TPCT's

College of Engineering, Osmanabad

Laboratory Manual

Advanced Processors & Microcontrollers

For

Third Year Students

Manual Prepared by

Prof.L.M.Deshpande

Author COE, Osmanabad

TPCT's

College of Engineering
Solapur Road, Osmanabad
Department of Electronics &Telecommunication

Vision of the Department:

To be recognized by the society at large as an excellent department offering quality higher education in the Electronics & Telecommunication Engineering field with research focus catering to the needs of the public ind being in tune with the advancing technological revolution.

Mission of the Department:

To achieve the vision the department will

- Establish a unique learning environment to enable the student's face the challenges of **the Electronics & Telecommunication Engineering field.**
- **Promote the establishment of centers of excellence in technology areas to nurture the spirit of innovation and creativity among the faculty & students.**
- **Provide ethical & value based education by promoting activities addressing the needs of the society.**
- **Enable the students to develop skill to solve complete technological problems of current times and also to provide a framework for promoting collaborative and multidisciplinary activities.**

College of Engineering

Technical Document

This technical document is a series of Laboratory manuals of Electronics and Telecommunication Department and is a certified document of College of Engineering, Osmanabad. 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 **Advanced Processors & Microcontrollers** ,keeping in view the vast coverage required for visualization of concepts of Different aspects of the Processors.

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 Electronics & Telecommunication engineering branch in the subject of **Advanced Processors & Microcontrollers** . This manual typically contains practical/Lab Sessions related to **Advanced Processors & Microcontrollers** 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.

Prof.L.M.Deshpande

SUBJECT INDEX

DOs and DON Ts in Laboratory:

1. Do not handle any equipment before reading the instructions/Instruction manuals
2. Observe type of sockets of equipment power to avoid mechanical damage

3. Do not forcefully place connectors to avoid the damage
4. Strictly observe the instructions given by the teacher/Lab Instructor
5. Reset the trainer before the peripheral card is connected.
6. Check the proper connection of the peripheral card before running the programs
7.PC should be properly shut down before power supply is switched off.

Instruction for Laboratory Teachers::

1. Submission related to whatever lab work has been completed should be done during the next lab session.

2. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

Pre-lab Requisite.

EXECUTION OF USER PROGRAM ON AT 8086 TRAINER KIT

Execution procedure

In AT-86 system user can execute the program by folloeing the following steps.

1. First of all user should ready with the hex code of the program.
2. After completing the Installation procedure means displaying the power on prompt on the LCD, Press any hex key then system will enter in to the address set mode.
3. Enter the starting address of your program from 8400 onwards.
4. Press **NEXT** key then system will enter into the data set mode.
5. Here feed your data, use **NEXT** key to enter the data in the next location.
6. After feeding if you want to check the data use **PREV/NEXT** keys.
7. After checking the code either you press **RESET** followed by **GO** or **GO** key directly.
8. Then system will ask for the execution starting address, Enter the program address from where execution should start then press **NEXT** key.
9. Then ' Executing…. ' Message will appear and Program will be executed.
10. Then check the result of the program.

EXPERIMENT NO: 01
ADDITION OF TWO SIXTEEN BIT NUMBERS

Aim: Addition of two sixteen bit numbers stored in memory locations. Result of sixteen bit to be stored in memory location.

Apparatus/software:

1. 8086 micro processor kit
2. Desktop Computer

Objective: To study of different Assembler directives & Data transfer instruction of 8086 Processor.

Theory: Description of some Assembler Directives.

ASSUME Directive - The ASSUME directive is used to tell the assembler that the name of the logical segment should be used for a specified segment. The 8086 works directly with only 4 physical segments: a Code segment, a data segment, a stack segment, and an extra segment.

Example: ASUME CS:CODE ;This tells the assembler that the logical segment named CODE contains the instruction statements for the program and should be treated as a code segment.

ENDS - This ENDS directive is used with name of the segment to indicate the end of that logic segment.

EQU - This EQU directive is used to give a name to some value or to a symbol. Each time the assembler finds the name in the program, it will replace the name with the value or symbol you given to that name.

END - END directive is placed after the last statement of a program to tell the assembler that this is the end of the program module. The assembler will ignore any statement after an END directive.

Data Transfer Instructions MOV Des, Src:

Src operand can be register, memory location or immediate operand.
Des can be register or memory operand.
Both Src and Des cannot be memory location at the same time.

E.g.: MOV CX, 037A H
MOV AL, BL
MOV BX, [0301 H]

Program:

```
CODE   SEGMENT
       ASSUME          CS: CODE, DS: CODE, ES: CODE, SS: CODE
       RSTINT   EQU     03H
       ADR1     EQU     8500H ; Operand 1 address
       ADR2     EQU     8502H ; Operand 2 address
       ADR3     EQU     8504H ; Result address
       ORG      0400H
```

```
START :
MOV    BX,ADR1          ;Load BX with operand1 address
MOV    AX,[BX] ;Load AX with operand1
MOV    BX,ADR2          ;Load BX with operand2 address
ADD     AX,[BX]         ;Add operand2 to AX
MOV    BX,ADR3          ;Load BX with Result address
MOV    [BX],AX;Move the result to Result address
INT      RSTINT      ;Reset the system
END     START
CODE   ENDS
END
```

Observation Table:

ADDRESS	8500H	8502H	8504H
CONTENTS			

Conclusion: The addition of two 16 numbers is done .The result is found to be as expected.

Program for practice: Addition of two eight bit numbers stored in memory locations. Result of sixteen bit to be stored in memory location.

EXPERIMENT NO: 02
FACTORIAL OF NUMBER

Aim: Find the factorial of a given number. Store the result in memory location.

Apparatus/software:

1. 8086 micro processor kit/TASM (TURBO ASSEMBLER)
2. Desktop Computer

Objective: To study TASM (Turbo Assembler) & of different Arithmatic instruction of 8086 Processor.

