Bharath

INSTITUTE OF HIGHER EDUCATION AND RESEARCH

Declared as **Deemed-to-be-University** u/s 3 of the UGC Act, 1956

B.Tech- Aerospace Engineering

U20ASCJ06 - AVIONICS

Lab Manual

Vision of the institute

"Bharath Institute of Higher Education & Research (BIHER) envisions and constantly strives to provide an excellent academic and research ambience for students and members of the faculties to inherit professional competence along with human dignity and transformation of community to keep pace with the global challenges so as to achieve holistic development."

Mission of the institute

- To develop as a Premier University for Teaching, Learning, Research and Innovation on par with leading global universities.
- To impart education and training to students for creating a better society with ethics and morals.
- To foster an interdisciplinary approach in education, research and innovation by supporting lifelong professional development, enriching knowledge banks through scientific research, promoting best practices and innovation, industry driven and institute oriented cooperation, globalization and international initiatives.
- To develop as a multi-dimensional institution contributing immensely to the cause of societal advancement through spread of literacy, an ambience that provides the best of international exposures, provide health care, enrich rural development and most importantly impart value based education.
- To establish benchmark standards in professional practice in the fields of innovative and emerging areas in engineering, management, medicine, dentistry, nursing, physiotherapy and allied sciences.
- To imbibe human dignity and values through personality development and social service activities.

B.Tech- Aerospace Engineering

Vision of the Department

Department of Aeronautical Engineering will endeavor to accomplish worldwide recognition with a focal point of Excellence in the field of Aeronautics by providing quality Education through world class facilities, enabling graduates turning out to be Professional Experts with specific knowledge in Aeronautical & Aerospace engineering.

Mission of the Department

- To be the state of art Teaching and Learning center with excellent infrastructure and empowered Faculties in Aeronautical & Aerospace Engineering.
- To foster a culture of innovation among students in the field of Aeronautics and Aerospace with updated professional skills to enhance research potential for sponsored research and innovative projects.
- To Nurture young individuals to be knowledgeable, skilful, and ethical professionals in their pursuit of Aeronautical & Aerospace Engineering.

B.Tech- Aerospace Engineering

Program Educational Objectives Statements (PEO)

PEO 1: Demonstrate a solid grasp of fundamental concepts in Mathematics, Science, and Engineering, essential for effectively addressing engineering challenges within the Aerospace industry.

PEO 2: Involve in process of designing, simulating, fabricating, testing, and evaluating in the field of Aerospace.

PEO 3: Obtain advanced skills to actively engage in research and development endeavors within emerging domains, while also pursuing further education opportunities.

PEO 4: Demonstrate efficient performance both as independent contributors and as valuable team members in diverse multidisciplinary projects.

PEO 5: Embrace lifelong learning and career advancement while adapting to the evolving social demands and needs.

B.Tech- Aerospace Engineering

Programme Outcomes (PO's)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and Engg. Specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, research literature, and analyze engineering problems to arrive at substantiated conclusions using first principles of mathematics, natural, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components, processes to meet the specifications with consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and teamwork: Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports documentation. Make effective presentations and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

B.Tech- Aerospace Engineering

Program Specific Outcomes (PSO) - R2020

PSO1: Design and analyze aerospace components/systems for aerospace industries.

PSO2: Acquire the concepts of spacecraft attitude dynamics for the prediction of spacecraft motion.

Course Outcomes (COs)

CO1	**Discuss** the working principles of various avionic sub-systems and automated flight control systems. **(Understand)**
CO2	**Compare** various display technologies used in civil and military cockpits. **(Understand)**
CO3	**Discuss** Avionics system architecture and various data-buses. **(Understand)**
CO4	**Discuss** the operational principle of Aircraft Navigation Systems. **(Understand)**
CO5	**Explain** Air data Instruments used in modern aircrafts. **(Understand)**
CO6	**Observe** and explain the functionality and importance of each system. (**Imitation**)
CO7	**Carry out** the demonstration and response of the avionics system. (**Manipulation**)
CO8	**Observe** the output of the digital circuits and verify the stability characteristics of the avionics system. (**Imitation**)

Mapping/Alignment of COs with PO & PSO

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	H											H	H	
CO2	H											H	H	
CO3	H											H	H	
CO4	H									H		H	H	
CO5	H									H		H	H	
CO6	H			M	H			H	H	H		H	H	
CO7	H	M		M				H	H	H		H	H	
CO8	H			M	H			H	H	H		H	H	

(Tick mark or level of correlation: H-High, M-Medium, L-Low)

U20ASCJ06	**AVIONICS**
OBJECTIVE : To learn about basic digital electronics circuits, programming with microprocessors, Stability analysis using MATLAB.	

LIST OF EXPERIMENTS:

1.	Addition/Subtraction of binary numbers.
2.	Multiplexer/Demultiplexer Circuits.
3.	Encoder/Decoder Circuits.
4.	Addition and Subtraction of 8-bit and 16-bit numbers.
5.	Sorting of Data in Ascending & Descending order.
6.	Root Locus Analysis for Pitch Displacement Autopilot Stability.
7.	Bode Plot Analysis for Pitch Displacement Autopilot Stability
8.	Design of P, PI, and PID controller for aircraft dynamics.

1.ADDITION / SUBTRACTION OF BINARY NUMBERS

I. HALF ADDER AND FULL ADDER

AIM

To design half adder and full adder using basic logic gates and to verify the truth table

APPARATUS REQUIRED

Digital IC trainer kit	-	1
IC7408 (AND Gate)	-	1
IC7486 (XOR Gate)	-	1
IC7432 (OR Gate)	-	1
Connecting Wires	-	as required

THEORY

An adder is a digital circuit that performs addition of numbers. In many computers and other kinds of processors adders are used in the arithmetic logic units or ALU. They are also utilized in other parts of the processor, where they are used to calculate addresses, table indices, increment and decrement operators, and similar operations.

The half adder adds two single binary digits A and B. It has two outputs, sum (S) and carry (C). The carry signal represents an overflow into the next digit of a multi-digit addition.

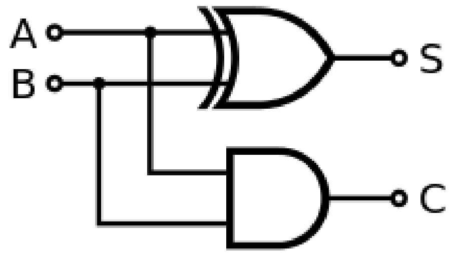
A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full adder adds three one-bit numbers, often written as A, B, and Cin; A and B are the operands, and Cin is a bit carried in from the previous less-significant stage. The full adder is usually a component in a cascade of adders, which add 8, 16, 32, etc. bit binary numbers.

PROCEDURE

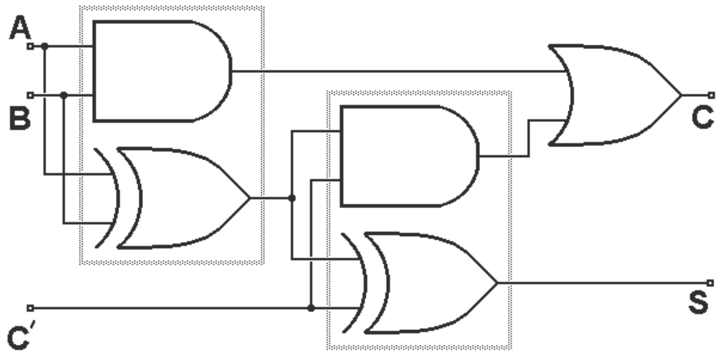
1. Make the connections as per the circuit diagram using the digital IC trainer kit and connecting wires
2. Switch ON the power supply
3. Note down the output values of sum and carry for various input combinations
4. Verify the truth table
5. Similarly, repeat the procedure for full adder

CIRCUIT DIAGRAM

HALF ADDER



FULL ADDER



TRUTH TABLE

HALF ADDER

A	B	SUM, S	CARRY, C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

FULL ADDER

A	B	Cin	SUM, S	CARRY, C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

RESULT

Thus, the half adder and full adder were designed using basic logic gates and the output was verified.

