

**Geethanjali College of Engineering and Technology  
Cheeryal (v), Keesara (M), Ranga Reddy District.**

**Microprocessor and Microcontrollers  
Laboratory Student Manual  
For  
III ECE- II SEM**



**DEPARTMENT OF  
ELECTRONICS & COMMUNICATOIN ENGINEERING  
2015-2016**

**INCHARGE  
(M.Laxmi)**

**HOD  
(Dr. P. Srihari)**

**GEETHANJALI COLLEGE OF ENGINEERING AND TECHNOLOGY**  
**DEPARTMENT OF *Electronics and Communication Engineering***

(Name of the Subject) : Microprocessors and Microcontrollers Course file

(JNTU CODE – A60494 )

Programme : UG

Branch : ECE

Year : III

Semester : II

Version No : 2

Document : GCET

No. of pages :120

Classification status (Unrestricted / Restricted) : Unrestricted

Distribution List : Dept. Library, Dept Office, Concerned Faculty

Prepared by:

- 1) Name : M. Laxmi
- 2) Sign :
- 3) Desg : Assoc. Professor.
- 4) Date : 01-07--2014

Updated by:

- 1) Name : P.SNEHA NAGA SHILPA
- 2) Sign :
- 3) Desg : Asst. Professor .
- 4) Date : 23-11-2015

Verified by :

\* For Q.C Only.

- 1) Name :
- 2) Sign :
- 3) Desg :
- 4) Date :

- 1) Name :
- 2) Sign :
- 3) Desg :
- 4) Date :

Approved by : (HOD )

- 1) Name : Dr.P.Srihari
- 2) Sign :
- 3) Date :

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY  
HYDERABAD****III year B.Tech. ECE- II SEM****L T/P/D C****0 -/3/- 2****(A60494)MICROPROCESSORS AND MICROCONTROLLERS LAB****List of Experiments**

The following programs/experiments are written for assembler and execute the same with 8086 and 8051 kits

1. Programs for 16 bit arithmetic operations for 8086 (using various addressing modes)
2. Program for sorting an array for 8086
3. Program for searching for a number or character in a string for 8086
4. Program for String manipulations for 8086
5. Program for digital clock design using 8086.
6. Interfacing ADC and DAC to 8086.
7. Parallel communication between two microprocessors using 8255.
8. Serial communication between two microprocessor kits using 8251.
9. Interfacing to 8086 and programming to control stepper motor.
10. Programming using arithmetic, logical and bit manipulation instructions of 8051
11. Program and verify Timer/Counter in 8051.
12. Program and verify interrupt handling in 8051.
13. UART operation in 8051.
14. Communication between 8051 kit and PC.
15. Interfacing LCD to 8051.
16. Interfacing matrix or keyboard to 8051.
17. Data transfer from peripheral to memory through DMA controller 8237/8257

Note: Minimum of 12 experiments to be conducted.

## **The Mission of the institute**

Our mission is to become a high quality premier educational institution, to create technocrats, by ensuring excellence, through enriched knowledge, creativity and self development.

## **The Vision of the institute**

Geethanjali visualizes dissemination of knowledge and skills to students, who would eventually contribute to the well being of the people of the nation and global community.

## **Vision of the Department**

To impart quality technical education in Electronics and Communication Engineering emphasizing analysis, design/synthesis and evaluation of hardware/embedded software using various Electronic Design Automation (EDA) tools with accent on creativity, innovation and research thereby producing competent engineers who can meet global challenges with societal commitment.

## **Mission of the Department**

- i. To impart quality education in fundamentals of basic sciences, mathematics, electronics and communication engineering through innovative teaching-learning processes.
- ii. To facilitate Graduates define, design, and solve engineering problems in the field of Electronics and Communication Engineering using various Electronic Design Automation (EDA) tools.
- iii. To encourage research culture among faculty and students thereby facilitating them to be creative and innovative through constant interaction with R & D organizations and Industry.
- iv. To inculcate teamwork, imbibe leadership qualities, professional ethics and social responsibilities in students and faculty.

**Program Educational Objectives of B. Tech (ECE) Program :**

- I. To prepare students with excellent comprehension of basic sciences, mathematics and engineering subjects facilitating them to gain employment or pursue postgraduate studies with an appreciation for lifelong learning.
- II. To train students with problem solving capabilities such as analysis and design with adequate practical skills wherein they demonstrate creativity and innovation that would enable them to develop state of the art equipment and technologies of multidisciplinary nature for societal development.
- III. To inculcate positive attitude, professional ethics, effective communication and interpersonal skills which would facilitate them to succeed in the chosen profession exhibiting creativity and innovation through research and development both as team member and as well as leader.

**Program Outcomes of B.Tech ECE Program:**

1. An ability to apply knowledge of Mathematics, Science, and Engineering to solve complex engineering problems of Electronics and Communication Engineering systems.
2. An ability to model, simulate and design Electronics and Communication Engineering systems, conduct experiments, as well as analyze and interpret data and prepare a report with conclusions.
3. An ability to design an Electronics and Communication Engineering system, component, or process to meet desired needs within the realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.
4. An ability to function on multidisciplinary teams involving interpersonal skills.
5. An ability to identify, formulate and solve engineering problems of multidisciplinary nature.
6. An understanding of professional and ethical responsibilities involved in the practice of Electronics and Communication Engineering profession.
7. An ability to communicate effectively with a range of audience on complex engineering problems of multidisciplinary nature both in oral and written form.

8. The broad education necessary to understand the impact of engineering solutions in a global, economic, environmental and societal context.
9. A recognition of the need for, and an ability to engage in life-long learning and acquire the capability for the same.
10. A knowledge of contemporary issues involved in the practice of Electronics and Communication Engineering profession
11. An ability to use the techniques, skills and modern engineering tools necessary for engineering practice.
12. An ability to use modern Electronic Design Automation (EDA) tools, software and electronic equipment to analyze, synthesize and evaluate Electronics and Communication Engineering systems for multidisciplinary tasks.
13. Apply engineering and project management principles to one's own work and also to manage projects of multidisciplinary nature.

**Course Overview:**

Microelectronics is increasingly pervading all aspects of industry, education and the home. A leading example of microelectronic techniques is the microprocessor, and as its use increases the need for knowledge and understanding will also grow. The microprocessor lab was designed to give an overview over the programming of such a microprocessor system. The students will write and debug assembly language programs using the Microsoft Macro Assembler (TASM)/Turbo Assembler(TASM). This Lab provides students with the opportunity to gain experience in microprocessor-based system design, assembly language programming, and I/O interfacing to microprocessors.

**Course Outcomes:**

After completing this course, the student will be able to:

1. Apply the fundamentals of assembly level programming of microprocessors.
2. Build a program on a microprocessor using instruction set of 8086.

3. Summarize the concepts of Assembly level language programming and its applications.
4. Develop the assembly level programming using 8086 instruction set.
5. Analyze abstract problems and apply a combination of hardware and software to address the problem
6. Contrast how different I/O devices can be interfaced to processor and will explore several techniques of interfacing.
7. Experiment with standard microprocessor interfaces including GPIO, serial ports, digital-to-analog converters and analog-to-digital converters;
8. Make use of standard test and measurement equipment to evaluate digital interfaces.

**INSTRUCTIONS TO THE STUDENTS:**

1. Students are required to attend all labs.
2. Students will work in a group of two in hardware laboratories and individually in computer laboratories.
3. While coming to the lab bring the lab manual cum observation book, record etc.
4. Take only the lab manual, calculator (if needed) and a pen or pencil to the work area.
5. Before coming to the lab, prepare the prelab questions. Read through the lab experiment to familiarize yourself with the components and assembly sequence.
6. Utilize 3 hours time properly to perform the experiment and noting down the readings. Do the calculations, draw the graph and take signature from the instructor.
7. If the experiment is not completed in the prescribed time, the pending work has to be done in the leisure hour or extended hours.
8. You will be expected to submit the completed record book according to the deadlines set up by your instructor.

9. For practical subjects there shall be a continuous evaluation during the semester for 25 sessional marks and 50 end examination marks.
10. Of the 25 marks for internal, 15 marks shall be awarded for day-to-day work and 10 marks to be awarded by conducting an internal laboratory test.

**INSTRUCTIONS TO LABORATORY TEACHERS:**

1. Observation book and lab records submitted for the lab work are to be checked and signed before the next lab session.
2. Students should be instructed to switch ON the power supply after the connections are checked by the lab assistant / teacher.
3. The promptness of submission should be strictly insisted by awarding the marks accordingly.
4. Ask viva questions at the end of the experiment.
5. Do not allow students who come late to the lab class.
6. Encourage the students to do the experiments innovatively.



**MICROPROCESSORS AND MICROCONTROLLERS LAB**

**LIST OF EXPERIMENTS**

**CYCLE-I (MICROPROCESSOR PROGRAMS)**

1. Study of TASM/MASM
2. 16-bit arithmetic Operations
3. Sorting an Array
4. Searching for Character in a String
5. Sting Manipulations
6. Digital Clock Design
7. Interfacing DAC
8. Interfacing ADC

**CYCLE-II (MICROCONTROLLER PROGRAMS)**

9. Arithmetic, Logical and Bit Manipulation operations
10. Timer/Counters operations
11. Interrupt Handling
12. UART Operation
13. LCD Interfacing
14. Interfacing Matrix keyboard

**CYCLE-III (ADDITIONAL EXPERIMENTS USING KEIL)**

15. Serial Transmission from PC to 8051uc
16. Port Programming of 8051

**CYCLE-IV (DESIGN & OPEN EXPERIMENTS)**

## MICROPROCESSORS AND MICROCONTROLLERS LAB

## INDEX

SI.No.	NAME OF THE EXPERIMENT	PAGE No.
<b>CYCLE -I</b>		
0	Study of TASM/MASM	14
1	Introduction to 8086 microprocessor	19
2	16-bit arithmetic Operations	25
3	Sorting an Array	34
4	Searching for Character in a String	40
5	String Manipulations	43
6	Digital Clock Design	51
7	Interfacing DAC	53
8	Interfacing ADC	58
9	Serial communication between two 8086 microprocessors	61
10	Interfacing stepper motor	65

11	Interfacing to 8086 and Programming to DMA controller	68
<b>CYCLE-II</b>		
12	Introduction to 8051	70
13	Arithmetic, Logical and Bit Manipulation operations	79
14	Timers and Counters	88
15	Interrupt Handling	91
<b>CYCLE-3 (ADDITIONAL EXPERIMENTS USING KEIL)</b>		
16	Introduction to KEIL $\mu$ vision	93
17	Serial Transmission from PC to 8051uc	99
18	Reading & writing data from/to 8051 ports	100
<b>CYCLE-4 (DESIGN &amp; OPEN EXPERIMENTS)</b>		

## INTRODUCTION TO TASM

### EDITOR:

An editor is a program, which allows you to create a file containing the assembly language statements for your program. As you type in your program, the editor stores the ASCII codes for the letters and numbers in successive RAM locations. When you have typed in all of your programs, you then save the file on a floppy or hard disk. This file is called source file. The next step is to process the source file with an assembler. In the TASM assembler, you should give your source file name the extension, .ASM

### ASSEMBLER:

An assembler program is used to translate the assembly language mnemonics for instructions to the corresponding binary codes. When you run the assembler, it reads the source file of your program from the disk, where you saved it after editing. On the first pass through the source program the assembler determines the displacement of named data items, the offset of labels and puts this information in a symbol table. On the second pass through the source program, the assembler produces the binary code for each instruction and inserts the offset etc that is calculated during the first pass. The assembler generates two files on floppy or hard disk. The first file called the object file is given the extension, .OBJ. The object file contains the binary codes for the instructions and information about the addresses of the instructions. The second file generated by the assembler is called assembler list file. The list file contains your assembly language statements, the binary codes for each instruction and the offset for each instruction. In TASM assembler, TASM source file name .ASM is used to assemble the file. Edit source file name .LST is used to view the list file, which is generated, when you assemble the file.

### LINKER:

A linker is a program used to join several object files into one large object file

and convert to an **exe** file. The linker produces a link file, which contains the binary codes for all the combined modules. The linker however doesn't assign absolute addresses to the program, it assigns is said to be relocatable because it can be put anywhere in memory to be run. In TASM, TLINK source filename is used to link the file.

### **DEBUGGER:**

A debugger is a program which allows you to load your object code program into system memory, execute the program and troubleshoot are debug it the debugger allows you to look at the contents of registers and memory locations after your program runs. It allows you to change the contents of register and memory locations and return the program. A debugger also allows you to set a break point at any point in the program. If you inset a breakpoint the debugger will run the program upto the instruction where the breakpoint is set and stop execution. You can then examine register and memory contents to see whether the results are correct at that point. In TASM, td filename is issued to debug the file.

### **DEBUGGER FUNCTIONS:**

1. Debugger allows us to look at the contents of registers and memory locations.
2. We can extend 8-bit register to 16-bit register which the help of extended register option.
3. Debugger allows us to set breakpoints at any point with the program.
4. The debugger will run the program upto the instruction where the breakpoint is set and then stop execution of program. At this point, we can examine registry and memory contents at that point.
5. With the help of dump we can view register contents.
6. We can trace the program step by step with the help of F7.
7. We can execute the program completely at a time using F8.

### **DEBUGGER COMMANDS:**

**ASSEMBLE:**

To write assembly language program from the given address

A starting address <cr>

Eg: a 100 <cr>

Starts program at an offset of 100.

**DUMP:**

To see the specified memory contents

D memory location first address last address

(While displays the set of values stored in the specified range, which is given above)

Eg: d 0100 0105 <cr>

Display the contents of memory locations from 100 to 105(including).

**ENTER:**

To enter data into the specified memory locations(s).

E memory location data data data data data ...<cr>

Eg: e 1200 10 20 30 40 ....

Enters the above values starting from memory locations 1200 to 1203, by loading 10 into 1200,20 into 1201 and soon.

**GO:**

To execute the program

G: one instruction executes (address specified by IP)

G address <cr>: executes from current IP to the address specified

G first address last addresses <cr>: executes a set of instructions specified between the given addresses

**MOVE:**

Moves a set of data from source location to destination location

M first address last address destination address

Eg: m100 104 200

Transfers block of data (from 100 to 104) to destination address 200.

**QUIT:**

To exit from the debugger.

Q <cr>

**REGISTER:**

Shows the contents of Registers

R register name

Eg: r ax

Shows the contents of register.

**TRACE:**

To trace the program instruction by instruction.

T = 0100 <cr>: traces only the current instruction. (Instruction specified by IP)

T = 0100 02 <cr>: Traces instructions from 100 to 101, here the second argument specifies the number of instructions to be traced.

**UNASSEMBLE:**

To unassemble the program.

Shows the opcodes along with the assembly language program.

