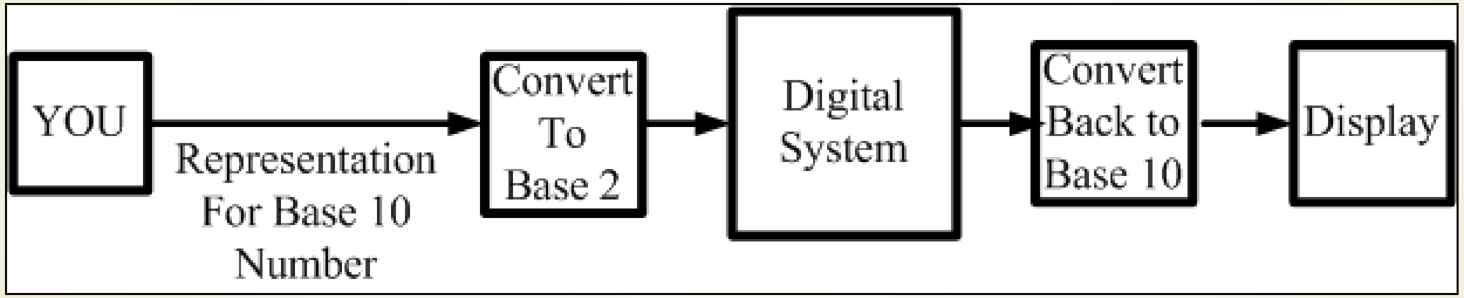
Codes

Presented by Nabanita Das

Human perception

- We naturally live in a base 10 environment
- Computer exist in a base 2 environment
- So give the computer/digital system the task of doing the conversions for us.



Codes

- In the coding, when numbers or letters are represented by a specific group of symbols, it is said to be that number or letter is being encoded. The group of symbols is called as **code**. The digital data is represented, stored and transmitted as group of bits. This group of bits is also called as **binary code**.
- Binary codes can be classified into two types.
- Weighted codes
- Unweighted codes
- If the code has positional weights, then it is said to be **weighted code**. Otherwise, it is an **unweighted code**. Weighted codes can be further classified as positively weighted codes and negatively weighted codes.

Classification of codes

- CODES
  - Weighted Codes
  - Non-weighted Code
  - Reflective Codes
  - Sequential Code
  - Alpha numeric

1. Weighted Codes

- Obey positional weight principle.
- A specific weight is assigned to each position of the number.
- Eg.: Binary, BCD codes

2. Non-weighted Codes

- Do not obey positional weight principle.
- Positional weights are not assigned.
- Eg.: excess-3 code, Gray code

3. Reflective Codes

- A code is said to be reflective when code for 9 is complement of code for 0, code for 8 is complement of code for 1, code for 7 is complement of code for 2, code for 6 is complement of code for 3, code for 5 is complement of code for 4.
- Reflectivity is desirable when 9's complement has to be found.
- Eg.: excess-3 code

4. Sequential Codes

- A code is said to be sequential when each succeeding code is one binary number greater than preceding code.
- Eg.: Binary, XS-3

5. Alphanumeric Codes

- Designed to represent numbers as well as alphabetic characters.
- Capable of representing symbols as well as instructions.
- Eg.: ASCII, EBCDIC

BCD(Binary Coded Decimal) Code

- In this code each digit is represented by a 4-bit binary number.
- The positional weights assigned to the binary digits in BCD code are 8-4-2-1 with 1 corresponding to LSB and 8 corresponding to MSB.

Positional Weights	8	4	2	1
	2^3 MSB	2^2	2^1	2^0 LSB

Conversion from decimal to BCD

- The decimal digits 0 to 9 are converted into BCD, exactly in the same way as binary.

