Unit-1

Binary Systems

Digital System

- It is a system that manipulates (processes) discrete elements of information.

- A digital system is a discrete information processing system.

-For e.g. calculator,digital voltmeters, general purpose computer etc.

- In such systems, all quantities are represented using two values i.e.O and 1.Analog System

-It is a system that manipulate physical quantities that are represented in analog form i.e continuous range of values.

-For e.g. voltmeter, speedometer, analog thermometer etc.

Block Diagram of Digital Computer

A digital computer is a programmable machine which read the binary instruction and processes the data which are presented in binary form.

The digital computer takes the binary data at input, processes according to the set of instructions called program and produces the digital output.

Working principles of digital computer:

1.Memory unit stores programs as well as input, output and intermediate data.

2.The control unit supervises the flow of information between various units and retrieve the instructions stored in memory unit.

3. After getting control signal memory unit sends the data to the processor.

4.For each instruction control unit informs the processor to execute the operation according to the instruction.

5.After getting control signal processor sends the process information to memory unit and memory unit sends those information to the output unit.

Advantages of digital system

- In case of digital system large number of ICs are available for performing various

operations hence digital systems are highly reliable, accurate, small in size and speed of operation is very high.

-Computer controls digital system can be controlled by software that allows new function to be added without changing hardware.

-Less expensive

-Easy to manipulate

Disadvantages of digital system

- It is difficult to install digital system because it required many more complex electronic circuits and ICs.

- In digital systems, if a single piece of data lost, large blocks of related data can completely change.

Number System

In general, in any number system there is an ordered set of symbols known as digits with rules defined for performing arithmetic operations like addition, multiplication etc. A collection of these digits makes a number in general has two parts- integer and fractional. Set apart by a radix point(.).

· There are mainly four number system which are used in digital electronics platform.

1.Decimal number system:

-The decimal number system contains ten unique digits from 0 to 9.

- The base or radix is 10.

2.Binary number system:

- The binary number system contains two unique digits 0 and 1.

-The base or radix is 2.

3. Octal number system:

-The octal number system contains eight unique digits from 0 to 7.

-The base or radix is 8.

4.Hexadecimal number system:

-The hexadecimal number system contains sixteen unique digits: O to 9 and six letters A,

B,C,D,E and F.

-The base or radix is 16.

1.Convert decimal fraction to binary number

Most Significant Bit (MSB)			Decimal Point			Least Significant Bit (LSB)
102	101	10°		10^{-1}	10^{-2}	10^{-3}
100	10	1		0.1	0.01	0.001

Let's take an example forn=4.47,k=3

Step 1: Conversion of 4 to binary

1.4/2 : Remainder = 0 : Quotientt

2. 2/2 : Remainder = θ : Quotien t=1

3.1/2 : Remainder =1 : Quotient = 0

So equivalent binary of integral part of decimal is *100.*

Step 2: Conversion of .47 to binary

1.0.47* 2=0.94,Integral part: 0

2.0.94*2=1.88,Integral part: 1

3.0.88*2=1.76,Integral part: 1

So equivalent binary of fractional part of decimal is.011

Step 3: Combined the result of step 1 and 2.

Final answer can be written as:

100+.011=100.011

Example-2: The number 2020.50 is interpreted as:-

=(2020.50)1e

$$= (1024 + 512 + 256 + 128 + 64 + 32 + 4 + (1/2))_{10}$$

$$= (2^{12} \times 1 + 2^{2} \times 1 + 2^{3} \times 1 + 2^{3} \times 1 + 2^{5} \times 1 + 2^{5} \times 1 + 2^{2} \times 0 + 2^{3} \times 0 + 2^{2} \times 1 + 2^{1} \times 0 + 2^{1} \times 1)_{10}$$

$=(111111100100.1)_2$

2.Convert binary fraction to decimal number

Let's take an example for n = 110.101

Step 1: Conversion of 110 to **decimal**

$\Rightarrow 11\square_2 = 22) + (1^* 22)\ \theta$ 2e)

$\Rightarrow 11\square_2 = 4 + 26$

$=> 11\square_2 = 6$

So equivalent decimal of binary integral is 6.

