Department of Electrical Engineering

College of Engineering Trivandrum

Lab Manual

Microprocessors and Microcontrollers Lab

(2019 scheme)

Department of Electrical Engineering

College of Engineering Trivandrum

This is a controlled document of the Department of Electrical Engineering of College of Engineering Trivandrum, Thiruvananthapuram. No part of this can be reproduced in any form by any means without the prior written permission of the professor and the Head of the Department of Electrical Engineering, College of Engineering Trivandrum.

Prepared By	Verified By	Approved By
Dr. Lekshmi Mohan **Prof. Sohan Placid John**	**Prof. Vipin VA**	**HOD**

VISION

National Level Excellence and International Visibility in Every Facet of Engineering Research and Education.

MISSION

To facilitate quality transformative education in Engineering and Management.

To foster innovations in Technology and its application for meeting global challenges. To pursue and disseminate Quality Research. To equip, enrich and transform students to be responsible professionals for better service to humanity.

DEPARTMENT OF ELECTRICAL ENGINEERING

VISION

Be a centre of excellence and higher learning in Electrical Engineering and allied areas.

MISSION

To impart quality education in Electrical Engineering and bring-up professionally competent engineers.

To mould ethically sound and socially responsible Electrical Engineers with leadership qualities.

To inculcate research attitude among students and encourage them to pursue higher studies.

Program Outcomes

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 IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
PO8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	Individual and 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

Program Specific Outcomes

PSO1	Apply engineering knowledge to analyse, model, design and operate modern systems for generation, transmission, distribution and control of electrical power.
PSO2	Design, develop and test modern hardware and software systems for signal processing, measurement, instrumentation and control applications.

Course Objectives

1. Familiarize and program microprocessors and microcontrollers.
2. Hardware implementation of the embedded systems.

Course Outcomes (COs)

At the end of the course students should be able to:

Course Outcome		Bloom's Knowledge Level (KL)
CO1	Develop and execute assembly language programs for solving arithmetic and logical problems using microprocessors/ microcontrollers.	K4
CO2	Design and Implement systems with interfacing circuits for various applications.	K4
CO3	Execute projects as a team using microprocessors / microcontroller for real life applications.	K3

Note: K1-Remember, K2-Understand, K3-Apply, K4-Analyse, K5-Evaluate, K6-Create

CO-PO Mapping (Mapping of Course Outcomes with Program Outcomes)

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	3	3	2	3	2			3	2	3	3	2		3
CO2	3	3	2	3	2			3	2	3	3	2		2
CO3	3	3	2	2	2	3		3	3	3	3	2		2

1:Slight (Low), 2:Moderate(Medium), 3:Substantial (High),-:No Correlation

List of Experiments

Exp. No	Title of experiment
1.	Study of Internal Architecture of 8085 Microprocessor and Pin diagram
2.	Data Transfer using Different Addressing Modes and Block Transfer
3.	Arithmetic Operations in Binary and BCD: Addition and Subtraction
4.	Arithmetic Operations: Multiplication and Division
5.	Binary to BCD Conversion and BCD to Binary Conversion
6.	Logical Operations
7.	Digital I/O using PP1-Square Wave Generation
8.	Interfacing D/A Converter : Generation of Simple Waveforms- Triangular, Ramp
9.	Blinking Internal LED of Arduino UNO module
10.	Arduino Based Voltage Measurement
11.	Introduction to 8051 Microcontroller
12.	Data Transfer: Block Data Movement, Exchanging Data
13.	Arithmetic Operations: Addition, Subtraction, Multiplication, Division
14.	Implementation of Boolean and Logical Instructions
15.	Counters: Hexadecimal and BCD Counters

EXPERIMENT 1

STUDY OF INTERNAL ARCHITECTURE OF 8085 MICROPROCESSOR AND PIN DIAGRAM

A microprocessor is a multipurpose, programmable logic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions and provides result as output. It includes an Arithmetic / Logic unit (ALU), a control unit and an array of registers as a small internal memory for holding data while it is being manipulated or processed. It is a general-purpose device which may be used for different purposes in different applications. Configuration of the system is flexible.

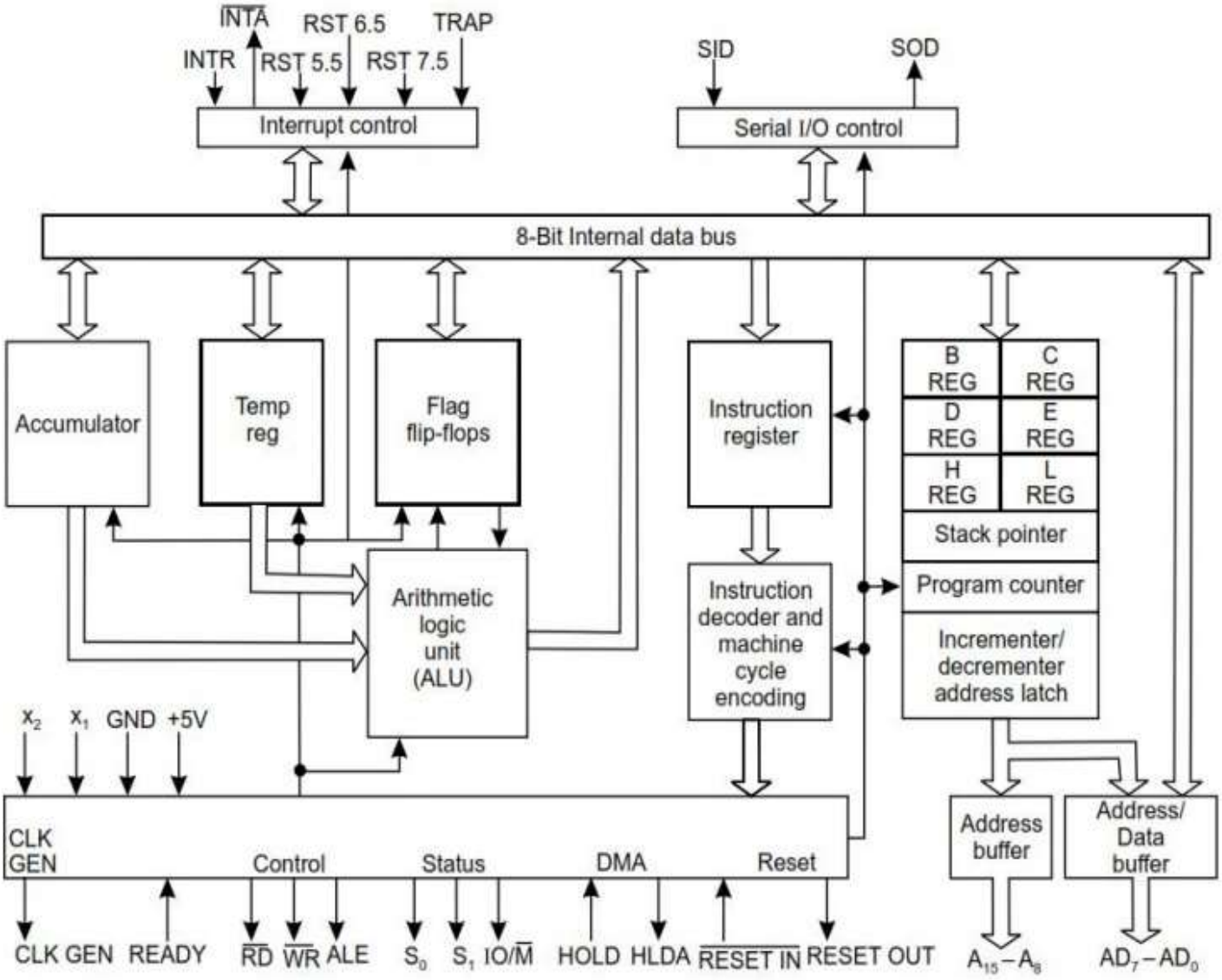


Fig 1.1 Hardware Architecture of 8085

INTERNAL ARCHITECTURE

ALU

The Arithmetic and logic unit (ALU) performs various arithmetic and logic operations like Addition, Subtraction, Logical AND, Logical OR, Logical exclusive OR, complement (Logical NOT), Increment (Add 1), Decrement (Subtract 1), Left shift (add input to itself) and clear (result is zero).

REGISTERS

Registers are small memories within the CPU. They are used by the microprocessor for temporary storage and manipulation of data and instructions. Data remain in the registers till they are sent to the memory or I/O devices.

Registers of 8085 are

- One 8-bit accumulator (ACC) ie, register A.
- Six 8-bit general purpose registers – B, C, D, E, H and L.
- One 16-bit program counter – PC.
- Instruction register – IR.
- Status register – Flag register
- One 16-bit Stack Pointer – SP.
- Temporary register – W and Z.

ACCUMULATOR

The accumulator, one of the most important 8 - bit registers of 8085, is mainly used for arithmetic, logic and rotate operations. The primary purpose of this register is to store temporary data and for the placement of final values of arithmetic and logic operations. It holds one of the operands.

GENERAL PURPOSE REGISTER

There are 6 general purpose registers in the 8085 processor, i.e. B, C, D, E, H& L. Each register can hold 8-bit data. These registers can work in pairs to hold 16-bit data and their pairing combination is like B-C, D-E & H-L. The H-L pair works as a memory pointer.

FLAG REGISTERS

The flag register is a group of flip-flops used to give the status of the result of different operations. The flag register in 8085 is an 8–bit register which contains 5 bit positions. These five flags are 1-bit F/F and are known as sign, zero, auxiliary carry, parity and carry.

CY - Carry flag, it is set when carry is generated and otherwise, it is reset.
Z – Zero flag is set if the result of an operation is zero otherwise it is reset.
S – Sign flag, Signed number is negative if S = 1 and positive if S = 0.
P – Parity flag, it is set for even parity and reset for odd parity.
AC - Auxiliary Carry flag is used for BCD operations. It is set when a carry is generated by digit D3 and passed to D4.

TEMPORARY REGISTER

There are 2 temporary registers, W and Z. It is also called operand register (8-bit). 8085 uses them internally to hold data temporarily during the execution of some instructions.

SPECIAL PURPOSE REGISTERS

It consists of three 16 bit registers – Program counter, Stack pointer, Incrementer / Decrementer Latch.

PROGRAM COUNTER

It holds the address of the next instruction to be executed to save time.

STACK POINTER

Stack is a portion of memory (RAM), that works in the LIFO concept. The stack pointer maintains the address of the last byte that is entered into the stack. Each time when the data is loaded into the stack, the Stack pointer gets decremented.

INCR/ DECR LATCH

It is used to increment or decrement the content of program counter and stack pointer register.

ADDRESS / DATA BUFFER and ADDRESS BUFFER

The contents of the stack pointer and program counter are loaded into the address buffer and address – data buffer. These buffers are then used to drive the external address bus and address–data bus. As the memory and I/O chips are connected to these buses, the CPU can exchange desired data to memory and I/O chips. The address data buffer can both send and receive data from internal data bus.

CONTROL UNIT

It performs data transfer and decision-making operations.

It consists of :

- Instruction Register
- Instruction Decoder
- Timing and Control unit

INSTRUCTION REGISTER

When an instruction like adding two data, moving a data, copying a data etc is fetched from memory, it is directed to the instruction register. So instruction registers are specifically to store the instructions that are fetched from memory.

INSTRUCTION DECODER

It decodes the information present in the instruction register for further processing. It then sends the decoded information to the timing and control unit.

TIMING AND CONTROL UNIT

It synchronizes the registers and flow of data through various registers and other units. This unit consists of an oscillator and sends control signals needed for internal and external control of data and other units. The oscillator generates clock signals.

Signals that are associated with this unit are:

- Control signals: READY, $\overline{RD}$, $\overline{WR}$, ALE
- Status signals: S0, S1, IO/$\bar{M}$
- DMA signals: HOLD, HLDA
- Reset signals: $\overline{RESETIN}$, RESET OUT

CONTROL AND STATUS SIGNALS

- RD – Read (active low) – Indicate that I/O or memory selected is to be read and data are available on the bus.

- WR – Write (active low) – Indicate that data available on the bus are to be written to memory or I/O ports.
- IO/$\bar{M}$ - Differentiate I/O operation or memory operations.
  0 – indicates a memory operation
  1 – indicates an I/O operation
- S1 and S0 – Status signals, tells current operation.

INTERRUPT CONTROLLER

Interrupt signals present in 8085 are:

1. INTR
2. TRAP
3. RST 7.5
4. RST 6.5
5. RST 5.5

Whenever the interrupt signal is enabled or requested, the microprocessor shifts the control from the main program to process the incoming request. After the request is completed, the control goes back to the main program.

