COMPUTER ORGANIZATION

STUDY MATERIAL

WEIGHTAGE of Marks in GATE - CS/IT

Computer organization

YEAR	TOTAL NO. OF QUESTIONS	TOTAL MARKS
2013	4	7
2012	4	6
2011	7	13
2010	4	7
2009	6	11
2008	13	22
2007	9	16
2006	6	12
2005	9	16
2004	9	16
2003	4	8
2002	6	10
2001	7	12

2000	2	4
1999	5	10
1998	5	10
1997	4	8
1996	4	8
1995	7	12
1994	Only conventional questions are given	
1993	2	4
1992	4	7

Syllabus

Chapter	Name
I	**C.P.U Organization & Design** Topics: Machine Instructions and Addressing Modes ALU and Data Paths
II	**MicroProgrammed Control Unit** Topics: Design and Applications of Hardware and MicroProgrammed Control Unit
III	**Input – output Organization** Topics : I/O Interface; (Interrupt & DMA); Serial Communication interface
IV	**Pipelining** Topics: Parallel Processing; Pipelining types and Applications; RISC & CISC
V	**Memory Organization** Topics: Cache Memory, Main Memory, Secondary Storage Memory and Memory Interfacing
VI	**Computer Arithmetic** Topics : Fixed and floating point operations and booth multiplication

CHAPTER-1

CPU ORGANIZATION & DESIGN

CPU is made up of three major parts:

1. **Arithmetic and Logic Unit (ALU)**
2. **Control Unit (CU) and**
3. **Processing Registers**

- ALU is used to perform the required Arithmetical and logic operation under the directions of control unit.
- C.U. supervises the transfer of information among the Registers and instructs the ALU as to which operation to perform.
- Processing Registers are used to store the data during execution
- The Computer instruction set provides the specifications for the design of the CPU.

Control Word: It is a binary word; generated by the CPU to perform one of the various operations.

Control Register: It is used to store the control word.

- If the length of the control word is n bit, total no. of operations that can be used to perform 2^n ; ranges from 00----0 (n) to 1-----1(n) and each one memory combination is used to assign one operation.

Microoperation : It is an operation executed on data stored in registers. Micro operation is a basic register to register transfer operation.

- Instruction is divided into 2 parts:
  1. Operation part (most side)
  2. Operand part (least side).
- For ‘n’ address bit CPU total no. of memory location to be accessed is 2^n and each memory location is able to store 1 word.
- Length of the word varies from one CPU to other and depends on the no. of data bits that can be transferred at a time (i.e. word size is equal to the no. of data bits a CPU has)
- The total memory is divided in to 2 parts namely
  1. User memory 2.Stack memory
- User memory is under the control of only user, but stack memory is under the control of both CPU and user.

- ☞ CPU can access the stack memory during interrupt for storing the address of the instruction which is interrupted.
- ☞ Stack organization is very effective for evaluating arithmetic expression.

Stack organization : It is a storage device that stores the information in such a manner that the item stored last is the first item to retrieve. Means LIFO (Last in first out)

- ☞ The register that holds the address of the stack is known as stack pointer.
- ☞ Push and pop are used to access stack memory.
- ☞ For accessing 2^n words stack memory, the length of the (stack pointer) required is 'n' bits.
- ☞ Always, SP is pointed at top of the stack; it is decremented during push operation and incremented during pop operation.
- ☞ In 8085 for pushing → pre decrement and for poping → post increment
- ☞ 2 no. of flags are used to know the status of the stack: 1. Empty 2. Full
- ☞ Initially, stack cleared to '0'. So EMPTY is set and Full is cleared.
- ☞ If all memory locations are filled in the stack; then empty flag resets and Full flag sets.
- ☞ But most computer do not provide hardware for checking over flow (Full stack) or under flow (empty stack); In this case the stack limits can be checked by using 2 processing Registers.
- ☞ One to hold the upper limit and other to hold the lower limit address
- ☞ After PUSH operation; SP is compared with the upper limit Register
- ☞ After POP operation SP is compared with the lower limit register

Different ways for performing the Arithmetical operations:

1. In fix notation (Ex : A+B)
2. Pre fix or polish (Ex : +AB)
3. Reverse polish notation (RPN) (Ex: AB+)

Stack works on RPN (Called as post fix notation)

Conversion of Infix to RPN

Ex: 1 $\Rightarrow A\times B+C\times D$

$= (AB\times)+\left(CD\times\right) = AB\times CD\times +$

Ex: 2 $(A+B)\times[C(D+E)+F]$

First complete the inner side of parenthesis

$= (AB+)[C\times(DE+)+F] =$

$= (AB+)[DE+C\times F+]$ $= AB+DE+C\times F+\times$

$= (AB+)\left[(CDE+\times)+F\right]$

$= (AB+)\left[CDE+\times F+\right]$

$= AB+CDE+\times F+\times$

Conversion of RPN to Infix

Ex:1 $AB\times CD\times +$

= $(AB\times)+(CD\times)$

= $(A\times B)+(C\times D)$

= $AB+CD$

Ex: 2 $AB+DE+C\times F+\times$

= $(A+B)\times[C(D+E)+F]$

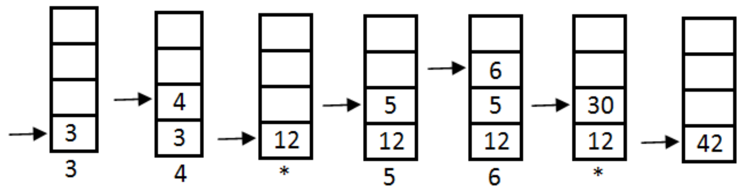
☞ RPN is used in some electronic calculators

☞ Before evaluating the operation, the arithmetic expression must be converted into RPN, the operations are pushed on to stack in the order in which they appear.

