Binary Codes:

BCD Code,
Excess-3 Code,
Self Complementary Code

Codes:

What comes to your mind when hear the word “Code”?

- In communications and information processing, **code** is a system of rules to convert information—such as a letter, word, sound, image, or gesture—into another form or representation, sometimes shortened or secret, for communication through a channel or storage in a medium.

Binary Codes:

Introduction

Binary Codes for Decimal Digits

Decimal Digit	2-out-of-5 Code
0	00011
1	00101
2	00110
3	01001
4	01010
5	01100
6	10001
7	10010
8	10100
9	11000

1-bit Error Detection

less Space

Binary Codes for Decimal Digits

Decimal Digit	
0	0000
1	0001
2	0011
3	0010
4	0110
5	1110
6	1010
7	1011
8	1001
9	1000

ASCII Code: binary code

C-prog.

Set = {A, -- Z, a, -- z,
0, -- 9, $, #, @, --
Control symbols}

128 elements

A = 100 0001
@ = 100 0000
a = 011 1001
NUL = 000 0000
DEL = 111 1111

Table 1.7

American Standard Code for Information Interchange (ASCII)

$b_7b_6 \cdots b_1$

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	–	=	M	]	m	}
1110	SO	RS	.	>	N	∧	n	~
1111	SI	US	/	?	O	–	o	DEL

l : 101 1100

w : 111 0111

TABLE 1-3 ASCII Code

Character	ASCII Code							Character	ASCII Code							Character	ASCII Code						
	A_6	A_5	A_4	A_3	A_2	A_1	A_0		A_6	A_5	A_4	A_3	A_2	A_1	A_0		A_6	A_5	A_4	A_3	A_2	A_1	A_0
space	0	1	0	0	0	0	0	@	1	0	0	0	0	0	0	'	1	1	0	0	0	0	0
!	0	1	0	0	0	0	1	A	1	0	0	0	0	0	1	a	1	1	0	0	0	0	1
"	0	1	0	0	0	1	0	B	1	0	0	0	0	1	0	b	1	1	0	0	0	1	0
#	0	1	0	0	0	1	1	C	1	0	0	0	0	1	1	c	1	1	0	0	0	1	1
$	0	1	0	0	1	0	0	D	1	0	0	0	1	0	0	d	1	1	0	0	1	0	0
%	0	1	0	0	1	0	1	E	1	0	0	0	1	0	1	e	1	1	0	0	1	0	1
&	0	1	0	0	1	1	0	F	1	0	0	0	1	1	0	f	1	1	0	0	1	1	0
'	0	1	0	0	1	1	1	G	1	0	0	0	1	1	1	g	1	1	0	0	1	1	1
(	0	1	0	1	0	0	0	H	1	0	0	1	0	0	0	h	1	1	0	1	0	0	0
)	0	1	0	1	0	0	1	I	1	0	0	1	0	0	1	i	1	1	0	1	0	0	1
*	0	1	0	1	0	1	0	J	1	0	0	1	0	1	0	j	1	1	0	1	0	1	0
+	0	1	0	1	0	1	1	K	1	0	0	1	0	1	1	k	1	1	0	1	0	1	1
,	0	1	0	1	1	0	0	L	1	0	0	1	1	0	0	l	1	1	0	1	1	0	0
–	0	1	0	1	1	0	1	M	1	0	0	1	1	0	1	m	1	1	0	1	1	0	1
.	0	1	0	1	1	1	0	N	1	0	0	1	1	1	0	n	1	1	0	1	1	1	0
/	0	1	0	1	1	1	1	O	1	0	0	1	1	1	1	o	1	1	0	1	1	1	1
0	0	1	1	0	0	0	0	P	1	0	1	0	0	0	0	p	1	1	1	0	0	0	0
1	0	1	1	0	0	0	1	Q	1	0	1	0	0	0	1	q	1	1	1	0	0	0	1
2	0	1	1	0	0	1	0	R	1	0	1	0	0	1	0	r	1	1	1	0	0	1	0
3	0	1	1	0	0	1	1	S	1	0	1	0	0	1	1	s	1	1	1	0	0	1	1
4	0	1	1	0	1	0	0	T	1	0	1	0	1	0	0	t	1	1	1	0	1	0	0
5	0	1	1	0	1	0	1	U	1	0	1	0	1	0	1	u	1	1	1	0	1	0	1
6	0	1	1	0	1	1	0	V	1	0	1	0	1	1	0	v	1	1	1	0	1	1	0
7	0	1	1	0	1	1	1	W	1	0	1	0	1	1	1	w	1	1	1	0	1	1	1
8	0	1	1	1	0	0	0	X	1	0	1	1	0	0	0	x	1	1	1	1	0	0	0
9	0	1	1	1	0	0	1	Y	1	0	1	1	0	0	1	y	1	1	1	1	0	0	1
:	0	1	1	1	0	1	0	Z	1	0	1	1	0	1	0	z	1	1	1	1	0	1	0
;	0	1	1	1	0	1	1	[	1	0	1	1	0	1	1	{	1	1	1	1	0	1	1
<	0	1	1	1	1	0	0	\	1	0	1	1	1	0	0	|	1	1	1	1	1	0	0
=	0	1	1	1	1	0	1	]	1	0	1	1	1	0	1	}	1	1	1	1	1	0	1
>	0	1	1	1	1	1	0	^	1	0	1	1	1	1	0	~	1	1	1	1	1	1	0
?	0	1	1	1	1	1	1	—	1	0	1	1	1	1	1	delete	1	1	1	1	1	1	1

