Computer Organization & Architecture

BRAJ KISHOR PRASAD
Asstt. Professor, Deptt. of MCA
Patna Women's College

Computer Organization & Architecture

1.1.1 INTRODUCTION

Computer Architecture deals with instructions, addressing mode, ALU, pipelining etc. i.e. internal design of the architecture. In other words, the view of a computer as presented to software designers.

Computer Organization deals with how various memory and Input/Output interacts with a system. In other words, the actual implementation of a computer in hardware is Computer Organization.

Computer Design deals with hardware design.

Computer Architecture and Organization consists of the following:

- **Number System**
- **Data Representation**
- **Basic Components:** Gate, Adder, Sub-tractor, Adder-Subtractor, Flip-Flop etc.
- **Memory Interfacing**
- **I/O Interfacing**
- **Control Unit Design**
- **ALU Data Path**
- **Pipelining**

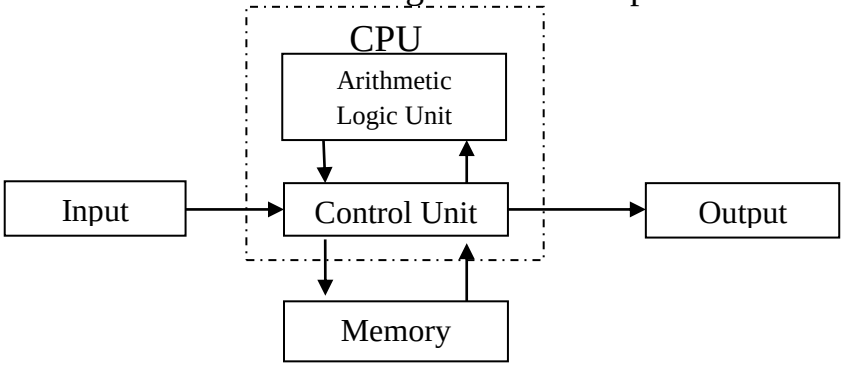
1.1.2 DEFINITION

Computer: A computer is a general purpose device that can be programmed to process information and yield meaningful results. It consists of CPU (ALU, Control Unit and Registers), Memory (Primary Memory and Secondary Memory) and I/O devices.

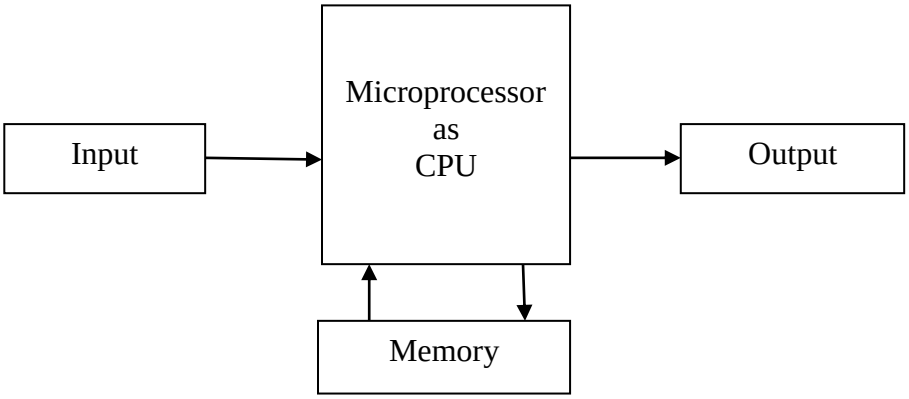
Processor: It is a chip that consists of ALU, Control Unit and Registers.

Microcontroller: It consists of Microprocessor as CPU, Memory, I/O devices and Peripheral devices.