II. HALF SUBTRACTOR AND FULL SUBTRACTOR

AIM

To design half subtractor and full subtractor using basic logic gates and to verify the truth table

APPARATUS REQUIRED

Digital IC trainer kit	-	1
IC7408 (AND Gate)	-	1
IC7486 (XOR Gate)	-	1
IC7432 (OR Gate)	-	1
IC7404 (NOT Gate)	-	1
Connecting Wires	-	as required

THEORY

A subtractor is a digital circuit that performs subtraction of numbers. In many computers and other kinds of processors subtractors are used in the arithmetic logic units or ALU. They are also utilized in other parts of the processor, where they are used to calculate addresses, table indices, increment and decrement operators, and similar operations.

The half Subtractor subtracts two single binary digits A and B. It has two outputs, difference (D) and borrow (B0). The borrow signal represents an overflow into the next digit of a multi-digit subtraction.

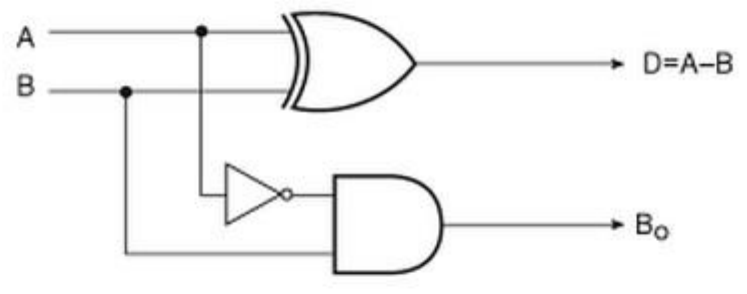
A full Subtractor subtracts binary numbers and accounts for values borrowed in as well as out. A one-bit full Subtractor subtracts three one-bit numbers, often written as A, B, and Cin; A and B are the operands, and Cin is a bit carried in from the previous less-significant stage. The full subtractor is usually a component in a cascade of subtractors, which subtract 8, 16, 32, etc. bit binary numbers.

PROCEDURE

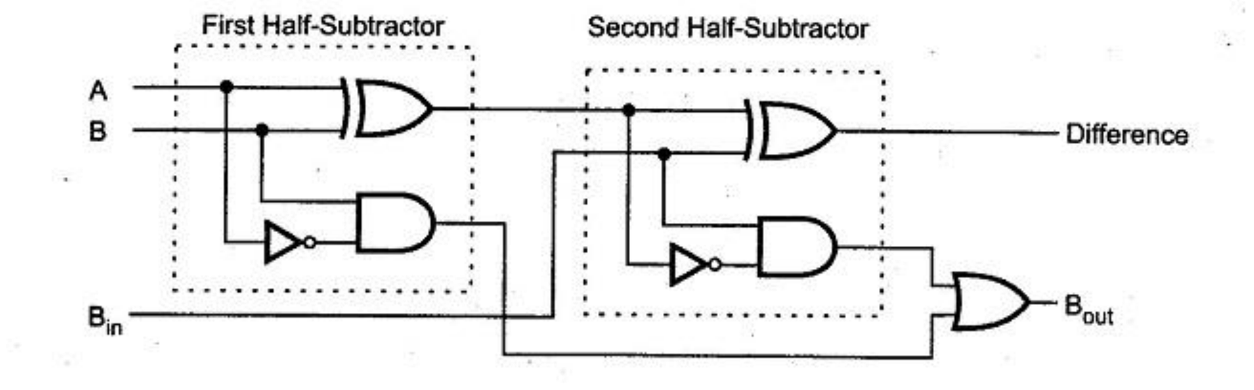
1. Make the connections as per the circuit diagram using the digital IC trainer kit and connecting wires
2. Switch ON the power supply
3. Note down the output values of sum and carry for various input combinations
4. Verify the truth table
5. Similarly, repeat the procedure for full subtractor

CIRCUIT DIAGRAM

HALF SUBTRACTOR



FULL SUBTRACTOR



TRUTH TABLE
HALF SUBTRACTOR

A	B	DIFFERENCE, D	BORROW, B0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

FULL SUBTRACTOR

A	B	Cin	DIFFERENCE, D	BORROW, B0
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

RESULT

Thus, the half subtractor and full subtractor were designed using basic logic gates and the output was verified.

2. MULTIPLEXER AND DEMULTIPLEXER

AIM

To design and implement multiplexer and demultiplexer and to verify the truth table

APPARATUS REQUIRED

Digital IC trainer kit	-	1
IC74151 (MUX)	-	1
IC74138 (DEMUX)	-	1
Mono Pulse Generator	-	1
Address Generator	-	1
Connecting Wires	-	as required

THEORY

The multiplexer, shortened to “MUX” or “MPX”, is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal. Multiplexers operate like very fast acting multiple position rotary switches connecting or controlling multiple input lines called “channels” one at a time to the output.

Multiplexers, or MUX’s, can be either digital circuits made from high speed logic gates used to switch digital or binary data or they can be analogue types using transistors, MOSFET’s or relays to switch one of the voltage or current inputs through to a single output.

In digital electronics, multiplexers are also known as data selectors because they can “select” each input line, are constructed from individual Analogue Switches encased in a single IC package as opposed to the “mechanical” type selectors such as normal conventional switches and relays.

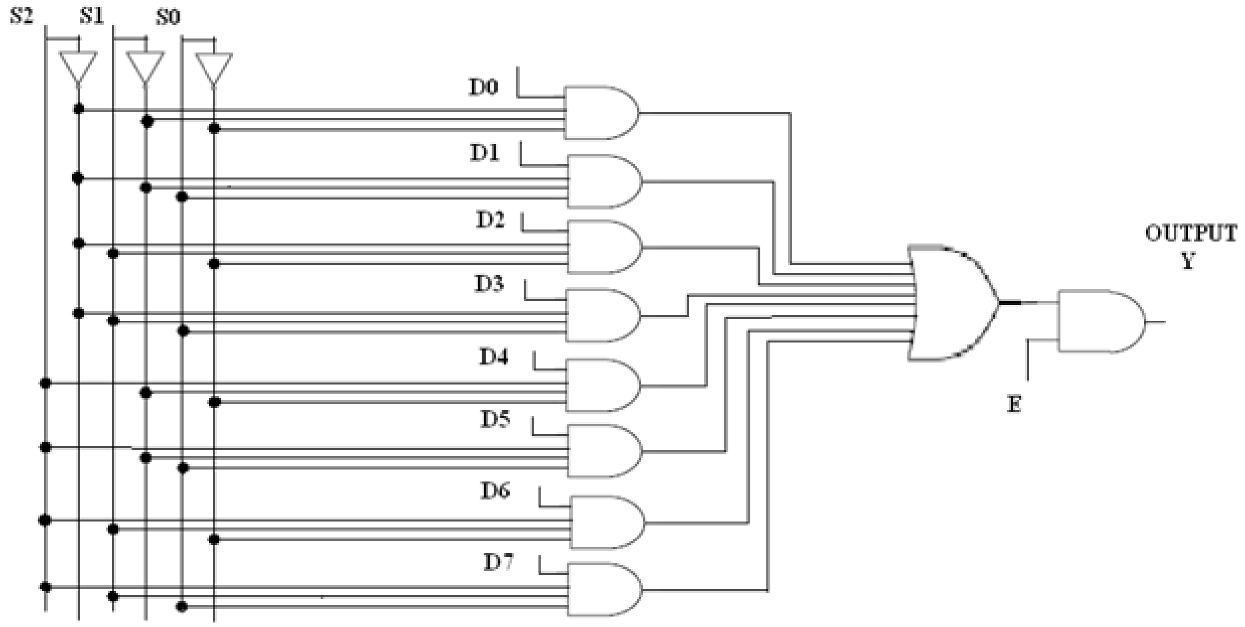
They are used as one method of reducing the number of logic gates required in a circuit design or when a single data line or data bus is required to carry two or more different digital signals.

The demultiplexer takes one single input data line and then switches it to any one of a number of individual output lines one at a time. The demultiplexer converts a serial data signal at the input to a parallel data at its output lines.

