Week 04
Information Presentations

Today: Information presentations

- **Background: Precision**
- **Code**
- **Presentation**
  - Alphanumeric
  - Images
  - Audio

4.3 Độ chính xác và lỗi tràn số

- **Độ chính xác trong biểu diễn số nguyên**
- **Độ chính xác trong tính toán số thực dấu chấm động**

Lỗi tràn số nguyên

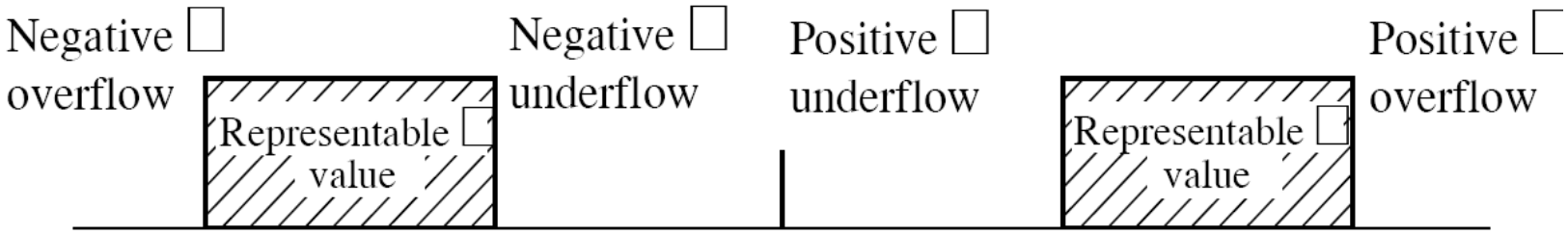
- **Lỗi tràn số:**
  - Hiện tượng xảy ra khi số cần biểu diễn vượt quá số bit cho trước để biểu diễn nó
  - Ví dụ
    - mẩu 4 bit cho biểu diễn bù 2 cho -6, -4
    - -6 → 1010, -4 → 1100
    - Kết quả phép cộng ở dạng bù 2 là 0110 là biểu diễn của +6, do đó kết quả bị sai.
  - Nguyên nhân: số lượng bit để biểu diễn quá ít

Lỗi tràn số nguyên

- **Khắc phục: tăng số lượng bit nhiều hơn**
  - **Mẫu 32 bit số dương lớn nhất +2.147.483.647**
- **<u>Tổng quát:</u>**
  - **Ở phép biểu diễn bù 1 hay bù 2 có n-bit**
    - **giá trị dương lớn nhất: $2^{n-1} - 1$**
    - **giá trị âm nhỏ nhất: -2^{n-1}**

Độ chính xác trong tính toán số thực dấu chấm động

- **Overflow: khi giá trị > giá trị tuyệt đối của khả năng biểu diễn lớn nhất.**
- **Underflow: 0 < giá trị < giá trị tuyệt đối của khả năng biểu diễn nhỏ nhất.**



Code

- BCD
- Gray
- ASCII
- Parity Method
- CRC
- ISO
- EBCDIC
- UNICODE

BCD

Binary coded decimal (BCD) is a **weighted code** that is commonly used in digital systems when it is necessary to show decimal numbers such as in clock displays.

The table illustrates the difference between straight binary and BCD. BCD represents each decimal digit with a 4-bit code. Notice that the codes 1010 through 1111 are not used in BCD.

Decimal	Binary	BCD
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

BCD

You can think of BCD in terms of column weights in groups of four bits. For an 8-bit BCD number, the column weights are: 80 40 20 10 8 4 2 1.

Question: What are the column weights for the BCD number 1000 0011 0101 1001?

Answer:

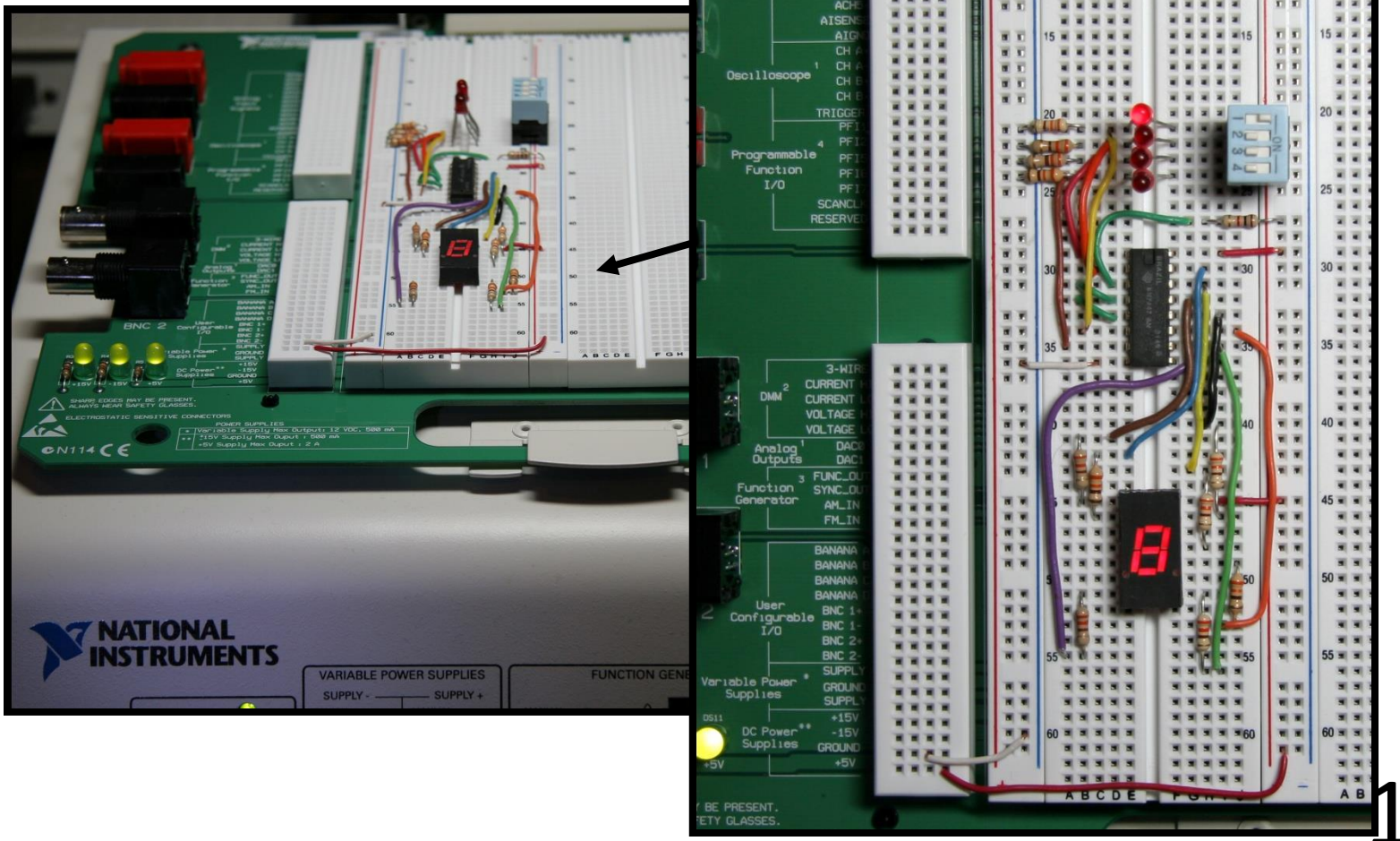
8000 4000 2000 1000 800 400 200 100 80 40 20 10 8 4 2 1

Note that you could add the column weights where there is a 1 to obtain the decimal number. For this case:

$8000 + 200 + 100 + 40 + 10 + 8 + 1 = 8359_{10}$

BCD

A lab experiment in which BCD is converted to decimal is shown.



10

Gray code

Gray code is an unweighted code that has a single bit change between one code word and the next in a sequence. Gray code is used to avoid problems in systems where an error can occur if more than one bit changes at a time.

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

Gray code

Binary-to-Gray Code Conversion Conversion between binary code and Gray code is sometimes useful. The following rules explain how to convert from a binary number to a Gray code word:

1. The most significant bit (left-most) in the Gray code is the same as the corresponding MSB in the binary number.
2. Going from left to right, add each adjacent pair of binary code bits to get the next Gray code bit. Discard carries.

For example, the conversion of the binary number 10110 to Gray code is as follows:

1	0	1	1	0	Binary
↓	↓	↓	↓	↓	
1	1	1	0	1	Gray

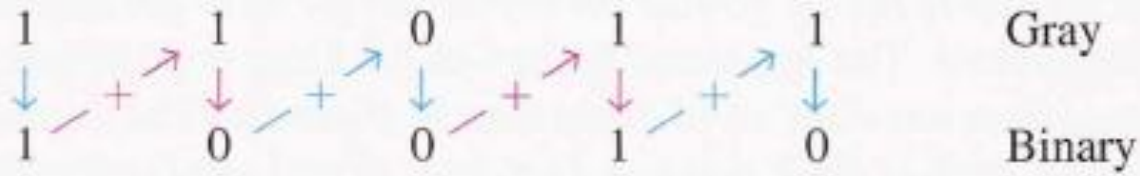
The Gray code is 11101.

Gray code

Gray-to-Binary Conversion To convert from Gray code to binary, use a similar method; however, there are some differences. The following rules apply:

1. The most significant bit (left-most) in the binary code is the same as the corresponding bit in the Gray code.
2. Add each binary code bit generated to the Gray code bit in the next adjacent position. Discard carries.

For example, the conversion of the Gray code word 11011 to binary is as follows:



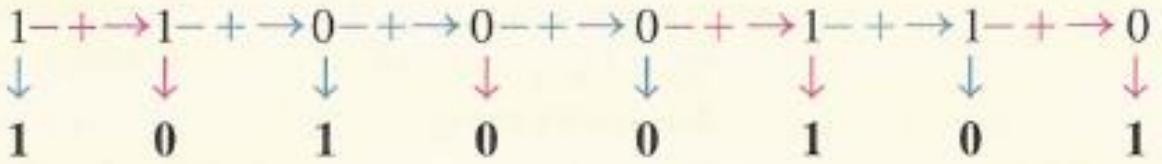
The binary number is 10010.

Gray code

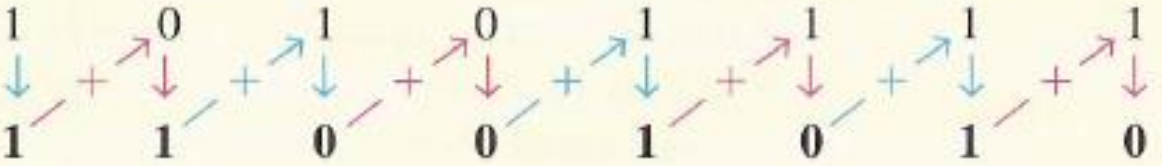
(a) Convert the binary number 11000110 to Gray code.

