NAME: PRERNA SUNIL JADHAV
SAPID: 60004220127
BRANCH: COMPUTER ENGINEERING

COURSE CODE: DJ19CEC805
COURSE: DIGITAL ELECTRONICS

TERMWORK ASSIGNMENT

Q1 convert and present $(45)_{10}$ in Hexadecimal, Binary and Octal.

Soln:

To convert $(45)_{10}$ to Binary divide 45 by 2 as Binary has base 2 until quotient is 0

	Quotient	Remainder
$\frac{45}{2}$	22	1
$\frac{22}{2}$	11	0
$\frac{11}{2}$	5	1
$\frac{5}{2}$	2	1
$\frac{2}{2}$	1	0
$\frac{1}{2}$	0	1

$\therefore (45)_{10} = (101101)_2$

To convent into octal,

$(45)_{10} = (101101)_2 = (55)_8$

5 5

To convert into Hexadecimal,

$(45)_{10} = (101101)_2 = (2D)_{16}$

2 D

Q2

b) Perform following operation using 2's compliment:
$(86)_{10} - (22)_{10}$

Soln:

converting $(86)_{10}$ and $(22)_{10}$ into Binary

	Quotient	Remainder
$\frac{86}{2}$	43	0
$\frac{43}{2}$	21	1
$\frac{21}{2}$	10	1
$\frac{10}{2}$	5	0
$\frac{5}{2}$	2	1
$\frac{2}{2}$	1	0
$\frac{1}{2}$	0	1

	Quotient	Remainder
$\frac{22}{2}$	11	0
$\frac{11}{2}$	5	1
$\frac{5}{2}$	2	1
$\frac{2}{2}$	1	0
$\frac{1}{2}$	0	1

$\therefore (86)_{10} = (1010110)_2$

$\therefore (22)_{10} = (10110)_2$

Now,

Performing 2's complement of $(0010110)_2$

1's complement → 1101001

$+ \quad 1$

2's complement → 1101010

Now, performing subtraction using 2's complement

$\therefore (86)_{10} = (1010110)_2$

$\therefore (22)_{10} = (10110)_2$ → 2's complement = $(1101010)_2$

$$\begin{array}{rl} & 1\ 0\ 1\ 0\ 1\ 1\ 0 \\ (+) & 1\ 1\ 0\ 1\ 0\ 1\ 0 \\ \hline \text{carry} \rightarrow \boxed{1} & 1\ 0\ 0\ 0\ 0\ 0\ 0 \end{array}$$

Here, carry generated is 1, which indicates that the answer is positive and need to ignore it.

$\therefore (1000000)_2 = (64)_{10}$

$\therefore (86)_{10} - (22)_{10} = (64)_{10} = (1000000)_2$

Q3

b) short note on HAMMING code.

Soln: Hamming code is a block code that is capable of detecting up to two simultaneous bit error and correcting single bit errors.
It was developed by R.W. Hamming for error correction. In this coding Method, the source encodes the messages by inserting redundant bits within the message. These redundant bits are extra bits that are generated & inserted at a specific positions in the message itself to enable error detection and correction. when the destination receives the message, it performs recalculations to detect errors and find the bit position that has error.

Encoding a message by Hamming code

Step 1: Calculating the number of redundant bit

Step 2: ~~Calcu~~ Positioning the redundant bits - placed at bit positions of powers of 2 i.e., 1 (r_0), 2 (r_1), 4 (r_2), 8 (r_3), 16 (r_4) etc.

Step 3: Calculating the value of each redundant bit
These bits are parity bits, which are the extra bits that makes the no. of 1's either even or odd. There are two types of parity -

1) Even parity: Here, the total no. of bits in the message is made even

2) odd parity: Here, the total no. of bits in the message is made odd.

Decoding a message in Hamming code, here receiver performs recalculations to detect, errors and correct them. Steps are -

Step 1: Calculation of the no. of redundant bits
$2^r \geq m + r + 1$, where m = no. of data bits,
r = no. of redundant bits.