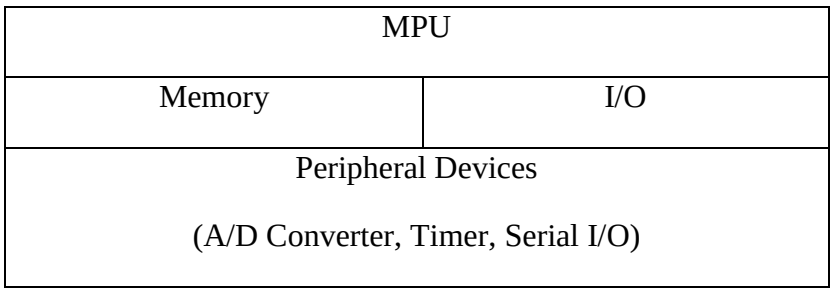
Traditional Block Diagram of a Computer



Block Diagram of a computer with the Microprocessor as CPU



Block diagram of a Microcontroller



1.1.3 OBJECTIVES

The objective of this subject is to minimize space and time complexity. So that, real life devices can be invented and developed based on either microprocessor or microcontroller.

1.1.4 FUTURE SCOPE (APPLICATIONS AND REQUIREMENTS)

- **Scientific/Numerical**: weather prediction, molecular modelling etc.
  - **Need**: large memory, floating-point arithmetic
- **Commercial**: inventory, payroll, web serving, e-commerce etc.
  - **Need**: integer arithmetic, high I/O
- **Embedded**: automobile engines, microwave, PDAs, Mobile etc.
  - **Need**: low power, low cost, interrupt driven
- **Home Computing**: multimedia, games, entertainment etc.
  - **Need**: high data bandwidth, graphics

1.1.5 NUMBER SYSTEM

Number System:- A Number System is the representation of distinct symbols that are used in that number system. This number of distinct symbols is known as base or radix(r) of that number system. In each number system, there is a minimum digit and maximum digit. Thus, there will be infinite number systems. A Number System also defines how a number can be represented using distinct symbols. They are broadly divided into two categories: (1) Non-Positional Number System and (2) Positional Number System.

1.1.5 NON-POSITIONAL NUMBER & POSITIONAL NUMBER SYSTEM

Non-Positional Number System:- A non-positional number system uses a limited number of symbols in which each symbol has a value. The position a symbol occupies in the number normally bears no relation to its value. The value of each symbol is fixed. To find the value of a number, we add the value of all symbols present in the representation.

Some Positional and Non-Positional Number System :

Positional Number System	Non-Positional Number System
1	I
2	II
3	III
4	IV
5	V
6	VI
7	VII
8	VIII
9	IX
10	X
50	L
100	C
500	D
1000	M

Positional Number System:- In a positional number system, the position a symbol occupies in the number determines the value it represents. It is divided

into two categories: (1) Standard number System and (2) Non-Standard Number System.

Standard Number System is unique and globally acceptable while Non-Standard Number System is not unique and individually acceptable.

Standard Number Systems that are used in computer

#	Number System	Radix or Base(r)	Min. Digit	Max. Digit
1	Decimal Number System (0,1, 2, 3, 4, 5, 6, 7, 8, 9)	10	0	9
2	Binary Number System (0, 1)	2	0	1
3	Octal Number System (0, 1, 2, 3, 4, 5, 6, 7)	8	0	7
4	Hexadecimal Number System (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)	16	0	F

1.1.7 REPRESENTATION OF A NUMBER

Representation of a Number: A number is represented by a string of digit symbols.

1.1.8 REPRESENTATION OF A NUMBER SYSTEM

Representaion of a Number System: A number system is represented by making radix of that number system as suffix of enclosing the string of digit symbols within small parenthesis. Ex.- $(2905)_{10}$, $(1011)_2$, $(6702)_8$, $(45A8)_{16}$ etc.

To determine the quantity that number represents, we multiply the each digit with its face value and place value and then sum them.