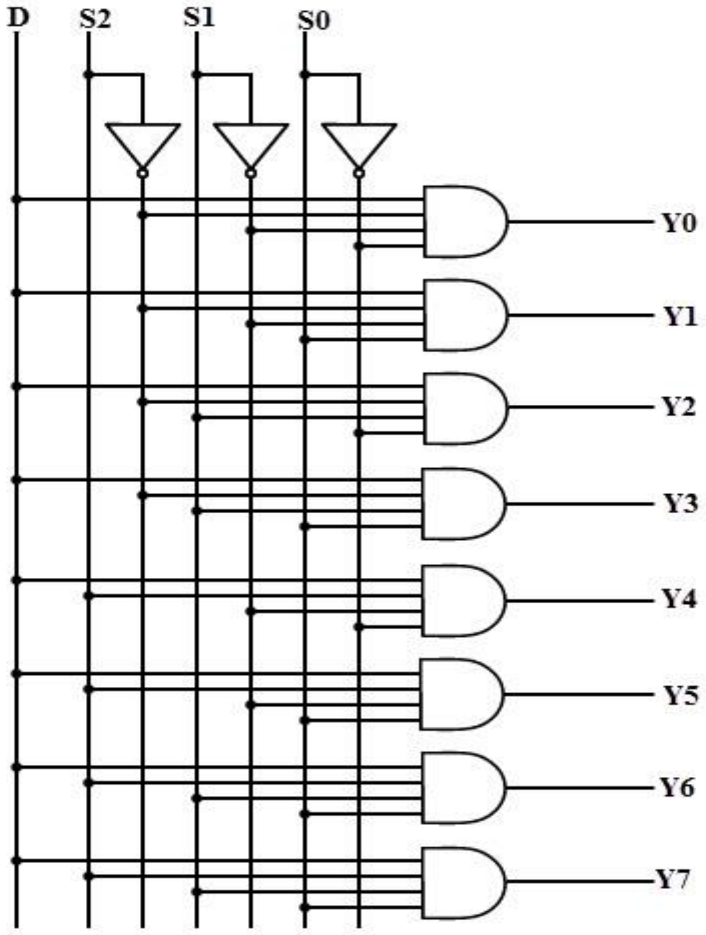
PROCEDURE

1. Make the connections as per the circuit diagram using the digital IC trainer kit and connecting wires
2. Switch ON the power supply
3. Note down the output values for various input combinations
4. Verify the truth table

CIRCUIT DIAGRAM
MULTIPLEXER



DEMULTIPLEXER

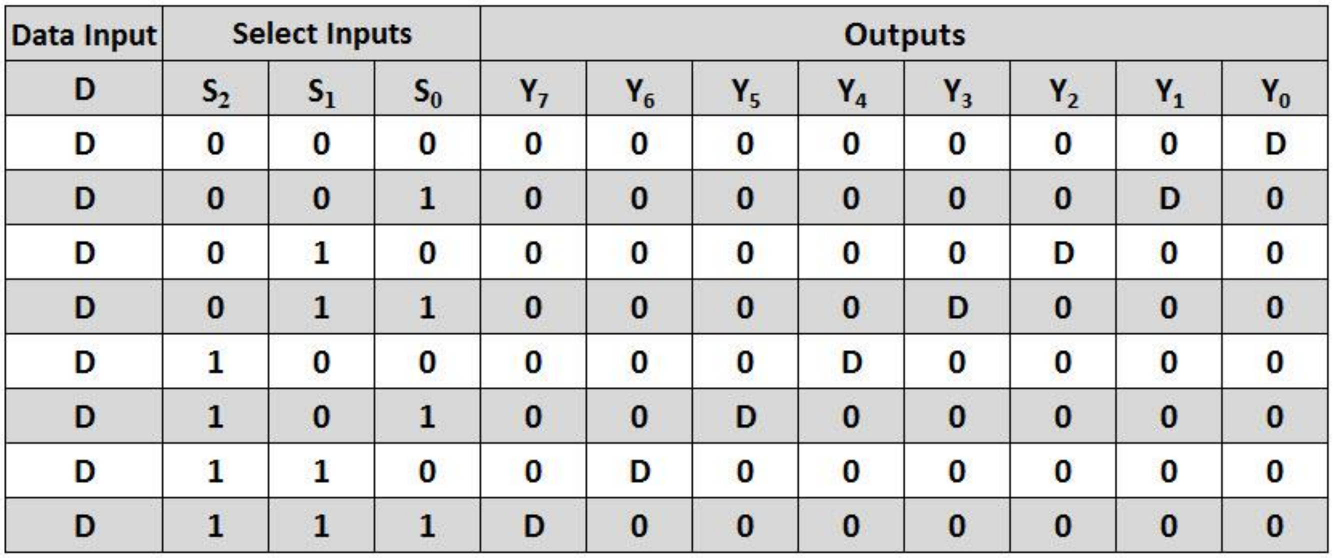


TRUTH TABLE

MULTIPLEXER

S2	**S1**	**S0**	**OUTPUT Y**
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

DEMULTIPLEXER



Data Input	Select Inputs			Outputs							
D	S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	0	0	0	0	0	D
D	0	0	1	0	0	0	0	0	0	D	0
D	0	1	0	0	0	0	0	0	D	0	0
D	0	1	1	0	0	0	0	D	0	0	0
D	1	0	0	0	0	0	D	0	0	0	0
D	1	0	1	0	0	D	0	0	0	0	0
D	1	1	0	0	D	0	0	0	0	0	0
D	1	1	1	D	0	0	0	0	0	0	0

RESULT

Thus, MUX and DEMUX were designed and implemented and their output was verified.

3. ENCODER AND DECODER

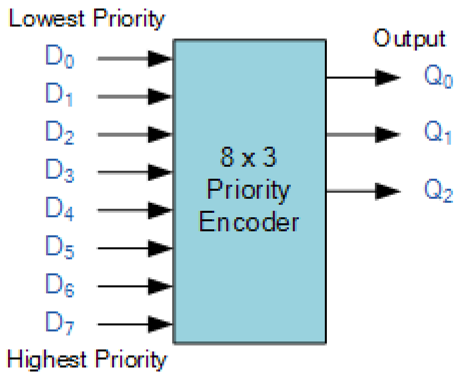
AIM
To study the encoder and decoder and their operation

APPARATUS REQUIRED

Encoder & Decoder trainer kit	-	1
Connecting Wires	-	as required

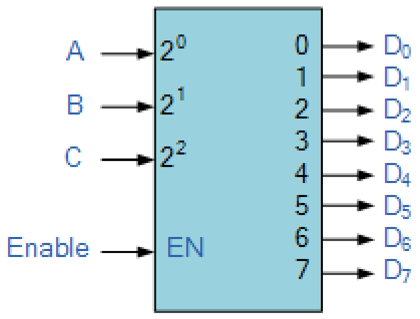
THEORY
ENCODER

An encoder is a digital circuit that performs inverse operation of a decoder. An encoder has 2ninput lines and n output lines. In encoder the output lines generates the binary codecorresponding to the input value. In octal to binary encoder it has eight inputs, one for each octaldigit and three output that generate the corresponding binary code. In encoder it is assumed thatonly one input has a value of one at any given time otherwise the circuit is meaningless. Here, when all inputs are zero the outputs are zero. The zero outputs can also begenerated when D0 = 1.



