Cover page

Hands on Experience for Faculty in Laboratories
Phase I
JNTUK, Kakinada

Report

Details of College

Name of the College	Lendi Institute of Engineering & Technology
College Code and District	KD,Vizianagaram
Name of the Principal	Dr.V.V.Rama Reddy
Contact No's	

Details of the Department

Name of the Department	ELECTRONICS AND COMMUNICATION ENGINEERING
Name of Head of the Department	Dr.M.Rajan Babu
Contact No's	9885239177

Details of the Faculty Member

Name of the Faculty Member	B.HEMANTHNAG
Qualification and Specialization	M.Tech, DIGITAL ELECTRONICS AND COMMUNICATION SYSTEMS
Contact No's	8500906200

Details of the Faculty Member

Name of the Faculty Member	S RAMA KRISHNA
Qualification and Specialization	M.Tech, VLSI SD
Contact No's	8499874636

Details of the Laboratory

Year and Semester of Lab	III YEAR II SEM
Name of the Laboratory	MICROPROCESSOR & MICROCONTROLLERS Lab
No of Experiments as per syllabus	32
No of Experiments conducted	32

Signature of Faculty Member Signature of HOD Signature of Principal

Hands-on Experience for Faculty in Laboratories

Phase I

Preamble

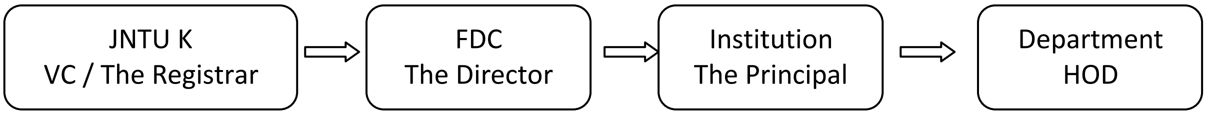
The "**Hands-on Experience for Faculty in Laboratories**" is a faculty development programme conceptualized and designed by the Directorate of Faculty Development, JNTU K under the scholarly guidance of The Hon'ble Vice-Chancellor Prof G Tulasi Ram Das, to address the quality concerns in technical education through empowerment and capacity building of the faculty. The programme provides in house opportunity for faculty to gain hands-on experience by practically doing experiments in the laboratories of the parent departments. The programme is being implemented in all the affiliated colleges of JNTU K to help the faculty to review and broaden their understanding of the practical aspects of the theoretical knowledge imparted by them to the students.

Objectives and Benefits

1. To mobilize and motivate the faculty to get familiarity with all the experiments of the apparatus, machinery, equipment, set up and facilities available in each laboratory of the parent departments
2. To broaden the understandingof the link between the theory and practice by making the faculty to do experiments
3. To help build the capacity of the faculty such that they can handle the laboratories of not only their specialization but also other specializations in the same department.
4. To serve indirect purpose of checking the working condition and maintenance of the apparatus, machinery, equipment, set up and facilities in the laboratories.
5. Weightage will be given in the ratifications to the faculty participating in this programme
6. Benefit to the student in instructions of relevance, importance and appreciation of experiments delivered by teachers.

The Organization

1. The Directorate of Faculty Development, JNTU K will organize, supervise and co-ordinate the programme " **Hands-on Experience for Faculty in Laboratories**" Phase I
2. All the I Semester Laboratories of 21 Departments are covered in the programme in Phase I as per details given in **Annexure 1**. The II Semester Laboratories will be covered in Phase II.
3. The programme is conducted in the departmental laboratories in all the affiliated colleges
4. The reporting mechanism, communication and the ownership will be as noted below



The implementation

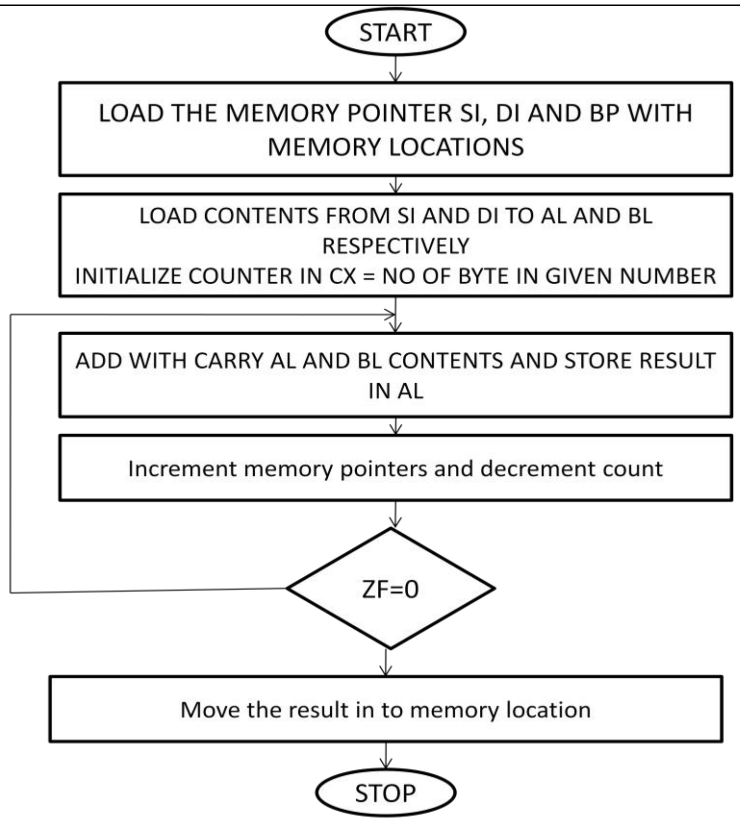
1. It is mandatory for all the faculty teachingUG courses in all affiliated colleges of JNTUK to participate in the programme(The Principal and HOD's are exempted as they have to monitor the programme)
2. The laboratories of 21 departments given in **Annexure 1**, are included in the programme.
3. The faculty will conduct all the possible experiments according to R13 and R10 syllabi, on the apparatus, machinery, equipment, set up and facilities available in each laboratory of their departments.
4. The Heads of the Departments (HOD's) need to create awareness about the importance of the programme among the faculty members of their departments and encourage them to participate in the programme
5. The HOD's take lead to create necessary environment and make required arrangements in the department to implement the programme.
6. The Principal shall send the list of faculty who have not participated in the programme, along with the explanations for non-compliance

The Duration of Programme and Report Submission

1. The "**Hands-on Experience**" programme shall be conducted and completed in all aspects from 1.5.2014 to 30.6.2014.
2. **Conduct of Experiments** :The faculty will conduct the experiments using observation note books. They shall record their observations, draw the graphs, and write all the relevant details in the observation note books. These shall be maintained for each lab separately and kept in the departments for inspection and verification by authorities of the University.
3. **Submission of Report** :The faculty will prepare report for each labon hard copy for submission to the University. The report shall be prepared as per the format enclosed on A4 size sheet. **The report for a lab with 10 experiments will have 11 papers(Cover page + 10 papers for ten experiments)**
4. The HOD's will collect the reports from faculty and submit them to The Principal. The Principal will in turn submit the reports to The Director (Faculty Development), JNTUK, Kakinada on or before 7.7.2014

The Queries

1. The queries can be sent to abbaiah@yahoo.com with the subject name of **Hands- on Experience-Query** for any clarifications
2. The queries will also be answered on calling **0884 2355677**

