Digital Electronics

By Sonika P. Biswal

Number System

- Number System that tells us how can we store or represent numbers in a digital system.
- A digital system can understand positional number system only where few symbols are called digits and these symbols represent different values depending on the position they occupy in the number.

A value of each digit in a number can be determined using

- Position of the digit in the number
- Base of the number system

(where base is defined as the total number of digits available in the number system).

System	Base	Symbols	Used By Humans?	Used By Computers?
Decimal	10	0,1,………..9	Yes	No
Binary	2	0,1	No	Yes
Octal	8	0,1,………..7	No	No
Hexa-decimal	16	0,1,………..9 A,B,………..F	No	No

Relation between Binary, Octal and Hexagonal Number System

Decimal	Binary	Octal	Hexa-Decimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7

Decimal	Binary	Octal	Hexa-Decimal
8	1000	10	8
9	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

Decimal	Binary
16	10000
17	10001
18	10010
19	10011
20	10100
21	10101
22	10110
23	10111

e.g. Represent 18 in binary and Octal.

$$(18)_{10} \rightarrow (10010)_2 \rightarrow \overset{2}{\overline{010}}\ \overset{2}{\overline{010}} \rightarrow (22)_8$$

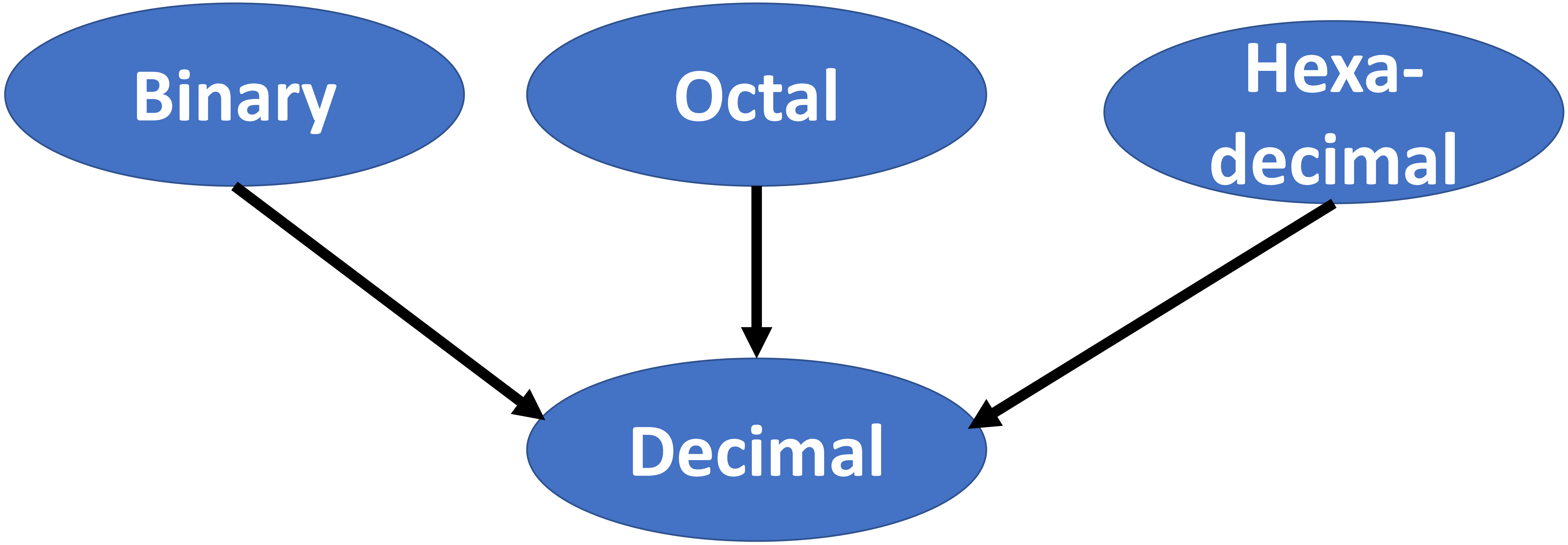
e.g. Represent 18 in binary and Hexadecimal.

