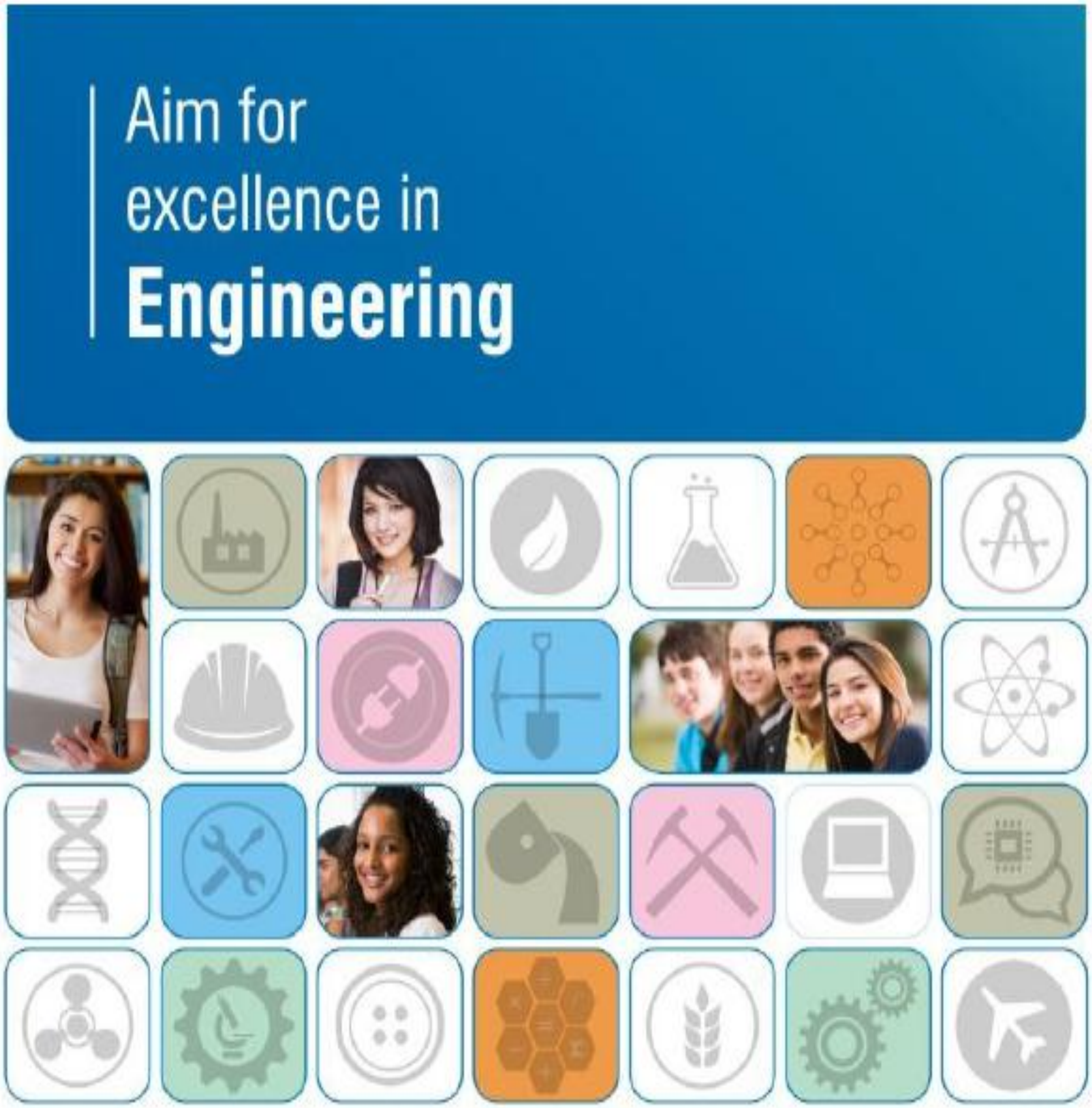


INSTRUMENTATION ENGINEERING
Digital Logic

CAREER AVENUES

INDEX

CHAPTER 1- NUMBER SYSTEMS

INTRODUCTION

"A set of values used to represent different quantities is known as Number System".
For example, a number system can be used to represent the number of students in a class or number of viewers watching a certain TV program etc. The digital computer represents all kinds of data and information in binary numbers. It includes audio, graphics, video, text and numbers. The total number of digits used in a number system is called its **base or radix**. The base is written after the number as subscript such as 512_{10}

Types of Number System

1) Decimal Number System.
2) Binary Number System
3) Hexadecimal Number System.
4) Octal Number System.

DECIMAL NUMBERS

The number system that we use in our day-to-day life is the decimal number system. Decimal number system has base or radix 10 as it uses 10 digits from 0 to 9. In decimal number system, the successive positions to the left of the decimal point represent units, tens, hundreds, thousands and so on.
Each position represents a specific power of the base (10). For example, the decimal number 1234 consists of the digit 4 in the units position, 3 in the tens position, 2 in the hundreds position, and 1 in the thousands position, and its value can be written as
$(1 \times 1000) + (2 \times 100) + (3 \times 10) + (4 \times l)$
$= (1 \times 10^3) + (2 \times 10^2) + (3 \times 10^1) + (4 \times l0^0)$
$= 1000 + 200 + 30 + 1$
$= 1234$

1) The column weights of decimal numbers are **powers of ten** that increase from right to left beginning with $10^0 =1$:…10^5 10^4 10^3 10^2 10^1 10^0.
2) For **fractional** decimal numbers, the column weights are **negative powers** of ten that decrease from left to right: 10^2 10^1 10^0. 10^{-1} 10^{-2} 10^{-3} 10^{-4} …
3) Decimal numbers can be expressed as the **sum of the products** of each digit times the column value for that digit. Thus, the number **9240** can be expressed as: $(\mathbf{9} \times 10^3) + (\mathbf{2} \times 10^2) + (\mathbf{4} \times 10^1) + (\mathbf{0} \times 10^0)$ or $\mathbf{9} \times 1{,}000 + \mathbf{2} \times 100 + \mathbf{4} \times 10 + \mathbf{0} \times 1$

Example: Express the number **480.52** as the sum of values of each digit.
Solution: **480.52** $= (\mathbf{4} \times 10^2) + (\mathbf{8} \times 10^1) + (\mathbf{0} \times 10^0) + (\mathbf{5} \times 10^{-1}) + (\mathbf{2} \times 10^{-2})$

BINARY NUMBERS

Characteristics

- Uses two digits, 0 and 1.
- Also called base 2 number system
- Each position in a binary number represents a 0 power of the base (2). Example 2^0
- Last position in a binary number represents a × power of the base (2). Example 2^x where x represents the last position - 1.