Theory: Assembling the program using TASM

The assembler is used to convert the assembly language instructions to machine code. It is used immediately after writing the Assembly language program. The assembler starts by checking the syntax or validity of the structure of each instruction in the source file .if any errors are found, the assemblers displays a report on these errors along with brief explanation of their nature. However If the program does contain any errors ,the assembler produces an object file that has the same name as the original file but with the "obj" extension

Linking the program:

The Linker is used convert the object file to an executable file. The executable file is the final set of machine code instructions that can directly be executed by the microprocessor. It is the different than the object file in the sense that it is self-contained and re-locatable. An object file may represent one segment of a long program. This segment can not operate by itself, and must be integrated with other object files representing the rest of the program ,in order to produce the final self-contained executable file

Executing the program

The executable contains the machine language code .it can be loaded in the RAM and executed by the microprocessor simply by typing, from the DOS prompt ,the name of the file followed by the carriage Return Key (Enter Key). If the program produces an output on the screen or sequence of control signals to control a piece of hard ware, the effect should be noticed almost immediately.However, if the program manipulates data in memory, nothing would seem to have happened as a result of executing the program.

Procedure to enter a program using TASM software

Start the PC

↓

Get the DOS command prompt

↓

Type CD\

↓

Ok

Display shows

↓

C :\> C:\

↓

C :\> CD TASM

↓

Press ENTER

↓

C: \TASM> EDIT FILENAME.ASM

Example edit abc.asm

↓

Press ENTER

↓

Then the display shows editor

↓

Type the asm program

↓

Then the save the program

↓

Exit from editor Using Alt+F keys

↓

Then Display shows C: \TASM>

↓

Enter the name TASM FILENAME.ASM

Example

↓

C: \TASM> TASM abc.asm

Then Display shows Errors,(0)Warnings(0)

If there is errors correct them

↓

Enter the name Tlink FILENAME.OBJ

Example

↓

C: \TASM> TLINK abc.obj

↓

Then the display shows

Turbo Link Version 3.0

↓

Enter the name TD FILENAME.EXE

Example

↓

C: \TASM> TD abc.exe

↓

Then the display shows

Program has no symbol table

Choose OK

↓

RUN the Program using F9 Key or Select the RUN Option

↓

See the result in in Registers /Memory locations

Program:

```
CODE   SEGMENT
       ASSUME          CS: CODE, DS: CODE, ES: CODE, SS: CODE
       RSTINT   EQU     03H
       NUM      EQU     0005H ;  Given number
       ADR      EQU     8500H ; Result address
       ORG      0400H
       START :
       MOV     BX,ADR          ;Load BX with Result address
       MOV     CX,NUM          ;Load CX with number
       MOV     AX,CX   ;Copy the number in to AX
       DEC      CX             ;Decrement CX
UP;    MUL     CX              ;Multiply AX with CX
      LOOP   UP               ;Decrement CX,repeat the loop till CX is not zero
       MOV     BX,ADR3         ;Load BX with Result address
       MOV     [BX],AX;Move the result to Result address
       INT      RSTINT      ;Reset the system
       END      START
       CODE   ENDS
       END
```

Observation Table:

ADDRESS	8500H
CONTENTS	

Conclusion: The Factorial of a given number is found .The result is found to be as expected.

Program for practice: Find the factorial of a number stored in memory location.

Experiment No. 03
STRING DATA TRANSFER

Aim: Transfer the given Byte string from source location in Data segment to destination location in Extar segment

Apparatus/software:

1. 8086 micro processor kit/TASM (TURBO ASSEMBLER)
2. Desktop Computer

Objective: To study the string related operations.

Theory:
String Manipulation Instructions A series of data byte or word available in memory at consecutive locations, to be referred as Byte String or Word String. A String of characters may be located in consecutive memory locations, where each character may be represented by its ASCII equivalent.

The 8086 supports a set of more powerful instructions for string manipulations for referring to a string, two parameters are required. I. Starting and End Address of the String. II. Length of the String. The length of the string is usually stored as count in the CX register.The incrementing or decrementing of the pointer, in string instructions, depends upon the Direction Flag (DF) Status. If it is a Byte string operation, the index registers are updated by one. On the other hand, if it is a word string operation, the index registers are updated by two.

MOVSB / MOVSW :Move String Byte or String Word Suppose a string of bytes stored in a set of consecutive memory locations is to be moved to another set of destination locations.The starting byte of source string is located in the memory location whose address may be computed using SI (Source Index) and DS (Data Segment) contents. The starting address of the destination locations where this string has to be relocated is given by DI (Destination Index) and ES (Extra Segment) contents.

CMPS : Compare String Byte or String Word The CMPS instruction can be used to compare two strings of byte or words. The length of the string must be stored in the register CX. If both the byte or word strings are equal, zero Flag is set. The REP instruction Prefix is used to repeat the operation till CX (counter) becomes zero or the condition specified by the REP Prefix is False.

LODS : Load String Byte or String Word The LODS instruction loads the AL / AX register by the content of a string pointed to by DS : SI register pair. The SI is modified automatically depending upon DF, If it is a byte transfer (LODSB), the SI is modified by one and if it is a word transfer (LODSW), the SI is modified by two. No other Flags are affected by this instruction.

STOS : Store String Byte or String Word The STOS instruction Stores the AL / AX register contents to a location in the string pointer by ES : DI register pair. The DI is modified accordingly, No Flags are affected by this instruction.

Algorithm:

1: Load source string, length in location 2000H of DS
2. Load starting address of the source string 3000H in SI
3. Load starting address of the destination string 4000H in DI
4. Clear DF
5. Execute MOVSB instruction for count number of times
using the prefix REP.
5. End of the program

Input: Source String		Expected output: Destination String	
Address	Data	Address	Data
DS:3000H	12H	ES:4000H	12H
DS:3001H	56H	ES:4001H	56H
DS:3002H	9AH	ES:4002H	9AH
DS:3003H	0DEH	ES:4003H	0DEH
DS:3004H	0ABH	ES:4004H	0ABH

Program:

```
                ASSUME  CS:CODE, DS;DATA, ES:EXTRA
                DATA   SEGMENT
                SOURCE  EQU 3000H
                COUNT  EQU  2000H
                DATA   ENDS
                EXTRA   SEGMENT
                DEST EQU 4000H
                EXTRA   ENDS
                CODE   SEGMENT
                START:
  START:        MOV             SI,COUNT
                MOV             CL,[SI]
                MOV             SI,SOURCE
                MOV              DI,DEST
                CLD
                REP MOVSB
                INT             03H
                END     START
                CODE   ENDS
                END
```

Observed result:

Destination String	
Address	Data
ES:4000H	12H
ES:4001H	56H
ES:4002H	9AH
ES:4003H	0DEH
ES:4004H	0ABH

Conclusion: As expected the given string has been copied to the destination location.