DECODER

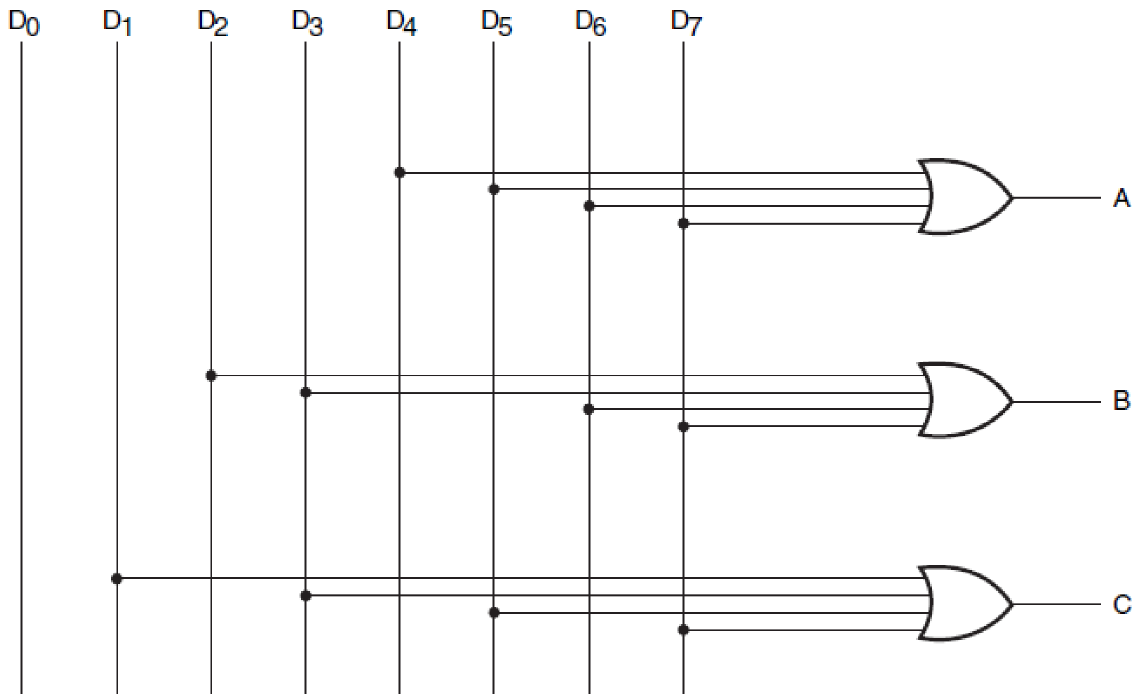
A decoder is a multiple input multiple output logic circuit which converts coded input into coded output where input and output codes are different. The input code generally has fewer bits than the output code. Each input code word produces a different output code word i.e there is one to one mapping can be expressed in truth table. In the block diagram of decoder circuit the encoded information is present as n input producing 2npossible outputs. 2noutput values are from 0through output 2n-1



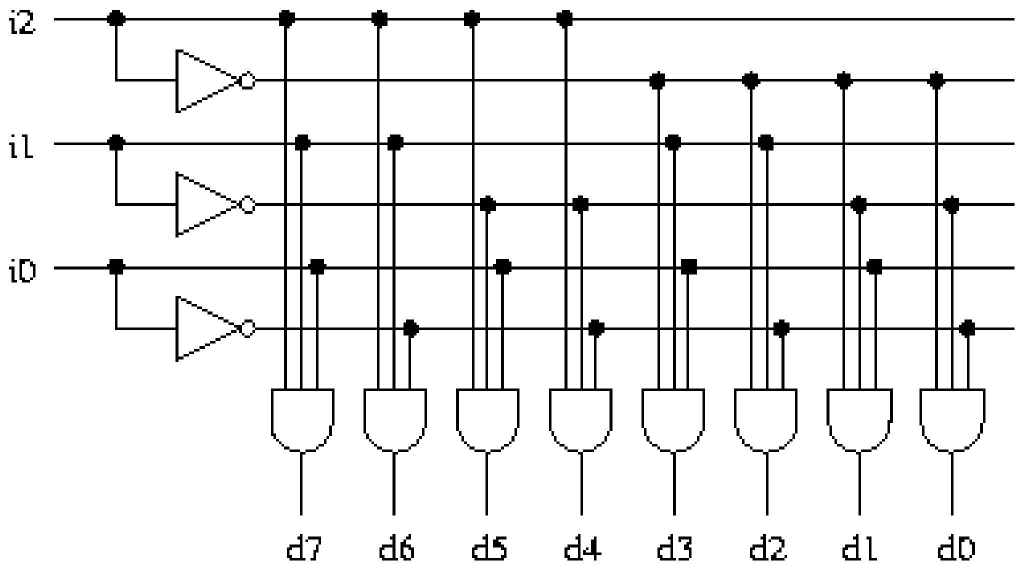
PROCEDURE
1. Make the connections as per the circuit diagram using the Universal Shift Registers trainer kit and connecting wires
2. Switch ON the power supply
3. Note down the output values for various input combinations
4. Verify the output

CIRCUIT DIAGRAM

ENCODER



DECODER



TRUTH TABLE
ENCODER

INPUT								OUTPUT		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

DECODER

INPUT			OUTPUT							
Q_2	Q_1	Q_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

RESULT

Thus, the encoder and decoder were studied and the output was verified.

4. ADDITION/SUBTRACTION OF BINARY NUMBERS.

i. SUM OF 8-BIT AND 16-BIT NUMBERS USING 8085

AIM

To write a program to perform sum of 8-bit and 16-bit numbers using 8085 microprocessor.

APPARATUS REQUIRED

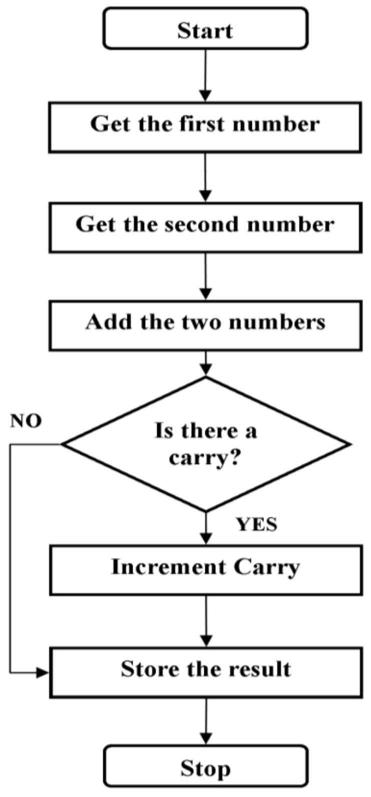
8085 Microprocessor kit,
Power supply.

SUM OF 8-BIT

ALGORITHM

Step 1: Start
Step 2: Get the first number and initialize carry
Step 3: Get the second number
Step 4: Add two numbers
Step 5: Store the result
Step 6: Stop

FLOW CHART



PROGRAM

ADDRESS	LABEL	MNEMONICS	HEX CODE	COMMENTS
4000		MVI C, 00H	0E	MOVE 00H TO REGISTER C
4001			00	
4002		LDA 4400H	3A	LOAD THE ACCUMULATOR WITH CONTENT OF 4400
4003			00	
4004			44	
4005		MOV B,A	47	MOVE THE CONTENT OF ACCUMULATOR TO B REGISTER
4006		LDA 4401H	3A	LOAD THE ACCUMULATOR WITH CONTENT OF 4401H
4007			01	
4008			44	
4009		ADD B	80	ADD CONTENT OF B REGISTER WITH THE CONTENT OF ACCUMULATOR
400A		JNC SUM	D2	JUMP ON NO CARRY TO LABEL ‘SUM’
400B			0E	
400C			40	
400D		INR C	0C	INCREMENT C REGISTER BY 1
400E	SUM	STA 4402H	32	STORE THE CONTENT OF ACCUMULATOR TO 4402
400F			02	

4010			44	
4011		MOV A,C	79	MOVE THE CONTENT OF C REGISTER TO THE ACCUMULATOR
4012		STA 4403H	32	STORE THE CONTENT OF ACCUMULATOR TO 4403
4013			03	
4014			44	
4015		HLT	76	END OF PROGRAM

