Lecture 5
Codes

Binary codes:- BCD, GRAY, EBCDIC, ASCII

- It is the symbolic representation of discrete information which may be represented in the form of numeric, alphabets & special characters.
- In Digital Electronics, the binary digits 0 & 1 are used to represent these symbols & are arranged according to the rules of specific code.
- Infect, a binary code is a group of n-bits that can represent distinct symbols.
- The interpretation of the binary information is possible only if the code in which this information is available is known.

Binary codes:- BCD, GRAY, EBCDIC, ASCII

(A) Straight Binary Code

Used to represent numbers using natural (or straight) binary form.

1.Natural BCD Code

In this code, decimal 0 through 9 are represented by their natural binary equivalents using four bits and each decimal digit is represented by this four bit code individually. This code is also known as 8-4-2-1 code where 8, 4, 2, & 1 are the weights of the four bits of the binary code of each decimal digit to the straight binary system.

2.EXCESS-3 Codes

This is another form of BCD code, in which each decimal is coded into a 4-bit binary code. The code for each decimal digit is obtained by adding decimal 3 to the natural BCD code of the digit. For ex- decimal 2 is coded as 0010+0011=0101 in Excess-3 code.

Binary codes:- BCD, GRAY, EBCDIC, ASCII

3. Gray Code

It is a very useful code in which a decimal number is represented in binary form in such a way so that each gray code number differs from preceding & the succeeding number by a single bit. For ex- the gray code for decimal number 5 is 0111 & for 6 is 0101. These two codes differ by only one bit position (third from left). This code is used extensively for the shaft encoders because of this property. These can be constructed by using the property given below:-

- A 1-bit Gray code has two code words 0 & 1 representing decimal numbers 0 & 1.
- An n-bit (n≥2) Gray code will have first 2^{n-1} with the leading 0 appended.
- The last 2^{n-1} Gray codes with the leading 1 appended, will be equal to the Gray code words of an (n-1)-bit Gray code, written in reverse order (assuming a mirror 2^{n-1} placed between first & last 2^{n-1} Gray codes) with a leading 1 appended.

Classification of Codes

1.Weighted & Non-weighted Codes
2.Self Complimentary & Sequential Codes
3. Alphanumeric codes
4. Error detection & Error correction Codes

1. ***Weighted Codes***

The main characteristic of weighted code is that each binary digit is assigned a specific weight. The common example of weighted code is BCD code or 8421 code in which the weights of different bits are 1, 2, 4 & 8.

Non-Weighted Code

These don’t follow the principle of positional weighting system i.e. each position within the number doesn’t follow or have any fixed weight. For ex- Excess-3, Gray code.

2. ***Sequential Code***

These are those codes in which each succeeding code is 1 binary number greater than the preceding code. This property is used for mathematical manipulation of data. For ex:- BCD
And Excess-3 Code.

3.Self-complementary Code

- A code is said to be self-complementary if the code for 9's complement of N i.e. 9-N can be obtained by interchanging all 0s and 1s.
- Decimal 9 is the complement of code for 0, 8 for 1, 7 for 2 and so on.
- For a code to be self complementing, the sum of all its weights must be 9. digit.8421 and 5421 codes are not self complementing codes whereas 5211,2421,3321, 4321 are self complementing.
- In general, a code is self-complementary if we produce a code by taking the first complement of the digit which is same as 9's complement of the number.