EXAMPLE
Binary Number: 10101_2

Calculating Decimal Equivalent:

Step	Binary Number	Decimal Number
Step 1	10101_2	$((1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$
Step 2	10101_2	$(16 + 0 + 4 + 0 + 1)_{10}$
Step 3	10101_2	21_{10}

Note: 10101_2 is normally written as 10101.
For fractional binary numbers, the column weights are negative powers of two that decrease from left to right:
$2^2\ 2^1\ 2^0.\ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ \ldots$

HEXADECIMAL NUMBERS

Characteristics

- Uses 10 digits and 6 letters: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- Letters represents numbers starting from 10. A = 10. B = 11, C = 12, D = 13, E = 14, F = 15.
- Also called base 16 number system
- Each position in a hexadecimal number represents a 0 power of the base (16). Example 16^0
- Last position in a hexadecimal number represents a x power of the base (16). Example 16^x where x represents the last position - 1.

EXAMPLE
Hexadecimal Number: $19FDE_{16}$
Calculating Decimal Equivalent:

Step	Binary Number	Decimal Number
Step 1	$19FDE_{16}$	$((1 \times 16^4) + (9 \times 16^3) + (F \times 16^2) + (D \times 16^1) + (E \times 16^0))_{10}$
Step 2	$19FDE_{16}$	$((1 \times 16^4) + (9 \times 16^3) + (15 \times 16^2) + (13 \times 16^1) + (14 \times 16^0))_{10}$
Step 3	$19FDE_{16}$	$(65536+ 36864 + 3840 + 208 + 14)_{10}$
Step 4	$19FDE_{16}$	106462_{10}

Note: $19FDE_{16}$ is normally written as 19FDE.

OCTAL NUMBERS

Characteristics

- Uses eight digits, 0,1,2,3,4,5,6,7.
- Also called base 8 number system
- Each position in a octal number represents a 0 power of the base (8). Example 8^0
- Last position in a octal number represents a × power of the base (8). Example 8^x where x represents the last position - 1.

EXAMPLE
Octal Number: 12570_8
Calculating Decimal Equivalent:

Step	**Octal Number**	**Decimal Number**
Step 1	12570_8	$((1 \times 8^4) + (2 \times 8^3) + (5 \times 8^2) + (7 \times 8^1) + (0 \times 8^0))_{10}$
Step 2	12570_8	$(4096 + 1024 + 320 + 56 + 0)_{10}$
Step 3	12570_8	5496_{10}

Note: 12570_8 is normally written as 12570.

Binary Numbers					
4	**2**	**1**		**Octal Value**	**Decimal Value**
0	0	0		0	0
0	0	1		1	1
0	1	0		2	2
0	1	1		3	3
1	0	0		4	4
1	0	1		5	5
1	1	0		6	6
1	1	1		7	7

Binary Numbers						
8	**4**	**2**	**1**		**Hexadecimal Value**	**Decimal Value**
0	0	0	0		0	0
0	0	0	1		1	1
0	0	1	0		2	2
0	0	1	1		3	3
0	1	0	0		4	4
0	1	0	1		5	5
0	1	1	0		6	6
0	1	1	1		7	7
1	0	0	0		8	8
1	0	0	1		9	9
1	0	1	0		A	10
1	0	1	1		B	11
1	1	0	0		C	12
1	1	0	1		D	13
1	1	1	0		E	14
1	1	1	1		F	15

Representation of Number System

As the decimal number is a weighted number, converting from decimal to binary (base 10 to base 2) will also produce a weighted binary number with the right-hand most bit being the **Least Significant Bit** or **LSB**, and the left-hand most bit being the **Most Significant Bit** or **MSB**, and we can represent this as:

Representation of a Binary Number

MSB	**Binary Digit**							**LSB**
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

NUMBER SYSTEM CONVERSIONS

Any numbering system can be summarized by the following relationship:

$N = b_i q^i$

where: N is a real positive number
b is the digit
q is the base value
and integer (i) can be positive, negative or zero

$N = b_n q^n \ldots b_3 q^3 + b_2 q^2 + b_1 q^1 + b_0 q^0 + b_{-1} q^{-1} + b_{-2} q^{-2} \ldots$ etc.

BINARY -TO-DECIMAL CONVERSIONS (base 2 to base 10)

The decimal equivalent of a binary number can be determined by adding the column values of all of the bits that are **1** and discarding all of the bits that are 0.

Example: Convert the binary number **100101.01** to decimal.
Solution: Start by writing the column weights; then add the weights that correspond to each **1** in the number.

$\mathbf{2^5}$	$\mathbf{2^4}$	$\mathbf{2^3}$	$\mathbf{2^2}$	$\mathbf{2^1}$	$\mathbf{2^0}$**.**	$\mathbf{2^{-1}}$	$\mathbf{2^{-2}}$
32	**16**	**8**	**4**	**2**	**1.**	**½**	**¼**
1	**0**	**0**	**1**	**0**	**1.**	**0**	**1**
32	**+4**	**+1**	**+¼**	**=**	**37¼**		

Example: convert $(1000100)_2$ to decimal
= 64 + 0 + 0+ 0 + 4 + 0 + 0
= $(68)_{10}$

DECIMAL-TO-BINARY CONVERSIONS (base 10 to base 2)

We can convert a decimal whole number to binary by reversing the procedure. Write the decimal weight of each column and place 1's in the columns that sum to the decimal number.
Example: Convert the decimal number 49 to binary.

Solution: Write down column weights until the last number is larger than the one you want to convert.

2^6	2^5	2^4	2^3	2^2	2^1	2^0.
64	32	16	8	4	2	1.
0	1	1	0	0	0	1.

Instead of division, multiplication by 2 is carried out and the integer part of the result is saved and Placed after the decimal point. The fractional part is again multiplied by 2 and the process repeated.

Example: convert $(68)_{10}$ to binary
Solution:
68/2 = 34 remainder is 0
34/ 2 = 17 remainder is 0
17 / 2 = 8 remainder is 1
8 / 2 = 4 remainder is 0
4 / 2 = 2 remainder is 0
2 / 2 = 1 remainder is 0
1 / 2 = 0 remainder is 1
Answer = 1 0 0 0 1 0 0
Note: the answer is read from bottom (MSB) to top (LSB) as 10001002

DECIMAL FRACTION-TO-BINARY FRACTION CONVERSIONS

You can convert a decimal fraction to binary by repeatedly multiplying the fractional results of successive multiplications by 2. The carries form the binary number.