Step 2: Positioning the redundant bits
The r redundant bits placed at bit position of powers of 2, i.e, 1, 2, 4, 8, 16 etc

Step 3: Parity checking
Parity bits are calculated based upon the data bits & the redundant bits using the same rule as during generation. Thus,
c_1 = parity (1, 3, 5, 7, 9, 11 and so on)
c_2 = parity (2, 3, 6, 7, 10, 11 and so on)
c_3 = parity (4-7, 12-15, 20-23 and so on)

Step 4: Error Detection and correction
The decimal equivalent of parity bits binary values is calculated. If it is 0, there is no error. Otherwise, the decimal value gives the bit position which has error. for example, if $c_1c_2c_3c_4 = 1001$, it implies that the data bit at position 9, decimal equivalent of 1001, has error. The bit is flipped to get the correct message.

Ex: Assuming even parity state, received code is wrong

P_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1	1	0	1	1

P_4: D_5 D_6 D_7
1 1 0 1 → odd
$P_4 = 1$

P_2 P_3 D_6 D_7
0 0 1
$P_2 = 0$

P_1 P_3 D_5 S_7
1 0 1 1
$P_1 = 1$

$\therefore P_4\ P_2\ P_1 = (101)_2 = 5_{10}$
5th bit has error
$\therefore (1001011)_2$ is correct

Q4

b)

1) Minimize and Implement following POS using universal gates.

$Y = (A+B)(A+C)(B+C)$

Soln: $Y = (A+B)(A+C)(B+C)$

$Y = (AA + AB + AC + BC)(B+C)$

$Y = AAB + ABB + ABC + BBC + AAC + ABC + ABC + BCC$

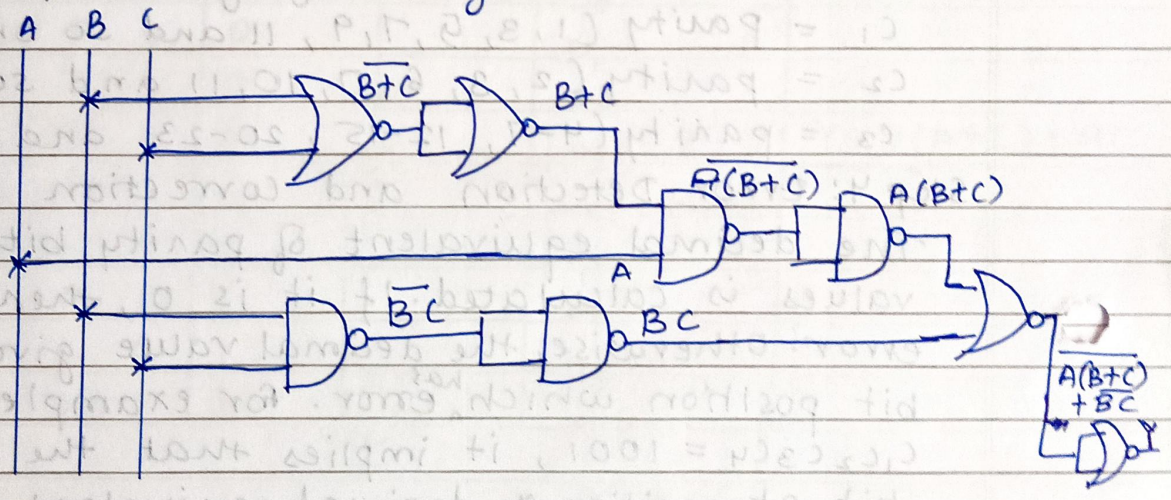
$Y = AB + ABC + BC + AC$ ... $[\because A \cdot A = A][\because A + AA = A]$

$Y = AB(1+C) + BC + AC$

$Y = AB + BC + AC$ ... $[\because 1 + A = 1]$

$Y = A(B+C) + BC$

Implementation using Universal Gates.