$$(18)_{10} \rightarrow (10010)_2 \rightarrow \overset{1}{\overline{0001}}\ \overset{2}{\overline{0010}} \rightarrow (12)_{16}$$

Relation between Binary, Octal and Hexagonal Number System

Decimal	Binary	Octal	Hexa-Decimal	Decimal	Binary	Octal	Hexa-Decimal	Decimal	Binary	Octal	Hexa-Decimal
0	0	0	0	8	1000	10	8	**16**	10000	20	10
1	1	1	1	9	1001	11	9	**17**	10001	21	11
2	10	2	2	10	1010	12	**A**	**18**	10010	22	12
3	11	3	3	11	1011	13	**B**	**19**	10011	23	13
4	100	4	4	12	1100	14	**C**	**20**	10100	24	14
5	101	5	5	13	1101	15	**D**	**21**	10101	25	15
6	110	6	6	14	1110	16	**E**	**22**	10110	26	16
7	111	7	7	15	1111	17	**F**	**23**	10111	27	17

Binary/Octal/ Hexadecimal to Decimal Conversion



Binary/Octal/ Hexadecimal to Decimal Conversion

- Multiply each bit by b^n, where n is the weight of the bit and b represents the base w.r.t binary. b = 2 (for Binary) = 8 (for Octal) = 16 (for Hexagonal)
- The weight of the position of the digit starting from 0 on the right.
- Add all results.
- e.g. $\underline{(101011)}_2$

$$(101011)_2 = \begin{array}{r} 1 \times 2^0 = 1 \\ 1 \times 2^1 = 2 \\ 0 \times 2^2 = 0 \\ 1 \times 2^3 = 8 \\ 0 \times 2^4 = 0 \\ 1 \times 2^5 = 32 \\ \hline = (43)_{10} \end{array}$$

Binary to decimal

$$(724)_8 = \begin{array}{r} 4 \times 8^0 = 4 \\ 2 \times 8^1 = 16 \\ 7 \times 8^2 = 448 \\ \hline = (468)_{10} \end{array}$$

Octal to decimal

$$(ABC)_{16} = \begin{array}{r} 12 \times 16^0 = 12 \\ 11 \times 16^1 = 176 \\ 10 \times 16^2 = 2560 \\ \hline = (2748)_{10} \end{array}$$

hexadecimal to decimal

Octal to Binary

- Represent the octal number in 3-bit equivalent binary.

  Or

  Octal → Decimal → Binary

Example:
$$\begin{array}{ccc} 7 & 1 & 2 \\ \downarrow & \downarrow & \downarrow \\ 111 & 001 & 010 \end{array}$$

$(712)_8 = (011100010010)_2$

$(712)_8$
$= 2 \times 8^0 + 1 \times 8^1 + 7 \times 8^2$
$= (458)_{10}$

$$(458)_{10} = \begin{array}{r|l l} 2 & 458 & \\ 2 & 229 & 0 \\ 2 & 114 & 1 \\ 2 & 57 & 0 \\ 2 & 28 & 1 \\ 2 & 14 & 0 \\ 2 & 7 & 0 \\ 2 & 3 & 1 \\ 2 & 1 & 1 \\ & 1 & \end{array} \uparrow$$

Ans = $(111001010)_2$

Binary to Octal

- Represent the binary number in 3-bit equivalent binary.
- Arrange the number from right to Left.

Example:

$(10010)_2 \rightarrow \overset{2}{\overline{010}}\ \overset{2}{\overline{010}} \rightarrow (22)_8$

$\xleftarrow{\hspace{3em}}$

$(1011010111)_2 = (?)_8$

$\underset{1}{001}\ \ \underset{3}{011}\ \ \underset{2}{010}\ \ \underset{7}{111} \quad = (1327)_8$

Hexadecimal to Binary

- Represent the Hexadecimal number in 4-bit equivalent binary.