only these we Remember

No need btw.

Dec	Hex	Char
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9

Dec	Hex	Char
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z

Dec	Hex	Char
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z

ASCII Code:

A —— Z

65 90

a —— z

97 122

0 —— 9

48 —— 57

Character	ASCII Code (binary Code)	Decimal Value	Hexa Decimal value
A	1000001	65	41
Y	1011001	89	59
y	1111001	121	79
0	0110000	48	30

[redacted]" The word "*Start*" is represented in ASCII code as follows:

1010011	1110100	1100001	1110010	1110100
S	t	a	r	t

Binary Numbers

binary Nuber System

- One can think of binary numbers as a kind of **code**:
  - In communications and information processing, **code** is a system of rules to convert information—such as a letter, word, sound, image, or gesture—into another form or representation, sometimes shortened or secret, for communication through a channel or storage in a medium.

binary Number → also a binary code

15 → 1111 ✓

30 → 11110

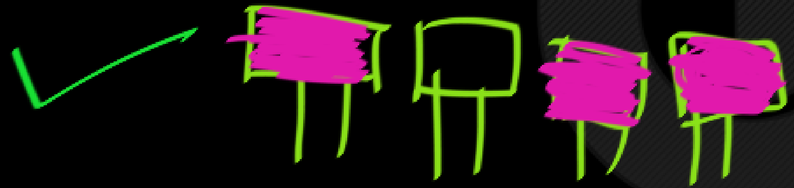
Binary Codes:

Pattern of 1's, 0's.

Why Binary(0,1) Codes?

Reason: Digital Circuits

$\boxed{1\ 0\ 1\ 1}$ — High level

 — lowest level

BINARY CODES

Digital systems use signals that have two distinct values and circuit elements that have two stable states. There is a direct analogy among binary signals, binary circuit elements, and binary digits. A binary number of *n* digits, for example, may be represented by *n* binary circuit elements, each having an output signal equivalent to 0 or 1. Digital systems represent and manipulate not only binary numbers, but also many other discrete elements of information. Any discrete element of information that is distinct among a group of quantities can be represented with a binary code (i.e., a pattern of 0's and 1's). The codes must be in binary because, in today's technology, only circuits that represent and manipulate patterns of 0's and 1's can be manufactured economically for use in computers. However, it must be realized that binary codes merely change the symbols, not the meaning of the elements of information that they represent. If we inspect the bits of a computer at random, we will find that most of the time they represent some type of coded information rather than binary numbers.

An n-bit binary code is a group of n bits that assumes up to 2^n distinct combinations of 1's and 0's, with each combination representing one element of the set that is being coded. A set of four elements can be coded with two bits, with each element assigned one of the following bit combinations: 00, 01, 10, 11. A set of eight elements requires a three-bit code and a set of 16 elements requires a four-bit code. The bit combination of an n-bit code is determined from the count in binary from 0 to $2^n - 1$. Each element must be assigned a unique binary bit combination, and no two elements can have the same value; otherwise, the code assignment will be ambiguous.