2) Minimize and Implement following POS using multiplexer

$Y = (A+B)(A+C)(B+C)$

Soln: $Y = (A+B)(A+C)(B+C)$

$Y = AAB + ABB + ABC + BBC + AAC + ABC + ABC + BCC$

$Y = AB + ABC + BC + AC$

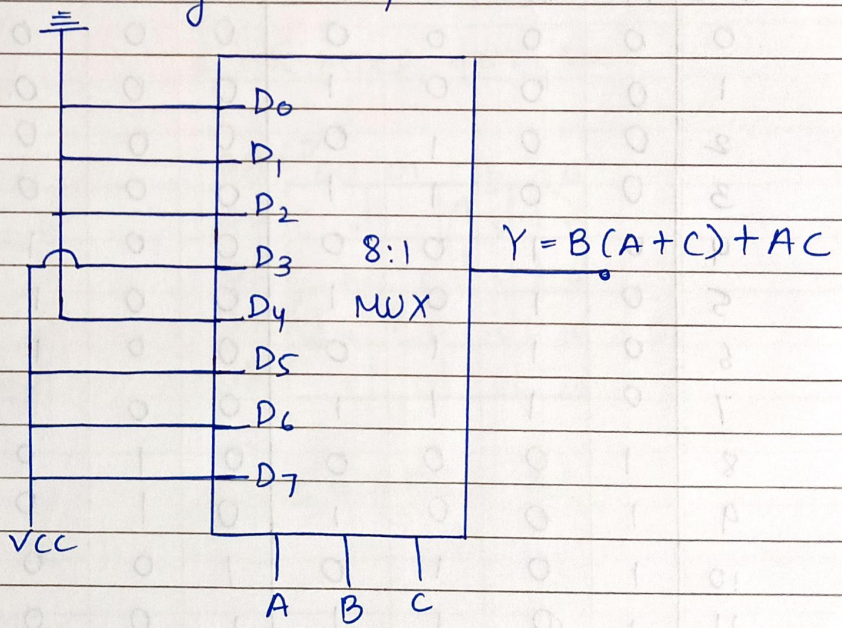
$Y = AB(1+C) + BC + AC$

$Y = AB + BC + AC$

$Y = B(A+C) + AC$

Implementation using Multiplexer

Dec	A	B	C	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



Q5

b)

1. Implement binary to BCD code converter using gates.

Soln: The input is a 4-bit binary code (A B C D) so 16 (2^4) combination are possible. Hence the output should have 8 bit, but first three bits will all be a 0 for all combinations of inputs, the output can be treated as 5 bit BCD code (W X Y Z E).

Truth Table.

	Binary				BCD				
Dec	A	B	C	D	W	X	Y	Z	E
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	1
2	0	0	1	0	0	0	0	1	0
3	0	0	1	1	0	0	0	1	1
4	0	1	0	0	0	0	1	0	0
5	0	1	0	1	0	0	1	0	1
6	0	1	1	0	0	0	1	1	0
7	0	1	1	1	0	0	1	1	1
8	1	0	0	0	0	1	0	0	0
9	1	0	0	1	0	1	0	0	1
10	1	0	1	0	1	0	0	0	0
11	1	0	1	1	1	0	0	0	1
12	1	1	0	0	1	0	0	1	0
13	1	1	0	1	1	0	0	1	1
14	1	1	1	0	1	0	1	0	0
15	1	1	1	1	1	0	1	0	1