**INTRODUCTION TO 8086 MICROPROCESSOR**

**8086 ARCHITECTURE:**

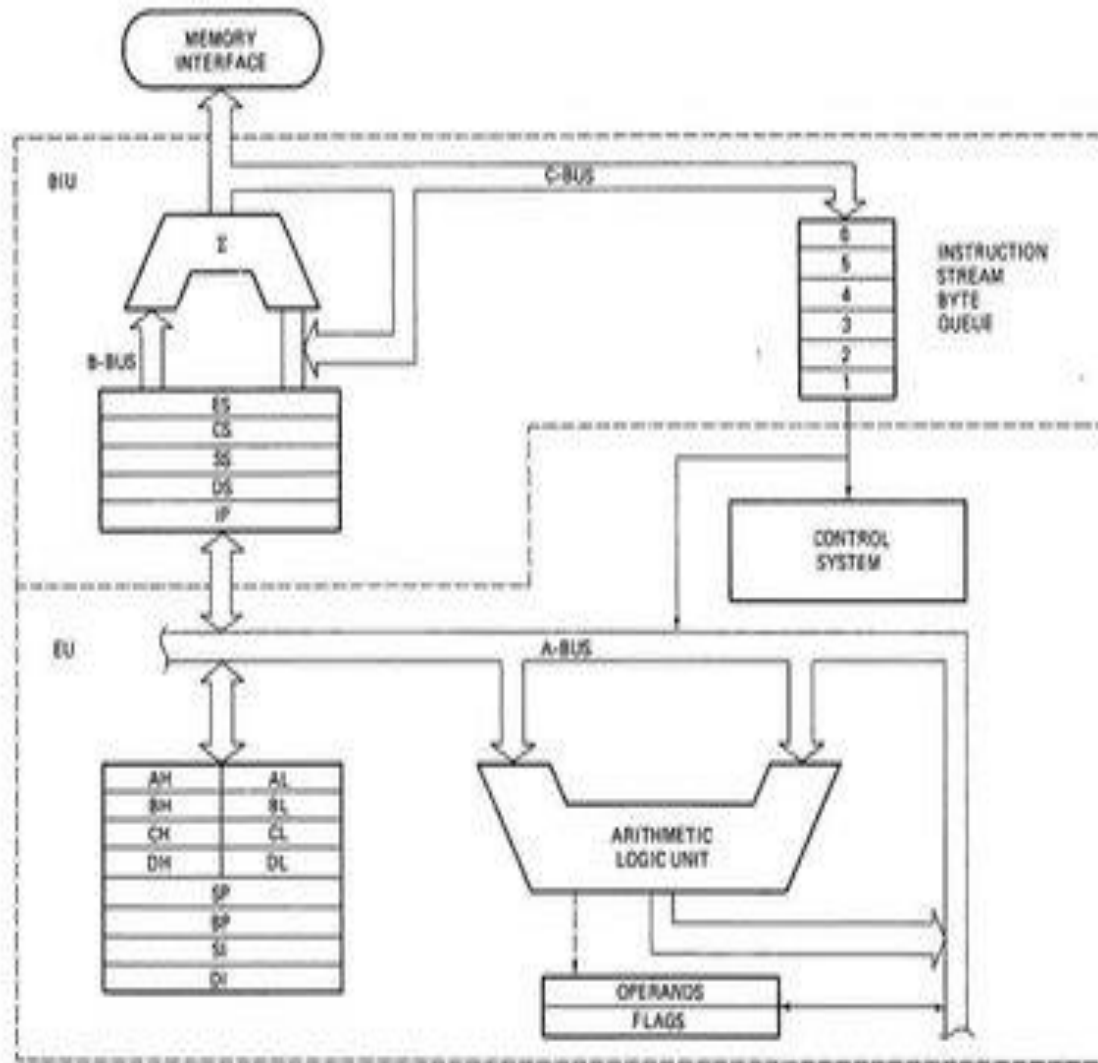
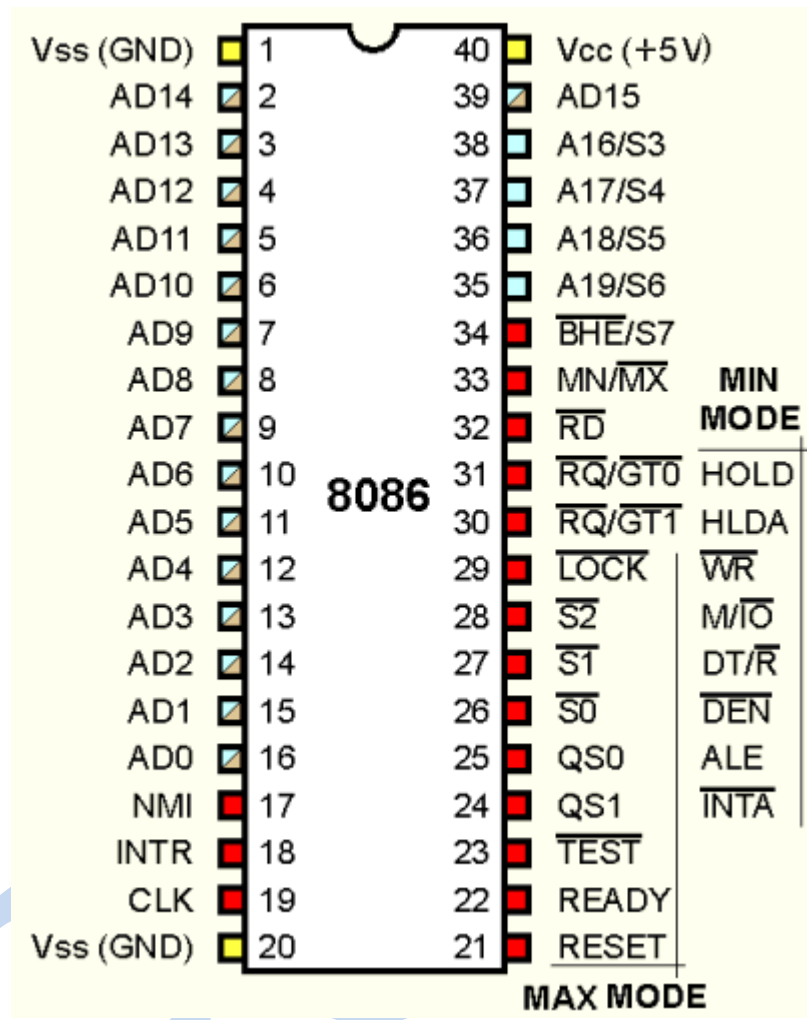


FIGURE 2-7 8086 internal block diagram. (Intel Corp.)



**PIN DIAGRAM:**



**8086 INSTRUCTION SET SUMMARY:**

The following is a brief summary of the 8086 instruction set:

**Data Transfer Instructions**

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

**Logical Instructions**

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

**Shift and Rotate Instructions**

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

**Arithmetic Instructions**

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

**Transfer Instructions**

<b>JMP</b>	:	Unconditional jump
<b>JA (JNBE)</b>	:	Jump if above (not below or equal)
<b>JAE (JNB)</b>	:	Jump if above or equal (not below)
<b>JB (JNAE)</b>	:	Jump if below (not above or equal)
<b>JBE (JNA)</b>	:	Jump if below or equal (not above)
<b>JE (JZ)</b>	:	Jump if equal (zero)
<b>JG (JNLE)</b>	:	Jump if greater (not less or equal)
<b>JGE (JNL)</b>	:	Jump if greater or equal (not less)
<b>JL (JNGE)</b>	:	Jump if less (not greater nor equal)
<b>JLE (JNG)</b>	:	Jump if less or equal (not greater)
<b>JC, JNC</b>	:	Jump if carry set, carry not set
<b>JO, JNO</b>	:	Jump if overflow, no overflow

<b>JS, JNS</b>	:	Jump if sign, no sign
<b>JNP (JPO)</b>	:	Jump if no parity (parity odd)
<b>JP (JPE)</b>	:	Jump if parity (parity even)
<b>LOOP</b>	:	Loop unconditional, count in CX
<b>LOOPE (LOOPZ)</b>	:	Loop if equal (zero), count in CX
<b>LOOPNE (LOOPNZ)</b>	:	Loop if not equal (not zero), count in CX
<b>JCXZ</b>	:	Jump if CX equals zero

### Subroutine and Interrupt Instructions

<b>CALL, RET</b>	:	Call, return from procedure
<b>INT, INTO</b>	:	Software interrupt, interrupt if overflow
<b>IRET</b>	:	Return from interrupt

### String Instructions

<b>MOVS</b>	:	Move byte or word string
<b>MOVSB, MOVSW</b>	:	Move byte, word string
<b>CMPS</b>	:	Compare byte or word string
<b>SCAS</b>	:	Scan byte or word string
<b>LODS, STOS</b>	:	Load, store byte or word string
<b>REP</b>	:	Repeat
<b>REPE, REPZ</b>	:	Repeat while equal, zero
<b>REPNE, REPNZ</b>	:	Repeat while not equal (zero)

### Processor Control Instructions

<b>STC, CLC, CMC</b>	:	Set, clear, complement carry flag
<b>STD, CLD</b>	:	Set, clear direction flag
<b>STI, CLI</b>	:	Set, clear interrupt enable flag
<b>LAHF, SAHF</b>	:	Load AH from flags, store AH into flags
<b>PUSHF, POPF</b>	:	Push flags onto stack, pop flags off stack
<b>ESC</b>	:	Escape to external processor interface
<b>LOCK</b>	:	Lock bus during next instruction
<b>NOP</b>	:	No operation (do nothing)
<b>WAIT</b>	:	Wait for signal on TEST input
<b>HLT</b>	:	Halt processor



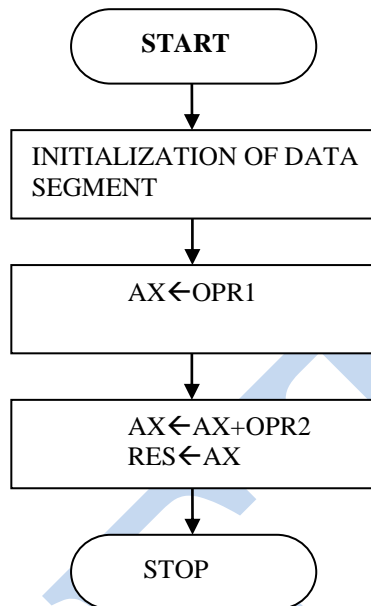
**EXPERIMENT NO.1****ARITHMETIC OPERATIONS IN 8086  
I.ADDITION OF TWO 16-BIT NUMBERS****OBJECTIVE:**

To write an assembly language program for performing addition of two 16-bit signed and unsigned numbers.

**TOOLS REQUIRED:** PC installed with TASM

**ALGORITHM:**

- Step I** : Initialize the data memory.
- Step II** : Load the first number into AX register.
- Step II** : Load the second number into BX register.
- Step IV** : Add two lower digits.
- Step V** : Adjust result to valid BCD number.
- Step VI** : Store the result in BL.
- Step VI** : Add the two upper digits with carry.
- Step VIII** : Adjust result to valid BCD number.
- Step IX** : Store the result in BH.
- Step X** : Display the result.
- Step XI** : Stop.

**FLOW CHART:****RESULT:****FLAGS:**

Before execution, c=0, s=0, z=0, o=0, p=0, a=0, i=1, d=0.

After execution, c=0, s=0, z=0, o=0, p=1, a=0, i=1, d=0.

**INPUT** : OPR1 = 4269H  
OPR2 = 1000H

**OUTPUT:** RES = 5269H

**SIGNED NUMBERS****FLAGS:**

Before execution, c=0, s=0, z=0, o=0, p=0, a=0, i=1, d=0.

After execution, c=1, s=0, z=0, o=1, p=0, a=0, i=1, d=0.

**INPUT** : OPR1 = 9763H  
OPR2 = A973H

**OUTPUT** : RES = 40D6H

**OUTCOME**

Upon completion of this experiment, the student will be able to:

1. Employ the arithmetic instructions in various programs.
2. Solve some mathematical operations by using the 8086 microprocessor

GCET

## II.SUBTRACTION OF TWO 16-BIT NUMBERS

### OBJECTIVE

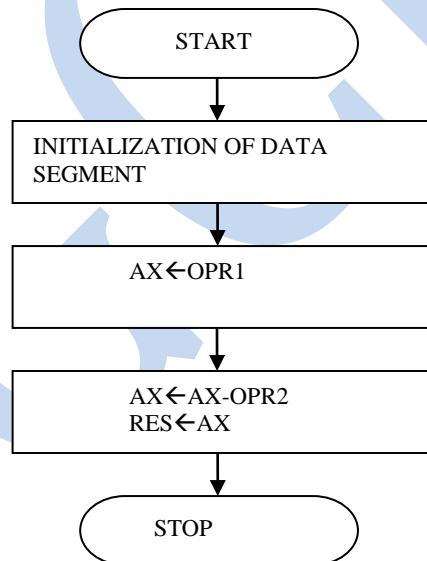
To write an assembly language program to perform subtraction of two 16-bit signed and unsigned numbers.

**TOOLS REQUIRED:** : PC installed with TASM

### ALGORITHM:

- Step I** : Initialize the data memory.  
**Step II** : Load the first number into AX register.  
**Step III** : Load the second number into BX register.  
**Step IV** : Sub AX from BX.  
**Step V** : Store result in AX  
**Step VI** : Display the result.  
**Step VII** : Stop.

### FLOW CHART:



### RESULT

#### FLAGS:

Before execution, c=0, s=0, z=0, o=0, p=0, a=0, i=1, d=0.

After execution, c=0, s=0, z=0, o=0, p=1, a=0, i=1, d=0.

**INPUT** : OPR1 = 4269H



:       OPR2 = 1000H  
OUTPU        :       RES = 3269H

**SIGNED NUMBERS**

FLAGS:

Before execution, c=0, s=0, z=0, o=0, p=0, a=0, i=1, d=0.

After execution, c=0, s=0, z=0, o=0, p=1, a=0, i=1, d=0.

INPUT           :       OPR1 = 9763H  
                  :       OPR2 = 8973H

OUTPUT       :       RES = 0DF0H

**OUTCOME:**

Upon completion of this experiment, the student will be able to:

1. Employ the arithmetic instructions in various programs.
2. Solve some mathematical operations by using the 8086 microprocessor

### III.MULTIPLICATION OF TWO 16-BIT NUMBERS

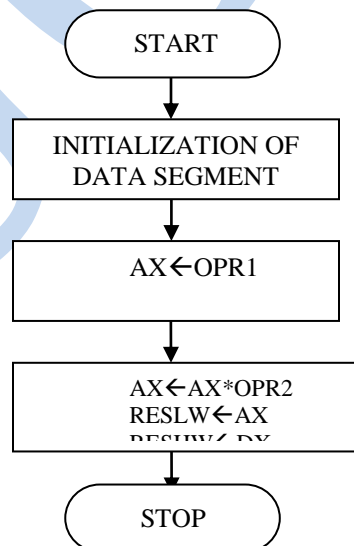
**OBJECTIVE:**

To write an assembly language program to perform multiplication of two 16-bit unsigned numbers.

**TOOLS REQUIRED:** ; PC installed with TASM

**ALGORITHM:**

- Step I** : Initialize the data memory.  
**Step II** : Load the first number into AX register.  
**Step III** : Load the second number into BX register.  
**Step IV** : Multiply AX with BX.  
**Step V** : store lower byte in accumulator.  
**Step VI** : Store Upper byte in DX register  
**Step VII** : Display the result.  
**Step VIII** : Stop.

**FLOW CHART:**

**RESULT:**

FLAGS:

Before execution, c=0, s=0, z=0, o=0, p=0, a=0, i=1, d=0.

After execution, c=1, s=0, z=0, o=1, p=0, a=0, i=1, d=0.

INPUT:        OPR1 = 2000H  
                  OPR2 = 4000H

OUTPUT:      RESLW = 0000H (AX)  
                  RESHW = 0800H (DX)

**OUTCOME:**

Upon completion of this experiment, the student will be able to:

1. Employ the arithmetic instructions in various programs.
2. Solve some mathematical operations by using the 8086 microprocessor

#### IV.DIVISION OF TWO NUMBERS

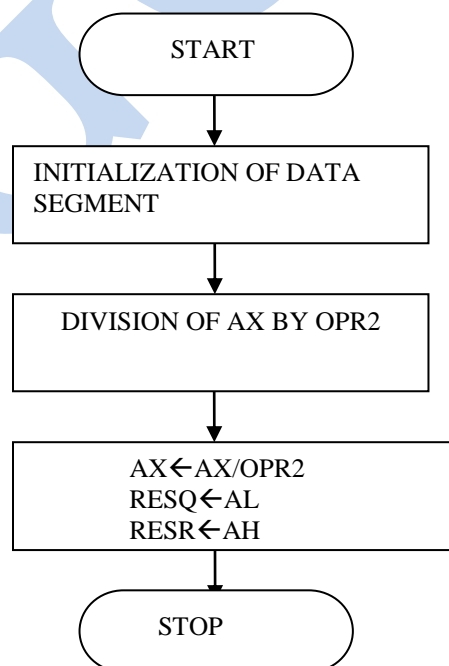
**OBJECTIVE:**

To write an assembly language program to perform division of 16-bit unsigned number by 8-bit unsigned number.

**TOOLS REQUIRED:** : PC installed with TASM

**ALGORITHM:**

- Step I** : Initialize the data memory.  
**Step II** : Load the first number into AX register.  
**Step III** : Load the second number into BX register.  
**Step IV** : Divide AX by BX.  
**Step V** : store Quotient in AL register.  
**Step VI** : Store remainder in AH register  
**Step VII** : Display the result.  
**Step VIII** : Stop.

**FLOW CHART:**

**RESULT:**

**FLAGS:**

Before execution, c=0, s=0, z=0, o=0, p=0, a=0, i=1, d=0.

After execution, c=0, s=0, z=0, o=0, p=0, a=1, i=1, d=0.

**INPUT:**

OPR1 = 2C58H (DIVIDEND)

OPR2 = 56H (DIVISOR)

**OUTPUT:**

RESQ = 84H (AL)

RESR = 00H (AH)

**VIVA QUESTIONS:**

- 1) How many bit 8086 microprocessor is?
- 2) What is the size of data bus of 8086?
- 3) What is the size of address bus of 8086?
- 4) What is the max memory addressing capacity of 8086?
- 5) Which are the basic parts of 8086?

**OUTCOME**

Upon completion of this experiment, the student will be able to:

1. Employ the arithmetic instructions in various programs.
2. Solve some mathematical operations by using the 8086 microprocessor

**EXPERIMENT NO.2****PROGRAM FOR SORTING AN ARRAY FOR 8086 MICROPROCESSOR****I.ASCENDING ORDER****OBJECTIVE:**

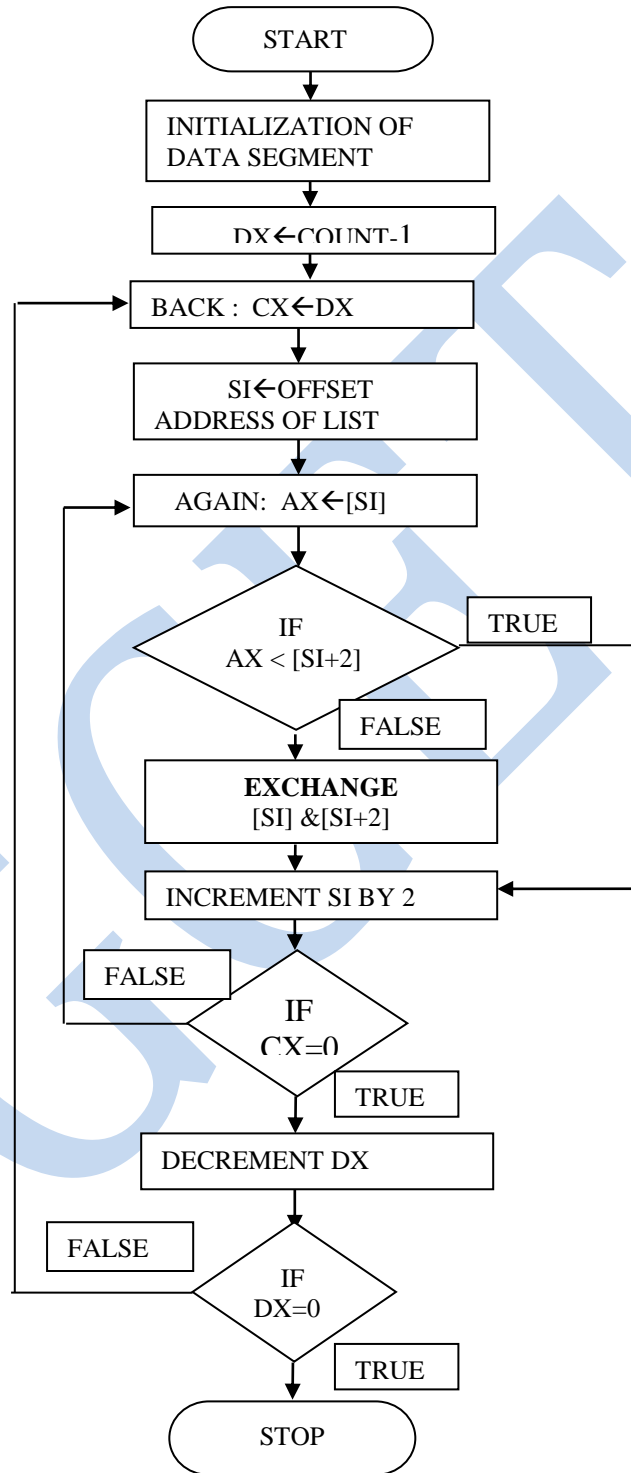
To write an assembly language program to arrange the given numbers in ascending order.

