Quiz EL301
Binary Codes and Data Representation

Lucas Monteiro Nogueira

PROBLEMS

Problem 1

Convert the following binary numbers to decimal form.

Problem 1.1: 1010
Problem 1.2: 10011
Problem 1.3: 10.1
Problem 1.4: 11.011

Convert the following decimal numbers to binary form.

Problem 1.5: 12
Problem 1.6: 25
Problem 1.7: 6.34
Problem 1.8: e (=2.7182818...)

Multiply the following binary numbers without converting them to decimal form.

Problem 1.9: 101×11
Problem 1.10: 1101×101
Problem 1.11: 110110×1101

Problem 2

Convert the following hexadecimal numbers to decimal form.

Problem 2.1: 2A4
Problem 2.2: 67E
Problem 2.3: 892E
Problem 2.4: ABCD
Problem 2.5: A3F.D

Convert the following decimal numbers to hexadecimal form.

Problem 2.6: 350
Problem 2.7: 982
Problem 2.8: 7200
Problem 2.9: 1050.1
Problem 2.10: π (=3.14159265...)

Problem 3

Problem 3.1: Find the 16's complement of $(B2FA)_{16}$.
Problem 3.2: Convert B2FA to binary.
Problem 3.3: Find the 2's complement of the result in Part 2.
Problem 3.4: Convert the answer in Part 3 to hexadecimal and compare with the answer in Part 1.

Problem 4 (Modified from Sedha, 2004)

A PC-compatible computer system uses 20-bit address codes to identify each of over 1 million memory locations.

Problem 4.1: How many hexadecimal characters are required to identify the address of a single memory location?
Problem 4.2: Assuming the address count begins at a hexadecimal number made up entirely of zeros and increases by one hexadecimal unit for each successive address, what is the hexadecimal address of the 501st memory location?
Problem 4.3: If 60 memory locations are used for data storage starting at location 00FF2, what is the location of the last data item? Convert the location of the last data item to decimal form.

Problem 5

Find the 9's and 10's complements of the following decimal numbers:
Problem 5.1: 25,000,000
Problem 5.2: 99,995,500
Problem 5.3: 52,784,630
Problem 5.4: 63,325,600

Perform subtraction on the given following numbers using the 10's complement of the subtrahend. Don't forget to assign the appropriate sign to each result.
Problem 5.5: 6,428 – 3,409
Problem 5.6: 125 – 1,800
Problem 5.7: 2,043 – 6,152
Problem 5.8: 1,631 – 745

Problem 6

Perform subtraction on the given following numbers using the 2's complement of the subtrahend. Don't forget to assign the appropriate sign to each result.
Problem 6.1: 10011 – 10001
Problem 6.2: 100010 – 100011
Problem 6.3: 1001 – 101000
Problem 6.4: 110000 – 10101

Problem 7

Problem 7.1: Represent the unsigned decimal numbers 842 and 537 in binary-coded decimal (BCD), and then show the steps necessary to form their sum.
Problem 7.2: Find the BCD sum for numbers 281 and 679.

Problem 8

Problem 8.1: Represent the unsigned decimal numbers 541 and 736 in excess-3 (XS-3) form, and then show the steps necessary to form their sum.
Problem 8.2: Find the XS-3 sum for numbers 806 and 307.

Problem 9

Consider the decimal number 5,137.
Problem 9.1: Write this number in BCD.
Problem 9.2: Use the BCD code obtained in Part 1 to determine the excess-3 code for the decimal number in question.
Problem 9.3: Use the excess-3 code obtained in Part 2 to determine the XS-3 representation for the decimal number 4,862.
Problem 9.4: Write the decimal number 6,251 in 2421 code and use its self-complementary property to determine the 2421-code representation of 3,748.

Problem 10

Convert the following binary codes to Gray code.
Problem 10.1: 10110
Problem 10.2: 1111001
Problem 10.3: Encode the decimal number 48 to Gray code.

Convert the following Gray codes to binary codes.
Problem 10.4: 10001
Problem 10.5: 1010101

Problem 11 (Modified from Mano and Ciletti, 2004)

Problem 11.1: Referring to Table 3, decode the following ASCII code.

1000010 1101001 1101100 1101100 1000111 1100001 1110100 1100101 1110011

Problem 11.2: Write the expression "G. Boole" in ASCII, using an eight-bit code. Include the period and the space. Treat the leftmost bit of each character as a parity bit. Each eight-bit code should have even parity.

Problem 12 (Mano and Ciletti, 2004)

The following is a string of ASCII characters whose bit patterns have been converted into hexadecimal form for compactness:

53 F4 E5 76 E5 4A EF 62 73

Of the eight bits in each pair of digits, the leftmost one is a parity bit. The remaining bits are the ASCII code.
Problem 12.1: Convert the string to bit form and decode the ASCII.
Problem 12.2: Determine the parity used: odd or even?

ADDITIONAL INFORMATION

Table 1 Binary-Coded Decimal (BCD)*, excess-3 and 2421 code numbers

**Numbers 0 to 9 in BCD are no different from Their "normal" binary representations.*

Decimal number	BCD	XS-3	2421 Code
0	0000	0011	0000
1	0001	0100	0001
2	0010	0101	0010
3	0011	0110	0011
4	0100	0111	0100
5	0101	1000	1011
6	0110	1001	1100
7	0111	1010	1101
8	1000	1011	1110
9	1001	1100	1111

Table 2 Gray code

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

Table 3 American Standard Code for Information Exchange (ASCII)*

b_7 denotes the most significant bit (MSB), while b_1 denotes the least significant bit (LSB)

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	**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

Control characters

NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

SOLUTIONS

P.1 Solution

Problem 1.1: First, note that the leftmost digit, a 1, is in the 2^3 place; it follows that this digit contributes $2^3 \times 1 = 8$ to the conversion to decimal. Next, a number 0 is in the 2^2 place and therefore contributes $2^2 \times 0 = 0$ to the conversion to decimal. Next, a number 1 is in the 2^1 place and therefore contributes $2^1 \times 1 = 2$ to the conversion to decimal. Lastly, a number 0 is in the 2^0 place and hence contributes $2^0 \times 0 = 0$ to the conversion to decimal. We conclude that

$$(1010)_{10} = \underbrace{2^3 \times 1}_{=8} + \underbrace{2^2 \times 0}_{=0} + \underbrace{2^1 \times 1}_{=2} + \underbrace{2^0 \times 0}_{=0} = \boxed{10}$$