Example: Convert the decimal fraction 0.188 to binary by repeatedly multiplying the fractional results by 2.
Solution:

$0.188 \times 2 = 0.376$	carry = 0	MSB
$0.376 \times 2 = 0.752$	carry = 0	
$0.752 \times 2 = 1.504$	carry = 1	
$0.504 \times 2 = 1.008$	carry = 1	
$0.008 \times 2 = 0.016$	carry = 0	LSB

Answer = .00110 (for five significant digits)

Example: convert (0.68)10 to binary fraction.

Solution:
0.68 × 2 = 1.36 integer part is 1
0.36 × 2 = 0.72 integer part is 0
0.72 × 2 = 1.44 integer part is 1
0.44 × 2 = 0.88 integer part is 0
Answer = 0. 1 0 1 0…..

Example: convert (68.68)10 to binary equivalent.
Answer = 1 0 0 0 1 0 0 . 1 0 1 0….

CONVERTING BINARY TO HEX AND OCTAL

Conversion of binary numbers to octal and hex simply requires grouping bits in the binary numbers into groups of three bits for conversion to octal and into groups of four bits for conversion to hex.

Groups are formed beginning with the LSB and progressing to the MSB.

Thus, $\underline{11}\ \underline{100}\ \underline{111}_2 = 347_8$ $\quad$ $\underline{1110}\ \underline{0111}_2 = E7_{16}$

$\underline{11}\ \underline{100}\ \underline{010}\ \underline{101}\ \underline{010}\ \underline{010}\ \underline{001}_2 = 3025221_8$ $\quad$ $\underline{1}\ \underline{1000}\ \underline{1010}\ \underline{1000}\ \underline{0111}_2 = 18A87_{16}$

CONVERTING HEX TO DECIMAL (base 16 to base 10)

Example: Express **$1A2F_{16}$** in decimal and binary.

Solution: Start by writing the column weights:

4096	**256**	**16**	**1**
1	**A**	**2**	**F**

1(4096) + 10(256) +2(16) +15(1) = 6703_{10}

Example: convert $(F4C)_{16}$ to decimal

= (F × 162) + (4 × 161) + (C × 160)
= (15 × 256) + (4 × 16) + (12 × 1)

CONVERTING OCTAL TO DECIMAL (base 8 to base 10)

Example: Express **3702_8** in decimal

Solution: Start by writing the column weights:

512	64	8	1
3	**7**	**0**	**2**

3(512) + 7(64) +0(8) +2(1) = 1986_{10}

CONVERTING DECIMAL TO HEX (base 10 to base 16)

Example: convert $(4768)_{10}$ to hex.
Solution: Start by dividing with 16

= 4768 / 16 = 298 remainder 0
= 298 / 16 = 18 remainder 10 (A)
= 18 / 16 = 1 remainder 2
= 1 / 16 = 0 remainder 1

Answer: 1 2 A 0 = $(12A0)_{16}$
Note: the answer is read from bottom to top, same as with the binary case.
= 3840 + 64 + 12 + 0
= (3916)10

CONVERTING DECIMAL TO OCTAL (base 10 to base 8)

Example: convert $(177)_{10}$ to octal
Solution: Start by dividing with 16

177 / 8 = 22 remainder is 1
22 / 8 = 2 remainder is 6
2 / 8 = 0 remainder is 2
Answer = 2 6 1 = $(261)_8$

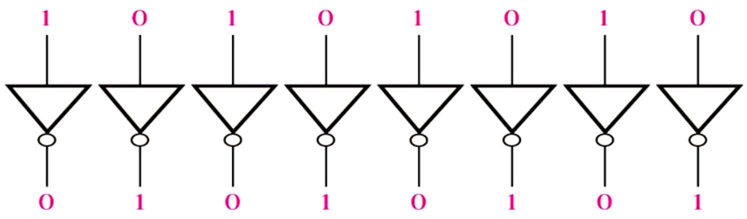
Note: the answer is read from bottom to top as $(261)_8$, the same as with the binary case.

Conversion of decimal fraction to octal fraction is carried out in the same manner as decimal to binary except that now the multiplication is carried out by 8. To convert octal to hex first convert octal to binary and then convert binary to hex. To convert hex to octal first convert hex to binary and then convert binary to octal.

COMPLEMENTS OF BINARY NUMBERS

1'S COMPLEMENT

The 1's complement of a binary number is just the inverse of the digits. To form the 1's complement, change all **0**'s to **1**'s and all **1**'s to **0**'s.
For example, the 1's complement of **11001010** is 00110101
In digital circuits, the 1's complement is formed by using inverters:



2’s COMPLEMENT:

The **2**’s complement of a binary number is found by adding **1** to the **LSB** of the **1**’s complement.

```
Recall that the 1’s complement of            11001010 is
                                             00110101 (1’s complement)
To form the 2’s complement, add 1:            +1
                                             ________
                                             00110110 (2’s complement)
```

Tip: To find 2’s compliment directly copy LSB to answer if it is 0 copy next LSB and so on till you get 1. After copying 1 do 1’s compliment of rest of MSB

GENERAL "B'S COMPLEMENT" NUMBERS FOR BASE B AND BASE 2

Everything flows in general the same as above except for the changes shown in the table below. In this table, "b" in column 2 represents a general base b number. Column 3 represents binary numbers.

Row 2 gives the representation of the number -1 (with leading digits equal to b-1, b the base).
Row 3 gives the equation for complementing each of the digits (in each case, subtract the digit from b-1, b the base)
Row 4 gives examples of positive numbers (with leading zeros extending leftward to infinity).
Row 5 shows the complement of the number in row 4 (and shows that the sum of the two equals zero)

	Base 10's (base - 1) = 10 - 1 = 9	Base b (base - 1) = (b - 1) = B	Base 2 (base - 1) = (2 - 1) = 1
Number = -1	$\cdots 9 \cdots 99999999999$	$\cdots B \cdots BBBBBBBBBBB$	$\cdots 1 \cdots 11111111111$
Complement d_k of digit c_k	$d_k = 9 - c_k$	$d_k = (b-1) - c_k$ $= B - c_k$	$d_k = 1 - c_k$ If $c_k = 1,\ d_k = 0$ If $c_k = 0,\ d_k = 1$
Example of 8 digit positive number (leading 0 required)	00034857	For base 5, as an example, 00340240	01011010
Example of 8 digit negative number (leading (b-1) required) with magnitude equal to positive number in row above	00034857 -> 99965143 00034857 + 99965143 00000000	00340240 -> 44104210 00340240 + 44104210 00000000	01011010 -> 10100110 01011010 + 10100110 00000000

ARITHMETIC FOR VARIOUS NUMBER SYSTEM

BINARY ADDITION

The **rules** for binary addition are:

0 + 0 = 0	Sum = 0, carry = 0
0 + 1 = 1	Sum = 1, carry = 0
1 + 0 = 1	Sum = 1, carry = 0
1 + 1 = 10	Sum = 0, carry = 1

When an input (carry = 1) due to a previous result, the **rule** is:

1 + 1 + 1 = 11	Sum = 1, carry = 1

Example: Add the binary numbers **00111** and **10101** and show the equivalent decimal addition

Solution:

```
 111
00111     7
10101    21
11100    28
```

BINARY SUBTRACTION

The **rules** for binary subtraction are:

0 - 0 = 0
1 - 1 = 0
1 - 0 = 1
10 **-** 1 = 1 with a borrow of **1**

Example: Subtract the binary number **00111** from **10101** and show the equivalent decimal subtraction.