Drawing of K-Map for each output

a) K-Map for W

AB \ CD	00	01	11	10
00	0_0	0_1	0_3	0_2
01	0_4	0_5	0_7	0_6
11	1_{12}	1_{13}	1_{15}	1_{14}
10	0_8	0_9	1_{11}	1_{10}

$\therefore W = AB + AC$

b) K-Map for X

AB \ CD	00	01	11	10
00	0_0	0_1	0_3	0_2
01	0_4	0_5	0_7	0_6
11	0_{12}	0_{13}	0_{15}	0_{14}
10	1_8	1_9	0_{11}	0_{10}

$\therefore X = A\bar{B}\bar{C}$

c) K-Map for Y

AB \ CD	00	01	11	10
00	0_0	0_1	0_3	0_2
01	1_4	1_5	1_7	1_6
11	0_{12}	0_{13}	1_{15}	1_{14}
10	0_8	0_9	0_{11}	0_{10}

$\therefore Y = \bar{A}B + BC$

d) K-Map for Z

AB \ CD	00	01	11	10
00	0	1	1_3	1_2
01	4	5	1_7	1_6
11	1_{12}	1_{13}	15	14
10	8	9	11	10

$\therefore Z = AB\bar{C} + \bar{A}C$

e) K-Map for E

AB \ CD	00	01	11	10
00	0	1_1	1_3	2
01	4	1_5	1_7	6
11	12	1_{13}	1_{15}	14
10	8	1_9	1_{11}	10

$E = D$

Minimized Expression for each output

$W = AB + AC$

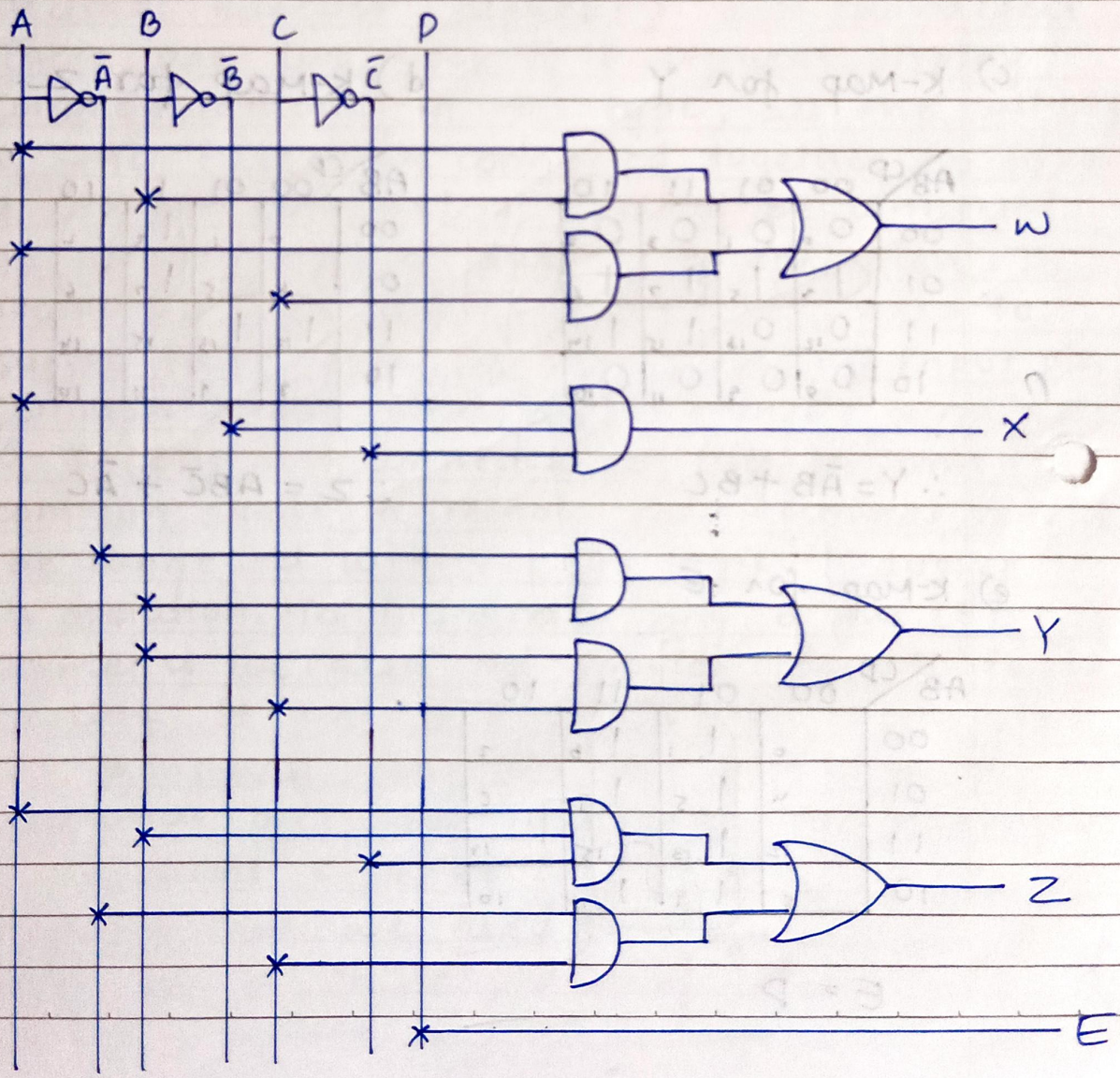
$X = A\bar{B}\bar{C}$

$Y = \bar{A}B + BC$

$Z = AB\bar{C} + \bar{A}C$

$E = D$

Logic circuit Diagram (Binary to BCD code converter)



Q5

b)

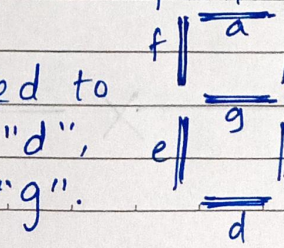
2) Short note on BCD to 7 segment display converter (single stage)