SERIAL I/O CONTROL

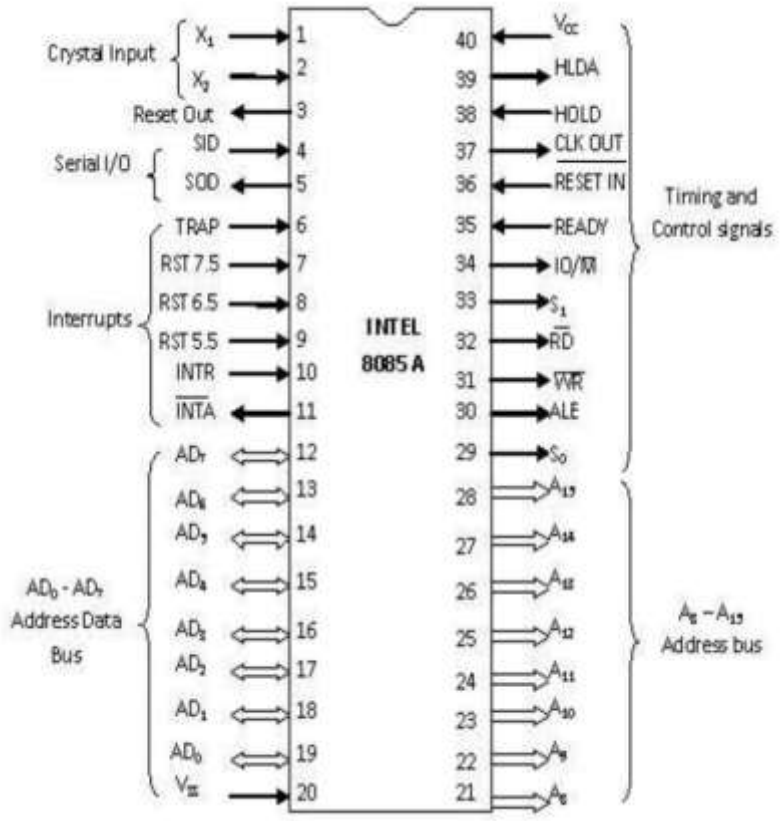
The input and output of serial data can be carried out using two instructions in 8085:

1. SID - Serial input data
2. SOD - Serial output data

Data on these line is accepted or transferred under software control by serial I/O control block, by using special instructions RIM & SIM.

8085 PIN DIAGRAM

8085 is an 8–bit, NMOS microprocessor. It is available as a 40-pin IC package fabricated on a single LSI chip. It uses a single +5V DC supply for its operation. 8085 microprocessor has a clock speed of about 3 MHz and the clock cycle is of 320ns. It has about 6500 transistors. It has 80 basic instructions and 246 opcodes. It consists of three main sections, arithmetic and logic unit, timing and control unit and several registers.



A8-A15 (Output):-
These are address bus and used for the most significant bits of memory address.

AD0-AD7 (Input/Output):-
These are time-multiplexed address data bus. These are used for the least significant 8 bits of the memory address during first clock cycle and then for data during the second and third clock cycle.

ALE (Address Latch Enable):-
It goes high during the 1st clock cycle of a machine. It enables the lower 8 bits of address to be latched either in the memory or external latch.

IO/M:-
It is status signal, when it goes high; the address on address bus is for I/O device, otherwise for memory.

S0, S1:-
These are status signals to distinguish various types of operation.

S1	S0	Operations
0	0	Halt
0	1	Write
1	0	Read
1	1	Opcode Fetch

RD (output):-
It is used to control read operation.

WR (output):-
It is used to control write operation.

HOLD (input):-
It is used to indicate that another device is requesting the use of the address & data bus.

HLDA (output):-
It is an acknowledgement signal used to indicate HOLD request has been received.

INTR (input):-
When it goes high, the microprocessor suspends its normal sequence of operations.

INTA (output):-
It is an interrupt acknowledgement signal sent by the microprocessor after INTR is received.

RST 5.5, 6.5, 7.5 and TRAP:-
These are various interrupt signals. Among them, TRAP is having highest priority.

RESET IN (input):-
It resets the PC to zero.

RESET OUT(output):-
It indicates that the CPU is being reset.

X1, X2 (input):-
This circuitry is required to produce a suitable clock for the operation of microprocessor. .

Clk (output):-
It is clock output for the user. Its frequency is the same at which the processor operates.

SID (input):-
It is used for data line for serial input.

SOD (output):-
It is used for data line for serial output.

Vcc:-
+5 volts supply.

Vss:-
Ground reference.

8085MICROPROCESSOR TRAINER KIT M85-03

M85-03 kit is a single-board Microprocessor training kit based on 8085 microprocessor. It provides monitor EPROM and user's RAM with battery backup. The kit has 28 keys hexadecimal keyboard and six digit seven segment displays for display. The kit also has the capability of interacting with a PC through an RS-232C serial link. The Input/Output structure of M85-03 provides 48 programmable I/O lines using 8255.

PROCEDURE

EXMEM(Examine memory) keyboard command is used to examine the memory locations. To examine the contents of the location for 2500 and 2501, the following key sequence has to be used.

RESET→EXMEM→2500→NEXT→2501

To enter the program

RESET→ EXMEM→ Enter Starting address of program → NEXT → Enter the machine code → NEXT

To execute the program

RESET →GO→ Starting address of program→. (Dot)(Fill Key)

To check the result

RESET→EXMEM→ Enter the address of the result location

To check the register content

Shift → EXREG → A/ B/C/D/E/H/L

EXPERIMENT 2

DATA TRANSFER USING DIFFERENT ADDRESSING MODES AND BLOCK TRANSFER

1. Write an ALP for loading registers A, B, C, D, E, H and L with single-byte data using immediate addressing and observe the register contents.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3E, 01	START:	MVI A,01H	Load A with 01
2002	06, 02		MVI B,02H	Load B with 02
2004	0E, 03		MVI C,03H	Load C with 03
2006	16, 04		MVI D,04H	Load D with 04
2008	1E, 05		MVI E,05H	Load E with 05
200A	26, 06		MVI H,06H	Load H with 06
200C	2E, 07		MVI L,07H	Load L with 07
200E	EF	END:	RST 05	Return to monitor program

2. Write an ALP for loading registers B, C, D, E, H and L with the same data using register addressing and observe the register contents.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3A, 50, 20	START:	LDA 2050H	Load accumulator with content of 2050
2003	47		MOV B, A	Move the content of A to B
2004	4F		MOV C, A	Move the content of A to C

2005	57		MOV D, A	Move the content of A to D
2006	5F		MOV E, A	Move the content of A to E
2007	67		MOV H, A	Move the content of A to H
2008	6F		MOV L, A	Move the content of A to L
2009	EF	END:	RST 05	Return to monitor program

3. Write an ALP for loading register pairs BC, DE and HL with 16-bit data using immediate addressing and observe the register pair contents.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	01, 50, 21	START:	LXI B, 2150H	Load BC register pair with data 2150
2003	11, 51, 21		LXI D, 2151H	Load DE register pair with data 2151
2006	21, 52, 21		LXI H, 2152H	Load HL register pair with data 2152
2009	EF	END:	RST 05	Return to monitor program

4. Write an ALP to copy a block of 8-bit data from 4 memory locations (2250-2253) to another 4 memory locations (2254-2257) using direct addressing.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3A, 50, 22	START:	LDA 2250H	Load data in 2250 to accumulator
2003	32, 54, 22		STA 2254H	Accumulator content stored in 2254
2006	3A, 51, 22		LDA 2251H	Load data in 2251 to accumulator
2009	32, 55, 22		STA 2255H	Accumulator data stored in 2255

200C	3A, 52, 22		LDA 2252H	Load data in 2252 to accumulator
200F	32, 56, 22		STA 2256H	Accumulator data stored in 2256
2012	3A, 53, 22		LDA 2253H	Load data in 2253 to accumulator
2015	32, 57, 22		STA 2257H	Accumulator data stored in 2257
2018	EF	END:	RST 05	Return to monitor program

5. Write an ALP to copy a block of 8-bit data from 4 memory locations (2250-2253) to another 4 memory locations (2254-2257) using 16-bit data transfer addressing mode direct addressing.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	2A, 50, 22	START:	LHLD 2250H	Data in 2250 to L register and datain 2251 to H
2003	22, 54, 22		SHLD 2254H	L register content to 2254 and H content to 2255
2006	2A, 52, 22		LHLD 2252H	Data in 2252 to L register and datain 2253 to H
2009	22, 56, 22		SHLD 2256H	L register content to 2256 and H content to 2257
200C	EF	END:	RST 05	Return to monitor program

6. Write an ALP to transfer a block of 8-bit data from 4 memory locations (2250-2253) to another 4 memory locations (2254-2257) using indirect addressing.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 50, 22	START:	LXI H, 2250H	Set up HL as a pointer of source.
2003	11, 54, 22		LXI D, 2254H	Set up DE as a pointer of destination
2006	06, 04		MVI B, 04	Set up the counter
2008	7E	LOOP:	MOV A, M	Get data from source to accumulator

2009	12		STAX D	Store data in destination
200A	23		INX H	Pointer to next source location
200B	13		INX D	Pointer to next destination location
200C	05		DCR B	Decrement counter
200D	C2, 08, 20		JNZ LOOP	If the transfer is not over, continue
2010	EF	END:	RST 05	Return to monitor program

EXPERIMENT 3

ARITHMETIC OPERATIONS IN BINARY AND BCD: ADDITION AND SUBTRACTION

1. Write an ALP to add two 8-bit numbers, sum 8 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H,2500H	Initialize memory pointer
2003	7E		MOV A, M	Load the first operand from memory to register A
2004	23		INX H	Increment content of H-L pair
2005	46		MOV B, M	Load the second operand from memory to register B
2006	80		ADD B	Add 1st and 2nd numbers
2007	23		INX H	Pointer to store the result
2008	77		MOV M, A	Store result to memory
2009	EF	END:	RST 05	Return to Monitor program

2. Write an ALP to add two 8-bit numbers, sum 16 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 01, 25	START:	LXI H, 2501H	Address of 1st number in H-L pair.
2003	7E		MOV A, M	1st number in accumulator.
2004	23		INX H	Address of 2nd number 2502 in H-L pair.
2005	46		MOV B, M	Load the second operand from memory to register B
2006	0E, 00		MVI C, 00H	MSBs of sum in register C. Initial value = 00.

2008	80		ADD B	1st number + 2nd number.
2009	D2, 0D, 20		JNC AHEAD	Is carry? No, go to the label AHEAD.
200C	0C		INR C	Yes, increment C.
200D	23	AHEAD:	INX H	Increment content of H-L pair
200E	77		MOV M, A	Move the result from A to memory.
200F	23		INX H	Increment content of H-L pair.
2010	71		MOV M, C	Move the result from C to memory.
2011	EF	END:	RST 05	Return to Monitor program

3.Write an ALP to add two16 bit numbers, sum 16 bits or more.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	2A, 01, 25	START:	LHLD 2501H	1st 16-bit number in H-L pair.
2003	EB		XCHG	Get 1st number in D-E pair.
2004	2A, 03, 25		LHLD 2503H	2nd 16-bit number in H-L pair.
2007	0E, 00		MVI C, 00H	MSBs of sum in register C. Initial value = 00.
2009	19		DAD D	1st number + 2nd number.
200A	D2, 0E, 20		JNC AHEAD	Is carry? No, go to the label AHEAD.
200D	0C		INR C	Yes, increment C.
200E	22, 05, 25	AHEAD:	SHLD 2505 H	Store LSBs of sum in 2505 and 2506 H.
2011	79		MOV A, C	MSBs of sum in accumulator
2012	32, 07,25		STA 2507H	MSBs of sum in 2507 H.

2015	EF	END:	RST 05	Return to Monitor program

4. Write an ALP to subtract two 8-bit numbers, difference 8 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H,2500H	Initialize memory pointer
2003	7E		MOV A, M	Load the first operand from memory to register A
2004	23		INX H	Increment content of H-L pair
2005	46		MOV B, M	Load the second operand from memory to register B
2006	90		SUB B	Subtract 2nd number from 1st number
2007	23		INX H	Pointer to store the result
2008	77		MOV M, A	Store result to memory
2009	EF	END:	RST 05	Return to Monitor program

5. Write an ALP for the decimal addition of two 8-bit numbers, sum 16 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 01, 25	START:	LXI H, 2501H	Address of 1st number in H-L pair.
2003	7E		MOV A, M	1st number in accumulator.
2004	23		INX H	Address of 2nd number 2502 in H-L pair.
2005	46		MOV B, M	Load the second operand from memory to register B
2006	0E, 00		MVI C, 00H	MSBs of sum in register C. Initial value = 00.
2008	80		ADD B	1st number + 2nd number.

