2b. Codes

- Students are required to do self-study for this topic.
- Essential concepts will be discussed in Tutorial 1.

Quick links to each section

Encoding

Numbers, letters or words are represented by a special group of symbols. A group of symbols is called a code.

For example, you may use the following binary code with your friend:

- 00: let's go eat lunch
- 01: let's go play basketball
- 10: let's go to the library
- 11: let's study for tomorrow's quiz

Both you and your friend must agree what each code word means for it to work.

Straight binary coding

In digital systems, numbers are probably the most common type of information that need to be represented.

It is very common to represent a numerical value in binary, i.e. base-2.

e.g. the decimal value 35 is simply represented as 100011 in binary. This is called straight binary coding or simply binary coding.

Note: 35_{10} = 2^5 + 2^1 + 2^0

There are other commonly used codes for representing numbers.

Binary-Coded-Decimal Code (BCD)

- Encode decimal numbers; combine some features of decimal and binary systems
- Each digit of a decimal number is represented by its 4-bit binary equivalent
- The legitimate digits are 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9
- Since the largest decimal digit is 9, 4 bits are required for each digit.

Decimal digit	BCD equivalent			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Notice that the following bit patterns are illegal in BCD code:

1010

1011

1100

1101

1110

1111

BCD code is used in digital machines whenever decimal information is either applied as inputs or displayed as outputs

Representation in BCD

E.g. Represent decimal 957 in BCD

- Decimal 9 = 1001 in BCD
- Decimal 5 = 0101 in BCD
- Decimal 7 = 0111 in BCD
- Thus decimal 957 = **1001 0101 0111** in **BCD**
- Contrast this with

  decimal 957 = **11 1011 1101** in **straight binary**

Characteristics of BCD

- Relative ease of conversion
- Consists of **groups of 4-bit codes** for decimal digits 0-9
- Important from hardware standpoint – logic circuits perform conversion to and from decimal digits, all digits can be converted simultaneously
- E.g. converting 957 to binary requires repeated division, but converting it to BCD does not.

BCD code is <u>not</u> used in

- **High speed computers**
  - it requires more bits than binary, and is therefore less efficient
  - E.g. decimal 3 in binary is 11, but decimal 3 in BCD is 0011
  - **arithmetic** processes represented in BCD code are more complicated and slower

Exercise

1. Convert 34_{10} to BCD

2. Convert 19.25_{10} to BCD

Answers:

0011 0100

0001 1001. 0010 0101

Gray code

- Belongs to a class of codes called **minimum-change** codes
- Only 1 bit in the code group changes when going from 1 step to the next
- **Unweighted** code: Bit positions do not have any specific weight (contrast with position-value numbers)
- Usually **cyclical**: the last codeword and the first codeword only has 1 bit difference

Decimal	4-bit Gray code			
0	0	0	0	**0**
1	0	0	**0**	1
2	0	0	1	**1**
3	0	**0**	1	0
4	0	1	1	**0**
5	0	1	**1**	1
6	0	1	0	**1**
7	**0**	1	0	0
8	1	1	0	**0**
9	1	1	**0**	1
10	1	1	1	**1**
11	1	**1**	1	0
12	1	0	1	**0**
13	1	0	**1**	1
14	1	0	0	**1**
15	**1**	0	0	0

Only 1 bit is changed from one value to the next

Example - Error occurring while using BCD

- what happens when a number increments from 1 to 2?

	BCD code			
Dec	**b3**	**b2**	**b1**	**b0**
1	**0**	**0**	**0**	**1**
2	**0**	**0**	**1**	**0**

Example - Error occurring while using BCD

- what happens when a number increments from 1 to 2?

Ideally, the bits b1 and b0 change at the same time:

Dec	b3	b2	b1	b0
1	0	0	0	1
↓			↓	↓
2	0	0	1	0

No error

- possible actual case, bit b0 changes before b1:

Dec	b3	b2	b1	b0
1	0	0	0	1
0	0	0	0	0
2	0	0	1	0

Transition error due to different speeds of bit change – **race condition**

Solution - Error occurring while using BCD

- Use Gray code instead and there will be no such problem since only 1 bit (b1) is changed when the number increments from 1 to 2

	Gray code			
Dec	**b3**	**b2**	**b1**	**b0**
1	**0**	**0**	**0**	**1**
			⇩	
2	**0**	**0**	**1**	**1**

Gray code is useful in situations where multiple bit change may lead to error.

Gray code is **not** suitable for **arithmetic operations**.

You may read section 2.11 of the textbook by Wakerly for details.

Alphanumeric Codes

- **Codes that represent**
  - alphabet (e.g. a, b, c, …, z)
  - punctuation marks
  - special characters and numbers
- **A complete set of alphanumeric code must include**
  - 26 lowercase letters (a – z)
  - 26 uppercase letters (A – Z)
  - 10 numeric digits (0 – 9)
  - 7 punctuation marks
  - 20 – 40 other characters such as +, -, /, <, #, %, …

ASCII Code

- Most widely used alphanumeric code
- 7-bit code, hence 128 (=2^7) possible code symbols
- There is also the 8-bit extended ASCII code
- Used for transferring alphanumeric data between digital devices
- Used in digital computers to store alphanumeric characters

Students are **NOT required** to memorize the ASCII table

American Standard Code for Information Interchange (ASCII)

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	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

b7 is MSB, b1 is LSB

Example:

Mp3 encoded into ASCII, will become

$$\underbrace{\mathbf{1001101}}_{\mathbf{M}}\underbrace{\mathbf{1110000}}_{\mathbf{p}}\underbrace{\mathbf{0110011}}_{\mathbf{3}}$$

Note: Lower case and upper case alphabets have different codes

Often times, hexadecimal digits are used to represent ASCII codes.