Problem 1.2: The conversion is performed below.

$$(10011)_{10} = \underbrace{2^4 \times 1}_{=16} + \underbrace{2^3 \times 0}_{=0} + \underbrace{2^2 \times 0}_{=0} + \underbrace{2^1 \times 1}_{=2} + \underbrace{2^0 \times 1}_{=1} = \boxed{19}$$

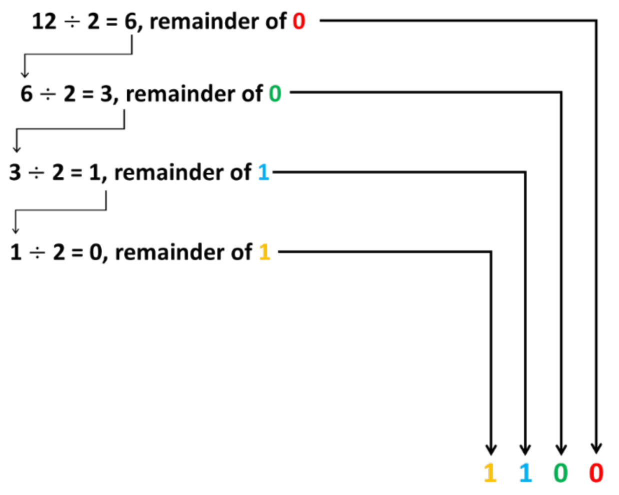
Problem 1.3: The conversion is performed below.

$$(10.1)_{10} = \underbrace{2^1 \times 1}_{=2} + \underbrace{2^0 \times 0}_{=0} + \underbrace{2^{-1} \times 1}_{=0.5} = \boxed{2.5}$$

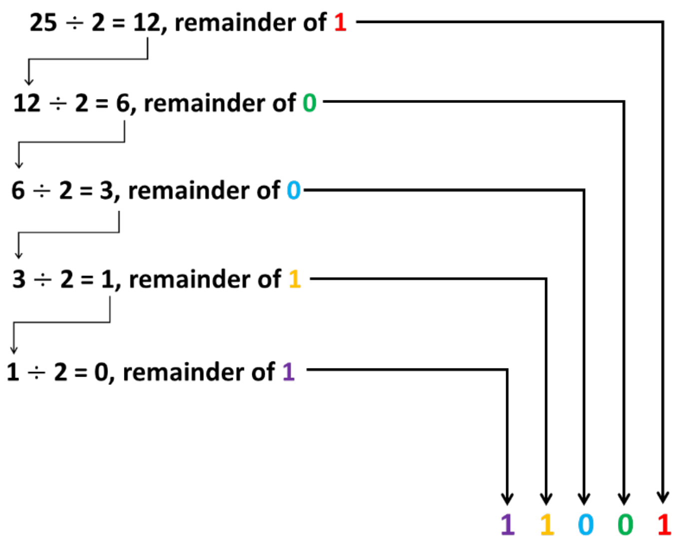
Problem 1.4: The conversion is performed below.

$$(11.011)_{10} = \underbrace{2^1 \times 1}_{=2} + \underbrace{2^0 \times 1}_{=1} + \underbrace{2^{-1} \times 0}_{=0} + \underbrace{2^{-2} \times 1}_{=0.25} + \underbrace{2^{-3} \times 1}_{=0.125} = \boxed{3.375}$$

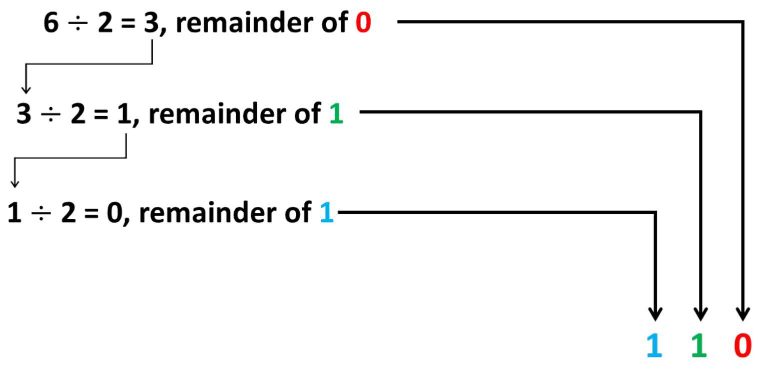
Problem 1.5: A decimal number can be easily expressed in binary form via the divide-by-2 process. Dividing 12 by 2 once gives 6 with a remainder of 0. The remainder is taken as the last digit in the binary representation we are looking for, while the quotient is carried over and used in a second division by 2. Upon dividing the quotient 6 by 2, we obtain a remainder of 0, which is taken as the second (right to left) digit in the binary representation we aim for; the quotient is 3, which we carry over to a third division by 2. Upon dividing the quotient 3 by 2, we obtain a quotient of 1 and a remainder of 1; the remainder is taken as the third digit (right to left) in the binary representation, while the quotient is carried over and divided by 2 once again. Dividing the quotient 1 by 2 results in zero with a remainder of 1; this remainder is taken as the fourth digit (right to left) in the binary representation. The fact that the quotient of this last division is zero indicates that there is no need to proceed to another division by 2; the binary form of 12 is found to be 1100.



Problem 1.6: Here, the procedure is identical to the one adopted in Problem 1.5. The binary representation of 25 is found to be 11001.



Problem 1.7: This number differs from the previous ones because it has a fractional component. To convert the integer part of the number, we proceed as we would in the two previous exercises.