2009	27		DAA	Decimal adjust
200A	D2, 0E, 20		JNC AHEAD	Is carry? No, go to the label AHEAD.
200D	0C		INR C	Yes, increment C.
200E	23	AHEAD:	INX H	Increment content of H-L pair
200F	77		MOV M, A	Move the result from A to memory.
2010	23		INX H	Increment content of H-L pair.
2011	71		MOV M, C	Move the result from C to memory.
2012	EF	END:	RST 05	Return to Monitor program

6. Write an ALP to add a series of 8-bit numbers, sum 16 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H, 2500H	Load the address of count to HL pair
2003	4E		MOV C, M	Load C with the count value.
2004	3E, 00		MVI A, 00H	LSBs of sum = 00 (initial value)
2006	47		MOV B, A	MSBs of sum = 00 (initial value)
2007	23	LOOP:	INX H	Point to next location.
2008	86		ADD M	Add memory content with accumulator.
2009	D2, 0D, 20		JNC AHEAD	When carry flag is 0, skip next task.
200C	04		INR B	Yes, add carry to MSBs of sum.
200D	0D	AHEAD:	DCR C	Decrement C register by 1.
200E	C2, 07, 20		JNZ LOOP	When Zero flag is not set, go to Loop.

2011	32, 50, 24		STA 2450H	Store LSBs of the sum in 2450 H.
2014	78		MOV A, B	Get MSBs of sum in accumulator.
2015	32, 51, 24		STA 2451H	Store MSBs of the sum in 2451 H.
2018	EF	END:	RST 05	Return to Monitor program

7. Write an ALP to add a series of 8-bit decimal numbers, sum 16 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H, 2500H	Load the address of count to HL pair
2003	4E		MOV C, M	Load C with the count value.
2004	3E, 00		MVI A, 00H	LSBs of sum = 00 (initial value)
2006	47		MOV B, A	MSBs of sum = 00 (initial value)
2007	23	LOOP:	INX H	Point to next location.
2008	86		ADD M	Add memory content with accumulator.
2009	27		DAA	Decimal adjust
200A	D2, 0E, 20		JNC AHEAD	When carry flag is 0, skip next task.
200D	04		INR B	Yes, add carry to MSBs of sum.
200E	0D	AHEAD:	DCR C	Decrement C register by 1.
200F	C2, 07, 20		JNZ LOOP	When Zero flag is not set, go to Loop.
2012	32, 50, 24		STA 2450H	Store LSBs of the sum in 2450 H.
2015	78		MOV A, B	Get MSBs of sum in accumulator.
2016	32, 51, 24		STA 2451H	Store MSBs of the sum in 2451 H.

2019	EF	END:	RST 05	Return to Monitor program

8. Write an ALP to shift an 8-bit number left by 1 bit.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3A, 01, 25	START:	LDA 2501H	Get data in accumulator.
2003	87		ADD A	Shift it left by one bit.
2004	32, 02, 25		STA 2502H	Store result in 2502 H
2007	EF	END:	RST 05	Return to monitor program

9. Write an ALP to shift an 8-bit number left by 2 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3A, 01, 25	START:	LDA 2501H	Get data in accumulator.
2003	87		ADD A	Shift it left by one bit.
2004	87		ADD A	Shift it left again by one bit.
2005	32, 02, 25		STA 2502H	Store result in 2502 H
2008	EF	END:	RST 05	Return to monitor program

10. Write an ALP to shift a 16-bit number left by 1 bit.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	2A, 01, 25	START:	LHLD 2501H	Get 16 bit data in HL pair.
2003	29		DAD H	Shift it left by one bit.
2004	22, 03, 25		SHLD 2503H	Store the result in 2503 and 2504 H.

2007	EF	END:	RST 05	Return to monitor program

11.Write an ALP to shift a 16-bit number left by 2 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	2A, 01, 25	START:	LHLD 2501H	Get 16 bit data in HL pair.
2003	29		DAD H	Shift it left by one bit.
2004	29		DAD H	Shift it left again by one bit.
2005	22, 03, 25		SHLD 2503H	Store the result in 2503 and 2504 H.
2008	EF	END:	RST 05	Return to monitor program

EXPERIMENT 4

ARITHMETIC OPERATIONS: MULTIPLICATION AND DIVISION

1. Write an ALP to multiply two 8-bit numbers stored at locations 2500H and 2501H and the product is stored at 2502H and 2503H.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H, 2500H	Load H-L pair with address 2500H
2003	46		MOV B, M	Get the first number in the B register
2004	23		INX H	Increment H-L pair
2005	4E		MOV C, M	Get the second number in the C register
2006	3E, 00		MVI A, 00H	Initialise accumulator with 00H
2008	16, 00		MVI D,00H	Initialise D register with 00H
200A	80	LOOP:	ADD B	Add content of Accumulator to register B.
200B	D2, 0F, 20		JNC AHEAD	Jump on no carry to AHEAD
200E	14		INR D	Increment D register if carry present
200F	0D	AHEAD:	DCR C	Decrement content of register C
2010	C2, 0A, 20		JNZ LOOP	Jump on not zero to LOOP
2013	23		INX H	Increment H-L pair
2014	77		MOV M, A	Move the result from accumulator to memory location 2502H
2015	23		INX H	Increment H-L pair
2016	72		MOV M, D	Move the carry from D register to memory location 2503H
2017	EF	END:	RST 05	Return to monitor program

2. Write an ALP to multiply a 16-bit number by an 8-bit number. Multiplicand is stored at locations 2100H and 2101H and the multiplier is in 2102H. The product is to be stored at 2103H and 2104H.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 02, 21	START:	LXI H, 2102H	Initialize memory pointer with 2102H
2003	46		MOV B, M	Load multiplier in B register
2004	11, 00, 00		LXI D, 0000H	Initialise the DE pair with 0000H
2007	2A, 00, 21		LHLD 2100H	Load multiplicand in H-L pair
200A	EB		XCHG	Exchange DE with HL pair
200B	19	BACK:	DAD D	Add DE and HL contents
200C	05		DCR B	Decrement register B
200D	C2, 0B, 20		JNZ BACK	If not zero, go to BACK
2010	22, 03, 21		SHLD 2103H	Store the product in HL pair to 2103H and 2104H
2013	EF	END:	RST 05	Return to monitor program

3. Write an ALP for binary division. The 8-bit divisor and dividend are stored at memory locations 2100H and 2101H respectively. The remainder and quotient should be stored at 2102H and 2103H respectively.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 21	START:	LXI H, 2100H	Initialize HL pair as memory pointer
2003	46		MOV B, M	Load divisor in B register
2004	23		INX H	Increment HL pair
2005	7E		MOV A, M	Load dividend to accumulator

2006	23		INX H	Increment HL pair
2007	0E, 00		MVI C, 00H	Initialize quotient as 00H
2009	B8		CMP B	Is dividend less than divisor?
200A	DA, 13, 20		JC AHEAD	If yes, jump to AHEAD
200D	90	BACK:	SUB B	Subtract divisor from dividend
200E	0C		INR C	Increment C register
200F	B8		CMP B	Is dividend less than divisor
2010	D2, 0D, 20		JNC BACK	If no carry, jump to BACK
2013	77	AHEAD:	MOV M, A	Store remainder at 2102H
2014	23		INX H	Increment HL pair
2015	71		MOV M, C	Store quotient at 2103H
2016	EF	END	RST 05	Return to monitor program

EXPERIMENT 5

BINARY TO BCD CONVERSION AND BCD TO BINARY CONVERSION

1. Write an ALP to convert BCD to Binary

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3A, 00, 25	START:	LDA 2500H	Load accumulator with content of address 2500H
2003	47		MOV B, A	Move data from accumulator to reg. B
2004	E6, F0		ANI F0H	AND F0 with accumulator content
2006	0F		RRC	Rotate accumulator content right by 1 bit
2007	0F		RRC	Rotate accumulator content right by 1 bit
2008	0F		RRC	Rotate accumulator content right by 1 bit
2009	0F		RRC	Rotate accumulator content right by 1 bit
200A	57		MOV D, A	Move data from accumulator to reg. D
200B	0E, 0A		MVI C, 0AH	Initialise C register with 0AH
200D	97		SUB A	Subtract A from A (clearing accumulator)
200E	82	BACK:	ADD D	Add D with A
200F	0D		DCR C	Decrement C register
2010	C2, 0E, 20		JNZ BACK	Jump if not zero to BACK
2013	57		MOV D, A	Move data from accumulator to reg D
2014	78		MOV A, B	Move data from reg B to accumulator
2015	E6, 0F		ANI 0FH	AND 0F with accumulator content
2017	82		ADD D	Add D with A

2018	32, 01, 25		STA 2501H	Store accumulator content in 2501H
201B	EF	END:	RST 05	Return to monitor program

2. Write an ALP to convert Binary to BCD

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	16,00	START:	MVI D, 00H	Initialise D with 00H
2002	1E,00		MVI E, 00H	Initialise E with 00H
2004	21, 00, 24		LXI H, 2400H	Load H-L pair with address 2400H
2007	7E		MOV A, M	Move data from memory to accumulator
2008	FE, 64	HUND:	CPI 64H	Compare data in accumulator with 64H
200A	DA 13, 20		JC TEN	Jump on carry to label TEN
200D	1C		INR E	Increment E register
200E	D6, 64		SUI 64H	Subtract 64H from accumulator
2010	C3, 08, 20		JMP HUND	Jump to label HUND
2013	FE, 0A	TEN:	CPI 0AH	Compare data in accumulator with 0AH
2015	DA, 1E, 20		JC UNIT	Jump if carry to label UNIT
2018	14		INR D	Increment D register
2019	D6, 0A		SUI 0AH	Subtract 0AH from accumulator
201B	C3, 13, 20		JMP TEN	Jump to label TEN
201E	23	UNIT:	INX H	Increment H-L pair
201F	73		MOV M,E	Move data from reg. E to memory

2020	4F		MOV C,A	Move data from accumulator to reg. C
2021	7A		MOV A,D	Move data from reg. D to accumulator
2022	07		RLC	Rotate accumulator content left by 1 bit
2023	07		RLC	Rotate accumulator content left by 1 bit
2024	07		RLC	Rotate accumulator content left by 1 bit
2025	07		RLC	Rotate accumulator content left by 1 bit
2026	81		ADD C	Add C with A
2027	23		INX H	Increment H-L pair
2028	77		MOV M, A	Move data from accumulator to memory
2029	EF	END:	RST 05	Return to monitor program

EXPERIMENT 6

LOGICAL OPERATIONS

1. Write an ALP to find the larger of two numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 01, 25	START:	LXI H, 2501H	Address of 1st number in H-L pair.
2003	7E		MOV A, M	1st number in accumulator.
2004	23		INX H	Address of 2nd number in H-L pair.
2005	BE		CMP M	Compare 2nd number with 1st number. Is the 2nd number >1st ?
2006	D2, 0A, 20		JNC AHEAD	No, larger number is in accumulator. Go to AHEAD.
2009	7E		MOV A, M	Yes, get 2nd number in accumulator.
200A	32, 03, 25	AHEAD:	STA 2503H	Store larger number in 2503H.
200D	EF	END:	RST 05	Return to monitor program

2. Write an ALP to find the smaller of two numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 01, 25	START:	LXI H, 2501H	Address of 1st number in H-L pair.
2003	7E		MOV A, M	1st number in accumulator.
2004	23		INX H	Address of 2nd number in H-L pair.
2005	BE		CMP M	Compare 2nd number with 1st number. Is the 2nd number >1st ?
2006	DA, 0A, 20		JC AHEAD	Yes, smaller number is in accumulator. Go to AHEAD.