☞ Numerical example : for the data $(3\times 4)+(5\times 6)$ using RPN

$34\times 56\times +$

Stack operations :



Compliers works generally with polish notation.

☞ In scientific calculators, some operations are performed with RPN and others with polish notation.

☞ The bits of the instructions are divided in to 2 fields, called as Operation field and Operand field.

☞ Again operand field is divided in to

1. Address field
2. Mode field.

☞ Operands residing in the memory are specified by their memory address

☞ Operands residing in the processing Registers are specified with a Register address (name)

☞ In any system; if k bits are used to specify the address of the Register, then the same CPU has 2^k no. of registers (max. no.)

☞ The no. of address fields in the instruction format of a computer depends on the internal organization of its registers.

Most computers fall into one of 3 types of computer organization

1. Single accumulator organization

2. General register organization

3. Stack organization

<u>Single accumulator organization</u>: (Advantage: length of the opcode is low, so it can be executed at faster rate.)

- ☞ In this all operations are performed with an implied accumulators
- ☞ It has only one address field. Instruction format:

$\rightarrow$

$op-code$	$Address$

Ex: ADD x

SUB x where 'x' is the address of the operand

<u>General Register organization:</u> (Length of the OP code is more)

The instruction format needs 3 Register address fields or 2 Register address fields.

Ex: ADD R1, R2, R3

R1=R2+R3

☞ For data transfer MOV operations, it requires 2 no. Registers.

Ex: MOV R1, R2

For Arithmetical; 3 Register Required

For Data transfer; 2 Register Required

<u>Stack organization CPU:</u>

☞ It uses push & POP instructions.

☞ Instruction format - $\boxed{op-code}$

☞ Push 'x' means, the word at the address of the 'x' is pushed to the top of the stack memory.

- In this operation, instruction don't use an address field in the stack organized computer, because the specified operation is performed on the two items that are on the top of the stack.

Depending on the length of the operand, instructions are divided as follows:

Three address Instruction

Ex: ADD R1, A, B R1=A+B

Two address instructions:

- There are used in the most common in commercial computers
- They have 2 no. of address registers in operand field

Ex1: MOV R1, R2

EX2: ADD R1, R2 R1=R1+R2

One address Instructions

- It is used for single Accumulator organization
- Implied Register is Accumulator
- All operations are performed in between Accumulator & Memory operand

Ex: Load A

ADD B

MUL C

Zero address instructions: (It is used for stack operation)

A stack organized CPU does not use the address field for the instruction

Ex: ADD, MUL etc.

- The push and POP operation need address field
- To evaluate the arithmetical expressions in a stack computer, it is necessary to convert the expression in to reverse polish rotation (RPN)

Ex: ADD

SUB

MUL during stack access

DIGITAL SYSTEMS

STUDY MATERIAL

CONTENT

CHAPTER-1

BINARY SYSTEM

Base Conversion: A number $a_n, a_{n-1} \ldots a_2, a_1 a_0 \cdot a_{-1} a_{-2} a_{-3} \ldots$ expressed in a base r system has coefficien multiplied by powers of r.

$$\boxed{a_n r^n + a_{n-1} r^{n-1} + a_{n-2} r^{n-2} + \ldots + a_1 r + a_0 + a_{-1} r^{-1} + a_{-2} r^{-2} + a_{-3} r^{-3} + \ldots} \quad \ldots(A)$$

Coefficients a_j; range from 0 to $r - 1$

Key Points:

To convert a number of base r to decimal is done by expanding the number in a power series as in (A) Then add all the terms.

Example 1: Convert following Binary number $(11010.11)_2$ in to decimal number.

Solution:

Base r = 2

$1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2}$

$(11010.11)_2 = (26.75)_{10}$

Example 2: Convert $(4021.2)_5$ in to decimal equivalent

Solution: $4\times 5^3 + 0\times 5^2 + 2\times 5^1 + 1\times 5^0 + 2\times 5^{-1}$

$= (511.4)_{10}$

Example 3: Convert $(127.4)_8$ in to decimal equivalent.

Solution: $1\times 8^2 + 2\times 8^1 + 7\times 8^0 + 4\times 8^{-1}$

$= (87.5)_{10}$

Numbers with Different bases:

Decimal (r = 10)	Binary (r = 2)	Octal (r = 8)	Hexadecimal (r = 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Example 4: Convert following hexadecimal number into decimal number: $(B65F)_{16}$

Solution: $11\times16^3+6\times16^2+5\times16^1+15\times16^0=(46.687)_{10}$

Conversion of decimal number to a number in base r:

- Separate the number into an integer part and fraction part.
- Divide the number and all successive quotients by r and accumulating the remainders.
- Conversion of decimal fraction is done by multiplying the fraction and all successive fraction and integers are accumulated.