Name of Experiment	ASSEMBLY LANGUAGE PROGRAM TO PERFORM MULTI BYTE ADDITION
Importance of Experiment	To perform the addition of two multi byte numbers using 8086
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**ADC**- the instruction ADC is used to add two 8-bit or 16-bit numbers along with the content of carry flag, the addition result is stored in destination register and the Carry flag, overflow flag, sign flag, auxiliary carry flags gets affected. Any carry generated from addition is shown in carry flag. **LOOP**: The loop instruction uses count register CX as the no of iterations the loop has to run. It decrements CX register and if zero it breaks the branching. **INPUT:** 9988776655443322 1111111111111111 **OUTPUT:** AA99887766554433
Correlation of experimental outcome with theoretical concept	 START LOAD THE MEMORY POINTER SI, DI AND BP WITH MEMORY LOCATIONS LOAD CONTENTS FROM SI AND DI TO AL AND BL RESPECTIVELY INITIALIZE COUNTER IN CX = NO OF BYTE IN GIVEN NUMBER ADD WITH CARRY AL AND BL CONTENTS AND STORE RESULT IN AL Increment memory pointers and decrement count ZF=0 Move the result in to memory location STOP **ALGORITHM:** 1. Load the pointers with memory address of the two numbers and the result. 2. Clear carry flag for the first addition to be with no carry. 3. Load the count with the number of bytes in each number to perform the same number of additions. 4. Load AL and BL with the bytes from their address

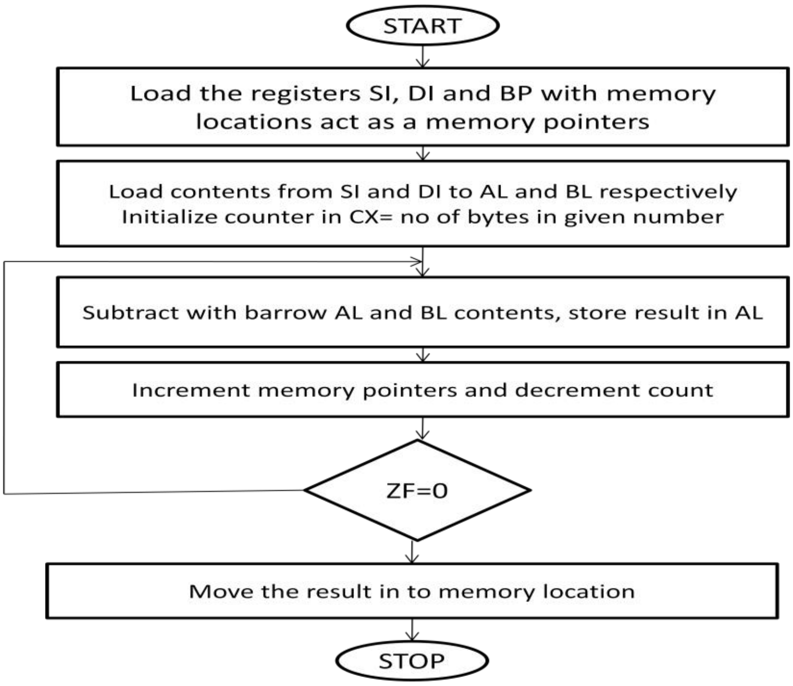
5. Perform addition with carry.
6. Store the sum in AL to result address.
7. Increment pointers so that they point to next byte of the number.
8. go to step 4 until the count is zero by decrementing the count and check for zero.
9. Preserve the carry flag of last addition to the result location to complete the addition.

program

```
            MOV SI,2350         ;address of first operand
            MOV DI,2750         ;address of second operand
            MOV BP,3000                 ;address of the result
            CLC                 ;clear the carry flag
            MOV CL,08           ;no of BYTES within the number
UP:         MOV AL,[SI]         ;word from the first operand
            MOV BL,[DI]         ;word from the second operand
            ADC AL,BL           ;addition of two bytes
            MOV [BP],AL                 ;sum is stored at the result
            INC SI              ;SI points to next word of first
operand
            INC DI              ;DI points to next word of second
operand
            INC BP                      ;BP points to next word of
result
            LOOP UP             ;decrement CL and jump to label
UP if CL≠0
            MOV AH,00           ;clear AL for carry
            JNC L1                      ;if CF=0 jump to label L1
            INC AH              ;increment Ah
L1:         MOV [BP],AL                 ;store the carry
            INT 03              ;invoking break-point interrupt
```

Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. Can be used as part of the ALU. 4. Can be used in address generation logic in processor design.
Can you design new experiment with this set up	Yes, can used as part of calculator experiment etc.
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	ALP PROGRAM TO MULTI BYTE SUBTRACTION
Importance of Experiment	To perform the Subtraction of two multi byte numbers using 8086
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**SBB-** the instruction SBB is used to perform the subtraction of subtrahend from minuend along with previous content of carry flag, the subtraction result is stored in destination register and the Carry flag, overflow flag, sign flag, auxiliary carry flags gets affected. Any borrow taken during the subtraction process is indicated in carry flag. **LOOP**: The loop instruction uses count register CX as the no of iterations the loop has to run. It decrements CX register and if zero it breaks the branching **INPUT:** 9988776655443322 1111111111111111 **OUTPUT:** 8877665544332211
Correlation of experimental outcome with theoretical concept	**Flowchart:**  START Load the registers SI, DI and BP with memory locations act as a memory pointers Load contents from SI and DI to AL and BL respectively Initialize counter in CX= no of bytes in given number Subtract with barrow AL and BL contents, store result in AL Increment memory pointers and decrement count ZF=0 Move the result in to memory location STOP **Algorithm:** 1. Load the pointers with memory address of the two numbers and the result. 2. Clear carry flag for the first subtraction to be with no borrow. 3. Load the count with the number of bytes in each number to perform the same number of byte subtractions. 4. Load AL and BL with the bytes from their address 5. Perform subtraction with borrow. 6. Store the difference in AL to result address. 7. Increment pointers so that they point to next byte of the number. 8. go to step 4 until the count is zero by decrementing the count and

check for zero.

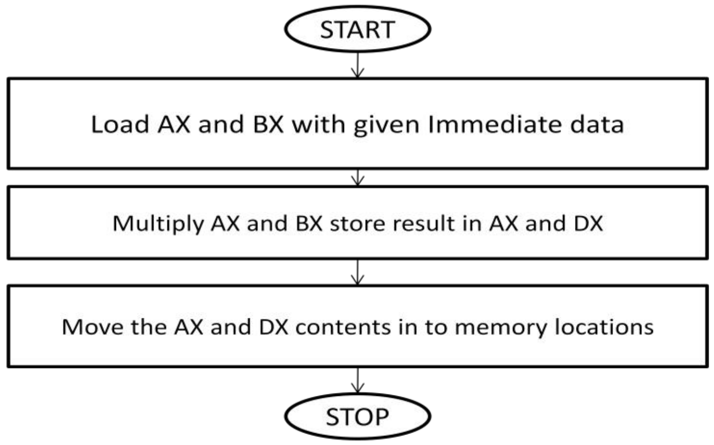
9. Preserve the carry flag of last subtraction to the result location to complete the multi-byte subtraction.

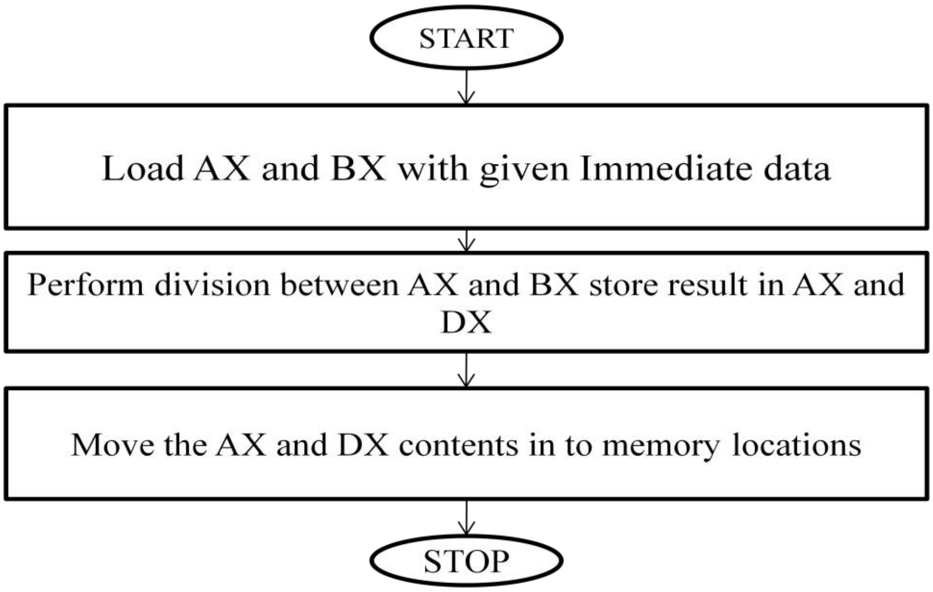
Program:

```
            MOV SI,2350         ;address of first operand
            MOV DI,2750         ;address of second operand
            MOV BP,3000                 ;address of the result
            CLC                 ;clear the carry flag
            MOV CL,08           ;no of BYTES within the number
UP:         MOV AL,[SI]         ;word from the first operand
            MOV BL,[DI]         ;word from the second operand
            SBB AL,BL           ; subtraction of two bytes
            MOV [BP],AL                 ;sum is stored at the result
            INC SI              ;SI points to next word of first
operand
            INC DI              ;DI points to next word of second
operand
            INC BP                      ;BP points to next word of
result
            LOOP UP             ;decrement CL and jump to label
UP if CL≠0
            MOV AH,00           ;clear AL for barrow
            JNC L1                      ;if CF=0 jump to label L1
            INC AH              ;increment AL
L1:         MOV [BP],AL                 ;store the barrow
            INT 03              ;invoking break-point interrupt
```

Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies.
Can you design new experiment with this set up	Yes, to be part of a ALU.
Is the experimental set up in working condition	yes

Signature of Faculty Member

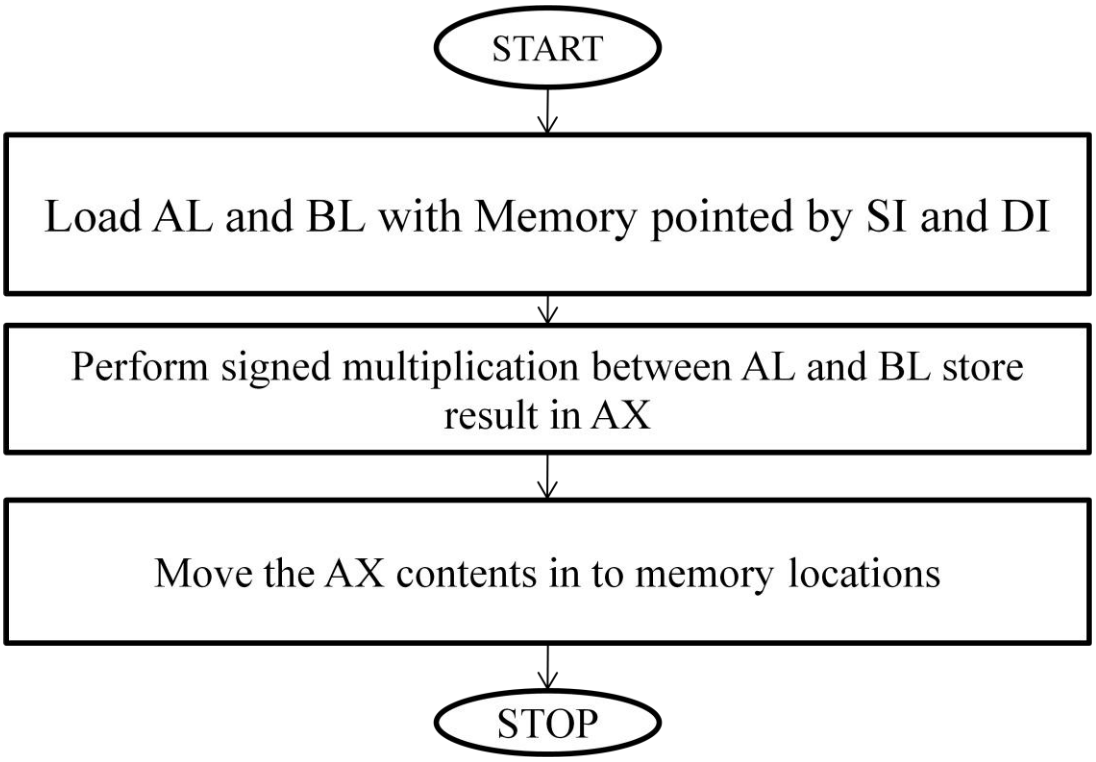
Name of Experiment	ALP PROGRAM TO 16 BIT MULTIPLICATION
Importance of Experiment	To perform the Subtraction of two multi byte numbers using 8086
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**MUL:** the instruction has implied register AL or AX as one of the operand for 8-bit or 16-bit multiplication. Any other general purpose register can be used as other number. The multiplication result is stored in AX or AX-DX for 8-bit or 16-bit multiplication. INPUT: 1234H X 1234H OUTPUT: 3000 - 5A90H 3002 - 014BH
Correlation of experimental outcome with theoretical concept	**Flowchart:**  START Load AX and BX with given Immediate data Multiply AX and BX store result in AX and DX Move the AX and DX contents in to memory locations STOP **Algorithm:** 1. Load AX and BX with numbers to be multiplied. 2. Multiply AX and BX using MUL instruction to get results in AX and DX 3. Move the content of AX and DX to the memory locations. **Program:** MOV AX,1234H ;load the ax register with first operand MOV BX,1234H ;load the bx register with second operand MUL BX ;perform multiplication of ax with bx MOV [3000],AX ;store the lower word of the result from ax into 3000 offset MOV [3002],DX ; store the higher word of the result from dx into 3002 offset. INT 03 ; invoke the break point interrupt
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. an important part of multiply and accumulate used in digital signal processors.
Can you design new experiment with this set up	Yes, can be used to be part of ALU, multiply and accumulate unit in DSP processors.
Is the experimental set up in working condition	yes

Name of Experiment	ALP PROGRAM TO DIVISION OF 32 BIT BY 16 BIT NUMBER
Importance of Experiment	To perform the division of 32 bit by 16 bit number using 8086 microprocessor
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**DIV:** the instruction DIV always takes AX or AX-DX for 16/8 or 32/16 division. The result of quotient is stored in AL or AX and reminder in AH or DX for 16/8 and 32/16 division respectively. INPUT: 3000 - 0008 3002 - 0000 4000 - 0002 OUTPUT: 5000 - 0004 5002 – 0000
Correlation of experimental outcome with theoretical concept	**Flowchart:**  START Load AX and BX with given Immediate data Perform division between AX and BX store result in AX and DX Move the AX and DX contents in to memory locations STOP **Algoirthm :** 1. Load the register AX and DX with 32bit dividend, load the 16bit divisor into BX. 2. Perform 32bit by 16bit division using DIV instruction. 3. Store the result of quotient in AX and reminder in DX to memory. **Program:** MOV SI,3000 ;loading the si register with 3000 offset address MOV DI,4000 ; loading the di register with 4000 offset address MOV BX,[DI] ; loading the bx register with divisor MOV AX,[SI] ; loading the ax with the lower word of dividend ADD SI,02 ; incrementing si by 2 to point to higher word of dividend MOV DX,[SI] ; loading the dx register with higher word of dividend DIV BX ; perform the double word by word division MOV SI,5000 ; load si with the offset address 5000 MOV [SI],AX ; load the remainder of division into 5000 address ADD SI,02 ; increment si to point to next word at 5002 address

	MOV [SI],DX ; store the quotient of division into 5002 address pointed by si INT 03
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3.an integral part of ALU.
Can you design new experiment with this set up	Yes, can be used as part of ALU.
Is the experimental set up in working condition	yes