Although the *minimum* number of bits required to code 2^n distinct quantities is n, there is no *maximum* number of bits that may be used for a binary code. For example, the 10 decimal digits can be coded with 10 bits, and each decimal digit can be assigned a bit combination of nine 0's and a 1. In this particular binary code, the digit 6 is assigned the bit combination 0001000000.

Binary Codes:

Why study so many different type of Binary Codes when we already have Binary Numbers?

Binary Numbers:

16 → 10000

a → ?

@ → ?

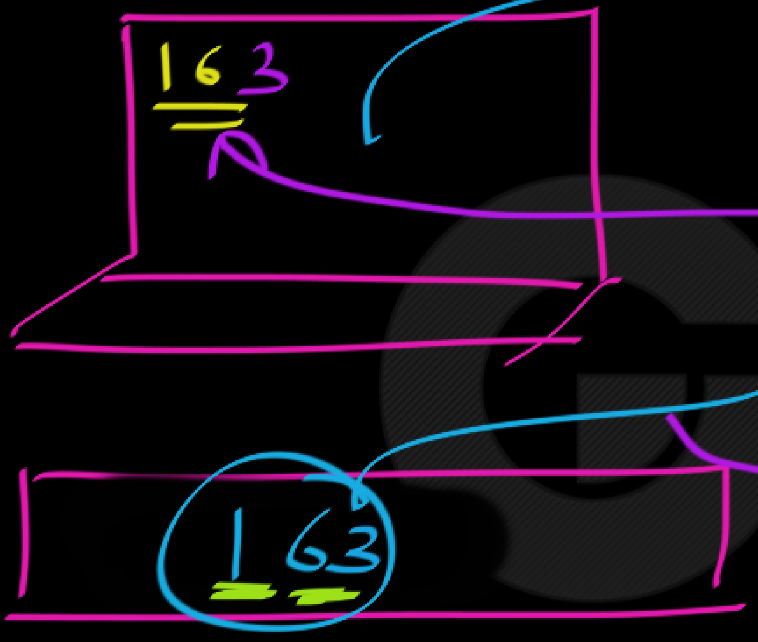
अ → ?

Reason 1.

- So how do we represent:
  - Numbers, letters, colors, etc. ?

Reason2:

interally CPU ⇒ binary Numbers



0001 0110 0011

keyboard → I/O Device

Input/output

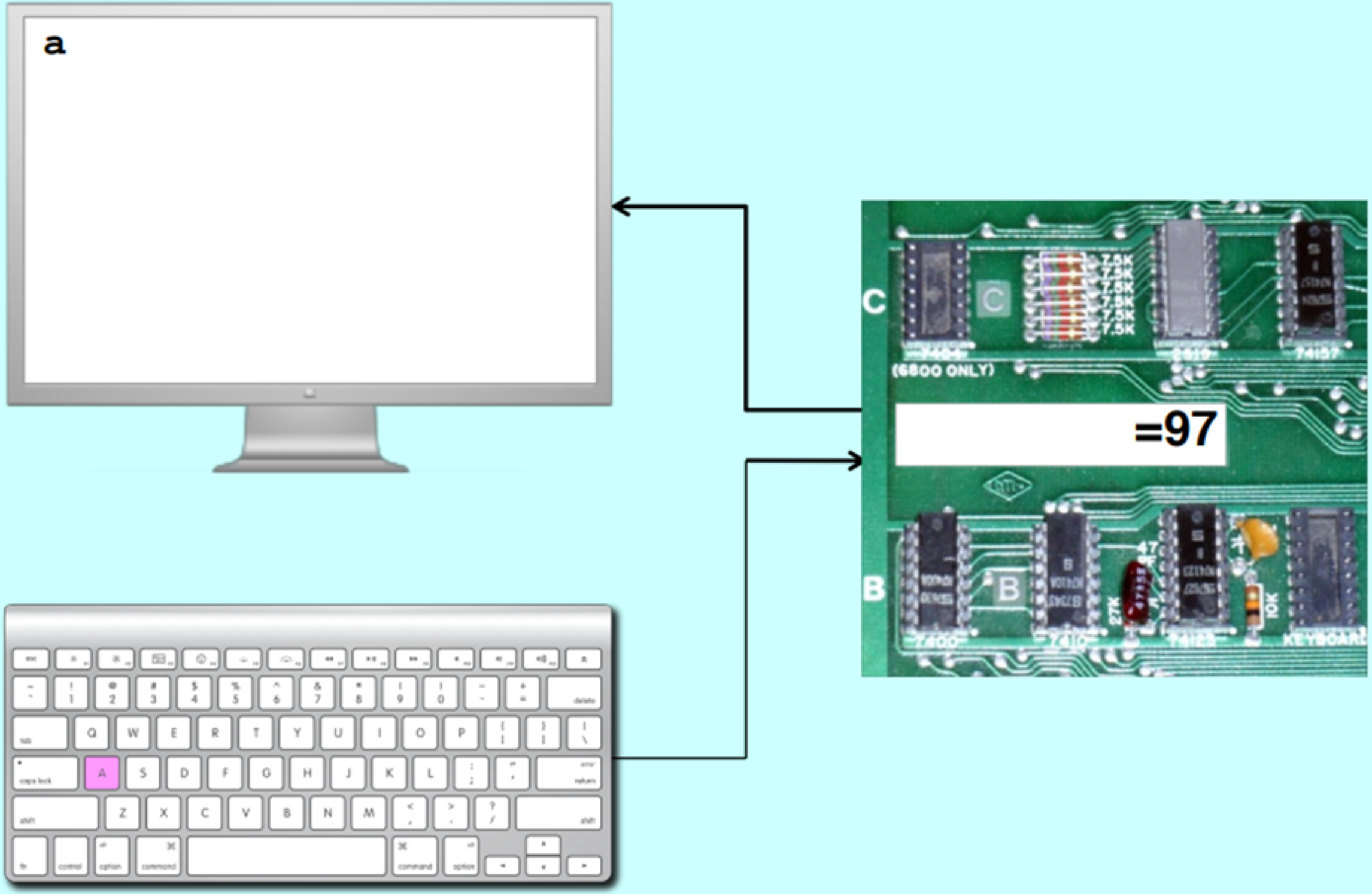
Binary Codes

Although most large computers work internally with binary numbers, the input-output equipment generally uses decimal numbers. Because most logic circuits only accept two-valued signals, the decimal numbers must be coded in terms of binary signals. In the simplest form of binary code, each decimal digit is replaced by its binary equivalent. For example, 937.25 is represented by

$$\begin{array}{ccccc c} 9 & 3 & 7 & . & 2 & 5 \\ 1001 & 0011 & 0111 & . & 0010 & 0101 \end{array}$$

This representation is referred to as binary-coded-decimal (BCD) or more explicitly as 8-4-2-1 BCD. Note that the result is quite different than that obtained by converting the number as a whole into binary. Because there are only ten decimal digits, 1010 through 1111 are not valid BCD codes.

Hardware Support for Characters



Reason 3 :

Applications / Specific Uses

Eg : Error Detection ;
Encryption ; - - - - - - -

Binary Codes:

1. "BCD" code

for Decimal Digits

BCD

Binary Coded Decimal

Binary Codes

Although most large computers work internally with binary numbers, the input-output equipment generally uses decimal numbers. Because most logic circuits only accept two-valued signals, the decimal numbers must be coded in terms of binary signals. In the simplest form of binary code, each decimal digit is replaced by its binary equivalent. For example, 937.25 is represented by

$$\overbrace{1001}^{9}\quad \overbrace{0011}^{3}\quad \overbrace{0111}^{7}\quad .\quad \overbrace{0010}^{2}\quad \overbrace{0101}^{5}$$

This representation is referred to as binary-coded-decimal (BCD) or more explicitly as 8-4-2-1 BCD. Note that the result is quite different than that obtained by converting the number as a whole into binary. Because there are only ten decimal digits, 1010 through 1111 are not valid BCD codes.

Table 1.4
Binary-Coded Decimal (BCD)

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

10 Decimal Digits

Data

23

206

Binary Number

10111

1 _ _ _ _ _ _ _

BCD

0010 0011

0010 0000 0110

a binary code

Binary-Coded Decimal Code

Although the binary number system is the most natural system for a computer because it is readily represented in today's electronic technology, most people are more accustomed to the decimal system. One way to resolve this difference is to convert decimal numbers to binary, perform all arithmetic calculations in binary, and then convert the binary results back to decimal. This method requires that we store decimal numbers in the computer so that they can be converted to binary. Since the computer can accept only binary values, we must represent the decimal digits by means of a code that contains 1's and 0's. It is also possible to perform the arithmetic operations directly on decimal numbers when they are stored in the computer in coded form.