Solution:

```
 111
10101    21
00111     7
01110    14
```

HEXADECIMAL ADDITION

Examples

```
Carry                  0          00          00
Addend  3B2      3B2        3B2        3B2
Augend  + 41C    + 41C      + 41C      + 41C
Sum                E          CE         7CE

Carry                  1          1          1          1
Addend  A27      A27        A27        A27        A27
Augend  + C3B    + C3B      + C3B      + C3B      + C3B
Sum                2          62         662        1662
```

HEXADECIMAL SUBTRACTION

Uses the same principle of "borrowing" that decimal and binary subtraction uses.

Example

borrow			
Minuend	6E	6E	6E
Subtrahend	- 29	- 29	-29
Difference		5	45

Minuend	AC3	AC3	AC3	AC3
Subtrahend	- 604	- 604	- 604	- 604
Difference		F	BF	4BF

OCTAL ADDITION

Examples

carry		1		
Addend	127	127	127	127
Augend	+ 42	+ 42	+ 42	+ 42
Sum		1	71	171

carry		1	11	11	1
Addend	1777	1777	1777	1777	1777
Augend	+ 777	+ 777	+ 777	+ 777	+ 777
Sum		6	76	776	2776

OCTAL SUBTRACTION

This is performed exactly like binary and decimal subtraction with the borrowing technique. Whenever the subtrahend is larger than the minuend, a 1 is borrowed from the next column.

Example

Minuend	124	124	124
Subtrahend	- 63	- 63	-63
Sum		1	41

SIGNED BINARY NUMBERS

There are several ways to represent signed binary numbers. In all cases, the **MSB** in a signed number is the **sign bit**, that tells you if the number is **positive** or **negative**.

Computers use a **modified 2**'s complement for signed numbers. **Positive** numbers are stored in ***true*** form (with a 0 for the sign bit) and **negative** numbers are stored in *complement* form (with a **1** for the sign bit).

For example, the **positive** number **58** is written using **8**-bits in **true** (**un-complemented**) as:

00111010 (**true** form).

Sign bit

Magnitude bits

Negative numbers are written as the **2**'s complement of the corresponding positive number.
The negative number -58 is written as:

11000110 (**complement** form)

Sign bit

Magnitude bits

An easy way to read a signed number that uses this notation is to assign the **sign bit** a column **weight** of **-128** (for an **8**-bit number). Then add the column weights for the **1**'s.

Example: Assuming that the sign bit = -128, show that **11000110** = **-58** as a **2**'s complement signed number.

Solution: Column weights

-128	64	32	16	8	4	2	1.	
1	**1**	0	0	0	**1**	**1**	0	
-128	**+64**				**+4**	**+2**		**= -58**

Arithmetic Operations with Signed Numbers

Addition

Using the **signe**d number notation with **negative** numbers in **2**'s complement form simplifies addition and subtraction of signed numbers.

Rules for addition: Add the two **signed** numbers. **Discard** any **final carries**. The result is in **signed** form.

Examples:

00011110 = +30
00001111 = +15
00101101 = 45

00001110 = +14
11101111 = -17
11111101 = -3

11111111 = -1
11111000 = -8
1 11110111 = -9

Discard Final carry

Overflow Condition

When two numbers are added and the **number of bits** required to represent the sum **exceeds** the number of bits in the two numbers, an **overflow** results as indicated by an **incorrect sign bit**. An overflow can occur only when both numbers have the same sign.

Two examples are:

01000000 = +128
01000001 = +129
10000001 = -126

Discard carry

10000001 = -127
10000001 = -127
100000010 = -2

Wrong! The answer is incorrect and the sign bit has changed

Subtraction

Rules for subtraction: 2's complement the subtrahend and add the numbers. Discard any final carries. The result is in signed form.

Examples:

00011110 = +30
11110001 = -15
1 00001111 = +15

Discard Final

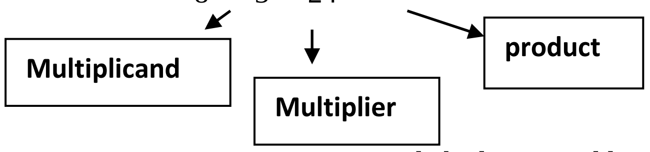
11111111 = -1
00001000 = +8
1 00000111 = +7

Discard Final

MULTIPLICATION

The numbers in a multiplication are the multiplicand, the multiplier, and the product. These are illustrated in the following decimal multiplication:

8 × 3 = 24



The multiplication operation in most computers is accomplished using addition in two different ways: direct addition and partial products.

In direct addition method, you add the multiplicand a number of times equal to the multiplier. The disadvantage of this approach is that it becomes very lengthy if the multiplier is a large number. For example, to multiply 350 × 75, you must add 350 to itself 75 times.

The partial product method is more common, the multiplicand is multiplied by each multiplier digit begins with the least significant digit, and the result is called a partial product. Each successive partial product is moved (shifted) one place to the left, and when all partial products have been produced, they are added to get the final product.

The rules for partial product are:
Determine if the sign of the multiplicand and multiplier are the same or different:
If the signs are the same, the product is positive
If the signs are different, the product is negative
Change any negative number to true form (uncomplemented).