2009	7E		MOV A, M	No, get 2nd number in accumulator.
200A	32, 04, 25	AHEAD:	STA 2504H	Store smaller number in 2504H.
200D	EF	END:	RST 05	Return to monitor program

3. Write an ALP to find the largest number in an array of 8-bit numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H, 2500H	Address for count in H-L pair.
2003	4E		MOV C, M	Count in register C.
2004	23		INX H	Address of 1st number in H-L pair.
2005	7E		MOV A, M	1st number in accumulator.
2006	0D		DCR C	Decrement count.
2007	23	LOOP:	INX H	Address of next number.
2008	BE		CMP M	Compare next no. with previous maximum. Is next no. > previous?
2009	D2, 0D, 20		JNC AHEAD	No, larger number is in accumulator. Go to the label AHEAD.
200C	7E		MOV A, M	Yes, get larger number in accumulator.
200D	0D	AHEAD:	DCR C	Decrement count.
200E	C2, 07, 20		JNZ LOOP	Jump if not zero.
2011	32, 50, 24		STA 2450H	Store result in 2450H.
2014	EF	END:	RST 05	Return to monitor program

4. Write an ALP to find the smallest number in an array of 8-bit numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H, 2500H	Address for count in H-L pair.
2003	4E		MOV C, M	Count in register C.
2004	23		INX H	Address of 1st number in H-L pair.
2005	7E		MOV A, M	1st number in accumulator.
2006	0D		DCR C	Decrement count.
2007	23	LOOP:	INX H	Address of next number.
2008	BE		CMP M	Compare next no. with previous maximum. Is next no. > previous?
2009	DA, 0D, 20		JC AHEAD	Yes, smaller number is in accumulator. Go to the label AHEAD.
200C	7E		MOV A, M	No, get smaller number in accumulator.
200D	0D	AHEAD:	DCR C	Decrement count.
200E	C2, 07, 20		JNZ LOOP	Jump if not zero.
2011	32, 51, 24		STA 2451H	Store result in 2451H.
2014	EF	END:	RST 05	Return to monitor program

5. Write an ALP to sort an array of 8-bit numbers in the descending order.

MEMORY ADDRESS	MACHINE CODES	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 26	START:	LXI H, 2600H	Set pointer for array.

2003	4E		MOV C, M	Load the Count.
2004	0D		DCR C	Decrement Count.
2005	51	REPEAT:	MOV D, C	Count the number of Comparisons in register D.
2006	21, 01, 26		LXI H, 2601H	Load starting address of data array.
2009	7E	LOOP:	MOV A, M	Copy content of memory location to Accumulator.
200A	23		INX H	Increment content of HL pair
200B	BE		CMP M	Compare the number with next number.
200C	D2, 14, 20		JNC SKIP	Jump to skip if carry not generated.
200F	46		MOV B, M	Copy content of memory location to B Register.
2010	77		MOV M, A	Copy content of Accumulator to memory location.
2011	2B		DCX H	Decrement content of HL pair
2012	70		MOV M, B	Copy content of B Register to memory location.
2013	23		INX H	Increment content of HL pair
2014	15	SKIP:	DCR D	Decrement D register
2015	C2, 09, 20		JNZ LOOP	Jump to LOOP if not Zero.
2018	0D		DCR C	Decrement C register
2019	C2, 05, 20		JNZ REPEAT	Jump to REPEAT if not Zero.
201C	EF	END:	RST 05	Return to monitor program

6. Write an ALP to sort an array of 8-bit numbers in the ascending order.

MEMORY ADDRESS	MACHINE CODES	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 26	START:	LXI H, 2600H	Set pointer for array.
2003	4E		MOV C, M	Load the Count.
2004	0D		DCR C	Decrement Count.
2005	51	REPEAT:	MOV D, C	Count the number of Comparisons in register D.
2006	21, 01, 26		LXI H, 2601H	Load starting address of data array.
2009	7E	LOOP:	MOV A, M	Copy content of memory location to Accumulator.
200A	23		INX H	Increment content of HL pair
200B	BE		CMP M	Compare the number with next number.
200C	DA, 14, 20		JC SKIP	Jump to skip if carry generated.
200F	46		MOV B, M	Copy content of memory location to B Register.
2010	77		MOV M, A	Copy content of Accumulator to memory location.
2011	2B		DCX H	Decrement content of HL pair
2012	70		MOV M, B	Copy content of B Register to memory location.
2013	23		INX H	Increment content of HL pair
2014	15	SKIP:	DCR D	Decrement D register
2015	C2, 09, 20		JNZ LOOP	Jump to LOOP if not Zero.
2018	0D		DCR C	Decrement C register

2019	C2, 05, 20		JNZ REPEAT	Jump to REPEAT if not Zero.
201C	EF	END:	RST 05	Return to monitor program

EXPERIMENT 7

DIGITAL I/O USING PPI- SQUARE WAVE GENERATION

Aim

To generate a pulse train of frequency 200 Hz and duty cycle 50%.

Theory

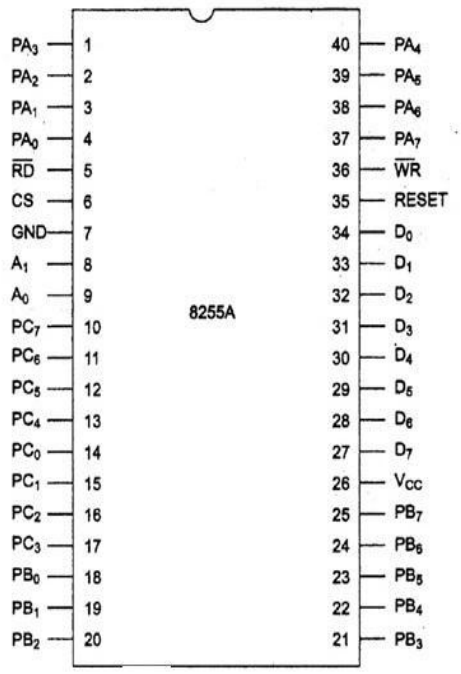
Waveform generation using microprocessor requires input-output ports interfaced to it. Programmable peripheral interface (PPI) 8255 is a general purpose programmable I/O device designed to interface the CPU with its outside world such as ADC, DAC, keyboard etc. IC 8255 provides 3 nos. of 8-bit ports (Port A, Port B and Port C). IC 8255 needs to be initialized before use. Initialization includes setting mode of 8255 (Input-Output or Bit Set Reset) and data direction in case of IO mode (Input or Output) for individual ports. In this case, 8255 is setup in I/O mode with all ports as output ports. So the initialization control word is 80H.

The data send out from microprocessor to 8255 may be directed to Port A, B, C or Control Word Register (CWR). Each register mentioned above is given an 8-bit address.

I/O address range

Port A = 00H; Port B = 01H; Port C = 02H; Control word Register CWR = 03H

To initialize the 8255, load control word (80H) in Accumulator and send it to CWR (03H)



8255-I CONNECTOR-CN4			
PIN	**SIGNALS**	**PIN**	**SIGNALS**
1	**P1C4**	14	**P1B1**
2	**P1C5**	15	**P1A6**
3	**P1C2**	16	**P1A7**
4	**P1C3**	17	**P1A4**
5	**P1C0**	18	**P1A5**
6	**P1C1**	19	**P1A2**
7	**P1B6**	20	**P1A3**
8	**P1B7**	21	**P1A0**
9	**P1B4**	22	**P1A1**
10	**P1B5**	23	**P1C6**
11	**P1B2**	24	**P1C7**
12	**P1B3**	25	**GND**
13	**P1B0**	26	**VCC**

- **PA0 – PA7 –** Pins of port A
- **PB0 – PB7 –** Pins of port B
- **PC0 – PC7 –** Pins of port C
- **D0 – D7 –** Data pins for the transfer of data
- **RESET –** Reset input
- **RD' –** Read input
- **WR' –** Write input
- **CS' –** Chip select
- **A1 and A0 –** Address pins

Algorithm- Square Wave Waveform

Delay Calculation

The delay time required for frequency of 200 Hz is 2500μs for low and high states. Time delay subroutines load a value in a register or register-pair and decrement it. When the value equals 0, it returns.
The statements from DCX D to JNZ REP is repeated N times (where N is loaded in DE register-pair). LXI D, N and RET are executed only once.

Total T states in time delay = 24N +17
System frequency = 3.072 MHz
Thus, 1 T state = 0.3255μs
Time delay, td = (24N + 17) x 0.3255μs

For a delay of 2500 μs,

$2500x10^{-6} = (24xN + 17) \times 0.3255x10^{-6}$

$N = 319_{10} = 013FH$

Algorithm:

1. Start
2. Set control word (Port A as output port)
3. Set port A low
4. Call delay
5. Set port A high
6. Call delay
7. Go to Step 3

Program

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3E, 80		MVI A, 80H	Load A with immediate data 80H
2002	D3, 03		OUT 03H	Send content of A to CWR
2004	3E, 00	LOOP:	MVI A, 00H	Load A with immediate data 00H
2006	D3, 00		OUT 00H	Send content of Acc to output port A
2008	CD, 15, 20		CALL DELAY	Call the delay subroutine
200B	3E, FF		MVI A, FFH	Load A with immediate data FFH
200D	D3, 00		OUT 00H	Send content of Acc to output port A
200F	CD, 15, 20		CALL DELAY	Call the delay subroutine
2012	C3, 04, 20		JMP LOOP	Jump to LOOP to repeat
DELAY SUBROUTINE				
2015	11, 3F, 01	DELAY:	LXI D, 013FH	Load DE register pair with value of N
2018	1B	REP:	DCX D	Decrement D
2019	7A		MOV A, D	Move content of D to A
201A	B3		ORA E	OR the value of E with A and store the result in A
201B	C2,18, 20		JNZ REP	Jump on non zero to REP
201E	C9		RET	Return to main program

Procedure

Enter the program for square wave generation from memory location 2000H onwards. Execute the program and observe the waveform available at the pins of port A. (Connect the probe to the corresponding Port A signal pins of CN4 connector)

Result

EXPERIMENT 8

INTERFACING D/A CONVERTER : GENERATION OF SIMPLE WAVEFORMS-TRIANGULAR, RAMP

i. Triangular Waveform

Aim

To generate a triangular wave of suitable amplitude using DAC interface.

Algorithm

1. To initialize 8255, load control word (80H) in Accumulator and sent it to CWR (03H)
2. Clear Accumulator
3. Send Accumulator content to output port A (00H)
4. Increment Accumulator data
5. If Accumulator content not equal to FFH, go to Step 3
6. Out Accumulator content to Port A (00H)
7. Decrement Accumulator
8. If Accumulator content not equal to 00H, go to Step 6
9. Go to Step 3

Program

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3E, 80		MVI A, 80H	Load Acc with immediate data 80H
2002	D3, 03		OUT 03H	Send content of Acc to CWR
2004	AF		XRA A	Clear the accumulator
2005	D3, 00	LOOP1:	OUT 00H	Output the contents of Acc to the output port A
2007	3C		INR A	Increment Acc
2008	FE, FF		CPI FFH	Compare the content of Accumulator with maximum
200A	C2, 05, 20		JNZ LOOP1	Jump to LOOP1 if the result of comparison is not equal to zero

200D	D3, 00	LOOP2:	OUT 00H	Output the value in acc at the port A
200F	3D		DCR A	Decrement Acc
2010	C2, 0D, 20		JNZ LOOP2	Jump to LOOP2 if result of comparison is not equal to zero
2013	C3, 05, 20		JMP LOOP1	Jump to LOOP1 to repeat the process

ii. Ramp (Sawtooth) Waveform

Aim

To generate a sawtooth waveform of suitable amplitude using DAC interface.

Algorithm:

1. To initialize 8255, load control word (80H) in Accumulator and sent it to CWR (03H)
2. Clear Accumulator
3. Out Accumulator content to Port A
4. Increment Accumulator
5. Go to Step 3

(Since Accumulator is an 8-bit register, incrementing from FFH results in 00H)

Program

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3E, 80		MVI A, 80H	Load A with immediate data 80H
2002	D3, 03		OUT 03H	Send the contents of A to output port
2004	AF		XRA A	EXOR the value of A with A itself. This resets/clear
2005	D3, 00	LOOP:	OUT 00H	Output the content of A to port
2007	3C		INR A	Increment A
2008	C3, 05, 20		JMP LOOP	Jump to LOOP

Procedure

Enter the program from memory location 2000H onwards. Connect the CN4 pins of 8255 with the DAC module. Execute the program and observe the output between X- Out and GND pins of the DAC.

Result

The following waveforms were generated using 8085.