Digital	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Invalid BCD codes:

- With 4 bits we can represent total sixteen numbers (0000 to 1111) but in BCD only first ten codes are used (0000 to 1001)
- Therefore remaining six codes **(1010 to 1111)are invalid in BCD**

Decimal and BCD

- The BCD is simply the 4 bit representation of the decimal digit.
- For multiple digit base 10 numbers, each symbol is represented by its BCD digit.
- What happened to 6 digits not used?
- We have 10 digits in decimal number system. To represent these 10 digits in binary, we require minimum of 4 bits. But, with 4 bits there will be 16 unique combinations of zeros and ones. Since, we have only 10 decimal digits, the other 6 combinations of zeros and ones are not required.

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Example:

- A second example
  - 3 0 0 1 1
  - +3 <u>0 0 1 1</u>
  - Getting 6 or 0 1 1 0
  - And in range and a BCD digit representation

Another example

- Add 7 + 6
  - have 7 0 1 1 1
  - plus 6 0 1 1 0
  - Giving 1 1 0 1 and again out of range
  - Adding 6 0 1 1 0
  - Giving 1 0 0 1 1 so a 1 carries out to the next BCD digit
  - FINAL BCD answer 0001 0011 or 13_{10}

Advantages of BCD codes:

- Its similar to decimal number system.
- We need to remember binary equivalents of decimal numbers 0 to 9 only.
- Conversions from decimal to BCD or BCD to decimal is very simple and no calculation is needed.

Disadvantages of BCD codes:

- Less efficient than binary, since conversion of a decimal number into BCD needs more bits than in binary
- BCD arithmetic is more complicated than binary arithmetic.

XS-3 Code

- Non-weighted code.
- Derived from BCD code (8-4-2-1 code)words by adding $(0011)_2$ or $(3)_{10}$ to each code word.

$$\text{Decimal} \xrightarrow{\text{Write each digit in 4-bit binary code}} \text{BCD} \xrightarrow{+ (0011)} \text{XS-3}$$

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

- Therefore, hence smallest number in XS-3 is 0011 i.e., 0 and largest is 1100 i.e., 9
- XS-3 is a reflective code since code for 9 is complement of code for 0, code for 8 is complement of code for 1, code for 7 is complement of code for 2, code for 6 is complement of code for 3, code for 5 is complement of code for 4.
- It is a sequential code since each number is 1 binary bit greater than its preceding number.

Binary to Excess-3 code conversion

1. Convert the binary number into decimal.
2. Add 3 in each digit of the decimal number.
3. Find the binary code of each digit of the newly generated decimal

Convert $(11110)_2$ to Excess-3 using binary number.

$(11110)_2$	$((1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0))_{10}$
	$(16 + 8 + 4 + 2 + 0)_{10}$
	$= (30)_{10}$

- Decimal number of the Binary number **$(11110)_2$ is $(30)_{10}$**

Now, we add 3 in each digit of the decimal number.

- The decimal number is 30. Now, we will add 3 into the decimal number 30.

3	0
+3	+3
=6	=3

Now, we find the binary code of each digit of the decimal number 63.

63_{10}	$(0110)_2\ (0011)_2$
	$(01100011)_{Excess\text{-}3}$

Excess-3 to Binary Conversion

- In the first step, we will make the group of 4 bits and write the equivalent decimal number.
- Subtract 3 in each digit of the decimal number.
- At last, we find the binary number of the decimal number using a decimal to binary conversion.

Convert $(01100011)_{Excess\text{-}3}$ to binary number.

- **Making groups of four bits and write their equivalent decimal number.**
- **$(01100011)_{Excess\text{-}3} = (0110\ 0011)_{xcess\text{-}3} = (6\ 3)_{10}$**
- **Subtract 3 in each digit of the decimal number.**
-

6	**3**
-3	**-3**
=3	**=0**

- **Now, find the binary number.**
- Now, find the binary number of the decimal number $(30)_{10}$ using a decimal to binary conversion is $\mathbf{(30)_{10}=(11110)_2}$
- So, the binary number of excess-3 code 01100011 is: $\mathbf{(11110)_2}$

Gray Code

- Non-weighted code.
- It has a very special feature that only one bit will change, each time the decimal number is incremented, therefore also called unit distance code.