**TOOLS REQUIRED:** ; PC installed with TASM

**ALGORITHM:**

- Step I** : Initialize the number of elements counter.
- Step II** : Initialize the number of comparisons counter..
- Step III** : Compare the elements. If first element < second element goto stepVIII Else go to step V.
- Step IV** : Swap the elements..
- Step V** : Decrement the comparison counter.
- Step VI** : Is count = 0 ? if yes go to step VIII else go to step IV.
- Step VII** : Insert the number in proper position.
- Step VIII** : Increment the number of elements counter.
- Step IX** : Is count = N ? If yes, go to step XI else go to step II
- Step X** : Store the result.
- Step XI** : Stop.

**FLOW CHART:**



**RESULT:**

INPUT: LIST (DS: 0000H) = 05H,04H,01H,03H,02H

OUTPUT: LIST (DS: 0000H) = 01H,02H,03H,04H,05H

**OUTCOME:**

Upon completion of this experiment the student will be able to:

1. Demonstrate the control transfer instructions.
2. Explain number searches and differentiates bigger and smaller numbers from large database.



## II. DESCENDING ORDER

### OBJECTIVE:

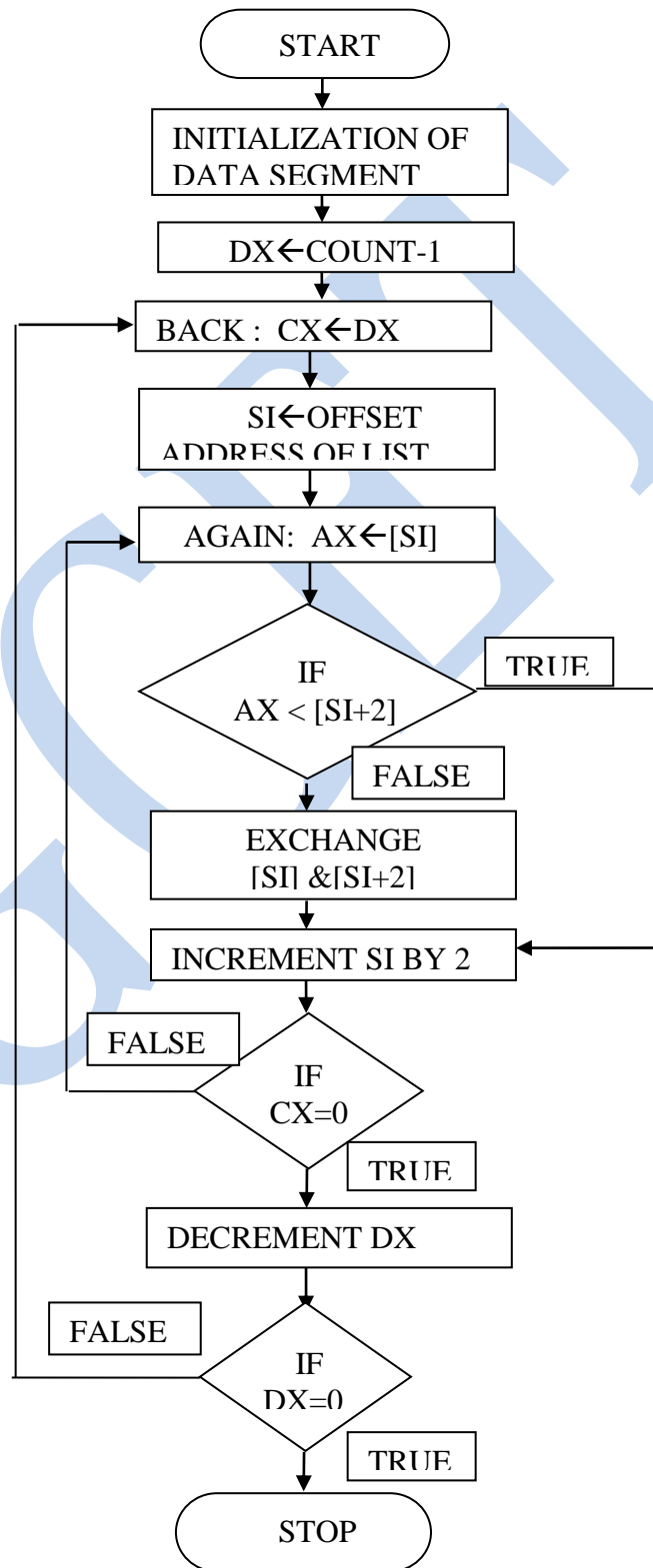
To write an assembly language program to arrange the given numbers in descending order.

**TOOLS REQUIRED:** : PC installed with TASM

### ALGORITHM:

- Step I** : Initialize the number of elements counter.
- Step II** : Initialize the number of comparisons counter..
- Step III** : Compare the elements. If first element >second element goto step VIII  
Else goto step V.
- Step IV** : Swap the elements..
- Step V** : Decrement the comparison counter.
- Step VI** : .Is count = 0 ? if yes goto step VIII else goto step IV.
- Step VII** : Insert the number in proper position.
- Step VIII** : Increment the number of elements counter.
- Step IX** : Is count = N ? If yes, goto step XI else goto step II
- Step X** : Store the result.
- Step XI** : Stop.

**FLOW CHART:**



**RESULT:**

INPUT: LIST (DS: 0000H) = 03H, 04H,01H,05H,02H

OUTPUT: LIST (DS: 0000H) = 05H, 04H, 03H,02H,01H

**VIVA QUESTIONS:**

- 1) What are the functions of BIU?
- 2) What are the functions of EU?
- 3) How many pin IC 8086 is?
- 4) What IC8086 is?
- 5) What is the size of instruction queue in 8086?

**OUTCOME:**

Upon completion of this experiment the student will be able to:

1. Demonstrate the control transfer instructions.
2. Explain number searches and differentiates bigger and smaller numbers from large database.

**EXPERIMENT NO.3**  
**PROGRAM FOR SEARCHING FOR A NUMBER/CHARACTER IN A STRING FOR**  
**8086 MICROPROCESSOR**

**SEARCHING FOR A NUMBER**

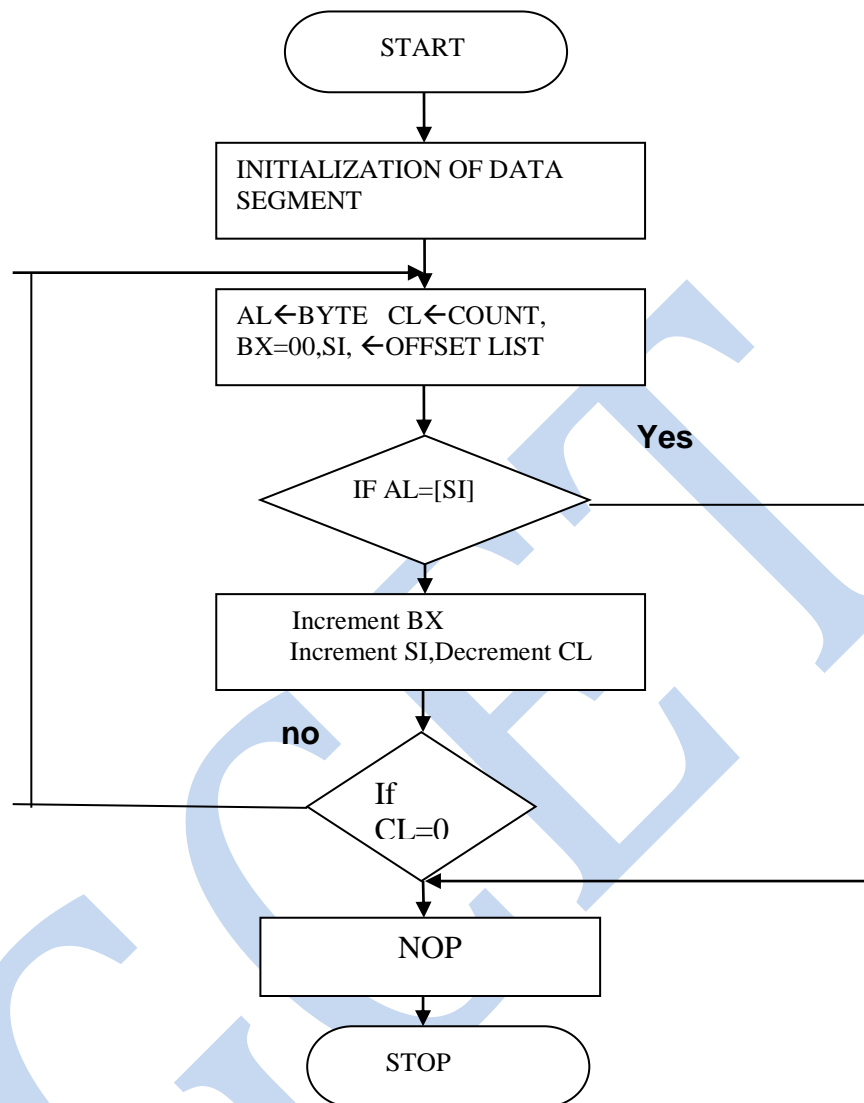
**OBJECTIVE:**

To find whether the given byte is in given string or not & find its relative address

**TOOLS REQUIRED:** : PC installed with TASM

**ALGORITHM:**

- Step I** : Initialize the extra segment (ES)
- Step II** : Initialize the start of string in the ES. (DI)
- Step III** : Move the number of elements in the string in CX register
- Step IV** : Move the byte to be searched in the AL register..
- Step V** : Scan for the byte in ES. If the byte is found ZF=0,move the address pointed by S:DI to BX
- Step VI** : Store the result
- Step VII** : Stop.

**FLOWCHART:****RESULT:****GIVEN DATA:**

N=19H,99H,45H,46H,34H

BYTE= 45H

**FLAGS:**

INITIALLY: C=0,Z=0,S=0,O=0,P=0,A=0,I=1,D=0

AFTER EXECUTION: C=0,Z=1,S=0,O=0,P=1,A=0,I=1,D=0

**OUTPUT:**

RES: 45H

ADDRESS: BX----0002

**VIVA QUESTIONS:**

- 1) What is the size of instruction queue in 8086?
- 2) Which are the registers present in 8086?
- 3) What do you mean by pipelining in 8086?
- 4) How many 16 bit registers are available in 8086?
- 5) Specify addressing modes for any instruction?

**OUTCOME:**

Upon completion of this experiment the student will be able to:

1. .Demonstrate the control transfer instructions.
2. Explain number searches and differentiates bigger and smaller numbers from large database.

**EXPERIMENT NO.4**

**PROGRAM FOR STRING MANIPULATIONS FOR 8086**

**I. LENGTH OF THE STRING**

**OBJECTIVE:**

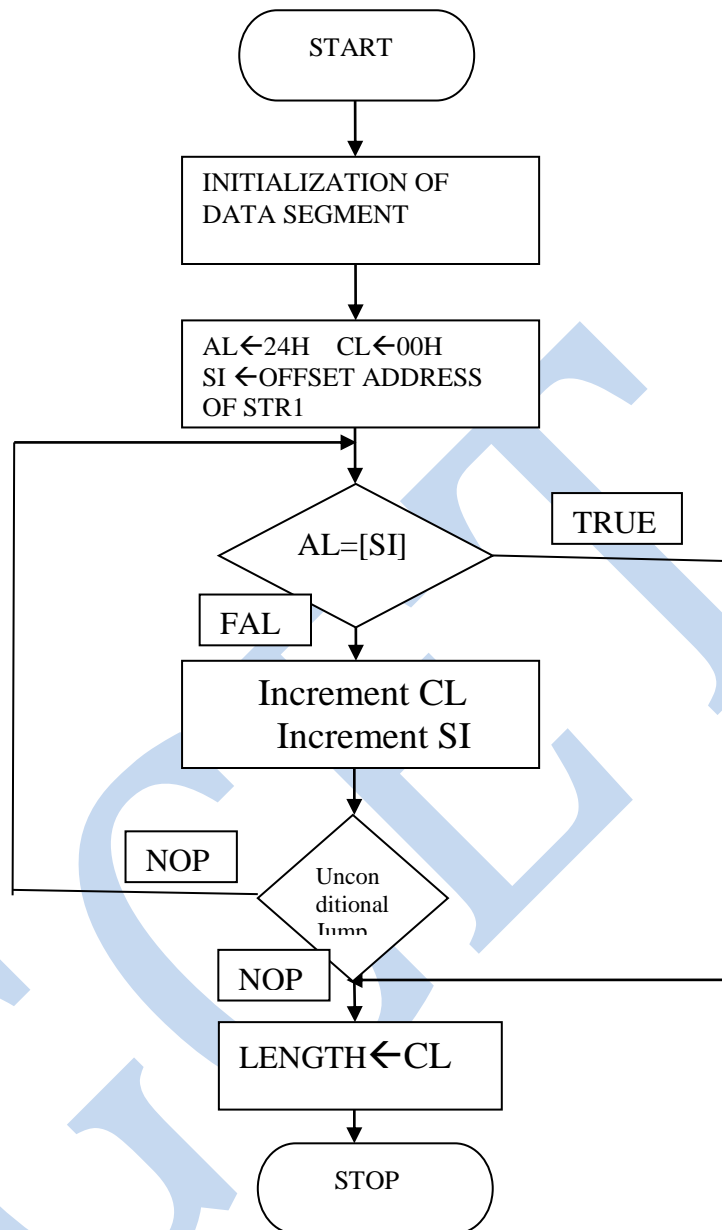
To write an assembly language program to find the length of the given string.

**TOOLS REQUIRED:** : PC installed with TASM

**ALGORITHM:**

- Step I** : Initialize the data segment(DS)
- Step II** : Initialize the code segment(CS)
- Step III** : Move the data to Accumulator(AX)
- Step IV** : Initialize the counter with 0.
- Step V** : Move string to SI register.
- Step VI** : Compare AL and SI. Until found ZF=0,increment counter.
- Step VII** : Store the result
- Step VIII** : Stop.

**FLOW CHART**





**RESULT:**

INPUT: STR (DS:0000H) = GEETHANJALI COLLEGE OF ENGINEERING AND TECHNOLOGY

OUTPUT: LENGTH =

**OUTCOME:**

Upon completion of this experiment the student will be able to:

1. Describe the memory management and select the proper memory.
2. Demonstrate the control transfer instructions

## II. DISPLAY THE STRING

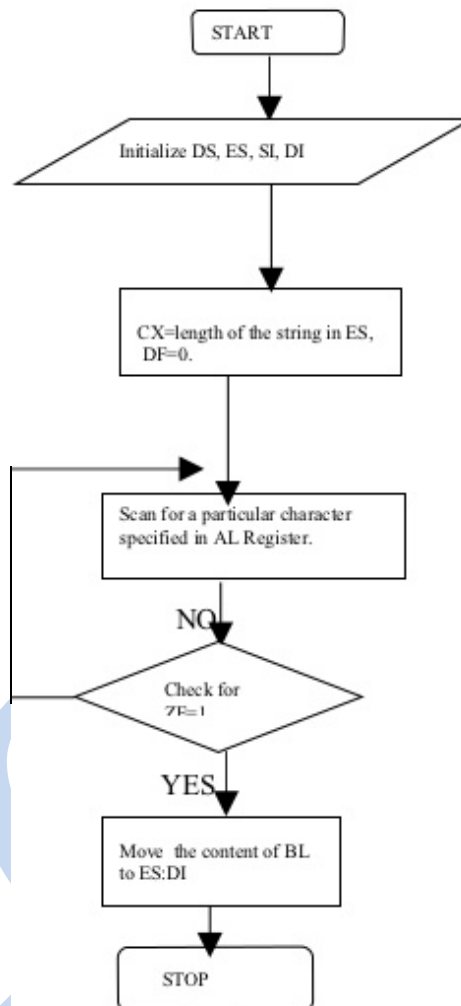
### OBJECTIVE:

To write an assembly language program to display the given string.(DOS PROGRAMMING)

TOOLS REQUIRED: PC installed with TASM

### ALGORITHM:

- Step I** : Initialize the data segment (DS)
- Step II** : Initialize the code segment (CS)
- Step III** : Move the data to Accumulator (AX)
- Step IV** : use the instruction 21H for DOS function calls.
- Step V** : Store the Offset address in DX
- Step VI** : Get the data from AL and display.
- Step VII** : Store the result
- Step VIII** : Stop.

