COMPUTER CODES

Application of Information and Communication Technologies

Dr. Muhammad Abdullah

Faculty of Computing and Information Technology (FCIT)
University of the Punjab, Lahore, Pakistan.

Data Types

- **Numeric Data** consists of only numbers 0, 1, 2, ..., 9
- **Alphabetic Data** consists of only the letters A, B, C, ..., Z, in both uppercase and lowercase, and blank character
- **Alphanumeric Data** is a string of symbols where a symbol may be one of the letters A, B, C, ..., Z, in either uppercase or lowercase, or one of the digits 0, 1, 2, ..., 9, or a special character, such as + - * / , . ( ) = etc.

Computer Codes

- Computer codes are used for internal representation of data in computers
- As computers use binary numbers for internal data representation, computer codes use binary coding schemes
- In binary coding, every symbol that appears in the data is represented by a group of bits
- The group of bits used to represent a symbol is called a **byte**

Computer Codes

- As most modern coding schemes use 8 bits to represent a symbol, the term byte is often used to mean a group of 8 bits
- Commonly used computer codes are BCD, EBCDIC, and ASCII

Binary Codes

1. Weighted Codes (8421, 5421)
2. Non Weighted codes (Excess-3 , BCD, Gray/Reflected codes)
3. Error Processing Codes (PARITY Bits, Hamming Codes)
4. Alphanumeric Codes (ASCII, EBCDIC)

1: Weighted Codes:

Weighted binary *codes* are those binary *codes* which obey the positional weight principle. Each position of the number represents a specific weight.

5 4 2 1		8 4 2 1
0 0 0 0	0	0
0 1 0 1	5	5
0 1 0 0	4	4
0 0 1 0	2	2
0 0 0 1	1	1
0 1 1 1	7	7
1 1 1 1	12	15

2: Non-Weighted Codes:

2.1: Excess-3 is a non-weighted coding method. With excess-3, we add 3 to a decimal number before converting it to binary.

Example:
$(0001)_2 = (0100)_{Excess-3}$
$(0010)_2 = (0101)_{Excess-3}$

2.2: BCD (Binary Coded Decimals) is a non-weighted coding method. Individual decimal digits are converted into equivalent binary bits.

Example:
$(321)_2 = (0011\ 0010\ 0000)_{BCD}$
$(000)_2 = (0000\ 0000\ 0000)_{BCD}$
$(80)_2 = (1000\ 0000)_{BCD}$
$(00)_2 = (0000\ 0000)_{BCD}$

2: Non-Weighted Codes:

2.3: Gray Codes: It is also called as Reflected Binary codes. It is generated via getting mirror image of given data.

Example : 4 bit Gray codes.

0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

2: Code Conversion:

Converting Binary Codes to Gray codes

Method:

1. Copy MSB of Binary code to MSB of Gray code.
2. Add MSB of Binary with Next MSB of Binary to get next Gray code.
3. Discard the carry
4. Repeat the same process till we get the LSB

Example: Convert (0001)2 into Gray code.

0 0 0 1

0+0= 0 0+0= 0 0+1= 1

0 0 0 1

2: Code Conversion:

Converting Gray Codes to Binary codes

Method:

1. Copy MSB of Gray code to MSB of Binary code.
2. If next bit of Gray code is "1", then invert the present binary bit as next bit.
3. If next bit of Gray code is "0" then copy the present binary bit as next bit.

Example: Convert (1011) gray into Binary code.

1	0	1	1
↓	↓	↓	↓
1	1	0	1

2: Code Conversion:

Converting Binary Codes to Gray codes

Method:

1. Copy MSB of Binary code to MSB of Gray code.
2. Add MSB of Binary with Next MSB of Binary to get next Gray code.
3. Discard the carry
4. Repeat the same process till we get the LSB

Example: Convert (1011)2 into Gray code.

1 0 1 1

1+0= 1 0+1= 1 1+1= 0

1 1 1 0

EBCDIC

- EBCDIC stands for **E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode
- It uses 8 bits to represent a symbol
- It can represent 256 (2^8) different characters

Coding of Alphabetic and Numeric Characters in EBCDIC

Char	EBCDIC Code		Hex
	Digit	Zone	
A	1100	0001	C1
B	1100	0010	C2
C	1100	0011	C3
D	1100	0100	C4
E	1100	0101	C5
F	1100	0110	C6
G	1100	0111	C7
H	1100	1000	C8
I	1100	1001	C9
J	1101	0001	D1
K	1101	0010	D2
L	1101	0011	D3
M	1101	0100	D4

Char	EBCDIC Code		Hex
	Digit	Zone	
N	1101	0101	D5
O	1101	0110	D6
P	1101	0111	D7
Q	1101	1000	D8
R	1101	1001	D9
S	1110	0010	E2
T	1110	0011	E3
U	1110	0100	E4
V	1110	0101	E5
W	1110	0110	E6
X	1110	0111	E7
Y	1110	1000	E8
Z	1110	1001	E9

Coding of Alphabetic and Numeric Characters in EBCDIC

Character	EBCDIC Code		Hexadecimal Equivalent
	Digit	Zone	
0	1111	0000	F0
1	1111	0001	F1
2	1111	0010	F2
3	1111	0011	F3
4	1111	0100	F4
5	1111	0101	F5
6	1111	0110	F6
7	1111	0111	F7
8	1111	1000	F8
9	1111	1001	F9

EBCDIC Coding Example

Example

Using binary notation, write EBCDIC coding for the word BIT. How many bytes are required for this representation?