Program for practice: Write & execute a program Word string transfer.

Experiment no. 04

Stepper motor control

Aim: ALP in 8086 to rotate the stepper motor in clockwise directions and anti-clockwise direction continuously, by interfacing stepper motor control module to 8086 microprocessor through Intel8255 .

Apparatus:

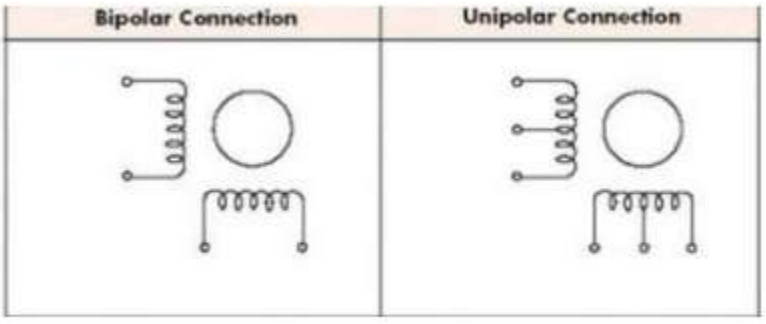
1. 8086 Microprocessor kit
2. Stepper motor module
3. Desktop Computer

Objective: To see the speed & direction control of stepper motor my means of 8086 processor program.

Theory:

A stepper motor is an electric motor that rotates in discrete step increments. The movement of each step is precise and repeatable; therefore the motor's position can be controlled precisely without any feedback mechanism, as long as the motor is carefully sized to the application. This type of control eliminates the need for expensive sensing and feedback devices such as optical encoders. The position is known simply by keeping track of the input step pulses. It is one of the most versatile forms of positioning systems. They are typically digitally controlled as part of an open loop system, and are simpler and more rugged than closed loop servo systems. Industrial applications include high speed pick and place equipment and multi-axis CNC machines, often directly driving lead screws or ballscrews. In the field of optics they are frequently used in precision positioning equipment such as linear actuators, linear stages, rotation stages, goniometers, and mirror mounts. Other uses are in packaging machinery, and positioning of valve pilot stages for fluid control systems. Commercially, stepper motors are used in floppy disk drives, flatbed scanners, computer printers, plotters, slot machines, image scanners, compact disc drives and many more devices.

Coil Windings and Stepping Mechanism:



Two common winding arrangements

There are two common winding arrangements for the electromagnetic coils: bipolar and unipolar (Fig 4). The described stepping sequence utilizes the bipolar winding. Each phase consists of a single winding. By reversing the current in the windings, electromagnetic polarity is reversed. A unipolar stepper motor has one winding with center tap per phase. Each section of windings is switched on for each direction of magnetic field. Since in this arrangement a magnetic pole can be reversed without switching the direction of current, the commutation circuit can be made very simple for each winding.

Program:

```
                ASSUME CS:CODE,DS:CODE,SS:CODE,ES:CODE
                CMDB8255  EQU    27H          ;Control register address
                PAB8255   EQU    21H ;Port A  address
                PBB8255   EQU    23H ;Port B address
                PCB8255   EQU    25H ;Port C  address
                ;CLOCK WISE 05H,06H,0AH,09H
                ;ANTI CLOCK WISE 09H,0AH,06H,05H
                ORG    400H

        CODE    SEGMENT
                START
                MOV     AL,80H          ;Move control word 80H into AL
                OUT     CMDB8255,AL   ;Send AL contents into Cntrol  register.
                MOV     AL,0FFH                 ;Move FFH into AL
                OUT     PAB8255,AL    ;Send the AL contents to Port A
START:          MOV     AL,05H                  ;Move FFH into AL
                OUT     PAB8255,AL    ;Send the AL contents to Port A
                 CALL   DELAY                   ;Call delay subroutine.
                MOV     AL,06H                  ;Move FFH into AL
                OUT     PAB8255,AL    ;Send the AL contents to Port A
                 CALL   DELAY                   ;Call delay subroutine.
                MOV     AL,0AH                  ;Move FFH into AL
                OUT     PAB8255,AL    ;Send the AL contents to Port A
                 CALL   DELAY                   ;Call delay subroutine.
                MOV     AL,09H                  ;Move FFH into AL
                OUT     PAB8255,AL    ;Send the AL contents to Port A
                 CALL   DELAY                   ;Call delay subroutine.
                JMP    START            ; Jump to START.

DELAY:
                MOV     CX,05FFH                ; Load CX with 05FFH.
LP:             DEC     CX                      ; Decrement CX by one.
                JZ     EXT                      ; Jump to EXT if ZF=1.
                JMP     LP                      ; Jump to LP.
EXT:            RET                             ; Return to main program
                END     START
        CODE    ENDS
                END
```

Circuit Diagram:

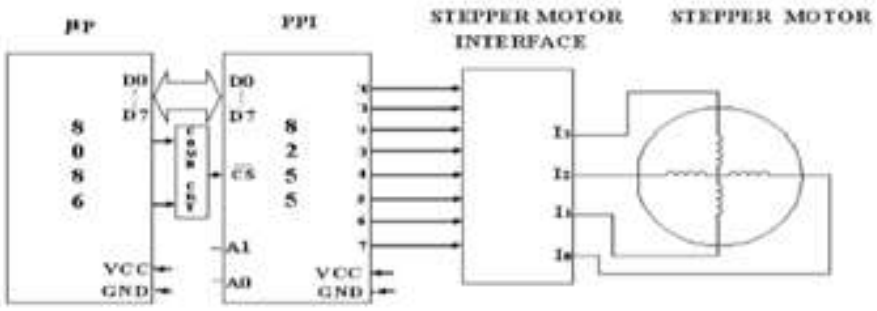


Fig: Interfacing stepper motor module to8086 microprocessor

Observed Result: When windings are excited in proper manner, stepper motor rotated in clockwise directions and anti-clockwise direction continuously,

Conclusion: The stepper motor is driven by the digital inputs and it controls the speed & the direction of the motor.

