines. These computers are capable of handling and processing very large amounts of data quickly. Mainframe computer are used in large institution such as government, banks and large cooperation's. They are measured in MIPS (million instructions per second) and can respond to hundreds of millions of users at a time. Super computer is focused on performing tasks involving

...numerical calculations such as weather forecasting, theoretical astrophysics and complex scientific computation.

Computers could be general and special purpose; general purpose computer are computers that are utilized for numerous sorts of work, yet each one of those assignment is ordinary, for example, writing a letter with word processing, setting up a record, printing reports, making a database, and so on. Special purpose computers are built for a particular task. The CPU capabilities in this additionally relate to that particular function.

**Reason for use of binary system in computers**

It has proved difficult to develop devices that can understand natural language directly due to the complexity of natural languages. However, it is easier to construct electric circuits based on the binary or ON and OFF logic. All forms of data can be represented in binary system format. Other reasons for the use of binary are that digital devices are more reliable, small and use less energy as compared to analog devices.

**Types of data representation**

Computers not only process numbers, letters and special symbols but also complex types of data such as sound and pictures. However, these complex types of data take a lot of memory and processor time when coded in binary form. This limitation necessitates the need to develop better ways of handling long streams of binary digits. Higher number systems are used in computing to reduce these streams of binary digits into manageable form. This helps to improve the processing speed and optimize memory usage.

**Number systems representation**

A number system is a set of symbols used to represent values derived from a common base or radix. As far as computers are concerned, number systems can be classified into two major categories:

- Decimal number system
- Binary number system
- Octal number system
- Hexadecimal number system

**Decimal number system**

The term decimal is derived from a Latin prefix deci, which means ten. Decimal number system has ten digits ranging from 0-9. Because this system has ten digits; it is also called a base ten number system or denary number system. A decimal number should always be written with a subscript 10. But since this is the most widely used number system in the world, the subscript is usually understood and ignored in written work. However, when many number systems are considered together, the subscript must always be put so as to differentiate the number systems. The magnitude of a number can be considered using these parameters:

- Absolute value
- Place value or positional value
- Base value

**The absolute value** is the magnitude of a digit in a number. For example, the digit 5 in 7458 has an absolute value of 5 according to its value in the number line.

**The place value** of a digit in a number refers to the position of the digit in that number i.e. whether; tens, hundreds, thousands etc. The total value of a number is the sum of the place value of each digit making the number.

**The base value** of a number also known as the radix, depends on the type of the number systems that is being used. The value of any number depends on the radix. For example, the number 10010 is not equivalent to 1002.

**Binary number system**

It uses two digits namely, 1 and 0 to represent numbers. Unlike in decimal numbers where the place value goes up in factors of ten, in binary system, the place values increase by the factor of 2. Binary numbers are written as X2. Consider a binary number such as 10112, the right most digit has a place value of 1×20 while the left most has a place value of 1×23.

**Octal number system**

Consists of eight digits ranging from 0-7, the place value of octal numbers goes up in factors of eight from right to left.

**Hexadecimal number system**

This is a base 16 number system that consists of sixteen digits ranging from 0-9 and letters A-F where A is equivalent to 10, B to 11 up to F which is equivalent to 15 in base ten system. The place value of hexadecimal numbers goes up in factors of sixteen. A hexadecimal number can be denoted using 16 as a subscript or capital letter H to the right of the number. For example, 94B can be written as $94B_{16}$ or 94BH.

To convert numbers from one system to another, the following conversions will be considered.

- Converting between binary and decimal numbers.
- Converting octal numbers to decimal and binary form.
- Converting hexadecimal numbers to decimal and binary form.

**Converting binary numbers to decimal numbers**

To convert a binary number to a decimal number, we proceed as follows:

- First, write the place values starting from the right-hand side.

Write each digit under its place value.

- Multiply each digit by its corresponding place value.
- Add up the products, the answer will be the decimal number in base ten.

EXAMPLE

Convert $101101_2$ to base 10 (or decimal) number

Place value $2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Binary digits 1 0 1 1 0 1

Multiply each digit by its place value

$N_{10} = (1*2^5) + (0*2^4) + (1*2^3) + (1*2^2) + (0*2^1) + (1*2^0)$

$N_{10} = 32+0+8+4+0+1$

$=45_{10}$

NB: remember to indicate the base subscript since it is the value that distinguishes the different systems.