Example:-

(a) Decimal Number System $(2425.5)_{10}$

Position		**3**	**2**	**1**	**0**	**-1**
Face value		**2**	**4**	**2**	**5 . 5**	
Positional value		$\mathbf{10^3}$	$\mathbf{10^2}$	$\mathbf{10^1}$	$\mathbf{10^0}$	$\mathbf{10^{-1}}$
Actual value	=	$\mathbf{2 \times 10^3 + 4 \times 10^2 + 2 \times 10^1 + 5 \times 10^0 + 5 \times 10^{-1}}$				
	=	**2 x 1000 + 4 x 100 + 2 x 10 + 5 x 1 + 5 x 1/10**				
	=	**2000 + 400 + 20 + 5 + 0.5**				
	=	$\mathbf{(2425.5)_{10}}$				

(b) Binary Number System $(1011)_2$

Position		**3**	**2**	**1**	**0**
Face value		**1**	**0**	**1**	**1**
Positional value		2^3	2^2	2^1	2^0
Actual value	=	1×2^3 +	0×2^2 +	1×2^1 +	1×2^0
	=	1×8 +	0×4 +	1×2 +	1×1
	=	$8 + 0 + 2 + 1 = (11)_{10}$			

(c) Octal Number System $(7261)_8$

Positional value		**3**	**2**	**1**	**0**
Face value		**7**	**2**	**6**	**1**
Positional value		8^3	8^2	8^1	8^0
Actual value	=	7×8^3 +	2×8^2 +	6×8^1 +	1×8^0
	=	7×512 +	2×64 +	6×8 +	1×1
	=	$3584 + 128 + 48 + 1$			
	=	$(3761)_{10}$			

(d) Hexadecimal Number System $(2AF5)_{16}$

Position		**3**	**2**	**1**	**0**
Face value		**2**	**A**	**F**	**5**
Positional value		16^3	16^2	16^1	16^0
Actual value	=	2×16^3 +	$A \times 16^2$ +	$F \times 16^1$ +	5×16^0
	=	2×4096 +	10×256 +	15×16 +	5×1
	=	$8192 + 2560 + 240 + 5$			
	=	$(10997)_{10}$			

1.1.9 CONVERSION

Conversion converts the one number system into another number system. In it the representation of a number changed but the value remains fixed.

Conversion of one number system into another number system

$(35.6875)_{10}$ into ($)_2$
i.e. Decimal number into Binary number

Integer = 35 Quotient:35, Divisor(radix)=2		Fraction = 0.6875 Multiplier(radix) = 2		
radix) Quotient	Remainder	Integer	Fraction x Radix	Product
2) 35	1		.6875 x 2	1.3750
2) 17	1	1	.3750 x 2	0.7500
2) 8	0	0	.7500 x 2	1.5000
2) 4	0	1	.5000 x 2	1.0000
2) 2	0	1	.0000	
2) 1	1			
0				
To get binary value of integer part, write all remainders from bottom to top		To get binary value of fractional part, write all integers from top to bottom		
$(35)_{10} = (100011\ )_2$		$(0.6875)_{10} = (1011)_2$		
$(35.6875)_{10} = (100011.1011)_2$				

$(100011.1011\ )_2$ into ($)_{10}$ into
i.e. Decimal number into Binary number

$(100011.1011)_2 = 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$

$= 1 \times 32 + 0 \times 16 + 0 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 + 1 \times ½ + 0 \times ¼ + 1 \times 1/8 + 1 \times 1/16$

$= 32 + 0 + 0 + 0 + 2 + 1 + 1 \times 0.5 + 0 \times 0.25 + 1 \times 0.125 + 1 \times 0.0625$