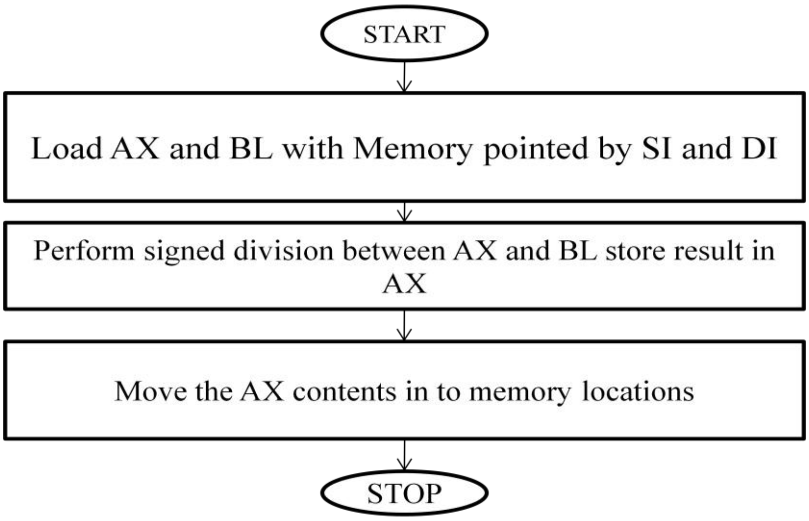
Signature of Faculty Member

Name of Experiment	MULTIPLICATION OF TWO 8-BIT SIGNED NUMBERS
Importance of Experiment	To perform the multiplication of two 8- bit signed numbers using 8086 microprocessor
Apparatus Required	1. ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**IMUL:** This instruction is going to perform the signed multiplication of two given signed numbers. The numbers are in signed 2's complement form. The instruction takes AL as the default register; the result is stored in AX register. INPUT: 3000 – 08H 3001 – 02H OUTPUT: 4000 – 10H

Correlation of experimental outcome with theoretical concept	**Flowchart:**
	 START
	Load AL and BL with Memory pointed by SI and DI
	Perform signed multiplication between AL and BL store result in AX
	Move the AX contents in to memory locations
	STOP
	ALGORITHM:
	1. Load the default register of the instruction that is AL and other register with the signed 8-bit numbers to be multiplied.
	2. Performing the signed multiplication using the IMUL instruction.
	3. Store the result from to memory location.
	
	PROGRAMS:
	`MOV SI,3000        ; the pointer si is loaded with offset address 3000`
	
	`MOV DI,3001        ; the pointer di loaded with offset address 3001`
	`MOV AL,[SI]        ; load al with the first operand from si pointer address`
	`MOV BL,[DI]        ; load bl with the second operand from di pointer address`
	`IMUL BL            ; perform signed multiplication of al with bl`
	`MOV SI,4000        ; load the si pointer with offset 4000`

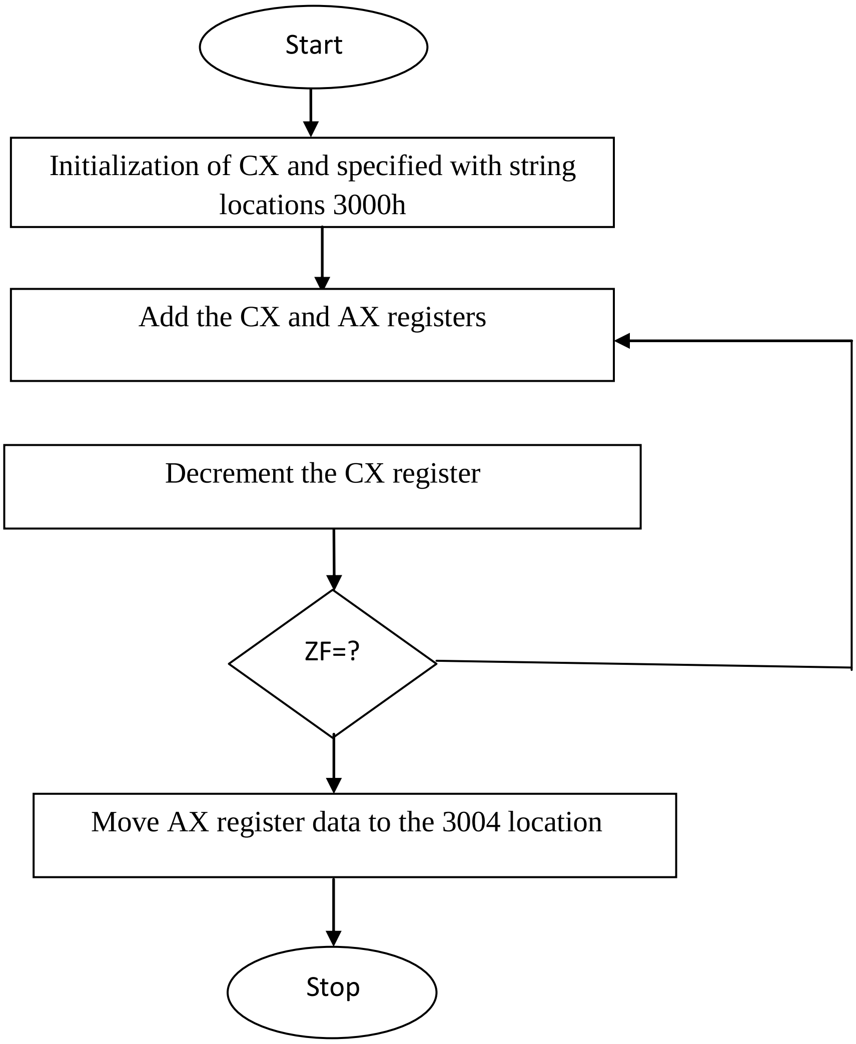
	MOV [SI],AX ; the signed multiplication result in ax is stored in 4000 offset INT 03 ; invoke the break point interrupt
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. can be a part of ALU.
Can you design new experiment with this set up	Yes, to be part of ALU.
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	DIVISION OF 16 BIT BY 8 BIT SIGNED NUMBER
Importance of Experiment	To perform the division of 16 bit by 8 bit signed numbers using 8086 microprocessor
Apparatus Required	1. ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**IDIV:** instruction has default register AX as dividend and other 8-bit register as divisor. The signed result of division has quotient in AL with the resulting sign of the division operation, reminder is in AH with the same sign as that of dividend. INPUT: 3000 – 0008H 4000 – 04H OUTPUT: 5000 – 02H 5001 – 00H
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**  START Load AX and BL with Memory pointed by SI and DI Perform signed division between AX and BL store result in AX Move the AX contents in to memory locations STOP **ALGORITHM:** 1. Load the default register AX with the dividend, and the divisor in to other 8-bit register. 2. Perform the signed division with the resulting quotient and reminder in AL and AH 3. Store the AX to a given memory locations. MOV SI,3000 ; load si with offset 3000 MOV DI,4000 ; load di with offset 4000 MOV AX,[SI] ; load ax with dividend from si pointer MOV BL,[DI] ; load bl with divisor from di pointer IDIV BL ; perform the signed division of word by byte MOV SI,5000 ; load the si with offset 5000 MOV [SI],AX ; store the signed division result from ax to si pointer location INT 03 ; invoke the break point interrupt.
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies.

	3. used as part of ALU.
Can you design new experiment with this set up	Yes, can be made part of ALU design.
Is the experimental set up in working condition	yes