**Converting a decimal fraction to binary**

Divide the integral part continuously by 2, for the fractional part, proceed as follows:

Multiply the fractional part by 2 and note down the product

- Take the fractional part of the immediate product and multiply it by 2 again.
- Continue this process until the fractional part of the subsequent product is 0 or starts to repeat itself.
- Combine the two parts together to set the binary equivalent.

**Example**

i. Convert $0.375_{10}$ into binary form

$0.375\times2=0.750$

$0.750\times2=1.500$

$0.500\times2=1.000$ (fraction becomes zero)

Therefore $0.375_{10}=0.011_2$

NB: When converting a real number from binary to decimal, work out the integral part and the fractional parts separately then combine them.

**Converting octal numbers to decimal numbers**

To convert a base 8 number to its decimal equivalent we use the same method as we did with binary numbers. However, it is important to note that the maximum absolute value of an octal digit is 7. For example, 982 is not

octal number because digit 9 is not an octal digit, but 7368 is valid because all the digits are in the range 0-7. Example below shows how to convert an octal number to a decimal number.

**Example**

Convert $512_8$ to its base 10 equivalent

Place value $8^2$ $8^1$ $8^0$

64 8 1

Octal digit 5 1 2

Write each number under its place value as shown below

Multiply each number by its place value

$N_{10} = (5 \times 8^2) + (1 \times 8^1) + (2 \times 8^0)$

$= (5 \times 64) + 8 + 2$

$= 320 + 8 + 2$

$N_{10} = 330_{10}$

Therefore $5128 = 33010$

**Converting octal numbers to binary numbers**

To convert an octal number to binary, each digit is represented by three binary digits because the maximum octal digit i.e. 7 can be represented with a maximum of seven digits. See table 1:

**Table 1: Octal to Binary**

| Octal Digits | Binary Equivalent |
|---|---|
| 0 | 000 |
| 1 | 001 |
| 2 | 010 |
| 3 | 011 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |

**Example**

i. Convert octal number $321_8$ to its binary equivalent

**Solution**

Working from left to the right, each octal number is represented using three digits and then combined we get the final binary equivalent. Therefore:

=011₂

=010₂

=001₂

Combining the three from left to right

3 2 1

011 010 001

$321_8 = 011010001_2$

**Converting hexadecimal numbers to decimal number**

To convert hexadecimal number to base 10 equivalent we proceed as follows:

First, write the place values starting from the right-hand side.

- If a digit is a letter such as 'A' write its decimal equivalent
- Multiply each hexadecimal digit with its corresponding place value and then add the products.
- The following examples illustrate how to convert hexadecimal number to a decimal number

**Example**

Convert the hexadecimal number $111_{16}$ to its binary equivalent

**Solution**

Place each number under its place value.

$16^2$ $16^1$ $16^0$

1 1 1

256 x1= 256
16 x 1 = 16
1 x 1= + 1
273
Therefore $111_{16} = 273_{10}$

**Converting Hexadecimal Numbers to Binary Numbers**

Since F is equivalent to a binary number 11112 the hexadecimal number are therefore represented using 4 digits as shown in the table 2 below.

**...decimal Table**

| Hexadecimal Digits | Decimal Equivalent | Binary Equivalent |
|---|---|---|
| 00 | 00 | 0000 |
| 01 | 01 | 0001 |
| 02 | 02 | 0010 |
| 03 | 03 | 0011 |
| 04 | 04 | 0100 |
| 05 | 05 | 0101 |
| 06 | 06 | 0110 |
| 07 | 07 | 0111 |
| 08 | 08 | 1000 |
| 09 | 09 | 1001 |
| A | 10 | 1010 |
| B | 11 | 1011 |
| C | 12 | 1100 |
| D | 13 | 1101 |
| E | 14 | 1110 |
| F | 15 | 1111 |

The simplest method of converting a hexadecimal number to binary is to express each hexadecimal digit as a four bits' binary digit number and then arranging the group according to their corresponding positions as shown in the example below.