Conversion from binary to octal is easily done by partitioning the binary number into groups of 3 digits each starting from binary point & proceeding to left and to the right.

The corresponding octal digit is then assigned to each group.

For conversion into hexadecimal, binary number is divided into group of 4 digits.

Example: $(2\ 6\ 1\ 5\ 3.7\ 4\ 6\ 0)_8$ to binary number

Solution:

010 110 001 101 011 . 111 100 110 000

Thus binary number is

$(010\ 110\ 001\ 101\ 011.111100110000)_2$

Example 5: Convert binary to hexadecimal number:

$(10\ 1100\ 0110\ 1011.1111\ 0010)_2$

0010 1100 0110 1011. 1111 0010

2 C 6 B F 2 = $(2C6B.F2)_{16}$

OPERATING SYSTEM

STUDY MATERIAL

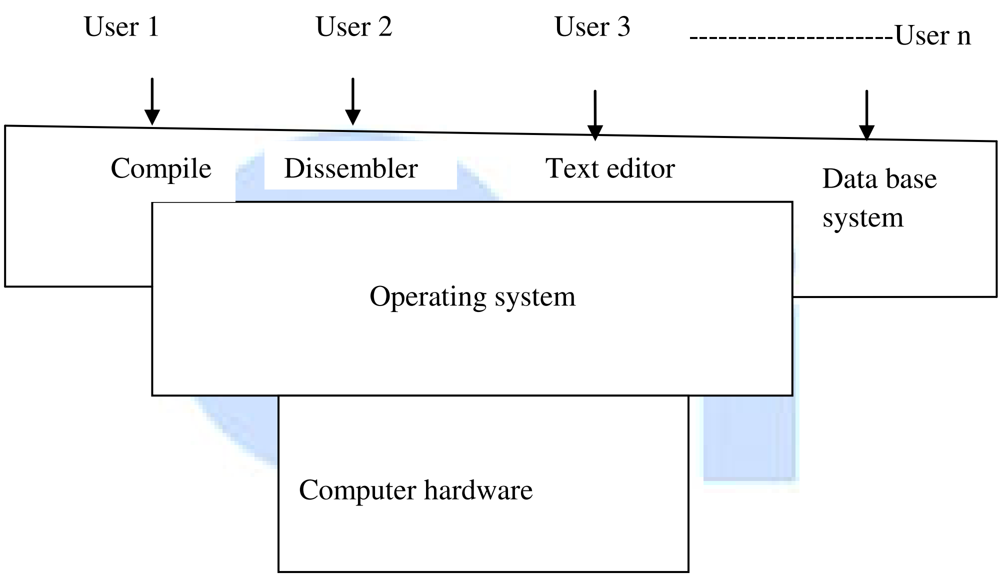
CONTENT

CHAPTER-1

INTRODUCTION

Operating system:

- An operating system is a program that acts as an intermediatory between a user of computer and computer hardware. The purpose of operating system is to provide an environment in which a user can execute program in a convenient and efficient manner.
- O.S. Implement the function which are Required to manage application program and Interact with hardware.
- O.S. is a set of function and transfer control from one job to another automatically.
- O.S. Attempt to schedule computational activity to ensure good performance of computer system. components of computer system



Abstract view of the components of a computer system

A computer system is divided into four components

1. Hardware	2. Operating system
3. Application program	4. The users

- Hardware: The central processing unit, the memory and the input/output etc, the Hardware devices and provide the basic computing resources.
- **Operating system:** control and co-ordinate for the use of the hardware among the various application program for various users.
- **Application program:** Such as Ms-Dos, database system, games and business program define the way in which these resources are used to solve computing problems of users.

Operating system can be viewed as:

- Resource Allocator: manage and allocate resources.
- Control program: A control program manages the execution of user program to prevent errors and improper use of computer. It is especially concerned with operating and control of I/O devices.

- Kernel: The are program running all the time.
  - Kernels are also set of function which implements the critical activity.

System call:

- Whenever application program need the service of O.S. then it call system call.
- Interface between application programs & O.S.
- In window system call is known as application program interface (API)

Functions of operating system:

- Process management
- Main memory management
- File management
- I/O system management
- Secondary storage management
- Networking
- Security and protection

Typical service provided by operating system

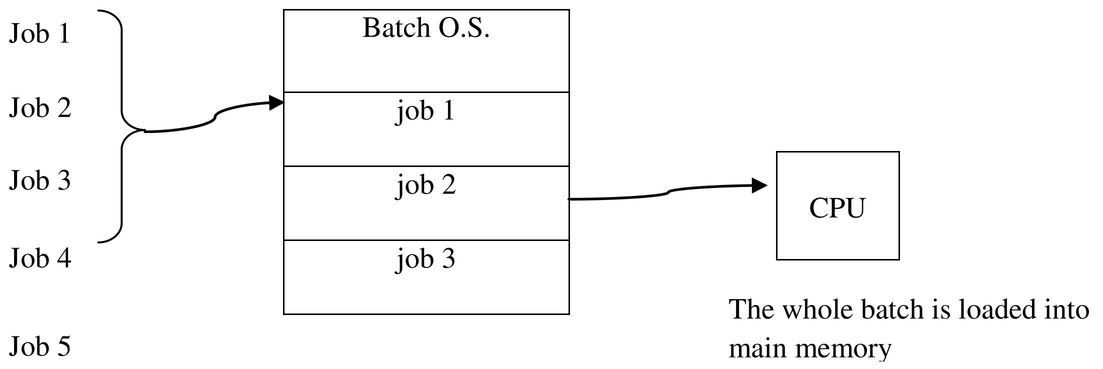
- Program creation
- Program execution
- Access to input/output devices
- Controlled access to files
- System access
- Error Detection and response
- Accounting

