DRONACHARYA
College of Engineering

LABORATORY MANUAL

B.Tech. Semester- VI

MICROPROCESSOR AND MICROCONTROLLER LAB
Subject code: LC-RA-314G

Prepared by:	Checked by:	Approved by:
Mrs. Monika Thakur	Mrs. Dimple Saproo	Name : Prof. (Dr.) Isha Malhotra
Sign.: ……	**Sign.: ……**	**Sign.: ……**

DEPARTMENT OF ROBOTICS AND AUTOMATION
DRONACHARYA COLLEGE OF ENGINEERING
KHENTAWAS, FARRUKH NAGAR, GURUGRAM (HARYANA)

Table of Contents

1. Vision and Mission of the Institute
2. Vision and Mission of the Department
3. Programme Educational Objective (PEOs)
4. Programme Outcomes (POs)
5. Programme Specific Outcomes (PSOs)
6. University Syllabus
7. Course Outcomes (Cos)
8. CO-PO and CO-PSO Mapping
9. Course Overview
10. List of Experiments
11. Dos and DON'Ts
12. General Safety Precautions
13. Guidelines for students for report preparation
14. Lab assessment criteria
15. Details of Conducted Experiments
16. Lab Experiments

Vision and Mission of the Institute

Vision:

To impart Quality Education, to give an enviable growth to seekers of learning, to groom them as World Class Engineers and Managers competent to match the expanding expectations of the Corporate World has been our ever enlarging vision extending to new horizons since the inception of Dronacharya College of Engineering.

Mission:

M1: To serve the society and improve the mode of life by imparting high quality education in the field of Engineering and Management catering to the explicit and implicit needs of the students, society, humanity and industry.

M2: To create an inspiring ambience that raises the motivation level for conducting quality research

M3: To provide an environment for acquiring ethical values and positive attitude.

Vision and Mission of the Department

Vision:

To be a globally recognized leader in robotics and automation education, research, and innovation, empowering students to excel in a technologically advanced world.

Mission:

M1: To provide high quality education and training in robotics and automation, equipping students with the knowledge, skills, and attitudes necessary for successful careers in the field.

M2: To foster a culture of innovation and entrepreneurship, encouraging student and faculty to develop and apply cutting-edge technologies in robotics and automation

M3: To conduct impactful research and development activities, addressing real-world challenges and advancing the field of robotics and automation

M4: To promote ethical practices, environmental sustainability and social responsibility in the deployment of technologies

M5. To collaborate with industry, academia and research organizations to create opportunities for industry-driven projects, internships, and placements ensuring the relevance of our programs and enhancing industry readiness or our graduates

Programme Educational Objectives (PEOs)

PEO1: To practice the profession of engineering using a systems perspective and analyze, design, develop, optimize & implement engineering solutions and work productively as engineers, including supportive and leadership roles on multidisciplinary teams

PEO2: To Continue their education in leading graduate programs in engineering & interdisciplinary areas to emerge as researchers, experts, educators & entrepreneurs and recognize the need for, an ability to engage in continuing professional development and life-long learning

PEO3: To Engineers, guided by the principles of sustainable development and global interconnectedness, will understand how engineering projects and affect society and the environment.

PEO4: To Promote Design, Research and implementation of products and services in the field of Engineering through strong Communication and Entrepreneurial skills.

PEO5: To Re-learn and innovate in ever-changing global economic and technological environments on the 21st century.

Programme Outcomes (POs)

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

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

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

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods 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 software 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 team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

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

Programme Specific Outcomes (PSOs)

PSO1: Identify the needs, analyze, design and develop simple robotic systems and programs for diverse applications in real time.

PSO2: Design, select and integrate appropriate automation and robotic subsystems for multi-domain engineering and integrate software applications tools.

PSO3: Develop impactful engineering solutions by using research-based knowledge and research methods in the fields of advanced robotics and other relevant fields.

PSO4: Evaluate existing engineering elements and processes, identifying areas for improvement. Propose innovative robotic and automation solutions to enhance the performance and efficiency of conventional systems.

PSO5: Identify suitable sensing, interfacing, control, actuation, and communication technologies to integrate various subsystems. Develop robots capable of analyzing data and implementing automated solutions through seamless connectivity between different components.

.

University Syllabus

1 1. Write a program using 8085 and verify for:

a. Addition of two 8-bit numbers.

b. Addition of two 8-bit numbers (with carry).

2. Write a program using 8085 and verify for:

a. 8-bit subtraction (display borrow)

b. 16-bit subtraction (display borrow)

3. Write a program using 8085 for multiplication of two 8- bit numbers by repeated addition method. Check for minimum number of additions and test for typical data.

4. Write a program using 8085 for multiplication of two 8- bit numbers by bit rotation method and verify.

5. Write a program using 8086 for finding the square root of a given number and verify.

6. Write a program using 8086 for copying 12 bytes of data from source to destination and verify.

7. Write a program using 8086 and verify for:

a. Finding the largest number from an array.

b. Finding the smallest number from an array.

8. Write a program using 8086 for arranging an array of numbers in descending order and verify.

9. Write a program using 8086 for arranging an array of numbers in ascending order and verify.

10. Write a program to interface a two-digit number using seven-segment LEDs. Use 8085/8086 microprocessor and 8255 PPI.

11. Write a program to control the operation of stepper motor using 8085/8086 microprocessor and 8255 PPI.

12. To study implementation & interfacing of Display devices Like LCD, LED Bar graph & seven segment display with Microcontroller 8051/AT89C51

13. To study implementation & interfacing of Different motors like stepper motor, DC motor & servo Motors.

14. Write an ALP for temperature & pressure measurement

15. Write a program to interface a graphical LCD with 89C51

Course Outcomes (COs)

Upon successful completion of the course, the student will be able to:

CO1: Understand the basic architecture, features, and working principles of microprocessors and microcontrollers. Identify and differentiate between microprocessor and microcontroller families and their applications.

CO2: Write assembly language programs for microprocessors to perform basic operations and computations. Develop programs using high-level programming languages to interface with microcontrollers.

CO3: Interface various input/output devices (such as switches, LEDs, LCDs) with microcontrollers and microprocessors. Implement serial communication protocols (such as UART, SPI, I2C) to establish communication with external devices.

CO4: Apply problem-solving techniques to resolve hardware and software-related problems.

CO-PO Mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	1	2	2	1	1		2		1	2	1
CO2	1	1	2	1		3				3	2	2
CO3	3	2	2	2	1	2	2			2		3
CO4	2	3	2	2	2	3		3		2	3	2
Average	**2.2**	**1.8**	**2.2**	**2**	**1.0**	**2.2**				**2**	**1.7**	**2**

CO-PSO Mapping

	PSO1	PSO2	PSO3
CO1	2	2	1
CO2	2	2	3
CO3	1	2	2
CO4	2	2	3
Average	**1.8**	**2.2**	**2.4**

Course Overview

The Microprocessors and Microcontrollers Lab course is designed to provide students with hands-on experience in working with microprocessors and microcontrollers and to enhance their understanding of embedded systems. This lab-based course complements the theoretical knowledge gained in the Microprocessors and Microcontrollers theory course, allowing students to apply their knowledge in practical applications. The course typically covers the topics: introduction, overview of microprocessors and microcontrollers and their applications. It gives differentiating information between microprocessors and microcontrollers based on their characteristics and applications. It gives information about writing assembly language programs to perform basic arithmetic operations and control flow. Practicing instruction sets, addressing modes, and assembly language programming techniques specific to the microprocessors being used. It also gives information about microcontroller Programming,overview of microcontroller architecture and programming model.

It gives information about interfacing microprocessors and microcontrollers with peripheral devices such as switches, LEDs, LCDs, and sensors. Implementing different communication protocols (UART, SPI, I2C) to establish communication with external devices. Configuring and utilizing timers, interrupts, and other peripheral modules for efficient device control and data transfer.

.

List of Experiments mapped with COs

S. No.	Name of the Experiments	Course Outcome
1	Write a program using 8085 and verify for: a). Addition of two 8-bit numbers. b). Addition of two 8-bit numbers (with carry).	CO1, CO4
2	Write a program using 8085 and verify for: a). 8-bit subtraction (display borrow) b).16-bit subtraction (display borrow)	CO2
3	3. Write a program using 8085 for multiplication of two 8- bit numbers by repeated addition method. Check for minimum number of additions and test for typical data.	CO3
4	Write a program using 8085 for multiplication of two 8- bit numbers by bit rotation method and verify.	CO2, CO3
5	Write a program using 8086 for finding the square root of a given number and verify.	CO5
6	Write a program using 8086 for copying 12 bytes of data from source to destination and verify.	CO4, CO3
7	Write a program using 8086 and verify for: a). Finding the largest number from an array. b). Finding the smallest number from an array.	CO3
8	Write a program using 8086 for arranging an array of numbers in descending order and verify.	CO1
9	Write a program using 8086 for arranging an array of numbers in ascending order and verify.	CO1
10	Write a program to interface a two-digit number using seven-segment LEDs. Use 8085/8086 microprocessor and 8255 PPI.	CO1,CO2
11	Write a program to control the operation of stepper motor using 8085/8086 microprocessor and 8255 PPI.	CO2
12	To study implementation & interfacing of Display devices Like LCD, LED Bar graph & seven segment display with Microcontroller 8051/AT89C51	CO2,CO3
13	To study implementation & interfacing of Different motors like stepper motor, DC motor & servo Motors.	CO2
14	Write an ALP for temperature & pressure measurement Write a program to interface a graphical LCD with 89C51	CO3
15	Write a program to interface a graphical LCD with 89C51	CO2,CO3

DOs and DONT's

Dos

1. Login-on with your username and password.
2. Log off the Computer every time when you leave the Lab.
3. Arrange your chair properly when you are leaving the lab.
4. Put your bags in the designated area.
5. Ask permission to print.