Soln:→ BCD to 7-segment display decoder is a special decoder which can convert binary coded decimal into another form which can be easily displayed through a 7-segment display. 7 segment display is a digital device that can be used for displaying decimal number, alphabets and characters There are two types of 7-segment display

→ Common cathode: In this type, all the cathodes of the 7 LEDs are connected together to form a common terminal. It should be connected to GND or logic '0' during its operation. To illuminate any LED of display, you need to supply logic '1' to its corresponding input pin.

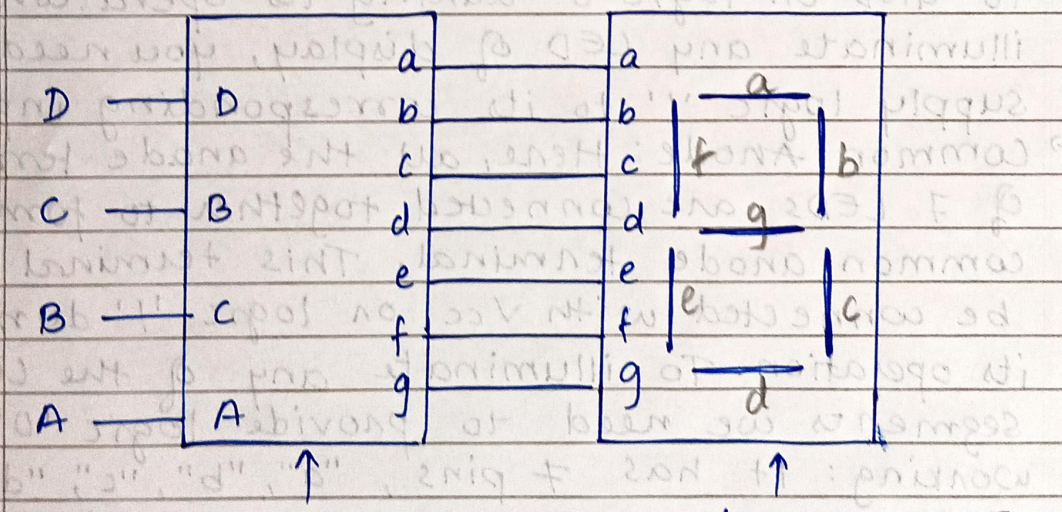
→ Common Anode: Here, all the anode terminals of 7 LEDs are connected together to form common anode terminal. This terminal should be connected with Vcc or logic '1' during its operation. To illuminate any of the LED segments we need to provide logic '0' to it.

→ Working: It has 7 pins, "a", "b", "c", "d", "e", "f", "g" which illuminate the according to what we want to display. For example if we want 0 to be displayed then we need to turn on "a", "b", "c", "d", "e", "f", & turn off "g".



Truth table.

	INPUT				OUTPUT						
Desp	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1



BCD to seven segment decoder

7 segment LED display