**Converting binary numbers to hexadecimal numbers**

To convert binary numbers to their binary equivalents, simply group the digits of the binary number into groups of four from right to left e.g. 11010001.The next step is to write the hexadecimal equivalent of each group e.g.

1101- D

0001- 1

The equivalent of 11010001 is D1H or $D1_{16}$

Example

Convert $321_{16}$ to its binary equivalents

| | 3 | 2 | 1 |
|---|---|---|---|
| Binary Equivalent | 0011 | 0010 | 0001 |

Combining the three sets of bits, we get $001100100001_2$

$321_{16} = 001100100001_2$

## BASIC LOGIC DESIGN

The operation of the digital computer is based on the storage and processing of binary data. In this unit we suggest how storage elements and circuits can be implemented in digital logic specifically with combinational and sequential circuits. Electric circuits in computers have little memory control devices called Logic Gates. Logic gates put in binary values and output a binary value as well. A train of logic gates is also known as a Logic circuit and they are designed to carry a specific function and have a switch on and off.

Logic Gates can be found in all devices such as computers and air conditioners. They perform basic logical functions and are the fundamental building blocks of digital integrated circuits . Most logic gates take an input of two binary values, and output a single value of a 1 or 0. Some circuits may have only a few logic gates, while others, such as microprocessors, may have millions of them. There are many types of gates such as the NOT and AND gate which has different outputs, some have 2 and others have 1 output.

### Logic Gates

These are examples of gates:

- NOT GATE
- AND GATE
- OR GATE
- NAND GATE
- NOR GATE
- XOR GATE

### Truth Tables

Truth tables are used to track down the output of logic gates or a circuit. The NOT Gate is the only gate to have one output and all the other gates have two outputs. Truth tables are constructed by binary values (1s and 0s) which are used to make different combinations for different gates.

### Boolean Expression:

As with any algebra, boolean algebra makes use of variables and operations. In this case, the variables and operations are logical variables and operations. Thus, a variable may take on the value 1 (TRUE) or 0 (FALSE).

logical operations are AND, OR and NOT, which are symbolically represented by dot, plus sign, and

bar:

A AND B=A-B

A OR B=A+B

NOT A = A

The operation AND yields true (binary value 1) if and only if both of its operands are true. The operation OR yields true if either or both of its operands are true. The unary operation NOT inverts the value of its operand.

For example, consider the equation

D=A+ (B, C)

D is equal to 1 if A is 1 or if both B = 0 and C = 1. Otherwise D is equal to 0.

Logical NOT is often indicated by an apostrophe: NOT A = A'.

Several points concerning the notation are needed. In the absence of parentheses, the AND operation takes precedence over the OR operation. Also, when no ambiguity will occur, the AND operation is represented by simple concatenation instead of the dot operator. Thus,

A+B·C=A+ (B·C) = A+BC

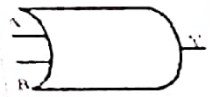
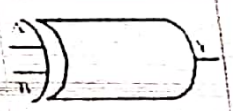
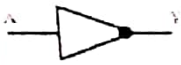
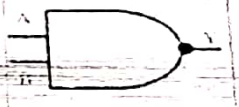
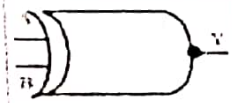
Take the AND of B and C; then take the OR of the result and A.

**Logic Circuits**

When Logic Gates are put together, they can carry out a function, to find out what function they do you can produce a truth table for it. In this truth table you should have the inputs, for example if there are only 2 inputs then you will only have the values ( 0 0 ) ( 1 0 ) ( 0 1 ) and ( 1 1 ) , the intermediate values (which are values which you get in the 'middle' or when you have a logic gate, they can be labelled any letter you want however do not repeat any letters) and output ( mainly written as X). When you have one input for example 0 1 into an AND gate, it will output 0, this 0 if there is no more gates left will be the output, if there is more, than it is an intermediate value.

