NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE

(Accredited by NAAC, Approved by AICTE New Delhi, Affiliated to APJKTU)

Pampady, Thiruvilwamala(PO), Thrissur(DT), Kerala 680 588

DEPARTMENT OF MECHATRONICS

LAB WORK BOOK

MR331 MICROPROCESSORS AND MICROCONTROLLERS LAB

VISION

To mould true citizens who are millennium leaders and catalysts of change through excellence in education.

MISSION

NCERC is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values.

We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

ABOUT DEPARTMENT

- Established in: 2013
- Course offered: B.Tech Mechatronics Engineering
- Approved by AICTE New Delhi and Accredited by NAAC
- Affiliated to the University of Dr. A P J Abdul Kalam Technological University.

DEPARTMENT VISION

To develop professionally ethical and socially responsible Mechatronics engineers to serve the humanity through quality professional education.

DEPARTMENT MISSION

MD 1: The department is committed to impart the right blend of knowledge and quality education to create professionally ethical and socially responsible graduates.

MD 2: The department is committed to impart the awareness to meet the current challenges in technology.

MD 3: Establish state-of-the-art laboratories to promote practical knowledge of mechatronics to meet the needs of the society.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: Graduates shall have the ability to work in multidisciplinary environment with good professional and commitment.

PEO2: Graduates shall have the ability to solve the complex engineering problems by applying electrical, mechanical, electronics and computer knowledge and engage in lifelong learning in their profession.

PEO3: Graduates shall have the ability to lead and contribute in a team with entrepreneur skills, professional, social and ethical responsibilities.

PEO4: Graduates shall have ability to acquire scientific and engineering fundamentals necessary for higher studies and research.

PROGRAM OUTCOMES (PO'S)

Engineering Graduates will be able to:

PO 1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2. 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.

PO 3. 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.

PO 4. 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.

PO 5. 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.

PO 6. 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.

PO 7. 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.

PO 8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10. 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.

PO 11. 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.

PO 12. 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.

PROGRAM SPECIFIC OUTCOMES (PSO'S)

PSO 1: Design and develop Mechatronics systems to solve the complex engineering problem by integrating electronics, mechanical and control systems.

PSO 2: Apply the engineering knowledge to conduct investigations of complex engineering problem related to instrumentation, control, automation, robotics and provide solutions.

COURSE OUTCOME

C309.1	Design and implement programs on 8086 microprocessor
C309.2	To provide solid foundation on interfacing the external devices to the processor according to the user requirements
C309.3	Design and implement 8051 microcontroller based systems
C309.4	To Understand the concepts related to I/O and memory interfacing
C309.5	To learn about interfacing stepper motor working and its interfacing
C309.6	To learn about different types of flag registers and their changes while performing arithmetic operations, generation of waveforms using microcontroller

CO VS PO'S AND PSO'S MAPPING

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO 1	PSO 2
C309.1	3	-	3	3	-	-	-	-	3	-	-	2	2	2
C309.2	3	-	3	3	-	-	-	-	3	-	-	2	2	2
C309.3	3	-	3	3	-	-	-	-	3	-	-	2	2	2
C309.4	3	-	3	3	-	-	-	-	3	-	-	2	2	2
C309.5	3	-	3	3	-	-	-	-	3	-	-	2	2	2
C309.6	3	-	3	3	-	-	-	-	3	-	-	2	2	2
ATTAINMENT	3	0	3	3	0	0	0	0	3	0	0	**2**	**2**	**2**

Note: H-Highly correlated=3, M-Medium correlated=2, L-Less correlated=1

PREPARATION FOR THE LABORATORY SESSION
GENERAL INSTRUCTIONS TO STUDENTS

1. Read carefully and understand the description of the experiment in the lab manual. You may go to the lab at an earlier date to look at the experimental facility and understand it better. Consult the appropriate references to be completely familiar with the concepts and hardware.

2. Make sure that your observation for previous week experiment is evaluated by the faculty member and you have transferred all the contents to your record before entering to the lab/workshop.

3. At the beginning of the class, if the faculty or the instructor finds that a student is not adequately prepared, they will be marked as absent and not be allowed to perform the experiment.

4. Bring necessary material needed (writing materials, graphs, calculators, etc.) to perform the required preliminary analysis. It is a good idea to do sample calculations and as much of the analysis as possible during the session. Faculty help will be available. Errors in the procedure may thus be easily detected and rectified.

5. Please actively participate in class and don't hesitate to ask questions. Please utilize the teaching assistants fully. To encourage you to be prepared and to read the lab manual before coming to the laboratory, unannounced questions may be asked at any time during the lab.

6. Carelessness in personal conduct or in handling equipment may result in serious injury to the individual or the equipment. Do not run near moving machinery/equipment. Always be on the alert for strange sounds. Guard against entangling clothes in moving parts of machinery.

7. Students must follow the proper dress code inside the laboratory. To protect clothing from dirt, wear a lab coat. Long hair should be tied back. Shoes covering the whole foot will have to be worn.

8. In performing the experiments, please proceed carefully to minimize any water spills, especially on the electric circuits and wire.

9. Maintain silence, order and discipline inside the lab. Don't use cell phones inside the laboratory.

10. Any injury no matter how small must be reported to the instructor immediately.

11. Check with faculty members one week before the experiment to make sure that you have the handout for that experiment and all the apparatus.