Program for practice: Write a program to rotate the stepper motor by the displacement equal to ten steps.

Experiment No.05
ANALOG TO Digital Converter

Aim: To write a program for conversion of analog data to digital output.

Apparatus:
1. 8086 Trainer.
2. A/D interface module
3. Desktop Computer

Objective: Interfacing an ADC & and acquiring the analog signal & observing the digital equivalent

Theory:

WORKING OF ANALOG TO DIGITAL CONVERTER CIRCUIT:

The working of the circuit starts with making ALE and OE pins high which are meant to choose the channel and also enable output. 5 V was the default reference voltage and it can be altered by feeding the voltage of our desire to the pins Vref+ and Vref-. The Channel selection should be done by using the pin ADDA to ADDC pins and here in this circuit diagram input channel 1 was selected. The below table will give the logic states of all pins and their respective channel selection.

SELECTED	ADDRESS LINE		
ANALOG CHANNEL	C	B	A
IN0	L	L	L
IN1	L	L	H
IN2	L	H	L
IN3	L	H	H
IN4	H	L	L
IN5	H	L	H
IN6	H	H	L
IN7	H	H	H

Now the Analog signal is fed into the channel you selected and then the state of the pin START should be made low from high to start for activation. And after the conversion the EOC pin goes high and it indicates the conversion is over. The EOC pin retains it low state as soon the next pulse is encountered. As you can see in the above circuit the EOC pin was connected to start pin which triggers a chain reaction resulting in continuous conversion to take place.

Finally we will get a 8 bit data from the pins OUT1 to OUT8 which can be used for further processing and display. You can even connect the LED's to these pins and visually view the output binary data, Led On indicates binary 1 and off indicates binary 0 data.

Program:

```
        ASSUME CS:CODE,DS:CODE,SS:CODE,ES:CODE

        KBINT    EQU 22H
        DISINT   EQU 23H
        CMD8255B EQU 27H
        PAB8255  EQU 21H
        PBB8255  EQU 23H
        PCB8255  EQU 25H
        DSPBUF   EQU 9E00H
CODE    SEGMENT
         ORG 400H
        MOV             AL,92H          ; Initialize port A & B as
        OUT             CMD8255B,AL ; I/p & C as o/p
LP2:    MOV     AL,07H
        OUT             PCB8255,AL      ; Select 7th channel
        OR               AL,18H
         OUT            PCB8255,AL      ; Set start of conversion
        AND             AL,0E7H                 ; Reset SOC bit(pulse)
        NOP
        OUT             PCB8255,AL
        NOP
        NOP
LP1:    IN              AL,PBB8255      ; Wait for EOC(end of
        AND             AL,01H                  ; conversion)
        JZ              LP1
        IN              AL,PCB8255
        OR              AL,20H
        OUT             PCB8255,AL              ; Set output enable
        NOP
        NOP
        IN              AL,PAB8255              ; Read ADC output
        MOV             DL,AL                   ; Store temporarily in DL
        MOV             SI,OFFSET RESMES+8000H
        MOV             DI,DSPBUF               ; Move result message
        MOV             CX,08H          ; into display buffer
        REPMOVSW

        MOV             AL,DL
        MOV             CL,04H          ; Get the upper nibble
        SHR             AL,CL
        CALL            ASCII
        MOV             DI,DSPBUF+0EH
        MOV             [DI],AL

        MOV             AL,DL
        AND             AL,0FH
        CALL            ASCII
        MOV             DI,DSPBUF+0FH
```

```
        MOV             [DI],AL
        INT             DISINT
        JMP             LP2                     ; Be in an infinite loop
RESMES:
        DB      'Digital O/P : 00'
ASCII:
        ADD             AL,30H                  ; If no. < 0AH,add 30H
        CMP             AL,3AH
        JC              RET1
        ADD             AL,07H          ; If no. >= 0AH,add 37H
RET1:
        RET

CODE    ENDS
END
```

Circuit Diagram:

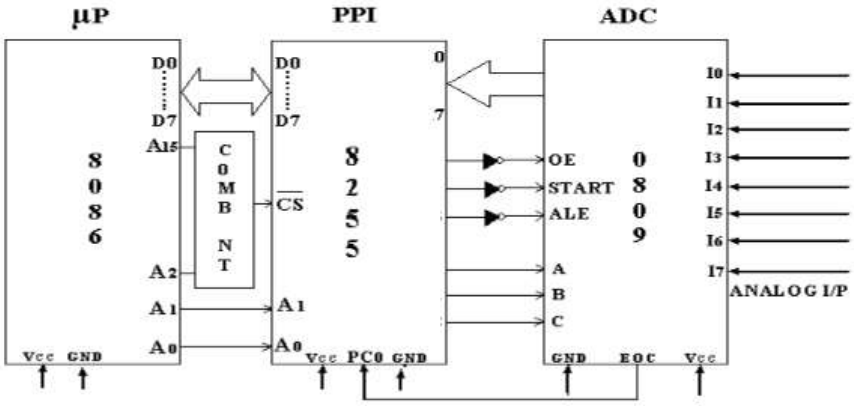


Fig: Interfacing ADC module to 8086 microprocessor through Intel 8255

Observation Table:

Analog I/P (Volts)			
Digital O/P(Hexadecimal)			

Conclusion: The ADC converted the applied Analog I/P into Digital O/P & value of the O/P is equal to the given I/P

Program for practice: Write program to save the displayed digital value in a memory location.

Experiment No.06
Block Transfer

Aim: Write a program to copy a block of 5 bytes of data from int RAM locations from address 35H to ext RAM locations from address40 60H

Objective: To understand the memory organization of 8051 microcontroller.