**FLOWCHART****RESULT**

WELCOME TO MICROPROCESSORS LAB

**OUTCOME:**

Upon completion of this experiment the student will be able to:

1. Describe the memory management and select the proper memory.
2. Demonstrate the DOS Interrupts.

### III.REVERSE THE STRING

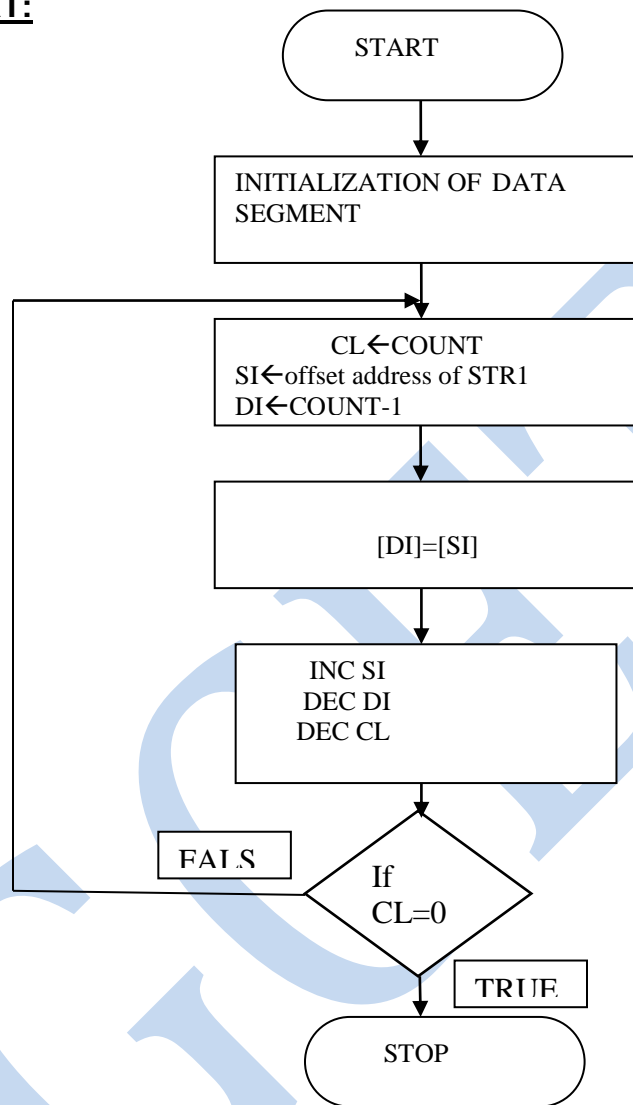
**OBJECTIVE:**

To write an assembly language program to reverse the given string.

**TOOLS REQUIRED:** ; PC installed with TASM

**ALGORITHM:**

- Step I** : Initialize the data segment(DS)
- Step II** : In the Data segment . initialize element in an array named as Src,initialize the empty array size as DS and Count the value
- Step III** : In code segment move the data segment value to data segment register
- Step IV** : Move count value (count +1) to count register and define offset address of destination to DI and move 04 H to DX
- Step V** : Define offset address of src to SI then move SI to BX and then BX to DX
- Step VI** : Decrement the destination index then subtract source index value
- Step VII** : Decrement CX if non zero go to step V
- Step VIII** : Store the result
- Step IX** : Stop

**FLOW CHART:****RESULT:**

INPUT: STR1 (DS:0000H) =

OUTPUT: STR1 (DS:0004H) =

**VIVA QUESTIOS:**

- 1) What do you mean by assembler directives?
- 2) What .model small stands for?
- 3) What is the supply requirement of 8086?
- 4) What is the relation between 8086 processor frequency & crystal Frequency?
- 5) Functions of Accumulator or AX register?

**OUTCOME:** Upon completion of this experiment the student will be able to:

1. Describe the memory management and select the proper memory.
2. Demonstrate the control transfer instructions

**EXPERIMENT NO.5****PROGRAM FOR DIGITAL CLOCK DESIGN USING 8086****PROGRAM TO DISPLAY CURRENT SYSTEM TIME****OBJECTIVE:**

To write an assembly language program to display current system time.(DOS PROGRAMMING)

**TOOLS REQUIRED:** : PC installed with TASM

**ALGORITHM:**

- Step1** : get the current system time
- Step2** : BX=offset address of the string TIME
- Step3** : PUSH AX onto the STACK,PUSH CX onto the STACK
- Step4** : set AL=CH , CH=hours
- Step5** : set [BX]=hr , [BX] is pointing to hr in the string TIME
- Step6** : set AL=CL , CL=minutes
- Step7** : set [BX+3]=min , [BX] is pointing to min in the string TIME
- Step8** : set AL=DH , DH=seconds
- Step9** : set [BX+6]=min , [BX] is pointing to sec
- Step10** : POP a value from STACK into CX
- Step11** : POP a value from STACK into AX
- Step12** : Store the result
- Step13** : Stop.

**VIVA QUESTIONS:**

- 1) Functions of BX register?
- 2) Functions of CX register?
- 3) Functions of DX register?
- 4) How Physical address is generated?
- 5) Which are pointers present in this 8086?

**OUTCOME:**

Upon completion of this experiment the student will be able to Demonstrate displaying current system time using DOS programming.



**EXPERIMENT NO.6****INTERFACING DIGITAL TO ANALOG CONVERTER TO 8086 FOR****GENERATION OF WAVE FORMS:****OBJECTIVE:**

To write a Program to generate following wave forms

- a. Ramp waveform
- b. Square waveform
- c. Step waveform
- d. Triangle waveform

**APPARATUS:** 1.ADS-SDA-86-STA kit  
2.8255 Study card  
3. Adapter, Keyboard, Cables, CRO Etc.

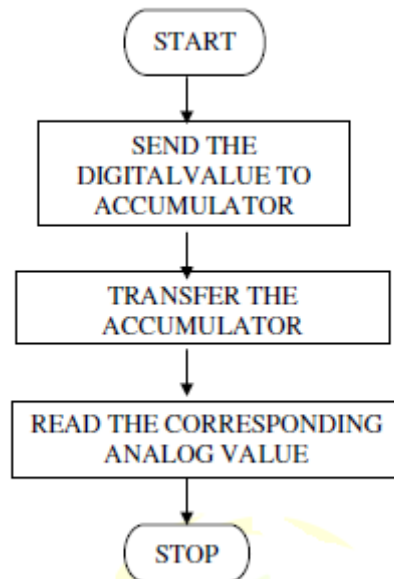
**PROCEDURE:**

1. Connect 8086 kit PC using RS232 cable.
2. Connect Power supply to 8086 kit and 8255 interfacing kit(only blue(+5v) and black(0v) lines Power cable to power supply)
3. Connect 8255 to CN4 of 8086 using 26 pin bus.
4. Connect the CRO probe to JP3 of 8255 kit
5. Keep the DIP switch in 1 & 7 on (8086kit), open TALK, and go to options select target device as 8086 and Connect.
6. Change dip switch into 1 & 5on, once reset 8086 kit.
7. Go to file →Download hex file
8. G-5000(on system keyboard), we can observe the output on 8086 kit and CRO.

**ALGORITHM:**

Measurement of analog voltage:

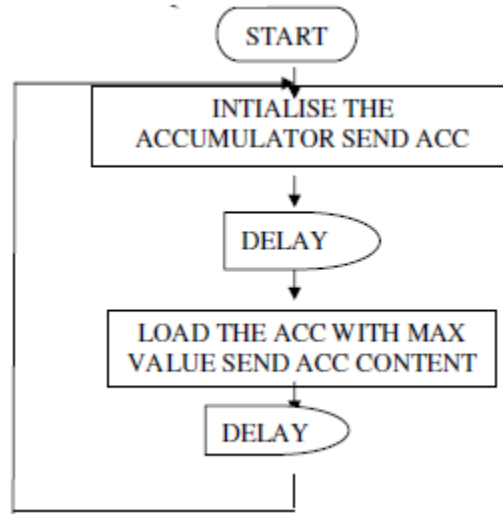
- Step I** : Send the digital value of DAC.  
**Step II** : Read the corresponding analog value of its output.

**FLOWCHART**

Waveform generation:

- Step I** : Square Waveform:  
**Step II** : Send low value (00) to the DAC.  
**Step III** : introduce suitable delay.  
**Step IV** : Send high value to DAC. Introduce delay.  
**Step V** : Repeat the above procedure.

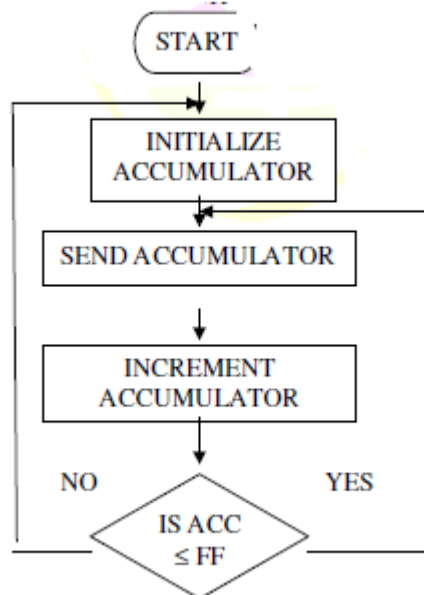
**FLOWCHART**



Saw-tooth waveform:

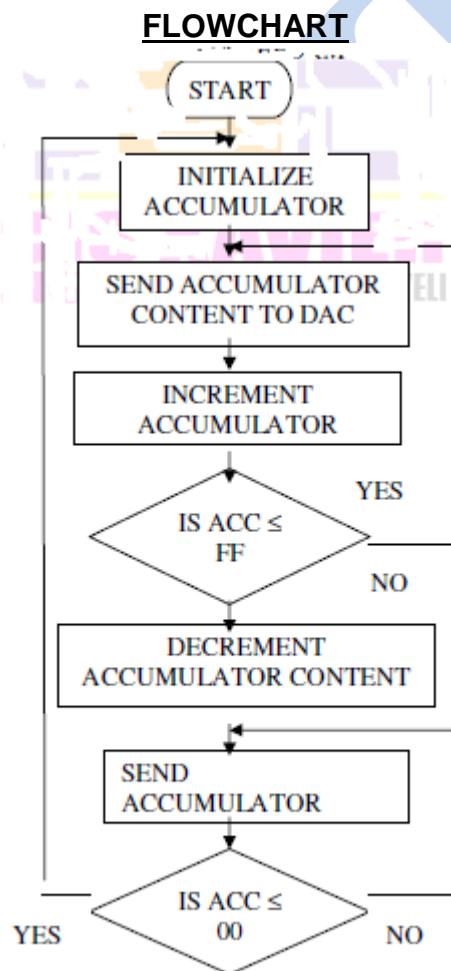
- Step I** : Load low value (00) to accumulator.
- Step II** : Send this value to DAC.
- Step III** : Increment the accumulator.
- Step IV** : Repeat step (ii) and (iii) until accumulator value reaches FF.
- Step V** : Repeat the above procedure from step 1.

**FLOWCHART**



Triangular waveform:

- Step I** Load the low value (00) in accumulator.  
**Step II** Send this accumulator content to DAC.  
**Step III** Increment the accumulator.  
**Step IV** Repeat step 2 and 3 until the accumulator reaches FF, decrement the  
**Step V** Decrementing and sending the accumulator contents to DAC.  
accumulator and send the accumulator contents to DAC.  
**Step VI** The above procedure is repeated from step (i)



**RESULT:** Thus the DAC was interfaced with 8085 and different waveforms have been generated.

**Outcome:** Upon completion of this experiment the student will be able to:

1. Demonstrate and 8255 PPI how the 8086 microprocessor is interfaced.
2. Design 8086 microprocessor system by using the peripheral devices
3. Interpret the data transfer from 8086 microprocessor to the peripheral device and vice versa.

GCET

**EXPERIMENT NO.7****INTERFACING ANALOG TO DIGITAL CONVERTER  
TO 8086**

**OBJECTIVE:** To write a Program to generate following wave forms

- a. Ramp waveform
- b. Square waveform
- c. Step waveform
- d. Triangle waveform

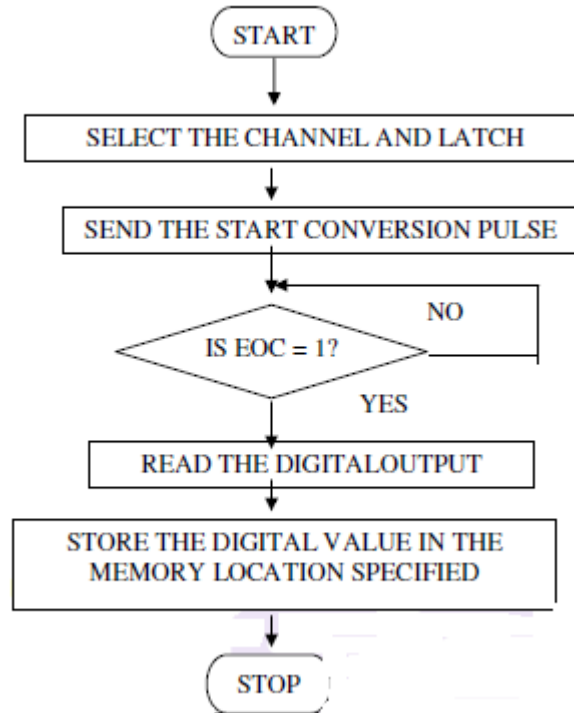
**APPARATUS:** 1.ADS-SDA-86-STA kit  
2.8255 Study card  
3. Adapter, Keyboard, Cables, CRO Etc . .

**PROCEDURE :**

1. Connect 8086 kit PC using RS232 cable.
2. Connect Power supply to 8086 kit and 8255 interfacing kit(only blue(+5v) and black(0v) lines Power cable to power supply)
3. Connect 8255 to CN4 of 8086 using 26 pin bus.
4. Connect the CRO probe to JP3 of 8255 kit
5. Keep the DIP switch in 1 & 7 on (8086kit), open TALK, and go to options select target device as 8086 and Connect.
6. Change dip switch into 1 & 5on, once reset 8086 kit.
7. Go to file →Download hex file
8. G-5000(on system keyboard), we can observe the output on 8086 kit and CRO.

**ALGORITHM**

- Step I** Select the channel and latch the address.  
**Step II** Send the start conversion pulse.  
**Step III** Read EOC signal.  
**Step IV** If EOC = 1 continue else go to step (iii)  
**Step V** Read the digital output.  
**Step VI** Store it in a memory location.

**FLOWCHART****RESULT:**

Thus the ADC was interfaced with 8086 and the given analog inputs were converted into its digital equivalent

**VIVA QUESTIONS:**

- 1) Which is by default pointer for CS/ES?
- 2) How many segments present in it?
- 3) What is the size of each segment?
- 4) Basic difference between 8085 and 8086?
- 5) Which operations are not available in 8085?

**Outcome:** Upon completion of this experiment the student will be able to:

1. Demonstrate and 8255 PPI how the 8086 microprocessor is interfaced.
2. Design 8086 microprocessor system by using the peripheral devices
3. Interpret the data transfer from 8086 microprocessor to the peripheral device and vice versa.

GCET



**EXPERIMENT NO.8****SERIAL COMMUNICATION BETWEEN TWO MICROPROCESSORS USING IC 8251**

**OBJECTIVE:** Write a program in ALP to establish Communication between two processors using 8251.

**APPARATUS:**

- 1.ADS-SDA-86-STA kit
- 2.8251 Study card
3. Adapter, Keyboard, Cables, Etc . . .

**PROCEDURE :****Transmission**

1. Connect 8086 kit PC using RS232 cable.
2. Connect Power supply to 8086 kit and 8251 interfacing kit(only blue(+5v) and black(0v) lines Power cable to power supply)
3. Connect 8251 to 8086 using 50pin and 26pin bus.
4. Short 5 & 6 pins of JP9 in 8251 kit
5. Keep the DIP switch in 1 & 7 on (8086kit), open TALK, and go to options select target device as 8086 and Connect.
6. Change dip switch into 1 & 5on, once reset 8086 kit.
7. Go to file →Download hex file
8. G-4000(on system keyboard), we can observe the output on 8251 kit.
9. Remove RS232 cable from 8086kit and connect it to 8251, transmitted data displayed on PC Monitor

**RECEIVEING**

1. Connect 8086 kit PC using RS232 cable.
2. Connect Power supply to 8086 kit and 8251 interfacing kit (only blue(+5v) and black(0v) lines Power cable to power supply)
3. Connect 8251 to 8086 using 50pin and 26pin bus.

4. Short 1 & 2 pins of JP9 in 8251 kit
5. Keep the DIP switch in 1 & 7 on (8086kit), open TALK, and go to options select target device as 8086 and Connect.
6. Change dip switch into 1 & 5on, once reset 8086 kit.
7. Go to file →Download hex file
8. Change the DIP switch into 1 & 7 on, once reset.
9. Remove RS232 cable from 8086 kit and connect it to 8251.
10. G-4000 (on 8086 kit keyboard) .enter
11. Give some input from system keyboard (Example press A, B, C, D enter),once reset 8086 kit That data will be received at 8086 kit at location FF00 (press E, enter address FF00 and press Comma you will get the ASCII values of A, B, C,D).

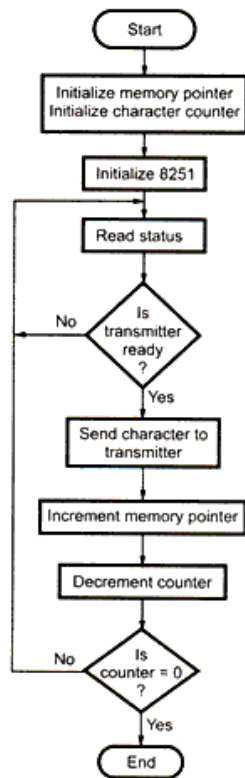
### **ALGORITHM**

**Step I** Initialize 8253 and 8251 to check the transmission and reception of a character

**Step II** Initialize8253 to give an output of 150Khz at channel 0 which will give a 9600 baud rate of 8251.

**Step III** The command word and mode word is written to the 8251 to set up for subsequent operations

**Step IV** The status word is read from the 8251 on completionof a serial I/O operation, or when the host CPU is checking the status of the device before starting the next I/O operation

**FLOWCHART**

**RESULT:** serial communication is established between two processors using 8251.

**VIVA-VOCE QUESTIONS:**

1. Expand USART?
2. Where do we prefer the serial communication?
3. What is the function of instruction pointer (IP) register?
4. What is the difference between IN and OUT instructions?
5. What is MODEM?