DON'Ts

1. Do not share your username and password.
2. Do not remove or disconnect cables or hardware parts.
3. Do not personalize the computer setting.
4. Do not run programs that continue to execute after you log off.
5. Do not download or install any programs, games or music on computer in Lab.
6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.
7. No Internet gaming activities allowed.
8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

General Safety Precautions

Precautions (In case of Injury or Electric Shock)

1. To break the victim with live electric source, use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.

2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.

3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.

4. Immediately call medical emergency and security. Remember! Time is critical; be best.

Precautions (In case of Fire)

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.

2. If fire continues, try to curb the fire, if possible, by using the fire extinguisher or by covering it with a heavy cloth if possible isolate the burning equipment from the other surrounding equipment.

3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.

4. Call security and emergency department immediately:

Emergency: Reception

Security : Main Gate

Guidelines to students for report preparation

All students are required to maintain a record of the experiments conducted by them. Guidelines for its preparation are as follows: -

1) All files must contain a title page followed by an index page. **The files will not be signed by the faculty without an entry in the index page.**

2) Student's Name, Roll number and date of conduction of experiment must be written on all pages.
3) For each experiment, the record must contain the following

(i) Aim/Objective of the experiment

(ii) Apparatus required with specification and Name plate details

(iii) Circuit diagrams, procedures, observations and calculations

(v) Results/ output

Note:

1. Students must bring their lab record along with them whenever they come for the lab.

2. Students must ensure that their lab record is regularly evaluated.

Lab Assessment Criteria

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment as well as internal marks in the lab course.

Grading Criteria	Exemplary (4)	Competent (3)	Needs Improvement (2)	Poor (1)
AC1: Pre-Lab written work (this may be assessed through viva)	Complete procedure with underlined concept is properly written	Underlined concept is written but procedure is incomplete	Not able to write concept and procedure	Underlined concept is not clearly understood
AC2: Program Writing/ Circuit connection	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied, Program/solution written is readable	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied	Assigned problem is properly analyzed & correct solution designed	Assigned problem is properly analyzed
AC3: Identification & Removal of Error / Troubleshooting	Able to identify errors/hardwiring and remove them	Able to identify errors/ circuit based errors and remove them with little bit of guidance	Is dependent totally on someone for identification of errors and their removal	Unable to understand the reason for errors even after they are explicitly pointed out
AC4:Execution & Demonstration	All variants of input /output are tested, Solution is well demonstrated and implemented concept is clearly explained	All variants of input /output are not tested, However, solution is well demonstrated and implemented concept is clearly explained	Only few variants of input /output are tested, Solution is well demonstrated but implemented concept is not clearly explained	Solution is not well demonstrated and implemented concept is not clearly explained
AC5:Lab Record Assessment	All assigned problems are well recorded with objective, design constructs and solution along with Performance analysis using all variants of input and output	More than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	Less than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	

LAB EXPERIMENTS

LAB EXPERIMENT 1(a)

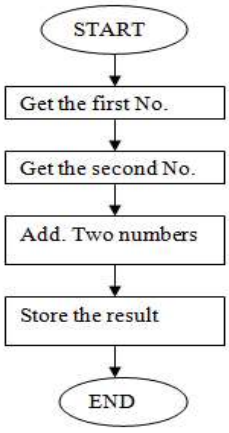
AIM : write a program using 8085 & verify for : Addition of two 8-bit numbers.

APPARATUS : 8085 microprocessor kit, 5V power supply, Keyboard.

THEORY (Program)

Memory address	Machine code	Mnemonics	Operands	Commands
7000	21,01,75	LXI	H,7501	Get address of 1st no. in HL pair
7003	7E	MOV	A,M	Move Ist no. in accumulator
7004	23	INX	H	HL points the address 7502H
7005	86	ADD	M	Add the 2nd no.
7006	23	INX	H	HL points 7503H
7007	77	MOV	M,A	Store result in 7503H.
7008	EF	RST 5		Terminate

FLOW CHART:



INPUT DATA

7501- 13H

7502- 12H

OUTPUT DATA

7503- 25H

PRECAUTIONS:-

1. Make sure that all the machine codes should be as per specified in the program.
2. Switch OFF the setup/ system when note in use.

EXPERIMENT NO. 1(b)

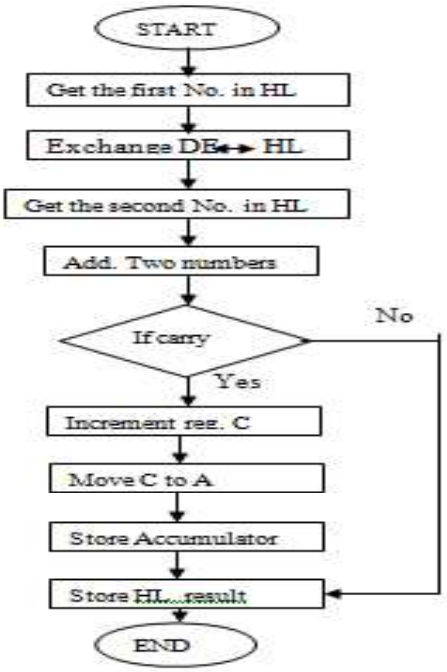
AIM : write a program using 8085 & verify for : Addition of two 16-bit numbers (with carry).

APPARATUS : 8085 microprocessor kit, 5V power supply, Keyboard.

THEORY (Program)

Memory address	Label	Machine code	Mnemonics	Operands	Commands
7000		2A,01,76	LHLD	7601H	Get 1st no. in HL pair from memory (7601)
7003		EB	XCHG		Exchange cont. of DE◆ HL
7004		2A,03,76	LHLD	7603H	Get 2st no. in HL pair from location 7603
7007		0E,00	MVI	C,00H	Clear reg. C.
7009		19	DAD	D	Get HL+DE & store result in HL
700A		D2,12,70	JNC	7012(loop)	If no carry move to loop/if carry then move to next step.
700D		0C	INR	C	Increment reg C
700E		79	MOV	A,C	Move carry from reg. C to reg. A
7011		32,02,75	STA	7502	Store carry at 7502H
7012	loop	22,00,75	SHLD	7500	Store result in 7500H.
7015		EF	RST 5		Terminate

FLOW CHART:-



INPUT DATA

7601 : 13H

7602 : 31H

7603 : 12H

7604 : 10H

OUTPUT DATA

7500 : 25H

7501 : 41H

7502 : 00H

PRECAUTIONS:-

1. Make sure that all the machine codes should be as per specified in the program.
2. Switch OFF the setup when note in use.

EXPERIMENT NO. 2(a)

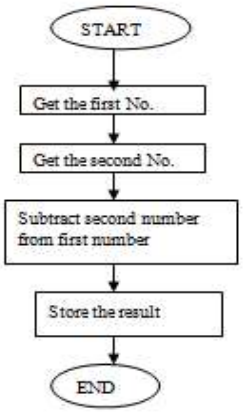
AIM : Write a program using 8085 & verify for : Subtraction of two 8-bit numbers. (display of barrow).

APPARATUS : 8085 microprocessor kit, 5V power supply, Keyboard.

THEORY(Program) :

Memory address	Opcode	Mnemonics	Operands	Comments
7000	21,01,75	LXI	H, 7501	Get address of ist no. in HL pair
7003	7E	MOV	A, M	Move Ist no. in accumulator
7004	23	INX	H	HL points 7502H.
7005	96	SUB	M	Substract 2nd no. from Ist no.
7006	23	INX	H	HL points 7503 H.
7007	77	MOV	M, A	Move contents of acc. to memory
7008	CF	RST 1		Stop

FLOW CHART :-



INPUT DATA

7501 : 20H

7502 : 10H

OUTPUT DATA

7503 : 10H

PRECAUTIONS:-

1. Make sure that all the machine codes should be as per specified in the program.
2. Switch OFF the setup when note in use.

EXPERIMENT NO. 2 (b)

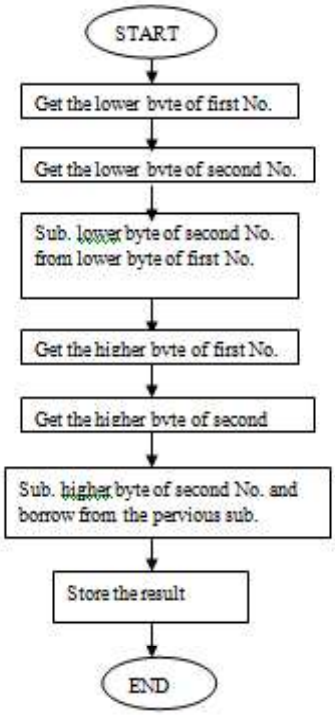
AIM : Write a program using 8085 & verify for : Subtraction of two 16-bit numbers. (display of barrow)

APPARATUS : 8085 microprocessor kit, 5V power supply, Keyboard.

THEORY (Program) :

Memory Address	Machine Code	Mnemonics	Operands	Comments
7000	2A, 01,75	LHLD	7501 H	Get 1st 16 bit no. in HL pair
7003	EB	XCHG		Exchange HL pair with DE.
7004	2A, 03,75	LHLD	7503 H	Get 2nd 16 bit no. in HL pair
7007	7B	MOV	A, E	Get lower byte of ist no.
7008	95	SUB	L	Subtract lower byte of 2nd no.
7009	6F	MOV	L, A	Store the result in reg. L
700A	7A	MOV	A, D	Get higher byte of Ist no.
700B	96	SBB	H	Subtract higher byte of 2nd no. with borrow
700C	67	MOV	H,A	Move from acc. To H
700D,E, F	22,05,75	SHLD	7505H	Store 16 bit result at 7505&7506
7010	EF	RST 5		Terminate

FLOW CHART:



INPUT DATA

7501 : 30H

7502 : 40H

7503 : 10H

7504 : 20H

OUTPUT DATA

7505 : 20H

7506 : 20H

PRECAUTIONS:-

1. Make sure that all the machine codes should be as per specified in the program.
2. Switch OFF the setup when note in use.