(b) Convert the Gray code 10101111 to binary.

Solution **(a)** Binary to Gray code:

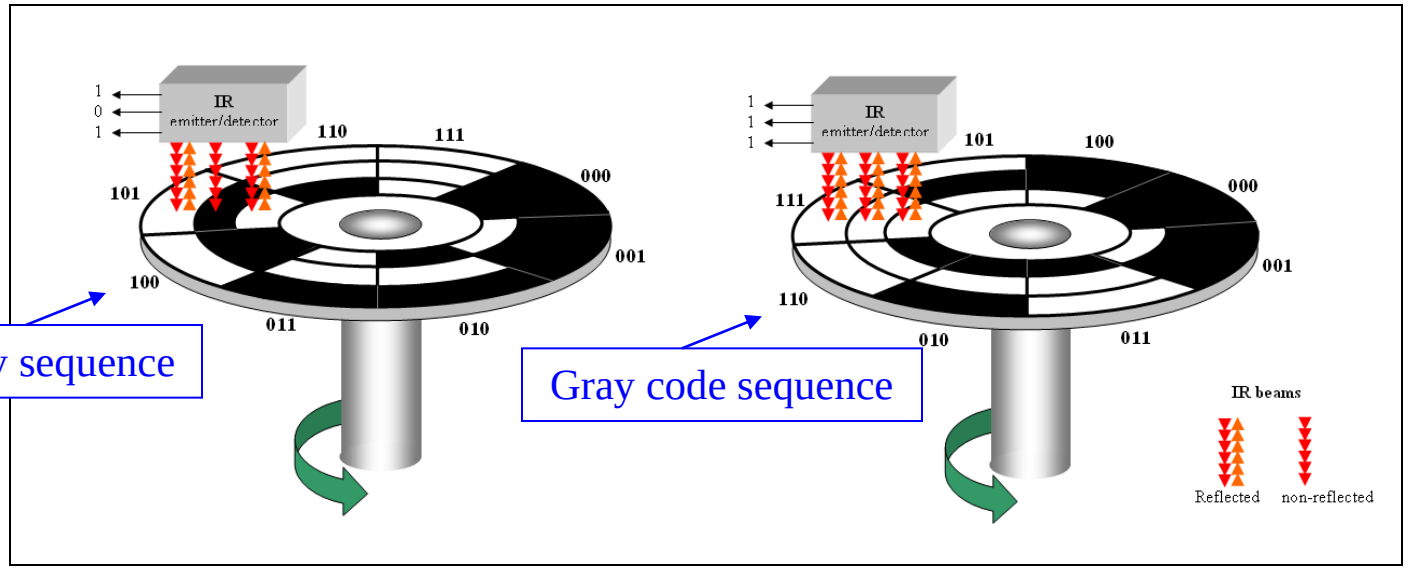


(b) Gray code to binary:



Gray code

A shaft encoder is a typical application. Three IR emitter/detectors are used to encode the position of the shaft. The encoder on the left uses binary and can have three bits change together, creating a potential error. The encoder on the right uses gray code and only 1-bit changes, eliminating potential errors.



15

ASCII

ASCII is the abbreviation for **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.

ASCII is a code for alphanumeric characters and control characters. In its original form, ASCII encoded 128 characters and symbols using 7-bits. The first 32 characters are control characters, that are based on obsolete teletype requirements, so these characters are generally assigned to other functions in modern usage.

In 1981, IBM introduced extended ASCII, which is an 8-bit code and increased the character set to 256. Other extended sets (such as Unicode) have been introduced to handle characters in languages other than English.

ASCII code table

CONTROL CHARACTERS				GRAPHIC SYMBOLS											
NAME	DEC	BINARY	HEX	SYMBOL	DEC	BINARY	HEX	SYMBOL	DEC	BINARY	HEX	SYMBOL	DEC	BINARY	HEX
NUL	0	0000000	00	space	32	0100000	20	@	64	1000000	40	`	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	'	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	–	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

ASCII

Determine the binary ASCII codes that are entered from the computer's keyboard when the following BASIC program statement is typed in. Also express each code in hexadecimal.

20 PRINT "A=";X

Solution The ASCII code for each symbol is found in Table 2–7.

Symbol	Binary	Hexadecimal
2	0110010	32_{16}
0	0110000	30_{16}
Space	0100000	20_{16}
P	1010000	50_{16}
R	1010010	52_{16}
I	1001001	49_{16}
N	1001110	$4E_{16}$
T	1010100	54_{16}
Space	0100000	20_{16}
"	0100010	22_{16}
A	1000001	41_{16}
=	0111101	$3D_{16}$
"	0100010	22_{16}
;	0111011	$3B_{16}$
X	1011000	58_{16}

Extended ASCII

The extended ASCII contains characters in the following general categories:

1. Foreign (non-English) alphabetic characters
2. Foreign currency symbols
3. Greek letters
4. Mathematical symbols
5. Drawing characters
6. Bar graphing characters
7. Shading characters