**OUTCOME:** Upon completion of this experiment the student will be able to:

1. Demonstrate and 8255 PPI how the 8086 microprocessor is interfaced.
2. Design 8086 microprocessor system by using the 8251 peripheral devices

3. Interpret the data transfer from 8086 microprocessor to the peripheral device and vice versa.



Fig1.8251 CONNECTIONS

**EXPERIMENT NO.9****INTERFACING TO 8086 AND PROGRAMMING TO CONTROL STEPPER MOTOR.**

**OBJECTIVE:** Write a program in ALP to interface stepper motor to 8086 and rotate it in clockwise And anticlockwise direction.

**APPARATUS:**

- 1.ADS-SDA-86-STA kit
2. Stepper motor interface card
3. 1 Amp Power Supply.
4. Stepper Motor
5. Adapter, Keyboard, Cables, Connecting Wires Etc . . .

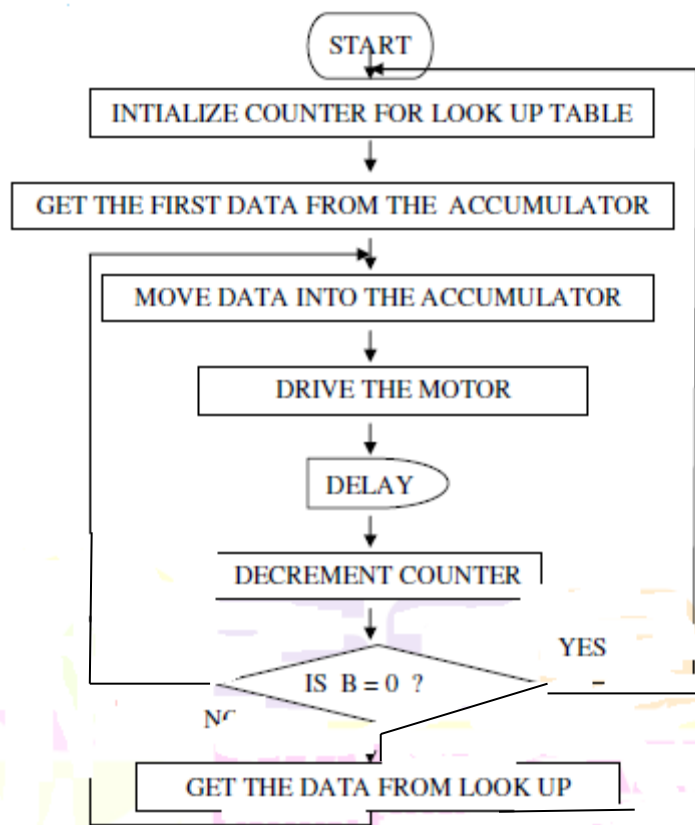
**PROCEDURE:**

1. Connect 8086 kit PC using RS232 cable.
2. Connect Power supply to 8086 kit
3. Connect 1Amp Power Supply to the Stepper Motor
4. Connect 8255 to CN4 of 8086 using 26 pin bus.
5. Keep the DIP switch in 1 & 7 on (8086kit), open TALK, and go to options select target device as 8086 and Connect.
6. Change dip switch into 1 & 5on, once reset 8086 kit.
7. Go to file →Download hex file
8. Keep the DIP switch in 1 & 7 on (8086kit)
9. G-4000(on kit keyboard), now the stepper motor will be rotating in clockwise direction

**ALGORITHM:**

- Step I** Load the stepping sequence in DI register
- Step II** Move the count value to CL register.
- Step III** Move the contents of destination index to AL.
- Step IV** Output the loaded values between delay

- Step V** Decrement the contents of DX register
- Step VI** If no zero jump to the L1
- Step VII** Increment destination index register.
- Step VIII** Loop to loop1.
- Step XI** Move the delay value to BX register.
- Step X** Decrement the contents of BX register
- Step XI** If no zero jump to L2.
- Step XII** Jump to beginning address
- Step XIII** Stop the program.
- Step XIV** Load the look up table values before execution

**FLOWCHART**

**RESULT:**

Thus the assembly language program for rotating stepper motor in both clockwise and anticlockwise directions is written and verified.

**VIVA QUESTIONS:**

- 1) What is the difference between min mode and max mode of 8086?
- 2) What is the difference between near and far procedure?
- 3) What is the difference between Macro and procedure?
- 4) What is the difference between instructions RET & IRET?
- 5) What is the difference between instructions MUL & IMUL?

**Outcome:** Upon completion of this experiment the student will be able to:

1. Demonstrate and 8255 PPI how the 8086 microprocessor is interfaced.
2. Design 8086 microprocessor system by using the peripheral devices
3. Interpret the data transfer from 8086 microprocessor to the peripheral device and vice versa.

**EXPERIMENT NO.10****INTERFACING TO 8086 AND PROGRAMMING TO DMA CONTROLLER**

**OBJECTIVE:** Write a program in ALP to interface DMA controller to 8086 and to verify transfer of data.

**APPARATUS:**

- 1.ADS-SDA-86-STA kit
2. DMA interface card
3. 1 Amp Power Supply.
- 4 Adapter, Keyboard, Cables, Connecting Wires Etc

**PROCEDURE:**

1. Connect 8086 kit PC using RS232 cable.
2. Connect Power supply to 8086 kit
3. Connect DMA controller board with 8086 kit
4. Keep the DIP switch in 1 & 7 on (8086kit), open TALK, and go to options select target device as 8086 and Connect.
5. Change dip switch into 1 & 5on, once reset 8086 kit.
6. Go to file →Download hex file
7. Keep the DIP switch in 1 & 7 on (8086kit)
8. G-4000(on kit keyboard), now the stepper motor will be rotating in clockwise direction

**ALGORITHM:**

- Step I** : DMA request is made.
- Step II** : As ACK is received, load DMA address register with starting address.
- Step III** : Load terminal count register with number of bytes to transfer
- Step IV** : Mode set register must be loaded with master/slave mode information
- Step V** : Transfer data. If data transfer completed ,then terminate DMA.



## FLOWCHART

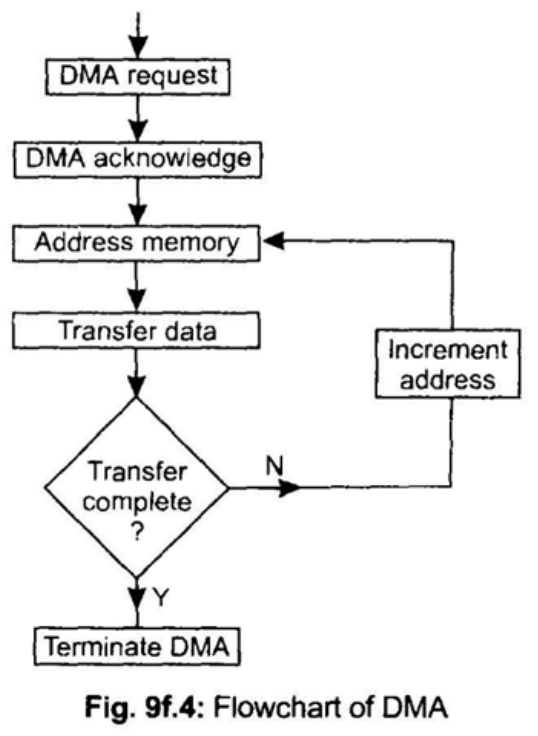


Fig. 9f.4: Flowchart of DMA

**Viva Questions:**

1. What is DMA?
2. What is the difference between processor and controller?
3. How the control transfers?
4. What is the advantage of DMA controller?
5. What is the working of DMA controller?

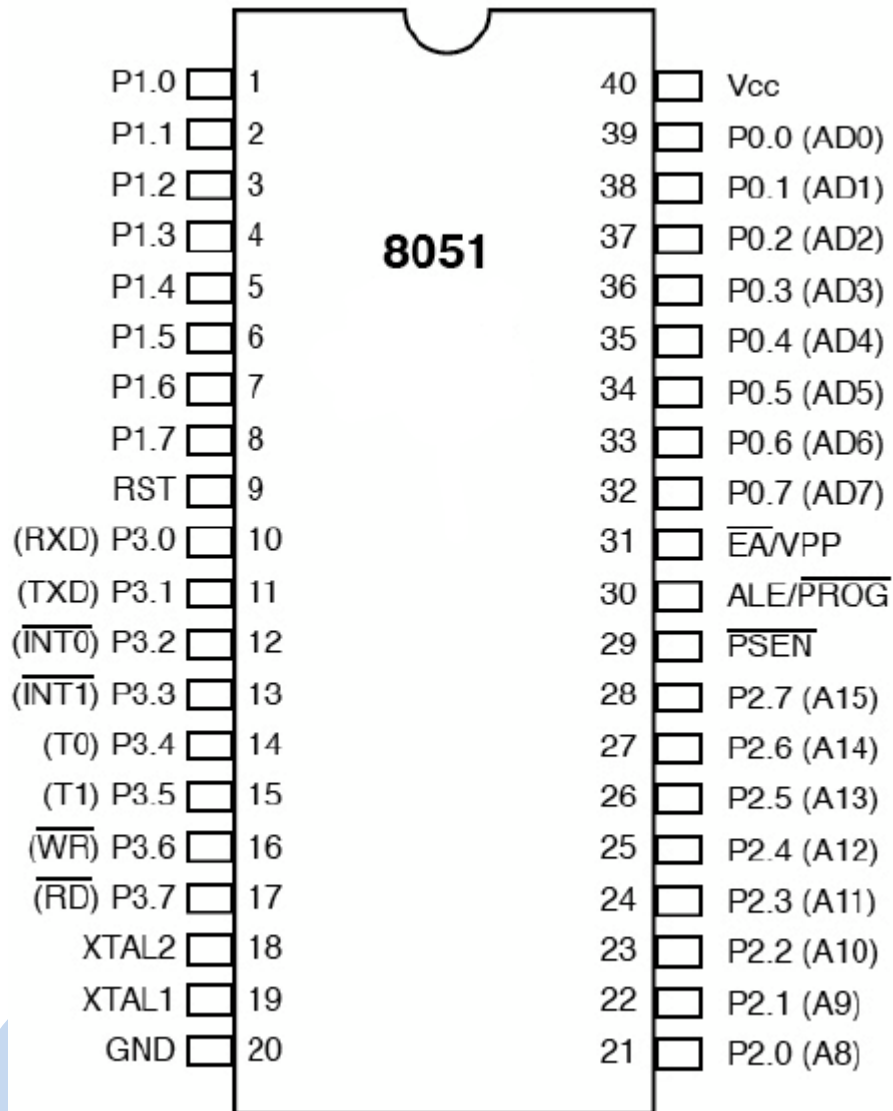
**Outcome:** Upon completion of this experiment the student will be able to:

1. Demonstrate DMA controller operation and data transfer methods.
2. Design 8086 microprocessor system by using the peripheral devices
3. Interpret the data transfer from 8086 microprocessor to the peripheral device and vice versa.

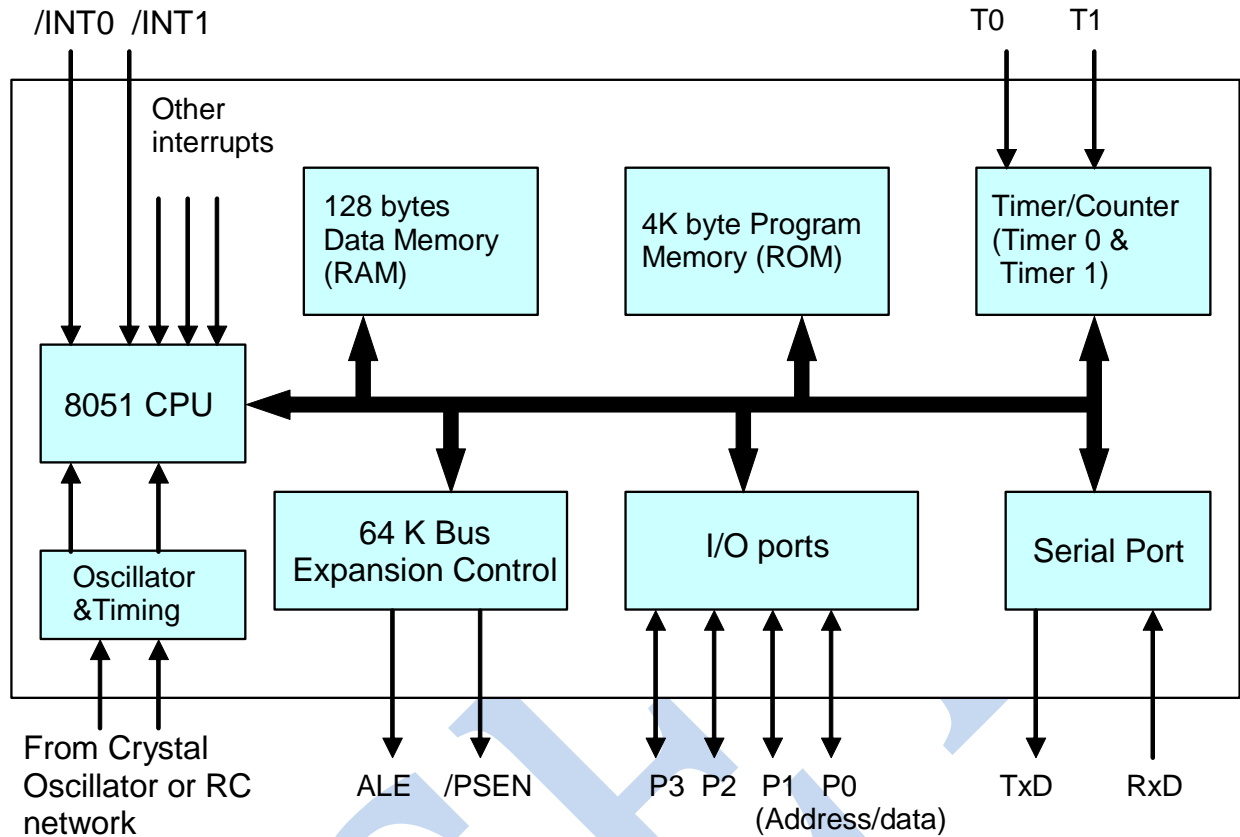
## INTRODUCTION TO 8051

The Intel 8051 is Harvard architecture, single chip microcontroller ( $\mu\text{C}$ ) which was developed by Intel in 1980 for use in embedded systems. 8051 is an 8-bit micro controller. The Important features of 8085 Architecture:

- 8-bit ALU, Accumulator and Registers;
- 8-bit data bus - It can access 8 bits of data in one operation
- 16-bit address bus - It can access  $2^{16}$  memory locations - 64 kB ( 65536 locations ) each of RAM and ROM
- On-chip RAM - 128 bytes ("Data Memory")
- On-chip ROM - 4 kB ("Program Memory")
- Four byte bi-directional input/output port
- UART (serial port)
- Two 16-bit Counter/timers
- Two-level interrupt priority
- Power saving mode
- 8051 have 128 user defined flags
- It consist of 16 bit address bus
- It also consist of 3 internal and two external interrupt
- Less power usage in 8051 with respect to other micro-controller
- It consist of 16-bit program counter and data pointer
- 8051 can process 1 million one-cycle instructions per second
- It also consist of 32 general purpose registers each of 8 bits
- Rom on 8051 is 4 Kbytes in size



Pin Diagram of 8051



Architecture of 8051

**Types of instructions:**

Depending on operation they perform, all instructions are divided in several groups:

- Arithmetic Instructions
- Branch Instructions
- Data Transfer Instructions
- Logic Instructions
- Bit-oriented Instructions

**Arithmetic instructions:**

Arithmetic instructions perform several basic operations such as addition, subtraction, division, multiplication etc. After execution, the result is stored in the first operand. For example:

ADD A,R1 - The result of addition ( $A+R1$ ) will be stored in the accumulator.

<b>Mnemonic</b>	<b>Description</b>
ADD A,Rn	Adds the register to the accumulator
ADD A,direct	Adds the direct byte to the accumulator
ADD A,@Ri	Adds the indirect RAM to the accumulator
ADD A,#data	Adds the immediate data to the accumulator
ADDC A,Rn	Adds the register to the accumulator with a carry flag
ADDC A,direct	Adds the direct byte to the accumulator with a carry flag
ADDC A,@Ri	Adds the indirect RAM to the accumulator with a carry flag
ADDC A,#data	Adds the immediate data to the accumulator with a carry flag
SUBB A,Rn	Subtracts the register from the accumulator with a borrow
SUBB A,direct	Subtracts the direct byte from the accumulator with a borrow
SUBB A,@Ri	Subtracts the indirect RAM from the accumulator with a borrow
SUBB A,#data	Subtracts the immediate data from the accumulator with a borrow
INC A	Increments the accumulator by 1
INC Rn	Increments the register by 1
INC Rx	Increments the direct byte by 1
INC @Ri	Increments the indirect RAM by 1
DEC A	Decrements the accumulator by 1
DEC Rn	Decrements the register by 1
DEC Rx	Decrements the direct byte by 1
DEC @Ri	Decrements the indirect RAM by 1
INC DPTR	Increments the Data Pointer by 1
MUL AB	Multiplies A and B
DIV AB	Divides A by B
DA A	Decimal adjustment of the accumulator according to BCD code

### **Branch Instructions:**

There are two kinds of branch instructions:

Unconditional jump instructions: upon their execution a jump to a new location from where the program continues execution is executed.