Starting with the LSB of the multiplier, generate partial products, shift each successive partial product one bit to the left, and add each successive partial product to the sum of the previous partial products to get the final product.
If the sign bit that was determined in step 1 is negative, take the 2' complement of the product. If positive, leave the product in true form
Attach the sign bit to the product.
Example: Multiply the signed binary numbers: 01010011 (multiplicand) and 11000101 (multiplier).
Solution:
Since the two numbers have different signs, the sign bit of the product will be negative (1).
Take the 2'complement of the multiplier to put it in true form
11000101 --> 00111011
The multiplication proceeds as follows. Notice that only the magnitude bits are used

	1010011	Multiplicand
x	0111011	Multiplier
	1010011	1st partial product
+	1010011	2nd partial product
	11111001	Sum of 1st and 2nd
+	0000000	3rd partial product
	011111001	Sum
+	1010011	4th partial product
	1110010001	Sum
+	1010011	5th partial product
	100011000001	Sum
+	1010011	6th partial product
	1001100100001	Sum
+	0000000	7th partial product
	1001100100001	Final (Sum) product

1. Since the final product should be negative, take the 2'complemrnt of the final product 1001100100001 0110011011111
2. Attach the sign bit $\longrightarrow$ 10110011011111

DIVISION

The numbers in a division are the dividend, the divisor, and the quotient. These are illustrated in the following standard format:

dividend / divisor = quotient

The division operation in computers is accomplished using subtraction. Since subtraction is done with an adder, division can also be accomplished with adder.

The result of a division is called the quotient; the quotient is the number of times that the divisor will go into the dividend. This means that the divisor can be subtracted from the dividend a number of times equal to the quotient, as illustrated by dividing 21 by 7 using subtraction operations:

21 – 7 = 14 – 7 = 7 – 7 = 0

In this simple example, the divisor was subtracted from the dividend 3 times before a reminder of 0 was obtained. Therefore, the quotient is 3.

When two binary numbers are divided, both numbers must be in true (uncomplemented) form. The basic rules in a division process are as follows:

1. Determine if the sign of the dividend and divisor are the same or different:
    - If the signs are the same, the quotient is positive
    - If the sign are different, the quotient is negative
2. Subtract the divisor from the dividend using 2's complement addition to get the first partial remainder and add 1 to the quotient. If the partial reminder is positive, go to step to step 3. If the partial reminder is zero or negative, the division is complete.
3. Subtract the divisor from the partial reminder and add 1 to the quotient. If the result is positive, repeat for the next partial reminder. If the result is zero or negative, the division is complete.
4. Continue to subtract the divisor from the dividend and the partial reminder until there is a zero or negative result.
5. Count the number of times that the divisor is subtracted and you have the quotient.

Example: Divide 01100100 by 00011001

Solution:

1. The sign of both numbers are positive, so the quotient will be positive. The quotient is initially zero: 00000000.

2. Subtract the divisor from the dividend using 2's complement addition (remember that the final carries are discarded).

```
   01100100      dividend
 + 11100111      2'complement of divisor
   01001011      positive 1st partial reminder
```

   Add 1 to quotient: 00000000 + 00000001 = 00000001

3. Subtract the divisor from the 1st partial reminder using 2's complement addition.

```
   01001011      1st partial reminder
 + 11100111      2'complement of divisor
   00110010      positive 2nd partial reminder
```

   Add 1 to quotient: 00000001 + 00000001 = 00000010

Subtract the divisor from the 2nd partial reminder using 2's complement addition.

```
   00110010      2st partial reminder
 + 11100111      2'complement of divisor
```

00011001 positive 3rd partial reminder

Add 1 to quotient: 00000010 + 00000001 = 00000011

5. Subtract the divisor from the 3rd partial reminder using 2's complement addition.

```
   00011001       3rd partial reminder
+  11100111       2'complement of divisor
         00       zero reminder
```

Add 1 to quotient: 00000011 + 00000001 = 00000100 (final quotient), the process is complete.

FLOATING POINT NUMBERS

Floating point notation is capable of representing very large or small numbers by using a form of scientific notation. A 32-bit single precision number is illustrated:

S	E (8 bits)	F (23 bits)

Sign bit   Biased exponent (+127)   Magnitude with MSB dropped

Example: Express the speed of light, *c*, in single precision floating point notation.
($c = 0.2998 \times 10^9$)

Solution:

Binary: $c = 0001\ 0001\ 1101\ 1110\ 1001\ 0101\ 1100\ 0000_2$

Scientific notation: $c = 1.001\ 1101\ 1110\ 1001\ 0101\ 1100\ 0000 \times 2^{28}$

Floating point notation:

0	10011011	001 1101 1110 1001 0101 1100

S = 0 because the number is positive.

$E = 28 + 127 = 155_{10} = 1001\ 1011_2$.

F = the next 23 bits after the first 1 is dropped

BINARY CODES

BINARY CODED DECIMAL (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

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.
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.

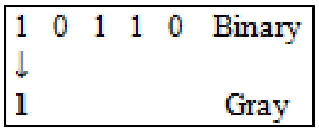
Decimal	Binary	Gray
0	000	0000
1	001	0001
2	010	0011
3	011	0010
4	100	0110
5	101	0111
6	110	0101
7	0111	0100

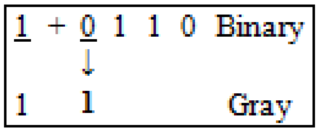
Binary-to-Gray Conversion

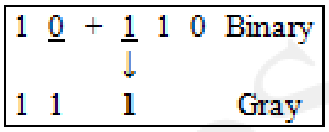
Following is algorithm to convert a binary value to its corresponding standard Gray code value :

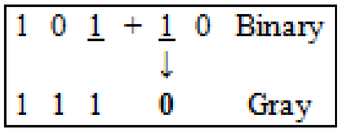
1. Retain the MSB.
2. From left to right, add each adjacent pair of binary code bits to get the next Gray code bit, discarding the carry.

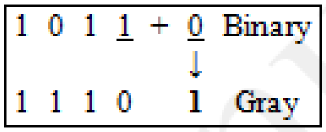
The following example shows the conversion of binary number $(10110)_2$ to its corresponding standard Gray code value, $(11101)_{Gray}$.









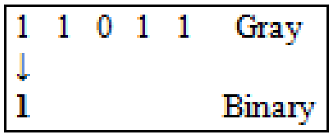


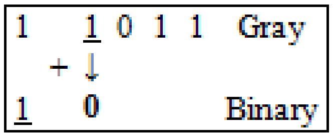
Gray-to-Binary Conversion

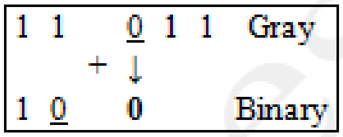
The algorithm to convert a standard Gray code value to its corresponding binary value is as follows:

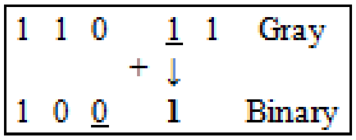
1. Retain the MSB.
2. From left to right, add each binary code bit generated to the Gray code bit in the next position, discarding the carry.