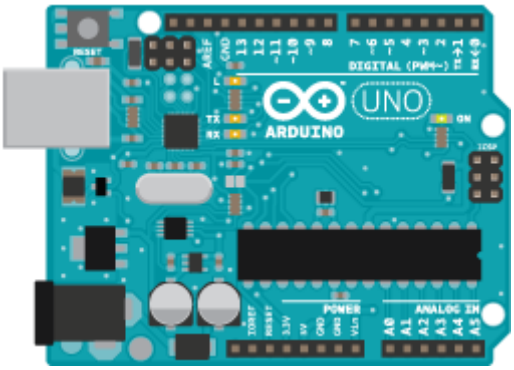
Triangular waveform

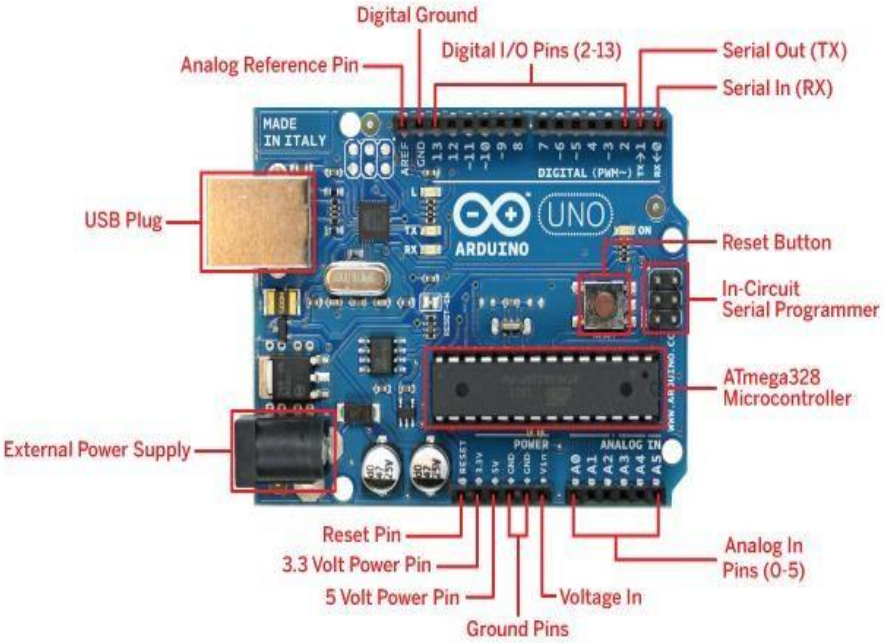
Ramp waveform

Arduino UNO module

Arduino is an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board. The name "Arduino" is a copyright held by the original team based in Italy that originally built the hardware, the IDE (integrated development environment) and the software libraries. Arduino development environment can be run on either Windows, Linux and MacOS for no cost other than for the hardware. The software is freely downloadable in one bundle from www.arduino.cc, the website that is ground-zero for all-things-Arduino.

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators.





The Arduino board comes with a single LED, often called the *Pin 13 LED* because it is electrically connected to Digital Pin 13. This LED is the board's only built-in indicator accessible to programs.

EXPERIMENT 9

BLINKING INTERNAL LED OF ARDUINO UNO MODULE

Aim

To blink internal LED of Arduino UNO

Procedure

In the menu of the Arduino IDE you can choose:

File ▸ Examples ▸ 01. Basics ▸ Blink

The IDE will open the code to blink the builtin LED automatically.

Uploading code to the Arduino

Now our program is ready to upload to the Arduino. First we have to connect our Arduino to the computer with the USB cable. Make sure you've selected the correct board in the IDE:

Tools ▸ Board ▸ Arduino/Genuino UNO

and the correct port:

Tools ▸ Port

If you are not sure which port to use, try them all until you can successfully upload your code. Then verify your code for possible errors. The IDE only checks if it can read your code. It does not check if you have written correct code for what you are trying to program.

If everything works, the IDE shows the Compiling completed message. You can now upload your code by pressing the upload button. The uploading is complete when the messages appears. Your program will immediately start after uploading. As a result you should now see your Arduino LED blink with 1000ms intervals.

Program

```
void setup()
  {
      // initialize digital pin LED_BUILTIN as an output.
       pinMode(LED_BUILTIN, OUTPUT);
   }
   // the loop function runs over and over again forever
void loop()
   {
     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
     delay(1000); // wait for a second
     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
```

```
    delay(1000); // wait for a second
}
```

Result

BLINKING EXTERNAL LED USING ARDUINO UNO MODULE

Aim

To blink an externally connected LED using Arduino UNO

Program

```
int LED = 8;
void setup()
    {
      // initialize digital pin LED as an output.
      pinMode(LED, OUTPUT);
     }
    // the loop function runs over and over again forever
void loop()
     {
      digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)
      delay(1000); // wait for a second
      digitalWrite(LED, LOW); // turn the LED off by making the voltage LOW
      delay(1000); // wait for a second
}
```

Result

EXPERIMENT 10

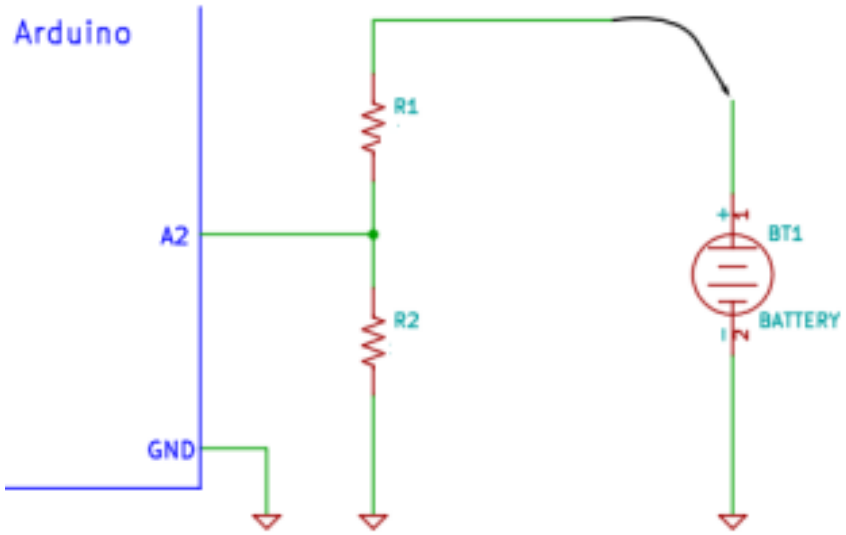
ARDUINO BASED VOLTAGE MEASUREMENT

Aim

To measure a DC voltage in range 0-9 V using Arduino UNO

Theory

A voltage divider circuit consisting of two resistors in series will divide the input voltage to bring it within the range of the Arduino analog inputs.



Design

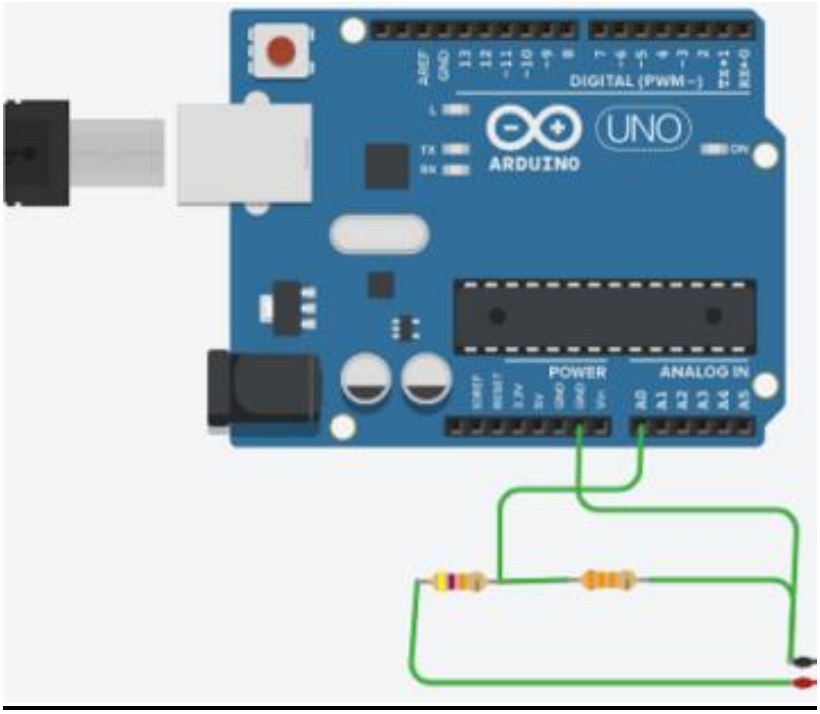
(Design the suitable values of R1 & R2, such that maximum voltage across R2 will be 5V, when actual maximum input voltage is applied across series combination, in this case, 9V)

Program

```
int value = 0;

float voltage;

float R1 = 100.0;

float R2 = 330.0;

void setup()

    {
```

```
    pinMode(A0, INPUT);
    Serial.begin(9600);
   }
void loop()
  {
   value = analogRead(A0);
   voltage = value * (5.0/1024)*((R1 + R2)/R2);
   Serial.print("Voltage =");
   Serial.println(voltage);
   delay(500);
}
```

Circuit Diagram



Result

Experiment 11

INTRODUCTION TO 8051 MICROCONTROLLER

Microcontroller is a programmable logic device that has computing and decision-making capability similar to that of a CPU of a computer.

The Microcontroller communicates and operates in the binary numbers 0 and 1 called bits. Each Microcontroller has a fixed set of instruction in the form of binary patterns called machine language. However, it is difficult for human to communicate in the language of 0s and 1s. Therefore, the binary instructions given abbreviated names called mnemonics, which form the assembly language for given microcontroller. An assembler is used to convert assembly language to machine language. For example, if we have to add two numbers in A and B. we can use the instruction ADD A, B. This add instruction is an example of mnemonics. Its machine language form will be 58, 65. This 58, 65 can be obtained from microcontroller manual. 58 in hexadecimal represents the machine language instruction for ADD and 65 represents A, B.

Each microcontroller recognizes and process a group of bits called the word and microcontrollers are classified according to their word length. For example, a controller with an 8-bit word is known as an 8-bit microcontroller and a controller with 32-bit word is known as a 32 bit microcontroller.

Organization of a Microcontroller Based system

CPU	RAM	ROM
I/O	Timer	Serial Comport

Micro controller is a self-contained system or self-sufficient system having CPU, internal RAM, internal ROM, Timers and counters, I/O ports, serial com port.

Micro controller is a specific purpose digital controller that is meant to read data, perform limited calculations on that data and control its environment based on those calculations

APPLICATIONS

1. Measuring instruments such as the oscilloscope, multimeter and the spectrum analyzer
2. Music related equipment such as synthesizers
3. House hold items, such as the microwave oven, doorbell, washing machine and television.
4. Defence equipment such as fighter planes, missiles and radar.
5. Medical equipment such as blood pressure monitors, blood analyzers and monitoring system

ARCHITECTURE

The accumulator register 'A':- The most important data register is the A register which acts as the accumulator. It is a mandatory that the A register carry one of the operands for all arithmetic instructions. The other operand may be in memory (RAM) or in any other register.

Register B:- The register B is not a frequently used register, because it can be used as an operand only for some specific operations like multiplication of two numbers, one operand should be in A , and the other should be B. Same is the case for division. But it can store data.

Internal RAM:- Totally, the 8051 has 256 bytes of RAM, but half of it is reserved to act as the "special function registers", that is , the registers which are used to handle the activities of the peripherals of the device. The remaining 128 bytes is what is referred to as internal RAM, and is divided into parts. The first 32 bytes act as register banks 0 to 3; each bank contains 8 data registers named R0 to R7. These registers are used for data manipulations and data movement. At a time, only one of these banks is operational. It is possible to switch from the current bank to another bank by using two bits of the PSW. By default, it is bank 0 that is the current bank. RAM locations from 0 to 7 are set aside for bank 0 ,where R0 is RAM location 0, R1 is RAM location1, R2 is location 2, and so on, until memory location

7, which belongs to R7 of bank 0. The second bank of registers R0- R7 starts at RAM location 08H and goes to location of 0F H. The third bank of R0-R7 starts at memory location 10H and goes to location 17H. Finally RAM locations 18H to IFH are set aside for the fourth bank of R0-R7.

Bank 1 uses the same RAM as the stack. A total of 16 bytes from locations 20 H to 2 FH are set aside for bit addressable read/write memory. A total of 80 bytes from locations 30 H to 7FH are used for read and write storage or what is normally called a scratch pad. These 80 locations of RAM are widely used for the purpose of storing data and parameters by 8051 programmers

Default register bank – Bank O

How to switch register banks? Register bank O is the default when the 8051 is powered up. We can switch to other banks by use of the PSW (program status word) register. Bits D4 and D3 of the PSW are used to select the desired register bank as shown in Table.

	RS1 (PSW.4)	RSO (PSW.3)
Bank 0	0	0
Bank 1	0	1
Bank 2	1	0
Bank 3	1	1

The D3 and D4 bits of register program status word(PSW) are often referred to as PSW.4 and PSW.3 since they can be accessed by the bit addressable instructions SETB and CLR. For example, "SETB PSB.3" will make PSW.3 = 1 and select bank register 1.

Stack in the 8051:- The stack is a section of RAM used by the CPU to store information temporarily. This information could be data or address. The CPU needs this storage area since there are only a limited number of registers.

How stacks are accessed in the 8051 :- The register used to access the stack is called the SP (stack pointer) register. The stack pointer in the 8051 is only 8 bits wide, which means that RAM location 08 is the first location used the stack by the 8051. The storing of a CPU register in the stack is called a PUSH, and pulling the contents off the stack back into a CPU register is called a pop. In other words, a register is pushed onto the stack to save it and popped off the stack to retrieve it.