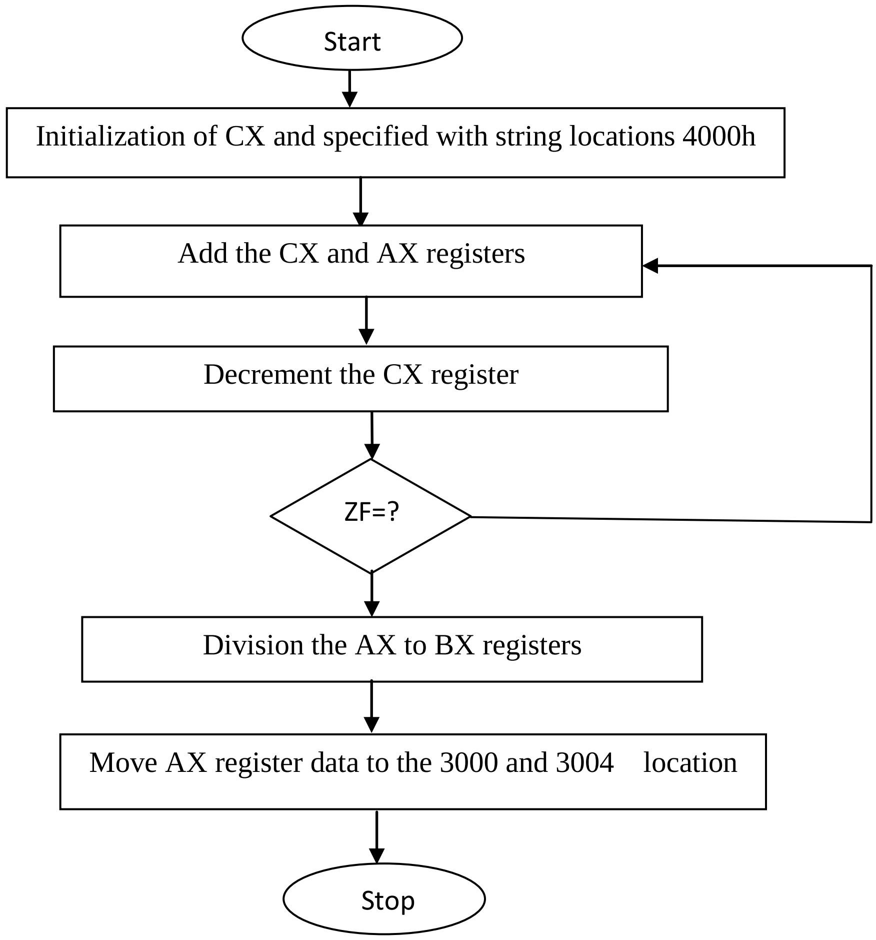
Signature of Faculty Member

Name of Experiment	ALP FOR SUM OF ‘N’ NATURAL NUMBERS
Importance of Experiment	To find the sum of n natural numbers using ESA-86/88 kit
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**DEC:** The instruction decrements the register or memory value, it effects the zero flag. **JNZ:** jump if not zero instruction checks for zero flag, if flag is set perform the jump to given location or else continue with next instruction. INPUT: 4000-0015H OUTPUT: 5000-00E7H
Correlation of experimental outcome with theoretical concept	FLOWCHART:  Start Initialization of CX and specified with string locations 3000h Add the CX and AX registers Decrement the CX register ZF=? Move AX register data to the 3004 location Stop **ALGORITHM:** 1. Enter the N natural numbers to be added.

	2. Empty the count value 3. Add the natural number to the count. 4. Decrement the natural number and if not zero goto step 3 else continue to step 5. 5. Store the sum result to memory. **PROGRAM:** MOV CX, [3000H] MOV AX, 0000H X: ADD AX, CX DEC CX JNZ X MOV [3004H], AX INT 03H
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. useful in statistical mathematical operations.
Can you design new experiment with this set up	Yes, can be used in mean, median, average of numbers programs.
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	AVERAGE OF ‘N’ NATURAL NUMBERS
Importance of Experiment	To find the average of n natural numbers using ESA-86/88 kit
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**DEC:** The instruction decrements the register or memory value, it effects the zero flag. **JNZ:** jump if not zero instruction checks for zero flag, if flag is set perform the jump to given location or else continue with next instruction. **DIV:** the division of 16/8 or 32/16 dividend divisor results in quotient and reminder of 8 or 16 bit respectively. INPUT: 4000-05H OUTPUT:5000-0003H 5002-0000H To observing the average of n natural numbers of the two decimal or hex numbers of the given
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**


Start
Initialization of CX and specified with string locations 4000h
Add the CX and AX registers
Decrement the CX register
ZF=?
Division the AX to BX registers
Move AX register data to the 3000 and 3004 location
Stop

ALGORITHM:

1. Load the total no of natural numbers to which the average is to be found.
2. Empty the sum value.
3. Add the natural number value to sum.
4. Decrement the value of natural number and if not zero goto step 3 or else continue with the next instruction.
5. Perform average by dividing with the value of total no of natural numbers.
6. Store the average result to memory.

PROGRAM:

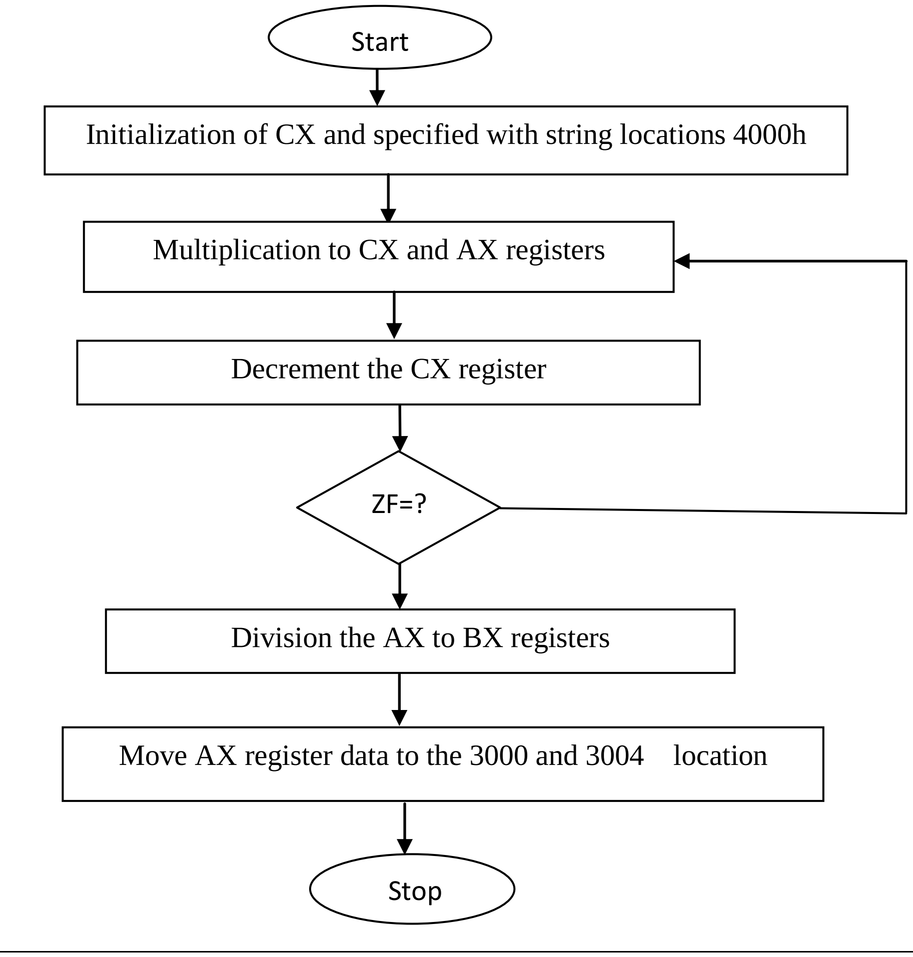
```
MOV CX, [4000H]
      MOV AX, 0000H
      MOV BX, CX
      MOV DX, 0000H
X:    ADD AX, CX
      DEC CX
      JNZ X
      DIV BX
      MOV [3000H], AX
      MOV [3004H], DX
       INT 03H
```

Practical Application

1. Commercial applications like simple calculator, toys, and remote cars etc.
2. And industrial wise large data processing and insurance companies.
3. used to developing the statistical mathematical operations.