Conditional jump instructions: a jump to a new program location is executed only if a specified condition is met. Otherwise, the program normally proceeds with the next instruction.

<b>Mnemonic</b>	<b>Description</b>
ACALL addr11	Absolute subroutine call
LCALL addr16	Long subroutine call
RET	Returns from subroutine
RETI	Returns from interrupt subroutine
AJMP addr11	Absolute jump
LJMP addr16	Long jump
SJMP rel	Short jump (from -128 to +127 locations relative to the following instruction)
JC rel	Jump if carry flag is set. Short jump.
JNC rel	Jump if carry flag is not set. Short jump.
JB bit,rel	Jump if direct bit is set. Short jump.
JBC bit,rel	Jump if direct bit is set and clears bit. Short jump.
JMP @A+DPTR	Jump indirect relative to the DPTR
JZ rel	Jump if the accumulator is zero. Short jump.
JNZ rel	Jump if the accumulator is not zero. Short jump.
CJNE A,direct,rel	Compares direct byte to the accumulator and jumps if not equal. Short jump.
CJNE A,#data,rel	Compares immediate data to the accumulator and jumps if not equal. Short jump.
CJNE Rn,#data,rel	Compares immediate data to the register and jumps if not equal. Short jump.
CJNE @Ri,#data,rel	Compares immediate data to indirect register and jumps if not equal. Short jump.
DJNZ Rn,rel	Decrements register and jumps if not 0. Short jump.
DJNZ Rx,rel	Decrements direct byte and jump if not 0. Short jump.
NOP	No operation

**Data Transfer Instructions:**

Data transfer instructions move the content of one register to another. The register the content of which is moved remains unchanged. If they have the suffix "X" (MOVX), the data is exchanged with external memory.

<b>Mnemonic</b>	<b>Description</b>
MOV A,Rn	Moves the register to the accumulator
MOV A,direct	Moves the direct byte to the accumulator
MOV A,@Ri	Moves the indirect RAM to the accumulator
MOV A,#data	Moves the immediate data to the accumulator
MOV Rn,A	Moves the accumulator to the register
MOV Rn,direct	Moves the direct byte to the register
MOV Rn,#data	Moves the immediate data to the register
MOV direct,A	Moves the accumulator to the direct byte
MOV direct,Rn	Moves the register to the direct byte
MOV direct,direct	Moves the direct byte to the direct byte
MOV direct,@Ri	Moves the indirect RAM to the direct byte
MOV direct,#data	Moves the immediate data to the direct byte
MOV @Ri,A	Moves the accumulator to the indirect RAM
MOV @Ri,direct	Moves the direct byte to the indirect RAM
MOV @Ri,#data	Moves the immediate data to the indirect RAM
MOV DPTR,#data	Moves a 16-bit data to the data pointer
MOVC A,@A+DPTR	Moves the code byte relative to the DPTR to the accumulator (address=A+DPTR)
MOVC A,@A+PC	Moves the code byte relative to the PC to the accumulator (address=A+PC)
MOVX A,@Ri	Moves the external RAM (8-bit address) to the accumulator
MOVX A,@DPTR	Moves the external RAM (16-bit address) to the accumulator
MOVX @Ri,A	Moves the accumulator to the external RAM (8-bit address)
MOVX @DPTR,A	Moves the accumulator to the external RAM (16-bit address)
PUSH direct	Pushes the direct byte onto the stack
POP direct	Pops the direct byte from the stack
XCH A,Rn	Exchanges the register with the accumulator
XCH A,direct	Exchanges the direct byte with the accumulator

XCH A,@Ri	Exchanges the indirect RAM with the accumulator
XCHD A,@Ri	Exchanges the low-order nibble indirect RAM with the accumulator

**Logic Instructions:**

Logic instructions perform logic operations upon corresponding bits of two registers. After execution, the result is stored in the first operand.

<b>Mnemonic</b>	<b>Description</b>
ANL A,Rn	AND register to accumulator
ANL A,direct	AND direct byte to accumulator
ANL A,@Ri	AND indirect RAM to accumulator
ANL A,#data	AND immediate data to accumulator
ANL direct,A	AND accumulator to direct byte
ANL direct,#data	AND immediate data to direct register
ORL A,Rn	OR register to accumulator
ORL A,direct	OR direct byte to accumulator
ORL A,@Ri	OR indirect RAM to accumulator
ORL direct,A	OR accumulator to direct byte
ORL direct,#data	OR immediate data to direct byte
XRL A,Rn	Exclusive OR register to accumulator
XRL A,direct	Exclusive OR direct byte to accumulator
XRL A,@Ri	Exclusive OR indirect RAM to accumulator
XRL A,#data	Exclusive OR immediate data to accumulator
XRL direct,A	Exclusive OR accumulator to direct byte
XORL direct,#data	Exclusive OR immediate data to direct byte
CLR A	Clears the accumulator
CPL A	Complements the accumulator (1=0, 0=1)
SWAP A	Swaps nibbles within the accumulator
RL A	Rotates bits in the accumulator left
RLC A	Rotates bits in the accumulator left through carry
RR A	Rotates bits in the accumulator right
RRC A	Rotates bits in the accumulator right through carry



## Bit-oriented Instructions

Similar to logic instructions, bit-oriented instructions perform logic operations. The difference is that these are performed upon single bits.

### Mnemonic Description

CLR C	Clears the carry flag
CLR bit	Clears the direct bit
SETB C	Sets the carry flag
SETB bit	Sets the direct bit
CPL C	Complements the carry flag
CPL bit	Complements the direct bit
ANL C,bit	AND direct bit to the carry flag
ANL C,/bit	AND complements of direct bit to the carry flag
ORL C,bit	OR direct bit to the carry flag
ORL C,/bit	OR complements of direct bit to the carry flag
MOV C,bit	Moves the direct bit to the carry flag
MOV bit,C	Moves the carry flag to the direct bit

### Description of all 8051 instructions:

Here is a list of the operands and their meanings:

- **A** - accumulator;
- **Rn** - is one of working registers (R0-R7) in the currently active RAM memory bank;
- **Direct** - is any 8-bit address register of RAM. It can be any general-purpose register or a SFR (I/O port, control register etc.);
- **@Ri** - is indirect internal or external RAM location addressed by register R0 or R1;
- **#data** - is an 8-bit constant included in instruction (0-255);
- **#data16** - is a 16-bit constant included as bytes 2 and 3 in instruction (0-65535);

- **addr16** - is a 16-bit address. May be anywhere within 64KB of program memory;
- **addr11** - is an 11-bit address. May be within the same 2KB page of program memory as the first byte of the following instruction;
- **Rel** - is the address of a close memory location (from -128 to +127 relative to the first byte of the following instruction). On the basis of it, assembler computes the value to add or subtract from the number currently stored in the program counter;
- **bit** - is any bit-addressable I/O pin, control or status bit; and
- **C** - is carry flag of the status register (register PSW).

**EXPERIMENT NO. 11****PROGRAMMING USING ARITHMETIC, LOGICAL AND BIT MANIPULATION  
INSTRUCTIONS OF 8051**

**OBJECTIVE:** To write ALP for performing Arithmetic, Logical , Bit manipulation operations in 8051

**APPARATUS:**

- 1.ADS-SDA-51-STA kit
2. Microprocessor Power Supply
3. RS 232 Cable
5. Adapter, Keyboard, Cables, Connecting Wires Etc . . .

**PROCEDURE:****For generating HEX File.**

1. Check for Driver folder (if driver is in D drive)
2. Go to Command Prompt
3. Type d: and Press enter (if driver is in D drive)
4. Type cd Driver and Press enter
5. Type cd x8051 and Press enter
6. Edit filename.asm Enter
7. Now a window appears in which you need to write the program and save it .
8. Go to file→Exit. You will return to command prompt
9. To check errors and generate .obj file Type x8051 and Press enter in the command prompt.
  - A. Select the option D and enter
  - B.Select the option Y to generate cross-reference and enter
  - C.Enter Input file name as: filename.asm
  - D.Enter output filename as: filename.obj
10. Now the .obj file is generated and errors will be displayed
11. To link the file now type link and Press enter
  - A .Enter input filename as: filename.obj
  - B. To skip the entry of input, output and library files which are already

- C. added press enter 3 times
- D. Enter offset address as: 0000
- E. Select the option H to generate .hex file

12. Hex file is generated and the length and memory address of hex file will be displayed.  
Now note down the starting and ending address.

### **To Observe the OUTPUT:**

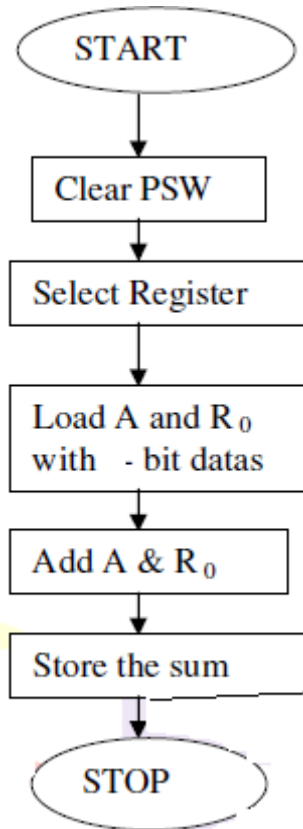
1. Connect the 8051 kit to the processor of desktop with Rs232 cable and power supply to the 8051 Kit.
2. Open the TALK icon which is on desktop now Talk window appears.
3. In that window go to options → Target Board → 8051 → ok.
4. Go to options → Connect.
5. Press E on kit keyboard to activate serial port of the kit. Now 'SERIAL PORT' displays on the kit and 'ALS 8051 STA' displays on talk window.
6. Go to file → Download Hex file. Select the HEX file which is generated by following the path D drive → driver → X8051 → filename.hex. Now I appears on Talk window to indicate that file is downloaded.
7. Type G and Starting address of HEX file and press enter.
8. Now program gets executed and register status will be displayed.
9. To generate opcode Press Z starting address of hex file and enter.

## **I.ARITHMETIC OPERATIONS**

### **ALGORITHM:**

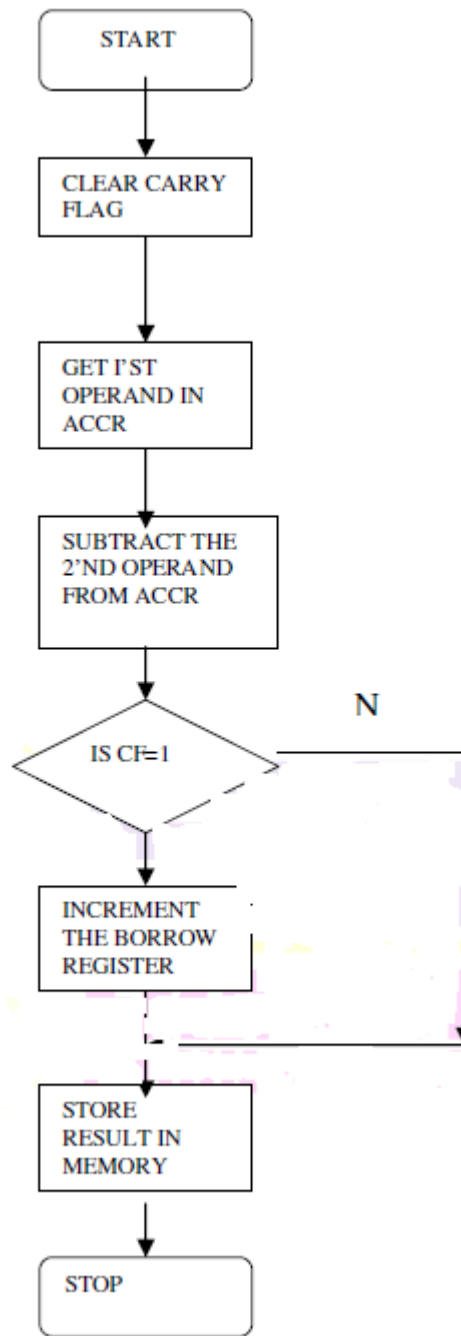
#### **addition without carry**

- Step 1** : Clear Program Status Word.
- Step 2** : Select Register bank by giving proper values to RS1 & RS0 of PSW.
- Step 3** : Load accumulator A with any desired 8-bit data.
- Step 4** : Load the register R0 with the second 8-bit data.
- Step 5** : Add these two 8-bit numbers.
- Step 6** : Store the result.

**FLOWCHART****Subtraction with carry**

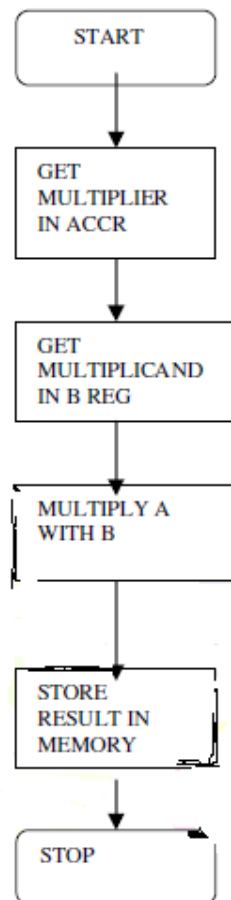
- Step 1** : Clear the carry flag.
- Step 2** : Initialize the register for borrow.
- Step 3** : Get the first operand into the accumulator.
- Step 4** : Subtract the second operand from the accumulator.
- Step 5** : If a borrow results increment the carry register.
- Step 6** : Store the result in memory

**FLOWCHART**



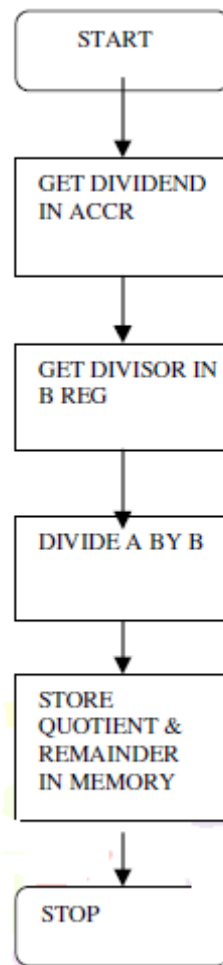
**Multiplication**

- Step 1** : Get the multiplier in the accumulator.  
**Step 2** : Get the multiplicand in the B register.  
**Step 3** : Multiply A with B.  
**Step 4** : Store the product in memory.

**FLOWCHART**

**Division**

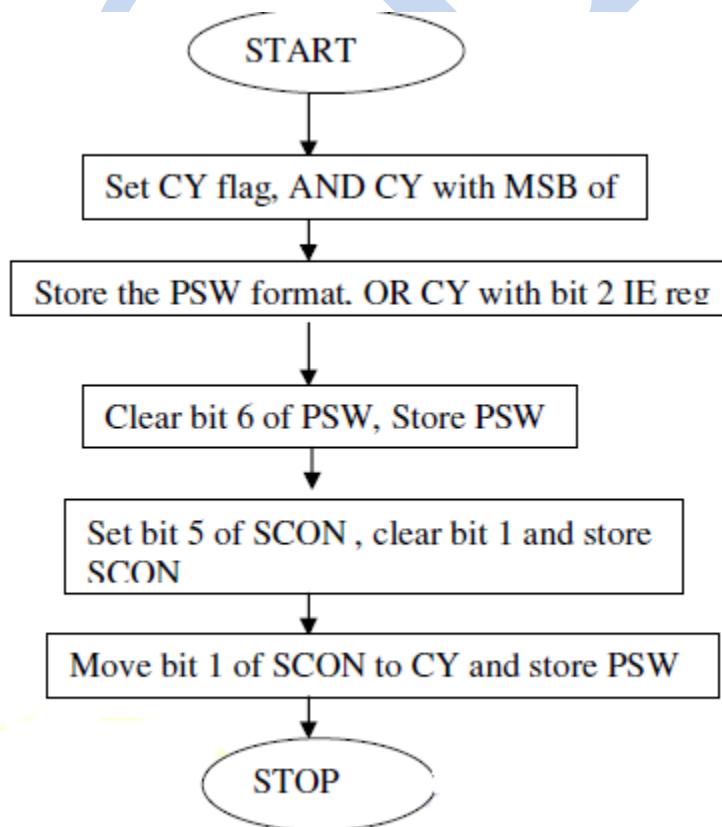
- Step 1** : Get the Dividend in the accumulator.  
**Step 2** : Get the Divisor in the B register.  
**Step 3** : Divide A by B.  
**Step 4** : Store the Quotient and Remainder in memory.

**FLOWCHART**



**II.LOGICAL OPERATIONS****ALGORITHM:**

- Step 1** : Initialize content of accumulator as FFH
- Step 2** : Set carry flag (cy = 1).
- Step 3** : AND bit 7 of accumulator with cy and store PSW format.
- Step 4** : OR bit 6 of PSW and store the PSW format.
- Step 5** : Set bit 5 of SCON.
- Step 6** : Clear bit 1 of SCON.
- Step 7** : Move SCON.1 to carry register.
- Step 8** : Stop the execution of program.