Now, converting the fractional part of the number involves successive *multiplication* by 2. We begin by multiplying the fractional part, 0.34, by 2, which yields 0.68. Then, we take the integer part or the result as the first fractional digit of the binary representation. In the case at hand, the number to the left of the dot in 0.68 is 0; accordingly, the first fractional digit in the binary form of 6.34 is 0, so we can write $(6.34)_{10} = (110.0)_2$. To find the next fractional digit, we take the fractional part of the number we got when finding the first fractional digit and multiply it by 2. We had obtained 0.68, so we take 0.68 and write 0.68 × 2 = 1.38. The integer part of this result constitutes the next decimal digit of the hexadecimal representation; such a digit is 1, so we may write $(6.34)_{10} = (110.01)_2$. Finding the third fractional digit is no different: the fractional part of the previous multiplication by 2 is 0.38; multiplying this result by 2 gives 0.76; thus, the binary representation is updated as $(1050.1)_{10} = (110.010)_2$. Next, we take 0.76 and multiply it by 2, giving 1.52; the binary representation is updated as $(1050.1)_{10} = (110.0101)_{hex}$. More fractional digits can be computed in this manner but, for convenience, we truncate the conversion at the fourth decimal digit and write

$$(6.34)_{10} = (110.0101...)_2$$

Problem 1.8: The procedure to convert e, an irrational number, to binary form is similar to the procedure adopted in Problem 1.7. Converting the integer part, which is 2, is elementary: $(2)_{10} = (10)_2$. Converting the fractional part is more involved. We begin by taking the fractional part of e and multiplying it by 2, that is, 2 × 0.718281828 = 1.436563656. The integer part, which is a 1, will be the first fractional digit in the hexadecimal representation:

$$\mathbf{(2.718281828)_{10} = (10.1...)_2}$$

Next, multiplying 0.436563656 by 2 gives 0.873127312. The integer part, which is a 0, will be the second fractional digit of the binary representation:

$$\mathbf{(2.718281828)_{10} = (10.10...)_2}$$

Next, multiplying 0.873127312 by 2 gives 1.746254624. The integer part, which is a 1, will be the third fractional digit of the binary representation:

$$(2.718281828)_{10} = (10.101...)_2$$

Next, multiplying 0. 746254624 by 2 gives 1.492509248. The integer part, which is a 1, will be the fourth fractional digit in the binary representation:

$$(2.718281828)_{10} = (10.1011...)_2$$

Next, multiplying 0. 492509248 by 2 gives 0.985018496. The integer part, which is a 0, will be the fifth fractional digit in the binary representation:

$$(2.718281828)_{10} = (10.10110...)_2$$

Since e is an irrational number, it has infinite, non-periodic fractional digits, be it in decimal or binary representation. We truncate our approximation at the fifth fractional space and write

$$(e)_{10} = (10.10110...)_2$$

Problem 1.9: To work out the multiplication of two binary numbers, we resort to the same partial sum algorithm used in decimal number multiplication. In the present case, we begin by multiplying the rightmost digit of the multiplier by the multiplicand. The multiplier at hand is 11, so we take its rightmost digit, which is 1, and multiply it by 101 (the multiplicand), as shown. The result of the multiplication is labeled *Partial sum #1*.

$$\begin{array}{rl} 101 & \times \\ 11 & \\ \hline 101 & \textit{Partial sum \#1} \end{array}$$

In the second step, we take the second digit of the multiplier from right to left and multiply it with the multiplicand. Before doing so, however, we place a zero in the rightmost space of the result and position the results of successive multiplications to the left of this zero, as shown. The result of the multiplication is labeled *Partial sum #2*.

$$\begin{array}{rl} 101 & \times \\ 11 & \\ \hline 101 & \textit{Partial sum \#1} \\ 1010 & \textit{Partial sum \#2} \end{array}$$

The two digits of the multiplier have been covered. The last step is to add the partial sums as we would in any binary addition. The result of this last addition is the product of the two binary numbers we began with. The result is $(1111)_2$.

$$\begin{array}{rl} 101 & \times \\ 11 & \\ \hline 101 & + \;\textit{P.S. \#1} \\ 1010 & \;\;\textit{P.S. \#2} \\ \hline 1111 & \end{array}$$

Problem 1.10: The first partial sum is computed below.

$$\begin{array}{rl} 1101 & \times \\ 101 & \\ \hline 1101 & \textit{P.S. \#1} \end{array}$$

The second partial sum is computed below. Since in this case the multiplier is a zero, all products turn out to yield zero, and the resulting partial sum will be nil; this is one reason why binary number products are often quicker to compute by hand than products of decimal quantities.

$$\begin{array}{rl} 1101 & \times \\ 101 & \\ \hline 1101 & \textit{P.S. \#1} \\ 00000 & \textit{P.S. \#2} \end{array}$$

The third and final partial sum is computed below.

$$\begin{array}{r} 1101\phantom{\text{ P.S. \#1}} \\ \times\ 101\phantom{\text{ P.S. \#1}} \\ \hline 1101\text{ P.S. \#1} \\ 00000\text{ P.S. \#2} \\ 110100\text{ P.S. \#3} \end{array}$$

The final step is to add the three partial sums. The result is $(1000001)_2$.

$$\begin{array}{r} 1101\phantom{\text{ P.S. \#1}} \\ \times\ 101\phantom{\text{ P.S. \#1}} \\ \hline 1101\text{ P.S. \#1} \\ +\ 00000\text{ P.S. \#2} \\ 110100\text{ P.S. \#3} \\ \hline 1000001\phantom{\text{ P.S. \#1}} \end{array}$$

Problem 1.11: The first partial sum is computed below.

$$\begin{array}{r} 110110\phantom{\text{ P.S. \#1}} \\ \times\ 1101\phantom{\text{ P.S. \#1}} \\ \hline 110110\text{ P.S. \#1} \end{array}$$

The second partial sum is computed below.

$$\begin{array}{r} 110110\phantom{\text{ P.S. \#1}} \\ \times\ 1101\phantom{\text{ P.S. \#1}} \\ \hline 110110\text{ P.S. \#1} \\ 0000000\text{ P.S. \#2} \end{array}$$

The third partial sum is computed below.

$$\begin{array}{r} 110110\phantom{\text{ P.S. \#1}} \\ \times\ 1101\phantom{\text{ P.S. \#1}} \\ \hline 110110\text{ P.S. \#1} \\ 0000000\text{ P.S. \#2} \\ 11011000\text{ P.S. \#3} \end{array}$$

The fourth partial sum is computed below.

$$\begin{array}{r} 110110\phantom{\text{ P.S. \#1}} \\ \times\ 1101\phantom{\text{ P.S. \#1}} \\ \hline 110110\text{ P.S. \#1} \\ 0000000\text{ P.S. \#2} \\ 11011000\text{ P.S. \#3} \\ 110110000\text{ P.S. \#4} \end{array}$$

The final step is to add the four partial sums. The result is $(1010111110)_2$.