$= 35 + 0.5 + 0 + 0.125 + 0.0625 = (35.6875)_{10}$

Table : NS01

$(35.6875)_{10}$ into ($)_8$
i.e. Decimal number into Octal number

Integer = 35 Quotient:35, Divisor(radix)=8		Fraction = 0.6875 Multiplier(radix) = 8		
radix) Quotient	Remainder	Integer	Fraction x radix	Product
8) 35	3		.6875 x 8	5.5000
8) 4	4	5	.5000 x 8	4.0000
0		4	.0000	
To get octal value of integer part, write all remainders from bottom to top		To get octal value of fractional part, write all integers from top to bottom		
$(35)_{10} = (43)_8$		$(0.6875)_{10} = (54)_8$		
$(35.6875)_{10} = (43.54)_8$				

$(43.54)_8$ into ($)_{10}$
i.e. Octal number into Decimal number

$(43,54)_8 = 4 \times 8^1 + 3 \times 8^0 + 5 \times 8^{-1} + 4 \times 8^{-2} = 4 \times 8 + 3 \times 1 + 5 \times 1/8 + 4 \times 1/64$

$= 32 + 3 + 5 \times 0.125 + 4 \times 0.015625 = 35 + 0.625 + 0.0625$

$= (35.6875)_{10}$

Table: NS02

$(35.6875)_{10}$ into ($)_{16}$
i.e. Decimal number into Hexadecimal number

Integer = 35 Quotient:35, Divisor(radix)=16		Fraction = 0.6875 Multiplier(radix) = 16		
radix) Quotient	Remainder	Intege	Fraction x radix	Product
16) 35	3		.6875 x 16	11.0000
16) 2	2	11=B	.0000	
0				
To get hexadecimal value of integer part, write all remainders from bottom to top		To get hexadecimal value of fractional part, write all integers from top to bottom		
$(35)_{10} = (23)_{16}$		$(0.6875)_{10} = (B)_{16}$		
$(35.6875)_{10} = (23.B)_{16}$				

$(23.B)_{16}$ into ($)_{10}$
i.e. Hexaecimal number into Decimal number

$(23.B)_{16} = 2 \times 16^1 + 3 \times 16^0 + B \times 16^{-1} = 2 \times 16 + 3 \times 1 + 11 \times 1/16$

$= 32 + 3 + 11 \times 0.0625 = 35 + 0.6875 = (35.6875)_{10}$

Table:NS03

$(23.B)_{16}$ into ($)_8$ **i.e. Hexadecimal number into Octal number**
Process: 1. Convert Hexadecimal number into Decimal number 2. Convert Decimal number into Octal number
1. By using Table:NS03, $(23.B)_{16} = (35.6875)_{10}$
2. By using Table:NS02, $(35.6875)_{10} = (43.54)_8$
$(23.B)_{16} = (43.54)_8$
$(43.54)_8$ into ($)_{16}$ **i.e. Octal number into Hexadecimal number**
Process: 1. Convert Octal number into Decimal number 2. Convert Decimal number into Hexadecimal number
1. By using Table:NS02, $(43.54)_8 = (35.6875)_{10}$
2. By using Table:NS03, $(35.6875)_{10} = (23.B)_{16}$
$(43.54)_8 = (23.B)_{16}$
Table:NS04 (Complex Process)

Conversion from Octal number into Binary-Coded Octal: Use 3-bits group to represent each Octal number as following table:

Octal Number	Binary-Coded Octal	Decimal Equivalent
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7
Table:NS05		

Conversion from Hexadecimal number into Binary-Coded Hexadecimal:
Use 4-bits group to represent each Hexadecimal number as following table:

Hexadecimal Number	Binary-Coded Hexadecimal	Decimal Equivalent
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15
Table:NS06		