A binary code will have some unassigned bit combinations if the number of elements in the set is not a multiple power of 2. The 10 decimal digits form such a set. A binary code that distinguishes among 10 elements must contain at least four bits, but 6 out of the 16 possible combinations remain unassigned. Different binary codes can be obtained by arranging four bits into 10 distinct combinations. The code most commonly used for the decimal digits is the straight binary assignment listed in Table 1.4. This scheme is called *binary-coded decimal* and is commonly referred to as BCD. Other decimal codes are possible and a few of them are presented later in this section.

Table 1.4
Binary-Coded Decimal (BCD)

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

Unused Combination
(Invalid BCD codes)

1010
1011
1100
1101
1110
1111

$(0011\ 1100)$ BCD

3

Invalid

Invalid BCD code / Combination

$(312)_{10} =$

$(0011\ 0001\ 0010)$ BCD

Table 1.4 gives the four-bit code for one decimal digit. A number with k decimal digits will require $4k$ bits in BCD. Decimal 396 is represented in BCD with 12 bits as 0011 1001 0110, with **each group of 4 bits representing one decimal digit.** A decimal number in BCD is the same as its equivalent binary number only when the number is between 0 and 9. A BCD number greater than 10 looks different from its equivalent binary number, even though both contain 1's and 0's. Moreover, **the binary combinations 1010 through 1111 are not used and have no meaning in BCD.** Consider decimal 185 and its corresponding value in BCD and binary:

$$(185)_{10} = \underbrace{(0001\ 1000\ 0101)_{BCD}}_{\text{BCD}} = \underbrace{(10111001)_2}_{\text{binary Number}}$$

Note: a BCD number is NOT a binary Number.

Eg: $(\underline{0010}\ \underline{1000})_{BCD} = (\underline{28})_{10}$

~~40~~

Decimal:

0, 1, 2, 3, - - - , 9

BCD

0000, 0001, 0010, - - - , 1001

The BCD value has 12 bits to encode the characters of the decimal value, but the equivalent binary number needs only 8 bits. It is obvious that the representation of a BCD number needs more bits than its equivalent binary value. However, there is an advantage in the use of decimal numbers, because computer input and output data are generated by people who use the decimal system.

It is important to realize that BCD numbers are decimal numbers and not binary numbers, although they use bits in their representation. The only difference between a decimal number and BCD is that decimals are written with the symbols 0, 1, 2, . . . , 9 and BCD numbers use the binary code 0000, 0001, 0010, . . . , 1001. The decimal value is exactly the same. Decimal 10 is represented in BCD with eight bits as 0001 0000 and decimal 15 as 0001 0101. The corresponding binary values are 1010 and 1111 and have only four bits.

BCD Code $\equiv$ 8421 Code

$(0\ 0\ 1\ 0)_{BCD} = (2)_{10}$

8 4 2 1

for any valid BCD 4-digit Code,

we can assign weight 8421.

$$(0\,1\,0\,1)_{BCD} = (5)_{10}$$

8 4 2 1

BCD

a binary code

for Decimal Digits

$\{0, --, 9\}$

Binary Codes:

2. Some Other codes for Decimals

2421 Code:

Decimal	2421 Coded
0	0000
1	0001
2	1000, 0010 (Preferred)
3	1001, 0011

by-default

2421 Code :

Decimal

4

2421 Coded

1010, 0100

1000 in 2421 $\Rightarrow$ 2 ✓

1 0 0 0
2 4 2 1

0010 in 2421 $\Rightarrow$ 2 ✓

0 0 1 0
2 4 2 1

2 in 2421 code

1000

0010 (Preferred)

both Correct

How to Decide which to prefer? (till y)

lower Decimal value as a binary number.

Self Complementary Code :

SCC

Decimal

2

9's Comp

7

Code

0010

1's

1101

BCD Code ⟹ Is SCC ? NO.