Step 2: Conversion of .101 to decimal

$\Rightarrow \square.101_2 = /2) + \theta\ /\ ) + (1^* 1/2^3)$

$$\Rightarrow \square.101_2 = 1^*\square.5 + \square^*\square.25 + 1^*\square.125$$

$$\Rightarrow \square.1\square1_2 = \square.625$$

So equivalent decimal of binary fractional is 0.625

Step 3: Add result of step 1 **and 2.**

⇒6+0.625=6.625

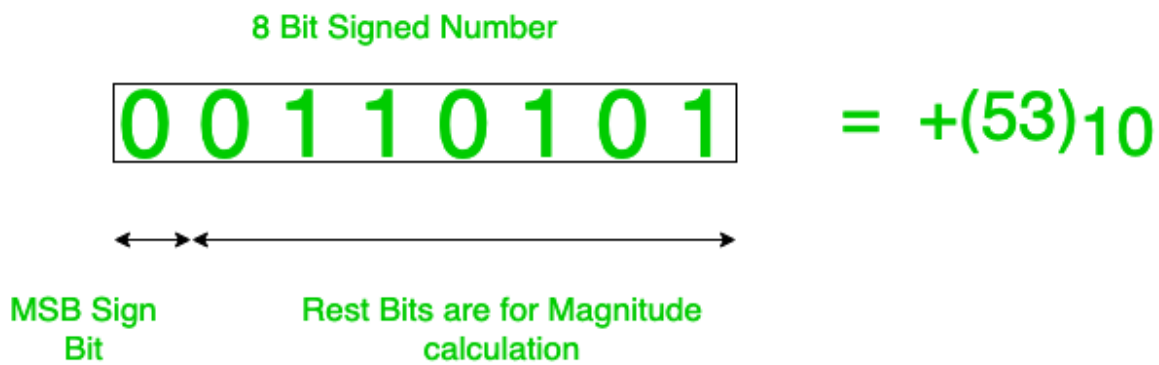
Signed Binary numbers

Core concepts of Signed Binary Numbers and Explanation of ranges of different data types. If representing the value of $(1000\ 0000)_2$ in decimal then certainly two ambiguous answers will come up $(-128)_{10}$ and $(+128)_{10}$. The answer is really ambiguous as both answers are correct. So the concept of signed and unsigned numbers comes up to help overcome the ambiguity. Now, if it is given to be unsigned number the $(+128)_{10}$ is the correct answer because there is no sign bit in case of unsigned numbers. Thus, here the MSB (Most Significant Bit) is not reserved to represent sign of number. But in case it is given that it is signed number, the $(-128)_{10}$ is the correct answer. In case of signed numbers the MSB is reserved to represent the sign of the number. Thus, if the number is of n bits, then in this 1 bit is used for representing sign of the number and rest (n-1) bits are used to represent the magnitude of the number. There are three methods to represent a Signed number:

1. **Sign Magnitude Form**
2. **1's Complement Form**

3. **2's Complement Form**

1. **Sign Magnitude Form:**
   Here, the MSB is reserved for signed bit, by using rest (n-1)bits we can directly get the decimal value using the normal formula of binary to decimal conversion(by multiplying 2i where i represents index position from LSB(Least Significant Bit)). Note that the index starts from 0 not 1 from LSB side.
2. **1's complement Form:**
   Here, the MSB is reserved for signed bit, and rest (n-1)bits are stored in form 1's complement of the number.



   This, will be clear from the below example: Consider to represent -7 using 4 bits in 1's complement. Here given, 4 bits so MSB 1 bit reserved to represent sign. So, now we are left with 3 bits. And we know $(+7)_{10}=(111_2$. But it is given to store the number in 1's complement form thus, 1's complement of (+7) is calculated. 1's complement of $(+7)_{10}=(000)_2$. (As 1's complement is calculated for flipping the zeros to ones and vice-versa). Thus, finally 1's complement representation of $(-7)_{10}=(1000)_2$. (using 4 bits).
3. **2's complement Form:**
   Here, the MSB is reserved for signed bit, and rest (n-1) bits are stored in form 2's complement of the number. What I mean to say will be clear from the below example: Consider the same example explained above i.e., represent -7 using 4 bits but now in 2's complement. Here given, 4 bits so MSB 1 bit reserved to represent sign. So, now we are left with 3 bits. And we know $(+7)_{10}=(111)_2$. But it is given to store the number in 2's complement form thus, 2's complement of (+7) is calculated. 1's complement of $(+7)_{10}=(001)_2$. (As 2's complement is calculated for flipping the zeros to ones and vice-versa) and adding 1 to the result we get after flipping the bits. In other words,