- **Cyclic codes**: Cyclic codes are those in which each successive code word differs from the preceding one in only one bit position.
- They are also called unit distance codes
- Example: gray code

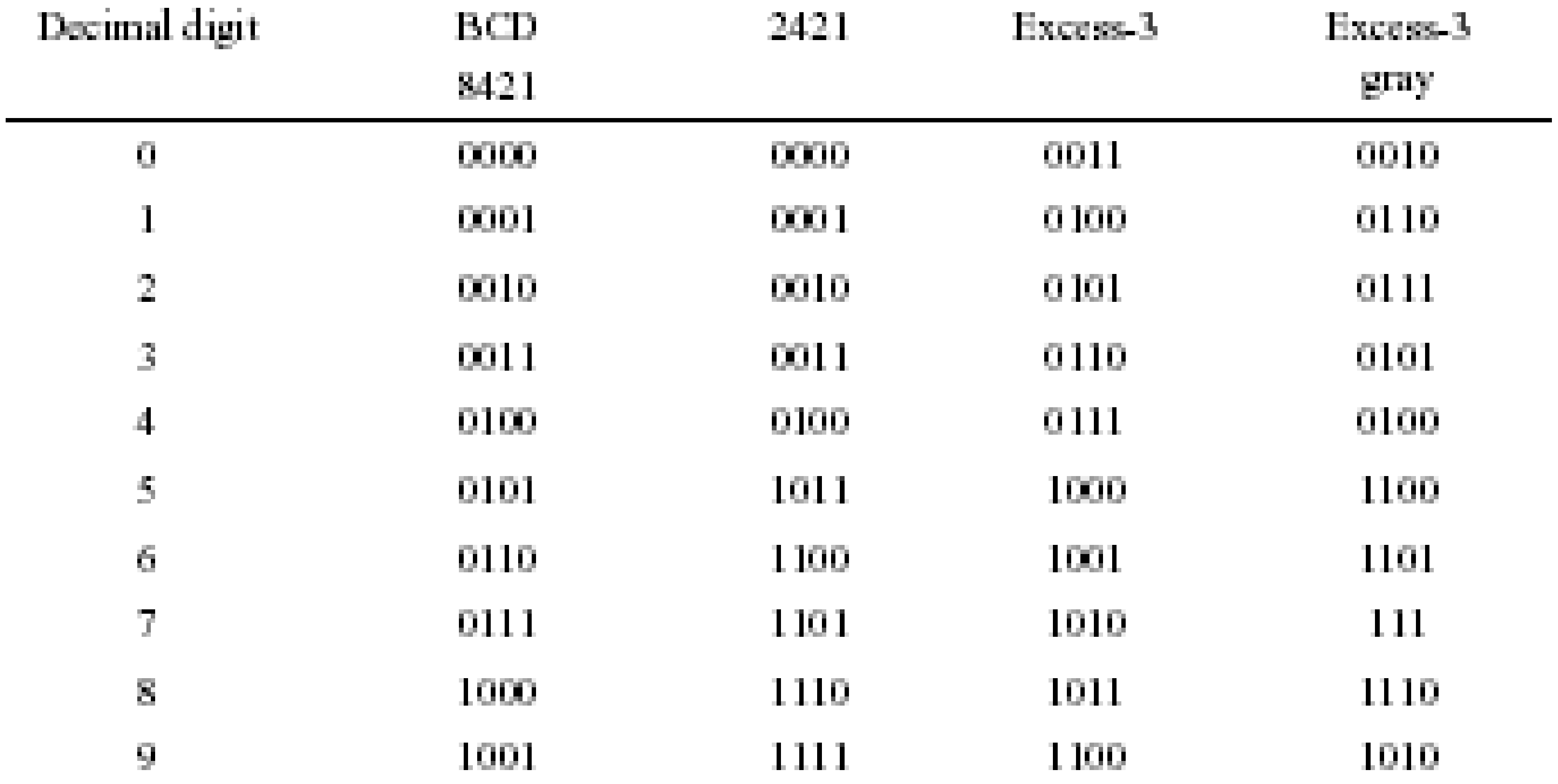
Reflective Code:

Example : Gray code.