TABULATION

INPUT		OUTPUT	
ADDRESS	**DATA**	**ADDRESS**	**DATA**

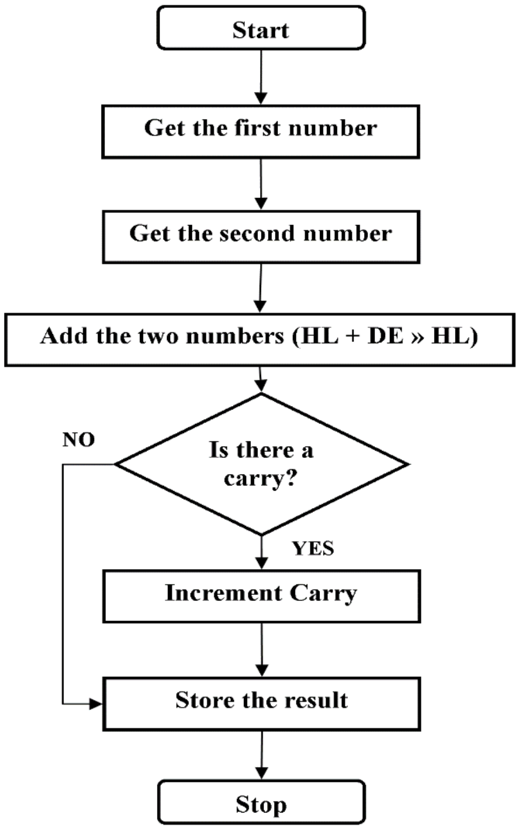
MANUAL CALCULATION

SUM OF 16-BIT

ALGORITHM

Step 1: Initialize the carry in Reg. C
Step 2: Load the first 16 bit data in HL and exchange it with DE
Step 3: Load the second 16 bit data in HL register
Step 4: Add the second data with the first data
Step 5: If the carry =0 go to step 7 otherwise go to the next step
Step 6: Increment the carry.
Step 7: Stop

FLOW CHART



PROGRAM

ADDRESS	LABEL	MNEMONICS	HEX CODE	COMMENTS
4000		MVI C, 00H	0E	MOVE 00H TO REGISTER C
4001			00	
4002		LHLD 4600H	2A	LOAD THE HL PAIR WITH THE FIRST 16 BIT DATA
4003			00	
4004			46	

4005		XCHG	EB	EXCHANGE THE DATA BETWEEN HL AND DE REGISTER PAIRS
4006		LHLD 4602H	2A	LOAD THE HL PAIR WITH THE SECOND 16 BIT DATA
4007			02	
4008			46	
4009		DAD D	19	ADD CONTENT OF DE PAIR WITH THE CONTENT OF HL(ACCUMULATOR) PAIR
400A		JNC SUM	D2	JUMP ON NO CARRY TO LABEL ‘SUM’
400B			0E	
400C			40	
400D		INR C	0C	INCREMENT C REGISTER BY 1
400E	SUM	SHLD 4604H	32	STORE THE CONTENT OF ACCUMULATOR TO 4604
400F			04	
4010			46	
4011		MOV A,C	79	MOVE THE CONTENT OF C REGISTER TO THE ACCUMULATOR
4012		STA 4606H	32	STORE THE CONTENT OF ACCUMULATOR TO 4606
4013			06	
4014			46	
4015		HLT	76	END OF PROGRAM

TABULATION

INPUT		OUTPUT	
ADDRESS	**DATA**	**ADDRESS**	**DATA**

MANUAL CALCULATION

RESULT

Thus the addition of 8-bit and 16 - bit numbers was performed and the outputs were verified.

ii. SUBTRACTION OF 8-BIT AND 16-BIT NUMBERS USING 8085

AIM

To write a program to execute subtraction of 8-bit and 16-bit numbers using 8085 microprocessor

APPARATUS REQUIRED

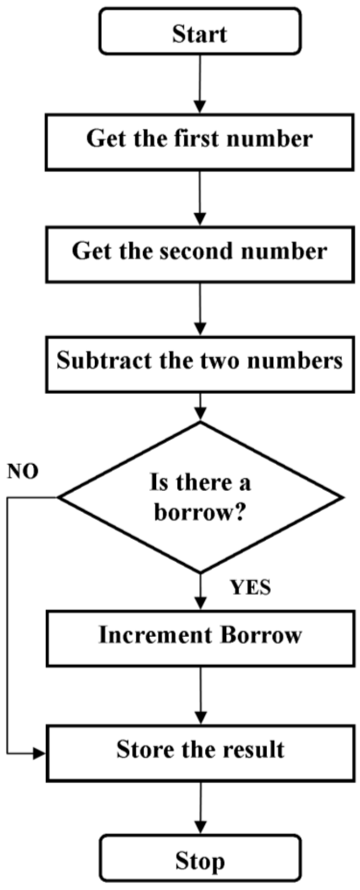
8085 Microprocessor kit,
Power supply

SUBTRACTION OF 8-BIT

ALGORITM

Step 1: Start
Step 2: Get the first number
Step 3: Get the second number
Step 4: Subtract second number from first number
Step 5: Store the result
Step 6: Stop

FLOW CHART



PROGRAM

ADDRESS	LABEL	MNEMONICS	HEX CODE	COMMENTS
4000		MVI C, 00H	0E	MOVE 00H TO REGISTER C
4001			00	
4002		LDA 4400H	3A	LOAD THE ACCUMULATOR WITH CONTENT OF 4400
4003			00	
4004			44	
4005		MOV B,A	47	MOVE THE CONTENT OF ACCUMULATOR TO B REGISTER
4006		LDA 4401H	3A	LOAD THE ACCUMULATOR WITH CONTENT OF 4401H
4007			01	
4008			44	
4009		SUB B	80	SUBTRACT CONTENT OF B REGISTER WITH THE CONTENT OF ACCUMULATOR
400A		JNC DIFF	D2	JUMP ON NO CARRY TO LABEL ‘DIFF’
400B			0E	
400C			40	
400D		INR C	0C	INCREMENT C REGISTER BY 1
400E	DIFF	STA 4402H	32	STORE THE CONTENT OF ACCUMULATOR TO 4402
400F			02	
4010			44	
4011		MOV A,C	79	MOVE THE CONTENT OF C REGISTER TO THE ACCUMULATOR

4012		STA 4403H	32	STORE THE CONTENT OF ACCUMULATOR TO 4403
4013			03	
4014			44	
4015		HLT	76	END OF PROGRAM

TABULATION

INPUT		OUTPUT	
ADDRESS	**DATA**	**ADDRESS**	**DATA**

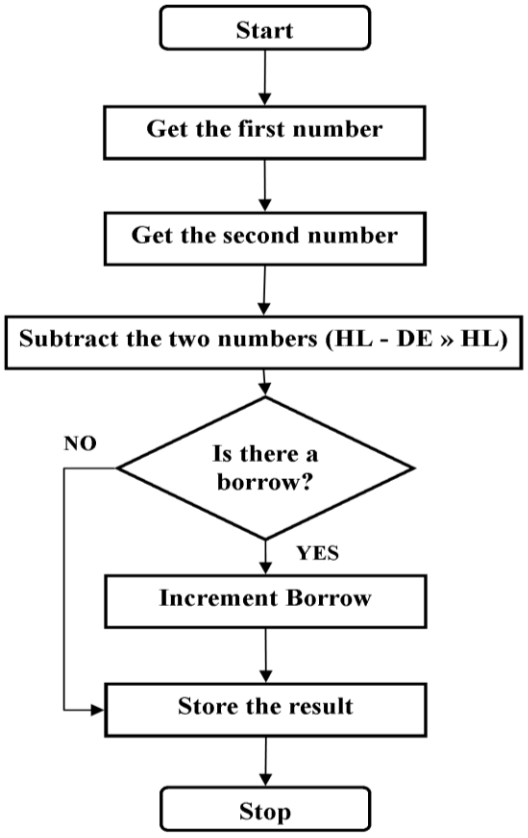
MANUAL CALCULATION

SUBTRACTION OF 16-BIT

ALGORITM

Step 1: Initialize the program
Step 2: Load the first data
Step 3: Load the second data
Step 4: Exchange 16 bit data between two registers
Step 5: Subtract second data from first data
Step 6: If carry = 0 go to step 8
Step 7: Increment the value of carry
Step 8: End the program

FLOW CHART



PROGRAM

ADDRESS	LABEL	MNEMONICS	HEX CODE	COMMENTS
4000		MVI C, 00H	0E	MOVE 00H TO REGISTER C
4001			00	
4002		LHLD 4600H	2A	LOAD THE HL PAIR WITH THE FIRST 16 BIT DATA
4003			00	
4004			46	
4005		XCHG	EB	EXCHANGE THE DATA BETWEEN HL AND DE REGISTER PAIRS