Apparatus/software:

1. 8051 microcontroller kit/KEIL IDE software
2. Desktop Computer

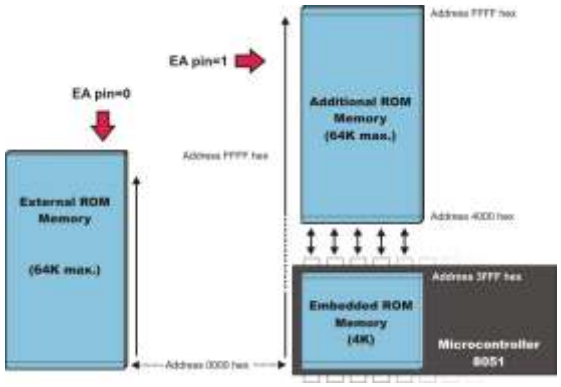
Theory:

8051 Memory Organization

The 8051 microcontroller's memory is divided into Program Memory and Data Memory. Program Memory (ROM) is used for permanent saving program being executed, while Data Memory (RAM) is used for temporarily storing and keeping intermediate results and variables.

Program Memory (ROM)

Program Memory (ROM) is used for permanent saving program (CODE) being executed. The memory is read only. Depending on the settings made in compiler, program memory may also used to store a constant variables. The 8051 executes programs stored in program memory only. code memory type specifier is used to refer to program memory.
8051 memory organization alows external program memory to be added. How does the microcontroller handle external memory depends on the pin EA logical state.



Internal Data Memory

Up to 256 bytes of internal data memory are available depending on the 8051 derivative. Locations available to the user occupy addressing space from 0 to 7Fh, i.e. first 128 registers and this part of RAM is divided in several blocks. The first 128 bytes of internal data memory are both directly and indirectly addressable. The upper 128 bytes of data memory (from 0x80 to 0xFF) can be addressed only indirectly.

External Data Memory

Access to external memory is slower than access to internal data memory. There may be up to 64K Bytes of external data memory. Several 8051 devices provide on-chip XRAM space that is accessed with the same instructions as the traditional external data space. This XRAM space is typically enabled via proper setting of SFR register and overlaps the external memory space. Setting of that register must be manualy done in code, before any access to external memory or XRAM space is made.

Source block:[35H]= 12H
[36H]= 54 H
[37H]=56 H
[38H]= 23 H
[39H]= A6 H

Program:

```
        ORG     0000H
        MOV     R0,#35H         ;R0 source pointer

        MOV     DPTR,#4060H ;DPTR as destination
                                ; pointer
        MOV     R3,#05H         ;R3 counter

BACK:   MOV     A,@R0           ;get a byte from source

        MOVX    @DPTR,A             ;copy it to destination

        INC     R0              ;increment source
                                ; pointer
        INC     DPTR            ;increment destination
                                ; pointer
        DJNZ    R3,BACK             ;keep doing for ten
                                    ;bytesLP:
        SJMP    LP              ; Be in the loop
        END
```

Observed result: Destination block:
[4060H]= 12H
[4061H]= 54 H
[4062H]=56 H
[4063H]= 23 H
[4064H]= A6 H

Conclusion: The given block is successfully transferred to the specified destination.

Program for practice: Write & execute the program for block transfer from External program meory to Register bank 1

Experiment No.07

Division of 8 bit numbers

Aim: Write a program for division of given 8 bit numbers .Store the quotient in R0 and reminder in R1

Apparatus/software:

1. 8051 microcontroller kit/KEIL IDE software
2. Desktop Computer

Objective: To study & implementing the multiplication & division operations by 8051 microcontroller.

Theory:

Multiplication of unsigned numbers:

The 8051 supports byte-by-byte multiplication only. The bytes are assumed to be unsigned data. The syntax is as follows:

```
MUL AB    ;A x B, place 16-bit result in B and A
```

In byte-by-byte multiplication, one of the operands must be in register A, and the second operand must be in register B. After multiplication, the result is in the A and B registers; the lower byte is in A, and the upper byte is in B. The following example multiplies 25H by 65H. The result is a 16-bit data that is held by the A and B registers.

Multiplication	Operand 1	Operand 2	Result
byte × byte	A	B	A = low byte, B = high byte

```
MOV  A,#25H    ;load 25H to reg. A
MOV  B,#65H    ;load 65H in reg. B
MUL  AB        ;25H * 65H = E99 where
               ;B = 0EH and A = 99H
```

Division of unsigned numbers:

In the division of unsigned numbers, the 8051 supports byte over byte only. The syntax is as follows.

```
DIV AB    ;divide A by B
```

When dividing a byte by a byte, the numerator must be in register A and the denominator must be in B. After the DIV instruction is performed, the quotient is in A and the remainder is in B. See the following

example.

Division	Numerator	Denominator	Quotient	Remainder
byte / byte	A	B	A	B

(If B = 0, then OV = 1 indicating an error)

```
MOV  A,#95     ;load 95 into A
MOV  B,#10     ;load 10 into B
DIV  AB        ;now A = 09 (quotient) and
               ;B = 05(remainder)
```

Given 8 bit numbers: Dividend: 2B H
Diviser: 0A H

Expected Result: Quotient R0=04 H Reminders R1=03 H

Program:

```
        ORG   0000H
        MOV   A, #2B H          ; load the Dividend into A
        MOV   B, #0A H          ; load the Divisor into B
        DIV   AB                ; perform the division A / B
        MOV   R0, A             ; save the Quotient in R0
        MOV   R1, B             ; save the Reminder  in R1
LP:     SJMP  LP                ; Be in the loop
        END
```

Observed Result:
Quotient R0=04 H Reminders R1=03 H

Conclusion:Division operation is done & the results are found to be correct.

Program for practice: Write & execute the program for multiplication of two 8 bit numbers.

Experiment No.08
Data Monitoring

Aim: Write a program to read the data on port 1 and send it to port 2

Apparatus/software:

1. 8051 microcontroller kit/KEIL IDE software
2. Desktop Computer

Objective: To study operation & applications of ports of 8051 microcontroller.