Alphanumeric Codes

- **Apart from numeric data, a computer system may process some alphanumeric data just like the employees' names, address as well as some special characters. An Alphanumeric data generally consist of sequence of characters where a character is any one of the following:-**
- **Letters or alphabets**
- **Digits 0-9**
- **Special characters(+,-,π)**
- **In the computer system, each character is stored in some code form depending upon the coding scheme. The character may take 6, 7, or 8 bits. There are number of codes which are used for some specific application i.e. ASCII Code, EBCDIC Code, UNIT Code, etc.**

VARIOUS DECIMAL CODES



Decimal digit	BCD 8421	2421	Excess-3	Excess-3 gray
0	0000	0000	0011	0010
1	0001	0001	0100	0110
2	0010	0010	0101	0111
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1100
6	0110	1100	1001	1101
7	0111	1101	1010	111
8	1000	1110	1011	1110
9	1001	1111	1100	1010

Importance of Codes

1. The code refers to encryption system.

2. Computer and other digital circuits process data in binary format.

3. Various binary codes are used to represent data.

4. The interpretation of data is only possible if the code in which the information is available is known.

Binary Coded Decimal (BCD)

- Would it be easy for you if you can replace a decimal number with an individual binary code?
  - Such as 0001 1001 = 19_{10}
- The 8421 code is a type of BCD to do that.
- BCD code provides an excellent interface to binary systems:
  - Keypad inputs
  - Digital readouts

Binary Coded Decimal (BCD)

- Used to represent the decimal digits 0 - 9.
- 4 bits are used.
- Each bit position has a weight associated with it (***weighted code***).
- Weights are: 8, 4, 2, and 1 from MSB to LSB (called 8-4-2-1 code).
- BCD Codes:

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

- Used to encode numbers for output to numerical displays
- Used in processors that perform decimal arithmetic.
- ***Example***: $(9750)_{10} = (\underbrace{1001}_{9}\ \underbrace{0111}_{7}\ \underbrace{0101}_{5}\ \underbrace{0000}_{0})_{BCD}$

Binary Coded Decimal

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

Note: 1010, 1011, 1100, 1101, 1110, and 1111 are **INVALID CODE!**

Let's crack these…

ex1: Decimal to BCD

(a) 35

(b) 98

(c) 170

(d) 2469

ex2: BCD-to-Decimal

(a) 10000110

(b) 001101010001

(c) 1001010001110000

BCD Addition

- BCD is a numerical code and <u>can be used</u> in arithmetic operations. Here is how to add two BCD numbers:
  - Add the two BCD numbers, using the rules for basic binary addition.
  - If a 4-bit sum is equal to or less than 9, it is a valid BCD number.
  - If a 4-bit sum > 9, or if a carry out of the 4-bit group is generated it is an invalid result. Add 6 (0110) to a 4-bit sum in order to skip the six invalid states and return the code to 8421. If a carry results when 6 is added, simply add the carry to the next 4-bit group.

BCD Addition

- Try these:

  ex: Add the following numbers

  （a） 0011+0100

  （b） 00100011 ＋ 00010101

  （c） 10000110 ＋ 00010011

  （d） 010001010000 ＋ 010000010111

  （e） 1001 ＋ 0100

  （f） 1001 ＋ 1001

  （g） 00010110 ＋ 00010101

  （h） 01100111 ＋ 01010011

BCD Subtraction

- Here is how to subtract two BCD numbers:
  - Subtract the two BCD numbers, using the rules for basic binary subtraction.
  - If there is no borrow from the next higher group, no correction is required.
  - If there is borrow from the next group, then (0110) is subtracted from the difference term of this group.
  - Example 38 – 15

```
 38      0011  1000                (38 in BCD)
 15      0001   0101                (15 in BCD)
– _________________________
–              0010   0011     (No borrow, so it is a correct answer)
```