$$\begin{array}{r} 110110\phantom{\text{ P.S. \#1}} \\ \times\ 1101\phantom{\text{ P.S. \#1}} \\ \hline 110110\text{ P.S. \#1} \\ +\ 0000000\text{ P.S. \#2} \\ 11011000\text{ P.S. \#3} \\ 110110000\text{ P.S. \#4} \\ \hline 1010111110\phantom{\text{ P.S. \#1}} \end{array}$$

P.2 ➡ *Solution*

Problem 2.1: First, note that the leftmost digit, a 2, is in the 16^2 place; it follows that this digit contributes $16^2 \times 2 = 512$ to the conversion to decimal. Next, the number *A*, which is equivalent to 10, is in the 16^1 place and therefore contributes $16^1 \times 10 = 160$ to the conversion to decimal. Lastly, the number 4 is in the 16^0 place and hence contributes $16^0 \times 4 = 4$ to the conversion to decimal. We conclude that

$$(2\text{A}4)_{10} = \underbrace{16^2 \times 2}_{=512} + \underbrace{16^1 \times 10}_{=160} + \underbrace{16^0 \times 4}_{=4} = \boxed{676}$$

Problem 2.2: The conversion is performed below.

$$(67\text{E})_{10} = \underbrace{16^2 \times 6}_{=1536} + \underbrace{16^1 \times 7}_{=112} + \underbrace{16^0 \times 14}_{=14} = \boxed{1662}$$

Problem 2.3: The conversion is performed below.

$$(892E)_{10} = \underbrace{16^3 \times 8}_{=32,768} + \underbrace{16^2 \times 9}_{=2304} + \underbrace{16^1 \times 2}_{=32} + \underbrace{16^0 \times 14}_{=14} = \boxed{35,118}$$

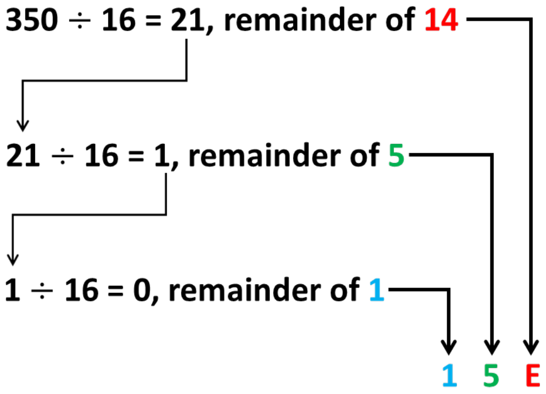
Problem 2.4: The conversion is performed below.,

$$(ABCD)_{10} = \underbrace{16^3 \times 10}_{=40,960} + \underbrace{16^2 \times 11}_{=2816} + \underbrace{16^1 \times 12}_{=192} + \underbrace{16^0 \times 13}_{=13} = \boxed{43,981}$$

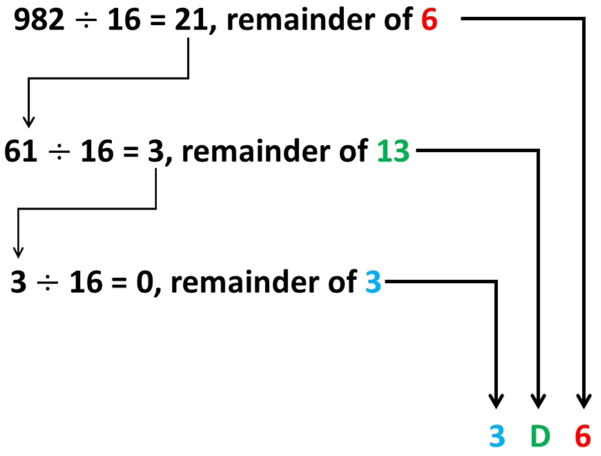
Problem 2.5: The conversion is performed below.

$$(A3F.D)_{10} = \underbrace{16^2 \times 10}_{=2560} + \underbrace{16^1 \times 3}_{=48} + \underbrace{16^0 \times 15}_{=15} + \underbrace{16^{-1} \times 13}_{=0.8125} = \boxed{2623.8125}$$

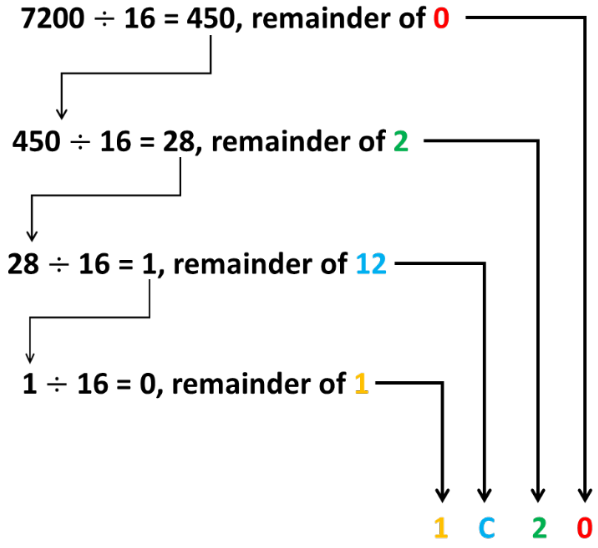
Problem 2.6: A decimal number is converted to a hexadecimal number via the divide-by-16 process. Dividing 350 by 16 once yields 21 with a remainder of 14. The remainder is taken as the last digit in the hexadecimal number we are looking for, as shown below, while the quotient is carried over and used in a second division by 16. Upon dividing the quotient 21 by 16, we obtain a remainder of 5, which is taken as the second (right to left) digit in the hexadecimal number we are looking for; the quotient is 1, which we carry over to a third division by 16. Upon dividing the quotient 1 by 16, we obtain a remainder of 1, which is taken as the third (right to left) digit in the hexadecimal number we are looking for; since the quotient of this last operation is zero, there is no need to proceed to a fourth division by 16. Gathering the three remainders, we conclude that the hexadecimal representation of 350 is $(15E)_{16}$.



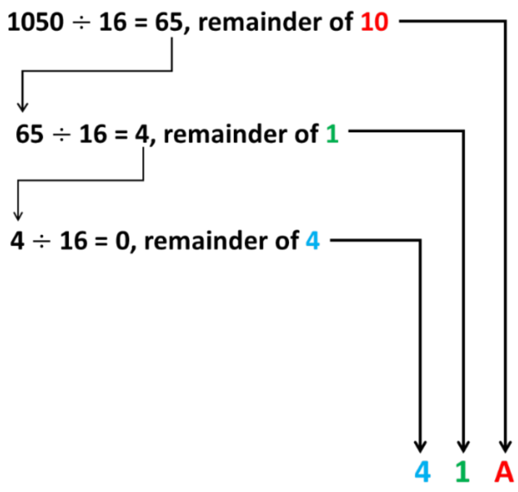
Problem 2.7: Here, the procedure is identical to the one adopted in Problem 2.6. The hexadecimal representation of 982 is found to be $(3D6)_{16}$.



