Jawaharlal Nehru Engineering College

Laboratory Manual

Microprocessors and Microcontroller

For

Third Year Students

Manual made by

Prof. Dr. V. A. More

Prof. Ms.S.D.Gavarskar

Prof.Ms. A. R. Salunke

Author JNEC, Aurangabad

MGM'S

Jawaharlal Nehru Engineering College

N-6, CIDCO, Aurangabad

Department of Electronics &Telecommunication

Vision of the Department:

To develop **GREAT** technocrats and to establish centre of excellence in the field of **Electronics and Telecommunications. _**

1. ***G***lobal technocrats with human values
2. ***R***esearch and lifelong learning attitude,
3. ***E***xcellent ability to tackle challenges
4. ***A***wareness of the needs of society
5. ***T***echnical expertise

Mission of the Department:

1. To provide good technical education and enhance technical competency by providing good infrastructure, resources, effective teaching learning process and competent, caring and committed faculty.
2. To provide various platforms to students for cultivating professional attitude and ethical values.
3. Creating a strong foundation among students which will enable them to pursue their career choice.

Jawaharlal Nehru Engineering College

Technical Document

This technical document is a series of Laboratory manuals of Electronics and Telecommunication Department and is a certified document of Jawaharlal Nehru Engineering College. The care has been taken to make the document error-free. But still if any error is found. Kindly bring it to the notice of subject teacher and HOD.

Recommended by,

HOD

Approved by,

Principal

Copies:

1. Departmental Library
2. Laboratory
3. HOD
4. Principal

FOREWORD

It is my great pleasure to present this laboratory manual for third year engineering students for the subject of Microprocessors and Microcontroller keeping in view the vast coverage required for visualization of concepts of processor.

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relived them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

H.O.D.

LABORATORY MANUAL CONTENTS

This manual is intended for the Third year students of engineering branches in the subject of Microprocessors and Microcontroller. This manual typically contains practical/Lab Sessions related Processors covering various aspects related to the subject to enhance understanding.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions

Prof. Dr. V. A. More

Prof. Ms.S.D.Gavarskar

Prof.Ms. A. R. Salunke

LIST OF EXEPRIMENTS

Prerequisite-I
Study of Number System

Aim: To study various number systems.

Definition: Number system is the way to represent a number in different forms.

Types of Number system:

1. **Binary Number System:** It is the number system with base value 2 means it has only two digits to represent the data. The digits are (0, 1). E.g. 00,01,10,11,100….
2. **Decimal Number System:** It is the number system with base value 10 means it has 10-digits to represent the data. The digits are(0-9). Eg. 0,1,2,3,4,5,6 ……
3. **Octal Number System:** It is the number system with base value 8 means it has 8 digits to represent the data. The digits are (0-7).
4. **Hexadecimal Number System :** It is the number system with base value 16 means it has 16 digits to represent the data. The digits are (0-15). Eg. 0,1,2,3…….,9,A,B,C,D,E,F

Bits & Bytes:

1 bit(binary digit) = 1 **digit**. For example: 1
1 byte = 8-**bits**
1 kilo byte= 2^{10} = 1024 **bytes**
1 mega byte = $2^{10} * 2^{10} = 2^{20}$ = 1024 **kilo bytes**
1 giga byte= 2^{30}= 1024 **mega bytes**
1 tera byte= 2^{40}= 1024 **giga bytes**

Binary Number System:
In binary number system is made up of 2 digits- 0 and 1. We use these two digits to represent data.

So we count in the same way as using decimal number system. For example:
Decimal number system start at 0 and then 1,2,3,4,5,6,7,8,9… now what after nine repeat the no in combination such as start at 0 again the add 1 to the left of 0 resultant 10 , 11 ,12… so on. 100, 1000 etc.

The same method we follow in Binary number system:

Decimal Number	Binary Value
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011

Octal number System:

The octal numeral system, or oct for short, is the base-8 number system, and uses the digits 0 to 7. Octal numerals can be made from binary numerals by grouping consecutive binary digits into groups of three (starting from the right)

Decimal numbers	Binary Number	Octal Number
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7
8	001 000	10
9	001 001	11
10	001 010	12
11	001 011	13
12	001 100	14
13	001 101	15
14	001 110	16
15	001 111	17

Hexadecimal Number system:

This number system has a base value 16. We can use (0-15) decimal numbers to represent hexadecimal numbers. It uses 16 distinct numbers to represent the values.

Decimal numbers	Binary Number	Hexadecimal Number
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

ASCII (American Standard Code Interchange Information) Code: **ASCII** is a character set that is used to interchange information to binary and from binary to decimal. ASCII is a 8-bit character containing 256 characters.

0-31	Control code
32-127	Standard, alphabet A-Z and a-z
128-255	Special Symbols, non standard characters

Convert one form of number into other form.

Conclusion: Thus we studied various number systems and their conversions.

Prerequisite-II
Introduction to Microprocessor 8085

Aim

To study the microprocessor 8085

Architecture of 8085 Microprocessor

a) General purpose register

It is an 8 bit register i.e. B,C,D,E,H,L. The combination of 8 bit register is known as register pair, which can hold 16 bit data. The HL pair is used to act as memory pointer is accessible to program.

b) Accumulator

It is an 8 bit register which hold one of the data to be processed by ALU and stored the result of the operation.

c) Program counter (PC)

It is a 16 bit pointer which maintain the address of a byte entered to line stack.

d) Stack pointer (Sp)

It is a 16 bit special purpose register which is used to hold line memory address for line next instruction to be executed.

e) Arithmetic and logical unit

It carries out arithmetic and logical operation by 8 bit address it uses the accumulator content as input the ALU result is stored back into accumulator.

f) Temporary register

It is an 8 bit register associated with ALU hold data, entering an operation, used by the microprocessor and not accessible to programs.

g) Flags

Flag register is a group of fire, individual flip flops line content of line flag register will change after execution of arithmetic and logic operation. The line states flags are

i) Carry flag (C)
ii) Parity flag (P)
iii) Zero flag (Z)
iv) Auxiliary carry flag (AC)
v) Sign flag (S)

h) Timing and control unit

Synchronous all microprocessor, operation with the clock and generator and control signal from it necessary to communicate between controller and peripherals.

i) Instruction register and decoder

Instruction is fetched from line memory and stored in line instruction register decoder the stored information.

j) Register Array

These are used to store 8 bit data during execution of some instruction.

PIN Description

Address Bus

6. The pins $A_o - A_{15}$ denote the address bus.
7. They are used for most significant bit

Address / Data Bus

4. $AD_0 - AD_7$ constitutes the address / Data bus
5. These pins are used for least significant bit

ALE : (Address Latch Enable)

5. The signal goes high during the first clock cycle and enables the lower order address bits.

IO / M

1. This distinguishes whether the address is for memory or input.
2. When this pins go high, the address is for an I/O device.