Pushing onto the stack: - In the 8051 the stack pointer (SP) points to the last location of the stack. As we push data onto the stack, the stack pointer (SP) is incremented by one. For every byte of data saved on the stack, SP is incremented only once.

Popping from the stack:- Popping the content of the stack back into a given register is the opposite process of pushing .With every pop, the top byte of the stack is copied to the register specified by the instructions and the stack pointer is decremented once.

The upper limit of the stack: Locations 08 to 0F in the 8051 RAM can be used for the stack. This is because locations 20- 2FH of RAM are reserved for bit addressable memory and must not be used by the stack. If in a given program we need more area, we can change the SP to point to RAM locations 30-7FH. This is done with the instruction "MOV SP, #XX".

CALL instruction and the stack: In addition using the stack to save registers, the CPU also used the stack to save the address of the instruction just below the CALL instruction. This is how the CPU knows where to resume when it returns from the called subroutine

PSW (program status word) register:- The PSW register is an 8-bit register. It is also referred to as the flag register. Although the PSW register is 8 bits wide, only 6 bits of it are used by the 8051. The two unused bits are user-definable flags. Four of the Flags are called conditional flags, meaning that they indicate some conditions that result after am instruction being executed. These four are CY (carry) AC (auxiliary carry) P (parity) and OV (over flow). The bits PSW.3 and PSW.4 are designated as RSO

and RSI, respectively and are used to change the bank registers. The PSW.5 and PSW.1 bits are general – purpose status flag bits and can be used by the programmer for any purpose

CY	AC	F0	RS1	RS0	OV	-	P

CY PSW.7 carry flag
AC PSW.6 Auxiliary carryflag
FO PSW.5 Available to the user for general purpose
RS1 PSW.4 Register Bank selector bit 1
RS0 PSW. 3 Register Bank selector bit 0
OV PSW.1 user definable bit
P PSW.0 parity flag

RS1	RS0	Register Bank
0	0	0
0	1	1
1	0	2
1	1	3

CY the carry Flag: - this flag is set whenever there is a carry out from the D7 bit. This flag bit is affected after an 8-bit addition or subtraction. It can also be set to 1 or 0 directly by an instruction such as "SETB C" and CLR C" where "SETB C" stands for "set bit carry" and "CLRC" for "clear carry"
Eg.:- MOV A, #9CH
ADD A, # 64 H
CY=1

AC, the auxiliary carry flag
If there is a carry from D3 to D4 during an ADD or SUB operation, this bit is set; otherwise, it is cleared. This flag is used by instructions that perform BCD arithmetic
Eg. MOV A, #9CH
ADDA, # 64 H'
AC=1

P, the parity flag
The parity flag reflects the number of 1s in the accumulator register only. If the A register contains an odd number of Is, then p=1. Therefor, p= 0 if A has an even number of 1s
Eg. MOV A, #9CH
ADD A, # 64H
P=0

OV the overflow flag
This flag is set whenever the result of a signed number operation is too large causing the high – order bit to overflow into the sign bit. In general, the carry flag is used to detect errors in unsigned arithmetic operations. The overflow flag is only used to detect errors in signed arithmetic operations.

ROM
ROM can be 4k on chip and 60k external ROM or 64k external

Addressing modes

The CPU can access data in various ways. The data could be in a register, or in memory, or be provided as an immediate value. These various ways of accessing data are called addressing modes. The various addressing modes of a microprocessor are determined when it is designed, and therefore cannot be changed by the programmer. The 8051 provides a total of five distinct addressing modes. They are as follows.

1. Immediate
2. Register
3. Direct
4. Register Indirect
5. Indexed

1.**Immediate, addressing mode**:- In this addressing mode, the source operand is a constant. In immediate addressing mode, as the name implies, when the instruction is assembled, the operand comes immediately after the opcode. The immediate data must be preceded by the pound sign, "#" This addressing mode can be used to load information into any of the registers including the DPTR register. Examples follows

```
MOV A, #25H          ;load 25H into A
MOV R4, #62          ;    load 62 into R4
MOV DPTR, #4521      ;    DPTR = 4521
```

2.**Register addressing mode** : Register addressing mode involves the use of registers to hold the data to be manipulated.

Eg : MOVA, R0; copy the contents of R0 into A.

The source and destination registers must match in size. In other words, coding "MOV DPTR, A" will give an error, since the source is an 8 bit register and the destination C5 a 16 bit register.

We can move data between the accumulator and Rn (n = 0 to 7) but movement of data between Rn register is not allowed. For example, the instruction "MOV R4, R7" is invalid.

3.**Direct addressing modes** : There are 128 bytes of RAM in the 8051. The RAM has been assigned addresses 00 to 7FH

1. RAM locations 00-1FH are assigned to the register banks and stack.
2. RAM locations 20-2FH are set aside as bit addressable space to save single bit data.
3. RAM locations 30-7FH is available as place to save byte sized data.

Although the entire 128 bytes of RAM can be accessed using direct addressing mode, it is most often used to access RAM locations 30-7FH. This is due to the Fact that register bank locations are accessed by the register names R0-R7, but there is no such name for other RAM locations. In the direct addressing mode the data is in RAM memory locations whose address is known, and this address is given as a part of the instruction. Contrast this with immediate addressing mode, in which the operand itself is provided with the instruction. The "#" sign distinguishes between the two modes.

MOV R0, 40H; save content of RAM location 40H in R0 RAM locations. These registers can be accessed in two ways

```
MOV A, 4     ; is same as
MOV A, R4    ; which means copy R4 into A
```

4.**Register indirect addressing mode**

In the register indirect addressing mode, a register is used as pointer to the data. Register R0 and

R1 are used for this purpose. In other words R2-R7 cannot be used to hold the address of an operand located in RAM when using this addressing mode when R0 and R1 are used as pointers, that is, when they hold the addresses of RAM locations, they must be preceded by the "@" sign, as show below MOV A, @R0; move contents of RAM location whose address is held by R0 into A.

```
MOV @ R1, B        ;    move contents of B into RAM locations
                        whose address is held by R1.
```

Adv : - one of the advantages of register indirect addressing mode is that it makes accessing data dynamic rather than static as in the case of direct addressing mode. Looping is not possible in direct addressing mode. This is the main difference between the direct and register indirect addressing modes.

5.**Indexed addressing modes** is widely used in accessing data elements of look-up table entries located in the program ROM space of the 8051. The instruction used for this purpose is "MOVC A, @ A+DPTR". The 16-bit register DPTR and register A are used to form the address of the data element stores in on-chip ROM. Because the data elements are stored in the program (code) space ROM of the 8051, the instruction MOVC is used instead of MOV. The "c" means code. In this instruction the contents of A are added to the 16bit register DPTR to form the 16 bit address of the needed data.

PORTS

For input output operations, 8051 has 4 ports.

PORT 0

Port 0 provides both address and data. The 8051 multiplexes address and data through port 0 to save pins. When ALE=0, it provides data D0-D7, but when ALE = 1 it has address A0-A7. Therefore, ALE is used for de multiplexing address and data with the help of a 74L5373 latch..

PORT1 and PORT2

In 8051 based systems with no external memory connection, both P1 and P2 are used as simple Input –Output. However, in 8031/8051 based systems with external memory connections, port 2 must be used along with P0 to provide the 16-bit address for the external memory

PORT3

Occupies a total of 8 pins. It can be be used as input or output. Although port is configured as an input port upon reset, this is not the way it is most commonly used. Ports has the additional function of providing some extremely important signals such as interrupts.

P3 bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	INTO	12
P3.3	INTI	13
P3.4	T0	14
P3.5	T1	15
P3.6	WR	16
P3.7	RD	17

P3.1 are used for the RXD and TXD serial communications signals. Bits P3.2 and P3.3 are set aside

for external interrupts. Bits P3.4 and P3.5 are used for Timers 0 and 1. P3.6 and P3.7 are used to provide the WR and RD signals of external memory connections.

Experiment 11.A
STUDY OF 8051 MICROCONTROLLER TRAINER KIT

AIM:

To familiarize 8051microcontroller kit and execute simple programs

HARDWARE SPECIFICATIONS OF THE 8051 MICROCONTROLLER:-

Make: KITEK

i. Processor, Clock Frequency

Intel 8051/89C51 at 12MHz (89C51 max upto 33MHz).

ii. Memory

System EPROM	:	0000 - 3FFFH & C000 - FFFFH
System RAM	:	4000 - BFFFH
Additional RAM	:	0000 - 3FFFH & C000 - FEFFH
Monitor Buffer	:	4000 - 40FFH
User Program / Data RAM area	:	4100 - BFFFH
User Data RAM area	:	0000 - 3FFFH & C000 - FEFFH
Memory mapped I/O	:	FF00 - FF1FH, FFC0 - FFFFH
Memory mapped I/O expansion	:	FF20 – FFBFH

Note: The RAM area is from 4000 - 40FF should not be accessed by the user since it is used by the monitor program

iii. Input / Output

Parallel	:	24 I/O lines using two numbers of 8255.
Serial	:	1 Number of RS232 Serial Interface using 8051 Serial Port.
Timer	:	8051 has two 16 bit Timer namely Timer 0 and Timer 1 89C51 has 3 16 bit Timer / Counter.
Printer	:	One Centronics Compatible Printer interface through 8255-I port.
Interrupt	:	8051 provides 5 interrupt sources. Among them two are external interrupts called INT0 and INT1.

iv. Display (Optional)

6 Digit, 0.3", 7-segment Red LED display with filter.

4 Digits for address display.

2 Digits for data display

Based on 8279 - keyboard and display controller.

.

v. LCD Interface

16 × 2 alpha numeric LCD display Module

vi. IBM PC Keyboard Interface

vii. Keyboard : 101 ASCII keyboard

viii. Onboard Battery Backup (Optional)

Onboard Battery backup facility is provided for 64kb RAM 4000 - BFFF.

ix. System Power Consumption

+ 5V : 3 Amp

+12V : 200mA

-12V : 100mA

+30V : 300mA

PARALLEL INTERFACE DETAILS

Intel 8255(Programmable Peripheral Interface) is used for parallel interface. I/O system mapping is used. Memory mapped I/O addresses are given below

Activerangeportaddress	Portnumbers	Selecteddevice
(FF0C–FF0F)	8255 – I PORT FF0C APORT FF0D BPORT FF0E C PORT FF0F CONTROL WORD	PROGRAMMABLE PERIPHERAL INTERFACE

PROCEDURE

Procedure for entering the program is given below.

1. For writing program select line assembler by simply pressing the key 'A' and Enter key. After the command the LCD screen will prompt you to enter the "Start Address"
2. Enter the starting address and press the enter key.
3. Now enter the instruction in assembly language one by one. After each instruction press enter key.
4. To examine or imply modify data in a memory location, type "SD" followed by the memory location, and press enter.
5. To un assemble, use "U" in the home menu and press enter
6. To execute press, type 'GO' followed by the starting address location of the program and press enter key

Further Commands available are:

1. Substitute Memory Command
Syntax:
#SD <Addr><CR> - for Data Memory

2. Register View / Modify
#R <CR>

3. GO Command
GO <Addr><CR>

4. Internal Ram Access command
#IR <Addr><CR>
To view that bytes in the internal RAM location (00 - 7F).

5. Assemble
#A <CR>

6. Unassemble
#U <CR>

EXPERIMENT 12

DATA TRANSFER: BLOCK DATA MOVEMENT, EXCHANGING DATA

1. Write an ALP to move a block of data from one location to another.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	78, 0A		MOV R0, #0AH	Setting count
4102	75, 82, 00		MOV DPL, #00H	Setting DPL = 00H
4105	75, 83, 45	LOOP:	MOV DPH, #45H	Setting DPH = 45H
4108	E0		MOVX A, @DPTR	Moving data from external memory to accumulator
4109	75, 83, 50		MOV DPH, #50H	Setting DPH = 50H
410C	F0		MOVX @DPTR, A	Moving data from accumulator to external memory
410D	A3		INC DPTR	Increment data pointer
410E	D8, F5		DJNZ R0, LOOP	Decrement R0, goto label LOOP till value of R0 becomes 0
4110	80, FE	HLT:	SJMP HLT	Stop

2. Write an ALP to exchange a block of data between two locations.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	78, 0A		MOV R0, #0AH	Setting count