Problem 2.8: Applying the divide-by-16 algorithm, the hexadecimal representation of 7200 is found to be $(1C20)_{16}$.



Problem 2.9: This number differs from the previous ones because it has a fractional component. To convert the integer part of the number, we proceed as we would in the previous exercises.



Now, converting the fractional part of the number involves successive *multiplication* by 16. We begin by multiplying the fractional part, 0.1, by 16, which yields 1.6. Then, we take the integer part as the first fractional digit of the hexadecimal representation. In the case at hand, the number to the left of the dot in 1.6 is 1; accordingly, the first fractional digit in the hexadecimal form of 1050.1 is 1, so we can write $(1050.1)_{10} = (41A.1)_{hex}$. To find the next fractional digit, we take the fractional part of the number we got when finding the first fractional digit and multiply it by 16. We had 1.6, so we take 0.6 and obtain 0.6 × 16 = 9.6. The integer part of this result constitutes the next decimal digit of the hexadecimal representation; such a digit is 9, so we may write $(1050.1)_{10} = (41A.19)_{hex}$. Finding the third fractional digit is no different: the fractional part of the previous multiplication by 16 is 0.6; multiplying this result by 16 gives 9.6; thus, the hexadecimal representation is updated as $(1050.1)_{10} = (41A.199)_{hex}$. Proceed similarly and you'll find that the next fractional digit is a 9 yet again, signaling that this is a continued fraction. Thus, we conclude that

$$(1050.1)_{10} = (41A.1999...)_{16}$$

Problem 2.10: The procedure to convert π, an irrational number, to hexadecimal form is similar to the procedure adopted in Problem 2.9. We begin by taking the fractional part of π and multiplying it by 16, that is, 16 × 0.1415927 = 2.2654825. The integer part, which is a 2, will be the first fractional digit in the hexadecimal representation:

$$(3.1415927)_{10} = (3.2...)_{16}$$

Next, multiplying 0.2654825 by 16 gives 4.2477200. The integer part, which is a 4, will be the second fractional digit of the hexadecimal representation:

$$(3.1415927)_{10} = (3.24...)_{16}$$

Next, multiplying 0.2477200 by 16 gives 3.96352. The integer part, which is a 3, will be the third fractional digit of the hexadecimal representation:

$$(3.1415927)_{10} = (3.243...)_{16}$$

Next, multiplying 0.96352 by 16 gives 15.4163. The integer part, which is a 15, will be the fourth fractional digit in the hexadecimal representation:

$$(3.1415927)_{10} = (3.243F...)_{16}$$

Next, multiplying 0.4163 by 16 gives 6.6608. The integer part, which is a 6, will be the fifth fractional digit in the hexadecimal representation:

$$(3.1415927)_{10} = (3.243F6...)_{16}$$

Since π is an irrational number, it has infinite, non-periodic fractional digits, be it in decimal or hexadecimal representation. We truncate our approximation at the fifth fractional space and write

$$(\pi)_{10} = (3.243F6...)_{16}$$

P.3 ➡ *Solution*

Problem 3.1: One easy way to find the 16's complement of a hexadecimal number is to first find the 15's complement, then add 1 to the result. The 15's complement can be determined by subtracting the starting number from *FFFF*, as shown.

FFFF	−
B2FA	
4D05	

Accordingly, the 16's complement is 4D05 + 0001 = 4D06.

Problem 3.2: Number *B* translates as 1011; number 2 becomes 0010; letter F translates as 1111; letter A translates as 1010. Thus, B2FA is translated as 1011_0010_1111_1010.

Problem 3.3: One simple way to find the 2's complement of a binary number is to first determine the 1's complement and then add 1 to the result. The 1's complement can be determined by swapping the zeros for ones and vice versa, as shown below for B2FA:

1011_0010_1111_1010
↓
0100_1101_0000_0101 (1's complement)

Then, we add 1 to obtain the 2's complement:

0100_1101_0000_0101
↓
0100_1101_0000_0110 (2's complement)

Problem 3.4: To convert the 2's complement to a hexadecimal, we have

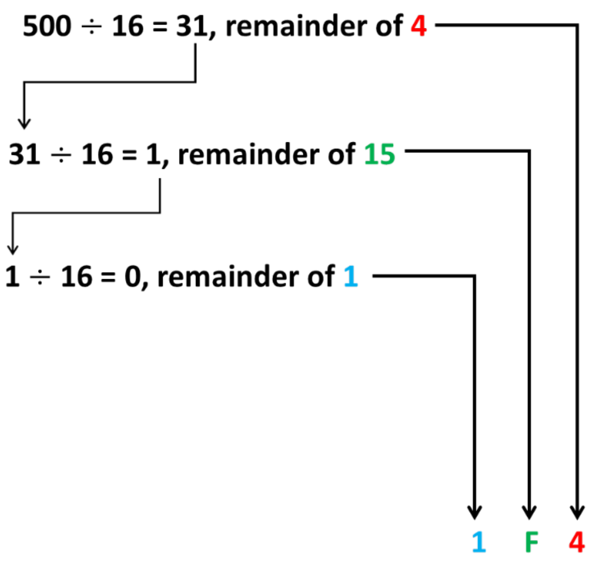
0100_1101_0000_0110
↓
4_D_0_6 (2's complement converted to hexadec.)

The hexadecimal form of the 2's complement is 4D06, which happens to be the 16's complement of the hexadecimal number we began with (B2FA).

P.4 ➡ *Solution*

Problem 4.1: Since a hex number represents 4 bits, we'd need 5 hex characters to specify a 20-bit address code.

Problem 4.2: The 501st address is simply the decimal number 500 written in hexadecimal form. To convert decimal number 500 to hex, use the division-by-16 method devised in Problem 2. The address of the memory in question is found to be $(1F4)_{hex}$.