Types of operating system

- Batch operating systems
- Multiprogrammed systems
- Time sharing O.S.
- Multi processor systems
- Distributed systems
- Real time system

Batch O.S.:

- Same type of jobs are combined in one unit known as Batch.



- Responsibility of Batch O.S. is to change control from one job to another job.

- In batch O.S. there is lock of Interaction between user and job while job is executing.
- Batch O.S are appropriate for the executing large jobs that need little interaction.

Drawback of Batch O.S.

- Least CPU utilization.

Because let I_1 Instruction executed, I_2 is fetched and it is I/O Instruction, so it is executed only when user give input. In this case Instruction I_3 will not be fetched and CPU Remains Idle.

- No interaction between users and executing jobs.
- Appropriate for only large jobs that need little interaction.

Multi programmed O.S:

Characteristics of multi programmed O.S.

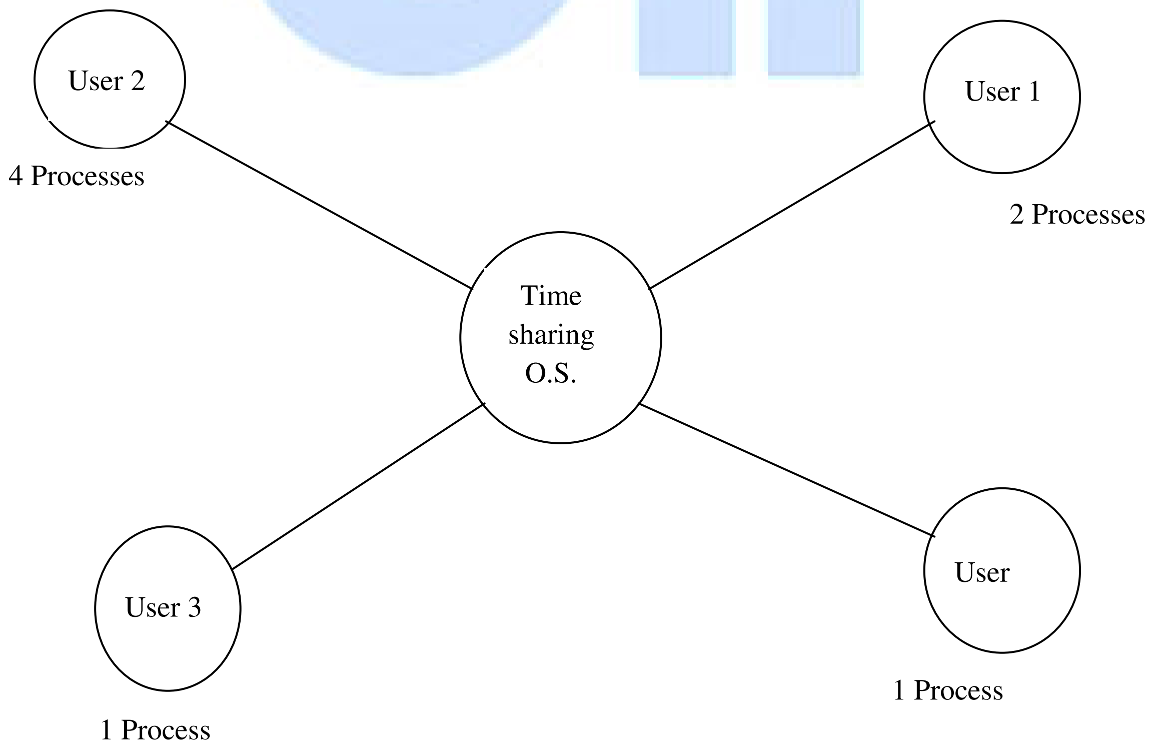
- More than one process can be loaded into main memory (you currently executed)
- If any request for I/O activity block the process than control is switched to another process which is decided by short term scheduler.
- If any process wants to execute the I/O activity then completion of I/O activity and execution of other process is done parallel.
- In a non multi programmed system the CPU would sit idle. In multi programming system the O.S. simple switches to and execute another job.

Drawback:

Multi programmed O.S. usually don't support multiuser.

Time sharing system

- Have no memory or CPU of its own



- All the process of the entire user is loaded into the main memory.

- Assign the fixed time quanta for each user- time quanta means maintain fair policy for all the users.
- In Round Robin fashion control transfer to users.
- In this time is never carry forward for example. If user 1 user 2 ms of 5 ms then system can't give 3 ms of user 2.