$S_0 - S_1$

S_0 and S_1 are status signal which provides different status and functions.

RD

1. This is an active low signal
2. This signal is used to control READ operation of the microprocessor.

WR

1. WR is also an active low signal
2. Controls the write operation of the microprocessor.

HOLD

1. This indicates if any other device is requesting the use of address and data bus.

HLDA

1. HLDA is the acknowledgement signal for HOLD
2. It indicates whether the hold signal is received or not.

INTR

1. INTE is an interrupt request signal
2. IT can be enabled or disabled by using software

INTA

1. Whenever the microprocessor receives interrupt signal
2. It has to be acknowledged.

RST 5.5, 6.5, 7.5

1. These are nothing but the restart interrupts
2. They insert an internal restart junction automatically.

TRAP

1. Trap is the only non-maskable interrupt
2. It cannot be enabled (or) disabled using program.

RESET IN

1. This pin resets the program counter to 0 to 1 and results interrupt enable and HLDA flip flops.

X_1, X_2

These are the terminals which are connected to external oscillator to produce the necessary and suitable clock operation.

SID

This pin provides serial input data

SOD

This pin provides serial output data

V_{CC} and V_{SS}

1. V_{CC} is +5V supply pin
2. V_{SS} is ground pin

Specifications

1. Processors

Intel 8085 at E144 MHz clock

2. Memory

Monitor RAM:	0000– IFFF
EPROM Expansion:	2000– 3FFF
	0000– FFF
System RAM:	4000– 5FFF
Monitor data area	4100– 5FFF
RAM Expansion	6000– BFFF

3. Input / Output

Parallel: A8 TTL input timer with 2 number of 32-55 only input timer available in 8085 EBI.

Serial: Only one number RS 232-C, Compatible, crucial interface using 8281A

Timer: 3 channel -16 bit programmable units, using 8253 channel '0' used for no band late. Clock generator. Channel '1' is used for single stopping used program.

Display: 6 digit – 7 segment LED display with filter 4 digit for adder display and 2 digit for data display.

Key board: 21 keys, soft keyboard including common keys and hexa decimal keys.
RES: Reset keys allow to terminate any present activity and retain to □ - 85 its on initialize state.
INT: Maskable interrupt connect to CPU's RST 7.5 interrupt

DEC: Decrement the adder by 1

EXEC: Execute line particular value after selecting address through go command.

NEXT: Increment the address by 1 and then display its content.

Conclusion: Thus 8085 microprocessor was studied successfully.

Experiment No.1

Programming for Arithmetic operations**:**

a.1) Addition of two 8 bit numbers.

Aim : To write assembly language program for addition of two 8 bit numbers.

Statement: Add two 8 bit numbers, where first no. is present in memory location 2400 and second no. in 2401.Store result in memory location 2402.

Algorithm:

1. Initialize HL pair to point to the memory location of first no.
2. Transfer the first no. to Register A.
3. Increment HL pair to point to the next no.
4. Add contents of the memory location pointed by HL pair and the contents of register A.
5. Store the result in the mentioned memory location.
6. Stop.

Sample program:

Memory Address	Opcode/data/ address	label	Mnemonics	Comments
7000,01,02	21,00,24		LXIH,2400 H	Initialize HL pair
7003	7E		MOV A, M	Transfer first no. to reg A
7004	23		INX H	Increment HL pair
7005	86		ADD M	Add the two nos.
7006,07,08	32,02,24		STA2402 H	Store the result in 2403H
7009	CF		RST1	stop

Data: 2400H=11H and 2401H=12H

Result: 2402H=23 H

Conclusion: Thus, the program written for addition of two 8 bit numbers is successfully executed

a.2) Subtraction of two 8 bit numbers.

Aim: To write assembly language program for Subtraction of two 8 bit numbers.

Statement: Subtract two 8 bit numbers. Subtract the second no.present in memory location 2401H from the first no. present in memory location 2400H and Store result in memory location 2402H.

Algorithm:

1. Initialize HL pair to point to the memory location of first no.
2. Transfer the first no. to Register A.
3. Increment HL pair to point to the next no.
4. Subtract the contents of the memory location pointed by HL pair from contents of register A.
5. Store the result in the mentioned memory location.
6. Stop.

Sample Program:

Memory Address	Opcode/data/ address	label	Mnemonics	Comments
7000,01,02	21,00,24		LXIH,2400 H	Initialize HL pair
7003	7E		MOV A, M	Transfer first no. to reg A
7004	23		INX H	Increment HL pair
7005	96		SUB M	Add the two nos.
7006,07,08	32,02,24		STA2402 H	Store the result in 2402H
7009	CF		RST1	stop

Data: 2400H=05H and 2401H=02H

Result: 2403H=03 H
Conclusion: Thus program written for Subtraction of two 8 bit numbers is successfully executed.

b.1) Addition of two 16 bit numbers.

Aim: To write assembly language program for addition of two 16 bit numbers.
Statement: Write an ALP to add two 16 bit numbers where the first number is stored in memory location 2501H (LSB) and 2502H (MSB). Second number is in memory location 2503H (LSB) and 2504H (MSB) and the sum is stored in memory location 2505 (LSB) to 2507(MSB). (Consider the carry generated in the result).

Algorithm

1. Get the first number from the memory location to HL pair.
2. Transfer it to DE pair.
3. Get the second number from the next two memory locations to HL pair
4. Initially, set register C to 0. (Reg C is used to store the carry generated.
5. Add the two sixteen bit numbers directly.
6. If carry is generated, increment C.
7. Store result in the given memory locations.
8. Stop.

Sample Program:

Memory Address	Opcode/data/ address	label	Mnemonics	Comments
7000,01,02	2A,01,25		LHLD 2501	1st 16 bit number in HL pair
7003	EB		XCHG	Transfer 1st no. to DE pair
7004,05,06	2A,03,25		LHLD2503	2nd 16 bit number in HL pair
7007,08	0E,00		MVI C,00	MSBs of sum in register C. initial value =00
7009	19		DAD D	1st number + 2nd no.
700A,0B,0C	D2,0E,20		JNC AHEAD	Is carry? No, go to the label AHEAD
700D	0C		INR C	Yes, increment C.
700E,0F,10	22,05,25	AHEAD	SHLD 2505	Store LSBs of sum in 2505 and 2506
7011	79		MOV A,C	MSBs of sum in accumulator
7012,13,14	32,07,25		STA 2507	Store MSBs of sum in 2507
7015	CF		RST1	Termination of program

Result:2505-E4H, LSB of sum. 2506-E9H, next higher byte result 2507-00H, MSB of result.

Conclusion: Thus program written for addition of two 16 bit numbers is successfully executed

b.1) Subtraction of two 16 bit numbers.

Aim: To write assembly language program for Subtraction of two 16 bit numbers

Statement: Two 16 bit numbers are stored from memory location. Subtract number stored at 2501H (LSB) & 2502H (MSB) from the 16 Bit number stored at 2503 (LSB) & 2504H (MSB).Store result at 2506H (LSB) & 2507H (MSB).