Extended ASCII

SYMBOL	DEC	HEX	SYMBOL	DEC	HEX	SYMBOL	DEC	HEX	SYMBOL	DEC	HEX
Ç	128	80	á	160	A0	└	192	C0	α	224	E0
ü	129	81	í	161	A1	┴	193	C1	β	225	E1
é	130	82	ó	162	A2	┬	194	C2	Γ	226	E2
â	131	83	ú	163	A3	├	195	C3	π	227	E3
ä	132	84	ñ	164	A4	─	196	C4	Σ	228	E4
à	133	85	Ñ	165	A5	┼	197	C5	σ	229	E5
å	134	86	ª	166	A6	╞	198	C6	µ	230	E6
ç	135	87	º	167	A7	╟	199	C7	τ	231	E7
ê	136	88	¿	168	A8	╚	200	C8	Φ	232	E8
ë	137	89	⌐	169	A9	╔	201	C9	Θ	233	E9
è	138	8A	¬	170	AA	╩	202	CA	Ω	234	EA
ï	139	8B	½	171	AB	╦	203	CB	δ	235	EB
î	140	8C	¼	172	AC	╠	204	CC	∞	236	EC
ì	141	8D	¡	173	AD	═	205	CD	φ	237	ED
Ä	142	8E	«	174	AE	╬	206	CE	ε	238	EE
Å	143	8F	»	175	AF	╧	207	CF	∩	239	EF
É	144	90	▒	176	B0	╨	208	D0	≡	240	F0
æ	145	91	▒	177	B1	╤	209	D1	±	241	F1
Æ	146	92	▒	178	B2	╥	210	D2	≥	242	F2
ô	147	93	│	179	B3	╙	211	D3	≤	243	F3
ö	148	94	┤	180	B4	╘	212	D4	⌠	244	F4
ò	149	95	╡	181	B5	╒	213	D5	⌡	245	F5
û	150	96	╢	182	B6	┐	214	D6	÷	246	F6
ù	151	97	╖	183	B7	╫	215	D7	≈	247	F7
ÿ	152	98	╕	184	B8	╪	216	D8	°	248	F8
Ö	153	99	╣	185	B9	┘	217	D9	∙	249	F9
Ü	154	9A	║	186	BA	┌	218	DA	·	250	FA
¢	155	9B	╗	187	BB	▉	219	DB	√	251	FB
£	156	9C	╝	188	BC	■	220	DC	η	252	FC
¥	157	9D	╜	189	BD	▌	221	DD	^{2}	253	FD
Pₜ	158	9E	╛	190	BE	▐	222	DE	■	254	FE
ƒ	159	9F	┐	191	BF	▀	223	DF	□	255	FF

Parity Method

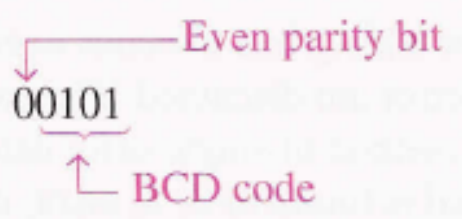
The parity method is a method of error detection for simple transmission errors involving one bit (or an odd number of bits). A parity bit is an "extra" bit attached to a group of bits to force the number of 1's to be either even (even parity) or odd (odd parity).

Example The ASCII character for "a" is 1100001 and for "A" is 1000001. What is the correct bit to append to make both of these have odd parity?

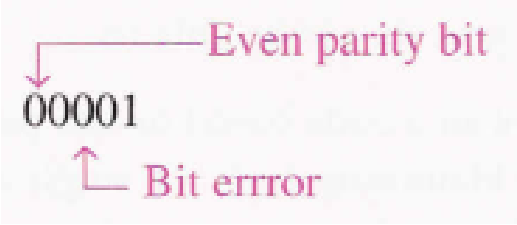
Solution The ASCII "a" has an odd number of bits that are equal to 1; therefore the parity bit is 0. The ASCII "A" has an even number of bits that are equal to 1; therefore the parity bit is 1.

Parity method for error detection

- **Detecting an Error: A parity bit provides for the detection of a single bit error (or any odd number of errors) but can not check for two errors in one group.**