EXPERIMENT NO. 3

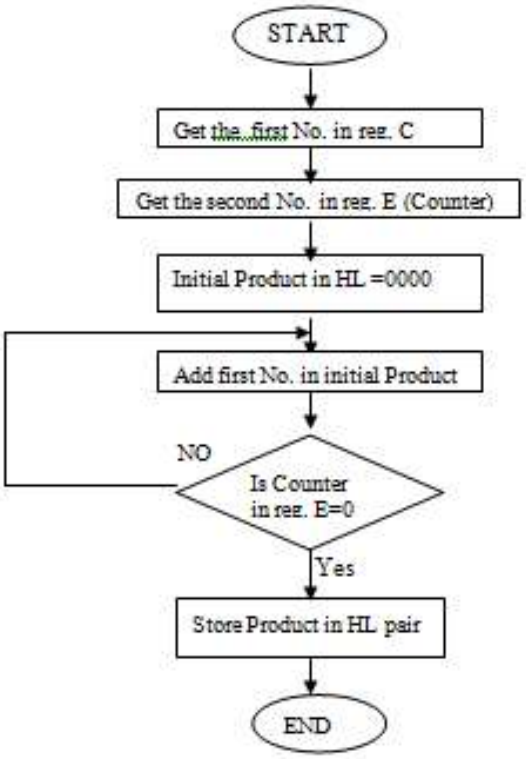
AIM : Write a program using 8085 for multiplication of two 8-bit numbers by repeated addition method check minimum number of addition & test for typical data.

APPARATUS : 8085 microprocessor kit, 5V power supply, Keyboard.

THEORY (Program) :

Memory Address	Label	Machine Code	Mnemonics	Operands	Comments
7000		0E,25	MVI	C,25	Move the no. in reg. C
7002		1E,05	MVI	E,05	Move the no. in reg. E
7004		06,00	MVI	B,00	Clear reg. B
7006		21,00,00	LXI	H,0000	Initial Product=0000
7009	UP1:	09	DAD	B	HL+BC=>HL
700A		1D	DCR	E	Decrement reg. E
700B		C2,09,70	JNZ	UP1(7009)	Jump if not zero to location up1
700E		22,00,75	SHLD	7500	Store HL at 7500
7011		CF	RST 1		Terminate

FLOW CHART:



INPUT DATA

Reg.C : 25H

Reg.E : 05H

Reg.B : 00H

OUTPUT DATA

HL pair : 00B9H

PRECAUTIONS:-

1.Make sure that all the machine codes should be as per specified in the program.

2. Switch OFF the setup when note in use

EXPERIMENT NO: 4

AIM: Write a program using 8085 for multiplication of two 8- bit numbers by bit rotation method and verify.

APPARATUS : 8085 microprocessor kit, 5V power supply, Keyboard.

Program:

Address	HEX Codes	Labels	Mnemonics	Comments
F000	21, 00, 80		LXI H,8000H	Point to first operand
F003	5E		MOV E,M	Load the first operand to E
F004	16, 00		MVI D,00H	Clear the register D
F006	23		INX H	Point to next location
F007	7E		MOV A,M	Get the next operand
F008	0E, 08		MVI C,08H	Initialize counter with 08H
F00A	21, 00, 00		LXI H, 0000H	Clear the HL pair
F00D	0F	LOOP	RRC	Rotate the acc content to right
F00E	D2, 12, F0		JNC SKIP	If carry flag is 0, jump to skip
F011	19		DAD D	Add DE with HL
F012	EB	SKIP	XCHG	Exchange DE and HL
F013	29		DAD H	Add HL with HL itself
F014	EB		XCHG	Exchange again the contents of DE and HL
F015	0D		DCR C	Decrease C register
F016	C2, 0D, F0		JNZ LOOP	if Z = 0, jump to LOOP
F019	22, 50, 80		SHLD 8050H	Store the result
F01C	76		HLT	Terminate the program

INPUT DATA:

8000: 25

8001: 2A

OUTPUT DATA

8051: 12

8050: 06

PRECAUTIONS:-

1.Make sure that all the machine codes should be as per specified in the program.

2. Switch OFF the setup when note in use

EXPERIMENT NO: 5

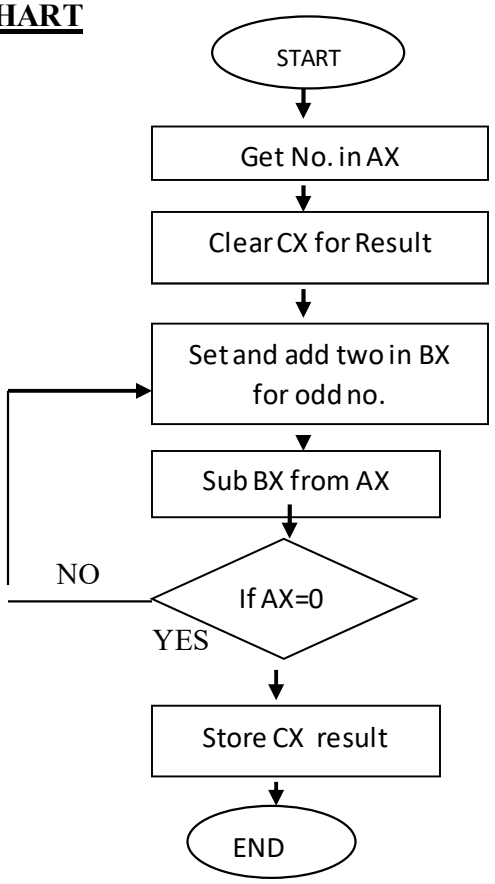
AIM: Write a program using 8086 for finding the square root of a given number and Verify.

APPARATUS REQUIRED:- 8086 Microprocessor Kit (NV5586A), Power Supply, Keyboard.

THEORY/PROGRAM :-

CS	IP	Machine code	Mnemonics	Comments
0000	0400	8B,06,00,05	MOV AX,[500]	Move the No. from memory to AX
0000	0404	B9,00,00	MOV CX,0000 H	Initialize the counter
0000	0407	BB,FF,FF	MOV BX,FFFF H	Set all 16 bits one in BX register
0000	040A	81,C3,02,00	UP: ADD BX,02	Add two to get the odd No.
0000	040E	41	INC CX	Increment CX by one
0000	040F	29,D8	SUB AX,BX	Sub BX from AX.
0000	0411	75,F7	JNZ UP	Jmp if AX is not zero
0000	0413	89,0E,00,06	MOV [600],CX	Move counter value to location
0000	0417	F4	HLT	

FLOW CHART



RESULT :

INPUT DATA

0500 : 09 H

OUTPUT DATA

0600 : 03 H

PRECAUTIONS:-

1.Make sure that all the machine codes should be as per specified in the program.

2. Switch OFF the setup when note in use

EXPERIMENT NO: 6

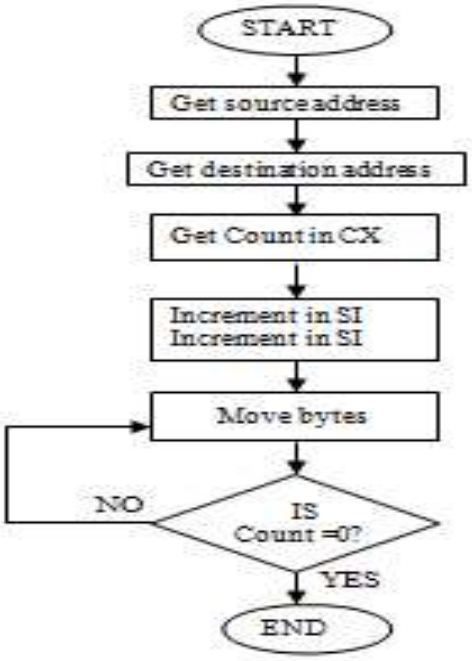
AIM: Write a program using 8086 for copying 12 bytes of data from source to destination & verify.

APPARATUS REQUIRED:- 8086 Microprocessor Kit (NV5586A), Power Supply, Keyboard.

THEORY/PROGRAM :-

Memory Address	Label	Machine Code	Mnemonics	Operands	Comments
0101		FC	CLD		Clear direction flag DF
0102		BE,00,03	MOV	SI,0300	Source address in SI
0105		BF,02,02	MOV	DI,0202	Destination address in DI
0108		8B,0C	MOV	CX,[SI]	Count in CX
010A		46	INC	SI	Increment SI
010B		46	INC	SI	Increment SI
010C	BACK	A4	MOV	SB	Move byte
010D		E2,FD	LOOP	BACK	Jump to BACK until CX =0
010F		CC	INT		Interrupt program

FLOWCHART:



RESULT :

INPUT DATA

0300 : 0B
0301 : 00
0302 : 03
0303 : 04
0304 : 05
0305 : 06
0306 : 15
0307 : 07
0308 : 12
0309 : 08
030A : 09
030B : 0A
030C : 0B
030D : 0E

OUTPUT DATA

0202 : 03
0203 : 04
0204 : 05
0205 : 06
0206 : 15
0207 : 07
0208 : 12
0209 : 08
020A : 09
020B : 0A
020C : 0B
020D : 0E

PRECAUTIONS:-

1. Make sure that all the machine codes should be as per specified in the program.

2. Switch OFF the setup when note in use

EXPERIMENT NO: 7 (a)

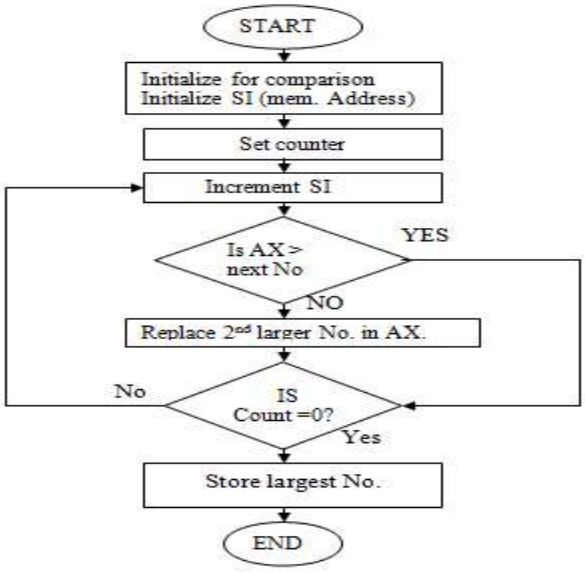
AIM: Write a program using 8086 & verify for finding the largest number from an data array.