Example:

Character	**ASCII**	**expressed in Hex**
M	0100 1101	4D
p	0111 0000	70
3	0011 0011	33

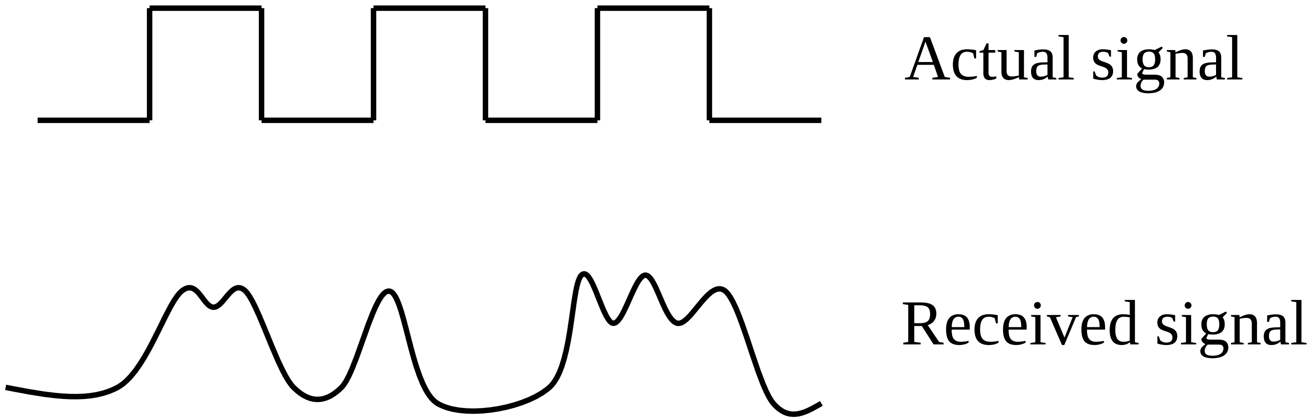
0 is padded to MSB before converting to Hex

Different ways to represent 14_{10}

- Straight binary: 1110
- Hexadecimal: E or e
- Octal: 16
- BCD: 0001 0100
- ASCII: 0110001 0110100
- Gray code: 1001
- Note the importance of knowing which representation is being used

Parity Method for Error Detection

- Transfer of binary data from one location to another can be corrupted by noise.



- **Result of error:**

  transmitted 0 becomes 1 at the receiver

  transmitted 1 becomes 0 at the receiver

- **e.g. 1010** wrongly received as **1011,** or

  1100 wrongly received as **1000**

- Multiple bit errors cannot be detected by this simple parity method. They are much less likely to happen than single bit errors.

Parity Bit

- An extra bit attached to a code group
- It forms part of the code being transmitted
- The parity bit is made either 0 or 1
- **Even parity** make total no. of '1' bits <u>even</u> BEFORE transmitting

$\boxed{0}$ 0 1 0 1 0 1 1

Parity bit

4 bits of 1 in data (4 is an even number)

- **Odd parity** make total no. of ‘1’ bits odd BEFORE transmitting

1   1 1 0 1 1 1 1

Parity bit

6 bits of 1 in data (7 is an odd number)

- Is able to detect single bit error only

- Receiver and transmitter must **agree** on odd/ even parity scheme

Examples:

7-bit data to be transmitted	8-bits are transmitted after adding parity bit	
	Odd parity	**Even parity**
1001000	11001000	01001000
0011100	00011100	10011100
0101110	10101110	00101110
1010111	01010111	11010111

Example: Limitation of parity method

- Transmitter and receiver agree on even parity system
- Data to be transmitted: 1010111
- Data transmitted with parity bit: <u>1</u>1010111
- Actual data received corrupted by noise: 11000111 – one bit error
- Receiver checks parity: odd
- Receiver correctly concludes data in error
- If actual data received: 01010110 – two bit error
- Receiver checks parity: even
- Receiver wrongly concludes data no error!

Commonly Used Prefixes

SI units

- k (kilo) = 10^3
- M (mega) = 10^6
- G (giga) = 10^9

JEDEC

- K (kilo) = 2^{10}
- M (mega) = 2^{20}
- G (giga) = 2^{30}
- T (tera) = 2^{40}

IEC

- Ki (kibi) = 2^{10}
- Mi (mebi) = 2^{20}
- Gi (gibi) = 2^{30}
- Ti (tebi) = 2^{40}

Binary prefix - Wikipedia, the free encyclopedia

Commonly Used Prefixes (cont)

Metric system

- m (milli) = 10^{-3}
- µ (micro) = 10^{-6}
- n (nano) = 10^{-9}
- p (pico) = 10^{-12}

Example:

- 0.1 µs = 100 ns = 10^{-7} second
- 200 mV = 0.2 V

About significant figures

The value of pi is 3.1415926…

It can also be written as

- 3.14159 (6 significant figures)
- 3.1416 (5 sf)
- 3.142 (4 sf)
- 3.14 (3 sf)
- 3.1 (2 sf)
- 3 (1 sf)