AFTER THE LABORATORY SESSION

1. Clean up your work area.

2. Check with the technician before you leave.

3. Make sure you understand what kind of report is to be prepared and due submission of record is next lab class.

4. Do sample calculations and some preliminary work to verify that the experiment was successful

MAKE-UPS AND LATE WORK

Students must participate in all laboratory exercises as scheduled. They must obtain permission from the faculty member for absence, which would be granted only under justifiable circumstances. In such an event, a student must make arrangements for a make-up laboratory, which will be scheduled when the time is available after completing one cycle. Late submission will be awarded less mark for record and internals and zero in worst cases.

LABORATORY POLICIES

1. Food, beverages & mobile phones are not allowed in the laboratory at any time.

2. Do not sit or place anything on instrument benches.

3. Organizing laboratory experiments requires the help of laboratory technicians and staff. Be punctual.

SYLLABUS

Course code	Course Name	L-T-P - Credits	Year of Introduction
MR331	Microprocessors and Microcontrollers lab	0-0-3:1	2016

Prerequisite: MR303 Microprocessors and microcontrollers

Course Objectives

- To enable students to do basic programming in the microprocessors and microcontrollers.

List of Exercises/Experiments :

8086 programming using kits / MASM(Any 6)

1. 8086 kit familiarization.
2. Basic arithmetic and Logical operations
3. Square, Square root and Cube program
4. Data transfer program
5. Programming exercise using BCD and Hexadecimal numbers
6. Programming exercise : sorting ,searching and string
7. Interfacing with A/D and D/A converters
8. Interfacing with stepper motors
9. IBM PC programming : Basic programs using DOS and BIOS interrupts

8051 programming using kits (Any 6)

1. Addition and subtraction of 8 bit numbers and 16 bit numbers
2. Multi byte addition
3. Programs on Data Transfer Instructions
4. Square, Square root and Cube program
5. 8 bit multiplication and division
6. Interfacing with A/D and D/A converters
7. Waveform generation using 8051
7. Interfacing with stepper motors
8. Parallel interfacing –LCD

Expected outcome .

On completion of the course the student will be able

1. To carry out basic arithmetic and logical calculations on 8086 and 8051 processors
2. To understand the interface of 8086 and 8051 processors with external devices
3. To understand the applications of microprocessors and microcontroller based system

Text Book:

1. A.K. Roy, K.M. Bhurchandi, *Advanced Microprocessors and Peripherals* McGraw- Hill International
2. Muhammad Ali Mazidi, Janice GillipseMazidi, Rolin D. Mckinlay, *"8051 Microcontroller and Embedded Systems Using Assembly and C"* Pearson Education, 2010

INDEX

FINAL VERIFICATION BY THE FACULTY
TOTAL MARKS:

INTERNAL EXAMINER **EXTERNAL EXAMINER**

EXPERIMENT – 1

INTRODUCTION TO 8086 MICROPRCESSOR

AIM

To familiarize with the register organization and instruction set of 8086

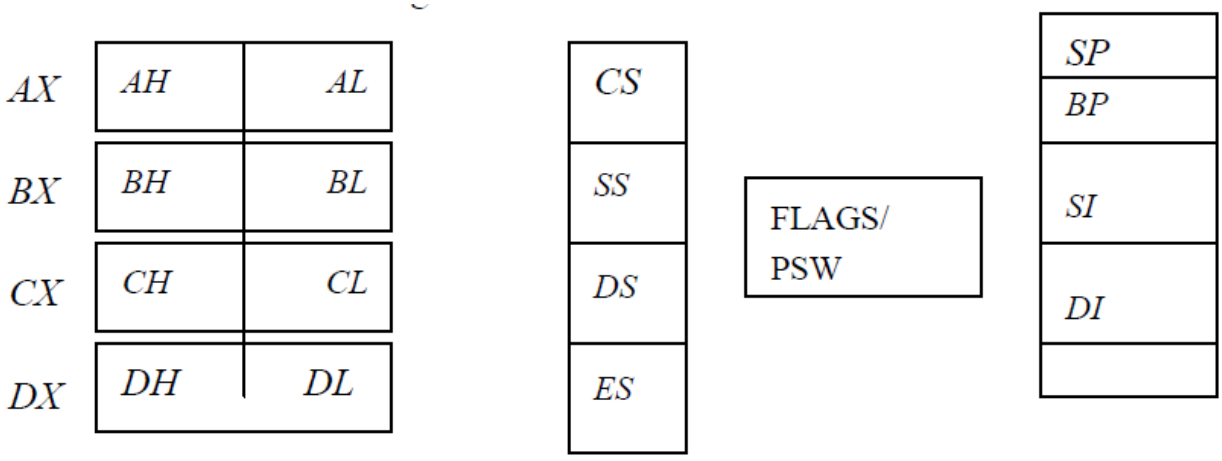
THEORY

Intel 8086 was the first 16-bit microprocessor introduced by Intel in 1978.

Register Organization of 8086

All the registers of 8086 are 16-bit registers. The general purpose registers can be used as either 8-bit registers or 16-bit registers.

The register set of 8086 can be categorized into 4 different groups. The register organization of 8086 is shown in the figure.



General Data Registers:

The registers *AX, BX,CX* and *DX* are the general purpose 16-bit registers. *AX* is used as 16-bit accumulator. The lower 8-bit is designated as *AL* and higher 8-bit is designated as *AH*. *AL* can be used as an 8-bit accumulator for 8-bit operation.