Algorithm:

1. Transfer the first number from the memory location to HL pair.
2. Transfer it to DE pair.
3. Load second 16 bit number from the next two memory locations in HL pair.
4. Subtract lower byte of 2nd number
5. Store result in L register.
6. Subtract higher byte of second number with borrow.
7. Store result in H register.
8. Stop

Sample Program:

Memory Address	Opcode/data/ address	label	Mnemonics	Comments
7000,01,02	2A,01,25		LHLD 2501	1st 16 bit number in HL pair
7003	EB		XCHG	Transfer 1st no. to DE pair
7004,05,06	2A,03,25		LHLD2503	2nd 16 bit number in HL pair
7007	7B		MOV A,E	Get lower byte of 1 number.
7008	95		SUB L	Subtract lower byte of 2 number
7009	6F		MOV L,A	Store the result in L register
700A	7A		MOV A,D	Get higher byte of 1 number
700B	9C		SBB H	Subtract higher byte of 2 number with borrow
7000C	67		MOV H,A	Store result in H register
700D,0E,0F	22,05,25		SHLD 2505	Store result at 2505H/2506H
7010	CF		RST1	Stop

Data: 2501H = 19H
2502H = 6AH
2503H = 15H
2504H = 5CH

Result: 6A19 H-5C15 H = 0E04 H
2505H = 04H
2506H = 0EH

Conclusion: Thus program written for Subtraction of two 16 bit numbers is successfully executed.

c) Addition of two 8 bit BCD numbers

Aim: To write assembly language program for addition of two 16 bit BCD numbers.

Statement: Write program to add two 4 digit BCD numbers. Assume data already exists in BC and DE register pairs.

Algorithm:

1. Add lower registers C and E.
2. Adjust the result to BCD
3. Save LSB of register in register L
4. Add higher registers B and D.
5. Adjust the result to BCD.
6. Save MSB of register in register H.
7. Stop.

Sample Program:

Memory Address	Opcode/data/ address	label	Mnemonics	Comments
7000	79		MOV A,C	Move contents of register C to register A
7001	83		ADD E	Add E to Acc.
7002	27		DAA	Adjust to BCD
7003	6F		MOV L,A	Move contents of register A to register L
7004	78		MOV A,B	Move contents of register B to register A
7005	8A		ADC D	A+CY+D->A
7006	27		DAA	Adjust to BCD
7007	67		MOV H,A	Move contents of register A to Reg H
7008	CF		RST1	Stop

Data: First 8bit BCD _____
Second 8bit BCD _____

Result: Addition is _____ in BCD form**.**

Conclusion: Thus program written for addition of two 16 bit BCD numbers using DAA is successfully executed.

Experiment No.2

Aim: To write assembly language program for logical operation.

g) Statement: Write program to mask upper four bits of given data.

Algorithm:

Move immediately ABH into accumulator.
AND immediately with 0F H with accumulator.
Stop.

Memory Address	Opcode/data address	Mnemonics	Comments
2000,01	3E AB	MVI A, AB H	Move immediately ABH into accumulator
2002,03	E6 0F	ANI 0FH	AND immediately with 0F H with accumulator
2004	CF	RST1	stop

Data: 8bit Number _____

Result: _ _ _ _ __ __ __

h) Statement: Write program to mask lower four bits of given data.

Algorithm:

Move immediately ABH into accumulator.
AND immediately with 0F H with accumulator.
Stop.

Memory Address	Opcode/data address	Mnemonics	Comments
2000,01	3E AB	MVI A, AB H	Move immediately ABH into accumulator
2002,03	E6 F0	ANI F0H	AND immediately with F0 H with accumulator
2004	CF	RST1	stop

Data: 8bit Number _____

Result: _ _ _ _ __ __ __

i) Statement: Write program to set D2 & D4 bit of a data .

Algorithm:

Move immediately ABH into accumulator.
AND immediately with 14 H with accumulator.
Stop.

Memory Address	Opcode/data address	Mnemonics	Comments
2000,01	3E AB	MVI A, AB H	Move immediately ABH into accumulator
2002,03	F6 14	ORI 14H	OR immediately with 14 H with accumulator
2004	CF	RST1	stop

Data: 8bit Number _____

Result: _ _ _ _ __ __ __

j) Statement: Write program to reset D7 bit of a data .

Algorithm:

Move immediately ABH into accumulator.
AND immediately with 0F H with accumulator.
Stop.

Memory Address	Opcode/data address	Mnemonics	Comments
2000,01	3E AB	MVI A, AB H	Move immediately ABH into accumulator
2002,03	E6 7F	ORI 14H	AND immediately with 7F H with accumulator
2004	CF	RST1	stop

Data: 8bit Number _____

Result: _ _ _ _ __ __ __

k) Statement: Write program to verify AND, OR, EXOR operation of given data .

Algorithm:

Move immediately ABH into accumulator.
Move immediately data to B reg.
Perform logical operation with B.
Stop.

Program for AND operation

Memory Address	Opcode/data address	Mnemonics	Comments
2000,01	3E AB	MVI A, AB H	Move immediately ABH into accumulator
2002,03	06 09	MVI B, 09 H	Move immediately 09H in B register
2004	A0	ANA B	AND B register with A
2005	CF	RST1	stop

Program for OR operation

Memory Address	Opcode/data address	Mnemonics	Comments
2000,01	3E AB	MVI A, AB H	Move immediately ABH into accumulator
2002,03	06 09	MVI B, 09 H	Move immediately 09H in B register
2004	B0	ORA B	OR B register with A
2005	CF	RST1	stop

Program for XOR operation

Memory Address	Opcode/data address	Mnemonics	Comments
2000,01	3E AB	MVI A, AB H	Move immediately ABH into accumulator
2002,03	06 09	MVI B, 09 H	Move immediately 09H in B register
2004	A8	XRA B	XOR B register with A
2005	CF	RST1	stop

Data: 8bit Number _____

_____

Result: AND_____OR_____EXOR_____

f) Complement of two 8 bit numbers

Aim: Write program to for 2's complement of 8 bit numbers.

Statement: To write assembly language program to find 2's complement of a number stored at 2200H and store the result at 2201H.

Algorithm:

Algorithm –

1. Load the data from memory 2200 into A (accumulator)
2. Complement content of accumulator
3. Add immediately 01H in accumulator.
4. Store content of accumulator in memory 2201.
5. Stop

Sample Program:

Memory Address	Opcode/data address	Mnemonics	Comments
2000	3E 00 22	LDA 2200H	Load the data from memory 2200 into A
2003	2F	CMA	Complement content of accumulator
2004	C6 01	ADI 01 H	Add immediately 01H in accumulator.
2006	32 01 22	STA 2201H	Store content of accumulator in memory 2201
2009	CF	RST1	stop

Data: _____

Result: 2's Complement is _____.