Decimal — BCD

9's Comp: 2 → 7

0010 → 0111 Not 1's comp

2421 Code :

self Complementary Code

has many Applications

Decimal

4

9's Comp 5

2421

0100

1011

1's Comp

2+2+1=5

Decimal	2421
0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

9's

1's comp

Other Decimal Codes

Binary codes for decimal digits require a minimum of four bits per digit. Many different codes can be formulated by arranging four bits into 10 distinct combinations. BCD and three other representative codes are shown in Table 1.5. Each code uses only 10 out of a possible 16 bit combinations that can be arranged with four bits. The other six unused combinations have no meaning and should be avoided.

BCD and the 2421 code are examples of weighted codes. In a weighted code, each bit position is assigned a weighting factor in such a way that each digit can be evaluated by adding the weights of all the 1's in the coded combination. The BCD code has weights of 8, 4, 2, and 1, which correspond to the power-of-two values of each bit. The bit assignment 0110, for example, is interpreted by the weights to represent decimal 6 because $8 \times 0 + 4 \times 1 + 2 \times 1 + 1 \times 0 = 6$. The bit combination 1101, when weighted by the respective digits 2421, gives the decimal equivalent of $2 \times 1 + 4 \times 1 + 2 \times 0 + 1 \times 1 = 7$. Note that some digits can be coded in two possible ways in the 2421 code. For instance, decimal 4 can be assigned to bit combination 0100 or 1010, since both combinations add up to a total weight of 4.

$\boxed{8, 4, -2, -1}$ Code $\Rightarrow$ SC Code

Decimal Digit	8, 4, -2, -1 Code
0	0000
1	0111
2	0110
3	0101
4	0100
5	1011

9's comp

1's comp

0 1 1 1 in 8, 4, −2, −1 Code

$8 \quad 4 \quad -2 \quad -1$

$= 4 + (-2) + (-1) = 1$ ✓

Table 1.5
Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Unused bit combi- nations	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

GO CLASSES

GO Classes

Table 1.5
Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Unused bit combi- nations	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

9's comp

1's comp

Excess-3 Code :

Decimal Digit	BCD Code		Excess-3 Code
5	0101	$\xrightarrow[+0011]{+3}$	1000
9	1001	$\xrightarrow[+0011]{+3}$	1100

Why Excess-3 when we Already have BCD ?

Reason: Self Comp. Code

The 2421 and the excess-3 codes are examples of self-complementing codes. Such codes have the property that the 9's complement of a decimal number is obtained directly by changing 1's to 0's and 0's to 1's (i.e., by complementing each bit in the pattern). For example, decimal 395 is represented in the excess-3 code as 0110 1100 1000. The 9's complement of 604 is represented as 1001 0011 0111, which is obtained simply by complementing each bit of the code (as with the 1's complement of binary numbers).

The excess-3 code has been used in some older computers because of its self-complementing property. **Excess-3 is an unweighted code in which each coded combination is obtained from the corresponding binary value plus 3.** Note that the BCD code is not self-complementing.

The 8, 4, −2, −1 code is an example of assigning both positive and negative weights to a decimal code. In this case, the bit combination 0110 is interpreted as decimal 2 and is calculated from $8 \times 0 + 4 \times 1 + (-2) \times 1 + (-1) \times 0 = 2$.

Binary Codes:

3. Self Complementing Binary Codes (scc)

Decimal Digit	XYZ Binary Code
a	0100
9-a	1011

9's Comp

1's Comp

then XYZ Code is SCC.

for Decimal Digits (0, 1, - -, 9)

Binary Codes:

Per Decimal Digit Codes

- BCD ≡ 8421 Code
- 2421 Code
- Excess-3 Code
- 8, 4, -2, -1 Code

	BCD	Excess-3	2421 code
15	0001 0101	0100 1000	0001 1011
201	0010 0000 0001	0101 0011 0100	0010 0000 0001

Weighted Code:

BCD = 8421

2421 ✓

8, 4, -2, -1 ✓

BCD+3

Excess-3 is NOT weighted code.

SCC

Excess-3

2421 ✓

8, 4, -2, -1 ✓

BCD+3

Excess-3 is NOT weighted code.

Non-weighted, scc ?

Excess-3 ✓

Weighted Code

abcd

SCC

$9a+b+c+d=9$

Not important