Multitasking:

- Multitasking is a logical extension of multi programming. Multitasking means time sharing multi programming.
- Mostly single user system can't support time sharing O.S.
- Multitasking in a single CPU is not possible.
- Multiple jobs are executed by CPU switching between them. But the switches occur frequently that the user may interact with each program while it is running.

Multiprocessor:

- Multiprocessor systems have more than one processor in close communication, sharing the computer bus the clock and sometime memory and peripheral Device. These systems are referred as tightly coupled system.
- One advantage of building this type of system is increased through put. By increasing the number of processor we hope to get more work done in a shorter period of Time.
- Multiprocessor can also save money because the processors can share peripheral, cabinet and power supply. If several program are to operate on same set of data, it is cheaper to store those data on the disk and to have all the processor share them, rather than to have many computer with local disk and many copies of data.
- Another advantage is increased reliability.

Two types of multiprocessor system:

1. Symmetric multiprocessing model:

In this model processor runs identical copy of O.S. and these copies communicate with each other when needed.

2. Asymmetric multiprocessing:

In this model each processor is assigned a specific task. A master processor control the system, the other processor either look to master for instruct or have predefined tasks. This scheme defines a master-slave relationship.

➢ Distributed system:

- Also referred as loosely coupled system i.e. do not share clock, memory instead each processor have its own local memory.
- If a number of different sites are connected to one another than a user at one site may be able to use the resources available at another.
- If one site fails in distributed system then the remaining site can potentially continue operating.
- When many sites are connected to one another by a communication network the processor at different site have the opportunity to exchange information.

Real time system:

- Used when there are vigid time requirement on the operating of processor or flow of date.
- System that control scientific experiments, medical imaging system, industrial control system and some display system are real time system.

- A real time operating system has well defined, fixed time constraints. Processing must be done within defined constraints or system will fail.

Two types of real time System:-

A hard time system:

Guarantee that critical task completed on the time. This goal requires that all delays in the system be bounded from the retrieval of stored data to the time that it takes the operating system to finish any request mode of it. Kernel delays need to be bounded.

Soft real time system:

A less restrictive type of real time system is soft real time system, when a critical real time gets priority over other tasks and retains that priority until it completes. Soft real time is an achievable goal that can be mixed with each other types of systems. However they have more limited utility than do hard real time system.

- ➢ Its frequency of execution is more as compared to other scheduler.
- ➢ It is involved each time the CPU require a new process for execution.
- ➢ Thus a short term scheduler make scheduling decision much more frequently than the long term, midterm scheduler- a scheduling decision will at a minimum have to be made after every time slice and these are short.
- ➢ This scheduler can be preemptive, implying that it is capable of forcibly removing processes from CPU when it decides to allocate that CPU to another process.

Process Termination:

- A process terminate when it finishes executing its last statement and asks the O.S. to delete it by the exit system call. At that point the process may return data to its parent process.
- All the resources of the process, including physical and virtual memory, open files and input/output buffer are deallocated by operating system.
- Also a process can cause the termination of another process via an appropriate system call. Usually such a system call can be invoked by only the parent of the process that is to be terminated.
- ➢ A parent may terminate the execution of one of its children for a variety of reason such as
  - The child has exceeded its usage of some of resources it has been allocated.
  - The task assigned to the child is no longer required.
  - The parent is exiting and operating system does not allow child to continue if its parent terminates.

PRINTED CLASSROOM NOTES

CONTENT

TOPIC

CHAPTER-1

C POINTERS

INTRODUCTION:

Pointers are frequently used in C, as they offer number of benefit to the programmer. They include

- Pointers are more efficient in handling array and data table.
- Pointer can be used to return multiple values from a function via function argument.
- Pointer permit reference to function and thereby facilitating passing of functions as argument to other functions.
- The use of pointer array to character string result in saving of data storage space in memory.
- Pointer allows C to support dynamic memory management.
- Pointer provides an efficient tool for manipulating dynamic data structure such as structure linked list queue, stacks and trees.
- Pointer reduces length and complexity of program.
- They reduce length and complexity of program.
- They increase the execution speed and thus reduce the program execution time.

Understanding pointers:

Whenever we declare a variable then system allocate somewhere in memory, an appropriate location to hold the value of variable.

Consider the following statement

$$\begin{array}{l} intx\ = 80; \\ x \leftarrow var\,iable \\ \boxed{80} \leftarrow value \\ 5000 \leftarrow address \end{array}$$

Representation of variable

- Pointer variable is nothing but a variable that contain an address which is a location of another variable in memory.

For example:

Variable	*Value*	*Address*
x	80	5000
p	5000	5048

Pointer variable

Value of variable P is the address of variable X

- Address of a variable can be determined by & operator in C

For example: $P = \& x$

Would assign 5000 to variable P

& operator can be used with simple variable or an array element. The following are the illegal use of address operator

- & 125 pointing constant
- int x [10]

& x (pointing an array name)

- & (x + y) (pointing at expression)

It x is an array then expression such as & x [0] and & x (i+3) are valid and represent the address of 0^{th} and $(i+3)^{th}$ element of

- int * P

The declaration causes the compiler to allocate memory location for the pointer variable. Since memory location has not been assigned any value. These locations may contain some unknown value in them. Therefore they point to some unknown location.