Conclusion: Thus program written for logical operation of 8 bit numbers and result verified.

Experiment No.3

Data transfer programs:

Aim: To write data transfer array programs and execute.

Statement: Five no. of bytes is stored from the memory locations 2201h. Transfer the entire block of data bytes from 2201H to 2301H onwards.

Algorithm (Logic):1.Initialize the source memory pointer.

1. Initialize the destination memory pointer.
2. Initialize the counter with 5.
3. Move the contents of the source memory to accumulator.
4. Do whatever manipulation is specified /required.
5. Transfer the accumulator contents to destination memory location.
6. Increment source, destination memory pointer and decrement the counter.
7. If the count is not zero, jump back to step 4.
8. If the count is zero, stop.

Sample:
N=5 bytes, source location: 2201 to 2205, destination location: 2301 to 2305.

Sample Program:

Memory Address	Opcode/data/ad dr	label	Mnemonics	Comments
7000,01,02	21,01,22		LXI H,2201H	Initialize HL pair with source (src) memory.
7003,04,05	01,01,23		LXI B,2301H	Initialize BC pair with destination (destn) memory.
7006,07	16,05		MVI D,05H	Initialize D with count=05
7008	7E	LOOP	MOV A,M	Transfer src data to accumulator (acc).
7009	02		STAX B	Transfer acc contents to destn.
700A	23		INX H	Increment HLpair by 1
700B	03		INX B	Increment BC pair by1
700C	15		DCRD	Decrement D by1
700D,0E,0F	C2,08,70		JNZ LOOP	Jump to loop if Zero flag is not set.
7010	CF		RST1	Stop
Data:	5 no. of bytes stored from 2201 onwards.			
Result:	Same 5 no. of bytes stored in 2301 onwards			

Conclusion: Thus program written for data transfer is successfully executed

Experiment No.4

a. Sort Series in Ascending Order

Aim: To write Assembly Language Program (ALP) for sorting the series in ascending order.

Statement: N no. of bytes are stored from the memory locations 2201h.(N can be mentioned directly or given in a location (2200H). Write an ALP to sort the series in ascending order.

Algorithm(Logic):

1.Initialize the source memory pointer.
2.Initialize two counters: one to compare the data and another to repeat the process of comparison till all the nos. are over.
3.The first and the next no. are compared. Smaller of the two is retained in the first memory location. The larger of the two is moved to the next memory location.
4.Decrement the count.
5.The second memory location is compared with the third and again step 3 is repeated till the count is zero.
6.At the end, the largest number will be at the last location.

Data: N no. of bytes stored from 2201 onwards.
Result: Numbers stored from 2201 are arranged in ascending order.
Sample:

N=(2200), source location: 2201 to 2205,destination location: 2201 to 2205.

Memory Address	Opcode/ data/addr	Label	Mnemonics	Comments
7000,01,02	3A,00,22		LDA 2200H	Load Acc with the count
7003	47		MOV B, A	Save count to reg B.
7004,05,06	21,01,22	LOOP2	LXI H, 2201H	Initialize HL pair with source (src) memory.
7007	48		MOV C, B	Initialize C with count
7008	7E	LOOP1	MOV A, M	Transfer src data to accumulator (acc).
7009	23		INX H	Increment HLpair by 1
700A	BE		CMP M	Compare the two nos.
700B,0C,0D	DA,13,70		JC DOWN	If no.1<no.2,go to Label DOWN
700E	56		MOV D,M	Save smaller no. in D
700F	77		MOV M,A	Larger no. in II memory location
7010	2B		DCX H	Decrement HL
7011	72		MOVM,D	Save smaller no. in I memory location.
7012	23		INX H	Increment HL to point to next memory location.
7013	0D	DOWN	DCR C	Decrement count by 1
7014,15,16	C2,08,70		JNZ LOOP1	If not zero, jump to loop1
7017	05		DCR B	Decrement count by 1
7018,19,1A	C2,04,70		JNZ LOOP2	If not zero, jump to loop2
701B	CF		RST1	Stop

Data: N no. of bytes stored from 2201 onwards.

Result: Numbers stored from 2201 are arranged in ascending order.

Conclusion: Thus, program written for arranging numbers in ascending order is successfully executed

b. Sort Series in Descending Order

Aim: To write Assembly Language Program (ALP) for sorting the series in Descending order.

Statement: N no. of bytes are stored from the memory locations 2201h.(N can be mentioned directly or given in a location (2200H) .Sort the series in Descending order.

Algorithm(Logic):

1.Initialize the source memory pointer.
2.Initialize two counters: one to compare the data and another to repeat the process of comparison till all the nos. are over.
3.The first and the next no. are compared .Larger of the two is retained in the first memory location. The smaller of the two is moved to the next mem location.
4.Decrement the count.
5 The second memory location is compared with the third and again step 3 is repeated till the count is zero.
6.At the end ,the smallest no. will be at the last location.

Sample:

N=(2200), source location: 2201 to 2205,destination location: 2201 to 2205.

Memory Address	Opcode/ data/addr	Label	Mnemonics	Comments
7000,01,02	3A,00,22		LDA 2200H	Load Acc with the count
7003	47		MOV B,A	Save count to reg B.
7004,05,06	21,01,22	LOOP2	LXI H,2201H	. Initialize HL pair with source(src) memory.
7007	48		MOV C,B	Initialize C with count
7008	7E	LOOP1	MOV A,M	Transfer src data to accumulator(acc).
7009	23		INX H	Increment HLpair by 1
700A	BE		CMP M	Compare the two nos.
700B,0C,0D	D2,13,70		JNC DOWN	If no.1>no.2,go to Label DOWN
700E	56		MOV D,M	Save smaller no.in D
700F	77		MOV M,A	Larger no. in II memory location.
7010	2B		DCX H	Decrement HL
7011	72		MOVM,D	Save smaller no. in I memory location.
7012	23		INX H	Increment HL to point to next memory location.
7013	0D	DOWN	DCR C	Decrement count by 1
7014,15,16	C2,08,70		JNZ LOOP1	If not zero ,jump to loop1
7017	05		DCR B	Decrement count by 1
7018,19,1A	C2,04,70		JNZ LOOP2	If not zero ,jump to loop2
701B	CF		RST1	Stop

Data: N no. of bytes stored from 2201 onwards.

Result: Numbers stored from 2201 are arranged in descending order.

Conclusion: Thus, program written for arranging numbers in descending order is successfully executed

Experiment No 5

Aim : To study 8255 Programmable peripheral Interface

Theory:

The 8255A is a general purpose programmable I/O device designed to transfer the data from I/O to interrupt I/O under certain conditions as required. It can be used with almost any microprocessor.

It consists of three 8-bit bidirectional I/O ports (24I/O lines) which can be configured as per the requirement.

Ports of 8255A

8255A has three ports, i.e., PORT A, PORT B, and PORT C.