BCD Subtraction

- Try these:

  ex: Add the following numbers

  (a) 0011−0100

  (b) 00100011 − 00010101

  (c) 10000110 − 00010011

  (d) 010001010000 − 010000010111

  (e) 1001 − 0100

  (f) 1001 − 1001

  (g) 00010110 − 00010101

  (h) 01100111 − 01010011

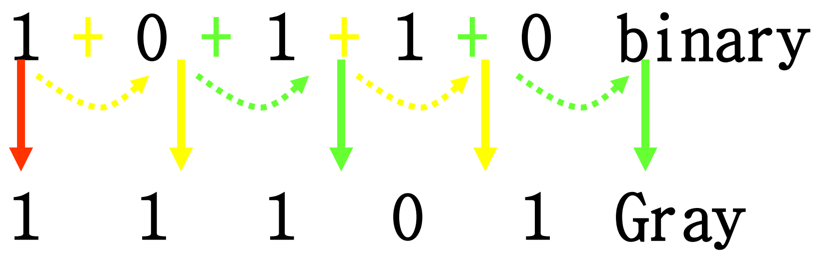
The Gray Code

- The Gray code is unweighted and is not an arithmetic code.
  - There are no specific weights assigned to the bit positions.
- Important: the Gray code exhibits only a single bit change from one code word to the next in sequence.
  - This property is important in many applications, such as shaft position encoders.

The Gray Code

- Binary-to-Gray code conversion
  - The MSB in the Gray code is the same as corresponding MSB in the binary number.
  - Going from left to right, add each adjacent pair of binary code bits to get the next Gray code bit. <u>Discard carries</u>.

ex: convert 10110_2 to Gray code

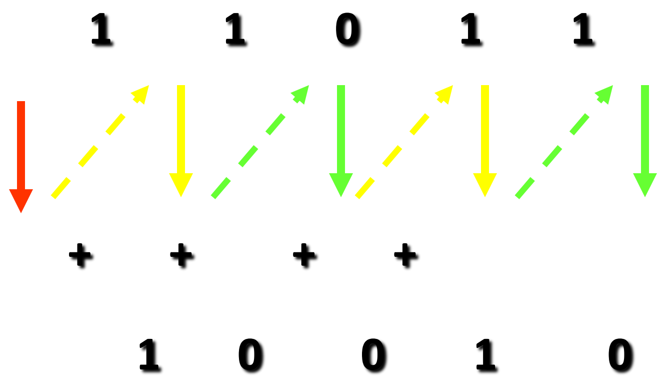


The Gray Code

- Gray-to-Binary Conversion
  - The MSB in the binary code is the same as the corresponding bit in the Gray code.
  - Add each binary code bit generated to the Gray code bit in the next adjacent position. Discard carries.