APPARATUS REQUIRED:- 8086 Microprocessor Kit (NV5586A), Power Supply, Keyboard.

THEORY/PROGRAM :-

Memory Address	Label	Machine Code	Mnemonics	Operands	Comments
0101		B0,00,00	MOV	AX,0000	Initial value for comparison
0104		BE,00,02	MOV	SI,0200	Memory address in SI
0107		8B,0C	MOV	CX,[SI]	Count in CX
0109	BACK	46	INC	SI	Increment SI
010A		46	INC	SI	Increment SI
010B		3B,04	CMP	AX,[SI]	Compare previous largest number with next number
010D		73,02	JAE	GO	Jump if number in AX is larger i.eCF=0
010F		8B,04	MOV	AX,[SI]	Save next larger number in AX
0111	GO	E2,F6	LOOP	BACK	Jump to BACK until CX becomes zero
0113		A3,51,02	MOV	(0251),AX	Store largest number in memory
0116			HLT/INT 5		Interrupt program

FLOW CHART:



INPUT DATA

0200 : 05 H
0201 : 00 H
0202 : 41 H
0203 : 83 H
0204 : 58 H
0205 : 72 H
0206 : 39 H
0207 : 46 H
0208 : 53 H
0209 : 84 H
020A : 30 H
020B : 96 H

OUTPUT DATA

251 : 30 H
252 : 96 H

PRECAUTIONS:-

1.Make sure that all the machine codes should be as per specified in the program.

2. Switch OFF the setup when note in use

EXPERIMENT NO: 7(b)

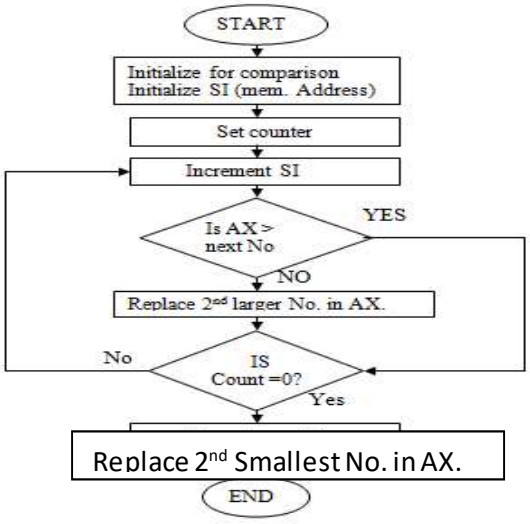
AIM: Write a program using 8086 & verify for finding the smallest number from an data array.

APPARATUS REQUIRED:- 8086 Microprocessor Kit (NV5586A), Power Supply, Keyboard.

THEORY/PROGRAM :-

Memory Address	Label	Machine Code	Mnemonics	Operands	Comments
0101		B0,00,00	MOV	AX,0000	Initial value for comparison
0104		BE,00,02	MOV	SI,0200	Memory address in SI
0107		8B,0C	MOV	CX,[SI]	Count in CX
0109	BACK	46	INC	SI	Increment SI
010A		46	INC	SI	Increment SI
010B		3B,04	CMP	AX,[SI]	Compare previous smallest number with next number
010D		73,02	JBE	GO	Jump if number in AX is larger i.eCF=0
010F		8B,04	MOV	AX,[SI]	Save next smaller number in AX
0111	GO	E2,F6	LOOP	BACK	Jump to BACK until CX becomes zero
0113		A3,51,02	MOV	(0251),AX	Store smallest number in memory
0116			HLT/INT 5		Interrupt program

FLOW CHART



INPUT DATA

0200 : 05 H
0201 : 00 H
0202 : 41 H
0203 : 83 H
0204 : 58 H
0205 : 72 H
0206 : 39 H
0207 : 46 H
0208 : 53 H
0209 : 84 H
020A : 30 H
020B : 96 H

OUTPUT DATA

251 : 39 H
252 : 46 H

PRECAUTIONS:-

1.Make sure that all the machine codes should be as per specified in the program.

2. Switch OFF the setup when note in use

EXPERIMENT NO: 8

AIM : Write a program using 8086 for arranging an array of numbers in descending order.

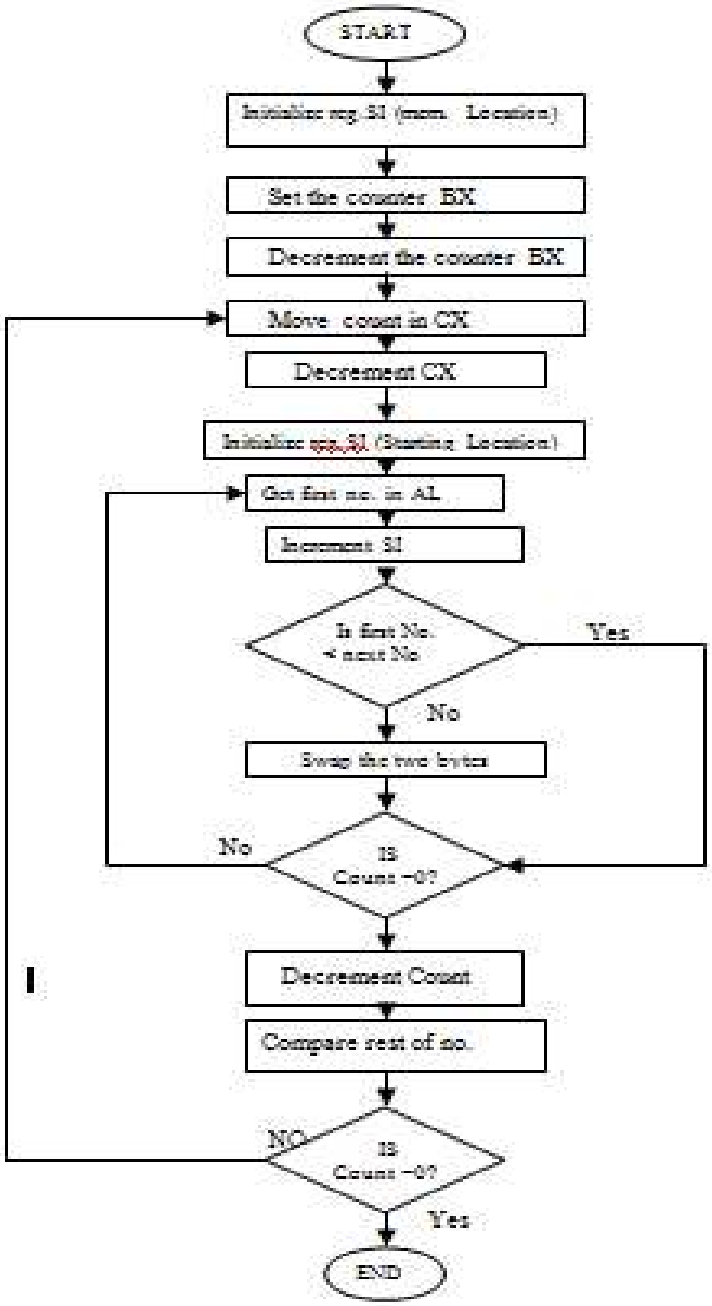
APPARATUS : 8086 microprocessor kit, 5V power supply, Keyboard.

THEORY/PROGRAM

Memory Address	Label	Machine Code	Mnemonics	Operands	Comments
0200		BE,00,03	MOV	SI,0300	Initialize SI Reg. with Memory Location. 0300.
0203		8B,1C	MOV	BX,[SI]	BX has no. of bytes
0205		4B	DEC	BX	Decrement the no. of bytes by one
0206	(3)	8B 0C	MOV	CX (SI)	Move no. of bytes in CX
0208		49	DEC	CX	Decrement the no. of bytes by one
0209		BE,02,03	MOV	SI,0303	Initialize SI reg. with the starting address of string
020C	(2)	8A,04	MOV	AL,[SI]	Move first data byte of string into AL
020E		46	INC	SI	Point at the next bytes of the string
020F		3A,04	COMP	AL,[SI]	Com. the two bytes of string.
0211		73,06	JAE	(1)	If two bytes are equal or 1st byte is above that the second byte branch to (1)
0213		86,04	XCHG	AL,[SI]	Else
0215		4E	DEC	SI	Second byte is less than first byte and swap the two bytes.
0216		88,04	MOV	[SI],AL	
0218		46	INC	SI	Point at next location of string
0219	(1)	E2,F1	LOOP	(2)	Loop if CX is not zero
021B		4B	DEC	BX	

021C		BE,00,03	MOV	SI,0300	
021F		75,E5	JNZ	(3)	
0221		F4	HLT		Halt.

FLOW CHART:



RESULTS:

INPUT DATA

0300 : 05
0301 : 00
0302 : 20
0303 : 25
0304 : 28
0305 : 15
0306 : 07

OUTPUT DATA

0302 : 28
0303 : 25
0304 : 20
0305 : 15
0306 : 07

PRECAUTIONS:-

1.Make sure that all the machine codes should be as per specified in the program.