$(23.B)_{16}$ into ($)_8$ **i.e. Hexadecimal number into Octal number**
Process: 1. Convert each digit of Hexadecimal number into its Binary-coded Hexadecimal using Table:NS06 2. Group Binary-Coded Hexadecimal for Binary-Coded Octal by using 3-bits group from left in integer part and right in fractional part add extra 0 in left and right (if required) 3. Convert 3-bits group into Octal number
1. $(23.B)_{16}$ = (0010 0011.1011)
2. (0010 0011.1011) = (000 100 011 . 101 100)
3. (000 100 011 . 101 100) = $(043.54)_8$
$(23.B)_{16} = (43.54)_8$
(43.54)$_8$ into ($)_{16}$ **i.e. Octal number into Hexadecimal number**
Process: 1. Convert each digit of Octal number into its Binary-Coded Octal using Table:NS05 2. Group Binary-coded Octal for Binary-coded Hexadecimal by using 4-bits group from left in integer part and right in fractional part, also add extra 0 in left and right (if required) 3. Convert 4-bits group into Hexadecimal number
1. $(43.54)_8$ = (100 011.101 100)
2. (100 011.101 100) = (0010 0011 . 1011 0000)
3. (0010 0011 . 1011 0000) = $(23.B)_{16}$
$(43.54)_8 = (23.B)_{16}$
Table:NS07 (Easy Process)

1.1.10 ARITHMETIC OPERATIONS ON BINARY NUMBERS

(a) Addition

0	0	1	1
+0	+1	+0	+1
0	1	1	10

(b) Subtraction

0	1 borrow 0	1	1
- 0	- 1	- 0	- 1
0	1	- 1	0

(c) Multiplication

0	0	1	1
x 0	x 1	x 0	x 1
0	0	0	1

(d) Division

$0/0 = 0$ $0/1 = 0$ $1/0 = \alpha$ $1/1 = 1$

Examples:

Addition

(a)
```
  11222210  ← carry
   1011011
   1001110
     10001
       110
+     1010
  11001010
```

(b)
```
  111110  ← carry
   11011
+ 01111
  101010
```

Subtraction

(a)
```
   02   ← Borrow
  110111
- 1010
  101101
```

(b)
```
    02  }
     02 } ← Borrow
   1101011
- 111011
  0110000
```

Multiplication

(a)
```
    10111
  x   101
  10111
 00000
10111
1110011
```

(b)
```
    110
  x 10
  000
 110
 1100
```

Division

```
                Dividend
Divisor 1011 ) 10010  ( 10  Quotient
               1011
               0110
               0000
                110  Remainder
```

1.1.11 COMPLEMENT

Complement: It is used for simplifying the subtraction operation and for logical operation in digital computer. There are two types of complements for each number system whose base is r: the (r – 1)'s complement and the r's complement.

(r-1)'s complement: The (r-1)'s complement of a number (N) in base r is **$(r^n – 1)$ – N** where n is number of digits in that number (N). Ex.- $(1234)_{10}$ Here, N=1234, n=4 and r=10 therefore (r – 1)'s i.e. 9's complement of 1234 is $(10^4 – 1)$ -1234 = (10000 – 1) – 1234 = 9999 – 1234 = 8765.

Another way to find (r – 1)'s complement: Subtract each digit form the maximum digit of that number system.

Ex.- 9's complement of $(1234)_{10}$ = (9-1)(9-2)(9-3)(9-4) = 8765

r's complement: The r's complement of a number (N) in base r is r^n - N where N is not equal to 0 and 0 for N = 0 and n is number of digits in that number(N). Ex.- $(1234)_{10}$ Here. N=1234, n=4 and r=10 therefore r's i.e 10's complement of 1234 is equal to 10^4 – 1234 10000 – 1234 = 8766.

Another way to find r's complement: (r – 1)'s complement + 1.