Theory:

8051 Microcontroller ports:

There are four ports P0, P1, P2 and P3 each use 8 pins, making them 8-bit ports. All the ports upon RESET are configured as output, ready to be used as output ports. To use any of these ports as an input port, it must be programmed.

Port 0: Port 0 occupies a total of 8 pins (pins 32-39) .It can be used for input or output. To use the pins of port 0 as both input and output ports, each pin must be connected externally to a 10K ohm pull-up resistor. This is due to the fact that P0 is an open drain, unlike P1, P2, and P3.Open drain is a term used for MOS chips in the same way that open collector is used for TTL chips. With external pull-up resistors connected upon reset, port 0 is configured as an output port.

Port 0 as Input : With resistors connected to port 0, in order to make it an input, the port must be programmed by writing 1 to all the bits. In the following code, port 0 is configured first as an input port by writing 1's to it, and then data is received from the port and sent to P1.

Dual role of port 0: Port 0 is also designated as AD0-AD7, allowing it to be used for both address and data. When connecting an 8051/31 to an external memory, port 0 provides both address and data. The 8051 multiplexes address and data through port 0 to save pins.

Port 1: Port 1 occupies a total of 8 pins (pins 1 through 8). It can be used as input or output. In contrast to port 0, this port does not need any pull-up resistors since it already has pull-up resistors internally. Upon reset, Port 1 is configured as an output port.

Port 1 as input: To make port1 an input port, it must programmed as such by writing 1 to all its bits

Port 2 : Port 2 occupies a total of 8 pins (pins 21- 28). It can be used as input or output. Just like P1, P2 does not need any pull-up resistors since it already has pull-up resistors internally. Upon reset,Port 2 is configured as an output port.

Port 2 as input : To make port 2 an input, it must programmed as such by writing 1 to all its bits

Dual role of port 2 : Port 2 is also designed as A8-A15, indicating the dual function. Since an 8051 is capable of accessing 64K bytes of external memory, it needs a path for the 16 bits of the address. While P0 provides the lower 8 bits via A0-A7, it is the job of P2 to provide bits A8-A15 of the address. In other words, when 8031 is connected to external memory, P2 is used for the upper 8 bits of the 16 bit address, and it cannot be used for I/O.

Port 3 : Port 3 occupies a total of 8 pins, pins 10 through 17. It can be used as input or output. P3 does not need any pull-up resistors, the same as P1 and P2 did not. Although port 3 is configured as an output port upon reset. Port 3 has the additional function of providing some extremely important signals such as interrupts

PORT 3 ALTERNATE FUNCTIONS :

P3 BIT	FUNCTION	PIN
P3.0	RXD	10
P3.1	TXD	11
P3.2	$\overline{INT0}$	12
P3.3	$\overline{INT1}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{WR}$	16
P3.7	$\overline{RD}$	17

.

Program:

```
        ORG     0000H
        MOV     A, #0FFH        ; move FF H into A
        MOV     P1, A           ;program port 1 as input port by
                                ; sending FF H to port 1
BACK:   MOV     A, P1           ;get data from Port 1
        MOV     P2, A           ;send it to port 2
        SJMP    BACK            ; keep doing it
        END
```

Observation: The data on the port 1 is read and sent to port 2

Conclusion: The status of the port 1 is monitored on port2 continuously

Program for practice: Write a program for monitoring the switch status connected to port 0.1 pin on port 0.1 pin continuously.

Experiment No.09
Generation of square wave

Aim: Write a program to generate a square wave on p1.5 pin.Use TIMER 0 to generate delay

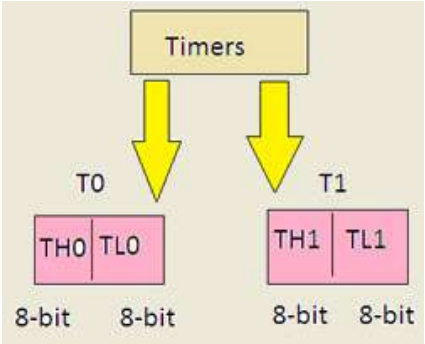
Apparatus/software:

1. 8051 microcontroller kit/KEIL IDE software
2. Desktop Computer
3.

Objective: Application of on chip Timer for generating square wave.

Theory:

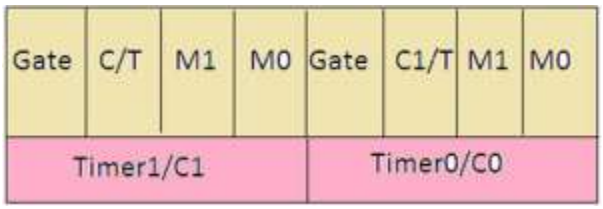
Timers in 8051 Microcontroller: There are two 16-bit timers and counters in 8051 microcontroller: timer 0 and timer 1. Both timers consist of 16-bit register in which the lower byte is stored in TL and the higher byte is stored in TH. Timer can be used as a counter as well as for timing operation that depends on the source of clock pulses to counters.



Timers and counters

Counters and Timers in 8051 microcontroller contain two special function registers: TMOD (Timer Mode Register) and TCON (Timer Control Register), which are used for activating and configuring timers and counters.

Timer Mode Control (TMOD): TMOD is an 8-bit register used for selecting timer or counter and mode of timers. Lower 4-bits are used for control operation of timer 0 or counter0, and remaining 4-bits are used for control operation of timer1 or counter1.This register is present in SFR register, the address for SFR register is 89th.



Timer Mode Control (TMOD)

Gate: If the gate bit is set to '0', then we can start and stop the "software" timer in the same way. If the gate is set to '1', then we can perform hardware timer.

C/T: If the C/T bit is '1', then it is acting as a counter mode, and similarly when set C+ =/T bit is '0'; it is acting as a timer mode.

M0	M1	Mode	Timer Pulses
0	0	M0	13-bit-2^13-8192
0	1	M1	16-bit-2^16-65535 pulses
1	0	M2	8-bit-autoreload mode-2^8= 256 pulses
1	1	M3	Split mode(load the values in T0 automatically start the T1

Mode Selection Bits

Mode select bits: The M1 and M0 are mode select bits, which are used to select the timer operations. There are four modes to operate the timers.