Or

Hexadecimal → Decimal → Binary

Example:

$$\begin{tikzcd}
7 \arrow[d] & 1 \arrow[d] & 2 \arrow[d] & & 1 \arrow[d] & 0 \arrow[d] & A \arrow[d] & F \arrow[d] \\
0111 & 0001 & 0010 & & 0001 & 0000 & 1010 & 1111
\end{tikzcd}$$

$$(712)_{16} = (011100010010)_2$$
$$(10AF)_{16} = (0001000010101111)_2$$

Binary to Hexadecimal

- Represent the binary number in 4-bit equivalent binary.
- Arrange the number from right to Left.

Example:

$$\xleftarrow{\hspace{3em}}$$
$$(1011010111)_2 = (?)_{16}$$

$$\begin{array}{ccc} 0010 & 1101 & 0111 \\ 2 & D & 7 \end{array} = (2D7)_{16}$$

Octal to Hexadecimal

- Convert Octal to Binary
- Represent Binary in hexadecimal

Octal→ Binary → Hexadecimal

Example: $(1076)_8 = (?)_{16}$

$$\begin{tikzcd}
1 \arrow[d] & 0 \arrow[d] & 7 \arrow[d] & 6 \arrow[d] \\
001 & 000 & 111 & 110
\end{tikzcd}$$

$\xleftrightarrow{\quad}\xleftrightarrow{\quad}\xleftrightarrow{\quad}$

0010 0011 1110

2 3 E

$(1076)_8 = (23F)_{16}$

Hexadecimal to Octal

- Convert Hexadecimal to Binary
- Represent Binary in Octal

Hexadecimal → Binary → Octal

Example: $(1F0C)_{16} = (?)_8$

$$\begin{tikzcd}
1 \arrow[d] & F \arrow[d] & 0 \arrow[d] & C \arrow[d] \\
0001 & 1111 & 0000 & 1100
\end{tikzcd}$$

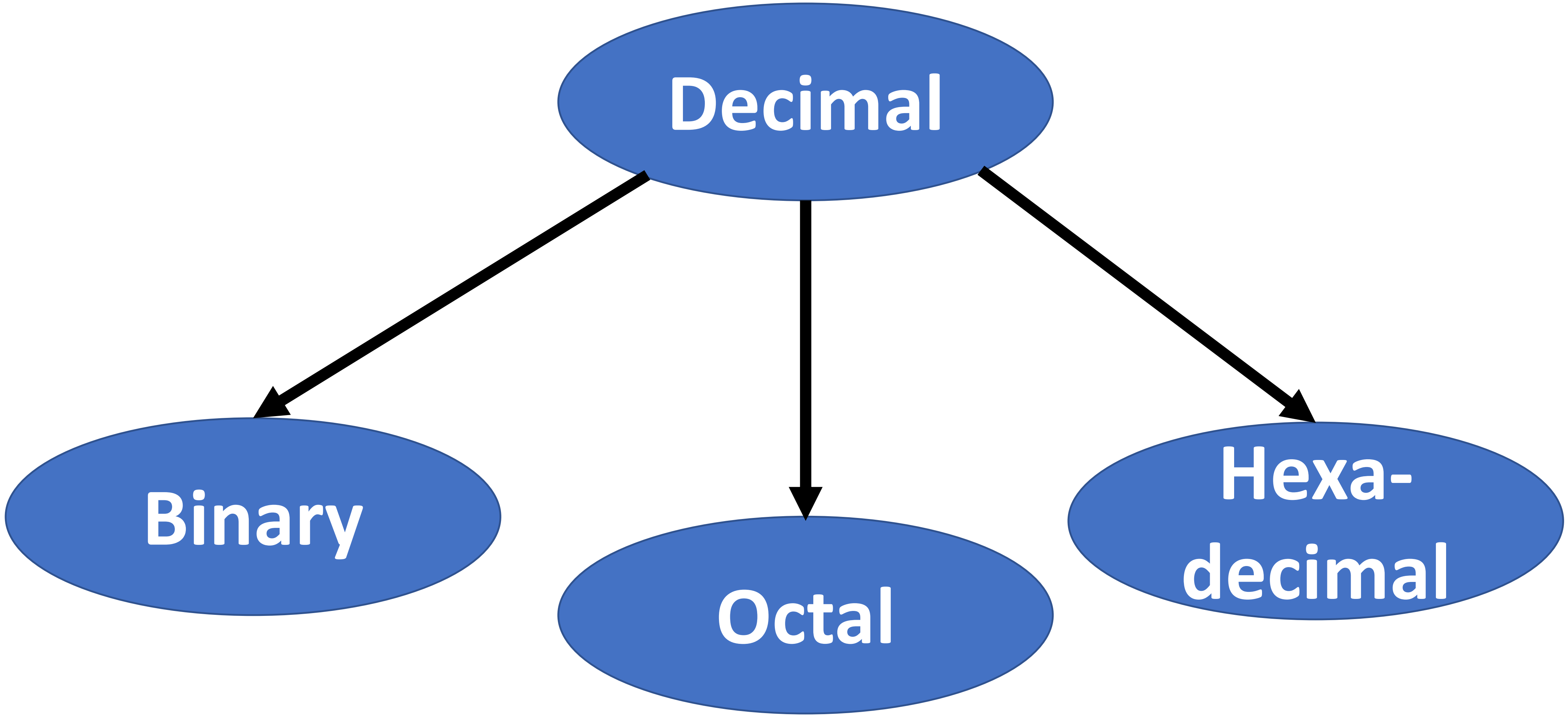
$\xleftrightarrow{\quad}\ \xleftrightarrow{\quad}\xleftrightarrow{\quad}\xleftrightarrow{\quad}\xleftrightarrow{\quad}$