All data register can be used as either 16 bit or 8 bit. *BX* is a 16 bit register, but *BL* indicates the lower 8-bit of *BX* and *BH* indicates the higher 8-bit of *BX*.

The register *CX* is used default counter in case of string and loop instructions.

The register *BX* is used as offset storage for forming physical address in case of certain addressing modes.

DX register is a general purpose register which may be used as an implicit operand or destination in case of a few instructions.

Segment Registers:

The 8086 architecture uses the concept of segmented memory. 8086 able to address to address a memory capacity of 1 megabyte and it is byte organized. This 1 megabyte memory is divided into 16 logical segments. Each segment contains 64 Kbytes of memory.

Code segment register (CS): is used from addressing memory location in the code segment of the memory, where the executable program is stored.

Data segment register (DS): points to the data segment of the memory where the data is stored.

Extra Segment Register (ES): also refers to a segment in the memory which is another data segment in the memory.

Stack Segment Register (SS): is used for addressing stack segment of the memory. The stack segment is that segment of memory which is used to store stack data.

While addressing any location in the memory bank, the physical address is calculated from two parts:

- The first is segment address, the segment registers contain 16-bit segment base addresses, related to different segment.
- The second part is the offset value in that segment.

The advantage of this scheme is that in place of maintaining a 20-bit register for a physical address, the processor just maintains two 16-bit registers which is within the memory capacity of the machine.

Pointers and Index Registers.

The pointers contain offset within the particular segments.

- The pointer register *IP* contains offset within the code segment.
- The pointer register *BP* contains offset within the data segment.
- He pointer register *SP* contains offset within the stack segment.

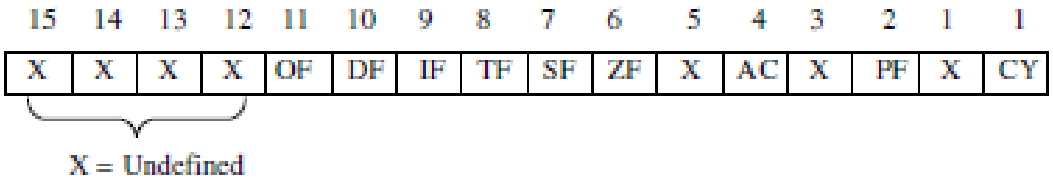
The index registers are used as general purpose registers as well as for offset storage in case of indexed, base indexed and relative base indexed addressing modes.

The register *SI* is used to store the offset of source data in data segment.

The register *DI* is used to store the offset of destination in data or extra segment.

The index registers are particularly useful for string manipulation.

Flag register



Flag Register of 8086

Conditional Flags

- Carry Flag (CY).
- Auxiliary Flag (AC)
- Parity Flag (PF)
- Zero Flag (ZF)
- Sign Flag (SF)

Control Flags

- Trap Flag (TF)
- Interrupt Flag (IF)
- Direction Flag (DF)

8086 INSTRUCTION SET SUMMARY

- **Data Transfer Instructions**

MOV	Move byte or word to register or memory
IN, OUT	Input byte or word from port, output word to port
LEA	Load effective address
LDS, LES	Load pointer using data segment, extra segment
PUSH, POP	Push word onto stack, pop word off stack
XCHG	Exchange byte or word
XLAT	Translate byte using look-up table

- **Logical Instructions**

NOT	Logical NOT of byte or word (one's complement)
AND	Logical AND of byte or word
OR	Logical OR of byte or word
XOR	Logical exclusive-OR of byte or word
TEST	Test byte or word (AND without storing)

- **Shift and Rotate Instructions**

SHL, SHR	Logical shift left, right byte or word by 1 or CL
SAL, SAR	Arithmetic shift left, right byte or word by 1 or CL
ROL, ROR	Rotate left, right byte or word by 1 or CL
RCL, RCR	Rotate left, right through carry byte or word by 1 or CL

- **Arithmetic Instructions**

ADD, SUB	Add, subtract byte or word
ADC, SBB	Add, subtract byte or word and carry (borrow)
INC, DEC	Increment, decrement byte or word
NEG	Negate byte or word (two's complement)
CMP	Compare byte or word (subtract without storing)
MUL, DIV	Multiply, divide byte or word (unsigned)
IMUL, IDIV	Integer multiply, divide byte or word (signed)
CBW, CWD	Convert byte to word, word to double word (useful before multiply/divide)
AAA, AAS, AAM, AAD	ASCII adjust for addition, subtraction, multiplication, division (ASCII codes 30-39)
DAA, DAS	Decimal adjust for addition, subtraction (binary coded decimal numbers)

- **Transfer Instructions**

JMP Unconditional jump (*short* 127/8, *near* 32K, *far* between segments)

Conditional jumps:

JA (JNBE)	Jump if above (not below or equal) +127, -128 range only
JAE (JNB)	Jump if above or equal (not below) +127, -128 range only
JB (JNAE)	Jump if below (not above or equal) +127, -128 range only
JBE (JNA)	Jump if below or equal (not above) +127, -128 range only
JE (JZ)	Jump if equal (zero) +127, -128 range only
JG (JNLE)	Jump if greater (not less or equal) +127, -128 range only
JGE (JNL)	Jump if greater or equal (not less) +127, -128 range only
JL (JNGE)	Jump if less (not greater nor equal) +127, -128 range only
JLE (JNG)	Jump if less or equal (not greater) +127, -128 range only
JC, JNC	Jump if carry set, carry not set +127, -128 range only
JO, JNO	Jump if overflow, no overflow +127, -128 range only
JS, JNS	Jump if sign, no sign +127, -128 range only
JNP (JPO)	Jump if no parity (parity odd) +127, -128 range only
JP (JPE)	Jump if parity (parity even) +127, -128 range only

Loop control:

LOOP	Loop unconditional, count in CX, short jump to target address
LOOPE (LOOPZ)	Loop if equal (zero), count in CX, short jump to target address
LOOPNE (LOOPNZ)	Loop if not equal (not zero), count in CX, short jump to target address
JCXZ	Jump if CX equals zero (used to skip code in loop)

- **Subroutine and Interrupt Instructions**

CALL, RET Call,	Return from procedure (inside or outside current segment)
IRET	Return from interrupt

- **String Instructions**

MOVS	Move byte or word string
MOVSB, MOVSW	Move byte, word string
CMPS	Compare byte or word string
SCAS	Scan byte or word string (comparing to A or AX)
LODS, STOS	Load, store byte or word string to AL or AX

Repeat instructions (placed in front of other string operations):

REP	Repeat
REPE, REPZ	Repeat while equal, zero
REPNE, REPNZ	Repeat while not equal (zero)

- **Processor Control Instructions**

Flag manipulation:

STC, CLC, CMC	Set, clear, complement carry flag
STD, CLD	Set, clear direction flag
STI, CLI	Set, clear interrupt enable flag
LAHF, SAHF	Load AH from flags, store AH into flags
PUSHF, POPF	Push flags onto stack, pop flags off stack

Coprocessor, multiprocessor interface:

ESC	Escape to external processor interface
LOCK	Lock bus during next instruction

Inactive states:

NOP	No operation
WAIT	Wait for TEST pin activity
HLT	Halt processor

RESULT

INFERENCE

EXPERIMENT 2

BASIC ARITHMETIC OPERATIONS AND LOGICAL OPERATIONS

I. 16 BIT BINARY ADDITION

AIM

To write an assembly language program to add two 16 bit binary numbers

ALGORITHM

1. Start
2. Load the first data in AX register.
3. Load the second data in BX register.
4. Add AX and BX register
5. Load SI register with the value
6. Clear CL register with value 0
7. Save the result in SI register and save the carry in SI + 2 register
8. Stop

PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENT
		BF0020	MOV SI,2000	Set SI register as pointer
			MOV AX,F897	Load 1st data in AX
		BB29E5	MOV BX,E529	Load 2nd data in BX
		B100	MOV CL, 00	Clear CL register
		01D8	ADD AX,BX	Add AX,BX
		D904	MOV [SI],AX	Store the sum in memory
		7302	JNCL	Jump on no carry
		FEC1	INC CL	If carry flag is set, increment CL
	L1	884C02	MOV [SI+2],CL	Store the carry in memory

RESULT

INFERENCE

II. 16 BIT BINARY SUBTRACTION

AIM

To write an assembly language program to subtract two 16 bit binary numbers

ALGORITHM

1 Start
2 Load the first data in AX register.
3 Load the second data in BX register.
4 Clear CL register
5 Subtract two data and get the difference in AX
6 Store the result in memory
7 Check for borrow
8 Increment CL if borrow is there
9 Store the borrow in memory
10 Stop

PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENT
		BE0030	MOV SI,3000	Set SI register as pointer
		B83000	MOV AX,30	Load 1st data in AX
		BB6000	MOV BX,60	Load 2nd data in BX
		B100	MOV CL, 00	Clear CL register
		29DB	SUB AX,BX	Subtract AX,BX and get the difference in AX
		7304	JNCL	Check for borrow
		FTD8	NEG AX	Negative AX
		FFC1	INC CL	If borrow, increment CL
	L1	8904	MOV[SI],AX	Store the difference in memory
			MOV [SI+2],CL	
		CC	INT03	Stop the program

RESULT

INFERENCE

III. 16 BIT BINARY MULTIPLICATION

AIM

To write an assembly language program to multiply two 16 bit binary numbers

ALGORITHM

1 Start
2 Load the first data in AX register.
3 Load the second data in BX register.
4 Multiply AX and BX register and get the product in registers AX and DX
5 Store the product in memory
6 Stop

PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENT
		B85804	MOV AX,458	Load 1st data in AX
		BBB70A	MOV BX,AB7	Load 2nd data in BX
		F7E3	MUL BX	Multiply AX,BX
		BE0030	MOV SI, 3000	Declare pointer SI
		8904	MOV [SI],AX	
		89540	MOV [SI+2] DX	
		CC	INT03	Stop the program

RESULT

INFERENCE

IV. 16 BIT BINARY DIVISION

AIM

To write an assembly language program to divide two 16 bit binary numbers

ALGORITHM

1 Start
2 Load the first data in AX register.
3 Load the second data in BX register.
4 Clear the DX register
5 Divide AX and BX register and get the quotient will be stored in AX and the remainder in DX
6 Store the result in memory
7 Stop

PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENT
		BE0030	MOV SI, 3000	Declare pointer SI
		B80500	MOV AX,0005	Load 1st data in AX
		BB0200	MOV BX,0002	Load 2nd data in BX
		BA0000	MOV DX, 00	Load 00 in to DX
		F7F3	DIV BX	Divide AX by BX
		8904	MOV [SI],AX	
			MOV [SI+2] DX	
		CC	INT03	Stop the program

RESULT

INFERENCE

V. LOGICAL OPERATIONS

AIM

To write an assembly language program to AND, OR & NOT logical operations

ALGORITHM

1. Start
2. Load the first data in AL register.
3. Load the second data in BL register.
4. OR AL and BL register
5. Store the product in memory
6. Again repeat the procedure for AND & NOT operation
7. Stop

PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENT
			MOV AL, 5	Load 1st data in AL
			MOV BL, 3	Load 2nd data in BL
			OR AL, BL	OR 2 data
			MOV CL, AL	Load AL into CL
			MOV AL, 5	Again Load data in AL
			AND AL, BL	AND 2 data
			MOV DL, AL	Load AL into DL
			MOV AL, 5	Again Load data in AL
			NEG AL	NOT AL
			MOV DH, AL	Load AL into DH
			INT 03	Stop the program

RESULT

INFERENCE

EXPERIMENT – 3

PROGRAMMING EXERCISE USING BCD AND HEXADECIMAL NUMBERS

AIM

To write an assembly language program to add two 16 bit BCD numbers

PROBLEM ANALYSIS

The 8086 processor will perform only binary addition. Hence for BCD addition the binary addition of data is performed and then the sum is corrected to get the result in BCD. After binary addition the following corrections should be made to get the result in BCD

1. If the sum of lower nibble exceeds 9 or if there is auxiliary carry then 6 is added to lower nibble
2. If the sum of upper nibble exceeds 9 or if there is auxiliary carry then 6 is added to upper nibble.

The above correction is taken care by DAA (Decimal Adjust Accumulator) instruction. Therefore after binary addition execute DAA instruction to do the above correction in the sum

ALGORITHM

1. Load the address of data in SI register.
2. Clear CL register for account for carry
3. Load the first data in AX register and the second data I BX register.
4. Perform binary addition of low byte of data to get the sum in AL register.
5. Adjust the sum of low bytes to BCD.
6. Save the sum of low bytes in DL register.
7. Get the high byte of first data in AL register.
8. Add the high byte of second data and previous carry to AL register Now the sum of high bytes will be in AL register
9. Adjust the sum of high bytes to BCD
10. Save the sum of high bytes in DH register
11. Check for carry .If carry flag is set then go to the next step,otherwise go to step 13
12. Increment CL register
13. Save the sum (DX register) in memory

14. Save the carry (CL register) in memory
15. Stop.

PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENT
		BE3000	MOV SI,3000	Set SI register as pointer
		B100	MOV CL,00	Clear CL register
		8B04	MOV AX,[SI]	Get first data in AX register
		8B 5C 02	MOV BX, [SI+2]	Load the second data I BX register
		02 C3	ADD AL, BL	get the sum of low byte of data in AL
		27	DAA	Adjust the sum of low bytes to BCD.
		8 D0	MOV DL,AL	Save the sum of low bytes in DL register.
		8A C4	MOV AL,AH	MOV the high byte of data to AL register
		12 C7	ADC AL,BH	Get the sum of high bytes will be in AL register
		27	DAA	Adjust the sum to BCD.
		88 4C06	MOV DH,AL	Save the sum of high bytes in DH register
		73 02	JNC AHEAD	Check for carry flag
		FE C1	INC CL	If carry flag is set then increment CL by one
	AHEAD	895404	MOV [SI+4],DX	Store the sum in memory
			MOV [SI+6],CL	Store the carry in memory
		CC	INT03	Stop the program

RESU1LT

INFERENCE

EXPERIMENT -4

PROGRAMMING EXERCISE: SORTING, SEARCHING AND STRING

1. FINDING SMALLEST NUMBER IN AN ARRAY

AIM

To write an assembly language program to find the smallest number in an array

PROBLEM ANALYSIS

The smallest number in an array of N elements is found out by N-1 comparisons of the consecutive elements. Each time two elements are compared to find the smallest among them and the obtained smallest number is again compared with the next element. This process is repeated until all the elements are compared. And final result will be the smallest number in the given array.

ALGORITHM

1. Load the starting address of the array in SI register.
2. Load the address of result in DI register
3. Load the number of bytes in the array in CL register
4. Increment array pointer(SI)
5. Get the first byte of array in AL
6. Decrement byte count(CL)
7. Increment array pointer(SI)
8. Get the next byte of array in BL
9. Compare current smallest (AL) and next byte (BL)
10. Check carry flag, If carry flag set then go to step 12, otherwise go to next step
11. Mov BL to AL
12. Decrement byte count(CL)
13. Check zero flag, If zero flag is reset then go to step 7, otherwise go to next step
14. Save the smallest number in the memory pointed by DI
15. Stop the program

PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENT
			MOV SI,1100	Set SI register as array pointer
			MOV DI,1200	Set DI as pointer of result
			MOV CL,[SI]	Set CL as count
			INC SI	Increment address pointer
			MOV AL, [SI]	Set first data as samllest
			DEC CL	Decrement the count of N-1 comparisons
		AGAIN	INC SI	Increment the pointer
			MOV BL,[SI]	Get the next element of array in BL register
			CMP BL, AL	Compare the content of AL and BL
			JC AHEAD	If AL<BL proceed to ahead
			MOV AL,BL	If AL>BL, copy the smaller BL to AL
		AHEAD	DEC CL	Decrement the count for repetitions
			JNZ AGAIN	Repeat the N-1comparisons until CL is zero
			MOV [DI], AL	Store the result
			INT 03	Stop the program

RESULT

INFERENCE

II. SORTING AN ARRAY IN ASCENDING ORDER

AIM

To write an assembly language program to sort an array of data in ascending order.