- **Assume that an error in the third bit from the left (the 1 becomes a 0)**

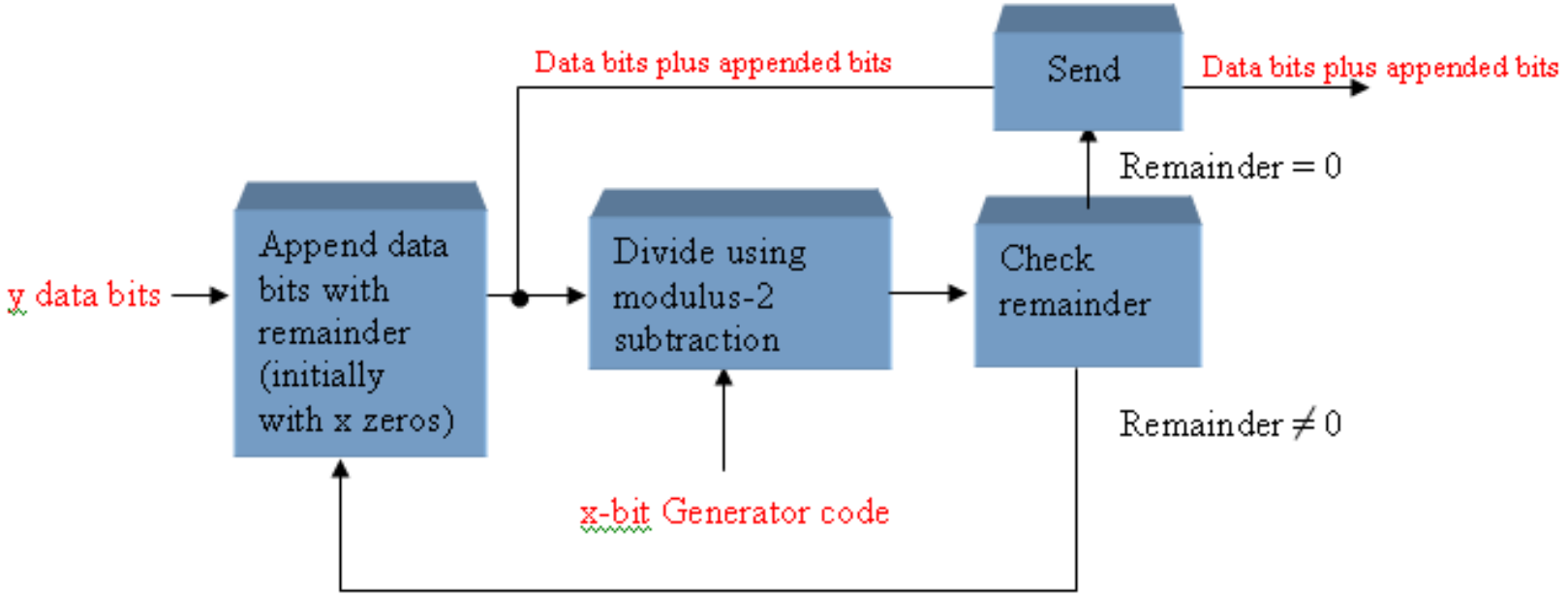


Parity bit for BCD code

EVEN PARITY		ODD PARITY	
P	BCD	*P*	BCD
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

Cyclic Redundancy Check

The cyclic redundancy check (CRC) is an error detection method that can detect multiple errors in larger blocks of data. At the sending end, a checksum is appended to a block of data. At the receiving end, the check sum is generated and compared to the sent checksum. If the check sums are the same, no error is detected.



4.4 Biểu diễn các ký tự (tt)

2. **Bộ mã ISO:**
   - Do International Organization for Standardization đưa ra năm 1967
   - Dùng 7 bit để biểu diễn ký tự, dựa trên bộ mã ASCII
   - https://en.wikipedia.org/wiki/ISO/IEC_646
3. **Bảng mã EBCDIC**
   - Extended Binary Code Decimal Interchange Code (IBM)
   - Mỗi ký tự được biểu diễn với 8 bit
   - https://en.wikipedia.org/wiki/EBCDIC
4. **Bảng mã UNICODE**
   - Mỗi ký tự được biểu diễn với 16 bit
   - Tất cả ký tự thuộc tất cả ngôn ngữ trên thế giới
   - http://en.wikipedia.org/wiki/Unicode

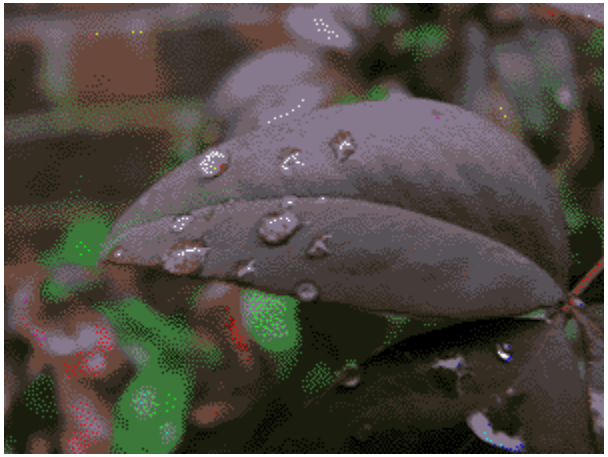
4.5 Biểu diễn hình ảnh và âm thanh

- **Biểu diễn hình ảnh**
  - Ảnh bitmap
  - Ảnh vector
- **Biểu diễn âm thanh**

Biểu diễn hình ảnh
- Ảnh bitmap

- **Bitmap: = map of bits**
  - Đây là loại ảnh "thô" dựa trên dãy các điểm ảnh, kích thước lớn.
- **Pixel (điểm ảnh):**
  - Đơn vị nhỏ nhất trong lưu trữ ảnh bitmap.
  - Mỗi điểm ảnh có thể được lưu trữ với 1, 2, 4, 8, 16, 24, 32, 48 hoặc 64 bit.
- **Một số loại file ảnh bitmap: GIF, JPEG, PNG, TIFF and BMP.**