ex: convert the Gray code word 11011 to binary

Gray Binary

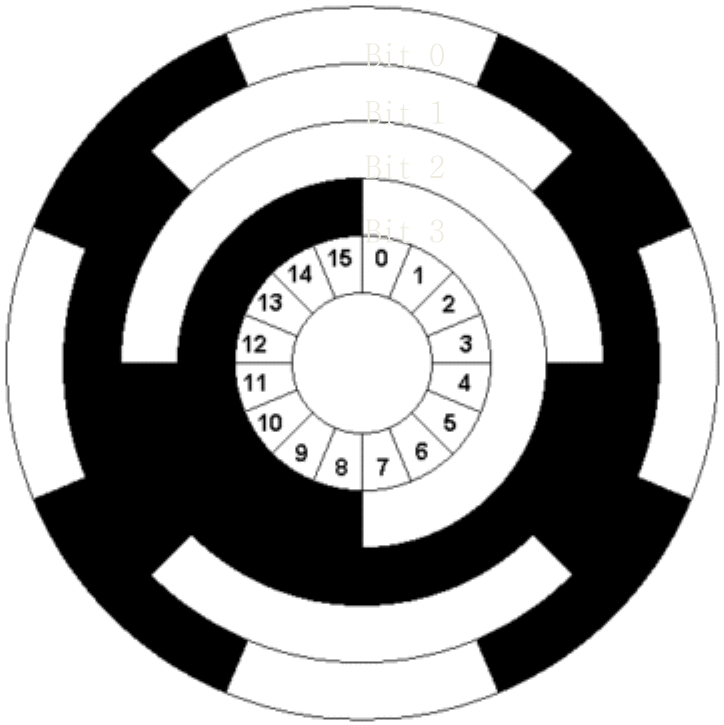


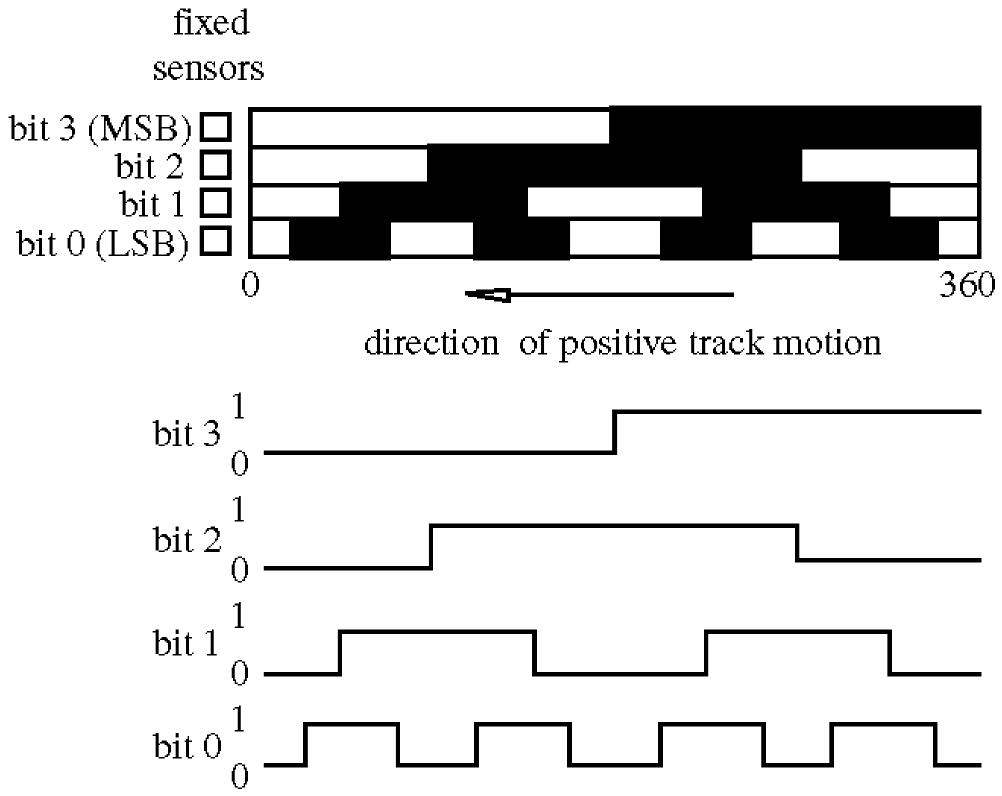
The Gray Code

Decima l	Binary	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

Decimal	Binary	Gray Code
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

The Gray Code - Application





http://www.mipraso.de/enzyklopaedie/g/gray-code-scheibe.gif

http://www.engr.colostate.edu/~dga/mechatronics/figures/9-11.gif

Excess 3 Code

- A BCD Code formed by adding 3 (0011) to its true four bit binary value.
- Excess 3 is a self complementing code. If the bits of the Excess-3 digit are inverted, they yield the 9's complement of the decimal equivalent.
- Excess-3 code is useful for performing decimal arithmetic digitally.