Ex.- 10's complement of $(1234)_{10}$ = 9's complement of 1234 + 1

= 8765 + 1 = 8766

Table: (r-1)'s and r's complement of $(25)_{10} = (11001)_2 = (31)_8 = (19)_{16}$ in different number system (complex process)					
Number System	**r**	**N**	**n**	**(r-1)'s complement i.e. $(r^n - 1) - N$**	**r's complement i.e. $(r^n - N)$**
Decimal Number System	10	25	2	$= (10^2 - 1) - 25$ $= (100 - 1) - 25$ $= 99 - 25 = 74$	$= 10^2 - 25$ $= 100 - 25$ $= 75$
Binary Number System	2	11001	5	$= (2^5 - 1)_{10} - 11001$ $= (32 - 1)_{10} - 11001$ $= (31)_{10} - 11001$ $= 11111 - 11001$ $= 00110$	$= 2^5 - 1101$ $= (32)_{10} - 11001$ $= 100000 - 11001$ $= 00111$
Octal Number System	8	31	2	$= (8^2 - 1)_{10} - 31$ $= (64 - 1)_{10} - 31$ $= (63)_{10} - 31$ $= 77 - 31$ $= 46$	$= (8^2)_{10} - 31$ $= (64)_{10} - 31$ $= 100 - 31$ $= 47$
Decimal Number System	16	19	2	$= (16^2 - 1)_{10} - 19$ $= (256 - 1)_{10} - 19$ $= (255)_{10} - 19 = FF - 19$ $= E6$	$= (16^2)_{10} - 19$ $= (256)_{10} - 19$ $= 100 - 19$ $= E7$

Table: (r-1)'s and r's complement of $(25)_{10} = (11001)_2 = (31)_8 = (19)_{16}$ in different number system (easy process)					
Number System	**r**	**N**	**Max. digit**	**(r-1)'s complement i.e. Subtract each digit form Max. digit**	**r's complement i.e. (r – 1)'s complement + 1**
Decimal Number System	10	25	9	$= (9-2)(9-5)$ $= 74$	$= 74 + 1$ $= 75$
Binary Number System	2	11001	1	$= (1^-1)(1-1)(1-0)(1-0)(1-1)$ $= 00110$	$= 00110 + 1$ $= 00111$
Octal Number System	8	31	7	$= (7-3)(7-1)$ $= 46$	$= 46 + 1$ $= 47$
Decimal Number System	16	19	15 or F	$= (15 - 1)(15 - 9)$ $= E6$	$= E6 + 1$ $= E7$

1.1.11 NUMBER REPRESENTATION

There are three ways to represent a number:

(a) **Signed Magnitude Method in Binary number system** – In Signed Magnitude Method, number is divided into two parts: 1st part - Left most bit (LSB) is used to represent sign and 2nd part – rest bits are used for magnitude of that number. 0 is used for positive number and 1 is used for negative number. Example-

(1) $(+5)_{10} = (0\ 101)_2$ (2) $(-5)_{10} = (1\ 101)_2$

(b) **(r-1)'s or 1's Complement Method in Binary number system** – In 1's complement, reverse each bit or subtract each bit from max. digit of that number system except sign bit. Example-

	Decimal Number	Signed Magnitude Method	1's Complement Method
(1)	+5	0 101	0 010
(2)	-5	1 101	1 010

(c) **r's or 2's Complement Method in Binary number system** – In 2's complemt, add 1 in 1's complement of that number or (r-1)'s complement + 1. Example-

	Decimal Number	Signed Magnitude Method	1's Complement Method	2's Complement Method
(1)	+5	0 101	0 101	0 101
(2)	-5	1 101	1 010	1 011

Some More Examples:

DN	S.M.M.	1's comp.	2's comp.
+5	0101	0101	0101
-3	1011	1100	1101
-4	1100	1011	1100

Arithmetic Operations using Decimal method, Signed Magnitude method, 1's complement method and 2's complement