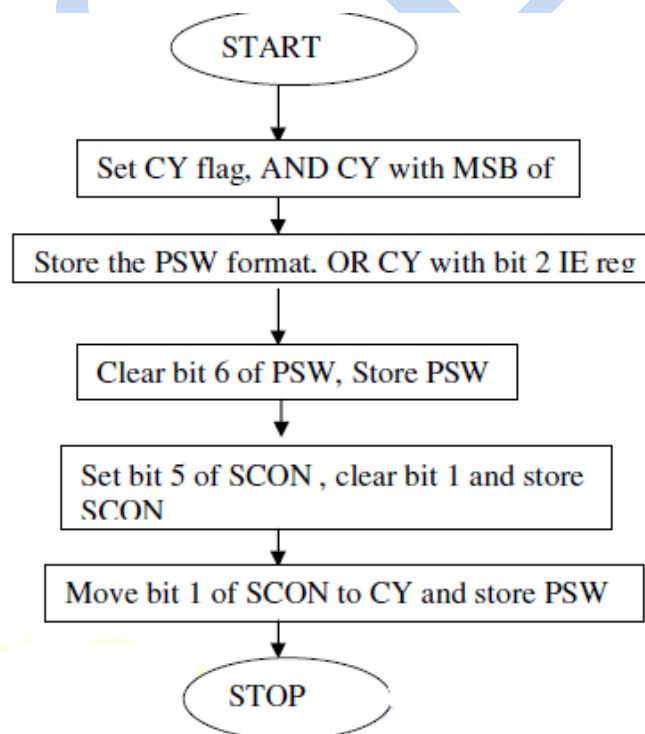
**FLOW CHART**

### III. BIT MANIPULATION OPERATIONS

#### ALGORITHM:

- Step 1** : Initialize content of accumulator as FFH
- Step 2** : Set carry flag (cy = 1).
- Step 3** : AND bit 7 of accumulator with cy and store PSW format.
- Step 4** : OR bit 6 of PSW and store the PSW format.
- Step 5** : Set bit 5 of SCON.
- Step 6** : Clear bit 1 of SCON.
- Step 7** : Move SCON.1 to carry register.
- Step 8** : Stop the execution of program.

#### FLOW CHART



**VIVA QUESTIONS:**

- 1) What is the function of 01h of Int 21h?
- 2) What is the function of 02h of Int 21h?
- 3) What is the function of 09h of Int 21h?
- 4) What is the function of 0Ah of Int 21h?
- 5) What is the function of 4ch of Int 21h?

**RESULT:**

Hence Performing Arithmetic, Logical , Bit manipulation operations in 8051

**OUTCOME:**

Upon completion of this experiment the student will be able to perform Arithmetic, logical and bit manipulation operations.

**EXPERIMENT NO.12****PROGRAM AND VERIFY TIMER/COUNTER IN 8051****OBJECTIVE:**

To program and verify Timer/counter in 8051(Timer 0 mode 2 counter).

**APPARATUS:**

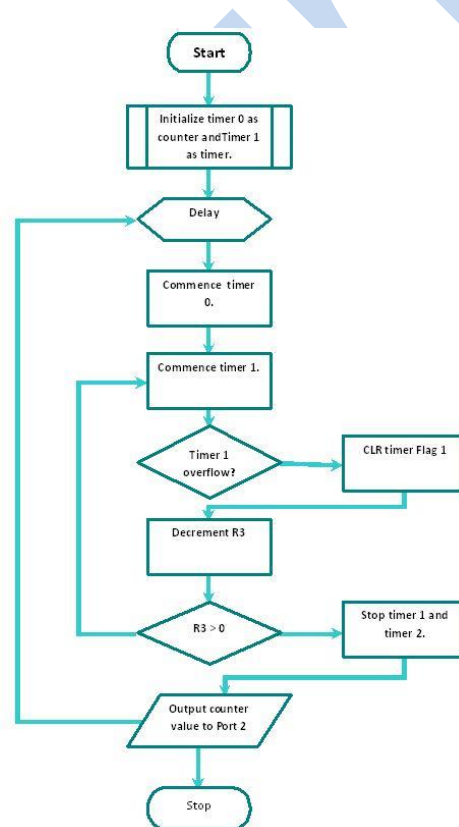
- 1.ADS-SDA-51-STA kit
2. Microprocessor Power Supply
3. RS 232 Cable
4. NIFC-26 study card
5. Adapter, Keyboard, Cables, Connecting Wires Etc . . .

**PROCEDURE:**

1. Connect the 8051 kit to the processor of desktop with Rs232 cable and power supply to the 8051 Kit.
- 2.Short 1 & 2 pins of JP1 of Study card.
- 3.Connect NIFC 26 study card to the 8051 using 50 line bus (P2 of 8051 kit to study card) and 10 line bus (JP12 of 8051 kit to JP3 of Study card).
- 4.Now switch on the power supply.
5. Open the TALK icon which is on desktop now Talk window appears.
6. In that window go to options → Target Board → 8051 → ok.
7. Go to options →Connect.
8. Press E on kit keyboard to activate serial port of the kit. Now '**SERIAL PORT**' displays on the kit and '**ALS 8051 STA**' displays on talk window.
9. Go to file → Download Hexfile. Select the HEX file by following the path E drive →Talk → MC COMM → Nifc26→Nifc26. Now I appears on TALK window to indicate that file is downloaded.
10. Type G 9200 (Starting address) and press enter.
11. Now program gets executed
12. Now data location displays on the LCD display of Kit and by pressing SW1 of study card we can observe the increment in the data field displayed.

**ALGORITHM:**

- Step 1** : Load TMOD register with mode of operation and timer to be used.
- Step 2** : Load timer registers with initial count values
- Step 3** : Start the timer .
- Step 4** : Monitor for Overflow Flag
- Step 5** : If TF = 1; Stop the timer.
- Step 6** : clear timer flag for the next cycle

**FLOWCHART****VIVA QUESTIONS:**

- 1) Can we use SP as offset address holder with CS?
- 2) Which is the base registers in 8086?
- 3) Which is the index registers in 8086?
- 4) What do you mean by segment override prefix?
- 5) Whether micro reduces memory requirements?

**RESULT:**

Timer0 as counter in mode 0 is programmed and verified.

**OUTCOME:**

Students will be able to understand Timer and Counter programming in 8051.

**EXPERIMENT NO.13****PROGRAM AND VERIFY INTERRUPT HANDLING IN 8051****OBJECTIVE:**

To program and verify interrupt handling in 8051

**APPARATUS**

- 1.ADS-SDA-51-STA kit
2. Microprocessor Power Supply
3. RS 232 Cable
- 4.NIFC-26 study card
5. Adapter, Keyboard, Cables, Connecting Wires Etc . . .

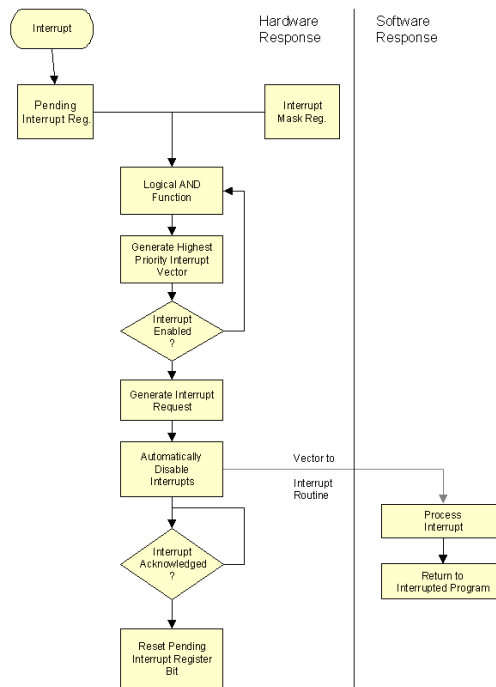
**PROCEDURE:**

- 1.Connect the 8051 kit to the processor of desktop with Rs232 cable and power supply to the 8051 Kit.
- 2.Short 1 & 2 pins of JP5 and JP6 of Study card.
- 3.Connect NIFC 26 study card to the 8051 using 50 line bus (P2 of 8051 kit to study card) and 10 line bus (JP12 of 8051 kit to JP3 of Study card).
- 4.Now switch on the power supply.
5. Open the TALK icon which is on desktop now Talk window appears.
6. In that window go to options → Target Board → 8051 → ok.
7. Go to options →Connect.
8. Press E on kit keyboard to activate serial port of the kit. Now '**SERIAL PORT**' displays on the kit and '**ALS 8051 STA**' displays on talk window.
9. Go to file → Download Hexfile. Select the HEX file by following the path E drive →Talk → MC COMM → Nifc26→Nifc26. Now **I** appears on TALK window to indicate that file is downloaded.
10. Type G 9700 (Starting address) and press enter.
11. Now program gets executed
12. Now data location displays on the LCD display of Kit (continues swapping of bits 26)
13. Give an external interrupt by pressing sw3 on study card now you can observe that data field counts from 0 to 2 and continues swapping of bits 26

**ALGORITHM :**

- Step1** : Load IE register with respect to interrupts to be handled
- Step 2** : Load TMOD register with MOD and Timer information.
- Step 3** : Load timer registers
- Step 4** : Wait for an interrupt. If interrupt occurs, the control has to be passed to sub-routine by calculating interrupt vector
- Step 5** : By executing RETI instruction the control will execute the main program.

**FLOWCHART:**





**VIVA QUESTIONS:**

- 1) How many no. of interrupts available for 8051?
- 2) Which is the highest priority interrupt for 8051?
- 3) What is an ISR and IVT?
- 4) What is the difference between software and hardware interrupt?
- 5) What is the vector address for serial communication interrupt?

**RESULT:**

Hence program and verification of interrupt handling in 8051 is done.

**OUTCOME:**

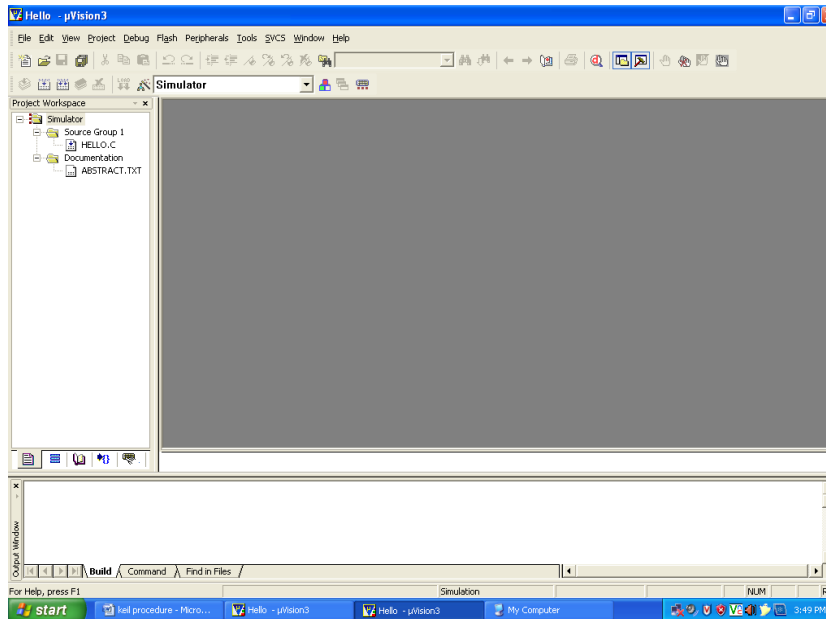
This program demonstrates the usage of priority bit and external internal type bit to handle the interrupt.

# ADDITIONAL EXPERIMENTS

## INTRODUCTION TO KEIL $\mu$ VISION3 TOOL

### Procedure to create a new $\mu$ vision project:

**Step 1:** Give a double click on  $\mu$ vision3 icon on the desktop, it will generate a window as shown below:



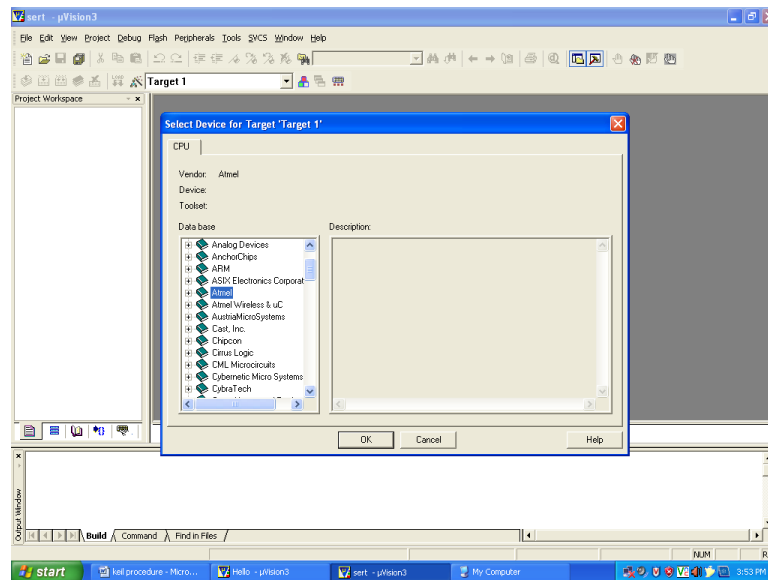
**Step 2:** To create new project, go to project, select new  $\mu$ vision project.

**Step 3:** Select a drive where you would like to create your project.

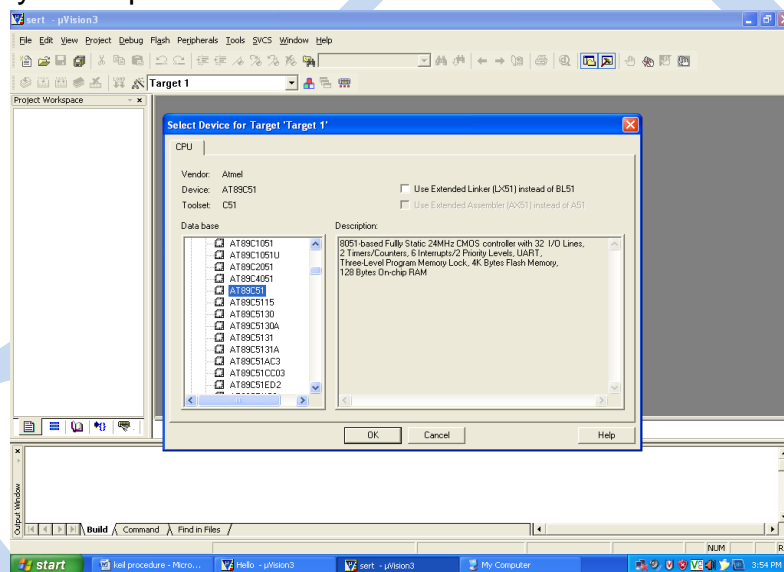
**Step 4:** Create a new folder and name it with your project name.

**Step 5:** Open that project folder and give a name of your project executable file and save it.

**Step 6:** After saving, it will show some window there select your microcontroller company i.e. Atmel.



**Step 7:** Select your chip as AT91C51.



**Step 8:** After selecting chip click on OK then it will display some window asking to add STARTUP file. Select YES.

**Step 9:** A target is created and startup file is added to your project target and is shown below.

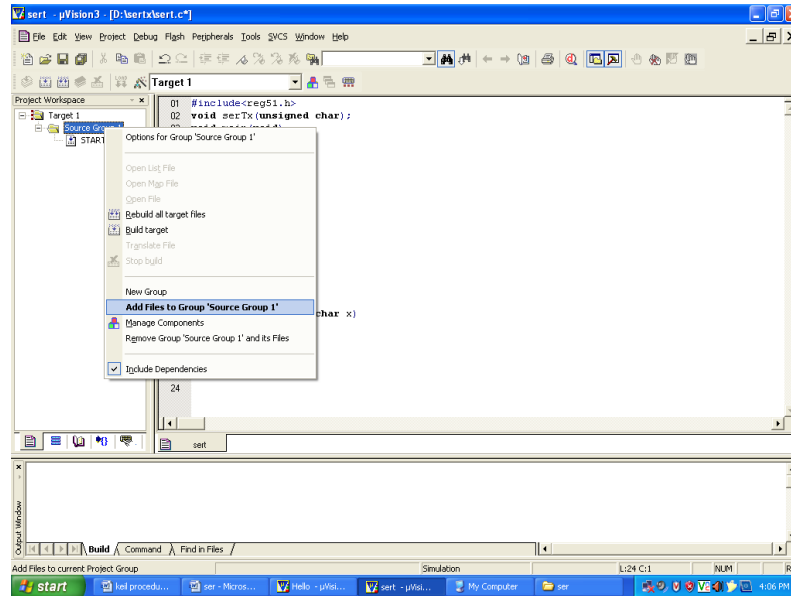
**Step 10:** To write your project code select a new file from FILE menu bar.

**Step 11:** It will display some text editor, to save that file select SAVE option from FILE menu bar.

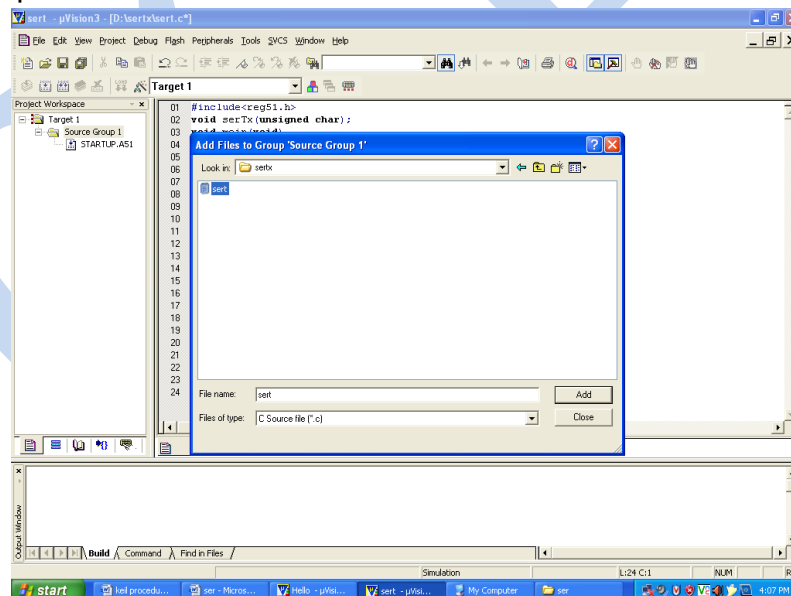
**Step 12:** Save file name with .c extension.