4006		LHLD 4602H	2A	LOAD THE HL PAIR WITH THE SECOND 16 BIT DATA
4007			02	
4008			46	
4009		MOV A,E	7B	MOVE CONTENT OF E TO A
400A		SUB L	95	SUBTRACT L REGISTER
400B		MOV L,A	6F	MOVE CONTENT OF A TO L
400C		MOV A,D	7A	MOVE CONTENT OF D TO A
400D		SBB H	9C	SUBTRACT CONTENT OF H FROM A
400E		MOV H,A	67	MOVE CONTENT OF A TO H
400F		JNC BORROW	D2	JUMP ON NO CARRY TO LABEL ‘BORROW’
4010			13	
4011			40	
4012		INR C	0C	INCREMENT C REGISTER BY 1
4013	BORROW	SHLD 4604H	32	STORE THE CONTENT OF ACCUMULATOR TO 4604
4014			04	
4015			46	
4016		MOV A,C	79	MOVE THE CONTENT OF C REGISTER TO THE ACCUMULATOR
4017		STA 4606H	32	STORE THE CONTENT OF ACCUMULATOR TO 4606
4018			06	

4019			46	
401A		HLT	76	END OF PROGRAM

TABULATION

INPUT		OUTPUT	
ADDRESS	**DATA**	**ADDRESS**	**DATA**

MANUAL CALCULATION

RESULT

Thus, the subtraction of 8-bit and 16 - bit numbers was performed and the outputs were verified.

5.SORTING OF DATA IN ASCENDING & DESCENDING ORDER

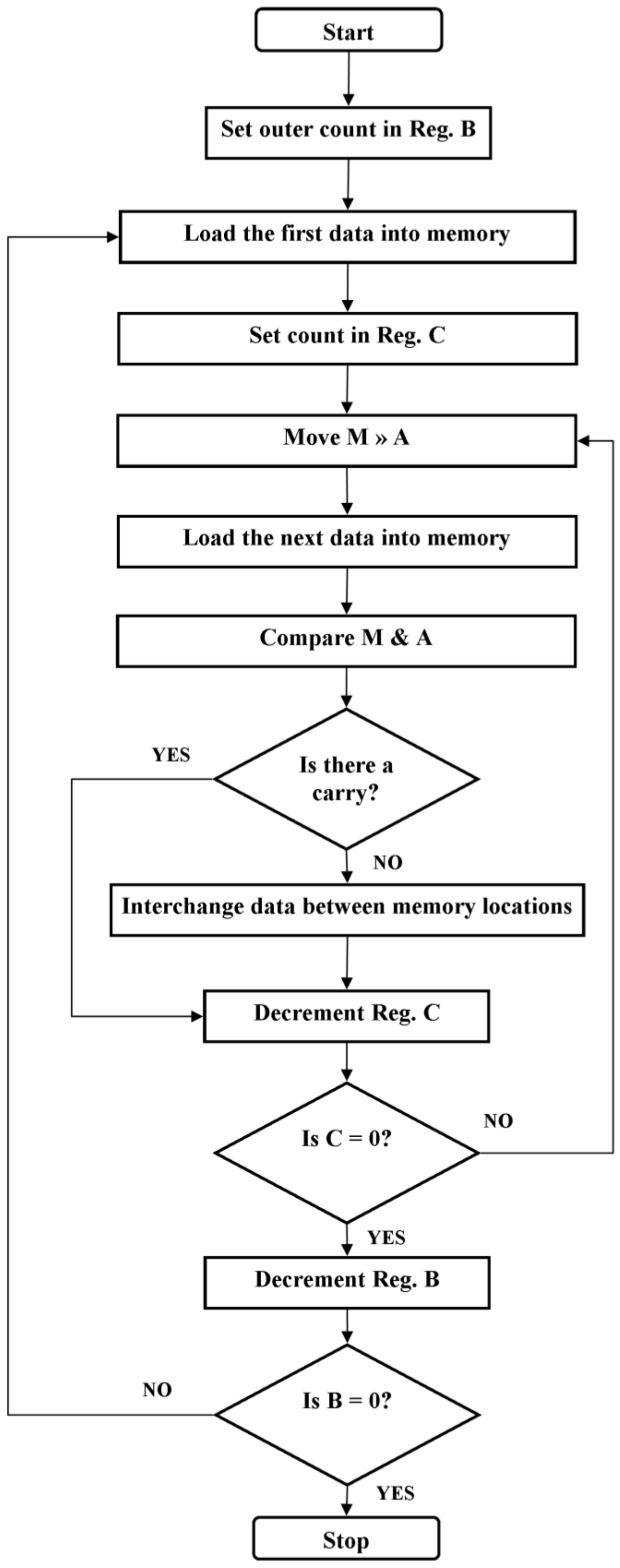
I. SORTING IN ASCENDING ORDER USING 8085

AIM
To write a program to sort the given series in ascending and descending order using 8085 microprocessor.

APPARATUS REQUIRED
8085 Microprocessor kit,
Power Supply.

SORTING IN ASCENDING ORDER
ALGORITHM
Step 1: Initialize outer count; set Reg. B = 03
Step 2: Load the first data into the memory
Step 3: Initialize count; set Reg. C = 03
Step 4: Move the content of memory to accumulator
Step 5: Load the next data into the memory
Step 6: Compare the data of memory with the data in the accumulator
Step 7: If Carry = 1, then go to step 9, else go to next step
Step 8: Interchange data between the two loaded memories
Step 9: Decrement C Reg. and if C is not equal to zero, go to step 4; else go to next step
Step 10: Decrement B Reg. and if B is not equal to zero, go to step 2; else go to next step
Step 11: End the program

FLOW CHART


Start
Set outer count in Reg. B
Load the first data into memory
Set count in Reg. C
Move M » A
Load the next data into memory
Compare M & A
Is there a carry?
YES
NO
Interchange data between memory locations
Decrement Reg. C
Is C = 0?
NO
YES
Decrement Reg. B
Is B = 0?
NO
YES
Stop

PROGRAM

ADDRESS	LABEL	MNEMONICS	HEX CODE	COMMENTS
4000		MVI B,03H	06	MOVE 03H IMMEDIATELY TO B REGISTER
4001			04	
4002	LOOP 3	LXI H, 4600H	21	LOAD THE ACCUMULATOR WITH THE CONTENT OF 4600H
4003			00	
4004			46	
4005		MVI C,03H	0E	MOVE 03H IMMEDIATELY TO C REGISTER (COUNT)
4006			04	
4007	LOOP 2	MOV A,M	7E	MOVE THE VALUE FROM MEMORY TO THE ACCUMULATOR
4008		INX H	23	INCREMENT H VALUE (NEXT ADDRESS)
4009		CMP M	BE	COMPARE THE CONTENT OF MEMORY WITH ACCUMULATOR
400A		JC LOOP 1	DA	JUMP ON CARRY TO LABEL ‘LOOP 1’
400B			12	
400C			40	
400D		MOV D,M	56	MOVE THE CONTENT OF MEMORY TO D REGISTER
400E		MOV M,A	77	MOVE THE CONTENT OF ACCUMULATOR TO MEMORY
400F		DCX H	2B	DECREMENT H VALUE
4010		MOV M,D	72	MOVE CONTENT OF D REGISTER TO MEMORY

4011		INX H	23	INCREMENT H VALUE
4012	LOOP 1	DCR C	0D	DECREMENT C REGISTER
4013		JNZ LOOP 2	C2	JUMP ON NO ZERO TO LABEL ‘LOOP 2’
4014			07	
4015			40	
4016		DCR B	05	DECREMENT B REGISTER
4017		JNZ LOOP 3	C2	JUMP ON NO ZERO TO LABEL ‘LOOP 3’
4018			02	
4019			40	
401A		HLT	76	END OF PROGRAM