4102	75, 82, 00		MOV DPL, #00H	Setting DPL = 00H
4105	75, 83, 45	LOOP:	MOV DPH, #45H	Setting DPH = 45H
4108	E0		MOVX A, @DPTR	Moving data from external memory to accumulator
4109	F9		MOV R1, A	Move acc content to R1
410A	75, 83, 50		MOV DPH, #50H	Setting DPH = 50H
410D	E0		MOVX A, @DPTR	Moving data from external memory to accumulator
410E	C9		XCH A, R1	Exchange A and R1 contents
410F	F0		MOVX @DPTR, A	Moving data from accumulator to external
4110	75, 83, 45		MOV DPH, #45H	Setting DPH = 45H
4113	E9		MOV A, R1	Move content of R1 to acc
4114	F0		MOVX @DPTR, A	Moving data from accumulator to external
4115	A3		INC DPTR	Increment data pointer
4116	D8, ED		DJNZ R0, LOOP	Decrement R0, goto label LOOP till R0 becomes 0
4118	80, FE	HLT:	SJMP HLT	Stop

EXPERIMENT 13

ARITHMETIC OPERATIONS : ADDITION, SUBTRACTION, MULTIPLICATION AND DIVISION

1. Write an ALP to add two 8 bit numbers, sum 16 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	90, 42, 00		MOV DPTR, #4200H	Setting Data pointer to the operand location
4103	78, 00		MOV R0, #00H	R0 reserved for higher order byte
4105	E0		MOVX A, @DPTR	Copying first data from external memory
4106	F9		MOV R1, A	Copying the data to R1
4107	A3		INC DPTR	Incrementing data pointer
4108	E0		MOVX A, @DPTR	Copying next data from external memory
4109	29		ADD A, R1	Performing addition
410A	50, 01		JNC SKIP	Checking if there is carry after add
410C	08		INC R0	If carry exist, increment R0, else skip
410D	A3	SKIP:	INC DPTR	Incrementing data pointer
410E	F0		MOVX @DPTR, A	Moving the lower order byte of result to external memory
410F	E8		MOV A, R0	Copying higher order byte to accumulator

4110	A3		INC DPTR	Incrementing data pointer
4111	F0		MOVX @DPTR, A	Moving the lower order byte of result to external memory
4112	80, FE	HLT:	SJMP HLT	Stop

2. Write an ALP to subtract one 8 bit number from another.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	90, 42, 00		MOV DPTR, #4200H	Setting Data pointer to the operand location
4103	78, 00		MOV R0, #00H	R0 reserved for higher order byte
4105	E0		MOVX A, @DPTR	Copying first data from external memory
4106	F9		MOV R1, A	Copying the data to R1
4107	A3		INC DPTR	Incrementing data pointer
4108	E0		MOVX A, @DPTR	Copying next data from external memory
4109	C3		CLR C	Clearing carry
410A	99		SUBB A, R1	Performing subtraction
410B	50, 04		JNC SKIP	Checking if there is borrow
410D	08		INC R0	If borrow exists, incrementing R0
410E	F4		CPL A	Complement A
410F	24,01		ADD A, #01H	If borrow exist, finding the 2’s complement of result

4111	A3	SKIP:	INC DPTR	Incrementing data pointer
4112	F0		MOVX @DPTR, A	Copying magnitude part of result into external memory
4113	E8		MOV A, R0	Moving the sign flag into accumulator
4114	A3		INC DPTR	Incrementing data pointer
4115	F0		MOVX @DPTR, A	Moving sign flag into external memory
4116	80, FE	HLT:	SJMP HLT	Stop

3. Write an ALP to multiply two 8 bit numbers, result being a 16 bit number.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	90,45,00		MOV DPTR,#4500H	Loading data pointer with location of first operand
4103	E0		MOVX A, @DPTR	Copying data from external memory
4104	F5, F0		MOV B, A	Copying first operand to B register
4106	A3		INC DPTR	Incrementing data pointer to access next location
4107	E0		MOVX A, @DPTR	Copying second operand from external memory
4108	A4		MUL AB	Multiplication carried out
4109	A3		INC DPTR	Incrementing data pointer to store output
410A	F0		MOVX @DPTR, A	Storing lower order byte of result to external memory
410B	A3		INC DPTR	Incrementing data pointer

410C	E5, F0		MOV A, B	Copying higher order data of result from B register
410E	F0		MOVX @DPTR, A	Storing higher order byte of result to external memory
410F	80, FE	HLT:	SJMP HLT	Stop

4. Write an ALP to perform 8 bit division operation.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	90,45,00		MOV DPTR, #4500H	Loading data pointer with location of divisor
4103	E0		MOVX A, @DPTR	Copying divisor from external memory
4104	F5, F0		MOV B, A	Copying first operand to B register
4106	A3		INC DPTR	Incrementing data pointer to access next location
4107	E0		MOVX A, @DPTR	Copying dividend from external memory
4108	84		DIV AB	Division carried out
4109	A3		INC DPTR	Incrementing data pointer to store output
410A	F0		MOVX @DPTR, A	Storing quotient part of result to external memory
410B	A3		INC DPTR	Incrementing data pointer
410C	E5, F0		MOV A, B	Copying remainder part of result to B register
410E	F0		MOVX @DPTR, A	Storing higher order byte of result to external memory
410F	80, FE	HLT:	SJMP HLT	Stop

5. Write an ALP for decimal addition of two 8 bit numbers, sum being a 16 bit number.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	90, 42, 00		MOV DPTR, #4200H	Setting Data pointer to the operand location
4103	78, 00		MOV R0, #00H	R0 reserved for higher order byte
4105	E0		MOVX A, @DPTR	Copying first data from external memory
4106	F9		MOV R1, A	Copying the data to R1
4107	A3		INC DPTR	Incrementing data pointer
4108	E0		MOVX A, @DPTR	Copying next data from external memory
4109	29		ADD A, R1	Performing addition
410A	D4		DA A	Decimal adjust
410B	50, 01		JNC SKIP	Checking if there is carry after add
410D	08		INC R0	If carry exist, increment R0, else skip
410E	A3	SKIP:	INC DPTR	Incrementing data pointer
410F	F0		MOVX @DPTR, A	Moving the lower order byte of result to external memory
4110	E8		MOV A, R0	Copying higher order byte to accumulator
4111	A3		INC DPTR	Incrementing data pointer
4112	F0		MOVX @DPTR, A	Moving the higher order byte of result to external memory
4113	80, FE	HLT:	SJMP HLT	Stop

6. Write an ALP to add a series of 8 bit numbers, sum 16 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	90,45,00		MOV DPTR, #4500H	Setting data pointer to the operand location
4103	7A,00		MOV R2, #00H	Initialise R2
4105	78,0A		MOV R0, #0AH	Initialise count
4107	E0		MOVX A, @DPTR	Copying operand into the accumulator
4108	18		DEC R0	Decrement R0
4109	F9	BACK:	MOV R1, A	Move A content to R1
410A	A3		INC DPTR	Increment DPTR
410B	E0		MOVX A, @DPTR	Copying operand into the accumulator
410C	29		ADD A, R1	Add A and R1 contents
410D	50,01		JNC SKIP	Jump on no carry to skip
410F	0A		INC R2	Increment R2
4110	D8, F7	SKIP:	DJNZ R0, BACK	Decrement R0, goto label LOOP till value becomes 0
4112	90,50,00		MOV DPTR, #5000H	Setting data pointer to the result location
4115	F0		MOVX @DPTR, A	Copying result to external memory
4116	A3		INC DPTR	Increment DPTR

4117	EA		MOV A, R2	Move R2 to A
4118	F0		MOVX @DPTR, A	Store the final carry
4119	80, FE	HLT:	SJMP HLT	Stop

7. Write an ALP to find square of an 8 bit number, result being a 16 bit number.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	90, 42, 00		MOV DPTR, #4200H	Setting data pointer to the operand location
4103	E0		MOVX A, @DPTR	Copying operand into the accumulator
4104	F5, F0		MOV B, A	Copying content of accumulator to B register
4106	A4		MUL AB	Performing multiplication
4107	A3		INC DPTR	Incrementing data pointer
4108	F0		MOVX @DPTR, A	Copying lower order byte of result to external memory
4109	E5, F0		MOV A, B	Copying higher order byte of result to accumulator
410B	A3		INC DPTR	Incrementing data pointer
410C	F0		MOVX @DPTR, A	Copying higher order byte of result to external memory
410D	80, FE	HLT:	SJMP HLT	Stop

EXPERIMENT 14

IMPLEMENTATION OF BOOLEAN AND LOGICAL INSTRUCTIONS

1. Write an ALP to find the larger of two numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	90,45,00		MOV DPTR,#4500H	Setting initial value of Data pointer
4103	E0		MOVX A,@DPTR	Obtaining first number from external memory
4104	F5,F0		MOV B,A	First number moved to B reg
4106	A3		INC DPTR	Incrementing data pointer
4107	E0		MOVX A,@DPTR	Obtaining second number from external memory
4108	B5,F0,02		CJNE A,B,LOOP1	Compare and jump if not equal
410B	80,06		SJMP LOOP2	If equal, jump to LOOP2
410D	40,02	LOOP1:	JC LOOP3	If carry, jump to LOOP3
410F	F5,F0		MOV B,A	Larger number in B
4111	E5,F0	LOOP3:	MOV A,B	Move the result to ACC
4113	A3	LOOP2:	INC DPTR	Incrementing data pointer
4114	F0		MOVX @DPTR,A	Result moved to external memory
4115	80,FE	HLT:	SJMP HLT	Stop

2. Write an ALP to find the smaller of two numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	90,45,00		MOV DPTR,#4500H	Setting initial value of Data pointer
4103	E0		MOVX A,@DPTR	Obtaining first number from external memory
4104	F5,F0		MOV B,A	First number moved to B reg
4106	A3		INC DPTR	Incrementing data pointer
4107	E0		MOVX A,@DPTR	Obtaining second number from external memory
4108	B5,F0,02		CJNE A,B,LOOP1	Compare and jump if not equal
410B	80,06		SJMP LOOP2	If equal, jump to LOOP2
410D	50,02	LOOP1:	JNC LOOP3	If no carry, jump to LOOP3
410F	F5,F0		MOV B,A	Larger number in B
4111	E5,F0	LOOP3:	MOV A,B	Move the result to ACC
4113	A3	LOOP2:	INC DPTR	Incrementing data pointer
4114	F0		MOVX @DPTR,A	Result moved to external memory
4115	80,FE	HLT:	SJMP HLT	Stop

3. Write an ALP to find the largest number in an array.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	90,45,00		MOV DPTR,#4500H	Setting initial value of Data pointer
4103	E0		MOVX A,@DPTR	Obtaining count from external memory
4104	FD		MOV R5, A	Storing the count in register R5
4105	75, F0,00		MOV B,#00H	Largest number to be stored in B register
4108	A3	LOOP2:	INC DPTR	Incrementing data pointer
4109	E0		MOVX A,@DPTR	Moves data into accumulator from external memory
410A	B5,F0,02		CJNE A,B,LOOP1	Compare and jump if not equal
410D	80,04		SJMP LOOP3	If equal, jump to LOOP3
410F	40, 02	LOOP1:	JC LOOP3	If carry from comparison, no need to update B
4111	F5,F0		MOV B,A	Update B
4113	DD,F3	LOOP3:	DJNZ R5,LOOP2	Decrement and Jump if not equal to Zero
4115	90,46,00		MOV DPTR,#4600H	Setting data pointer with address to store result
4118	E5,F0		MOV A,B	Largest number available in accumulator
411A	F0		MOVX @DPTR,A	Storing the largest number to external memory
411B	80,FE	HLT:	SJMP HLT	Stop

4. Write an ALP to find the smallest number in an array.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	90,45,00		MOV DPTR,#4500H	Setting initial value of Data pointer
4103	E0		MOVX A,@DPTR	Obtaining count from external memory
4104	FD		MOV R5, A	Storing the count in register R5
4105	75,F0,FF		MOV B,#0FFH	Smallest number to be stored in B register
4108	A3	LOOP2:	INC DPTR	Incrementing data pointer
4109	E0		MOVX A,@DPTR	Moves data into accumulator from external memory
410A	B5,F0,02		CJNE A,B,LOOP1	Compare and jump if not equal
410D	80,04		SJMP LOOP3	If equal, jump to LOOP3
410F	50, 02	LOOP1:	JNC LOOP3	If no carry from comparison, no need to update B
4111	F5,F0		MOV B,A	Update B
4113	DD,F3	LOOP3:	DJNZ R5,LOOP2	Decrement and Jump if not equal to Zero
4115	90, 46,00		MOV DPTR,#4600H	Setting data pointer with address to store result
4118	E5,F0		MOV A,B	Smallest number available in accumulator
411A	F0		MOVX @DPTR,A	Storing the largest number to external memory
411B	80,FE	HLT:	SJMP HLT	Stop