Mode 0: This is a 13-bit mode that means the timer operation completes with "8192" pulses.

Mode 1: This is a16-bit mode, which means the timer operation completes with maximum clock pulses that "65535".

Mode 2: This mode is an 8-bit auto reload mode, which means the timer operation completes with only "256" clock pulses.

Mode 3: This mode is a split-timer mode, which means the loading values in T0 and automatically starts the T1.

Program:

```
        ORG     0000H
        MOV TMOD,#01        ;Timer 0, mode 1(16-bit mode)
HERE:   CPL P1.5                    ;Toggle P1.5
        ACALL DELAY         ;Call delay subroutine
        SJMP HERE           ;Repeat the loop
DELAY:  MOV TL0,#00H                ;TL0=00H, the low byte
```

```
        MOV TH0,#0FFH       ;TH0=FFH, the high byte
        SETB TR0                    ;Start the timer 0
AGAIN:  JNB TF0,AGAIN               ;Monitor timer0 oerflow  flag for  1
        CLR TF0                     ;Clear timer0 oerflow  flag
        RET                         ;Return to main program
        END
```

Observed result: As expected a square wave is generated on p1.5

Conclusion: The required delay is generated using Timer & square wave is generated.

Program for practice: Write & execute the program for square wave generation without using Timer.

Experiment No.10
Data transmission using serial port

Aim:Write a program for the 8051 to transfer COE serially at 9600 baud rate.

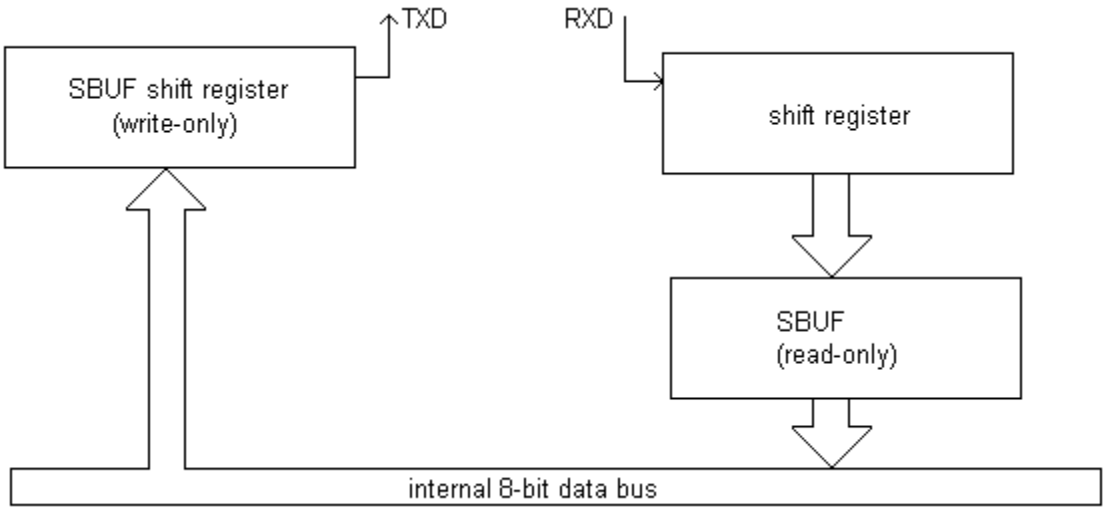
Apparatus/software:

1. 8051 microcontroller kit/KEIL IDE software

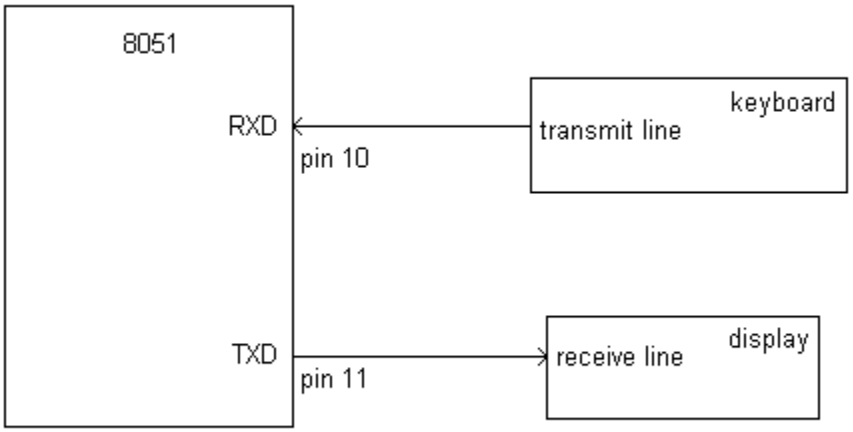
Objective: To study the 8051 serial port operation

Theory:The 8051 Serial Port

The 8051 includes an on-chip serial port that can be programmed to operate in one of four ifferent modes and at a range of frequencies. In serial communication the data is rate is known as the baud rate, which simply means the number of bits transmitted per second. In the serial port modes that allow variable baud rates, this baud rate is set by timer 1.



The 8051 serial port is full duplex. In other words, it can transmit and receive data at the same time. The block diagram above shows how this is achieved. If you look at the memory map you will notice at location 99H the serial buffer special function register (SBUF). Unlike any other register in the 8051, SBUF is in fact two distinct registers - the write-only register and the read-only register. Transmitted data is sent out from the write-only register while received data is stored in the read-only register. There are two separate data lines, one for transmission (TXD) and one for reception (RXD). Therefore, the serial port can be transmitting data down the TXD line while it is at the same time receiving data on the RXD line.



The TXD line is pin 11 of the microcontroller (P3.1) while the RXD line is on pin 10 (P3.0). Therefore, external access to the serial port is achieved by connecting to these pins. For example, if you wanted to connect a keyboard to the serial port you would connect the transmit line of the keyboard to pin 10 of the 8051. If you wanted to connect a display to the serial port you would connect the receive line of the display to pin 11 of the 8051. This is detailed in the diagram below.