PROBLEM ANALYSIS

The array can be sorted in ascending order by bubble sorting. In bubble sorting of N data, N-1 comparisons are performed by taking two consecutive data at a time. After each comparison the two data can be rearranged in the ascending order

ALGORITHM

1. Load the starting address of the array in SI register.
2. Set CL register as count forN-1 repetition
3. Initialize array pointer
4. Set CH register as count for N-1 comparisons
5. Increment array pointer
6. Compare the next element of array with AL
7. Check the carry flag, if carry flag is set then go to step 12 otherwise go to next step.
8. Exchange the content of memory pointed by SI and AL register
9. Decrement the count for comparison (CH register)
10. Check zero flag, If zero flag is reset then go to step6, otherwise go to next step
11. Decrement the count for repetition (CL register)
12. Check zero flag, If zero flag is reset then go to step3, otherwise go to next step
13. Stop the program

PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENT
		BE 00 70	MOV SI,0700	Set SI register as pointer
		8A 0C	MOV CL,[SI]	Set CL as count
		FE C9	DEC CL	Decrement the count
	REP	BE 00 70	MOV SI,0700	Set SI register as pointer
		8A 2C	MOV CH , [SI]	Set CH register as counter for N-1 comparisons

		FE CD	DEC CH	Decrement the count of N-1 comparisons
		46	INC SI	Increment the pointer
	STORE	8A 04	MOV AL,[SI]	Get the element of array in CL register
		46	INC SI	Increment the pointer
		3A 04	CMP AL,[SI]	Compare with next element of array in memory
		72 05	JC AHEAD	If AL is less than memory then go to AHEAD
		86 04	XCHG AL,[SI]	If AL is not less than memory then exchange the content of memory and AL register
		4E	DEC SI	Decrement the pointer
		8A 04	MOV [SI] , AL	
		46	INC SI	
	AHEAD	FE CD	DEC CH	Decrement the count for comparisons
		75 F0	JNZ STORE	Repeat the comparisons until CH is zero
		FE C9	DEC CL	Decrement the count for repetitions
		75 E4	JNZ REP	Repeat the N-1comparisons until CL is zero
		CC	INT03	Stop the program

RESULT

INFERENCE

EXPERIMENT -5

DATA TRANSFER PROGRAM

AIM

To write an assembly language program to count the occurrence of a given number.

ALGORITHM

1. Start
2. Set SI as pointer
3. Set CX as counter
4. Move the contents of SI to AL
5. Decrement cx register
6. Increment the SI reg
7. Exchange the content of SI with AL
8. Decrement SI
9. Move AL to SI
10. Increment the SI reg
11. Increment the SI reg
12. Repeat until CX=0
13. Stop the program

PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENT
			MOV SI,1000	Set SI as pointer
			MOV CX,0010	Set CX as counter
	L1		MOV AL,[SI]	Move the contents of SI to AL
			DEC CX	Decrement cx register

			INC SI	Increment the SI reg
			XCHG AL,[SI]	Exchange the content of SI with AL
			DEC SI	Decrement SI
			MOV [SI],AL	Move AL to SI
			INC SI	Increment the SI reg
			INC SI	Increment the SI reg
			LOOP L1	Repeat until CX=0
			INT 03	Stop the program

RESULT

INFERENCE

EXPERIMENT – 6

INTERFACING STEPPER MOTOR WITH 8086

AIM

To rotate a stepper motor in anticlockwise direction

ALGORITHM

1. Load AL with 80 to set the control word
2. Move the content of AL to control register
3. Load AL with data FA and move that to Port A for rotation
4. Call the delay sub routine
5. Load AL with data F6 and move that to Port A for rotation
6. Call the delay sub routine
7. Load AL with data F5 and move that to Port A for rotation
8. Call the delay sub routine
9. Load AL with data F9 and move that to Port A for rotation
10. Call the delay sub routine
11. Jump to step 3

PROGRAM

ADDRESS	LABEL	OPCODE	MNEMONICS	COMMENT
		B0 80	MOV AL,80	Input the control word
		E6 76	OUT 76,AL	Out the control word to port
	START	B0 FA	MOV AL, FA	Load the first data in AL
		E6 70	OUT 70,AL	Move the data to output port
		E8 04 22	CALL DELAY1	Call the delay sub routine
		B0 F6	MOV AL, F6	Load the next data in AL
		E6 70	OUT 70,AL	Move the data to output port
		E8 04 22	CALL DELAY1	Call the delay sub routine

		B0 F5	MOV AL, F5	Load the next data in AL
		E6 70	OUT 70,AL	Move the data to output port
		E8 04 22	CALL DELAY1	Call the delay sub routine
		B0 F9	MOV AL, F9	Load the next data in AL
		E6 70	OUT 70,AL	Move the data to output port
		E8 04 22	CALL DELAY	Call the delay sub routine
		EB E2	JMP START	Jump to the label start for repeat
	DELAY1	B9 00 08	MOV CX,800	Move count to CX register
	LP1	49	DEC CX	Decrement count in CX
		75 FD	JNZ LP1	If not zero, then jump to label LP1.
		C3	RET	Stop the program

RESULT

INFERENCE

EXPERIMENT NO 7

BASIC PROGRAMS USING 8051

AIM

To familiarize with 8051 and to execute some basic programs on 8051 trainer kit.