001 111 100 001 100

1 7 4 1 4

$(1F0C)_{16} = (17414)_8$

Decimal to Binary/Octal/ Hexadecimal



Decimal to Binary

- Divide the decimal number by 2 and keep track of the remainder.

$(125)_{10} = (?)_2$

$$= \begin{array}{r|l l}
2 & 125 & \\ \hline
2 & 62 & 1 \\ \hline
2 & 31 & 0 \\ \hline
2 & 15 & 1 \\ \hline
2 & 7 & 1 \\ \hline
2 & 3 & 1 \\ \hline
2 & 1 & 1 \\ \hline
 & 0 & 1
\end{array} \uparrow$$

Ans :
$(125)_{10} = (01111101)_2$

Decimal to Octal

- Divide the decimal number by 8 and keep track of the remainder.

$(1234)_{10} = (?)_8$

$$= \begin{array}{r|l l}
8 & 1234 & \\ \hline
8 & 154 & 2 \\ \hline
8 & 19 & 2 \\ \hline
8 & 2 & 3 \\ \hline
 & 0 & 2
\end{array} \uparrow$$

Ans :
$(1234)_{10} = (2322)_8$

Decimal to Hexagonal

- Divide the decimal number by 16 and keep track of the remainder.

$(1234)_{10} = (?)_{16}$

$$= \begin{array}{r|l l}
16 & 1234 & \\ \hline
16 & 77 & 2 \\ \hline
16 & 4 & 13 = D \\ \hline
 & 2 & 4
\end{array} \uparrow$$

Ans : $(1234)_{10} = (4D2)_{16}$

Representation of Signed Numbers

1) Signed Magnitude form

0 = Positive (+)

1 = Negative (-)

Syntax: sign bits Actual binary number

e.g. +7 → 0111

-7 → 1111

2) Compliment Form

1. 1's Compliment → Complimenting/ Reversing each and every bit of a binary number.
2. 2's Compliment → Adding 1 to the 1's compliment number.

Example-1:

1'S Compliment of 10 : 1010 (In Binary)

↓↓↓↓

0101 (Compliment)

Ans = 0101

Example-2:

1'S Compliment of 125 : 01111101 (In Binary)

↓↓↓↓↓↓↓↓

10000010 (Compliment)

Ans = 0101

Example-3:

2'S Compliment of 10: 1010 (In Binary)

↓↓↓↓

$$\begin{array}{rl} 0101 & \text{(1's Compliment)} \\ \underline{+ \quad 1} & \\ 0110 & \text{(2's Compliment)} \end{array}$$

Contd..

Example-4: Represent -12 in 1’s compliment and 2’s compliment form.