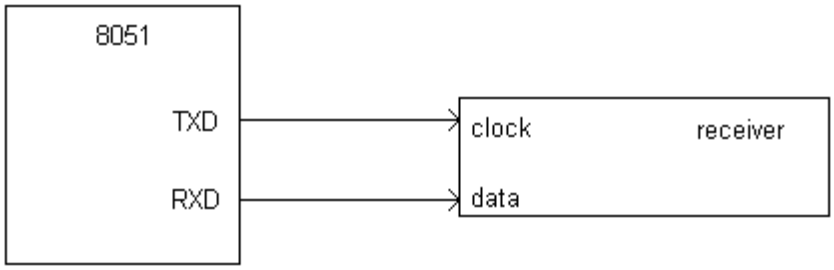
Transmitting and Receiving Data

Essentially, the job of the serial port is to change parallel data into serial data for transmission and to change received serial data into parallel data for use within the microcontroller.

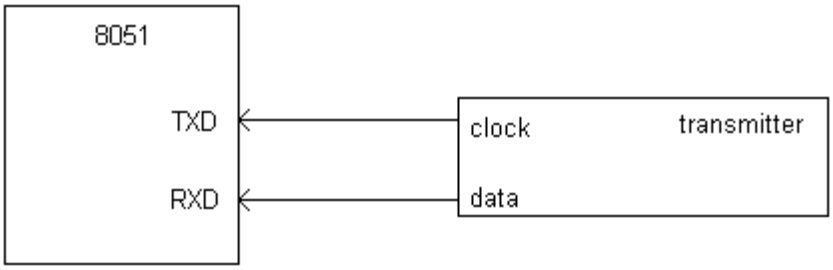
- Serial transmission is changing parallel data to serial data.
- Serial reception is changing serial data into parallel data.
- Both are achieved through the use of shift registers.

Mode 0

As detailed in the table above, mode 0 is simply a shift register. To put the serial port into mode 0 you clear both SM0 and SM1. The diagram below illustrates the serial port in mode 0.



8051 serial port in mode 0 - transmitting data



8051 serial port in mode 0 - receiving data

As can be seen in the diagram above, the terms TXD and RXD are misleading in mode 0. TXD serves as the clock while RXD is used for both receiving and transmitting data. In mode 0, the serial port is only half duplex; it cannot transmit and receive data at the same time because the same line (RXD) is being used for both transmit and receive.

The serial port in mode 0 is an example of synchronous communication; the clock signal is sent with the data on the TXD line. TXD pulses at the same frequency as the machine cycle. In other words, TXD runs at 1/12th the frequency of the system clock. If we are using a 12MHz system clock, then the frequency of TXD is 1MHz, which implies its cycle length is 1us. Therefore, each bit is active on the RXD line for 1us. To shift the entire 8-bit word along RXD takes 8us.

Program:

```
ORG     0000H
MOV     TMOD,#20H       ;timer 1,mode 2
MOV     TH1,#0FDH       ;9600 baud rate
MOV     SCON,#40H       ;8-bit, 1 stop, REN enabled
SETB    TR1                             ;start timer 1
MOV     A,#"M"
CALL    TRANS
MOV     A,#"."
CALL    TRANS
MOV     A,#"S"
CALL    TRANS
```

```
        MOV     A,#"c"
        CALL    TRANS
LOOP:   SJMP    LOOP

TRANS:  MOV     SBUF,A          ;letter “A” to transfer
HERE:   JNB     TI,HERE                 ;wait for the last bit
        CLR     TI              ;clear TI for next char
        RET
        END
```

Observed result: The data is transferred to system console

Conclusion: The data transmission is carried out successfully through serial port of 80510 controllers.
Program for practice:

Program for practice: Write a program to send the message “COE OSMANABAD” continuously with baud rate of 4800.

4. Quiz on the Subject

1.8086 is how many bit processors?
2. How many address lines 8086 processor has?
3. What is the function of Execution Unit?
4. What do you mean by memory segment?
5. Flag register of 8086 processor contains how many flags?
6. SP register contents represent what?
10. Physical address is of how many bits?
11. DAA stands for what?
12. How many segment registers 8086 processor has?
13. Name hardware interrupts of 8086 processor.
14. Assemble directive ENDS stands for?
15. How many port 8255 PPI can provide?
16.0808 is how many bit ADC?
17. How many ports lines are needed to interface Stepper Motor?
18. What is the use of Assembler ?
20. When we use the instruction MOVSB ?
21.80386 is how many bit processor?
22.2764 memory IC capacity is how much?
23. BHE stands for what?
24. Name the modes of 8086 operation.
25. In string operations which register is used to specify the string lenth ?
26. What do you mean by Microcontroller?
28.8051 is how many bit controller?
28. List some SFRs in 8051 microcontroller.
29. How much Program memeory can be externally interfaced to 8051?
30. How many ports 8051 has got internally?
31. What is the difference between Timer & Counter?
32. List External interrupts in 8051 microcontroller.
33. What is the role of SBUF register in serial communication.
34. How much internal Data memory 8051 has built in ?
35. A register bank contains how many registers?
36. Give the memory map of internal program memory of 8051 .
37. How many Timers the 8051 has in built?
38. How much Program memeory can be externally interfaced to 8051?
    a. 16KB b. 32KB c. 64KB d. 128KB
39. How many ports 8051 has got internally?
40. How many internal interrupts 8051 microcontroller has ?
41. SBUF register is related to which peripheral.
42. How much internal Data memory 8051 has built in ?
43. A register bank of 8051 contains how many registers?
44. Which register contains the flags in 8051 controller.
45. Which of the 8051 register is 16 bit length?

5. Conduction of VIVA-VOCE Examinations:

Teacher should conduct oral exams of the students with full preparation. Normally the objective questions with guess are to be avoided. To make it meaningful, the questions should be such that depth of the student in the subject is tested. Oral Exams are to be conducted in co-cordial situation. Teachers taking oral exams should not have ill thoughts about each other & courtesies should be offered to each other in case of opinion, which should be critically suppressed in front of the students.

6. Evaluation and marking system:

Basic honesty in the evaluation and marking system is essential and in the process impartial nature of the evaluator is required in the exam system. It is a primary responsibility of the teacher to see that right students who really put their effort & intelligence are correctly awarded.

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