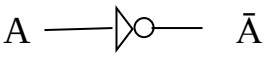
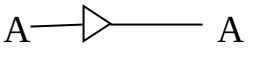
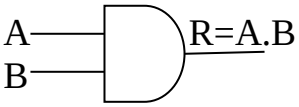
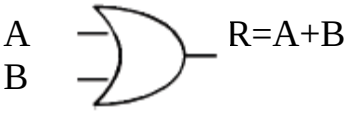
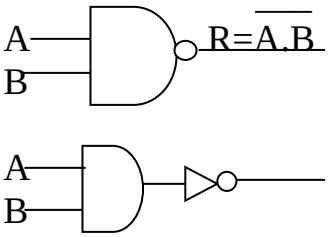
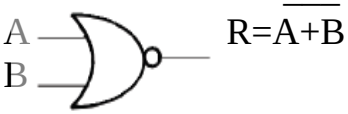
Decimal Method	Sign Magnitude Method	1's complement	2's complement
+5	0 0101	0 0101	0 0101
+3	0 0011	0 0011	0 0011
+8	0 1000	0 1000	0 1000
+5	0 0101	0 0101	0 0101
-3	1 0011	1 1100	1 1101
+2	0 0010	10 0001 + 1	10 0010 Discard
		0 0010	0 0010
-5	1 0101	1 1010	1 1011
+3	0 0011	0 0011	0 0011
-2	1 0010	1 1101	1 1110
		1 0010	1 0010
-5	1 0101	1 1010	1 1011
-3	1 0011	1 1100	1 1101
-8	1 1000	11 0110 + 1	11 1000 Discard
		1 0111	1 1000
		1 1000	

1.2.1 BINARY VARIABLE

Binary Variable: A variable that has either 0 or 1 value only is called Binary Variable. **Ex**.: A = 1; A' = 0, B = 0; B' = 1 etc.

1.2.2 GATE

Gate: It is a logical circuit of boolean variable(s) that shows the relation between input boolean variable(s) and output boolean variable(s) by using Boolean Algebra and in tabular form by Truth Table. There are a number of gates. They are NOT, Buffer, AND, OR, NAND, NOR and XOR.

<table>
<tr><th>#</th><th>Gate</th><th>Expression</th><th>Symbol</th><th>Truth Table</th></tr>
<tr><td>1</td><td>NOT</td><td>$\bar{A}$</td><td> 
</td><td>
<table>
<tr><th>A</th><th>A'</th></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>Buffer</td><td>A</td><td> 
</td><td>
<table>
<tr><th>A</th><th>Buffer A</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>AND</td><td>A.B</td><td> 
</td><td>
<table>
<tr><th>A</th><th>B</th><th>R=A.B</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>OR (Inclusive OR)</td><td>A+B</td><td> 
</td><td>
<table>
<tr><th>A</th><th>B</th><th>R=A+B</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>5</td><td>NAND</td><td>$\overline{A.B}$</td><td> 
</td><td>
<table>
<tr><th>A</th><th>B</th><th>R</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>6</td><td>NOR</td><td>$\overline{A+B}$</td><td> 
</td><td>
<table>
<tr><th>A</th><th>B</th><th>R</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>7</td><td>XOR (Exclusive OR)</td><td>A ⊕ B</td><td> 
</td><td>
<table>
<tr><th>A</th><th>B</th><th>R</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
</table>

1.2.3 BOOLEAN ALGEBRA

1) 0 + A = A	**2) 1 + A = 1**	**3) A + A = A**
4) A + A' = 1	**5) 0.A = 0**	**6) 1.A = A**
7) A.A = A	**8) A.A' = 0**	**9) A' = 0**
10) A + B = B + A	**11) A.B = B.A**	**12) A + (B + C) = (A + B) + C**
13) A.(B.C) = (A.B).C	**14) A + AB = A**	**15) A.(B + C) = A.B + A.C**
16) A(A + B) = A	**17) (A')' = A**	**18) A + A'B = A + B**
19) (A + B)' = A'B'	**20) (AB)'=A' + B'**	**21) (A+B).(A+C)=A+BC**

Simplify the following expressions using Boolean algebra:

Ex. Q = X'Y'Z' + X'YZ' + XY'Z' + XYZ'
= X'(Y'Z' + YZ') + X(Y'Z' + YZ') = (Y'Z' + YZ')(X' + X)
= (Y'Z' + YZ') . 1 = Y'Z' + YZ' = (Y' + Y)Z' = 1.Z' = Z'