1’S Compliment of -12 : 1100 (12 In Binary)

$\downarrow\downarrow\downarrow\downarrow$

0011 (1’s compliment of 12)

10011 (1’s compliment of -12)

Ans = 10011: Here, 1 represents the signed bit

Syntax :

Sign bit	1’s Compliment of Actual Binary

2’S Compliment of -12 : 0011 (1’s Compliment of 12)

$$\begin{array}{r} 0011 \\ +\quad 1 \\ \hline 0100 \end{array}$$

0100 (2’s Compliment of 12)

10100 (2’s Compliment of -12)

Ans = 10100: Here, 1 represents the Sign bit)

Syntax :

Sign bit	2’s Compliment of Actual Binary

Binary Arithmetic

Binary Addition : It is a key for binary subtraction, multiplication, division.

Case	A + B	Sum	Carry
1	0 + 0	0	0
2	0 + 1	1	0
3	1 + 0	1	0
4	1 + 1	0	1

Binary Multiplication:

Case	A - B	Multiplication
1	0 × 0	0
2	0 × 1	0
3	1 × 0	0
4	1 × 1	1

Example:

$$\begin{array}{rl}
0011010 & = 26_{10} \\
\times\ 0001100 & = 12_{10} \\ \hline
0000000 & \\
0000000 & \\
0011010 & \\
0011010 & \\ \hline
0100100000 & = 312_{10}
\end{array}$$

Binary Division:

Example: 101010/000110 = ?

Binary Subtraction:

Case	A - B	Sub	Borrow
1	0 - 0	0	0
2	0 - 1	0	1
3	1 - 0	1	0
4	1 - 1	0	0

Binary Code

- When numbers, letters or words are represented by a particular group of symbols, it is said that the number, letter or word is being encoded.
- The group of symbols is called as a code.
- The digital data is represented, stored and transmitted as group of binary bits. This group is also called as **binary code**.
- The binary code is represented by the number as well as alphanumeric letter.

Advantages:

- Binary codes are suitable for the computer applications.
- Binary codes are suitable for the digital communications.
- Binary codes make the analysis and designing of digital circuits if we use the binary codes.
- Since only 0 & 1 are being used, implementation becomes easy.

Classification of binary codes:

The codes are broadly categorized into following categories.

1. Weighted Codes
2. Non-Weighted Codes
3. Binary Coded Decimal Code
4. Alphanumeric Codes
5. Error Detecting Codes
6. Error Correcting Codes

Weighted Codes

Classified into Two types,

1) Positive Weighted Code e.g. 8421, 2421, 3321, 4221
2) Negative Weighted Code e.g. $84\bar{2}\bar{1}$, $74\bar{2}\bar{1}$

Representation of 8421:

- Add each bits in **8421** such a manner that the result must give decimal number.
- The contributing bit will be assigned as **1** and rest are **0**.
- Similar process for **2421, 3321, 4221, $84\bar{2}\bar{1}$, $74\bar{2}\bar{1}$** (here $\bar{2}$, $\bar{1}$ means -2 , -1)

Decimal	8421	
0	0000	all bit '0' ,as no bits are contributing.
7	0111	Three bits are '1' , three bits (4,2,1) are contributing to result 7. i.e. 4+2+1=7
10	1010	Two bits are '1' , as two bits (8 & 2) are contributing to result 10. i.e. 8+2=10
15	1111	All bits are '1' ,as all bits (8,4,2,1) are contributing to result 7. i.e. 8+4+2+1=15

Decimal	2421
0	0000
7	0111
9	1111

Decimal	$84\bar{2}\bar{1}$
0	0000
1	0111
5 ~~9~~	1011

4-2-1 = 1, as 4,2, & 1 are responsible to result 1. So the bits in place of 4, 2', 1' are 1

8+4-2-1 =9, so the bits in place of 8 4 2' 1' are 1

Exclusive addition (⊕)

$$\begin{array}{r} 0\,0\,1\,1 \\ \oplus\ \underline{0\,1\,0\,1} \\ 0\,1\,1\,0 \end{array}$$

Non-weighed Codes

Classified into two types,

1) **Excess-3 Code** Convert the decimal to 8421 binary and then add 011. or add 3 to decimal then convert to Binary.
2) **Gray Code** Keep the MSB constant, take exclusive addition (⊕) among other bits or Use Shifting method.