Can you design new experiment with this set up	Yes
Is the experimental set up in working condition	yes

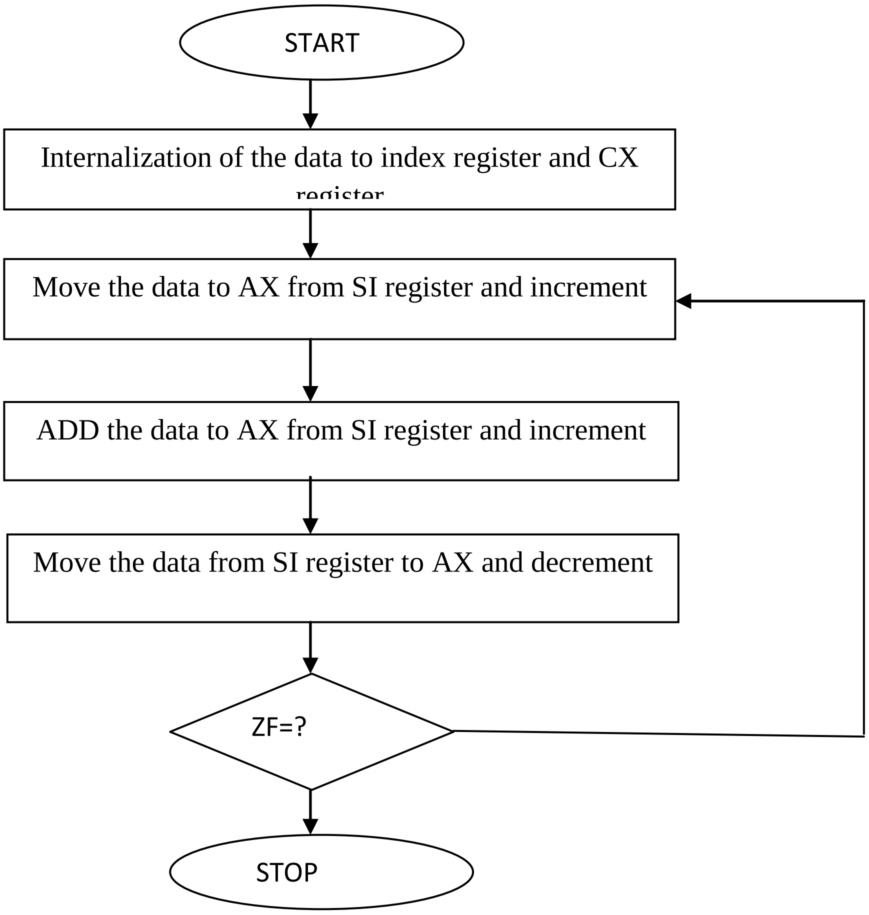
Signature of Faculty Member

Name of Experiment	**FACTORIAL OF A GIVEN NUMBER**
Importance of Experiment	Program to find the factorial of given number present in the memory location [3000H] store the result in [3002H].
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**DEC:** The instruction decrements the register or memory value, it effects the zero flag. **JNZ:** jump if not zero instruction checks for zero flag, if flag is set perform the jump to given location or else continue with next instruction. **MUL:** the instruction has implied register AL or AX as one of the operand for 8-bit or 16-bit multiplication. Any other general purpose register can be used as other number. The multiplication result is stored in AX or AX-DX for 8-bit or 16-bit multiplication. INPUT: 4000H-0003H OUTPUT: 5000H-0006H 5002H-0000H
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**  Start Initialization of CX and specified with string locations 4000h Multiplication to CX and AX registers Decrement the CX register ZF=? Division the AX to BX registers Move AX register data to the 3000 and 3004 location Stop

	ALGORITHM: 1. Clear the DX register and AX register as 1 for the factorial. 2. Load the number to which the factorial is to be calculated. 3. Multiply the number with the factorial result register AX. 4. Decrement the number, if not zero goto step 3 or else continue with next instruction. 5. Store the register result pair AX-DX to memory. **PROGRAM:** MOV DX, 0000H MOV CX, [3000H] MOV AX, 0001H L:MUL CX DEC CX JNZ L MOV [3002H], AX MOV [3004H], DX INT 03H
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. Can be used in mathematical applications like permutations, combinations and probability.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	FIBONACCI SERIES
Importance of Experiment	Write a program for Fibonacci series. Test the program in ESA-86/88 kit.
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**DEC:** The instruction decrements the register or memory value, it effects the zero flag. **JNZ:** jump if not zero instruction checks for zero flag, if flag is set perform the jump to given location or else continue with next instruction. **INC:** The instruction increments the register or memory value, it effects the zero flag. To observing Fibonacci series of decimal or hex numbers of the given INPUT: DS:SI: 00H,01H CX: 05H OUTPUT:00H,01H,01H,02H,03H,05H,08H
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**


START
Internalization of the data to index register and CX register
Move the data to AX from SI register and increment
ADD the data to AX from SI register and increment
Move the data from SI register to AX and decrement
ZF=?
STOP

ALGORITHM:

1. Store the initial numbers of the series in memory.
2. Load the number of elements of the series to be generated.
3. Load the number in the series.
4. Load the next number in the series.
5. Add the two numbers and store as the next number of the series.
6. Point to the previous number.
7. Decrement the count of numbers and if zero goto step 3 or else continue with the next instruction.
8. Stop the program.

PROGRAM:

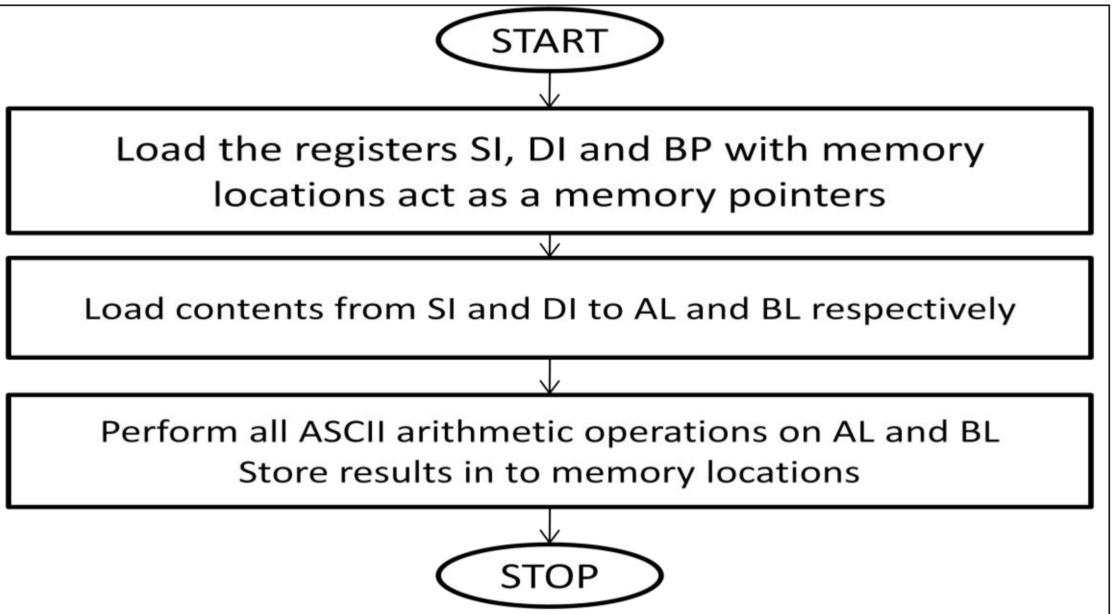
```
MOV SI, 3000H
       MOV CX, [4000H]
   L1: MOV AX, [SI]
       INC    SI
        INC SI
       ADD AX, [SI]
       INC SI
        INC SI
       MOV [SI], AX
       DEC SI
        DEC SI
       DEC CX
       JNZ   L1
        INT 03H
```

Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc.