1 bit



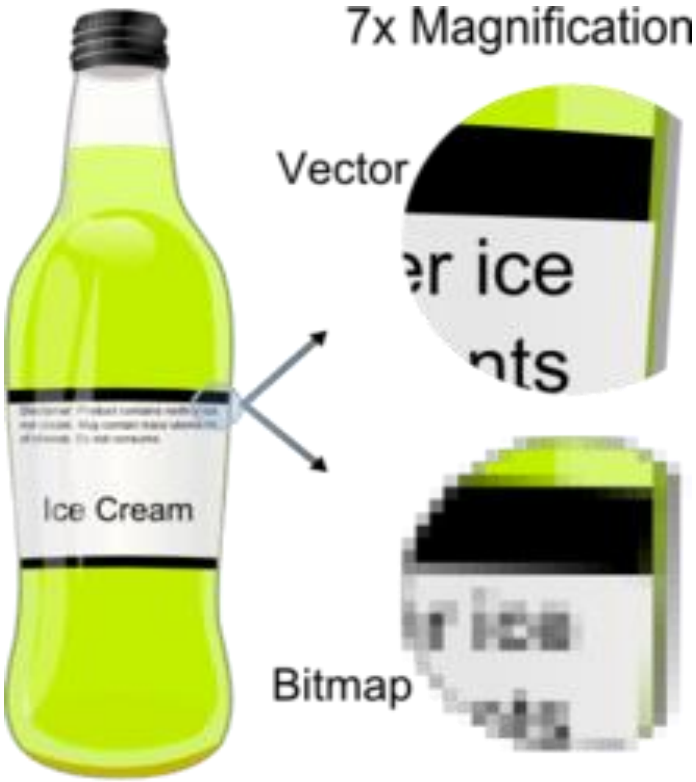
4 bit

8 bit

24 bit

Biểu diễn hình ảnh
- Ảnh vector

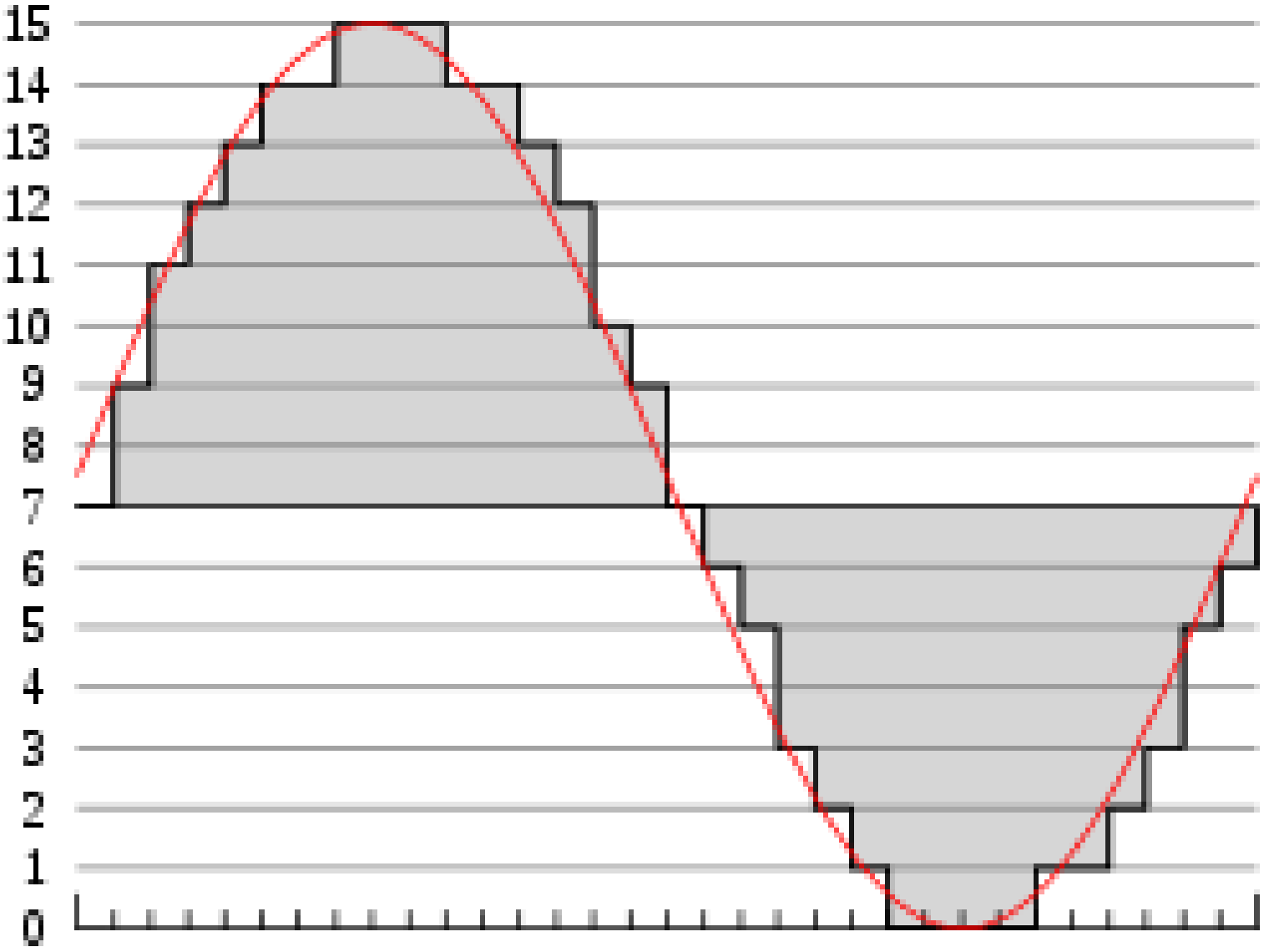
- Ảnh vector sử dụng các đối tượng hình học cơ bản như điểm, đường thẳng, đường cong và các đa giác, dựa trên các phương trình toán học, để biểu diễn hình ảnh trong máy tính.
- Một số loại file ảnh vector: AI, DXF, CGM, MIF and SVG.