Ex. C = A'B + AB' + AB
= A'B + A(B' + B) = A'B + A = A + A'B = A + B

Ex. Q = (X'YZ') + (X'YZ) + (XYZ')
= X'Y(Z' + Z) + XYZ' = X'Y.1 + XYZ' = X'Y + XYZ'
= Y(X' + XZ') = Y(X' + Z') = X'Y + YZ'

Ex. D = A'B'C' + A'BC' + A'BC + ABC'
= A'B'C' + A'BC + A'BC' + ABC' = A'C(B' + B) + BC'(A' + A)
= A'C.1 + BC'.1 = A'C + BC'

Ex. A + AB = A(1 + B) =A.1 = A

Ex. AB + AB' =A(B+B') = A.1 = A

Ex. Show that (A + B)'(A' + B')' = 0
L.H.S. = (A + B)'(A' + B')'
=(A'B')(AB) = 0 = R.H.S.

Ex. Show that A + A'B + A'B' = 1
L.H.S. = A + A'B + A'B'
= A + A'(B + B') = A + A' = 1 = R.H.S.

Ex. Given the Boolean Expression F = X'Y + XYZ'. Derive an algebraic expression for the F'.

Here, F = X'Y + XYZ'

Therefore F' = (X'Y + XYZ')' = (X'Y)'.(XYZ')'

= ((X')' + Y') . (X' + Y' + (Z')') =(X + Y').(X' + Y' + Z)

= XX' + XY' + XZ + X'Y' + Y'Y' + Y'Z

= 0 + XY' + XZ + X'Y' + Y' + Y'Z

= Y' + XY' + X'Y' + Y'Z + XZ

= Y'(1 + X + X' + Z) + XZ = Y'(1 + X' + Z) + XZ

= Y'(1 + Z) + XZ = Y' + XZ

1.2.4 KARNAUGH MAP (K. MAP)

K.Map is used to minimize the expression i.e. reducing the number of gates in the circuit. It is efficiently applicable to two variables, three variables and four variables. It is based on Hamming Distance or Gray Code.

Gray Code

Gray Code changes by only one bit as it sequences from one number to the next.

Gray Code Counter

A Gray code counter is a counter whose flip-flops go through a sequence of states as Gray code.

Table: The reflected binary or Gray Code of

Two bit		Three bit		Four bit	
BN	**Gray Code**	**BN**	**Gray Code**	**BN**	**Gray Code**
00	00	000	000	0000	0000
01	01	001	001	0001	0001
10	11	010	011	0010	0011
11	10	011	010	0011	0010
		100	110	0100	0110
		101	111	0101	0111
		110	101	0110	0101
		111	100	0111	0100
				1000	1100
				1001	1101
				1010	1111
				1011	1110
				1100	1010
				1101	1011
				1110	1001
				1111	1000

Other Decimal Codes

Binary codes for decimal digits require a minimum four bits. Many different codes are used by arranging four or more bits in 10 distinct combinations.

Table:Different Codes for the					
Decimal digit	BCD 8421	2421	Excess-3 (BCD+0011)	Excess-3 gray	Gray Code
0	0000	0000	0011	0010	0000
1	0001	0001	0100	0110	0001
2	0010	0010	0101	0111	0011
3	0011	0011	0110	0101	0010
4	0100	0100	0111	0100	0110
5	0101	1011	1000	1100	0111
6	0110	1100	1001	1101	0101
7	0111	1101	1010	1111	0100
8	1000	1110	1011	1110	1100
9	1001	1111	1100	1010	1101
Unused bit combinations	1010	0101	0000	0000	1111
	1011	0110	0001	0001	1110
	1100	0111	0010	0011	1010
	1101	1000	1101	1000	1011
	1110	1001	1110	1001	1001
	1111	1010	1111	1011	1000