	2. And industrial wise large data processing and insurance companies. 3. Fibonacci numbers are important in the computational run-time analysis of euclid's algorithm to determine the greatest common divisor of two integers- the worst case input for this algorithm is a pair of consecutive Fibonacci numbers. 4. Fibonacci numbers are used by some pseudorandom number generator.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	ASCII Arithmetic Operations
Importance of Experiment	To perform the ASCII Arithmetic Operations using 8086 microprocessor
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**AAA:** the ASCII arithmetic adjusts after addition is used to produce the ASCII result for addition of ASCII numbers. The two added numbers are to be in unpacked BCD form. The result of the sum and AAA will be in unpacked BCD form of the sum. **AAS:** the ASCII arithmetic adjusts after subtraction is used to produce the ASCII result for subtraction of ASCII numbers. The two subtracted numbers are to be in unpacked BCD form. The result of the subtraction and AAS will be in unpacked BCD form of the difference. **AAM:** the ASCII arithmetic adjusts after multiplication is used to produce the ASCII result for multiplication of ASCII numbers. The numbers to be multiplied are to be in unpacked BCD form. The result of the product and AAM will be in unpacked BCD form of the result. **AAD:** the ASCII arithmetic adjusts before division. The dividend and divisor are in unpacked BCD form. The dividend is adjusted to a form suitable for division. After division the result of quotient and reminder are in their unpacked form to be converted to ASCII with OR-ing with 30H. INPUT: 3000 – 37H 3001 – 32H OUTPUT: 5000 – 39H 5001 –30H 5002 –35H 5003 –30H 5004 –3EH 5005 –30H 5006 –33H 5007 –31H
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**


START
Load the registers SI, DI and BP with memory locations act as a memory pointers
Load contents from SI and DI to AL and BL respectively
Perform all ASCII arithmetic operations on AL and BL Store results in to memory locations
STOP

ALGORITHM:

1. Load the ASCII numbers in to AL and BL registers.
2. Perform addition using ADD instruction followed by AAA instruction to get ASCII adjusted result after addition.
3. The result in AX is in unpacked BCD form, convert the unpacked BCD to ASCII by ORing with 3030h.
4. Store the result in memory and increment the address for further results.
5. Reload the AL register with the ASCII number.
6. Perform subtraction using SUB instruction followed by AAS instruction to get ASCII adjusted result after subtraction.
7. Again the result is in unpacked BCD form, covert the unpacked BCD to ASCII by ORing with 3030h.
8. Store the result in memory location and increment the address for further results.
9. Reload the AL register with the ASCII number.
10. Perform the multiplication using MUL instruction followed by AAM to get the result to be in unpacked BCD form.
11. Again the result is in unpacked BCD form, covert the result to ASCII by ORing with 3030h.
12. Store the result in memory and increment the address for further results.
13. Load AX with the ASCII form of dividend.
14. Covert it to unpacked BCD by ANDing with 0F0Fh
15. Apply the AAD instruction to perform the ascii adjust before division.
16. Perform the division operation using the DIV instruction.
17. The result is in unpacked BCD form, perform ORing with 3030h
18. Store the result to memory locations.

PROGRAM:

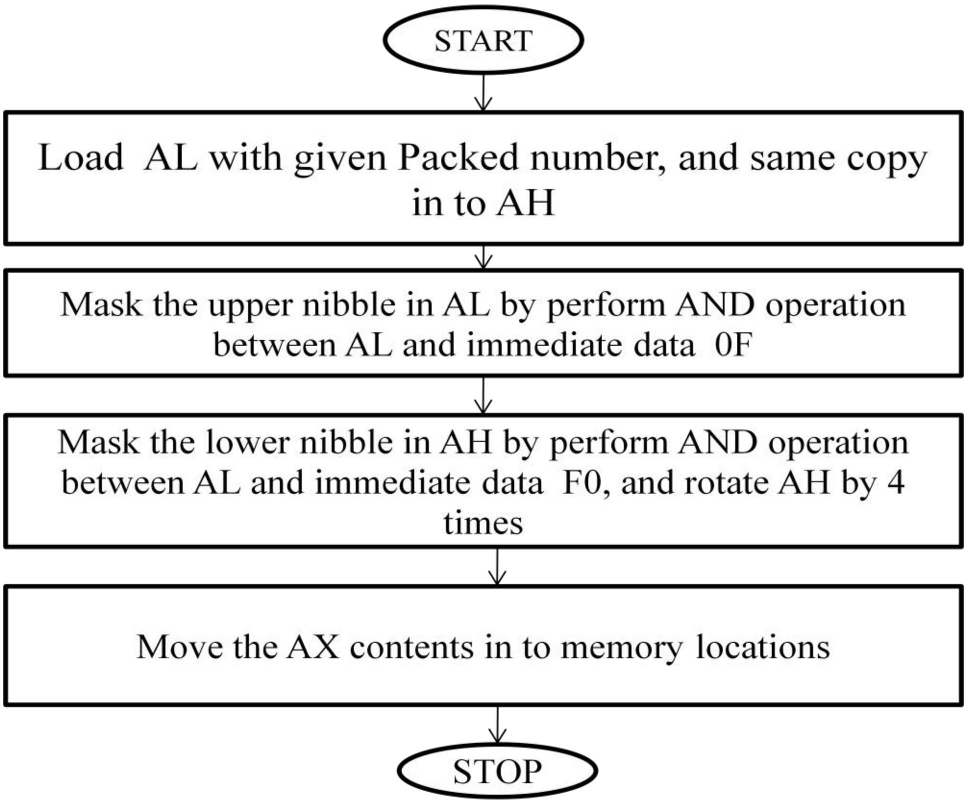
```
                MOV SI,3000             ;load si with the offset 3000
                MOV DI,3001             ; load di with offset 3001
                MOV AL,[SI]             ; load al with first ascii value
                MOV BL,[DI]             ; load bl with second ascii value
                ADD AL,BL               ; perform addition
                AAA                     ; ascii adjust after addition
                ORA AX,3030             ; perform or operation on the adjusted unpacked
bcd result
                MOV BP,5000             ; load the bp register with 5000 offeset address
                MOV [BP],AX             ; store the ascii addition result into bp pointer
address.
                MOV AL,[SI]             ; load again the first operand into al register
                SUB AL,BL               ; perform subtraction of bl from al
                AAS                     ; ascii adjust after subtraction
                ORA AX,3030             ; perform or operation on the adjusted unpacked
```

	bcd result ADD BP,02H ; point bp to the next word address MOV [BP],AX ; store the ascii subtraction result to bp pointer address MOV AL,[SI] ;load again the first operand into al register AND AL,OFH ; convert the ascii number into unpacked bcd number AND BL,OFH ; convert the ascii number into unpacked bcd number MUL BL ; perform 8-bit multiplication of al with bl AAM ; ascii adjust after multiplication ORA AX,3030 ; perform or operation on the adjusted unpacked result ADD BP,02 ; point bp to next word address MOV [BP],AX ;store the result of ax into the bp pointer address MOV AX,0302 ; load ax with the unpacked bcd form of the dividend AAD ; ascii adjust before division DIV BL ; perform division of ax with bl ORA AX,3030 ; perform or operation on division result for proper ascii value ADD BP,02H ; point bp to the next word address MOV [BP],AX ; store the ascii division result into bp pointed address INT 03 ; invoke breakpoint interrupt
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3.arithemetic operation from ASCII character devices like LCD, monitor, keyboard etc
Can you design new experiment with this set up	Yes, calculator with input from ASCII keyboard.
Is the experimental set up in working condition	yes

Signature of Faculty Member

<table>
<tr><td><b>Name of Experiment</b></td><td>Conversion from packed BCD to unpacked BCD</td></tr>
<tr><td><b>Importance of Experiment</b></td><td>To write an ALP program to convert packed number in to unpacked number using 8086 microprocessor</td></tr>
<tr><td><b>Apparatus Required</b></td><td>1. ESA 86/88E V3 TRAINER BOARD.
2.Key board
3.Power supply</td></tr>
<tr><td><b>Inference /Outcome</b></td><td><b>AND:</b> this instruction is used to perform the bit-wise and operation between 8-bit and 16-bit numbers.
<b>ROR:</b> rotate right instruction works on AL or AX register with the bits in the register shift towards right, the shifted out LSB is loaded as MSB bit. The numbers of shifts are given by the count register CX.
INPUT: 3000 -43H
OUTPUT: 4000 -04H
4001 -03H</td></tr>
<tr><td><b>Correlation of experimental outcome with theoretical concept</b></td><td>

<b>FLOWCHART:</b>


START
Load AL with given Packed number, and same copy in to AH
Mask the upper nibble in AL by perform AND operation between AL and immediate data 0F
Mask the lower nibble in AH by perform AND operation between AL and immediate data F0, and rotate AH by 4 times
Move the AX contents in to memory locations
STOP

<b>ALGORITHM:</b>

1. Load the packed BCD to AL register, copy the same to AH
2. Separate the lower nibble by ANDing the AL with 0Fh
3. Using the rotate right instruction on AH swap the position of lower and upper nibble.
4. Perform the AND operation with 0Fh on AH register.
5. Store the AX register to memory.