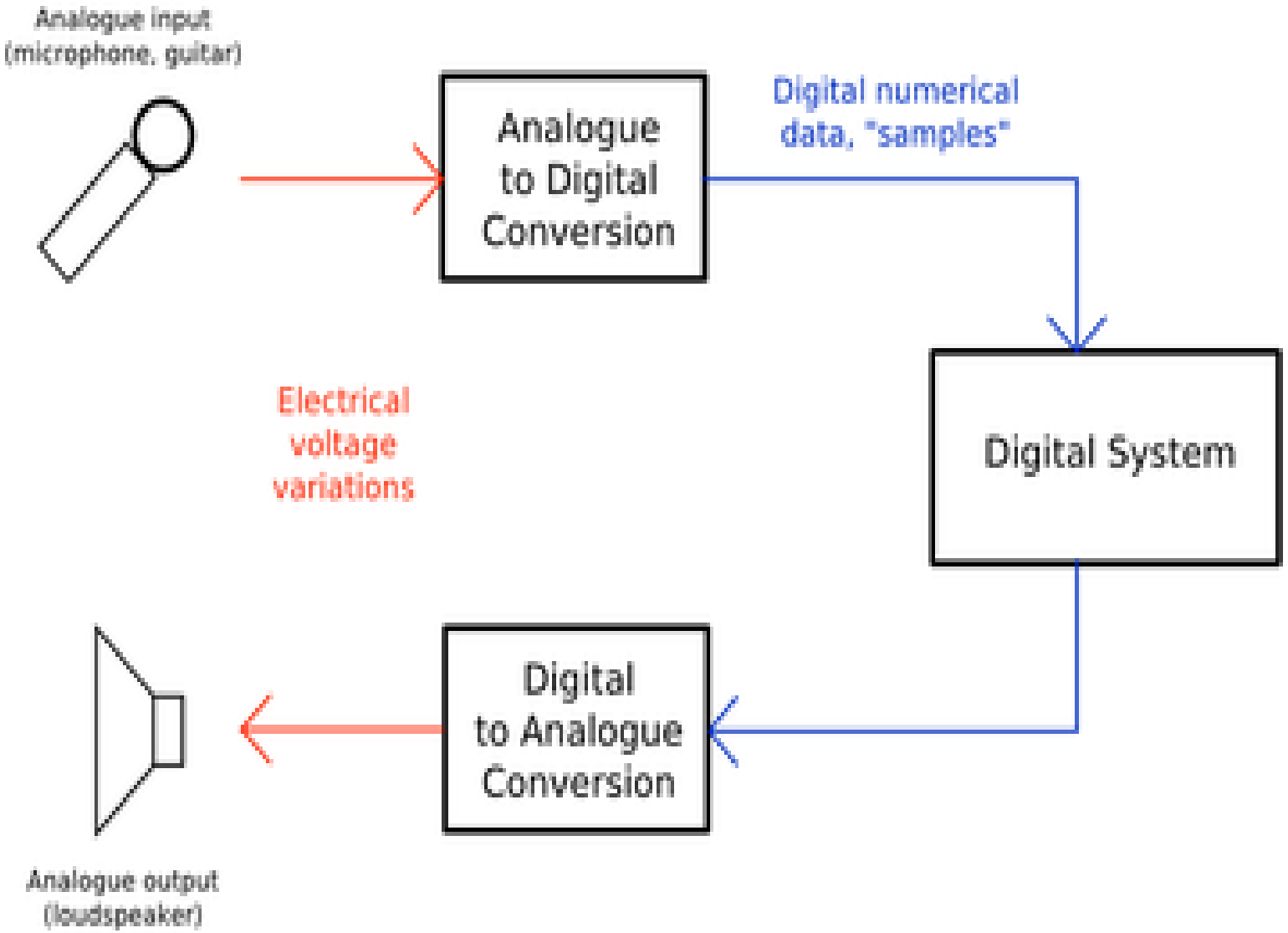


https://www.youtube.com/watch?v=fy9Pby0Gzsc

Biểu diễn âm thanh

- Âm thanh số sử dụng các tín hiệu số để tái lập âm thanh, gồm có: bộ chuyển đổi tín hiệu tuần tự - số, số - tuần tự, lưu trữ và truyền dẫn.
- Các định dạng file âm thanh:
  - Không nén: wav, aiff, au.
  - Nén không mất mát: FLAC, APE, WV, TAK, TTA, WMA không mất mát.
  - Nén có mất mát: MP3, Vorbis, AAC, WMA có mất mát.
- Tham khảo: http://en.wikipedia.org/wiki/Audio_file_format


15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0


Analogue input
(microphone, guitar)
Analogue
to Digital
Conversion
Digital numerical
data, "samples"
Digital System
Electrical
voltage
variations
Digital
to Analogue
Conversion
Analogue output
(loudspeaker)

Tóm tắt

- Chuyển đổi giữa các số hệ 10 và hệ k bất kỳ như thế nào?
- Chuyển đổi giữa hệ nhị phân và hệ thập lục phân như thế nào?
- Biểu diễn bù 2 của một số nhị phân được thực hiện như thế nào?
- Biểu diễn số nguyên âm trong máy tính như thế nào?
- Số thực dấu chấm động được biểu diễn ntn?
- Tràn trên (overflow) và tràn dưới (underflow) trong biểu diễn số thực dấu chấm động là gì?
- Có những bộ mã biểu diễn ký tự nào?
- Hình ảnh và âm thanh được số hóa ntn?