2. Switch OFF the setup when note in use

EXPERIMENT NO: 9

AIM : **AIM:** Study of implementation, analysis and interfacing of seven segment display.

APPARATUS: NV5001 Microcontroller development Board, MC-04 Kit, power cord and connecting leads.

PROGRAM: for displaying "1234" on seven segment

```
SEG_A EQU P1.0
SEG_B EQU P1.1
SEG_C EQU P1.2
SEG_D EQU P1.3
------------------------------------------------------------------------------------------------------
ORG 0000H
JMP START
------------------------------------------------------------------------------------------------------
ORG 0200H
START:  MOV P0, #00H
MOV A, #00H
CLR C
LOOP: MOV P2, #0FFH
CLR SEG_A
CLR SEG_B
CLR SEG_C
CLR SEG_D
SETB SEG_A
MOV P2, #00000110B
LCALL  DELAY_1S
CLR SEG_A
CLR SEG_B
CLR SEG_C
CLR SEG_D
MOV P2, #00H
SETB SEG_B
MOV P2, #01011011B
LCALL  DELAY_1S
CLR SEG_B
CLR SEG_A
CLR SEG_C
CLR SEG_D
MOV P2, #00H
SETB SEG_C
MOV P2, #01001111B
LCALL  DELAY_1S
```

```
CLR SEG_C
CLR SEG_B
CLR SEG_A

CLR SEG_D
MOV P2, #00H
SETB SEG_D
MOV P2, #01100110B
LCALL  DELAY_1S
CLR SEG_D
CLR SEG_B
CLR SEG_A
CLR SEG_C
MOV P2, #00H
LJMP LOOP
-------------------------------------------------------------------------------------------------------
--------
DELAY_1S:  MOV R2, #06
DO3: MOV R3, #10
DHERE1:  MOV R4, #10
DAGAIN:  NOP
NOP
NOP
NOP
DJNZ R4, DAGAIN
DJNZ R3, DHERE1
DJNZ R2, DO3
RET
-------------------------------------------------------------------------------------------------------
--------
END
```

PROCEDURE:

1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch on the power supply.
4. Program seven segment display.hex file (Via CD - NV5001/ Modules programs\MC04 Display module\Seven segment module) in AT89C52 Microcontroller via programmer.
5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIF socket
6. Switch 'Off' the programmer switch in Programmer unit (in NV5001).
7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to seven segment display Interface block left side socket/connector (MC04) to Port P2 in NV5001 Trainer.

9. Connect 20 Pin FRC cable to seven segment Interface block right side socket/connector (MC04) to Port P1 in NV5001 Trainer.
10. Switch 'On' the power supply.
11. Observe "1234'' is coming on seven segment.
12. Observe the status of segment selection pins on tp28, tp29, tp30 & tp31.output waveform between RS pin of LCD (P3.5) and ground, on oscilloscope.
13. Observe the status of data bits on tp17 to tp24.

RESULT: "1234" displayed on seven segment display.

Question & Answer:

1. What is scanning in keyboard and what is scan time?
   Ans: The process of sending a zero to each row of a keyboard matrix and reading the columns for key actuation is called scanning. The scan time is the time taken by the processor to scan all the rows one by one starting from first row and coming back to the first row again.
2. What is programmable peripheral device?
   Ans: If the function performed by the peripheral device can be altered or changed by a program instruction then the peripheral device is called programmable device. It have control register. The device can be programmed by sending control word in the prescribed format to the control register.
3. What is baud rate?
   Ans:The baud rate is the rate at which the serial data are transmitted. Baud rate is defined as (The time for a bit cell). In some systems one bit cell has one data bit, then the baud rate and bits/sec are same.
4. What is a port?
   Ans:The port is a buffered I/O, which is used to hold the data transmitted from the microprocessor to I/O devices and vice versa.

EXPERIMENT NO. 10

AIM : Study and analyze the interfacing of 16 x 2 LCD.

APPARATUS: NV5001 Microcontroller development Board, MC-04 Kit, power cord and connecting leads.

PROGRAM: for displaying 'NVIS Technologies' on LCD.

```
RS_LCD EQU P3.5
RW_LCD EQU P3.6
E_LCD EQU P3.7
-------------------------------------------------------------------------------------------------------------
ORG 0000H
JMP START
ORG 0200H
START: MOV P0, #00H
MOV A, #00H
LCALL LCDINIT
MOV A, #086H
LCALL COMMAND
MOV A, #"N"
LCALL DISPLAY
MOV A, #"v"
LCALL DISPLAY
MOV A, #"i"
LCALL DISPLAY
MOV A, #"s"
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
-------------------------------------------------------------------------------------------------------------
   --
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
-------------------------------------------------------------------------------------------------------------
```

```
   --
MOV A, #082H
LCALL COMMAND

MOV A, #"T"
LCALL DISPLAY
MOV A, #"e"
LCALL DISPLAY
MOV A, #"c"
LCALL DISPLAY
MOV A, #"h"
LCALL DISPLAY
MOV A, #"n"
LCALL DISPLAY
MOV A, #"o"
LCALL DISPLAY
MOV A, #"l"
LCALL DISPLAY
MOV A, #"o"
LCALL DISPLAY
MOV A, #"g"
LCALL DISPLAY
MOV A, #"i"
LCALL DISPLAY
MOV A, #"e"
LCALL DISPLAY
MOV A, #"s"
LCALL DISPLAY
-------------------------------------------------------------------------------------------------
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LJMP START
-------------------------------------------------------------------------------------------------
DELAY_1S:
MOV R2, #11
LOOP3: MOV R3, #98
LOOP2: MOV R4, #106
LOOP1: NOP
NOP
NOP
NOP
NOP
DJNZ R4, LOOP1
DJNZ R3, LOOP2
DJNZ R2, LOOP3
```

```
RET
-------------------------------------------------------------------------------------------------------------
    -
LCDINIT: MOV A, #38H
LCALL  COMMAND
MOV A, #0CH
LCALL  COMMAND


MOV A, #01H
LCALL  COMMAND
MOV A, #06H
LCALL  COMMAND
RET
-------------------------------------------------------------------------------------------------------------
    -
COMMAND:
ACALL  READY
MOV P0, A
CLR RS_LCD
CLR RW_LCD
SETB E_LCD
CLR E_LCD
RET
-------------------------------------------------------------------------------------------------------------
    -
DISPLAY:
ACALL  READY
MOV P0, A
SETB RS_LCD
CLR RW_LCD
SETB E_LCD
CLR E_LCD
RET
-------------------------------------------------------------------------------------------------------------
    -
READY:  SETB P0.7
CLR RS_LCD
SETB RW_LCD
WAIT: CLR E_LCD
SETB E_LCD
JB P0.7, WAIT
RET
-------------------------------------------------------------------------------------------------------------
END
```

PROCEDURE:

1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector
(in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch on the power supply.
4. Program LCD interface module.hex file (Via CD - NV5001/ Modules programs\MC-04 Display module
\LCD module) in AT89C52 Microcontroller via programmer.
5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIF socket.
6. Switch 'Off' the programmer switch in Programmer unit (in NV5001).
7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to LCD Interface block left side socket/connector (MC04) to Port P3 in
NV5001 Trainer.
9. Connect 20 Pin FRC cable to LCD Interface block right side socket/connector (MC04) to Port P0 in NV5001 Trainer.

10. Turn contrast control potentiometer in LCD interface block to clockwise position (in MC04).
11. Turn Backlight control potentiometer in LCD interface block to anticlockwise position (in MC04).
12. Switch 'On' the power supply.
13. Observe "NVIS'' is coming on LCD and after some delay "Technologies".
14. Observe the output waveform between RS pin of LCD (P3.5) and ground, on oscilloscope.
15. Observe the output waveform between R/W pin of LCD (P3.6) and ground, on oscilloscope
16. Observe the output waveform between E pin of LCD (P3.7) and ground, on oscilloscope.
17. Turn contrast control potentiometer and observe the contrast change on LCD.
18. Turn Backlight control potentiometer and observe the change on backlight of LCD.

RESULT: 'NVIS Technologies' on displayed on 16x2 LCD.

Question & Answer:

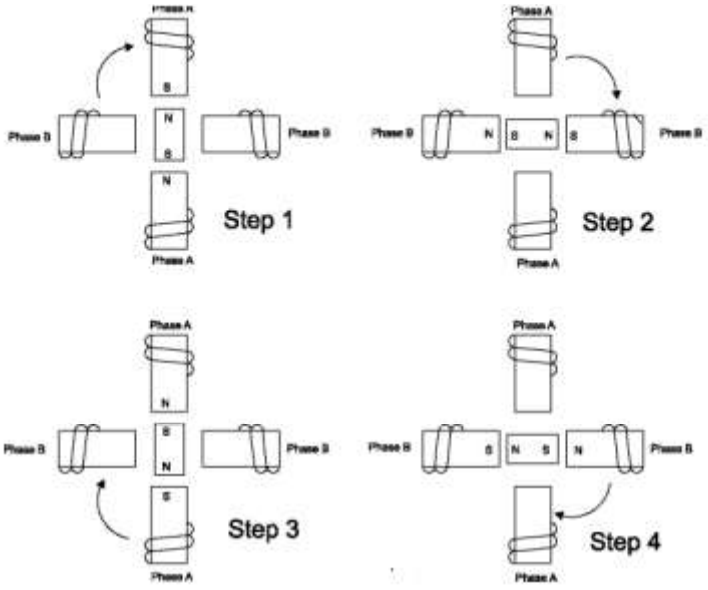
1. The end-of-conversion on the ADC0804 is done by using the: **$\overline{\text{EOC}}$**
2. What is the difference between the 8031 and the 8051: **The 8031 is ROM-less**.
3. The I/O port that does not have a dual-purpose role is: **Port 1**
4. The ADC0804 has resolution of: **8 Bit**
5. A HIGH on which pin resets the 8051 microcontroller: **RST**
6. An alternate function of port pin P3.1 in the 8051 is: **Serial Port Output**
7. An alternate function of port pin P3.0 (RXD) in the 8051 is: **Serial Port Input**
8. Magnetic tape is a **Direct Access Storage Device**
9. Which parts of the computer perform arithmetic calculation **ALU**
10. Vacuum tube based electronic computers are: **Hoover generation**

EXPERIMENT NO 11

AIM: Study of implementation of stepper motor angle control.