- **Port A** contains one 8-bit output latch/buffer and one 8-bit input buffer.
- **Port B** is similar to PORT A.
- **Port C** can be split into two parts, i.e. PORT C lower (PC0-PC3) and PORT C upper (PC7-PC4) by the control word.

These three ports are further divided into two groups, i.e. Group A includes PORT A and upper PORT C. Group B includes PORT B and lower PORT C. These two groups can be programmed in three different modes, i.e. the first mode is named as mode 0, the second mode is named as Mode 1 and the third mode is named as Mode 2.

Operating Modes

8255A has three different operating modes −

- **Mode 0** − In this mode, Port A and B is used as two 8-bit ports and Port C as two 4-bit ports. Each port can be programmed in either input mode or output mode where outputs are latched and inputs are not latched. Ports do not have interrupt capability.
- **Mode 1** − In this mode, Port A and B is used as 8-bit I/O ports. They can be configured as either input or output ports. Each port uses three lines from port C as handshake signals. Inputs and outputs are latched.
- **Mode 2** − In this mode, Port A can be configured as the bidirectional port and Port B either in Mode 0 or Mode 1. Port A uses five signals from Port C as handshake signals for data transfer. The remaining three signals from Port C can be used either as simple I/O or as handshake for port B.

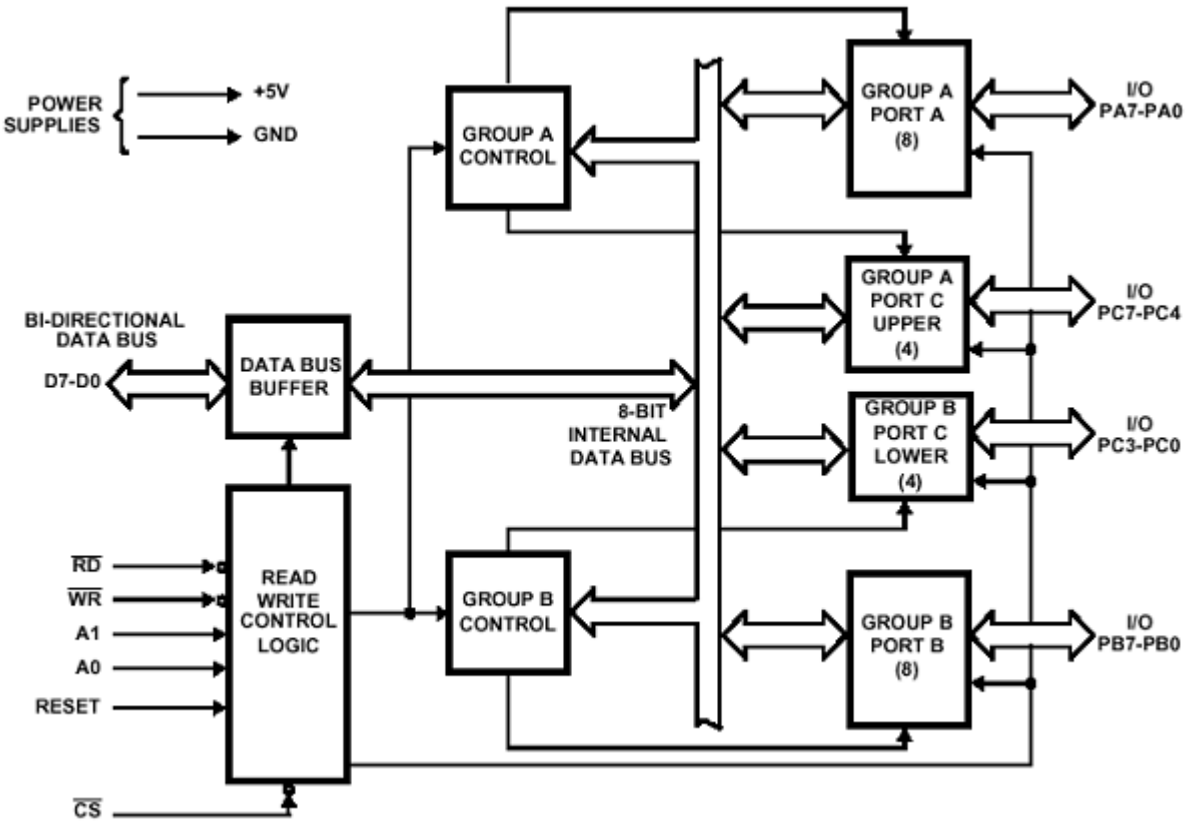
Features of 8255A

The prominent features of 8255A are as follows −

- It consists of 3 8-bit IO ports i.e. PA, PB, and PC.
- Address/data bus must be externally demux'd.
- It is TTL compatible.
- It has improved DC driving capability.

8255 Architecture

The following figure shows the architecture of 8255A −



The 8255A is a general purpose programmable I/O device designed to transfer the data from I/O to interrupt I/O under certain conditions as required. It can be used with almost any microprocessor.

It consists of three 8-bit bidirectional I/O ports (24I/O lines) which can be configured as per the requirement.

Ports of 8255A

8255A has three ports, i.e., PORT A, PORT B, and PORT C.

- **Port A** contains one 8-bit output latch/buffer and one 8-bit input buffer.
- **Port B** is similar to PORT A.
- **Port C** can be split into two parts, i.e. PORT C lower (PC0-PC3) and PORT C upper (PC7-PC4) by the control word.

These three ports are further divided into two groups, i.e. Group A includes PORT A and upper PORT C. Group B includes PORT B and lower PORT C. These two groups can be programmed in three different modes, i.e. the first mode is named as mode 0, the second mode is named as Mode 1 and the third mode is named as Mode 2.

Operating Modes

8255A has three different operating modes −

- **Mode 0** − In this mode, Port A and B is used as two 8-bit ports and Port C as two 4-bit ports. Each port can be programmed in either input mode or output mode where outputs are latched and inputs are not latched. Ports do not have interrupt capability.

- **Mode 1** − In this mode, Port A and B is used as 8-bit I/O ports. They can be configured as either input or output ports. Each port uses three lines from port C as handshake signals. Inputs and outputs are latched.
- **Mode 2** − In this mode, Port A can be configured as the bidirectional port and Port B either in Mode 0 or Mode 1. Port A uses five signals from Port C as handshake signals for data transfer. The remaining three signals from Port C can be used either as simple I/O or as handshake for port B.

Features of 8255A

The prominent features of 8255A are as follows −

- It consists of 3 8-bit IO ports i.e. PA, PB, and PC.
- Address/data bus must be externally demux'd.
- It is TTL compatible.
- It has improved DC driving capability.

Conclusion: Thus, we have studied 8255 Programmable peripheral Interface

Experiment No.6

LED INTERFACING USING 8255

Aim: Write an ALP to interface a LEDs with 8085 using 8255 to turn the LEDS on and off.