Decimal	8421	Excess-3	
0	0000	0000 (8421 BCD) + 011 0011	0 +3 0011
4	0100	0100 (8421 BCD) + 011 0111	4 +3 7 (0111)
9	1001	1100	

Decimal	8421	Gray
6	0101	0 1 0 1 0 1 1 1
9	1001	1101

Shifting method

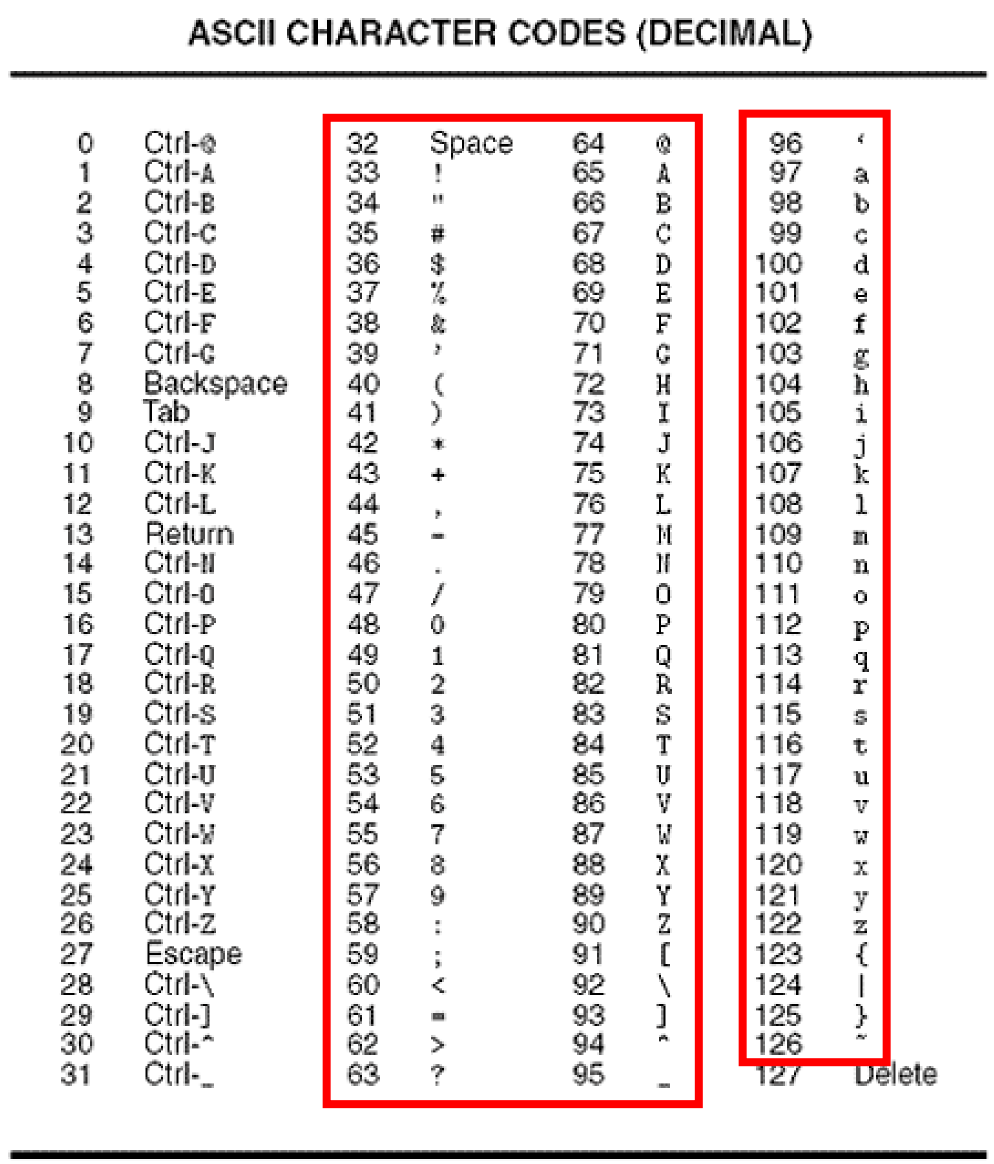
$$\begin{array}{r} 1\,0\,0\,1 \\ \oplus\ \underline{1\,0\,0\,1} \\ 1\,1\,0\,1 \end{array}$$