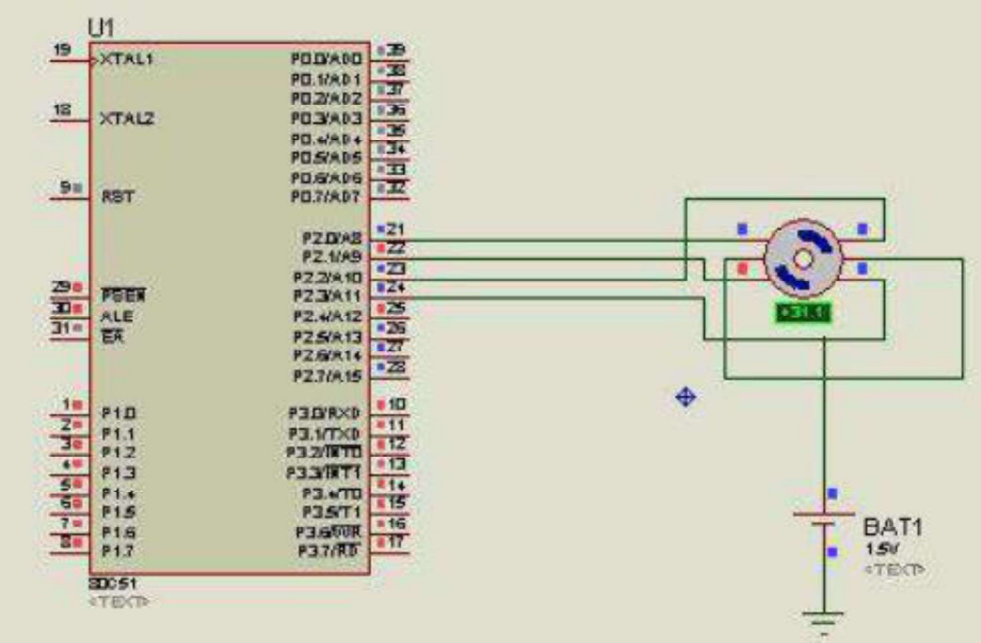
APPARATUS: NV5001 Microcontroller development Board, MC-05 Kit, power cord and connecting leads.

DESCRIPTION: A Stepper Motor or a step motor is a brushless, synchronous motor which divides a full rotation into a number of steps. Unlike a brushless DC motor which rotates continuously when a fixed DC voltage is applied to it, a step motor rotates in discrete step angles. The Stepper Motors therefore are manufactured with steps per revolution of 12, 24, 72, 144, 180, and 200, resulting in stepping angles of 30, 15, 5, 2.5, 2, and 1.8 degrees per step. The stepper motor can be controlled with or without feedback. Stepper motors work on the principle of electromagnetism. There is a soft iron or magnetic rotor shaft surrounded by the electromagnetic stators. The rotor and stator have poles which may be teethed or not depending upon the type of stepper. When the stators are energized the rotor moves to align itself along with the stator (in case of a permanent magnet type stepper) or moves to have a minimum gap with the stator (in case of a variable reluctance stepper). This way the stators are energized in a sequence to rotate the stepper motor.
The experiment has been designed to have a clear understanding of how motors are interfaced and controlled with microcontroller. The Motor drive module is made in such a way that student can understand the whole concepts of steeper motor. The object is to connect and program a microcontroller to do any operation with motors. It has input and output terminals for connection of external real world applications.



PROGRAM: To monitor the status of switch and rotate the stepper motor.

```
PROGRAM:
ORG 0000H
MOV A,#66H
LOOP:MOV P2,A
ACALL DELAY
RR A
SJMP LOOP
DELAY:MOV R5,#0AH
AGAIN:MOV R3,#0FFH
BACK:DJNZ R3,BACK
DJNZ R5,AGAIN
RET
END
```



PROCEDURE:

1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch 'On' the power supply.
4. Program angle control.hex file (Via CD - NV5001/ \Modules programs**MC05 Drive module**\Stepper motor interface module\Angle control program) in AT89C52 Microcontroller via programmer.
5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIF socket.
6. Switch 'Off' the programmer switch in Programmer unit (in NV5001).

7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to Stepper motor interface block socket (MC05) to Port P2 in NV5001 Trainer.
9. Switch 'On' the power supply.
10. Check the status of port pins on tp1 to tp 4 i.e., A to D pins.
11. Observe the status of Angle control switch 'On' SW2 line or tp 6.
12. Press Angle control switch and observe the direction of rotation of stepper motor.
13. Check the status of port pins on tp1 to tp 4 i.e., A to D pins.
14. Repeat steps 12 and 13 three times and observe the angle change of stepper motor.

RESULT: Stepper motor control is observed.

EXPERIMENT NO 12

AIM: Study and observation of Position control of Servo Motor.

APPARATUS: NV5001 Microcontroller development Board, MC-05 Kit, power cord and
connecting leads.

PROGRAM: To monitor the status of position control switch and control the angle of servo motor.

```
SERVO_PIN EQU P2.0
SW_PIN EQU P2.1
-------------------------------------------------------------------------------------------------
---------
ORG 0000H
JMP START
-------------------------------------------------------------------------------------------------
---------
ORG 0200H
START : CLR SERVO_PIN
SETB SW_PIN
LOOP_S : LCALL DELAY
SETB SW_PIN
SETB SERVO_PIN
LCALL DELAY_15MS_P
CLR SERVO_PIN
LCALL DELAY_16MS
JNB SW_PIN, SW_1_1
LCALL DELAY
SJMP LOOP_S
-------------------------------------------------------------------------------------------------
--
SW_1_1 : LCALL DELAY
SETB SW_PIN
SETB SERVO_PIN
LCALL DELAY_25MS_P
CLR SERVO_PIN
LCALL DELAY_16MS
LCALL DELAY
SJMP SW_1_1
-------------------------------------------------------------------------------------------------
--
DELAY_16MS : MOV R2, #150
DHERE1_16 : MOV R3, #32
DAGAIN_16 : NOP
DJNZ R3, DAGAIN_16
```

```
DJNZ R2, DHERE1_16
RET
-------------------------------------------------------------------------------------------------------------
--
DELAY_25MS_P : MOV R2, #20
DHERE1_25_P : MOV R3, #37
DAGAIN_25_P : NOP
DJNZ R3, DAGAIN_25_P
DJNZ R2, DHERE1_25_P
RET
-------------------------------------------------------------------------------------------------------------
--
DELAY_15MS_P : MOV R2, #20
DHERE1_15_P : MOV R3, #20
DAGAIN_15_P : NOP
DJNZ R3, DAGAIN_15_P
DJNZ R2, DHERE1_15_P
RET
DELAY : MOV R5, #250
DHERE1 : MOV R4, #220
DAGAIN : NOP
NOP
DJNZ R4, DAGAIN
DJNZ R5, DHERE1
RET
-------------------------------------------------------------------------------------------------------------
---
END
```

PROCEDURE:

1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch 'On' the power supply.
4. Program servo motor module.hex file (Via CD - NV5001/ \Modules programs\MC05 Drive module \DC motor interface module\Servo motor module) in AT89C52 Microcontroller via programmer.
5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIF socket
6. Switch 'Off' the programmer switch in Programmer unit (in NV5001).
7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to servo motor interface block socket (MC05) to Port P2 in NV5001 Trainer.
9. Switch 'On' the power supply.

10. Check the status of port pins on tp12 to tp13.
11. Observe servo motor rotates and stop in the centre position or in 90 degree angle.
12. Press position control switch and repeat steps 10.
13. Observe servo motor rotates and stop in l80 degree angle or in a left side position.

RESULT: Angle rotation of the servo motor observed.

Question & Answer:

1. What is a Data pointer register? **The data pointer register (DPTR) consists of a high byte(DPH) and a low byte (DPL) functions to hold 16 bit address. It may be manipulated as a 16-bit data register or as independent 8-bit registers. It serves as a base register in indirect jumps, look up table instructions and external data transfer.**
2.
3. What is Interrupt service register(ISR)? **The interrupt service register stores all the levels**

EXPERIMENT NO. 13

AIM: To study implementation and programming of Pressure measurement.

APPARATUS: NV5001 Microcontroller development Board, MC-15 Kit, power cord and connecting leads.

DESCRIPTION: Sensor module, MC15 has input and output terminals for connection of External real world applications. Pressure Sensor and Temperature Sensor Interface Module MC15 is generally used in the applications such as Monitoring and Controlling in Industries and many more.