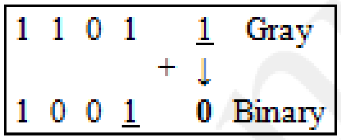
The following example shows the conversion of the standard Gray code value $(11011)_{Gray}$ to its corresponding binary value, $(10010)_2$.











EXCESS-3 CODE

The Excess-3 code uses a bias value of three. It means we have to add 3(0011) to the corresponding BCD code. Hence the codes for the ten digits 0, 1, …, 9 are 0011, 0100, …, 1100 respectively. The decimal number 294 would be represented as 0101 1100 0111.

84-2-1 CODE

The 84-2-1 code uses the weights of 8, 4, -2 and -1 in the coding. The decimal number 294 would be represented as 0110 1111 0100.

2421 CODE

The 2421 code uses the weights of 2, 4, 2 and 1 in the coding. According to the weights, certain digits may have alternative codes. For instance, the digit 3 could be represented as 0011 or 1001. However, we pick the former in the standard 2421 coding, so that the codes for the first five digits 0 – 4 begin with 0, whereas the codes for the last five digits 5 – 9 begin with 1. The decimal number 294 would be represented as 0010 1111 0100.

ASCII CODE

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.

ERROR DETECTION

- Transmitted binary information is subject to noise that could change bits 1 to 0 and vice versa
- An *error detection code* is a binary code that detects digital errors during transmission
- The detected errors cannot be corrected, but can prompt the data to be retransmitted
- The most common error detection code used is the *parity bit*
- A parity bit is an extra bit included with a binary message to make the total number of 1's either odd or even

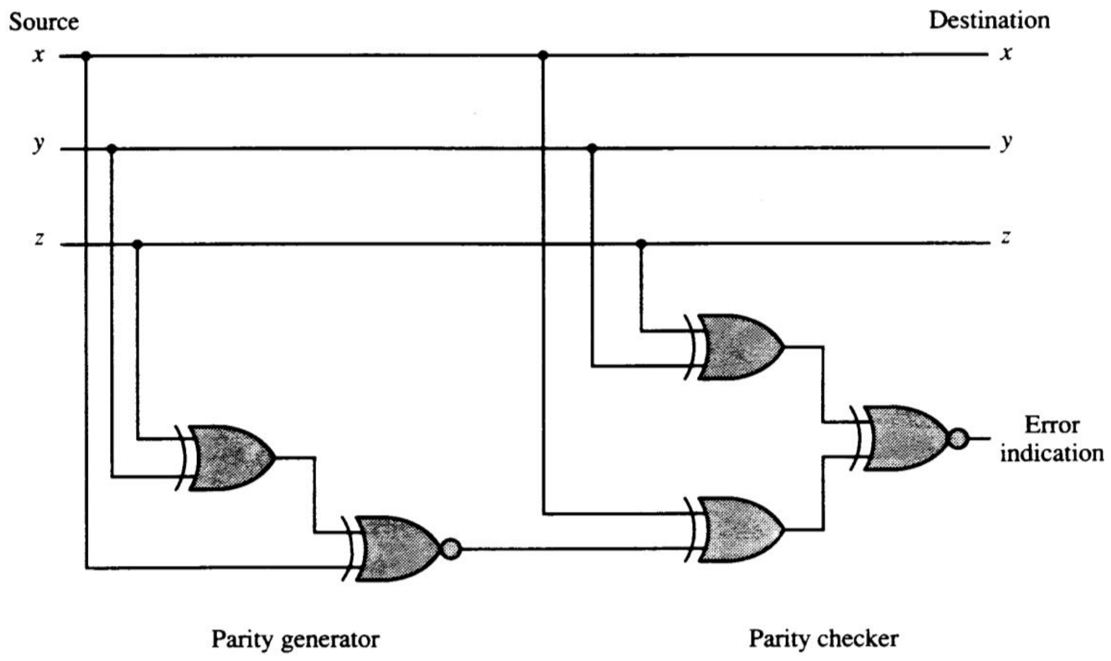
TABLE 3-7 Parity Bit Generation

Message *xyz*	*P*(odd)	*P*(even)
000	1	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	0	1

- The P(odd) bit is chosen to make the sum of 1's in all four bits odd
- The even-parity scheme has the disadvantage of having a bit combination of all 0's
- Procedure during transmission:
    - At the sending end, the message is applied to a *parity generator*
    - The message, including the parity bit, is transmitted
    - At the receiving end, all the incoming bits are applied to a *parity checker*
    - Any odd number of errors are detected

- Parity generators and checkers are constructed with XOR gates (odd function)
- An odd function generates 1 iff an odd number if input variables are 1

Figure 3-3 Error detection with odd parity bit.



Quick Review Questions

2-1. Convert the binary number 1011011 to its decimal equivalent.
a. 5 b. 63 c. 91 d. 92 e. 139

2-2. What is the weight of the digit '3' in the base-7 number 12345?
a. 3 b. 7 c. 14 d. 21 e. 49

2-3. Which of the following has the largest value?
a. $(110)_{10}$ b. $(10011011)_2$ c. $(1111)_5$ d. $(9A)_{16}$ e. $(222)_8$

2-4. If $(321)_4 = (57)_{10}$, what is the decimal equivalent of $(321000000)_4$?
a. 57×10^4 b. 57×10^6 c. 57×4^4 d. 57×4^6 e. 57^4

2-5. Convert each of the following decimal numbers to binary (base two) with at most eight digits in the fractional part, rounded to eight places.
a. 2000 b. 0.875 c. 0.07 d. 12.345

2-6. Convert each of the decimal numbers in Question 2-5 above to septimal (base seven) with at most six digits in the fractional part, rounded to six places.

2-7. Convert each of the decimal numbers in Question 2-5 above to octal (base eight) with at most four digits in the fractional part, rounded to four places.

2-8. Convert each of the decimal numbers in Question 2-5 above to hexadecimal (base sixteen) with at most two digits in the fractional part, rounded to two places.