Binary and Gray conversions:

- For Gray to binary or binary to Gray conversions let's understand rules for Ex-OR (Ex-OR is represented by symbol ⊕)

Rules for EX-OR:

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

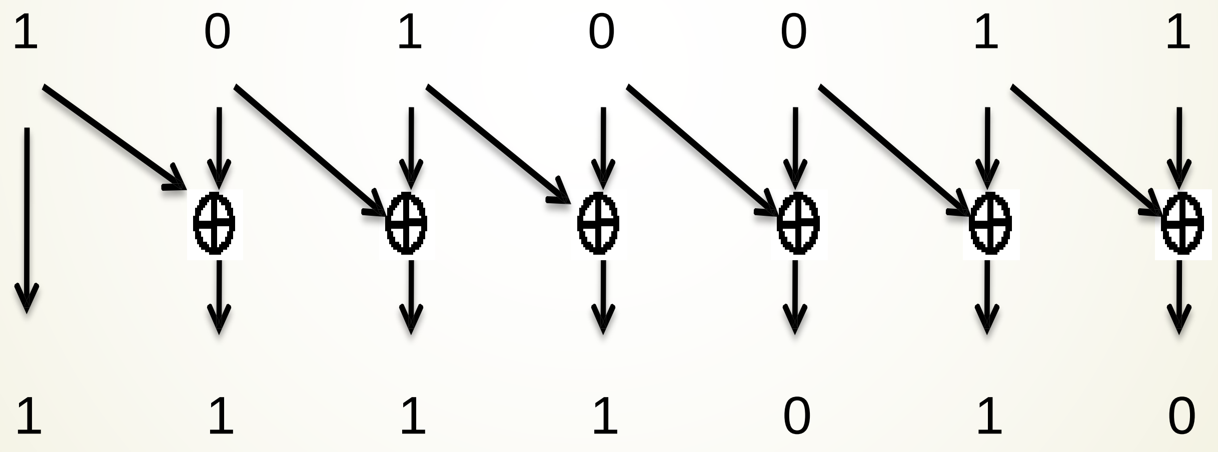
Conversion from Binary to Gray code:

Step 1: Write MSB of given Binary number as it is.

Step 2: Ex-OR this bit with next bit of that binary number and write the result.

Step 3: Ex-OR each successive sum until LSB of that binary number is reached.

- Eg.: Convert $(1010011)_2$ to its equivalent Gray code.



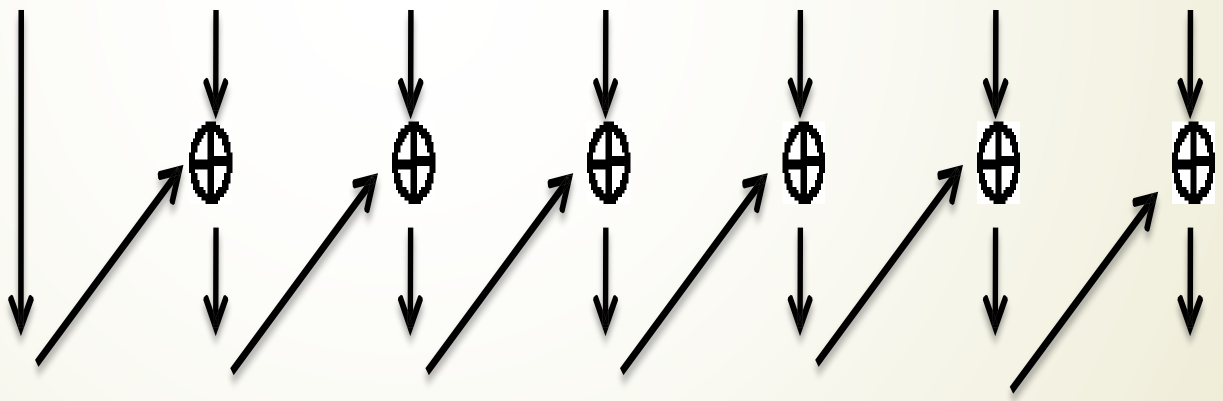
Therefore $(1010011)_2 = (1111010)_{Gray}$

Conversion from Gray to Binary:

- Step 1: Write MSB of given Binary number as it is.
- Step 2: Ex-OR this bit with next bit of that binary number and write the result.
- Step 3: Continue this process until LSB of that binary number is reached.