PROGRAM:

```
#include <LPC214x.H>                                    /* LPC214x definitions */
#include <stdio.h>
/*****************************************************************************
* Function Prototypes
*****************************************************************************/
void LCD_Init(void);                                    /* LCD Init Function */
void LCD_Delay(unsigned int);                           /* LCD Delay Function */
void LCD_Cmd(unsigned long);                            /* LCD Command Function */
void LCD_Data(unsigned long);                           /* LCD Data Function */
void LCD_Disp(unsigned char, char *);                   /* LCD Display Function */
void display_lcd1 (unsigned char location, char *d);
void ADC_Init(void);                                    /* ADC Init Function */
void ADC_Convert(unsigned int);                         /* ADC Display Function */
void ADC_CALL(void);
/*****************************************************************************
**
* LCD Pin Out Description
*****************************************************************************
**/
#define RS_Set IO1SET = 0x20000000;
#define RS_Clr IO1CLR = 0x20000000;
#define EN_Set IO1SET = 0x80000000;
#define EN_Clr IO1CLR = 0x80000000;
unsigned int ADC_Val;                                   /* ADC Result (HEX)
*/
unsigned int FirstBit,SecondBit,ThrdBit,FourthBit;
/*****************************************************************************
**

*Delay
*Description : This function provide Delay in Mili Sec.
*****************************************************************************
**/
```

```
void LCD_Delay(unsigned int Time)
{
  unsigned int i,j;
  for(i=0;i<=Time;i++)
  for(j=0;j<110;j++);
}


/*****************************************************************************
***
* ADC initialization
* Description: This function initializes the LCD module by the following steps:
*1. Select ADC Channel
*2. Set A/D: 10-bit AIN0 @ 12MHz
* Note: This function should be called once before any of the other functions of ADC.
*****************************************************************************
**/
void ADC_Init()
{
PINSEL0 |= 0x00003000;                                  /* channel
AD1.0*/
AD1CR   = 0x01210400;                          /* Setup A/D: 10-bit AIN0 @ 3MHz
*/
}
/*****************************************************************************
***
* ADC Conversion
* Description: This function convert ADC data into ASCII by the following steps:
* 1. Convert each byte into ASCII
* 2. Each Value will be used for LCD Display
* Arguments   : 'RADC_Value' is the Value which needs to convert in ASCII.
*****************************************************************************
**/
void ADC_Convert(unsigned int ADC_Value)
{
 unsigned int X,Y,Z;                                /* Intermidiate
Variables */
 FirstBit=0,SecondBit=0,ThrdBit=0,FourthBit=0;
 X = ADC_Value/10;
 FirstBit  = ADC_Value%10;
 FirstBit  = 0x30|FirstBit;                          /* First Byte(LSB)
*/
 Y = X/10;
 SecondBit  = X % 10;
 SecondBit  = 0x30|SecondBit;                        /* Second
Byte */
 Z = Y/10;
```

```
 ThrdBit  = Y % 10;
 ThrdBit  = 0x30|ThrdBit;                                        /* Third Byte
*/
 FourthBit  = Z;                                                 /* Last Byte(MSB)
*/
 FourthBit=0x30|FourthBit;
}
/*************************************************************************
***
* LCD initialization
* Description : This function initializes the LCD module by the following steps:
* 1. Set 8bit : 2 Line 5x7 Dots (0x38)
* 2. Display On curser Off (0x0C)
* 3. Clear Display (0x01)
*4. Entry Mode (0x06)
* Note: This function should be called once before any of the other functions
*************************************************************************
*/
void LCD_Init(void)
{
LCD_Cmd(0x38);                                  /* Function Set 8bit : 2 Line 5x7 Dots
*/
LCD_Cmd(0x0C);                                  /* Display On curser Off
*/
LCD_Cmd(0x01);                                  /* Clear Display
*/


LCD_Cmd(0X06);                                  /* Entry Mode
*/
}
/*************************************************************************
***
* LCD Command (Shifting is Done here)
* Description : This function initializes the LCD module by the following steps:
* Note : Here we have selected Pin P1.16 to P1.23 as LCD data line that's
* why we need to shift data by 16.
*************************************************************************
**/
void LCD_Cmd(unsigned long Cmd)
{
 unsigned long Shifted_Cmd;
 Shifted_Cmd = Cmd << 16;   /* because We have selected P1.16 to P1.23 as LCD data
line */

 RS_Clr;                                                         /* RS Pin
Clear */
```

```
 LCD_Delay(10);                                          /* Delay for
Clock*/
 EN_Set;                                                 /* Enable Pin
SET */
 LCD_Delay(10);
 IO1SET = Shifted_Cmd;                          /* Write Command Value to LCD Data
Pin */
 LCD_Delay(20);                                        /* Delay for enable
Clock*/
 EN_Clr;                                                 /* Enable Pin
Clear */
 LCD_Delay(10);
 IO1CLR = 0x00FF0000;                                   /* Clear All pins of
LCD */
}

/*****************************************************************************
***
* LCD Data
* Description: This function initializes the LCD module by the following steps:
* 1. Set Register select(RS) to High
* 2. Set enable pin high to low
* 3. Send data to LCd data pin
* Note: Here we have selected Port 1 Pin P1.16 to P1.23 as LCD data line
* thats why we need to shift data by 16.
*****************************************************************************
**/
void LCD_Data(unsigned long Data)
{
 RS_Set;                                                 /* RS Pin
Clear */
 LCD_Delay(10);                                          /* Delay for
Clock*/
 EN_Set;                                                 /* Enable Pin
SET */
 LCD_Delay(10);
 IO1SET = Data << 16;                           /* Write Command Value to LCD Data
Pin */
 LCD_Delay(20);                                        /* Delay for enable
Clock*/
 EN_Clr;                                                 /* Enable Pin
Clear */
 LCD_Delay(10);
 IO1CLR = 0x00FF0000;                                   /* Clear All pins of
LCD */
}
```

```
/*****************************************************************************
***
* LCD Display
* Description : This function initializes the LCD module by the following steps:
* 1. Send Loc from where data needs to write


* 2. Display string on LCD
*****************************************************************************
**/
void LCD_Disp(unsigned char Loc, char *String)
{
 LCD_Cmd(Loc);                                          /* Send Command to
LCD */
 while(*String)                                        /*  wait untill Null char come
*/
{
        LCD_Data(*String++);                                  /* Write data to LCD
*/
        IO1CLR = 0x00FF0000;                                  /* Clear All pins of LCD
*/
       }
}

void display_lcd1 (unsigned char location,  char *d)
{

 unsigned long shift_data;
 shift_data=  0x80 | location;
 shift_data=  shift_data<<16;
 LCD_Cmd(shift_data);
 LCD_Delay(100);
 while(*d)
       {
        LCD_Data(*d++);
        LCD_Delay(100);
        IO1CLR = 0x00FF0000;                        // ADDED NEW

}

}

void ADC_CALL(void)
{
  AD1CR |= 0x01000000;                                  /* start of ADC conversion
*/
  do{
```

```
        ADC_Val = AD1DR0;                                                    /* 10 bit
value  */
       }while  ((ADC_Val  & 0x80000000) == 0);          /* Wait ADC Conversion  Complete
*/

  AD1CR &= ~0x01000000;                                                  /* Again  start
ADC */
  ADC_Val = (ADC_Val >> 6) & 0x03FE;
}

/*************************************************************************
***
* Main: Initialize  and start RTX Kernel
* Description  : This  function  Initialize  the LCD,RTC  and ADC and RTX Kernal
*************************************************************************
**/
int  main  (void)          /* program exec. starts here  */
{
 float  volt[20],dispvolt;
char k[5];int  i;
IO1DIR = 0xFFFF0000;                  /* Define  Port pin  P1.16 to P1.31 as Output  for
LCD */
LCD_Init();                                                              /* LCD Initialize
*/
ADC_Init();                                                               * ADC
Initialize  */

display_lcd1(0x80,"      Voltage");
while(1)
          {
            for(i=0;i<=19;i++)
            {
             ADC_CALL();
             volt[i]  = (ADC_Val * 3.3) / 1023.0;
             }
dispvolt  =
(volt[0]+volt[1]+volt[2]+volt[3]+volt[4]+volt[5]+volt[6]+volt[7]+volt[8]+volt[9]+volt[10]
+ volt[11]  +volt[12]+volt[13]+volt[14]+volt[15]+volt[16]+volt[17]+volt[18]+volt[19])/20;
sprintf(k,"%f",dispvolt  + 0.01);
LCD_Cmd(0xC6);
LCD_Data(k[0]);
LCD_Cmd(0xC7);
LCD_Data(k[1]);
LCD_Cmd(0xC8);
LCD_Data(k[2]);
LCD_Cmd(0xC9);
LCD_Data(k[3]);
```

```
LCD_Cmd(0xCA);
LCD_Data(k[4]);
LCD_Cmd(0xCC);
LCD_Data('V');
LCD_Delay(200000);
}
}
/*-------------------------------------------------------------------------
* end of file
*-------------------------------------------------------------------------*/
```

Pressure Sensor Module MC-15 with NV500X series

PROCEDURE:

1. Connect 20 Pin FRC cable between Sensor Module (MC15) & Port of NV500Xseries Trainer as mentioned in respective programs.
2. Connect LCD Module compatible with NV500X series Trainer LCD Data and Control Pins to Ports of NV500X series trainers as mentioned in respective programs (Optional).
3. Make the connection of Sensor Module with NV500X series as given in the respective program.
4. Observer the voltage that is being displayed on the LCD and note down the readings.

RESULT: Pressure measured and corresponding digital value observed on the LCD.

EXPERIMENT NO 14

AIM: To study implementation and programming of Temperature measurement.

APPARATUS: NV5001 Microcontroller development Board, MC-15 Kit, power cord and connecting leads.