Excess 3 Examples

- 3 = 0011 + 0011 = 0110 = 6 in E-3.
- 1 = 0001 + 0011 = 0100 = 4 in E-3
- If we complement 1's = 1011 in E-3 this is the code for an 8.
- 9s Complement of 1(0100) = (9 - 1) = 8 Self Complement

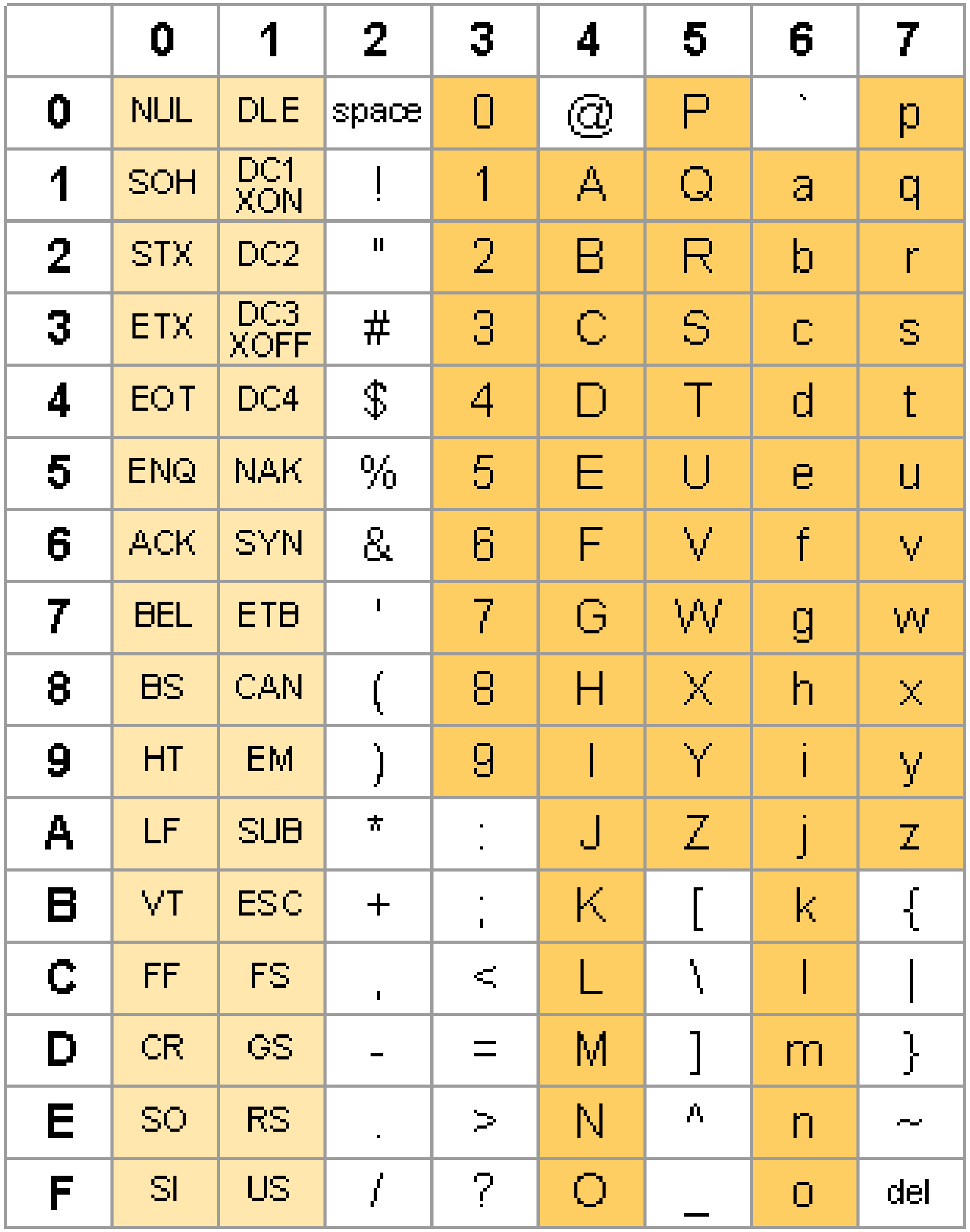
Alphanumeric Codes

- Represent numbers and alphabetic characters.
  - Also represent other characters such as symbols and various instructions necessary for conveying information.
- The ASCII is the most common alphanumeric code.
  - ASCII = American Standard Code for Information Interchange