Kiểm tra ngắn (30 phút)

Câu 1. Với chuẩn dấu chấm động IEEE 32bit, hãy cho biết dạng biểu diễn các giá trị sau đây dưới dạng nhị phân:

a. -5 **b. 53.203125**

Yêu cầu: thực hiện đúng các bước:

(1) **Đổi từ hệ 10 sang hệ 2**

(2) **Chuẩn hóa**

(3) **Biểu diễn dạng dấu chấm động IEEE 32bit**

Câu 2. Hãy cho biết giá trị thực trong hệ 10 của biểu diễn dấu chấm động IEEE 32bit sau đây:

01000010001110000000000000000000

Đáp án

- **Câu 1:**
  a. -5:
    - B1: Đổi nhị phân:-5 = - 101_2
    - B2: Chuẩn hóa: -101 = -1.01 x 2 mũ 2
    - B3: Biểu diễn:
      - Bit dấu: 1
      - 8 bit mũ: 2 + 127 = 129 = 10000001_2
      - 23 bit định trị: 0100..0
    - B4: kết luận: **1**10000001 0100..0

  b. 53.203125
    - B1: Đổi nhị phân: 53.203125 = +110101.001101_2
    - B2: Chuẩn hóa: = 1.10101001101 x 2 mũ 5
    - B3: Biểu diễn:
      - Bit dấu: 0
      - 8 bit mũ: 5 + 127 = 132 = 10000100_2
      - Định trị: 1010100110100..00
    - Kết luận: **0** **10000100** 101010011010...0

Câu 2: 0**10000100**00111000000000000000000 gồm: mũ (1000100 = 132 → mũ = 132 – 127 = 5),

chuẩn hóa: 1.00111 x 2 mũ 5 = $100111 = 39_{10}$

Practice

Practice Problem 2.4

Without converting the numbers to decimal or binary, try to solve the following arithmetic problems, giving the answers in hexadecimal. **Hint:** Just modify the methods you use for performing decimal addition and subtraction to use base 16.

A. `0x503c + 0x8 = _____`

B. `0x503c – 0x40 = _____`

C. `0x503c + 64 = _____`

D. `0x50ea – 0x503c = _____`

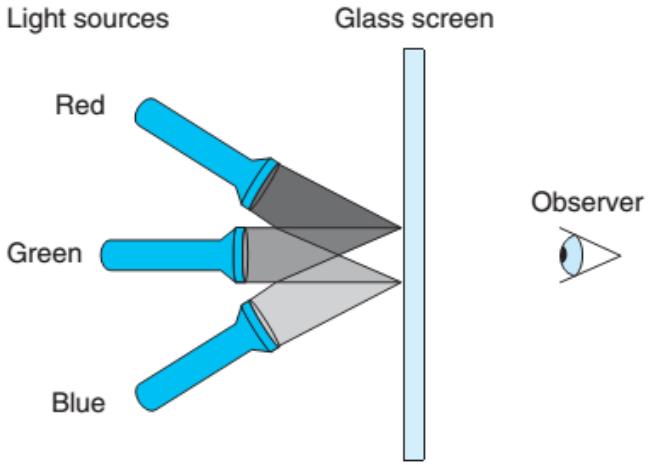
Practice Problem 2.8

Fill in the following table showing the results of evaluating Boolean operations on bit vectors.

Operation	Result
a	[01101001]
b	[01010101]
$\sim a$	_____
$\sim b$	_____
$a \,\&\, b$	_____
$a \mid b$	_____
$a \,\hat{}\, b$	_____

Practice Problem 2.9

Computers generate color pictures on a video screen or liquid crystal display by mixing three different colors of light: red, green, and blue. Imagine a simple scheme, with three different lights, each of which can be turned on or off, projecting onto a glass screen:



We can then create eight different colors based on the absence (0) or presence (1) of light sources R, G, and B:

R	G	B	Color
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

Each of these colors can be represented as a bit vector of length 3, and we can apply Boolean operations to them.

A. The complement of a color is formed by turning off the lights that are on and turning on the lights that are off. What would be the complement of each of the eight colors listed above?

B. Describe the effect of applying Boolean operations on the following colors:

Blue	|	Green	=	_____
Yellow	&	Cyan	=	_____
Red	^	Magenta	=	_____

Practice Problem 2.14

Suppose that `x` and `y` have byte values `0x66` and `0x39`, respectively. Fill in the following table indicating the byte values of the different C expressions:

Expression	Value	Expression	Value
`x & y`	_____	`x && y`	_____
`x | y`	_____	`x || y`	_____
`~x | ~y`	_____	`!x || !y`	_____
`x & !y`	_____	`x && ~y`	_____

Practice Problem 2.16

Fill in the table below showing the effects of the different shift operations on single-byte quantities. The best way to think about shift operations is to work with binary representations. Convert the initial values to binary, perform the shifts, and then convert back to hexadecimal. Each of the answers should be 8 binary digits or 2 hexadecimal digits.

x		x << 3		(Logical) x >> 2		(Arithmetic) x >> 2	
Hex	Binary	Binary	Hex	Binary	Hex	Binary	Hex
0xC3	_____	_____	_____	_____	_____	_____	_____
0x75	_____	_____	_____	_____	_____	_____	_____
0x87	_____	_____	_____	_____	_____	_____	_____
0x66	_____	_____	_____	_____	_____	_____	_____