APPARATUS REQUIRED

1. 8051 Trainer kit
2. User manual
3. Keyboard

THEORY

A Microcontroller is a powerful CPU tightly coupled with memory, various input output features such as serial port, parallel port, timer/counter interrupt controller etc. 8051 is a powerful 8 bit microcontroller on a single chip. It is optimized for control applications 64kB program memory address space of which 4 kB on chip. Major features of 8051 are:

1.8bit CPU optimized for control applications.

2. Extensive Boolean processing capabilities.

3. 64kB data and program memory addressing.

4.4kB of on chip program memory

5. On chip oscillation and clock circuitry.

I. PROGRAM TO PERFORM 16BIT ADDITION

ADDRESS	LABEL	MNEMONICS	COMMENT
3000		CLR C	Clear carry flag
		MOV A,#34	Copy 34 to A
		ADD A,#78	Add immediate value 78 with the content of the Acc
		MOV DPTR,#4100	Initialize memory location
		MOVX @DPTR,A	Move the contents from Acc to

			memory
		INC DPTR	Increment data pointer
		MOV A,#12	Copy 12 to Acc
		ADDC A,#56	Add with carry, the contents of Acc with 56
		MOVX @DPTR,A	Copy the contents of Acc to memory
	STOP:	SJMP STOP	

RESULT

INFERENCE

II PROGRAM TO PERFORM 8BIT SUBTRACTION

ADDRESS	LABEL	MNEMONICS	COMMENT
3000		CLR C	Clear carry flag
		MOV A,#20	Copy 20 to a
		SUB A,#10	Subtract the immediate value from the Acc content
		MOV DPTR,#4100	Initialize memory location
		MOVX @DPTR,A	Move the content from Acc to memory location
	STOP:	SJMP STOP	

RESULT

INFERENCE

EXPERIMENT NO 8

MULTI BYTE ADDITION

AIM

To write an assembly language program to find sum of elements in an array of size 10

PROGRAM

ADDRESS	LABEL	MNEMONICS	COMMENT
3000		MOV DPTR,#4200	Initialize memory location
		MOVX A,@DPTR	Copy the contents form memory location to accumulator
		MOV R0,#10	Copy the contents from acc to reg R0
		MOV B,#00	Clear B reg
		MOV R1,B	Copy the contents from B to R1
		CLR C	Clear carry flag
	LBL1	INC DPTR	Increment data pointer
		MOVX A,@DPTR	Copy the contents from memory location to acc
		ADD A,B	Add the contents of B with A
		MOV B,A	Copy contents of A to B
		JNC LBL2	Jump to LBL2 if A+B does not produce any carry
		INC R1	Increment R1 register to indicate carry
	LBL2	DJNZ R0,LBL1	Decrement R0 register and go back to LBL1 if R0 is not equal to zero
		MOV	Set Data pointer at memory

		DPTR,#4500	location 4500
		MOV A,R1	Copy the carry stored in R1 to Acc
		MOVX @DPTR,A	Copy the carry in Acc to memory
		INC DPTR	Increment Data pointer to point to the next memory location
		MOV A,B	Copy result to Acc
		MOVX @DPTR,A	Store result in memory
	STOP:	SJMP STOP	

RESULT

INFERENCE

EXPERIMENT NO 9

SQUARE AND SQUARE ROOT PROGRAM

AIM

To write an assembly language program to find square and square root of a number using 8051

PROGRAM

ADDRESS	LABEL	MNEMONICS	COMMENT
3000		MOV DPTR,#4000H	Initialize memory
		MOVX A,@DPTR	Copy number to acc
		MOV B,A	Copy number to B reg
		MOV R1,A	Copy number to R1 reg
		MUL A,B	A=A*B
		INC DPTR	Point DPTR to next memory location
		MOVX @DPTR,A	Copy Acc content to memory
		INC DPTR	Point DPTR to next memory location
		MOV A,B	Copy contents of B reg to Acc
		MOVX @DPTR,A	Copy Acc content to memory
		MOV A,R1	Copy contents of R1 reg to Acc
		MOV R0,#01	R0 reg is loaded with 01
		MOV R1,#00	R1 reg is loaded with 00
	L1	SUB A,R0	A=A-R0
		INC R1	Increment the contents of R1 reg

		INC R0	Increment the contents of R0 reg
		INC R0	Increment the contents of R0 reg
		CJNE A,#00,L1	Jump to L1 if Acc content is not 0
		INC DPTR	Point DPTR to next memory location
		MOV A,R1	Copy contents of R1 reg to Acc
		MOVX @DPTR,A	Copy Acc content to memory
	STOP	SJMP STOP	

RESULT

INFERENCE

EXPERIMENT NO 10

8 BIT MULTIPLICATION AND DIVISION

AIM

To write an assembly language program to do 8 bit multiplication and division.