asic Logic Gates

# Basic Logic Gates

| Logic | Schematic | Boolean Expression | Truth Table | English Expression |
|---|---|---|---|---|
| AND |  | $A \cdot B = Y$ | A B Y<br>0 0<br>0 1<br>1 0<br>1 1 | The only time the output is positive is when all the inputs are positive. |
| OR |  | $A+B=Y$ | A B Y<br>0 0<br>0 1<br>1 0<br>1 1 | The output will be positive when any one or all inputs are positive. |
| XOR |  | $A \oplus B = Y$ | A B Y<br>0 0<br>0 1<br>1 0<br>1 1 | The only time the output is positive is when the inputs are not the same |
| NOT |  | $\bar{A}=Y$ | A Y<br>0<br>1 | The output is the opposite of the input |
| NAND |  | $\overline{A \cdot B}=Y$ | A B Y<br>0 0<br>0 1<br>1 0<br>1 1 | The output is positive provided all the inputs are not positive. |
| NOR |  | $\overline{A+B}=Y$ | A B Y<br>0 0<br>0 1<br>1 0<br>1 1 | The only time the output is positive is when all the inputs are negative |
| XNOR |  | $\overline{A \oplus B}=Y$ | A B Y<br>0 0<br>0 1<br>1 0<br>1 1 | The only time the output is positive is when all the inputs are the same |

Logic Circuits in real life can be found in any electronics, air conditioners to televisions, however the amount of logic gates used can differ. Usually electronics will use as little logic gates as possible to carry out a function since this will reduce the price of the components and evidently leads to more profit for the owners. You can use the NAND gate to make any gate you want, for example you can use the NAND gate to make an AND gate or even a XOR gate, this is what you call a building block.

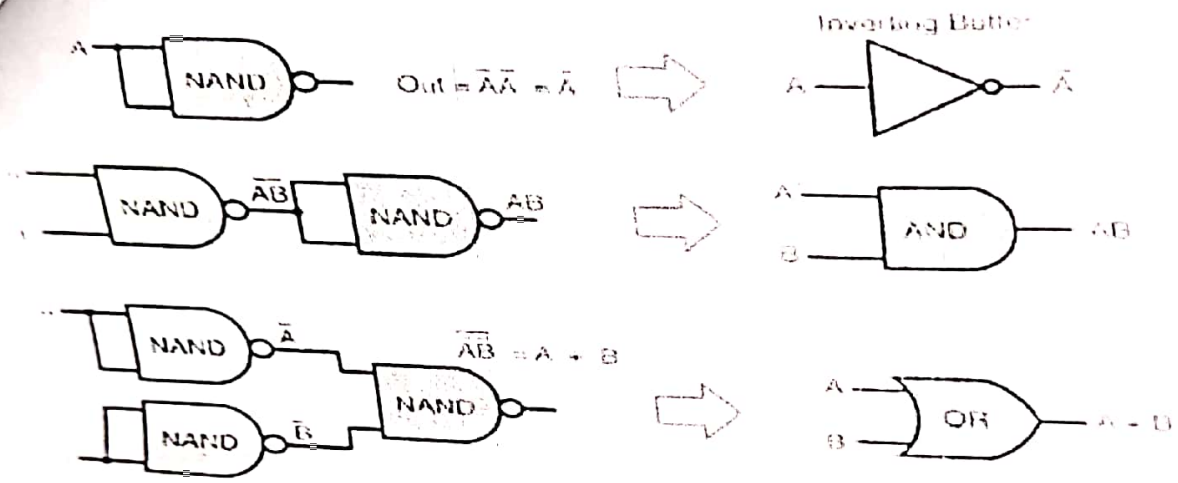



**Figure 11: Logic Gates**

The logic gates that we've presented here are abstract representations of real devices. A logic gate describes any device that can take in values of 0 or 1 and output a 0 or 1 according to its truth table. In most modern computers, logic gates are built using transistors combined with other electrical components like resistors and diodes. Those are all wired together to make sure that they transform the inputs in the way we expect.

Our powerful computers now require billions of gates, so manufacturers have figured out how to make electronic parts very small. A Mac for example has almost 5.6 billion transistors that are just 14 nanometers wide. We can understand and use logic gates without needing to know exactly how they're implemented. That's the power of an abstraction, enabling us to ignore the details and focus on the higher-level functionality.