2's complement = 1'complement + 1
=(flipping of bits to get 1's complement)+1

Thus, finally 2's complement representation of $(-7)_{10}=(1001)_2$.(using 4 bits). But wait there is ambiguity, how can $(-128)_{10}=(1000\ 0000)_2$ in 8 bit 2's form signed number? But the trick is here, let us think that we don't know what $(1000\ 0000)_2$ represents and add $(+127)_{10}$ to it and after adding the result we got is $(1111\ 1111)_2= (-1)_{10}$. Thus, the only possible value of $(1000\ 0000)_2$ is $(-128)_{10}$ in 8 bit 2's form signed number. Now, this concept is allowed in 2's complement because in 2's complement there is concept of (negative 0). If we want to calculate by taking 2's complement of $(1000\ 0000)_2$ then after ignoring the MSBs(sign bit) the 1's complement of number is (111 1111) and 2's complement is(000 0000). Thus magnitude is 0 and sign is negative .Decimal number we should get is (-0) but as I have already said there is no concept of (negative 0) in 2's complement. Thus, this confirms $(-128)_{10}=(1000\ 0000)_2$.

Convert a negative decimal into a binary

Consider you have to represent -23

23=0001 0111

To represent it we have 3 ways you can use any but compliment representation is generally preferred

1. 1001 0111 signed magnitude representation 1st bit 1 means negative 0 for positive
2. 1110 1000 1's compliment representation
3. 1110 1001 2's compliment representation

Convert a negative binary into a decimal

If it is positive, simply convert it to decimal. If it is negative, make it positive by inverting the bits and adding one. Then, convert the result todecimal. The negative of this number is the valueof the original binary.

- Interpret 11011011 as a two's complement binary number, and give its decimal equivalent.

    1. First, note that the number is negative, since it starts with a 1.
    2. Change the sign to get the magnitude of the number.

$$\begin{array}{rcccccccc} & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ \neg & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ + & & & & & & & & 1 \\ \hline & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$$

3. Convert the magnitude to decimal: 00100101$)1_2 = 25_{16} = 2\times16 + 5 = 37_{10}$

4. Since the original number was negative, the final result is -37.

Complements

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulations.

There are two types of complements for each base-r system:

a) The r's complement and

b) The (r-1)'s complement.

r's complement is known as 10's complement in base 10 and 2's complement in base 2.☐

(r-1)'s complement is known as 9's complement in base 10 and 1's complement in base 2.

· **r's** complement

Given a positive number N in base r with an integer part of n digits, the r's complement of N is defined as

The r's complement$tof\ N = \{^{r^n - N, ifN \neq}_{0, ifN = 0}$

E.g.

$$10^{\circ}s\ complement\ of\ 52520 = 105 - 52520 = 47480$$

$$10^{\circ} s\ complement\ of\ 0.3267 = 100 - 0.3267 = 0.6733$$

$$10^{\circ} s\ complement\ of\ 25.639 = 102 - 25.639 = 74.361$$

$$2^{*} S\ COmplement\ of\ (101100)_2 = (26)_{100}(1011100)_2 = (1000000 - 1011100)_2 = 010100$$

$$2^{*} s\ complement\ of\ (0.0110)_2 = (20)_{10} - 0.0110 = 0.1010$$

· **(r-1)'s** complement Given a positive number N in base r with an integer part of n digits and a fractional part of m digits, the (r - 1)'s complement of N is defined as: The (r - 1)'s complement of N=rn-r m-N

E.g.

$$9^{*} \square \square \square \square \square \square \square (52520)10 = (10^{5} - 100 - 52520) = 47479$$

9's complement of (0.3267)10 is (100-10-4-0.3267)=0.6732

9's complement of (25.693)10 is (102-10-3-25.693)=74.306

1's complement of (101100)2i(26-20)10-(101100)2=1111111-101100=01001

1's complement of (0.0110)2 is (1-2-4)10-(0.0110)2=0.1001

Subtraction with Complements

Subtraction with r's Complements

Subtraction of two positive numbers (M- N), both of base r, may be done as follows:

Step-1: Add the minuend M to the r's complement of subtrahend *N.*

Step-2: Inspect the result obtained in step 1 for an end carry:

(a) If an end carry occurs,discard it.