2-9. Which of the following octal values is equivalent to the binary number $(110001)_2$?
a. $(15)_8$ b. $(31)_8$ c. $(33)_8$ d. $(49)_8$ e. $(61)_8$

2-10. Convert the binary number $(1001101)_2$ to
a. quaternary b. octal c. decimal d. hexadecimal

2-11. What is $(1011)_2 \times (101)_2$?
a. $(10000)_2$ b. $(110111)_2$ c. $(111111)_2$ d. $(111011)_2$ e. $(101101)_2$

2-12. Perform the following operations on binary numbers.
a. $(10111110)_2 + (10001101)_2$
b. $(11010010)_2 - (01101101)_2$
c. $(11100101)_2 - (00101110)_2$

2-13. In a 6-bit 2's complement binary number system, what is the decimal value represented by $(100100)_{2s}$?
a. –4 b. 36 c. –36 d. –27 e. –28

2-14. In a 6-bit 1's complement binary number system, what is the decimal value represented by $(010100)_{1s}$?
a. –11 b. 43 c. –43 d. 20 e. –20

2-15. What is the range of values that can be represented in a 5-bit 2's complement binary system?

a. 0 to 31 b. –8 to 7 c. –8 to 8 d. –15 to 15 e. –16 to 15

2-16. In a 4-bit 2's complement scheme, what is the result of this operation: $(1011)_{2s} + (1001)_{2s}$?

a. 4 b. 5 c. 20 d. –12 e. overflow

2-17. Assuming a 6-bit 2's complement system, perform the following subtraction operations by converting it into addition operations:

a. $(011010)_{2s} - (010000)_{2s}$
b. $(011010)_{2s} - (001101)_{2s}$
c. $(000011)_{2s} - (010000)_{2s}$

2-18. Assuming a 6-bit 1's complement system, perform the following subtraction operations by converting it into addition operations:

a. $(011111)_{1s} - (010101)_{1s}$
b. $(001010)_{1s} - (101101)_{1s}$
c. $(100000)_{1s} - (010011)_{1s}$

2-19. Which of the following values cannot be represented accurately in the 8-bit sign-and-magnitude fixed-point number format shown in Figure 2.4?

a. 4 b. –29.5 c. 20.2 d. –3.75 e. 12.25

2-20. What does 1 110 1001 represent in this floating-point number scheme: 1-bit sign, 3-bit normalized mantissa, followed by 4-bit 2's complement exponent?

a. 0.125×2^9 b. -0.125×2^9 c. -0.75×2^{-1} d. -0.75×2^{-6} e. -0.75×2^{-7}

2-21. How to represent $(246)_{10}$ in the following system/code?

a. 10-bit binary b. BCD c. Excess-3 d. 2421 code e. 84-2-1 code

2-22. The decimal number 573 is represented as 1111 0110 1011 in an unknown self-complementing code. Find the code for the decimal number 642.

2-23. Convert $(101011)_2$ to its corresponding Gray code value.

a. $(101011)_{Gray}$ b. $(010100)_{Gray}$ c. $(110010)_{Gray}$ d. $(111110)_{Gray}$ e. $(43)_{Gray}$

2-24. Convert $(101011)_{Gray}$ to its corresponding binary value.

a. $(101011)_2$ b. $(010100)_2$ c. $(110010)_2$ d. $(111110)_2$
e. $(010101)_2$

Answers to Quick Review Questions

2-1. (c) 2-2. (e) 2-3. (c) 2-4. (d)

2-5. (a) $(2000)_{10} = (11111010000)_2$ (b) $(0.875)_{10} = (0.111)_2$
(c) $(0.07)_{10} = (0.00010010)_2$ (d) $(12.345)_{10} = (1100.01011000)_2$

2-6. (a) $(2000)_{10} = (5555)_7$ (b) $(0.875)_{10} = (0.606061)_7$
(c) $(0.07)_{10} = (0.033003)_7$ or $(0.033004)_7$
(d) $(12.345)_{10} = (15.226223)_7$

2-7. (a) $(2000)_{10} = (3720)_8$ (b) $(0.875)_{10} = (0.7)_8$
(c) $(0.07)_{10} = (0.0437)_8$ (d) $(12.345)_{10} = (14.2605)_8$

2-8. (a) $(2000)_{10} = (7D0)_{16}$ (b) $(0.875)_{10} = (0.E)_{16}$
(c) $(0.07)_{10} = (0.12)_{16}$ (d) $(12.345)_{10} = (C.58)_{16}$

2-9. (e)

2-10. (a) $(1031)_4$ (b) $(115)_8$ (c) $(77)_{10}$ (d) $(4D)_{16}$

2-11. (b)

2-12. (a) $(101001011)_2$ (b) $(01100101)_2$ (c) $(10110111)_2$

2-13. (e) 2-14. (d) 2-15. (e) 2-16. (e)

2-17. (a) $(001010)_{2s} = (10)_{10}$ (b) $(001101)_{2s} = (13)_{10}$ (c) $(110011)_{2s} = -(13)_{10}$

2-18. (a) $(001010)_{1s} = (10)_{10}$ (b) $(011100)_{1s} = (28)_{10}$ (c) overflow

2-19. (c) 2-20. (e)

2-21. (a) $(0011110110)_2$ (b) $(0010\ 0100\ 0110)_{BCD}$ (c) $(0101\ 0111\ 1001)_{Excess\text{-}3}$
(d) $(0010\ 0100\ 1100)_{2421}$ (e) $(0110\ 0100\ 1010)_{84\text{-}2\text{-}1}$

2-22. 642 = 0100 0000 1001

2-23. (d) 2-24 .(c)

SUMMARY:

- To convert binary to decimal Start by writing the column weights; then adds the weights that correspond to each **1** in the number.
- To convert decimal to any number system, divide by base of that number system and reminders from bottom to top will be answer.
- To convert decimal fraction to any number system, multiply by base of that number system and carry from top to bottom will be answer.
- Binary to hex/octal can be converted by grouping 4/3 bits and writing equivalent hex/octal digits.
- Hex/octal to binary can be converted by writing 4/3 bit binary value corresponding to each digit.
- The 1's complement of a binary number is just the inverse of the digits
- The **2**'s complement of a binary number is **1**'s complement+1. It represent negative number in signed arithmetic i.e. 2's compliment of +8 in binary gives -8.
- In signed number MSB is 0 for positive number and 1 for negative number
- Floating point numbers

S	E (8 bits)	F (23 bits)
Sign bit	Biased exponent (+127)	Magnitude

- B
  inary-to-Gray conversion: Retain the MSB. From left to right, add each adjacent pair of binary code bits to get the next Gray code bit, discarding the carry.
- Gray-to-Binary Conversion: Retain the MSB. From left to right, add each binary code bit generated to the Gray code bit in the next position, discarding the carry.
- *Excess-3 code* uses a bias value of three. i.e. *Excess-3* = BCD+0011

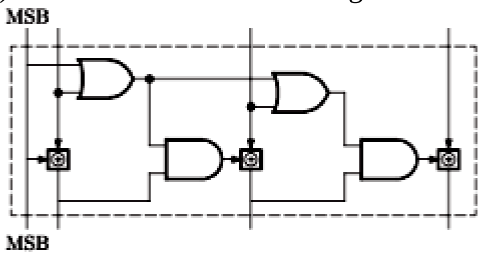
GATE Tip

Number system conversions and arithmetic's can be done with scientific calculators and they are rarely asked in GATE. Concentrate more on 1's and 2's compliments, Binary codes, and number systems with base other than 2, 8 and 16

Simple way to do signed arithmetic is discard sign (MSB) convert rest of bit in decimal (you can use calculator for this) now apply +/- sign according to sign bit. now perform require arithmetic calculation and covert answer in binary and at the end replace sign by sign bit.