Solution:

B = 1100 0010 in EBCDIC binary notation
I = 1100 1001 in EBCDIC binary notation
T = 1110 0011 in EBCDIC binary notation

Hence, EBCDIC coding for the word BIT in binary notation will be

$\underbrace{11000010}_{B} \quad \underbrace{11001001}_{I} \quad \underbrace{11100011}_{T}$

3 bytes will be required for this representation because each letter requires 1 byte (or 8 bits)

ASCII

- ASCII stands for **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.
- ASCII is of two types – ASCII-7 and ASCII-8
- ASCII-7 uses 7 bits to represent a symbol and can represent 128 (2^7) different characters
- ASCII-8 uses 8 bits to represent a symbol and can represent 256 (2^8) different characters
- First 128 characters in ASCII-7 and ASCII-8 are same

Coding of Alphabetic and Numeric Characters in ASCII

Character	ASCII-7 / ASCII-8		Hexadecimal Equivalent
	Zone	Digit	
0	0011	0000	30
1	0011	0001	31
2	0011	0010	32
3	0011	0011	33
4	0011	0100	34
5	0011	0101	35
6	0011	0110	36
7	0011	0111	37
8	0011	1000	38
9	0011	1001	39

Coding of Alphabetic and Numeric Characters in ASCII

Character	ASCII-7 / ASCII-8		Hexadecimal Equivalent
	Zone	Digit	
A	0100	0001	41
B	0100	0010	42
C	0100	0011	43
D	0100	0100	44
E	0100	0101	45
F	0100	0110	46
G	0100	0111	47
H	0100	1000	48
I	0100	1001	49
J	0100	1010	4A
K	0100	1011	4B
L	0100	1100	4C
M	0100	1101	4D

Coding of Alphabetic and Numeric Characters in ASCII

Character	ASCII-7 / ASCII-8		Hexadecimal Equivalent
	Zone	Digit	
N	0100	1110	4E
O	0100	1111	4F
P	0101	0000	50
Q	0101	0001	51
R	0101	0010	52
S	0101	0011	53
T	0101	0100	54
U	0101	0101	55
V	0101	0110	56
W	0101	0111	57
X	0101	1000	58
Y	0101	1001	59
Z	0101	1010	5A

ASCII Table

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

ASCII-7 Coding Scheme

Example

Write binary coding for the word BOY in ASCII-7. How many bytes are required for this representation?

Solution:

B = 1000010 in ASCII-7 binary notation
O = 1001111 in ASCII-7 binary notation
Y = 1011001 in ASCII-7 binary notation

Hence, binary coding for the word BOY in ASCII-7 will be

$$\underset{B}{\underline{1000010}} \quad \underset{O}{\underline{1001111}} \quad \underset{Y}{\underline{1011001}}$$

Since each character in ASCII-7 requires one byte for its representation and there are 3 characters in the word BOY, 3 bytes will be required for this representation

ASCII-8 Coding Scheme

Example

Write binary coding for the word SKY in ASCII-8. How many bytes are required for this representation?

Solution:

S = 01010011 in ASCII-8 binary notation
K = 01001011 in ASCII-8 binary notation
Y = 01011001 in ASCII-8 binary notation

Hence, binary coding for the word SKY in ASCII-8 will be

$$\underbrace{01010011}_{S} \quad \underbrace{01001011}_{K} \quad \underbrace{01011001}_{Y}$$

Since each character in ASCII-8 requires one byte for its representation and there are 3 characters in the word SKY, 3 bytes will be required for this representation

Unicode

- **Why Unicode:**
  - No single encoding system supports all languages
  - Different encoding systems conflict
- **Unicode features:**
  - Provides a consistent way of encoding multilingual plain text
  - Defines codes for characters used in all major languages of the world
  - Defines codes for special characters, mathematical symbols, technical symbols, and diacritics

Unicode

- **Unicode features (continued):**
  - Capacity to encode as many as a million characters
  - Assigns each character a unique numeric value and name
  - Reserves a part of the code space for private use
  - Affords simplicity and consistency of ASCII, even corresponding characters have same code
  - Specifies an algorithm for the presentation of text with bi-directional behavior
- **Encoding Forms**
  - UTF-8, UTF-16, UTF-32

Collating Sequence

- Collating sequence defines the assigned ordering among the characters used by a computer
- Collating sequence may vary, depending on the type of computer code used by a particular computer
- In most computers, collating sequences follow the following rules:

  1. Letters are considered in alphabetic order
     $(A < B < C \ldots < Z)$
  2. Digits are considered in numeric order
     $(0 < 1 < 2 \ldots < 9)$

Sorting in EBCDIC

Example

Suppose a computer uses EBCDIC as its internal representation of characters. In which order will this computer sort the strings 23, A1, 1A?

Solution:

In EBCDIC, numeric characters are treated to be greater than alphabetic characters. Hence, in the said computer, numeric characters will be placed after alphabetic characters and the given string will be treated as:

A1 < 1A < 23

Therefore, the sorted sequence will be: A1, 1A, 23.

Sorting in ASCII

Example

Suppose a computer uses ASCII for its internal representation of characters. In which order will this computer sort the strings 23, A1, 1A, a2, 2a, aA, and Aa?

Solution:

In ASCII, numeric characters are treated to be less than alphabetic characters. Hence, in the said computer, numeric characters will be placed before alphabetic characters and the given string will be treated as:

1A < 23 < 2a < A1 < Aa < a2 < aA

Therefore, the sorted sequence will be: 1A, 23, 2a, A1, Aa, a2, and aA