(b) If an end carry does not occur, take the r's complement of the number obtained in step 1 and place a negative sign in front.

Subtraction with (r-1)'s Complements

The subtraction of M-N,both positive number in base r, may be calculated in the following manner:

Step-1: Add the minuend M to the (r - 1)'s complement of the subtrahend N.

Step-2: Inspect the result obtained in step 1 for an end carry.

(a) If an end carry occurs, add 1 to the list significant digit (end-round carry)

(b) If an end carry does not occurs, take the (r - 1)'s complement of the number obtained in step 1 and place a negative sign in front.

Binary Codes

When numbers, letters or words are represented by a special group of binary symbols/combinations, we say that they are being encoded and the groupof symbols is called a code. Some familiar binary codes are: Decimal Codes, Error-detection Codes,The Reflected Code, Alphanumeric Codes etc.

Decimal Codes

The representation of decimal digits by binary combinations is known as decimal codes. Binary codes from decimal digits require minimum of four bits. Numerous different codes

can be obtained by arranging four or more bits in ten distinct possible combinations. Some decimal codes are-

Binary-Coded-Decimal (BCD) Code

If each digit of a decimal number is represented by its binary equivalent, the result is a code called binary coded decimal (BCD). It is possible to assign weights to the binary bits according to their positions. The weights in the BCD code are 8,4,2,1.

Excess-3 Code (XS-3)

This is an un-weighted code. Its code assignment is obtained from the corresponding value of BCD after the addition of 3.

Decimal	BCD 8421	Excess-3 BCD+0011
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Error-detection Codes

- An error detection codes can be used to detect errors during transmission. A parity bit is an extra bit included with a message to make the total number of 1's either odd or even.

-For a message of four bits parity (P) is chosen so that the sum of all 1's is odd(in all five bits) or the sum of all 1's is even. In the receiving end, all the incoming bits (in thiscase five) are applied to a “parity-check" network for checking.

- An error is detected if the check parity does not correspond to the adopted one. The parity method detects the presence of one, three or any odd combination of errors. An even combination of errors is undetectable. Additional error-detection schemes may be needed to take care of an even combination of errors.

An error detection code is a binary code that detects digital errors during transmission. To detect errors in the received message, we add some extra bits to the actual data. Data-word bits along with parity bits is called a codeword. The parity bit is added to the message bits on the sender side, to help in error detection at the receiver side.

Without the addition of redundant bits, it is not possible to detect errors in the received message. There are 3 ways in which we can detect errors in the received message :

1. Parity Bit

2. CheckSum

3. Cyclic Redundancy Check (**CRC**)

Advantages of Parity Bit Method

- Parity bit method is a promising method to detect any odd bit error at the receiver side.
- Overhead in data transmission is low, as only one parity bit is added to the message bits at sender side before transmission.
- Also, this is a simple method for odd bits error detection.

Disadvantages of Parity Bit Method

- The limitation of this method is that only error in a odd number of bits are identified and we also cannot determine the exact location of error in the data bit, therefore, error correction is not possible in this method.
- If the number of bits in even parity check increase or decrease (data changed) but remained to be even then it won't be able to detect error as the number of bits are still even and same goes for odd parity check.

  Parity with 4-bit message:-

Message	P (Odd)	Total Message	Message	P (even)	Total Message
0000	1	10000	0000	0	00000
0001	0	00001	0001	1	10001
0010	0	00010	0010	1	10010
0011	1	10011	0011	0	00011
0100	0	00100	0100	1	10100
0101	1	10101	0101	0	00101
0110	1	10110	0110	0	00110
0111	0	00111	0111	1	10111
1000	0	01000	1000	1	11000
1001	1	11001	1001	0	01001
1010	1	11010	1010	0	01010
1011	0	01011	1011	1	11011
1100	1	11100	1100	0	01100
1101	0	01101	1101	1	11101
1110	0	01110	1110	1	11110
1111	1	11111	1111	0	01111

The Reflected Code / Gray Code

-The Reflected code, also called Gray code is un-weighted and is not an arithmetic code; that is, there are no specific weights assigned to the bit positions.