GATE Questions

1) The circuit shown in figure converts



A) BCD to binary code
B) Binary to excess - 3 code
C) Excess -3 to gray code
D) Gray to Binary code

2) The range of signed decimal numbers that can be represented by 6-bits 1's complement number is
A) -31 to +31 B) -63 to +63 C) -64 to +63 D) -32 to +31

3) 11001, 1001, 111001 correspond to the 2's complement representation of which one of the following sets of number
A) 25,9, and 57 respectively
B) -6, -6, and -6 respectively
C) -7, -7 and -7 respectively
D) -25, -9 and -57 respectively

4) Decimal 43 in Hexadecimal and BCD number system is respectively
A) B2, 0100 011 B) 2B, 0100 0011 C) 2B, 0011 0100 D) B2, 0100 0100

5) X =01110 and Y =11001 are two 5·bit binary numbers represented in 2's complement format. The sum of X and Y represented in 2's complement format using 6 bit is
A) 100111 B) 001000 C) 000111 D) 101001

6) 4-bit 2's complement representation of a decimal number is 1000. The number is
A) + 8 B) 0 C) - 7 D) – 8

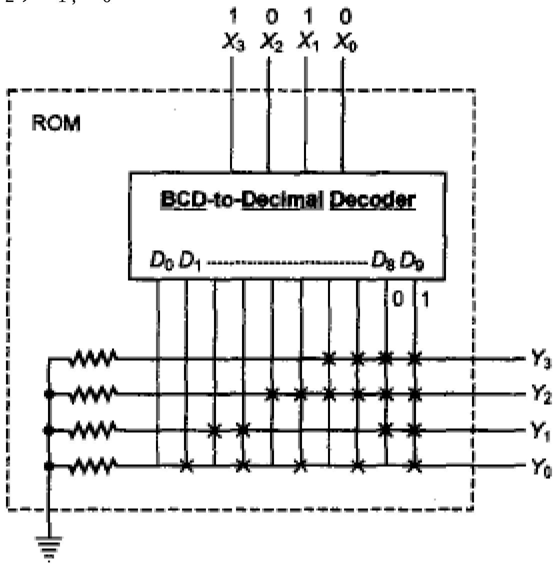
7) The 2's complement representation of - 17 is
A) 101110 B) 101111 C) 111110 D) 110001

8) The two numbers represented in signed 2's complement form are P =11101101 and Q = 11100110. If Q is subtracted from P, the value obtained in signed 2's complement form is
A) 100000111 B) 00000111 C) 11111001 D) 111111001

9) A new Binary Coded Pentary (BCP) number system is proposed in which every digit of a base-5 number is represented by its corresponding 3-bit binary code. For example, the base-5 number 24 will be represented by its BCP code 01 01 00. In this numbering system, the BCP code 10001 0011 001 corresponds to the following number in base-5 system

A) 423 B) 1324 C) 2201 D) 4231

10) If the inputs X_3 , X_2 , X_1 , X_0 to the ROM in figure are 8-4-2-1 BCD number then the output Y_3, Y_2 , Y_1 , Y_0 are



A) Gray code numbers B) 2-4-2-1 BCD numbers

C) Excess-3 code numbers D) None of these

Answer Key / Explanatory Answer

1-D 2-A 3-C 4-B 5-C 6-D 7-B 8-B 9-D 10-B

1. Ans: D
Explanation:
Let input be 1010; output will be 1101 Let input be 0110; output will be 0100
Thus it convert gray to Binary code.

2. Ans: A
Explanation:
The range of signed decimal number that can be represented by n-bits
1’s complement number is $–(2^{n-1}-1)$ to $+(2^{n-1}-1)$
Thus for n=6 we have
Range = $-(2^{6-1}-1)$ to $+(2^{6-1}-1)$
Hence (A) is correct answer.

3. Ans: C
Explanation:

$$\begin{array}{rrr} 11001 & 1001 & 111001 \\ 00110 & 0110 & 000110 \\ +1 & +1 & +1 \\ \hline 00111 & 0111 & 000111 \\ 7 & 7 & 7 \end{array}$$

Thus 2’s complement of 11001 1001 and 111001 is 7. So, the number given in the question are 2’s complement correspond to -7

4. Ans: B
Explanation:
Dividing 43 by 16 we get Quotient 2 and Reminder 11
11 in decimal is equivalent is B in hexamal.
Thus, $43_{10} \leftrightarrow 2B_{16}$
Now $4_{10} \leftrightarrow 0100_2$
$3_{10} \leftrightarrow 0011_2$
Thus $43_{10} \leftrightarrow 01000011_{BCD}$
Hence (B) is correct answer.

5. Ans: C
Explanation:
In 2's complements form, the MSB represents the sign bit.
X =01110 =14
Y=11001=-7
X+Y=14+(-7)=7
2's complement representation is 000111.

6. Ans: D
Explanation:
As the MSB is 1 i.e., the decimal representation is negative and its value is 8. 2's complement representation of 1000 is - 8.

7. Ans: B
Explanation:

(17)10 =(010001), 2's complement 01+ 17will beequivalentto-17 so -17 =101111

8. Ans: B
Explanation:

In signed 2's complement, the M5B represents the Signal
P =11101101=(-19)10
Q =11100110 =(- 26)10
P -Q =-19 -(- 26) =(7)10
(17)10 =00000111

9.Ans: D
Explanation:

in binary coded pentary (BCD) number system,
24 is represented as

$$\begin{array}{cc} \underline{010} & \underline{100} \\ \downarrow & \downarrow \\ 2 & 4 \end{array}$$

Thus ,

$$\begin{array}{cccc} \underline{100} & \underline{010} & \underline{011} & \underline{001} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 4 & 2 & 3 & 1 \end{array}$$

Corresponds to 4231

10. Ans: B
Explanation:

X_3	X_2	X_1	X_0	Y_3	Y_2	Y_1	Y_0
1	0	0	1	1	1	1	1
1	0	0	0	1	1	1	0
0	1	1	0	1	1	0	0

Converts 2-4-2-1 BCD number.