ASCII

- ASCII has 128 characters and symbols represented by a 7-bit binary code.
  - It can be considered an 8-bit code with the MSB always 0. (00h-7Fh)
    - 00h-1Fh (the first 32) – control characters
    - 20h-7Fh – graphics symbols (can be printed or displayed)

ASCII Table



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

http://ascii-table.com/img/table.gif

Extended ASCII

- There are an additional 128 characters that were adopted by IBM for use in their PCs. It’s popular and is used in applications other than PCs → unofficial standard.
  - The extended ASCII characters are represented by an 8-bit code series from 80h-FFh

Extended ASCII Table

	8	9	A	B	C	D	E	F
0	Ç	É	á	░	└	╨	α	≡
1	ü	æ	í	▒	┴	╤	ß	±
2	é	Æ	ó	▓	┬	╥	Γ	≥
3	â	ô	ú	│	├	╙	π	≤
4	ä	ö	ñ	┤	─	╘	Σ	⌠
5	à	ò	Ñ	╡	┼	╒	σ	⌡
6	å	û	ª	╢	╞	╓	µ	÷
7	ç	ù	º	╖	╟	╫	τ	≈
8	ê	ÿ	¿	╕	╚	╪	Φ	°
9	ë	Ö	⌐	╣	╔	┘	Θ	∙
A	è	Ü	¬	║	╩	┌	Ω	·
B	ï	¢	½	╗	╦	█	δ	√
C	î	£	¼	╝	╠	▄	∞	ⁿ
D	ì	¥	¡	╜	═	▌	φ	²
E	Ä	₧	«	╛	╬	▐	ε	■
F	Å	ƒ	»	┐	╧	▀	∩	

Unsigned Numbers BCD Addition

Use binary arithmetic to add the BCD digits:

```
  8        1000   Eight
 +5       +0101   Plus 5
 13        1101   is 13 (> 9)
```

```
  3
 +5
 ---
  8  OK (< 9)
```

If result is > 9, it must generate a carry and be corrected!

To correct the digit, add 0110 in the result.

```
  8              1000   Eight
 +5             +0101   Plus 5
 13              1101   13 ( is > 9)
                +0110   so add 6 (always, for results > 9)
      carry = 1 0011   giving 3 + carry
           0001 | 0011   Final answer (two digits)
```

We try to avoid subtraction!
Replacing it with addition!

The adder circuit utilizes the resulting carry bit by sending it as carry-in to the next digit

Add 2905_{BCD} to 1857_{BCD} showing carries and digit corrections.

```
  1010               1          0           1          0
  1857            0001       1000        0101       0111
 +2905          + 0010       1001        0000       0101
                  0100  4<9 10001  17>9  0110  6<9  1100  12>9
                        →+0        →+6         →+0        →+6
        Add 6 or 0
                + 0000       0110        0000       0110
  ______
  4762            0100 1 0111            0110 1 0010
```

Excess-3 Code

A BCD Code formed by adding 3 (0011) to its
true 4-bit binary value.
Excess-3 is a self-complementing code:
A negative code equivalent can be found by
inverting the binary bits of the positive code
Inverting the bits of the Excess-3 digit yields
9’s Complement of the decimal equivalent.

Example : Excess -3 code of decimal 4 is 0111. (0100 + 0011 = 0111)

(4) = 0111

(-4) = 1000   (inverting the bits) which is Excess -3 code of decimal 5.

It is 9’s complement of the decimal equivalent. (9 – 4 = 5)

Excess-3 Examples

- 3 = 0011 + 0011 = 0110 = 6 in E3.
- 1 = 0001 + 0011 = 0100 = 4 in E3.
- If we complement 1 = 1011 in E3, this is the code for an 8.
- 9’s Complement of 1 = (9 – 1) = 8 (SelfComplement)