TABULATION

INPUT		OUTPUT	
ADDRESS	**DATA**	**ADDRESS**	**DATA**

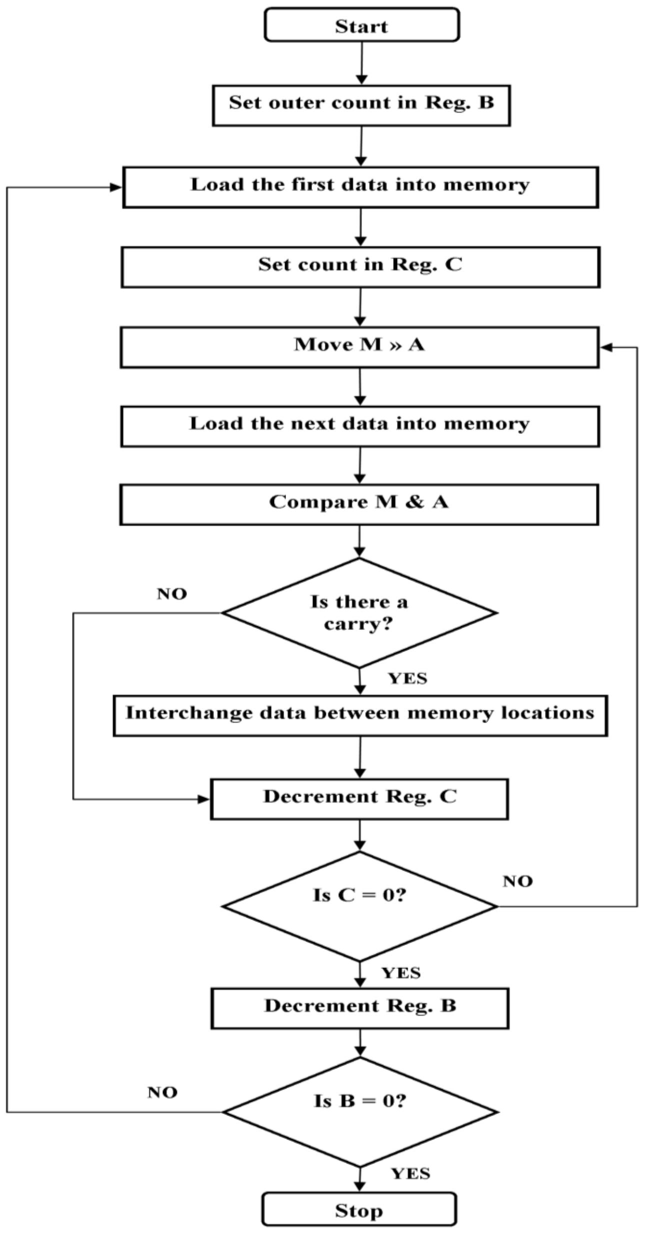
MANUAL CALCULATION

SORTING IN DESCENDING ORDER

ALGORITHM

Step 1: Initialize outer count; set Reg. B = 03
Step 2: Load the first data into the memory
Step 3: Initialize count; set Reg. C = 03
Step 4: Move the content of memory to accumulator
Step 5: Load the next data into the memory
Step 6: Compare the data of memory with the data in the accumulator
Step 7: If Carry = 0, then go to step 9, else go to next step
Step 8: Interchange data between the two loaded memories
Step 9: Decrement C Reg. and if C is not equal to zero, go to step 4; else go to next step
Step 10: Decrement B Reg. and if B is not equal to zero, go to step 2; else go to next step
Step 11: End the program

FLOW CHART



PROGRAM

ADDRESS	LABEL	MNEMONICS	HEX CODE	COMMENTS
4000		MVI B,03H	06	MOVE 03H IMMEDIATELY TO B REGISTER
4001			04	
4002	LOOP 3	LXI H, 4600H	21	LOAD THE ACCUMULATOR WITH THE CONTENT OF 4600H
4003			00	
4004			46	
4005		MVI C,03H	0E	MOVE 03H IMMEDIATELY TO C REGISTER (COUNT)
4006			04	
4007	LOOP 2	MOV A,M	7E	MOVE THE VALUE FROM MEMORY TO THE ACCUMULATOR
4008		INX H	23	INCREMENT H VALUE (NEXT ADDRESS)
4009		CMP M	BE	COMPARE THE CONTENT OF MEMORY WITH ACCUMULATOR
400A		JNC LOOP 1	DA	JUMP ON NO CARRY TO LABEL ‘LOOP 1’
400B			12	
400C			40	
400D		MOV D,M	56	MOVE THE CONTENT OF MEMORY TO D REGISTER
400E		MOV M,A	77	MOVE THE CONTENT OF ACCUMULATOR TO MEMORY
400F		DCX H	2B	DECREMENT H VALUE
4010		MOV M,D	72	MOVE CONTENT OF D REGISTER TO MEMORY
4011		INX H	23	INCREMENT H VALUE

4012	LOOP 1	DCR C	0D	DECREMENT C REGISTER
4013		JNZ LOOP 2	C2	JUMP ON NO ZERO TO LABEL ‘LOOP 2’
4014			07	
4015			40	
4016		DCR B	05	DECREMENT B REGISTER
4017		JNZ LOOP 3	C2	JUMP ON NO ZERO TO LABEL ‘LOOP 3’
4018			02	
4019			40	
401A		HLT	76	END OF PROGRAM

TABULATION

INPUT		**OUTPUT**	
ADDRESS	**DATA**	**ADDRESS**	**DATA**

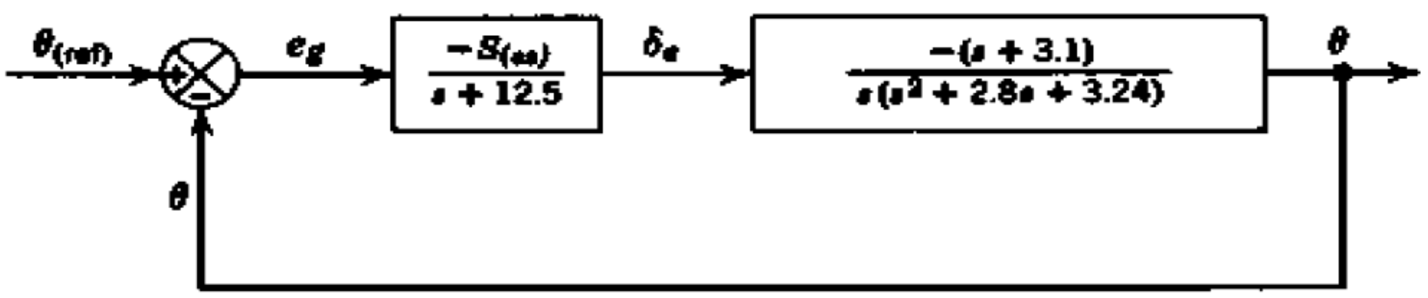
MANUAL CALCULATION

RESULT

Thus, a program was executed to sort the given series in both ascending and descending order using 8085 microprocessor and the output was verified.

6. ROOT LOCUS ANALYSIS FOR PITCH DISPLACEMENT AUTOPILOT STABILITY

Perform stability analysis of Displacement Autopilot as shown in fig.



$\theta_{(ref)}$
e_g
$\frac{-S_{(es)}}{s + 12.5}$
δ_e
$\frac{-(s + 3.1)}{s(s^2 + 2.8s + 3.24)}$
θ
θ

Figure 2-2 Block diagram for the conventional transport and autopilot.

AIM

To perform of stability analysis of Displacement Autopilot using root locus techniques.

THEORY

A pitch displacement autopilot is a control system that stabilizes the pitch motion of an aircraft.

Stability analysis involves examining the behavior of the system as parameters such as gains or coefficients change. The root locus is a graphical representation of the locations of the system's poles as a parameter varies. In a stable system, all poles should have negative real parts.

System stability can be determined by finding the intersection of the root locus in the imaginary axis. The effect of function adding poles and zeros to the transfer function on the system stability can be studied.

PROGRAM:

```
%DISPLACEMENT AUTOPLOT STABILITY ANALYSTE USING ROOTLOCUS
%TRANSFER FUNCTION OF ELEVATOR SERVO
elevator=tf([-1], [1 12.5])
%TRANSFER FUNCTION OF AIRCRAFT
anum=-1*[1 3.1]
aden=conv ([1 0], [1 2.8 3.24])
aircraft=tf (anum, aden)
%FORWARD TRANSFER FUNCTION
forward=elevator*aircraft
figure (1)
rlocus(forward)
figure (1)
axis ([-15 2 -6 6])
title('STABILITY ANALYSIS USING ROOT LOCUS FOR DISPLACEMENT AUTOPLOT')
```

RESULT

Thus the stability analysis were performed using root locus for the given transfer function. The maximum value of the gain for the stable system is less than …………

7. BODE PLOT ANALYSIS FOR PITCH DISPLACEMENT AUTOPILOT STABILITY

Aim

To analyze the stability and robustness of a pitch displacement autopilot system using Bode plot analysis.