- It is a binary numeral system where two successive values differ in only one bit (binary digit).

- For instance, in going from decimal 3 to decimal 4, the Gray code changes from 0010 to 0110,while the binary code changes from 0011 to 0100, a change of three bits. The only bit change is in the third bit from the right in the Gray code; the others remain the same.

Decimal Digit	Binary	Reflected or gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Alphanumeric Code

- In order to communicate, we need not only numbers, but also letters and other symbols. In the strictest sense, alphanumeric codes are codes that represent numbers and alphabetic characters (letters). Most such codes, however, also represent other characters such as symbols and various instructions necessary for conveying information.

-The ASCII is the most common alphanumeric code.

ASCII Code

ASCII is the abbreviation for American Standard Code for Information Interchange. ASCII is a universally accepted alphanumeric code used in most computers and other electronic equipment. Most computer keyboards are standardized with the ASCII. When we enter a letter, a number, or control command, the corresponding ASCII code goes into the computer.

- ASCII has 128 characters and symbols represented by a 7-bit binary code. Actually, ASCII can be considered an 8-bit code with the MSB always 0. This 8-bit code is 00through 7F in hexadecimal.

-The first thirty-two ASCII characters are non-graphic commands that are never printed or displayed and are used only for control purposes. Examples of the control characters are ""null," "line feed," "start of text," and "escape."

-The other characters are graphic symbols that can be printed or displayed and include the letters of the alphabet (lowercase and uppercase), the ten decimal digits, punctuation signs and other commonly used symbols.

dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	|
29	1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

www.alpharithms.com

Extended ASCII characters

In addition to the 128 standard ASCII characters, there are an additional 128 characters that were adopted by IBM for use in their PCs (personal computers). Because of the popularity of the PC, these particular extended ASCII characters are also used in applications other than PCs and have become essentially an unofficial standard. The extended ASCII characters are represented by an 8-bit code series from hexadecimal 80 to hexadecimal FF.

Extended ASCII Printing Characters Chart (character codes 128-255)

This has many variations - below is the Microsoft variation and is called ANSI or Windows-

Decimal	Character	Decimal	Character	Decimal	Character	Decimal	Character
128	€	160		192	À	224	à
129		161	¡	193	Á	225	á
130	‚	162	¢	194	Â	226	â
131	ƒ	163	£	195	Ã	227	ã
132	„	164	¤	196	Ä	228	ä
133	…	165	¥	197	Å	229	å
134	†	166	¦	198	Æ	230	æ
135	‡	167	§	199	Ç	231	ç
136	ˆ	168	¨	200	È	232	è
137	‰	169	©	201	É	233	é
138	Š	170	ª	202	Ê	234	ê
139	‹	171	«	203	Ë	235	ë
140	Œ	172	¬	204	Ì	236	ì
141		173	-	205	Í	237	í
142	Ž	174	®	206	Î	238	î
143		175	¯	207	Ï	239	ï
144		176	°	208	Ð	240	ð
145	‘	177	±	209	Ñ	241	ñ
146	’	178	²	210	Ò	242	ò
147	“	179	³	211	Ó	243	ó
148	”	180	´	212	Ô	244	ô
149	•	181	µ	213	Õ	245	õ
150	–	182	¶	214	Ö	246	ö
151	—	183	·	215	×	247	÷
152	˜	184	¸	216	Ø	248	ø
153	™	185	¹	217	Ù	249	ù
154	š	186	º	218	Ú	250	ú
155	›	187	»	219	Û	251	û
156	œ	188	¼	220	Ü	252	ü
157		189	½	221	Ý	253	ý
158	ž	190	¾	222	Þ	254	þ
159	Ÿ	191	¿	223	ß	255	ÿ

EBCDIC character code

EBCDIC (Extended Binary Coded Decimal Interchange Code) is another alphanumeric code used in IBM equipment. It uses eight bits for each character. EBCDIC has the same character symbols as ASCII, but the bit assignment for characters is different. As the name implies, the binary code for the letters and numerals is an extension of the binary-coded decimal (BCD) code. This means that the last four bits of the code range from 0000 through 1001 as in BCD.