$P \quad \boxed{?} \rightarrow \; ?$

Garbage point to unknown

Location

- We can have multiple declaration in same statement

$*$ int $^*p, q, ^*r;$

$*$ int $x, ^*p, y;$

$x = 10$;

$p = \&x$

- It is also possible to combine the declaration of data variable and declaration of pointer variable and Initialization of pointer variable in one step.

int $x, ^*p = \&x$;

- We can also define the pointer variable with initial value of null or zero. We can also assign the constant value to pointer variable.

- Pointer are flexible we can make same pointer to point different data variable in different statement.

- We can also use different pointer to point the same data variable.

- int $x,\ y,\ ^*ptr$;

$ptr\ =\ \&x$;

$y =^*\ ptr$ //Assign the value pointed by pointer ptr to y

$*ptr = 25$; // This statement put the value of 25 at the location whose address is the value of pointer variable; we know the value of pointer variable (ptr) is the address of x.

Therefore old value of x is replaced by 25.

It is equivalent to assigning value 25 to x.

- **<u>Illustration of Pointer Assignments:</u>**

Stage	Value in storage	Address

Declaration	x	y	ptr	
	$\boxed{}$	$\boxed{}$	$\boxed{}$	
	4104	4108	4106	$\leftarrow$ *Address*
$x = 10$	$\boxed{10}$	$\boxed{}$	$\boxed{}$	
	4104	4108	4106	$\leftarrow$ *Address*
$ptr = \& x$	$\boxed{10}$	$\boxed{}$	$\boxed{4104}$	
	4104	4108	4106	$\leftarrow$ *Address*
$Y = * ptr$	$\boxed{10}$	$\boxed{10}$	$\boxed{4104}$	
	4104	4108	4106	$\leftarrow$ *Address*
$* ptr = 25$	$\boxed{25}$	$\boxed{10}$	$\boxed{4104}$	
	4104	4108	4106	$\leftarrow$ *Address*

➢ Chain of pointers:

$$\begin{tikzcd}
p_2 & p_1 & Variable \\
\boxed{address2} \arrow[r] & \boxed{Address1} \arrow[r] & \boxed{Value}
\end{tikzcd}$$

Pointer p_2 contain the address of pointer p_1 which points to the location that contain the desired value

For example

$\text{int } ** p_2 ;$

➢ Arithmetic operation on pointers

CONTENT

CHAPTER-1

ALPHABETS, STRINGS AND LANGUAGES

FUNDAMENTAL CONCEPTS :

Languages:

A general definition of language must cover a variety of distinct categories: natural languages, programming languages, mathematical languages, etc. The notion of natural languages like English, Hindi, etc. is familiar to us. Informally, language can be defined as a system suitable for expression of certain ideas, facts, or concepts, which includes a set of symbols and rules to manipulate these. The languages we consider for our discussion is an abstraction of natural languages. That is, our focus here is on formal languages that need precise and formal definitions. Programming languages belong to this category. We start with some basic concepts and definitions required in this regard.

→ A string is a finite sequence of symbols from Σ.

→ The length of a string S_1 denoted | S |, is the no. of symbols in it

→ The empty string is the string of length zero we usually write it like λ.

→ Σ^* denotes the set of all sequences of strings that are composed of zero or more symbols of Σ.

→ Σ^* denotes the set of all sequences of strings composed of one or more symbols of Σ. That is

$$\Sigma^+ = \Sigma^* - \{\lambda\}$$

$$\Sigma^* = \Sigma^+ + \{\lambda\}$$

$\boxed{\text{A Language is a subset of } \Sigma^*}$

*

Definition: - The collection of the strings where we can apply some restrictions / conditions in the formation of a string is called language

Ex : Σ = { 0, 1 }

$L = \{ 0^m 1^n \mid m = n\}$

$L = \{0^m 1^n \mid m > n\}$

$L = \{ 0^m\ 1^n \mid m > n\}$

Symbols:

Symbols are indivisible objects or entity that cannot be defined. That is, symbols are the atoms of the world of languages. A symbol is any single object such as ♠, *a*, 0, 1, #, begin, or do. Usually, characters from a typical keyboard are only used as symbols.

Alphabets:

- An *alphabet* is a finite, non-empty set of symbols.
  - {*0,1* } is a binary alphabet.
  - { *A*, *B*, …, *Z*, *a*, *b*, …, *z* } is an English alphabet.
- A *string* over an alphabet Σ is a sequence of any number of symbols from Σ.
  - *0*, *1*, *11*, *00*, and *01101* are strings over {*0*, *1* }.

– *Cat*, *CAT*, and *compute* are strings over the English alphabet.

Example:

$\Sigma = \{0, 1\}$

$\Sigma = \{a, b, c\}$

$\Sigma = \{a, b, c, \&, z\}$

$\Sigma = \{\#, \nabla, \spadesuit, \beta\}$

Empty String

- **An *empty string*, denoted by ε, is a string containing no symbol.**
  - **ε is a string over any alphabet.**

Strings or Words over Alphabet :

A string or word over an alphabet ∑ is a finite sequence of concatenated symbols of ∑.

Example :0110, 11, 001 are three strings over the binary alphabet { 0, 1 } .

aab,*abcb*, *b*, *cc* are four strings over the alphabet { *a*, *b*, *c* }.