Problem 4.3: If the data item addresses begin at FF2, the 50th data item will be located at FF2 + $(49)_{16}$ = 103D. To convert this result to decimal form, we add powers of 16 as we did in Problem 2,

$$(103D)_{16} = \underbrace{16^3 \times 1}_{=4096} + \underbrace{16^2 \times 0}_{=0} + \underbrace{16^1 \times 3}_{=48} + \underbrace{16^0 \times 13}_{=13} = \boxed{(4{,}157)_{10}}$$

P.5 ➡ *Solution*

Problem 5.1: One straightforward way to obtain the 9's complement of a decimal number is to subtract each digit from 9, as illustrated in the table below.

(1) Number made up of 9's	(2) Decimal number	(3) Col. (1) minus col. (2)
9	2	7
9	5	4
9	0	9
9	0	9
9	0	9
9	0	9
9	0	9
9	0	9

The 9's complement of 25,000,000 is given by column (3), that is, 74,999,999. To determine the 10's complement of the number, add 1 to the 9's complement obtained above.

74,999,999	+
1	
75,000,000	

Problem 5.2: We apply the same technique employed in Problem 5.1.

(1) Number made up of 9's	(2) Decimal number	(3) Col. (1) minus col. (2)
9	9	0
9	9	0
9	9	0
9	9	0
9	5	4
9	5	4
9	0	9
9	0	9

The 9's complement of 99,995,500 is given by column (3), that is, 00,004,499. The 10's complement of the number is computed below.

00,004,499	+
1	
00,004,500	

Problem 5.3: We apply the same technique employed in 5.1.

(1) Number made up of 9's	(2) Decimal number	(3) Col. (1) minus col. (2)
9	5	4
9	2	7
9	7	2
9	8	1
9	4	5
9	6	3
9	3	6
9	0	9

The 9's complement of 52,784,630 is given by column (3), that is, 47,215,369. The 10's complement of the number is computed below.

47,215,369	+
1	
47,215,370	

Problem 5.4: We apply the same technique employed in 5.1.

(1) Number made up of 9's	(2) Decimal number	(3) Col. (1) minus col. (2)
9	6	3
9	3	6
9	3	6
9	2	7

9	5	4
9	6	3
9	0	9
9	0	9

The 9's complement of 63,325,600 is given by column (3), that is, 36,674,399. The 10's complement of the number is computed below.

36,674,399	+
1	
36,674,400	

Problem 5.5: We first establish the 10's complement of the subtrahend.

3409	→	03409	→	99999 – 03409 = 96590 (9's complement of subtrahend)	96590 + 1 = 96591 (10's complement of subtrahend)

Instead of subtracting 3,409 from 6,428, we could add the 10's complement of 3,409 to 6,428.

$$06428 - 03409 = 06428 + 96591 = 103019$$

Lastly, we exclude the end carry digit 1, leaving us with 03019 or, written correctly, 3,019. The occurrence of an end carry indicates that the result is positive, that is,

$$6428 + 96591 = +\not{1}03019 = \boxed{+3,019}$$

Problem 5.6: We first establish the 10's complement of the subtrahend.

1800	→	01800	→	99999 – 01800 = 98199 (9's complement of subtrahend)	98199 + 1 = 98200 (10's complement of subtrahend)

Instead of subtracting 1,800 from 125, we could add the 10's complement of 1,800 to 125.

$$00125 - 01800 = 00125 + 98200 = 98325$$

Since there is no end carry, the result is negative and given by the 10's complement of 98325, namely, 1675.

$$125 - 1,800 = \boxed{-1,675}$$

Problem 5.7: We begin by computing the 10's complement of the subtrahend.

6152	→	06152	→	99999 – 06152 = 93847 (9's complement of subtrahend)	93847 + 1 = 93848 (10's complement of subtrahend)

Instead of subtracting 6,152 from 2,043, we could add the 10's complement of 6,152 to 2,043.

$$02043 - 06152 = 02043 + 93848 = 95891$$

Since there is no end carry, the result is negative and given by the 10's complement of 95891, namely, 4109.

$$2,043 - 6,152 = \boxed{-4,109}$$

Problem 5.8: We begin by computing the 10's complement of the subtrahend.

745	→	0745	→	9999 – 0745 = 9254 (9's complement of subtrahend)	9254 + 1 = 9255 (10's complement of subtrahend)

Instead of subtracting 745 from 1,631, we could add the 10's complement of 745 to 1,631.

$$1,631 - 745 = 1631 + 9254 = 10885$$

The occurrence of an end carry digit indicates that the result is positive.

$$1631 + 9254 = +\not{1}0885 = \boxed{+885}$$

P.6 ➡ *Solution*

Problem 6.1: We begin by computing the 2's complement of the subtrahend.

10001	→	010001	→	111111 – 010001 = 101110 (1's complement of subtrahend)	101110 + 1 = 101111 (2's complement of subtrahend)

Instead of subtracting 10001 from 10011, we add the 2's complement of 10001 to 10011.

$$10011-10001=010011+101111=1000010$$

The occurrence of an end carry indicates that the result is positive.

$$010011+101111=+\not{1}000010=\boxed{+00010}$$

Alternatively, we could represent the positive sign with a leftmost zero digit and write **0**_00010.

Problem 6.2: We begin by computing the 2's complement of the subtrahend.

100011	→	100011	→	1111111–100011 = 011100 (1's complement of subtrahend)	011100 + 1 = 011101 (2's complement of subtrahend)

Instead of subtracting 100011 from 100010, we add the 2's complement of 100011 to 100010.

$$100010-100011=100010+011101=111111$$

Since there is no end carry, the result is negative and given by the 2's complement of 100001, namely −000001. Alternatively, we could represent the negative sign with a leftmost one digit and write **1**_000001.

Problem 6.3: We begin by computing the 2's complement of the subtrahend.

101000	→	101000	→	111111– 101000 = 010111 (1's complement of subtrahend)	010111 + 1 = 011000 (2's complement of subtrahend)

Since there is no end carry digit, the result is negative. Instead of subtracting 101000 from 1001, we add the 2's complement of 101000 to 1001.

$$1001-101000=001001+011000=100001$$

Since there is no end carry, the result is negative and given by the 2's complement of 100001, namely, −011111. Alternatively, we could represent the negative sign with a leftmost one digit and write **1**_011111.

Problem 6.4: We begin by computing the 2's complement of the subtrahend.