Delete this

Ans

ASCII Code

- ASCII Stands for American Standard for information interchange.
- An alphanumeric code used for data communication/ information interchange in digital computers.
- It consists of numbers, alphabets and special characters.
- The ASCII is a 7-bit code. It can result 128 number of possible characters.
- Out of 128 numbers, 34 are printable and 94 are printable.
- Extended ASCII, 8-bit code → can provide 256 number of characters.



ASCII CHARACTER CODES (DECIMAL)

0	Ctrl-@	32	Space	64	@	96	`
1	Ctrl-A	33	!	65	A	97	a
2	Ctrl-B	34	"	66	B	98	b
3	Ctrl-C	35	#	67	C	99	c
4	Ctrl-D	36	$	68	D	100	d
5	Ctrl-E	37	%	69	E	101	e
6	Ctrl-F	38	&	70	F	102	f
7	Ctrl-G	39	'	71	G	103	g
8	Backspace	40	(	72	H	104	h
9	Tab	41	)	73	I	105	i
10	Ctrl-J	42	*	74	J	106	j
11	Ctrl-K	43	+	75	K	107	k
12	Ctrl-L	44	,	76	L	108	l
13	Return	45	-	77	M	109	m
14	Ctrl-N	46	.	78	N	110	n
15	Ctrl-O	47	/	79	O	111	o
16	Ctrl-P	48	0	80	P	112	p
17	Ctrl-Q	49	1	81	Q	113	q
18	Ctrl-R	50	2	82	R	114	r
19	Ctrl-S	51	3	83	S	115	s
20	Ctrl-T	52	4	84	T	116	t
21	Ctrl-U	53	5	85	U	117	u
22	Ctrl-V	54	6	86	V	118	v
23	Ctrl-W	55	7	87	W	119	w
24	Ctrl-X	56	8	88	X	120	x
25	Ctrl-Y	57	9	89	Y	121	y
26	Ctrl-Z	58	:	90	Z	122	z
27	Escape	59	;	91	[	123	{
28	Ctrl-\	60	<	92	\	124	|
29	Ctrl-]	61	=	93	]	125	}
30	Ctrl-^	62	>	94	^	126	~
31	Ctrl-_	63	?	95	_	127	Delete

Binary to ASCII

- Arrange the binary number to set of 7-bit.
- Convert Binary to equivalent decimal.
- Convert the decimal to ASCII (as per the table).

64 32 16 8 4 2 1 64 32 16 8 4 2 1 64 32 16 8 4 2 1

Example 1001001 1000001 1001101

73 65 77

I A M = Ans

$77 = 1 \times 1 + 0 \times 2 + 1 \times 4 + 1 \times 8 + 0 \times 16 + 0 \times 32 + 1 \times 64$

$65 = 1 \times 1 + 0 \times 2 + 0 \times 4 + 0 \times 8 + 0 \times 16 + 0 \times 32 + 1 \times 64$

$73 = 1 \times 1 + 0 \times 2 + 0 \times 4 + 1 \times 8 + 0 \times 16 + 0 \times 32 + 1 \times 64$

Error Detection and Correction Code

- Error detection and correction code plays an important role in the transmission of data from one source to another.
- The noise also gets added into the data when it transmits from one system to another, which causes errors in the received binary data at other systems.
- The bits of the data may change(either 0 to 1 or 1 to 0) during transmission. It is impossible to avoid the interference of noise, but it is possible to get back the original data with the help of error correction code.
- The error detection codes are the code used for detecting the error in the received data bitstream. In these codes, some bits are included appended to the original bitstream.

Error Correction Codes can be broadly categorized into two types −

•**Block codes** − The message is divided into fixed-sized blocks of bits, to which redundant bits are added for error detection or correction.