Logic: 1 Connect Leds to Port A of 8255, according to the circuit diagram shown.
Port A is used in output mode. Mode0.
2 Control Word is given to control register to set ports in different modes.
3 To make LEDs on, data FF should be sent on Port A.
4 Delay is provided.
5 To Off the LEDs, 00 should be sent.
6 Delay is provided.
7 Repeat steps 3 to 6 for continuous operation.

Subroutine:
1S.Initialise one register pair with count(Max=FFFFH).
2S.Decrement the reg pair.
3S.Check if the count has become zero.
4S.If no jump to step 2S.
5S. Otherwise return to main program.

Port addresses	Port A	Port B	Port C	Control Register
8255 Upper	00H	01H	02H	03H
8255 Lower	08H	09H	0AH	0BH

Control Word:
BSR/IO modeA PA PCu ModeB PB PCl
1 0 0 0 ** * * =80H

Write a program according to the logic of the program.

Theory:
Explain working of LED interfacing circuit**.**

Explain the interfacing of LED with 8085

Calculate the Delay .Generate a delay of 0.5sec.

Conclusion: Thus, we have interfaced a LEDs with 8085 using 8255 to turn the LEDS on and off.

Experiment no. 7

Aim - To study of assembler/simulator of 8051 microcontroller using KEIL µVision 3 software.

Theory-

Assembler-

An assembler allows to you to writer program using MCU instructions. It is used where almost speed, small code size and exact hardware control is essential. The KEIL assembler translate symbolic assembler language mnemonics into executable machine code while supporting source level , symbolic debugger powerful capabilities like macro processing.

The assembler translates assembly source files into reloadable object modules and optionally creates listing files with symbol tables and cross reference details, complete line number, symbol type information is written to generate object files. This information allows exact display of program variables in your debugger. Line numbers are used for source level debugging at the other third party debugging tools.

KEIL assembler supports several different types of macro processors depending on architecture. Standard macro processor is easier macro processor to use. It allows you to define and use macros in your assembly programs using syntax that is compatible with that used in many other assemblers. Macro processing language at the MPL is a string replacement facility that is compatible with in ASM-51. Micro processor MPL has several predefined macro processor. Macros save development and maintenance since common sequence need to be developed once.

Simulation-

Simulator mode configures the µvision debugger as software only product that accurate simulates target systems including systems including instruction and on chip peripherals. This allows application code testing before hardware is available and gives you several benefits for rapid, reliable embedded software development.

Simulation allows software testing on your desktop with no hardware environment. Early software debugging on functional bases improves overall software reliability. Simulation allows break points that are impossible with hardware debuggers. Simulation offers optimal inputs signals hardware debugger add extra noise. Single stepping through single processing algorithm is possible. External signal are stepped when CPU halts. Failure entries that would destroy real hardware peripherals are easily chore.

KEIL Compiler-

The µvision IDE is the easiest way for the most developers to create embedded SIM programs. To launch µvision click on your icon desktop at the selected KEIL µvision3 for start menu. The µvision screen provides a menu bar that contains means of commands various tool bars that contains buttons for common commands and window that displays project details, source files, dialogue box, and other information of course, multiple windows can be open simultaneously.

1. Menu Bar –

It provides access to most µvision commands including file operations, editor operations, project maintenance development tool setting program debugging, window selection and manipulation and online help.

2. File Menu –

It includes commands that are open, save, print and close source files device database and license manager dialogs are accessed from this menu.

3. Edit Menu-

It includes editing commands like undo, cut, copy, paste and indentation, bookmark functions advanced editor functions, editor configuration.

4. View Menu-

It includes commands that select which to do editor window and debugger window to display as well as commands that configure window viewing options.

5. Project Menu-

It includes commands that open, import, close project files. In addition project group and file options project more functions are accessed from this menu.

6. Debug Menu-

It includes commands that start and stop depression, run the program, single step, half program execution modify the program memory map. In addition commands are available to manage break points, setting trace recording, enable execution profile analyzer and performance analyzer and manage debug.

7. Flash Menu –

It include commands you use configure, program flash memory for embedded target system.

8. Peripheral Menu-

It includes dialogues that display and allow you change on-chip peripheral settings. The content of this menu are failed to explain specific microcontroller you to do.

9. Tool Menu-

It allows you to configure and integrate your several third party utilities directly from µvision IDE.

10. **SVCS Menu-**

It allows you to configure and integrate your several third party µvision.

11.Window Menu-

It includes arrange, split, select and close work space currently open bottom of menu.

12.Help menu-

It includes lone help system, list peripherals, access requests to technical support , check for product uploads display product µvision.

TOOL BARS-

The µvision IDE incorporates several tool bars that contain buttons for most commonly used commands.

i) File tool bar contains buttons for commands used to edit source file.
ii) Build tool bar contains buttons for commands used to build target.
iii) Debug tool bar contains buttons for commands used in debugger.

Simulator and programmer for 8051-

A simulator is software to mimic a microcontroller operation with a Personal Computer. This helps in running the assembly language program off-line and debug for errors. This is also a powerful learning tool before actually working with a Microcontroller A programmer is hardware used to transfer the machine code to the internal program memory of a microcontroller.

Working with KEIL Compiler µVision3-

KEIL Compiler µVision3 is a simulator/assembler for 8051 microcontroller to write and edit the code in assembly language, compile it and also to run the code. Output of the assembly language program can be verified using simulator.

Steps to use KEIL Compiler µVision3-

1. After installing the software, open8051 IDE
2. To create new project, go to: Project New Project, Save project (save project without any extension) select Microcontroller.
3. To write the assembly code in the editor , go to File New
4. After writing the assembly code in the editor, save the file with .asm extension.
5. Then go to work space and right click on source group and then add saved file.
6. Then write click on the added file and select built option to check the errors.
7. Check for the errors in the output window View Output
8. Once the error free code was made, debug the code. Debug Start debug session
9. Debug options are
    a. Step into – Each time only one instruction will be executed (single step mode).
    b. Run– To run the whole code at once.
10. Additional things:

a. To view RAM, program memory, SFRs, and External memory, serial window, logic analyzer, performance analyzer window use the option VIEW.
b. Toset break points in the code (where debugging stops at that point) Debug Insert Break Point.

11.To stop the simulation : Debug stop session
After checking the code in the simulator, the code (file with .HEX extension in Intel HEX format) is loaded into Atmel 89C51 microcontroller using Universal Programmer. **Although a separate 8051 assembly can be used at times assemble and generate Hex code for an assembly language program, KEIL Compiler µVision3 can perform that task.**

Conclusion: Thus, we have studied assembler/simulator of 8051 microcontroller using KEIL µVision 3 software.