10101	→	010101	→	111111– 010101 = 101010 (1's complement of subtrahend)	101010 + 1 = 101011 (2's complement of subtrahend)

Instead of subtracting 10101 from 110000, we add the 2's complement of 10101 to 110000.

$$110000-10101=110000+101011=1011011$$

The occurrence of an end carry indicates that the result is positive.

$$110000+101011=+\not{1}011011=\boxed{+011011}$$

Alternatively, we could represent the positive sign with a leftmost zero digit and write 0_011011.

P.7 ➡ *Solution*

Problem 7.1: Referring to Table 1, we can represent 842 in BCD form as

$$(842)_{\text{BCD}}=1000\ 0100\ 0010$$

Likewise for 537,

$$(537)_{\text{BCD}}=0101\ 0011\ 0111$$

We move on to add the digits, proceeding as we would in a typical binary addition.

842	1000	0100	0010	+
537	0101	0011	0111	
	1101	0111	1001	

Herein lies a crucial difference: note that the result underscored in red is greater than 1010; such numbers have no meaning in BCD, and are modified by the addition of a $(0110)_2$, or a 6, which converts the digit to a number that is convertible to BCD code and produces a carry as required. We proceed accordingly in the calculation at hand:

842	1000	0100	0010	+
537	0101	0011	0111	
+	1101	0111	1001	
	0110			
0001	0011	0111	1001	
1	3	7	9	

Problem 7.2: The procedure is no different from the one in Problem 7.1; the difference here is that all three BCD digits need correction with a 0110 before we arrive at the final result. The final BCD code is 0001 0001 0010 0010, which is equivalent to decimal 1122.

285	0010	1000	0101	+
837	1000	0011	0111	
	1	1		
+	1010	1011	1100	
	0110	0110	0110	
0001	0001	0010	0010	
1	1	2	2	

P.8 ➡ *Solution*

Problem 8.1: Refer to Table 1. To represent 541 in excess-3 form we write

$$(541)_{\text{XS-3}} = 1000\ 0111\ 0100$$

Proceeding similarly with 736, we write

$$(736)_{\text{XS-3}} = 1010\ 0110\ 1001$$

To find the sum of the two numbers, we first add the representations of each digit as we would in a binary number addition.

541	1000	0111	0100	+
736	1010	0110	1001	
0001	0010	1101	1101	

At this point, how we proceed depends on the results of the operation. If no carries were produced, as in the case of the two results underscored in blue, we *subtract* $(0011)_2$, or a decimal 3, from the result. If carries were produced, we *add* $(0011)_2$.

541	1000	0111	0100	+
736	1010	0110	1001	
+ 0001	+ 0010	− 1101	− 1101	
0011	0011	0011	0011	
0100	0101	1010	1010	
1	2	7	7	

Put together, the four codes in blue represent the result in excess-3, which is equivalent to decimal 1277.

Problem 8.2: The procedure is identical to the one adopted in Problem 11.1. The final XS-3 code result is 0100 0100 0100 0110, which is equivalent to decimal 1113.

806	1011	0011	1001	+
307	0110	0011	1010	
	0001	0001	0111	0011
	+ 0011	+ 0011	− 0011	+ 0011
	0100	0100	0100	0110
	1	1	1	3

P.9 ➡ *Solution*

Problem 9.1: To write the BCD for 5,137, refer to Table 1. The corresponding code is 0101 001 0011 0111, as shown below.

0101	0001	0011	0111
5	1	3	7

Problem 9.2: One way to establish the excess-3 code for 5,137 is to add $(0011)_2$, or a 3, to each digit of its BCD code representation. The XS-3 code for the number in question is found to be 1000 0100 0110 1010, as shown below.

	0101	0001	0011	0111
+	0011	0011	0011	0011
	1000	0100	0110	1010
	5	1	3	7

Problem 9.3: The excess-3 code is self-complementing. This means that 1's complement of an excess-3 number is the excess-3 code for the 9's complement of the corresponding decimal number. The excess-3 code for decimal 5137 was obtained above; from the reasoning posed here, it follows that the XS-3 representation of 4862, which is the 9's complement of 5137, can be established by swapping 1's for 0's and vice versa in the XS-3 representation of 5137:

(XS-3 for 5137)	1000	0100	0110	1010
(XS-3 for 4862)	0111	1011	1001	0101
	4	8	6	2

Problem 9.4: To write the number in question in 2421 code, refer to Table 1. The code we aim for is found to be 1100 0010 1011 0001, as shown.

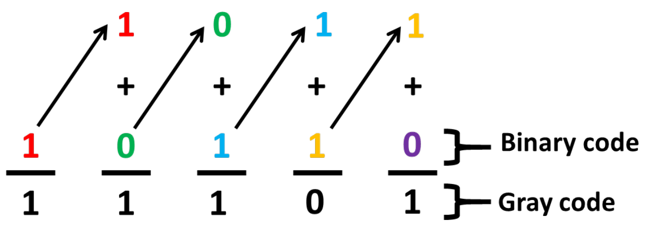
1100	0010	1011	0001
6	2	5	1

It is easy to see that the 9's complement of 6,251 is the decimal number 3,748. From the self-complementary property of the code in question, the representation of 3,748 in 2421 code can be obtained by swapping the 0's for 1's and vice versa (i.e., by finding the 1's complement) in the 2421-code representation of 3,748.

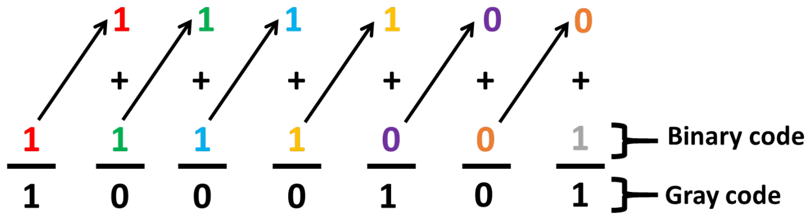
(2-4-2-1 for 6251)	1100	0010	1011	0001
(2-4-2-1 for 3748)	0011	1101	0100	1110
	3	7	4	8