5. Write an ALP to sort a given array in ascending order.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	78,05		MOV R0,#05H	Setting count in R0
4102	18		DEC R0	Decreasing count by one, max value of loop variable
4103	E8	LOOP3:	MOV A,R0	Copying loop variable to accumulator
4104	F9		MOV R1,A	Copying loop variable to R1
4105	90,45,00		MOV DPTR,#4500H	Setting initial value of Data pointer
4108	C0,83	LOOP2:	PUSH DPH	Saving DPH in stack
410A	C0,82		PUSH DPL	Saving DPL in stack
410C	E0		MOVX A,@DPTR	One number to acc
410D	F5,F0		MOV B,A	Number moved to B reg
410F	A3		INC DPTR	Incrementing data pointer
4110	E0		MOVX A,@DPTR	Second no. to accumulator
4111	B5,F0,02		CJNE A,B,LOOP1	Compare and jump if not equal
4114	80,0B		SJMP LOOP	Jump if equal to LOOP
4116	50,09	LOOP1:	JNC LOOP	Jump on no carry to LOOP
4118	D0,82		POP DPL	Stack contents to DPL
411A	D0,83		POP DPH	Stack contents to DPH

411C	F0		MOVX @DPTR,A	Acc content to external memory
411D	A3		INC DPTR	Updating data pointer
411E	E5,F0		MOV A,B	Copying B content to acc
4120	F0		MOVX @DPTR,A	Acc content to external memory
4121	D9,E5	LOOP:	DJNZ R1,LOOP2	Decrement and Jump if not Zero (for comparison)
4123	D8,DE		DJNZ R0,LOOP3	Decrement and Jump if not Zero (for pass)
4125	80,FE	HLT:	SJMP HLT	Stop

6. Write an ALP to sort a given array in descending order.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	78,05		MOV R0,#05H	Setting count in R0
4102	18		DEC R0	Decreasing count by one, max value of loop variable
4103	E8	LOOP3:	MOV A,R0	Copying loop variable to accumulator
4104	F9		MOV R1,A	Copying loop variable to R1
4105	90,45,00		MOV DPTR,#4500H	Setting initial value of Data pointer
4108	C0,83	LOOP2:	PUSH DPH	Saving DPH in stack
410A	C0,82		PUSH DPL	Saving DPL in stack
410C	E0		MOVX A,@DPTR	One number to acc

410D	F5,F0		MOV B,A	Number moved to B reg
410F	A3		INC DPTR	Incrementing data pointer
4110	E0		MOVX A,@DPTR	Second no. to accumulator
4111	B5,F0,02		CJNE A,B,LOOP1	Compare and jump if not equal
4114	80,0B		SJMP LOOP	Jump if equal to LOOP
4116	40,09	LOOP1:	JC LOOP	Jump on no carry to LOOP
4118	D0,82		POP DPL	Stack contents to DPL
411A	D0,83		POP DPH	Stack contents to DPH
411C	F0		MOVX @DPTR,A	Acc content to external memory
411D	A3		INC DPTR	Updating data pointer
411E	E5,F0		MOV A,B	Copying B content to acc
4120	F0		MOVX @DPTR,A	Acc content to external memory
4121	D9,E5	LOOP:	DJNZ R1,LOOP2	Decrement and Jump if not Zero (for comparison)
4123	D8,DE		DJNZ R0,LOOP3	Decrement and Jump if not Zero (for pass)
4125	80,FE	HLT:	SJMP HLT	Stop

EXPERIMENT 15

COUNTERS: HEXADECIMAL AND BCD COUNTERS

1. Write an ALP for implementing an 8-bit hexadecimal up-counter with a time delay of 1 second between consecutive numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	75, 50, 00		MOV 50H,#00H	Setting internal ram location 50H with zero
4103	E5, 50	LOOP:	MOV A,50H	Copying content of internal ram location 50H
4105	F5,90		MOV P1,A	Copy content of Accumulator to port P1
4107	04		INC A	Increment content of accumulator
4108	F5,50		MOV 50H,A	Update content of location 50H
410A	31, 0E		ACALL DELAY	Call delay sub routine
410C	80, F5		SJMP LOOP	Jump to label “Loop”
410E	79, 03	DELAY:	MOV R1,#03H	Start of delay subroutine, setting content of R1 to 03
4110	90,38,00	L1:	MOV DPTR,#3800H	Using data pointer as a up counter
4113	A3	L2:	INC DPTR	Increment data pointer
4114	E5, 83		MOV A,DPH	Moving DPH to accumulator
4116	45, 82		ORL A,DPL	OR content of DPH and DPL
4118	70, F9		JNZ L2	If result of OR operation is non zero, jump to label L2
411A	D9, F4		DJNZ R1,L1	Decrement R1 and jump if not zero to L1

411C	22		RET	End of delay subroutine

Time delay calculation

Label	Mnemonics	Number of machine cycles per execution	Number of executions	Total no of machine cycles
	MOV R1,#03H	1	1	1
L1:	MOV DPTR,#X D	2	3	6
L2:	INC DPTR	2	3x (65535-X)	6x (65535-X)
	MOV A,DPH	1	3x (65535-X)	3x (65535-X)
	ORL A,DPL	1	3x (65535-X)	3x (65535-X)
	JNZ L2	2	3x (65535-X)	6x (65535-X)
	DJNZ R1,L1	2	3	6
	RET	2	1	2
		Total number of machine cycles		**1179645-18X**

Total number of machine cycles = 1179645 – 18X
One machine cycle consists of 12 clock cycles.
Clock Frequency applied to 8051 = 11.0592 MHz
Time delay to be generated = $\frac{1179645-18X}{11.0592 \times 10^6} \times 12$=1 second
1179645 – 18X = $\frac{11.0592 \times 10^6}{12}$=921600
X = $\frac{1179645-921600}{18} = 14335.83 \cong 14336\ D = 3800\ H$

2. Write an ALP for implementing an 8-bit hexadecimal down-counter with a time delay of 1 second between consecutive numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	75, 50, FF		MOV 50H,#0FFH	Setting internal ram location 50H with FFH
4103	E5, 50	LOOP:	MOV A,50H	Copying content of internal ram location 50H

4105	F5,90		MOV P1,A	Copy content of Accumulator to port P1
4107	14		DEC A	Decrement content of accumulator
4108	F5,50		MOV 50H,A	Update content of location 50H
410A	31, 0E		ACALL DELAY	Call delay sub routine
410C	80, F5		SJMP LOOP	Jump to label “Loop”
410E	79, 03	DELAY:	MOV R1,#03H	Start of delay subroutine, setting content of R1 to 03
4110	90,38,00	L1:	MOV DPTR,#3800H	Using data pointer as a up counter
4113	A3	L2:	INC DPTR	Increment data pointer
4114	E5, 83		MOV A,DPH	Moving DPH to accumulator
4116	45, 82		ORL A,DPL	OR content of DPH and DPL
4118	70, F9		JNZ L2	If result of OR operation is non zero, jump to label L2
411A	D9, F4		DJNZ R1,L1	Decrement R1 and jump if not zero to L1
411C	22		RET	End of delay subroutine

Time delay calculation

Label	**Mnemonics**	**Number of machine cycles per execution**	**Number of executions**	**Total no of machine cycles**
	MOV R1,#03H	1	1	1
L1:	MOV DPTR,#X D	2	3	6
L2:	INC DPTR	2	3x (65535-X)	6x (65535-X)

	MOV A,DPH	1	3x (65535-X)	3x (65535-X)
	ORL A,DPL	1	3x (65535-X)	3x (65535-X)
	JNZ L2	2	3x (65535-X)	6x (65535-X)
	DJNZ R1,L1	2	3	6
	RET	2	1	2
		Total number of machine cycles		**1179645-18X**

Total number of machine cycles = 1179645 – 18X
One machine cycle consists of 12 clock cycles.
Clock Frequency applied to 8051 = 11.0592 MHz
Time delay to be generated = $\frac{1179645-18X}{11.0592 \times 10^6} \times 12$=1 second
1179645 – 18X = $\frac{11.0592 \times 10^6}{12}$=921600
X = $\frac{1179645-921600}{18} = 14335.83 \cong 14336\ D = 3800\ H$

3. Write an ALP for implementing an 8-bit BCD up-counter with a time delay of 1 second between consecutive numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	75, 50, 00		MOV 50H,#00H	Setting internal ram location 50H with zero
4103	E5, 50	LOOP:	MOV A,50H	Copying content of internal ram location 50H
4105	F5,90		MOV P1,A	Copy content of Accumulator to port P1 register
4107	24,01		ADD A,#01H	Increment content of accumulator
4109	D4		DA A	Decimal adjust accumulator
410A	F5,50		MOV 50H,A	Update content of location 50H
410C	31, 10		ACALL DELAY	Call delay sub routine
410E	80, F3		SJMP LOOP	Jump to label “Loop”

4110	79,03	DELAY:	MOV R1,#03H	Start of delay subroutine, setting content of R1 to 03
4112	90,38,00	L1:	MOV DPTR,#3800H	Using data pointer as a up counter
4115	A3	L2:	INC DPTR	Increment data pointer
4116	E5, 83		MOV A,DPH	Moving DPH to accumulator
4118	45, 82		ORL A,DPL	OR content of DPH and DPL
411A	70, F9		JNZ L2	If result of OR operation is non zero, jump to label L2
411C	D9, F4		DJNZ R1,L1	Decrement R1 and jump if not zero to L1
411E	22		RET	End of delay subroutine

Time delay calculation

Label	**Mnemonics**	**Number of machine cycles per execution**	**Number of executions**	**Total no of machine cycles**
	MOV R1,#03H	1	1	1
L1:	MOV DPTR,#X D	2	3	6
L2:	INC DPTR	2	3x (65535-X)	6x (65535-X)
	MOV A,DPH	1	3x (65535-X)	3x (65535-X)
	ORL A,DPL	1	3x (65535-X)	3x (65535-X)
	JNZ L2	2	3x (65535-X)	6x (65535-X)
	DJNZ R1,L1	2	3	6
	RET	2	1	2
		Total number of machine cycles		**1179645-18X**

Total number of machine cycles = 1179645 – 18X
One machine cycle consists of 12 clock cycles.
Clock Frequency applied to 8051 = 11.0592 MHz
Time delay to be generated = $\frac{1179645-18X}{11.0592 \times 10^6} \times 12$=1 second

$1179645 - 18X = \frac{11.0592 \times 10^6}{12} = 921600$

$X = \frac{1179645 - 921600}{18} = 14335.83 \cong 14336\ D = 3800\ H$

4. Write an ALP for implementing an 8-bit BCD down-counter with a time delay of 1 second between consecutive numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
4100	75,50,99		MOV 50H,#99H	Setting internal ram location 50H with 99H
4103	E5, 50	DOWN:	MOV A,50H	Copying content of internal ram location 50H
4105	F5, 90		MOV P1,A	Copy content of Accumulator to port P1 register
4107	24, 99		ADD A,#99H	Adding 99H to accumulator
4109	D4		DA A	Decimal adjust accumulator
410A	F5, 50		MOV 50H,A	Update content of location 50H
410C	31, 10		ACALL DELAY	Call delay sub routine
410E	80, F3		SJMP DOWN	Jump to label “Loop”
4110	79, 03	DELAY:	MOV R1,#03H	Start of delay subroutine, setting content of R1 to 03
4112	90,38,00	L1:	MOV DPTR,#3800H	Using data pointer as a up counter
4115	A3	L2:	INC DPTR	Increment data pointer
4116	E5, 83		MOV A,DPH	Moving DPH to accumulator
4118	45, 82		ORL A,DPL	OR content of DPH and DPL

411A	70, F9		JNZ L2	If result of OR operation is non zero, jump to label L2
411C	D9, F4		DJNZ R1,L1	Decrement R1 and jump if not zero to L1
411E	22		RET	End of delay subroutine

Time delay calculation

Label	**Mnemonics**	**Number of machine cycles per execution**	**Number of executions**	**Total no of machine cycles**
	MOV R1,#03H	1	1	1
L1:	MOV DPTR,#X D	2	3	6
L2:	INC DPTR	2	3x (65535-X)	6x (65535-X)
	MOV A,DPH	1	3x (65535-X)	3x (65535-X)
	ORL A,DPL	1	3x (65535-X)	3x (65535-X)
	JNZ L2	2	3x (65535-X)	6x (65535-X)
	DJNZ R1,L1	2	3	6
	RET	2	1	2
		Total number of machine cycles		**1179645-18X**

Total number of machine cycles = 1179645 – 18X
One machine cycle consists of 12 clock cycles.
Clock Frequency applied to 8051 = 11.0592 MHz
Time delay to be generated = $\frac{1179645-18X}{11.0592 \times 10^6} \times 12$=1 second
1179645 – 18X = $\frac{11.0592 \times 10^6}{12}$=921600
X = $\frac{1179645-921600}{18} = 14335.83 \cong 14336\ D = 3800\ H$