Experiment No 8

8051 ARTHMETIC OPERATIONS-I USING KEIL

Aim:

To do the arithmetic operation (HEX and BCD add and subtract) using 8051 microcontroller

Algorithm:

Addition / Subtraction

(HEX)

Step 1 : Move first number to a register.
Step 2 : Move second number to another register
Step 3 : Add / Subtract two registers
Step 4 : Stop

1) 8 BIT ADDITION

```
MOV A, #05
MOV B,#06
ADD A,B
RET
```

INPUT		OUTPUT	
REGISTER	DATA	REGISTER	DATA
A	05	A	0B
B	06		

1) 8 BIT SUBSTRACTION

```
MOV A, #04
MOV B,#02
SUBB A,B
RET
```

INPUT		OUTPUT	
REGISTER	DATA	REGISTER	DATA
A	04	A	02
B	02		

Algorithm:

Addition / Subtraction

(BCD)

Step 1 : Move first number to a register.
Step 2 : Move second number to another register
Step 3 : Add/Subtract two registers
Step 4 : Adjust Accumulator
Step 5 : Stop

2) 8 BIT ADDITION

```
MOV A, #04
MOV B,#02
ADD A,B
DA A
RET
```

INPUT		OUTPUT	
REGISTER	DATA	REGISTER	DATA
A	05	A	11
B	06		

2) 8 BIT SUBSTRACTION

MOV A, #0F
MOV B,#04
SUBB A,B
DA A
RET

INPUT		OUTPUT	
REGISTER	**DATA**	**REGISTER**	**DATA**
A	**0F**	**A**	**11**
B	**04**		

Conclusion:

This experiment performs HEX and BCD addition successfully and result is verified.

Experiment No 9

8051 ARTHMETIC OPERATIONS-II USING KEIL

Aim:

To do the arithmetic operation multiplication and division using 8051 microcontroller

Algorithm:

Multiplication/Division

Step 1 : Move first number to a register.
Step 2 : Move second number to another register
Step 3 : Multiply /divide two registers
Step 4 : Stop

a) Multiplication of two 8 bit numbers

```
MOV A, #02
MOV B,#06
MUL  AB
RET
```

INPUT		OUTPUT	
REGISTER	**DATA**	**REGISTER**	**DATA**
A	**02**	**A**	**0C**
B	**06**		

b) Division of two 8 bit numbers

```
MOV A, #04
MOV B,#02
DIV AB
RET
```

INPUT		OUTPUT	
REGISTER	**DATA**	**REGISTER**	**DATA**
A	**04**	**A**	**02**
B	**02**		

Conclusion:

This experiment performs multiplication and division successfully and result is verified.

Experiment no. 10

Aim: Examining flags and stacks.(Kiel)

Theory:

CPU Registers

These topics provide an overview of the CPU registers available on the x51 variants. In addition to the CPU registers R0 - R7, all x51 variants have an SFR space that is used to address on-chip peripherals and I/O ports. The SFR area includes the CPU registers SP (stack pointer), PSW (program status word), A (accumulator, accessed via the SFR space as ACC), B, DPL and DPH (16-bit register DPTR).

8051 Variants

The classic 8051 provides 4 register banks of 8 registers each. These register banks are mapped into the DATA memory area at address 0 – 0x1F. In addition the CPU provides a 8-bit A (accumulator) and B register and a 16-bit DPTR (data pointer) for addressing XDATA and CODE memory. These registers are also mapped into the SFR space as special function registers.

Program Status Word (PSW)

The Program Status Word (PSW) contains status bits that reflect the current CPU state. The 8051 variants provide one special function register called PSW with this status information. The 251 provides two additional status flags, Z and N, that are available in a second special function register called PSW1.

PSW Register (all 8051 and 251 variants)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CY	AC	FO	RS1	RS0	OV	UD	P

Additional PSW1 Register (on 251 Architecture only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CY	AC	N	RS1	RS0	OV	Z	—

The following table describes the status bits in the PSW:
RS1 RS0 Working Register Bank and Address
0 0 Bank0 (D:0x00 - D:0x07)
0 1 Bank1 (D:0x08 - D:0x0F) ,
1 0 Bank2 (D:0x10 - D:0x17),
1 1 Bank3 (D:0x18H - D:0x1F)

Symbol	Function
CY	Carry flag
AC	Auxiliary Carry flag (For BCD Operations)
F0	Flag 0 (Available to the user for General Purpose)
RS1, **RS0**	Register bank select: RS1 RS0 Working Register Bank and Address 0 0 Bank0 (D:0x00 - D:0x07) 0 1 Bank1 (D:0x08 - D:0x0F) 1 0 Bank2 (D:0x10 - D:0x17) 1 1 Bank3 (D:0x18H - D:0x1F)
0V	Overflow flag
UD	User definable flag
P	Parity flag
—	Reserved for future use (251 Only)
Z	Zero flag (251 Only)
N	Negative flag (251 Only)

Sample Code

```
MOV A,#02H
MOV B,#25H
MUL AB
MOV R1,A
MOV R2,B
END
```

RESULT

Conclusion: Thus we have examined flags and stack status.

Post requisite -I

Aim: Program to test I\O ports using 8051(Bit manipulation)

Theory:

All 8051 microcontrollers have 4 I/O ports each comprising 8 bits which can be configured as inputs or outputs. Accordingly, in total of 32 input/output pins enabling the microcontroller to be connected to peripheral devices are available for use

PORT 0 :

Port-0 can be used as a normal bidirectional I/O port or it can be used for address/data interfacing for accessing external memory. When control is '1', the port is used for address/data interfacing. When the control is '0', the port can be used as a bidirectional I/O port.

P O RT 0 as an Input Port

Let us assume that control is '0'. When the port is used as an input port, '1' is written to the latch. In this situation both the output MOSFETs are 'off'. Hence the output pin have floats hence whatever data written on pin is directly read by read pin.

P O RT 0 as an Output Port

Suppose we want to write 1 on pin of Port 0, a '1' written to the latch which turns 'off' the lower FET while due to '0' control signal upper FET also turns off as shown in fig. above. Here we wants logic '1' on pin but we getting floating value so to convert that floating value into logic '1' we need to connect the pull up resistor parallel to upper FET . This is the reason why we needed to connect pull up resistor to port 0 when we want to initialize port 0 as an output port.

PORT 1:

Port-1 dedicated only for I/O interfacing. When used as output port, not needed to connect additional
pull-up resistor like port 0. It have provided internally pull-up resistor. The pin is pulled up or down
through internal pull-up when we want to initialize as an output port. To use port-1 as input port, '1' has
to be written to the latch. In this input mode when '1' is written to the pin by the external device then
it read fine. But when '0' is written to the pin by the external device then the external source must sink
current due to internal pull-up. If the external device is not able to sink the current the pin voltage may
rise, leading to a possible wrong reading.

PORT 2:

It has 8-pins (P2.0-P2.7).Port-2 we use for higher external address byte or a normal input/output port. The I/O operation is similar to Port-1. Port-2 latch remains stable when Port-2 pin are used for external memory access. Here again due to internal pull-up there is limited current driving capability. PORT 3:

Port-3 (P3.0-P3.7) having alternate functions to each pin,The internal structure of a port-3 pin is shown in fig below.