P.10 ➡ *Solution*

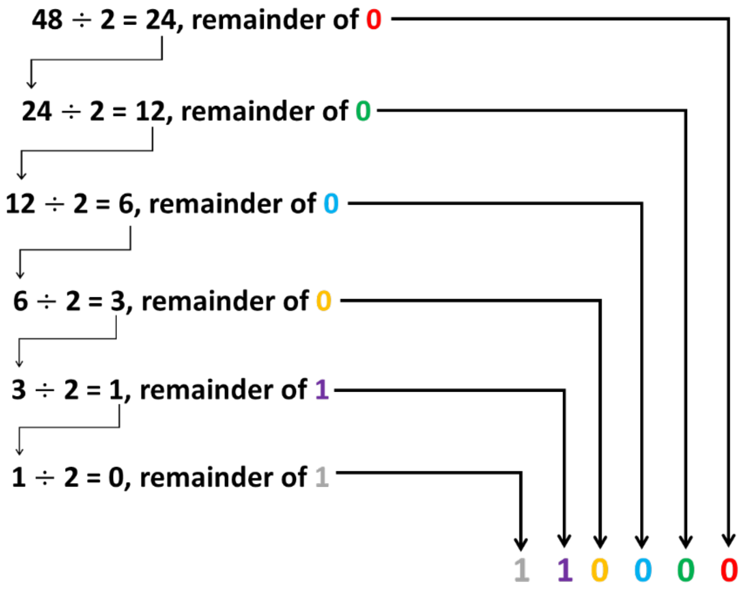
Problem 10.1: The computation of the Gray code that corresponds to 101110 is shown below. The pertaining Gray code is found to be 11101.



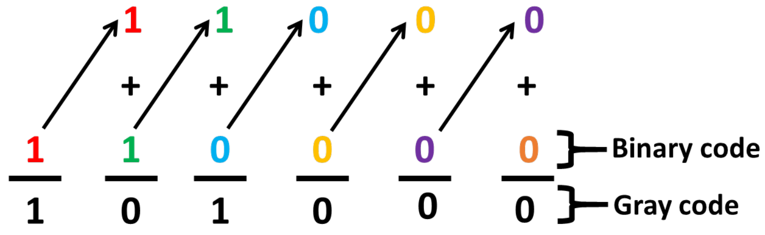
Problem 10.2: The computation of the Gray code that corresponds to 1111001 is shown below. The pertaining Gray code is found to be 1000101.



Problem 10.3: To write 48 in Gray code, we first write it in binary. The conversion is performed below.



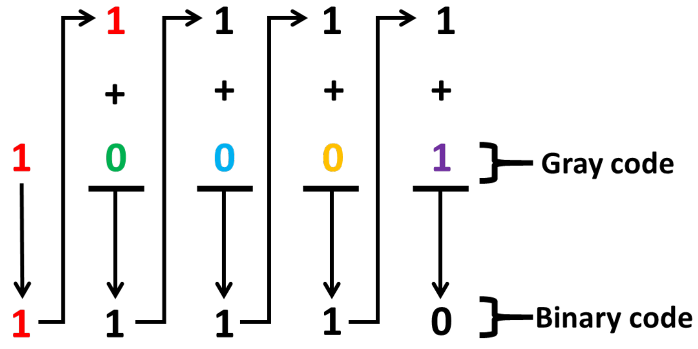
Equipped with 48 in binary form, we use binary-to-Gray conversion. The Gray code for 48 is found to be 101000, as shown.



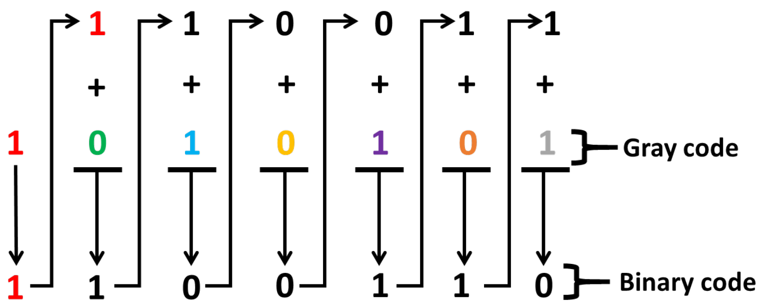
Problem 10.4: The procedure used to convert a Gray code to a binary number is similar to the reverse operation. For convenience, we summarize the steps below:

1. The MSB of the binary number is the same in the binary number as in the Gray code.
2. The second MSB of the binary number is obtained by adding the first MSB of the binary number to the second MSB of the Gray code number. Ignore carries if any occur.
3. The third MSB is obtained by adding the second MSB of the binary number to the third MSB of the Gray code. Again, ignore carries if any occur.
4. Continue performing additions in consonance with steps 2 and 3 until the least significant bit of the code is reached.

The computation of the binary code that corresponds to Gray code 10001 is shown below. The pertaining binary code is found to be 11110.



Problem 10.5: The computation of the binary code that corresponds to Gray code 1010101 is shown below. The pertaining binary code is found to be 1100110.



P.11 ➡ *Solution*

Problem 11.1: The code translates as "BillGates".

1000010	1101001	1101100	1101100	1000111	1100001	1110100	1100101	1110011
B	i	l	l	G	a	t	e	s

Problem 11.2: As the reader should know, an ASCII character has seven bits, and an eighth is added to assign parity; even parity is obtained by assigning a zero to the leftmost bit of the character. To translate "G. Boole" to this character code, refer to Table 3.

G	1000111
.	0101110
[space]	0100000
B	1000010
o	1101111
o	1101111
l	1101100
e	1100101

The desired string is then "**1000111010111001000000100001011011111110111111011001100101**"

P.12 ➡ *Solution*

Problem 12.1: First, we convert the digit pairs to 8-bit binary codes, as shown.

53 F4 E5 76 E5 4A EF 62 73
01110011 11110100 11100101 01110110 11100101 01001010 11101111 01100010 01110011

To establish the ASCII characters associated with each 8-bit code, we need only consider the final seven digits; below, these are highlighted in green, along with the corresponding ASCII entry, which can be read from Table 3. The code translates as "*steveJobs*".

01110011	11110100	11100101	01110110	11100101	01001010	11101111	01100010	01110011
s	t	e	v	e	J	o	b	s

Problem 12.2: The parity of the code is determined by the number of 1's in each byte. Note that all bytes have an odd number of 1's; it follows that the parity used is *odd*.

REFERENCES

- MANO, M.M. and CILETTI, M.D. (2004). *Digital Design.* 4th edition. Upper Saddle River: Pearson.
- SEDHA, R.S. (2004). *A Textbook of Digital Electronics.* New Delhi: S. Chand and Company.
- TANNA, S. (2018). *Advanced Binary for Programming and Computer Science*. India: Answers 2000 Limited.