It is not the case that a string over some alphabet should contain all the symbols from the alphabet. For example, the string cc over the alphabet { *a*, *b*, *c* } does not contain the symbols a and b. Hence, it is true that a string over an alphabet is also a string over any superset of that alphabet.

DATA BASE MANAGEMENT SYSTEM

STUDY MATERIAL

CONTENT

CHAPTER-1
FUNDAMENTALS OF DBMS AND ER DIAGRAMS

Database

A **database** consists of an organized collection of interrelated data.
Examples of databases: Database for Educational Institute, a Bank, Library, Railway Reservation system etc.

Database Applications

- Airlines reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Online retailers: order tracking, customized recommendations
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

Disadvantages of File system over DBMS

Most explicit and major disadvantages of file system when compared to database management system are as follows:

- **Data Redundancy:** The files are created in the file system as and when required by an enterprise over its growth path. So in that case the repetition of information about an entity cannot be avoided.

Example: The addresses of bank customers will be present in the two files

- One in savings account files and
- Other in current account, files.

Disadvantage: Wastage of storage space

- ***Data Inconsistency***: Data redundancy leads to greater problem than just wasting the storage i.e. it may lead to inconsistent data. Same data which has been repeated at several places may not match after it has been updated at some places.

Disadvantage: Mismatch of Data

Example: If a customer requests to change the address for his account in the Bank and it has been updated at the saving bank account file only but his current bank account file is not updated. Afterwards the addresses of the same customer present in saving bank account file and current bank account file will not match. Finally there is no way to find out which address is latest out of these two.

- **Difficulty in Accessing Data:** For analyzing the data the programs are not already present and only programmers will have to write a new program to generate requested report or will have to work manually. This is going to take impractical time and will be more expensive.

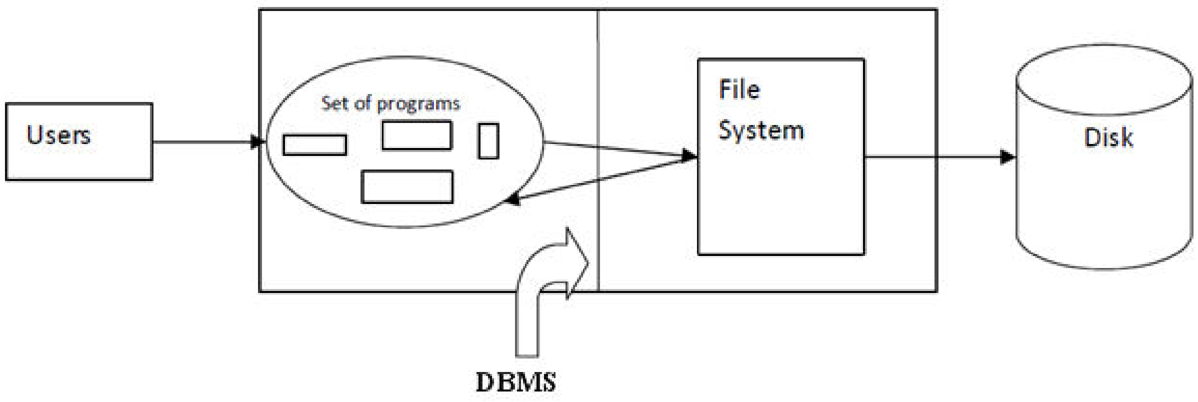
- **Data Isolation:** Since the data files are created at different times and supposedly by different people in different formats the structures of different files generally will not match. The data will be scattered in different files for a particular entity. So it will be difficult to obtain appropriate data.

Example: Suppose the Address in Saving Account file have fields:
Add line1, Add line2, City, State, Pin
While the fields in address of Current account are: ***House No., Street No., Locality*, City, State, Pin**.
Administrator is asked to provide the list of customers living in a particular locality. Providing consolidated list of all the customers will require looking in both files. But they both have different way of storing the address.
Writing a program to generate such a list will be difficult.

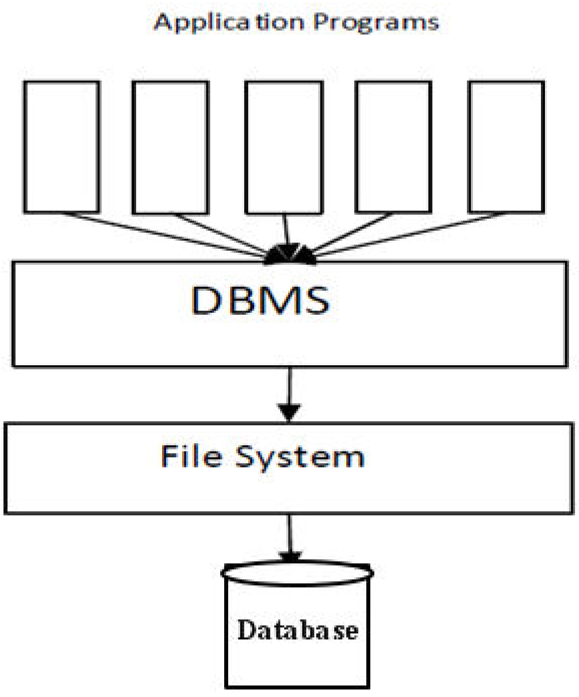
- ***Security and access control:*** Database should be protected from unauthorized users for privacy and security purpose. Every user should not be allowed to access every data. For example:
  1. The Payroll Personnel in a bank should not be allowed to access accounts information of the customers
  2. An employee can't see salary of other employees

Role of DBMS:

The earlier Information system will work as follows:



While the DBMS will be another layer of software package placed between the file system and set of application programs. The Role of DBMS can describe by the following diagram at a very high level.