E.g.: Convert $(1010111)_{Gray}$ to its equivalent Binary number.

- 1 0 1 0 1 1 1



1 1 0 0 1 0 1

Therefore $(1010111)_{Gray} = (1100101)_2$

Alphanumeric Codes

- A binary bit can represent only two symbols ‘0’ and ‘1’. But it is not enough for communication between two computers because there we need many more symbols for communication.
- These symbols are required to represent

- 26 alphabets with capital and small letters
- Numbers from 0 to 9
- Punctuation marks and other symbols

- Alphanumeric codes represent numbers and alphabetic characters. They also represent other characters such as punctuation symbols and instructions for conveying information.
- Therefore instead of using only single binary bits, a group of bits is used as a code to represent a symbol.

The ASCII code

b_7						0	0	0	0	1	1	1	1
b_6						0	0	1	1	0	0	1	1
b_5						0	1	0	1	0	1	0	1
Bits	b_4 ↓	b_3 ↓	b_2 ↓	b_1 ↓	Column → Row ↓	0	1	2	3	4	5	6	7
	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
	0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
	0	0	1	0	2	STX	DC2	"	2	B	R	b	r
	0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
	0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
	0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
	0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
	0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
	1	0	0	0	8	BS	CAN	(	8	H	X	h	x
	1	0	0	1	9	HT	EM	)	9	I	Y	i	y
	1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
	1	0	1	1	11	VT	ESC	+	;	K	[	k	{
	1	1	0	0	12	FF	FC	,	<	L	\	l	|
	1	1	0	1	13	CR	GS	-	=	M	]	m	}
	1	1	1	0	14	SO	RS	.	>	N	^	n	~
	1	1	1	1	15	SI	US	/	?	O	_	o	DEL

ASCII- (American Standard Code for Information Interchange)

- Universally accepted alphanumeric code.
- Used in most computers and other electronic equipments. Most computer keyboards are standardized with ASCII.
- When a key is pressed, its corresponding ASCII code is generated which goes to the computer.
- Contains 128 characters and symbols.
- Since $128 = 2^7$ hence we need 7 bits to write 128 characters. Therefore ASCII is a 7 bit code.
- Can be represented in 8 bits by considering MSB = 0 always.
- Hence we have ASCII codes from 0000 0000 to 0111 1111 in binary or from 00 to 7F in hexadecimal.
- The first 32 characters are non-graphic control commands (never displayed or printed) eg., null, escape
- The remaining characters are graphic symbols (can be displayed and printed). This includes alphabets (capital and small), punctuation signs and commonly used symbols.
- So ASCII code consists of 94 printable characters, 32 non printable control commands and “Space” and “Delete” characters = 128 characters

Encode the following in ASCII Code:

1. We the people

W	1010111
e	1100101
	0100000
t	1110100
h	1101000
e	1100101
	0100000
P	1010000
e	1100101
o	1101111
p	1100001
l	1101100
e	1100101

EBCDIC-(Extended Binary Coded Decimal Interchange Code)

- 8-bit code.
- Total 256 characters are possible, however all are not used.
- There is no parity bit used to check error in this code set.
- The main **difference between ASCII** and **EBCDIC** is that the **ASCII** uses seven bits to represent a character while the **EBCDIC** uses eight bits to represent a character. It is easier for the computer to process numbers. On the other hand, **EBCDIC** is mainly used for IBM based systems. It represents 256 **characters.**
- Digits are coded F0 through F9 in EBCDIC