•**Convolutional codes** − The message comprises of data streams of arbitrary length and parity symbols are generated by the sliding application of a Boolean function to the data stream

Hamming Code

- Hamming code is a block code (Error Correction Code) that can detect up to two simultaneous bit errors and correcting single-bit errors.
- Here the source encodes the message by inserting Parity bits (redundant bits) within the message. (for security purpose)
- These Parity bits are extra bits that are generated and inserted at specific positions in the message itself to enable error detection and correction.
- When the receiver receives this message, it performs recalculations to detect errors and find the bit position that has error.
- The Parity bit can be Even or Odd.

Procedure to encode a message by Hamming Code

- **Step 1** − Calculation of the number of parity bits.
- **Step 2** − Positioning of the parity bits.
- **Step 3** − Calculating the values of each parity bit.

Satisfying criteria to determine the position of parity bits $2^P \geq P + M + 1$

M = no. of message bits

P = Parity bits

Example: M_1 M_2 M_3 M_4 (Transmitted Bits)

$$\downarrow$$

P_1 M_1 P_2 M_2 P_3 M_3 P_4 M_4

(Hamming Code)

P_1, P_2, P_3, P_4 are the Parity Bits.

Example : Generate Hamming Code (HC) for the message 1110 in both Even and Odd Parity.

$2^P \geq P + M + 1$

$\geq P + 5$

$P = 3 \rightarrow 8 \geq 8$ (satisfying the criteria)

i.e. To transmit 4 message bits we need at least 3 parity bits.

i.e. To transmit 4 message bits we need at least 3 parity bits.

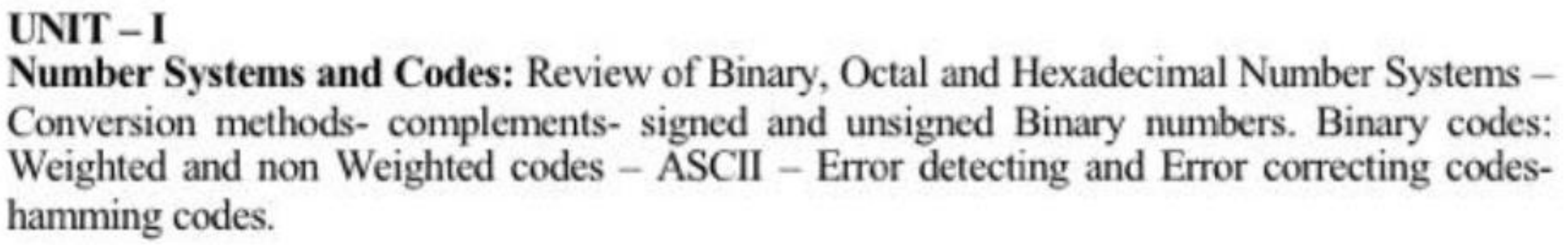
Position for Even Parity bit (Even)

P_1= 1,3,5,7,9,11…… $\rightarrow P_1$ 1 1 0 $\rightarrow$ 0 1 1 0

P_2= 2,3,6,7,10,11….. $\rightarrow P_2$ 1 1 0 $\rightarrow$ 0 1 1 0

P_3= 4,5,6,7,12 to 15, 20 to 23….. $\rightarrow P_3$ 1 1 0 $\rightarrow$ 0 1 1 0

Position of the Parity Bits $\rightarrow$	2^0	2^1		2^2				2^3
Position of the coded message $\rightarrow$	1	2	3	4	5	6	7	8
	P_1	P_2	M_1	P_3	M_2	M_3	M_4	
	P_1	P_2	1	P_3	1	1	0	
	0	0	1	0	1	1	0	HC with Even Parity
	1	1	0	1	0	0	1	HC with Odd Parity (take compliment of HC with Even Parity)



UNIT – I

Number Systems and Codes: Review of Binary, Octal and Hexadecimal Number Systems – Conversion methods- complements- signed and unsigned Binary numbers. Binary codes: Weighted and non Weighted codes – ASCII – Error detecting and Error correcting codes- hamming codes.

Completed