I. PROGRAM TO PERFORM 8BIT MULTIPLICATION

ADDRESS	LABEL	MNEMONICS	COMMENT
3000		MOV A,#0A	Copy 0A to Acc
		MOV B,#08	Copy 88 to B reg
		MUL AB	Acc=Acc*B reg
		MOV DPTR,#4500	Initialize memory location
		MOVX @DPTR,A	Move the contents from Acc to memory location
		INC DPTR	Increment data pointer
		MOV A,B	Copy the contents of B to Acc
		MOVX @DPTR,A	Move the contents from Acc to memory location
	STOP:	SJMP STOP	

RESULT

INFERENCE

II. PROGRAM TO PERFORM 8 BIT DIVISION

ADDRESS	LABEL	MNEMONICS	COMMENT
3000		MOV A,#65	Copy 65 to Acc
		MOV B, #08	Copy 08 to B
		DIV AB	Divide Accby B
		MOV DPTR,#4500	Initialize that memory location
		MOVX @DPTR,A	Copy contents from Acc to memory location
		INC DPTR	Increment data pointer
		MOV A,B	Copy the content of B to Acc
		MOVX @DPTR,A	Copy the content of Acc to memory
	STOP:	SJMP STOP	

RESULT

INFERENCE

EXPERIMENT NO 11

INTERFACING WITH A/D AND D/A CONVERTERS

AIM

To generate sinewave using DAC in 8051.

PROBLEM ANALYSIS

8051 microcontroller can also be used for generating sinewave using digital to analog converter. In this case we provide a digital input, outputs a sinewave.

ALGORITHM

1. Initialize the control port.
2. Input the control word to assign all ports as output ports.
3. Load DPTR with an external memory location.
4. Load R1 register with number of samples to be taken.
5. Load the data to accumulator.
6. Load he first sample to register A
7. Point DPTR to the address to port C.
8. Load the content of content of reg A to port C.
9. Point DPTR to the external to the external memory location.
10. Decrement R1 as if not zero then jump to 5.
11. If zero then jump to step 4.

I PROGRAM TO CONVERT DIGITAL INPUT TO ANALOG OUTPUT

ADDRESS	LABEL	MNEMONICS	COMMENT
3000		MOV DPTR,#0FF03	Initialize control port
		MOV A.#80	Input the control word
		MOVX@DPTR,A	
		MOVDPTR,#4000	Load DPTR with an external memory location

	rep	MOV R1,#3C	Load R1 with number of samples
	nxt	MOV A,R1	Load the data to accumulator
		MOVC A, @A+DPTR	Load the first sample to reg A
		MOV DPTR,#0FF00	Point DPTR to the address of port C
		MOVX@DPTR,A	Load the content of reg A to port C
		MOV DPTR,#4000	
		DJNZ R1, nxt	Decrement R1 by 1 if not zero jump to 5
		SJMP rep	If zero then jump to 4

RESULT

NFERENCE

III. PROGRAM TO CONVERT ANALOG INPUT TO DIGITAL OUTPUT

ADDRESS	LABEL	MNEMONICS	COMMENT
3000		MOV DPTR,#0FF03H	Initialize memory
		MOV A,#98H	
		MOVX @DPTR,A	Store number to memory
		MOV DPTR,#0FF01H	Load DPTR with 0FF01H
		MOV A,#0H	Clear acc
		MOVX @DPTR,A	Copy contents of acc to memory
		MOV DPTR,#0FF02H	Load DPTR with 0FF02H
		MOV A,#0H	Clear acc
		MOVX @DPTR,A	Copy content of A to memory
		MOV A,#03H	Load 03H to acc
		MOVX @DPTR,A	Load contents of acc to memory
		MOV A,#0H	Clear acc
		MOVX @DPTR,A	Load contents of acc to memory
	DL1	MOVX A,@DPTR	Copy contents of memory to acc
		ANL A,#10H	AND A with 10H
		JZ DL1	Jump if acc is 0 to DL1
		MOV A, #04H	Load 04H to acc
		MOVX @DPTR,A	Load contents of acc to memory
		MOVX @DPTR,#0FF00H	Load contents of DPTR with 0FF00H

		MOVX A,@DPTR	Copy contents of memory to acc
		MOV R1, A	Load contents of acc to reg R1
	STOP	SJMP STOP	End the program

RESULT

INFERENCE

EXPERIMENT NO 12

WAVEFORM GENERATION USING 8051

AIM

To generate a square wave generation using 8051

PROBLEM ANALYSIS

8051 microcontroller can be used to generate square wave of desired frequency and amplitude. In that we need to do a subroutine program and in this case 0F9 is copied to register R2.

ALGORITHM

1. Complement Port 1
2. Call Delay subroutine.
3. Jump to Step 1.
4. Copy the data 0F9 to register R2.
5. Decrement the data until it becomes 0.
6. Return.

PROGRAM

ADDRESS	LABEL	MNEMONICS	COMMENT
3000	REPT	CPL 90	COMPLEMENT PORT 1
		LCALL DELAY	CALL DELAY
		SJMP REPT	JUMP TO LABEL
	DELAY	MOV R2, #0F9	COPY 0F9 TO Reg R2
	L1	DJNZ R2, L1	DECREMENT R2
		RET	

RESULT

INFERENCE