Modeling Basics

Static Information:

Data -- Entities

Associations -- Relationships among entities

Dynamic Information:

Processes -- Operations/transactions

Integrity constraints -- Business rules/regulations and data meanings

Data Model

A data model describes:

- The manner in which data can be stored, organized and manipulated
- Data relationships
- Data semantics
- Data constraints
- **A data model is a collection of concepts that can be used to describe the structure of a database.**

Data models are of following types:

- ***Conceptual Data Model***
  - high level description
  - useful for requirements understanding.
- ***Representational Data Model***
  - describing the logical representation of data without giving details of physical representation.
- ***Physical Data Model***
  - ➢ description giving details about record formats, file structures etc.

Types of Databases

There are several common types of databases; each type of database has its own data model (how the data is structured). They include; Flat Model, Hierarchical Model, Relational Model and Network Model.

The Flat Model Database

In a flat model database, there is a two dimensional (flat structure) array of data. For instance, there is one column of information and within this column it is assumed that each data item will be related to the other. For instance, a flat model database includes only zip codes. Within the database, there will only be one column and each new row within that one column will be a new zip code.

The Hierarchical Model Database

The hierarchical model database resembles a tree like structure, such as how Microsoft Windows organizes folders and files. In a hierarchical model database, each upward link is nested in order to keep data organized in a particular order on a same level list. For instance, a hierarchal database of sales, may list each days sales as a separate file. Within this nested file are all of the sales (same types of data) for the day.

The Network Model

In a network model, the defining feature is that a record is stored with a link to other records – in effect networked. These networks (or sometimes referred to as pointers) can be a variety of different types of information such as node numbers or even a disk address.

The Relational model

It is a lower level data model as compared to E-R model

- SQL uses relational model to ease user's interaction with the system

Definitions:

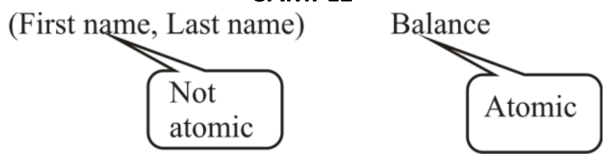
- A **row / tuple** in a table represents a relationship among a set of values
- A **table/relation** is an entity set (Set of tuples)
- **Attributes**
  - Each attribute has a name
  - For each attribute then is a permitted values called the domain of the attribute
  - Attributes are atomic (indivisible) in a nature.

Example:

(1) Balance>500

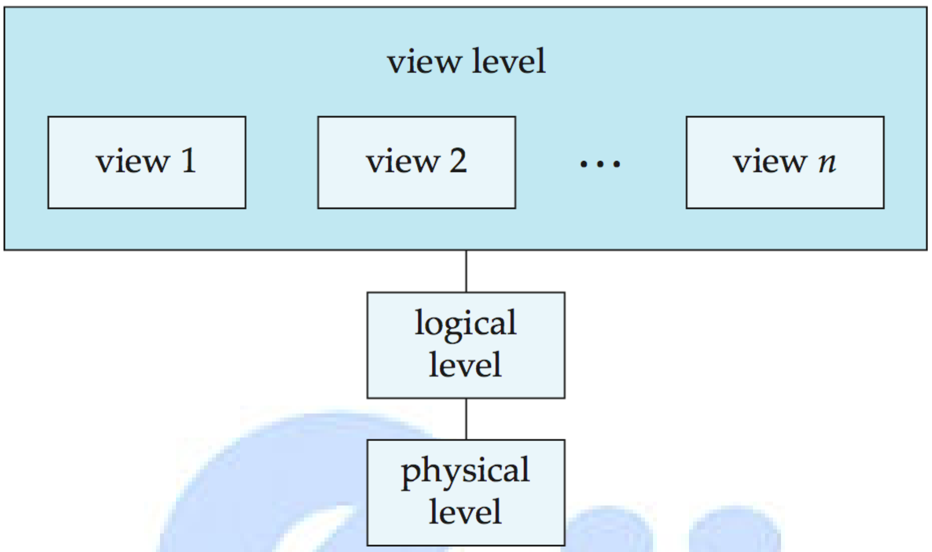
(2) Name must start with an alphabet

Example:



Note to memorize: Null is a member of every domain.

Database system Architecture



View

A view is a virtual table in a relational database that provides easy access to the database.

The Database administrator can restrict what to show to users

Advantage of views over tables:

- Views can represent a subset of the data contained in a table.
- Views can join & simplify multiple tables into a single virtual table.
- Views take very little space for storage.

Note: the database contains only the definition of a view (query) not a copy of all the data it present.

- Views can act as aggregated tables.

Example: A view can have aggregated data (sum average etc of some attributes) of the relation

Views can limit the degree of exposure of a table or tables to the outer world