**Step 13:** Write the code of your project and save it.

**Step 14:** To add c file to target, give a right click on Source Group, choose “ADD files to Group” option.

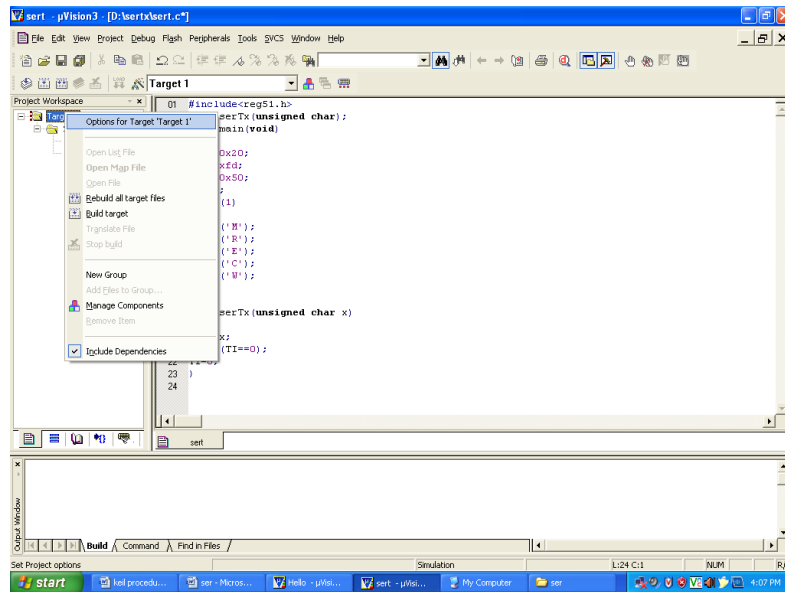


**Step 15:** It will display some window there select the file you have to add and click on ADD option.



**Step 16:** The file will be added to target and it is shown in the project window.

**Step 17:** Now give a right click on target in the project window and select “Options for Target”.

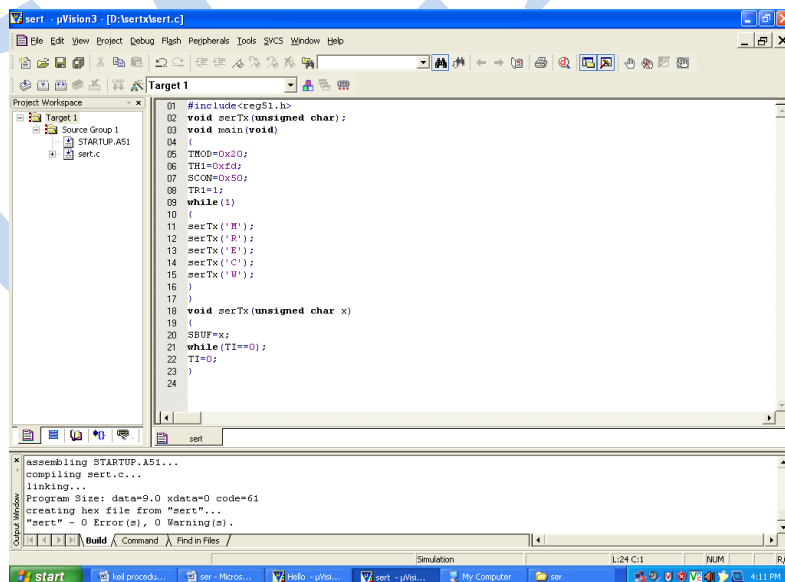


**Step 18:** It will show some window, in that go to output option and choose Create Hex file option by selecting that box.

**Step 19:** In the same window go to Linker option and choose use memory layout from target dialog by selecting the box, and Click OK.

**Step 20:** Now to compile your project Select build target option or press F7.

**Step 21:** In the build OUTPUT window, you can see the errors and warnings of the code. Project HEX file will be created.



**Experiment No: 14**  
**SERIAL TRANSMISSION FROM PC TO 8051 $\mu$ C**

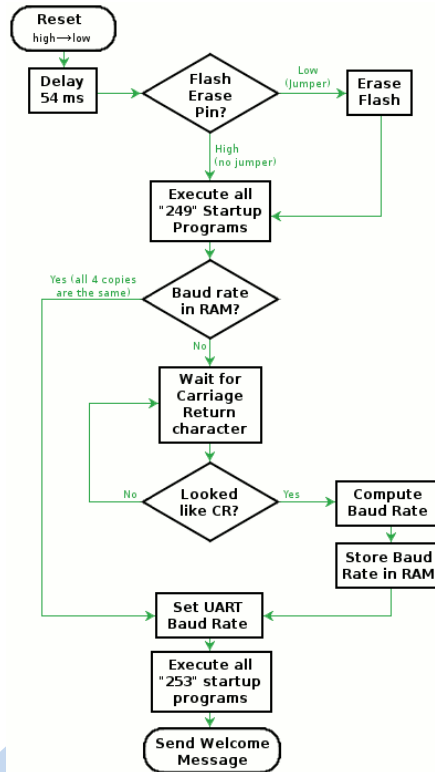
**OBJECTIVE:**

To transmit characters from PC to 8051uc serially and to see the output on terminal window.

**TOOLS:** PC installed with KEIL  $\mu$ vision3

**ALGORITHM:**

- Step1** : The TMOD register is loaded with the value 20H, indicating the use of Timer 1 in mode 2 (8-bit auto-reload) to set the baud rate
- Step2** : The TH1 is loaded with one of the values TO set the baud rate For serial data transfer (assuming XTAL = 11.0592 MHz).
- Step 3** : The SCON register is loaded with the value 50H, indicating serial mode 1, where an 8-bit data is framed with start and stop bits.

**FLOWCHART****RESULT:**

By connecting 8051 to PC students can transmit data serially.

**OUTCOME:**

Students will be able to understand

- 1) Demonstration of serial communication
- 2) How to interface UART.




**Experiment No: 15**  
**PORT PROGRAMMING OF 8051**

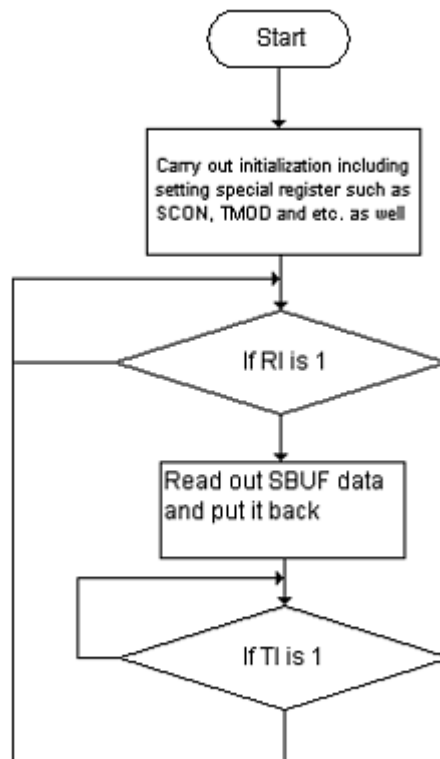
**OBJECTIVE:**

To write 8051 C program to toggle all the bits of P0 and P2 continuously with some delay.

**TOOLS:** PC installed with KEIL  $\mu$ vision3.

**ALGORITHM**

- 
- Step1** : The TMOD register is loaded with the value 20H, indicating the use of Timer 1 in mode 2 (8-bit auto-reload) to set the baud rate
- Step 2** : TH1 is loaded with one of the values in to set the baud rate (assuming XTAL = 11.0592MHz).
- Step 3** : The SCON register is loaded with the value 50H, indicating serial mode 1, where 8-bit data is framed with start and stop bits and receive enable is turned on.
- Step 4** : TR1 is set to 1 to start Timer 1
- Step 5** : RI is cleared with the "CLR RI" instruction
- Step 6** : The RI flag bit is monitored with the use of the instruction "JNB RI, xx" to see if an entire character has been received yet
- step 7** : When RI is raised, SBUF has the byte. Its contents are moved into a safe place
- Step 8** : To receive the next character, go to Step 5.

**FLOW CHART****RESULT:**

By this program students are able to toggle all the bits of two ports with some delay.

**OUTCOME:**

By the end of this program students will be able to understand the port programming.



**VIVA QUESTIONS & ANSWERS****1. Define microprocessor.**

A) A microprocessor is a multipurpose, programmable, clock-driven, register-based electronic device that reads binary instructions from a storage device called memory accepts binary data as input and processes data according to instructions, and provides result as output.

**2. Define microcomputer.**

A) A computer that is designed using a microprocessor as its CPU, It includes microprocessor, memory, and I/O.

**3. Define ROM.**

A) A memory that stores binary information permanently, the information can be read from this memory but cannot be altered.

**4. What is an ALU?**

A) The group of circuits that provides timing and signals to all the operations in the computer also controls data flow.

**5. Define small-scale integration.**

A) The process of designing a few circuits on a single chip, the term refers to the technology used to fabricate logic gates on a chip.

**6. What is an instruction?**

A) An instruction is a binary pattern entered through an input device in memory to command the microprocessor to perform specific function.

**7. What are the four primary operations of a MPU?**

- A)
1. Memory read
  2. Memory write
  3. I/O read
  4. I/O write

**8. What do you mean by address bus?**

A) The address bus is a group of 16 lines generally identified as A0 to A15. The address bus is unidirectional: bits flow from MPU to peripheral devices.

**9. How many memory locations can be addressed by a microprocessor with 14 address lines?**

A) The 8085 MPU with its 14-bit address is capable of addressing  $2^{14} = 16,384$  (ie) 16K memory locations.

**10. Why does the data bus bi-directional?**

A) The data bus is bi-directional because the data flow in both directions between the MPU and memory and peripheral devices.

**11. What is the function of the accumulator?**

A) The accumulator is the register used to store the 8-bit data to perform the arithmetic and logical operations.

**12. Define control bus.**

A) This is single line that is generated by the MPU to provide timing of various operations.

**13. What is a flag?**

A) The data conditions, after arithmetic or logical operations, are indicated by setting or resetting the flip-flops called flags.

**14. Why are the program counter and the stack pointer 16-bit registers?**

A) Memory locations for the program counter and stack pointer have 16-bit addresses. So the PC and SP have 16-bit registers.

**15. Define memory word.**

A) The number of bits stored in a register is called a memory word.

**16. Specify the number of registers and memory cells required in a 128 x 4 memory chip.**

A) Number of registers = 128  
Memory cells required is  $128 \times 4 = 512$

**17. Explain the function of ALE and IO/M signals in the 8086 architecture.**

A) The ALE signal goes high at the beginning of each machine cycle indicating the availability of the address on the address bus, and the signal is used to latch the low-order address bus. The IO/M signal is a status signal indicating whether the machine cycle is I/O or memory operation. The IO/M signal is combined with the RD and WR control signals to generate IOR, IOW, MEMW, MEMR.

**18. Specify the contents of the accumulator and the status of the S, Z, and CY flags, if 8086 adds 87H and 79H.**

A) The sum of 87H and 79H is 100H. Therefore, the accumulator will have 00H, and the flags will be  $S=0$ ,  $Z = 1$ ,  $CY = 1$

**20. Write down the control and status signals.**

A) Two Control signals and three status signals  
Control signals: RD and WR

Status signals: IO/M, S1, and S2

**21. Define machine cycle.**

A) Machine cycle is defined, as the time required completing one operation of accessing memory, I/O, or acknowledging an external request.

**22. Define T-state.**

A) T-state is defined as one subdivision of the operation of performed in one clock period.

**23. What is a transparent latch? Why is it necessary to latch with output devices such as LED's?**

A) A transparent latch is a flip-flop; its output changes according to input when the clock signal is high, and it latches the input on the trailing edge of the clock. The latch is necessary for output devices to retain the result; otherwise the result will disappear

**24. Give the bit positions reserved for the flags.**

D7 D6 D5 D4 D3 D2 D1 D0

S Z AC P CY

**25. Define instruction cycle.**

A) Instruction cycle is defined, as the time required completing the execution of the instruction.

**26. What is an instruction set?**

A) The entire group of instructions determines what functions the microprocessor can perform are called instruction set.

**27. Give the functional categories of 8086 microprocessor instructions of data transfer operations**

- Arithmetic operations o Logical operations
- Branching operations
- Machine control operations

**28. Define Op-code and operand.**

A) The operation to be performed is called Op-code. The data to be operated is called operand.

**29. Define the types of branching operations.**

A) Jump: to test the conditions

Call, Return, And Restart: Change the sequence of the program

**30. Define two-byte instruction with one example.**

A) In a 2-byte instruction, the first byte specifies the Opcode; the second byte specifies the operand.

Example:	Op-code	operand	binary code	hex code	
	MVI	A, Data	011 1110	3E	First byte
			Data Data		SecondByte

6. Write an instruction to load the hexadecimal numbers 65H in register C, and 92H in the accumulator A, and display the number 65H at PORT0 and 92H at PORT1?

```

MVI C,
  65H
MVI A,
  92H
OUT PORT1 ; DISPLAY 92H
MOVA, C ; COPY C INTO A FOR
  DISPLAY
OUT PORT0 ; DISPLAY 65H
HLT

```

**31. Which operation can be performed by using the instruction ADD A?**

A) The instruction ADD A, adds the content of an accumulator to itself, this is equivalent to multiplying by 2.

**32. Which operation can be performed by using the instruction SUB A? Specify the status of Z and CY?**

A) The instruction SUB A will clear the accumulator. The flag status will be CY = 0 and Z = 1.

**33. How does the instruction set classified?**

A) The instruction set is classified in three groups according to the word size:

- 1-byte instruction
- 2-byte instruction
- 3-byte instruction

**34. What is JNC 16-bit address?**

A) It changes the program sequence to the location specified by the 16-bit address if the carry flag is reset.

**35. Give the instructions that perform the logical operations?**

- AND, OR, Exclusive-OR
- Rotate
- Compare
- Complement

**36. What is a three-byte instruction?**

A) In a three-byte instruction, the first byte specifies the opcode, and the following two bytes specifies the 16-bit address.

**37. Define a program.**

A) A program is a sequence of instructions written to tell the computer to perform a specific function.

**38. Define ASCII code.**

A) ASCII code is a 7-bit code that represents both decimal numbers, alphabets. Extended ASCII is an 8-bit code.

**39. What is STA in data transfer instruction?**

A) Copy the data from the accumulator in the memory location specified by the 16-bit address

**40. What is an IN instruction?**

A) This is a 2-byte instruction. It accepts data from the input port specified in the second byte.

**41. What is an OUT instruction?**

A) This is a 2-byte instruction. It sends the content of the accumulator to the output port specified in the second byte.

**42. Give the difference between JZ and JNZ?**

A) JZ change the program sequence to the location specified by the 16-bit address if the zero flag is set. JNZ change the program sequence to the location specified by the 16-bit address if the zero flag is reset.

**43. What is CMA?**

A) It complements the data in the accumulator.

**44. What is the function of CALL instruction?**

A) CALL instruction changes the sequence to the location of a subroutine.

**45. Why the number of out ports in the peripheral-mapped I/O is restricted to 256 ports?**

A) The number of output ports in the peripheral I/O is restricted to 256 ports because the operand of the OUT instruction is 8-bits; it can have only 256 combinations

**46. If an input and output port can have the same 8-bit address how does the 8085 differentiate between the ports.**

A) In The 8085 differentiates between the input and output ports of the same address by the control signal. The input port requires the RD and the output port requires the WR signal.

**47. What are the control signals necessary in the memory mapped I/O?**

A) RD, WR, I/O (low)

**48. Why a latch is used for the output port and a tri-state buffer is used for the input port?**

A) A latch is necessary to hold the output data for display. The input data byte is obtained by enabling a tri-state buffer and placed in the accumulator.

**49. What happens when the 8086 execute the out instruction?**

A) When the 8086 executes the out instruction, in the third machine cycle, it places the output port address on the low-order address bus, duplicates the same port address on the high-order bus, places the contents of the accumulator on the data bus and asserts the control signal WR.

**50. How does the port number be affected if we decode the higher-order address lines A15 – A8 rather than lower-order address lines A7 – A0?**

A) The port address will remain the same because the I/O port address is duplicated on both segments of the address bus.

**51. Define Memory mapped I/O.**

A) Instead of a memory register, if an output device is connected at the address, the accumulator contents will be transferred to the output device. This is called memory mapped I/O.

**52. What is an interrupt I/O?**

A) The interrupt I/O is a process of data transfer whereby an external device or a peripheral can inform the processor that it is ready for communication and it requests attention

**53. What is Partial Decoding?**

A) The output port can be selected by decoding some of the address lines is called partial decoding

**54. What is the purpose for the 8255 PPI?**

A) The 8255A is a widely used, programmable, parallel I/O device. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O.



**55. List the operating modes of 8255A PPI.**

- A) Two 8-bit ports (A and B)
- Two 4-bit ports (Cu and CL)
- Data bus buffer
- Control logic

**56. Specify the bit of a control word for the 8255, which differentiates between the I/O mode and the BSR mode.**

- A) BSR mode D7= 0, and I/O mode D5 = 1

**57. Write the input/output feature in Mode 0 for the 8255A PPI?**

- A) Outputs are latched
- Inputs are not latched
- Ports do not have handshake or interrupt capability

**58. Write the control word format in the BSR mode.**

D7	D6	D5	D4	D3D2	D1D0
0	X	X	X	Bit Select	S/R
BSR mode	Not used			Set = 1	
	Generally Set= 0				

**59. Write down the output control signals used in 8255A PPI.**

- A) OBF output Buffer Full
- ACK Acknowledge
- INTR Interrupt request
- INTE Interrupt Enable

**60. What is the use of mode 2 in 8255A PPI?**

- A) This mode is used primarily in applications such as data transfer between two computers or floppy disk controller interface.