Following are the alternate functions of port 3:

It work as an IO port same like Port 2. only alternate function of port 3 makes its architecture different than other ports.

Algorithm:

1. Move data to accumulator.
2. Move data from accumulator to port 0, port 1 and port 2.
3. Delay subroutine.
4. Move another data to accumulator.
5. Move data from accumulator to port 0, port 1 and port 2.
6. Give delay
7. In delay subroutine initialize the counter.
8. Decrement count value until 0.
9. stop

Sample code:

```
ASSEMBLY PROGRAM
ORG 0000H
BACK: MOV A,#55H
      MOV P0,A
      MOV P1,A
      MOV P2,A
      ACALL DELAY
      MOV A,#0AAH
      MOV P0,A

      MOV P1,A
      MOV P2,A
      ACALL DELAY
      SJMP BACK
DELAY: MOV R5,#11
      MOV R4,#246
H3:    MOV R3,#255
H2:    DJNZ R3,H1
H1:    DJNZ R4,H2
      DJNZ R5,H3
      RET
      END
```

Conclusion: Thus we simulate I/O ports and observed bit manipulation simultaneously.

Post requisite -II

Aim:
Theory:

1. **Write the description about Stepper Motor Interfacing.**
2. **Prepare case study**

Conclusion: Thus we

III. Quiz on the subject.

Quiz should be conducted on tips in the laboratory, recent trends and subject knowledge of the subject. The quiz questions should be formulated such that questions are normally are from the scope outside of the books. However twisted questions and self formulated questions by the faculty can be asked but correctness of it is necessarily to be thoroughly checked before the conduction of the quiz.

Sample Questions on Microprocessor:

1. Define Microprocessor.
2. Define stack, stack pointer.
3. Define Memory.
4. What is RAM? Is RAM a volatile memory?
5. What is ROM? Is ROM used to store the binary codes for the instructions or lookup table? Why?
6. What is the function of "Timing and control unit" in microprocessor?
7. Which are the different types of buses used in microprocessor?
8. Explain fetching, decoding and execution operations of microprocessor.
9. Explain the difference between PROM,EPROM AND EEPROM.
10. Explain Different Blocks Of Microprocessor.
11. How many data lines, address lines are present in 8085.
12. How many address lines are required to access 2MB of memory.
13. List the internal registers in 8085.Describe the primary function of each register.
14. Give the clock frequency of 8085 operating with each of following
15. Frequency crystals:6.25MHZ,6.144MHz,5MHz,4MHz
16. Give the format of Flag Register in 8085.Explain each flag
17. Why AD0-AD7 lines are multiplexed?
18. What is the use of ALE signal?
19. What is the use of „clock out" and „reset out" signals of 8085?
20. Describe function of following pins in 8085:
    (1) READY (2) ALE (3) IO/M" (4) HOLD (5) RESET
    List the instructions related to DMA operation in 8085.
21. Stress out the necessity of having two status lines S1 and S0 in 8085.
22. List out different control signals used by 8085.
23. On power on reset, what is the content of PC ?
24. List the instructions related to serial operation in 8085.
25. List the different addressing modes of 8085.
26. Explain following instructions:
    1) PUSH 2)POP 3)CALL 4)RET
27. Explain 8255.
28 Explain 8253.
29. Explain 8279.
30. Explain 8259.

Quiz on basics of Microcontroller 8051

1. Abbreviate CISC and RISC.
    a) Complete Instruction Set Computer, Reduced Instruction Set Computer
    b) Complex Instruction Set Computer, Reduced Instruction Set Computer
    c) Complex Instruction Set Computer, Reliable Instruction Set Computer
    d) Complete Instruction Set Computer, Reliable Instruction Set Computer
    Answer: b
2. What is the order decided by a processor or the CPU of a controller to execute an instruction?
    a) decode,fetch,execute

b) execute,fetch,decode
c) fetch,execute,decode
d) fetch,decode,execute
Answer: d

3. If SUBB A,R4 is executed, then actually what operation is being applied?
a) R4+A
b) R4-A
c) A-R4
d) R4+A
Answer: c

5. In 8 bit signed number operations, OV flag is set to 1 if:
a) a carry is generated from D7 bit
b) a carry is generated from D3 bit
c) a carry is generated from D7 or D3 bit
d) a carry is generated from D7 or D6 bit
Answer: c

6. When we add two numbers the destination address must always be.
a) some immediate data
b) any register
c) accumulator
d) memory
Answer: c

7. Which out of the four ports of 8051 needs a pull-up resistor for using it is as an input or an output port?
a) PORT 0
b) PORT 1
c) PORT 2
d) PORT 3
Answer: a

8 Which of the ports act as the 16 bit address lines for transferring data through it?
a) PORT 0 and PORT 1
b) PORT 1 and PORT 2
c) PORT 0 and PORT 2
d) PORT 1 and PORT 3
Answer: c

9. Which operator is the most important while assigning any instruction as register indirect instruction?
a) $
b) #
c) @
d) &
Answer: c

10. Which pins of a micro controller are directly connected with 8255?
a) RD
b) WR
c) D0-D7
d) all of the mentioned
Answer: d

11 8051 series has how many 16 bit registers?

a) 2
b) 3
c) 1
d) 0

Answer: a

12 On power up, the 8051 uses which RAM locations for register R0- R7

a) 00-2F
b) 00-07
c) 00-7F
d) 00-0F

Answer: b

13. How many bytes of bit addressable memory is present in 8051 based micro controllers?

a) 8 bytes
b) 32 bytes
c) 16 bytes
d) 128 bytes

Answer: c

14. Which architecture is followed by general purpose microprocessors?

a) Harvard architecture
b) Von Neumann architecture
c) None of the mentioned
d) All of the mentioned

Answer: b

15 Which architecture provides separate buses for program and data memory?

a) Harvard architecture
b) Von Neumann architecture
c) None of the mentioned
d) All of the mentioned

Answer: a

IV. Conduction of Viva-Voce Examinations:

Teacher should conduct oral exams of the students with full preparation. Normally, the objective questions with guess should be avoided. To make it meaningful, the questions should be such that depth of the students in the subject is tested. Oral examinations are to be conducted in cordial environment amongst the teachers taking the examination. Teachers taking such examinations should not have ill thoughts about each other and courtesies should be offered to each other. Difference of opinion, if any, should be critically suppressed in front of the students.

V. Evaluation and marking system:

Basic honesty in the evaluation and marking system is absolutely essential and in the process impartial nature of the evaluator is required in the examination. It is a wrong approach to award the students by way of easy marking to get cheap popularity among the students, which they do not deserve. It is a primary responsibility of the teacher to see that right students who are really putting up lot of hard work with right kind of intelligence are correctly awarded.

The marking patterns should be justifiable to the students without any ambiguity and teacher should see that students are faced with just circumstances.