PROGRAM:

```
#include <LPC214x.H>                                        /* LPC214x definitions
*/
#include <stdio.h>
/*****************************************************************************
**
 Function Prototypes

      ******************************************************************************
*******/
void LCD_Init(void);                                         /* LCD Init
Function */
void LCD_Delay(unsigned int);                                /* LCD Delay
Function */
void LCD_Cmd(unsigned long);                                 /* LCD Command
Function */ void LCD_Data(unsigned long);                              /* LCD
Data Function */  void LCD_Disp(unsigned char,unsigned char *);              /*
LCD Display Function */
void ADC_CALL(void);
void ADC_Init(void);                                         /* ADC Init
Function */
void ADC_Convert(unsigned int);                              /* ADC Display
Function */
/*****************************************************************************
**
* LCD Pin Out
Discription

      ******************************************************************************
*******/
#define RS_Set IO1SET = 0x20000000;
#define RS_Clr IO1CLR = 0x20000000;

#define RW_Set IO1SET = 0x40000000;
#define RW_Clr IO1CLR = 0x40000000;

#define EN_Set IO1SET = 0x80000000;
```

```
#define  EN_Clr IO1CLR = 0x80000000;

unsigned  int ADC_Val;                                          /* ADC Result
(HEX) */
unsigned  int FirstBit,SecondBit,ThrdBit,FourthBit;
/*****************************************************************************
***
       *Delay :
* Description : This function provide Delay in Mili Sec.
*****************************************************************************
***/
void Delay(unsigned  int Time)
{
  unsigned  int i,j;
  for(i=0;i<=Time;i++)
  for(j=0;j<1275;j++);
}


/*****************************************************************************
**
* LCD_Delay
* Description : This function provide Delay in Mili Sec.
*****************************************************************************
***/
void LCD_Delay(unsigned  int Time)
{
unsigned  int i,j;
for(i=0;i<=Time;i++)
 for(j=0;j<1005;j++);
}

/*****************************************************************************
****
*  ADC initialization
* Description : This function initializes  the LCD module  by the following  steps:
* 1. Select ADC Channel
* 2. Set A/D: 10-bit AIN0 @ 12MHz
* Note: This function  should  be called  once before any of the other functions  of ADC.
*****************************************************************************
*****/
void ADC_Init()
{
 PINSEL0 |= 0x00003000;                                         /* channel
AD1.0*/
AD1CR |= 0x01210400;
}
```

```
/*************************************************************************
***
* ADC Conversion
* Description : This function convert ADC data into ASCII by the following
steps:*

* 1. Conver each byte into ASCII
* 2. Each Value will be used for LCD Display
* Arguments : 'RADC_Value' is the Value which needs to convert in ASCII.
*************************************************************************
****/
void ADC_Convert(unsigned int ADC_Value)
{
 unsigned int X,Y,Z;                                    /* Intermidiate
Variables */
 FirstBit=0,SecondBit=0,ThrdBit=0,FourthBit=0;
 X = ADC_Value/10;
 FirstBit = ADC_Value%10;
 FirstBit = 0x30|FirstBit;                              /* First Byte
(LSB) */
   Y = X/10;
SecondBit = X % 10;
SecondBit = 0x30|SecondBit;                             /* Second
Byte */

 Z = Y/10;
ThrdBit = Y % 10;
ThrdBit = 0x30|ThrdBit;                                 /* Third
Byte */
 FourthBit = Z;                                         /* Last
Byte(MSB) */
 FourthBit=0x30|FourthBit;
}

/*************************************************************************
**
* LCD initialization
* Description : This function initializes the LCD module by the following steps:
* 1. Set 8bit : 2 Line 5x7 Dots (0x38)
* 2. Display On curser Off (0x0C)
* 3. Clear Display (0x01)
* 4. Entry Mode (0x06)
* Note : This function should be called once before any of the other functions
*************************************************************************
****/
void LCD_Init(void)
```

```
{
 LCD_Cmd(0x38);                                    /* Function Set 8bit : 2 Line 5x7
Dots */
 LCD_Cmd(0x0C);                                    /* Display On curser
Off */
 LCD_Cmd(0x01);                                    /* Clear
Display */
 LCD_Cmd(0X06);                                    /* Entry
Mode */
}
/*****************************************************************************
***
      *LCD Command (Shifting is Done here)
* Description : This function initializes the LCD module by the following steps:
* Note : Here we have selected Pin P1.16 to P1.23 as LCD data line thats why we need to
 * shift data by 16.
*****************************************************************************
****/
void LCD_Cmd(unsigned long Cmd)
{
unsigned long Shifted_Cmd;
Shifted_Cmd = Cmd << 16;        /* because we have selected P1.16 to P1.23 as LCD data
line */
// RW_Clr;                                          /* RW Pin
Clear */

RS_Clr;                                             /* RS Pin
Clear */
LCD_Delay(5);                                       /* Delay for
Clock*/
EN_Set;                                             /* Enable Pin
SET */
LCD_Delay(5);
IO1SET = Shifted_Cmd;                               /* Write Command Value to LCD Data
Pin */
LCD_Delay(10);                                      /* Delay for enable
Clock*/
EN_Clr;                                             /* Enable Pin
Clear */
LCD_Delay(5);
IO1CLR = 0x00FF0000;                                /* Clear All pins of
LCD */
}

/*****************************************************************************
*
* LCD Data
```

```
* Description : This function initializes the LCD module by the following steps:

* 1. Set Register select(RS) to High
* 2. Set enable pin high to low
* 3. Send data to LCd data pin
* Note: Here we have selected Port 1 Pin P1.16 to P1.23 as LCD data line thats why we need
to shift
        data by 16.
*********************************************************************************/

void LCD_Data(unsigned long Data)
{

RS_Set;                                                         /* RS Pin
Clear */
LCD_Delay(5);                                                   /* Delay for
Clock*/
EN_Set;                                                         /* Enable Pin
SET */
LCD_Delay(5);
IO1SET = Data << 16;                                    /* Write Command Value to LCD Data
Pin */
LCD_Delay(10);                                                  /* Delay for enable
Clock*/
EN_Clr;                                                         /* Enable Pin
Clear */
LCD_Delay(5);
IO1CLR = 0x00FF0000;                                            /* Clear All pins of
LCD */
}

/*********************************************************************************
***
* LCD Display
* Description : This function initializes the LCD module by the following steps:
* 1. Send Loc from where data needs to write
* 2. Display string on LCD
*********************************************************************************
***/

void LCD_Disp(unsigned char Loc, unsigned char *String)
{
 LCD_Cmd(Loc);                                                  /* Send Command to
LCD */
 while(*String != '\n')                                         /* Wait untill Null char
come */
```

```
{
LCD_Data(*String++);                                    /* Write data to
LCD */
LCD_Delay(10);
CLR = 0x00FF0000;                                       /* Clear All pins of
LCD */
 }
}
void ADC_CALL(void)
{
 do{
     ADC_Val = AD1DR0;                                  /* 10 bit
value */
    }while ((ADC_Val & 0x80000000) == 0);               /* Wait ADC Conversion
Complete */

AD1CR &= ~0x01000000;                                   /* Again start
ADC */
ADC_Val = (ADC_Val >> 6) & 0x03FE;
}


/*************************************************************************
***
* Main: Initialize and start RTX Kernel
* Description : This function Initialize the LCD,RTC and ADC and RTX Kernal
*************************************************************************
***/
int main (void)                                         /* program exec. starts
here   */
{
int i;
 float temp[10],
disptemp;IO1DIR  = 0xF8FF0000;              /* Define  Port pin P1.16 to P1.31 as Output for
LCD */
LCD_Init();                                             /* LCD
Initialize  */
ADC_Init();                                             /* ADC
Initialize*/
LCD_Cmd(0x01);                                          /* Clear
Display  */
while(1)                                                /* Loop
Continue*/
 {
LCD_Cmd(0x83);                                          /* Set Curser at
'0x83' */
LCD_Data('T');                                          /* Display ADC Value on
```

```
LCD */
LCD_Cmd(0x84);
LCD_Data('e');
LCD_Cmd(0x85);
LCD_Data('m');
LCD_Cmd(0x86);
LCD_Data('p');
LCD_Cmd(0x87);
LCD_Data('e');
LCD_Cmd(0x88);
LCD_Data('r');
LCD_Cmd(0x89);
LCD_Data('a');
LCD_Cmd(0x8A);
LCD_Data('t');
LCD_Cmd(0x8B);
LCD_Data('u');
LCD_Cmd(0x8C);
LCD_Data('r');
LCD_Cmd(0x8D);
LCD_Data('e');
for(i=0;i<=9;i++)
{
ADC_CALL();
temp[i] = ((ADC_Val * 10) /33);
}
  disptemp =
(temp[0]+temp[1]+temp[2]+temp[3]+temp[4]+temp[5]+temp[6]+temp[7]+temp[8]+temp[9])/
10;

 //   ADC_Val = ADC_Val
/4;
       /* Convert ADC Result into Actual Temperature*/
//ADC_Val = ((ADC_Val * 10) /33);
ADC_Convert(disptemp  + 4);


 /* Convert ADC value into ASCII for LCD Display  */

LCD_Cmd(0xC5);                                          /* Display ADC Value on
LCD */
LCD_Data(ThrdBit);
LCD_Cmd(0xC6);
LCD_Data(SecondBit);
LCD_Cmd(0xC7);
LCD_Data(FirstBit)  ;
LCD_Cmd(0xC8);
```

```
LCD_Data(0xDF)  ;
LCD_Cmd(0xC9);
LCD_Data('C') ;

Delay(10000);                                                    /* Delay
*/
}
}
/*-------------------------------------------------------------------------------------------------------
--
* end of file
*-------------------------------------------------------------------------------------------------------
---*/
```

Temperature Sensor Module MC-15 with NV500X series

PROCEDURE:

1. Connect 20 Pin FRC cable between Sensor Module (MC15) & Port of NV500Xseries Trainer as mentioned in respective programs.
2. Connect LCD Module compatible with NV500X series Trainer LCD Data and Control Pins to Ports of NV500X series trainers as mentioned in respective programs (Optional).
3. Make the connection of Sensor Module with NV500X series as given in the respective program.
4. Connect +12V DC supply from respective NV500X series trainer.
5. Observer the voltage that is being displayed on the LCD and note down the readings.

RESULT: Temperature varying digital readings observed.

Question & Answer:

Q1. What is temperature sensor?
Ans: It detects and sense the temperature of an object .
Q2. What are the units of temperature?
Ans. Celsius, Fahrenheit and degree

This lab manual has been updated by

Er. Monika Rani

(monika.rani@ggnindia.dronacharya.info)

Cross checked by

HoD

(Robotics and Automation Engineering)