Theory

Bode Plot Analysis:

a. Gain Margin (GM): The gain margin is the amount by which the gain of the system can be increased before the system becomes unstable. In a Bode plot, it corresponds to the gain at the phase crossover frequency where the phase is -180 degrees. A positive gain margin indicates stability.

- If $GM > 0$: The system is stable.
- If $GM = 0$: The system is critically stable.
- If $GM < 0$: The system is unstable.

b. Phase Margin (PM): The phase margin is the amount by which the phase of the system can be increased before the system becomes unstable. It is the phase difference between the actual phase and -180 degrees at the gain crossover frequency where the gain is 0 dB.

- If $PM > 0$: The system is stable.
- If $PM = 0$: The system is critically stable.
- If $PM < 0$: The system is unstable.

c. Delay Margin: The delay margin quantifies the additional time delay that a system can tolerate before instability. It is derived from the phase margin and provides insights into the system's robustness in the presence of time delays.

$$\tau_m = \frac{1}{\omega_c} \tan\left(\frac{PM}{180^\circ}\pi\right)$$

PROGRAM:

```
% Transfer function of elevator servo
num_elevator = -1;
den_elevator = [1, 12.5];
H_elevator = tf(num_elevator, den_elevator);

% Transfer function of aircraft
num_aircraft = [-1, -3.1];
den_aircraft = [1, 2.8, 3.24, 0];
H_aircraft = tf(num_aircraft, den_aircraft);

% Overall transfer function
G = H_elevator * H_aircraft;

% Bode plot
figure;
```

```
bode(G);
title('Bode Plot-pitch displacement autopilot');
% Calculate gain and phase margins
[GM, PM, ~, ~] = margin(G);
fprintf('Gain Margin (GM): %.2f dB\n', 20*log10(GM));
fprintf('Phase Margin (PM): %.2f degrees\n', PM);
```

RESULT

System Stability

The positive gain margin (GM) of 37.45 dB indicates a significant safety margin against gain increases, contributing to the overall stability of the system.

Phase Margin (PM):

The phase margin of 87.27 degrees is well above the stability threshold, ensuring stability and providing a comfortable buffer against phase lead.

Delay Margin

The delay margin of 19.9 seconds underscores the system's robustness in handling time delays. A larger delay margin indicates a higher tolerance to delays, enhancing the system's performance in real-world scenarios.

System Robustness:

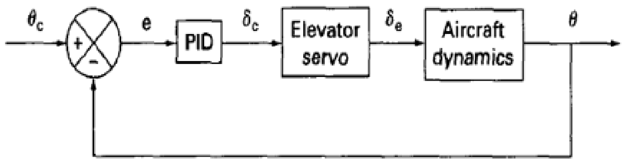
The substantial gain and phase margins, along with the notable delay margin, collectively highlight the system's robustness. The system exhibits resilience to uncertainties, disturbances, and time delays.

In summary, the pitch displacement autopilot system, as characterized by the Bode plot analysis, not only demonstrates stability but also exhibits robustness, both in terms of gain and phase margins and the ability to handle significant time delays.

8. PID CONTROLLER DESIGN FOR AIRCRAFT DYNAMICS

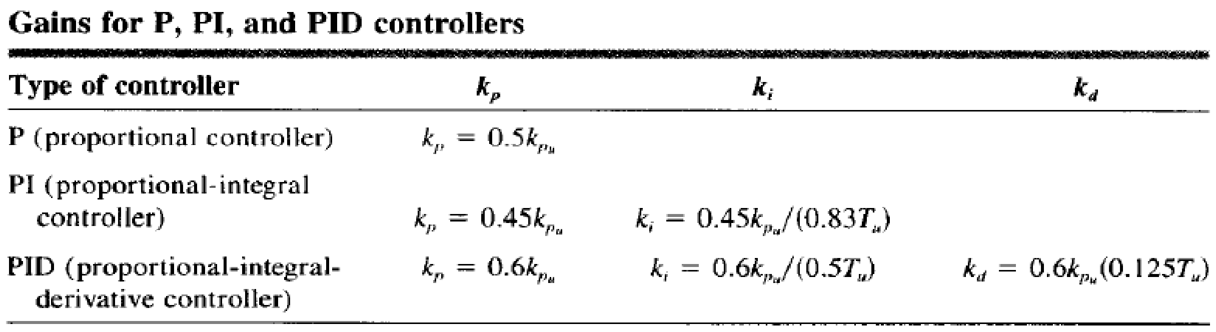
Control System Design

Design the PID controller for a pitch attitude autopilot as illustrated in Figure .
The transfer functions for each component are given in
Elevator Servo = -10/(s+10)
Aircraft Dynamics = $-3/(s^2 + 2s + 5)$



AIM
To design the PID Controller for aircraft dynamics.

THEORY
The selection of the gains for the PID controller can be determined by a method developed by Ziegler and Nichols.
To apply this technique the root locus plot for the control system with the integral and derivative gains set to 0 must become marginally stable. That is, as the proportional gain is increased the locus must intersect the imaginary axis. The proportional gain, kp for which this occurs is called the ultimate gain, k_{pu}. The purely imaginary roots, $\lambda = i\omega$, determine the value of $T_u = 2\pi/\omega_n$.
One additional restriction must be met: All other roots of the system must have negative real parts; that is, they must be in the left-hand portion of the complex s plane. If these restrictions are satisfied the P, PI, or PID gains easily can be determined.



Gains for P, PI, and PID controllers

Type of controller	k_p	k_i	k_d
P (proportional controller)	$k_p = 0.5k_{p_u}$		
PI (proportional-integral controller)	$k_p = 0.45k_{p_u}$	$k_i = 0.45k_{p_u}/(0.83T_u)$	
PID (proportional-integral-derivative controller)	$k_p = 0.6k_{p_u}$	$k_i = 0.6k_{p_u}/(0.5T_u)$	$k_d = 0.6k_{p_u}(0.125T_u)$

PROGRAM

```
% Aircraft Dynamics Transfer Function
num = [3];
den = conv([1 10], [1 2 5]);
G = tf(num, den);
H = [1];

% Open-loop system
M = feedback(G, H);

% Closed-loop systems with P, PI, and PID controllers
Kp_P = 44.35;  % Placeholder values for P controller
Gc_P = pid(Kp_P, 0, 0);
Mc_P = feedback(Gc_P * G, H);
```

```
Kp_PI = 39.92;  % Placeholder values for PI controller
Ki_PI = 39.42;
Gc_PI = pid(Kp_PI, Ki_PI, 0);
Mc_PI = feedback(Gc_PI * G, H);

Kp_PID = 53.22;  % Placeholder values for PID controller
Ki_PID = 87.24;
Kd_PID = 8.12;
Gc_PID = pid(Kp_PID, Ki_PID, Kd_PID);
Mc_PID = feedback(Gc_PID * G, H);

% Plot step responses
figure;
step(M, 'b-', Mc_P, 'r--', Mc_PI, 'g-.', Mc_PID, 'k:');
legend('Open-Loop System', 'P Controller', 'PI Controller', 'PID Controller');
title('Step Response Comparison');
xlabel('Time');
ylabel('Amplitude');
grid on;
```

RESULT

The P controller alone might not be sufficient to eliminate steady-state error in your system. Steady-state error can often be addressed by incorporating integral action, as seen in both the PI and PID controllers.
Both the PI and PID controllers have similar final values, but the PID controller might offer improved transient response due to the inclusion of derivative action. This can result in faster settling time and reduced overshoot.
The choice between PI and PID controllers depends on the specific requirements of your system. If a faster transient response is desirable and overshoot needs to be minimized, the PID controller might be preferred.
In summary, the PID controller appears to offer a good balance between steady-state performance and transient response.