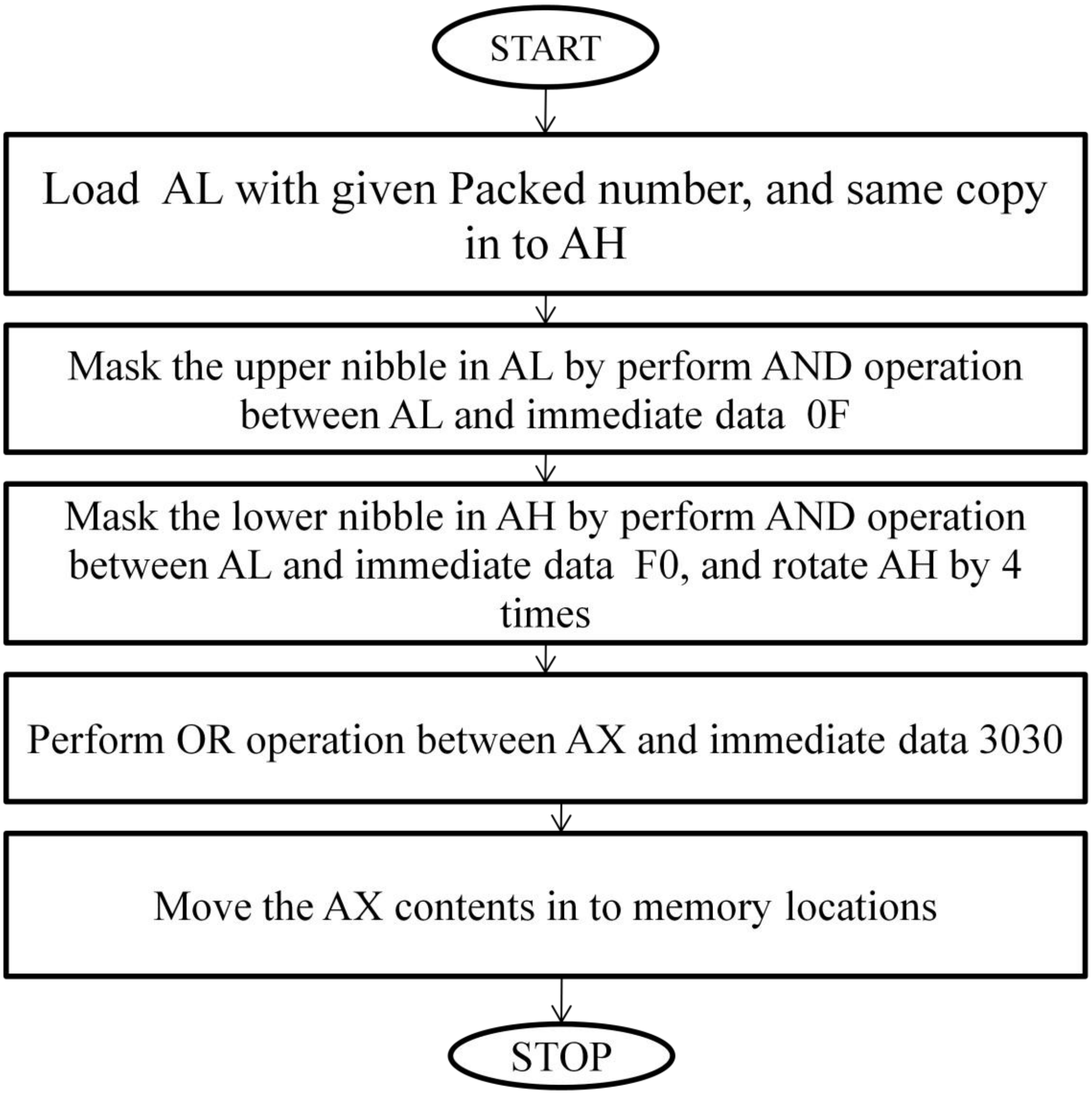
<b>PROGRAMS:</b>

```
MOV AL,[3000]        ;load al with the content of 3000 address
MOV AH,AL            ; load ah with the value of al
AND AL,0FH           ; perform AND operation to remove higher
```

</td></tr>
</table>

	nibble of al register MOV CL,04H ; initialize the counter with 4 AND AH,F0H ; perform AND operation to remove lower nibble of ah register ROR AH,CL ; rotate right ah with the counter as no of iterations MOV [4000],AX ; store the unpacked result into 4000 address INT 03 ; invoke the breakpoint interrupt
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. can be used to provide data display as displays only require the BCD numbers in the unpacked form.
Can you design new experiment with this set up	Yes, display interface to 7-segment display.
Is the experimental set up in working condition	yes

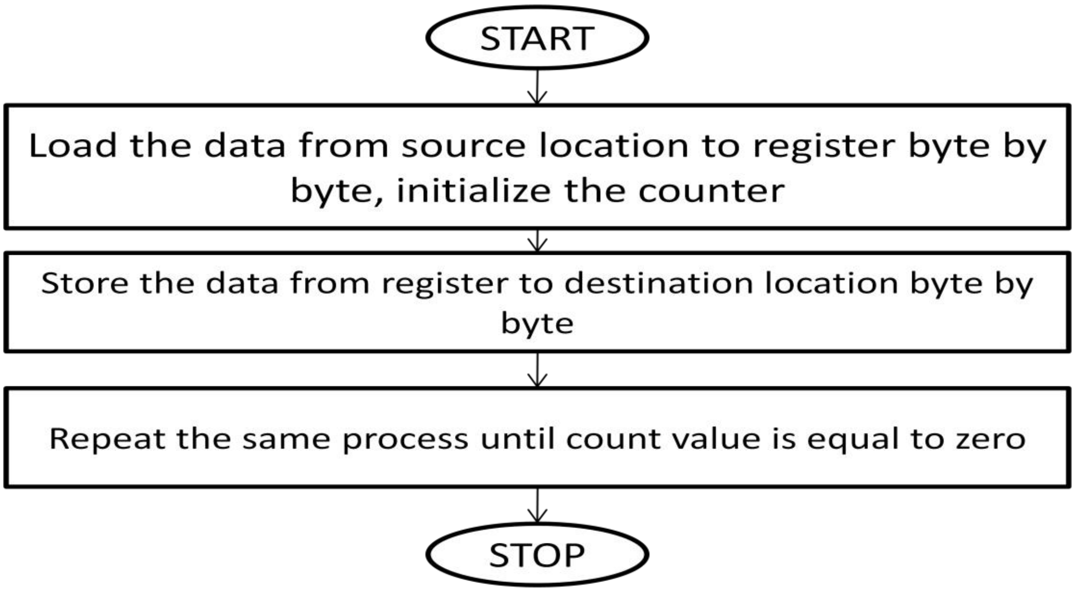
Signature of Faculty Member

Name of Experiment	Conversion from BCD to ASCII
Importance of Experiment	To write an ALP program to convert 8 bit packed BCD number in to ASCII number using 8086 microprocessor
Apparatus Required	1. ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	**AND:** this instruction is used to perform the bit-wise and operation between 8-bit and 16-bit numbers. **ROR:** rotate right instruction works on AL or AX register with the bits in the register shift towards right, the shifted out LSB is loaded as MSB bit. The numbers of shifts are given by the count register CX. **OR:** this instruction is used to perform the bit-wise and operation between the 8 bit and 16 bit numbers. INPUT: 3000 -43H OUTPUT: 4000 -34H 4001-33H
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**  START Load AL with given Packed number, and same copy in to AH Mask the upper nibble in AL by perform AND operation between AL and immediate data 0F Mask the lower nibble in AH by perform AND operation between AL and immediate data F0, and rotate AH by 4 times Perform OR operation between AX and immediate data 3030 Move the AX contents in to memory locations STOP **ALGORITHM:**

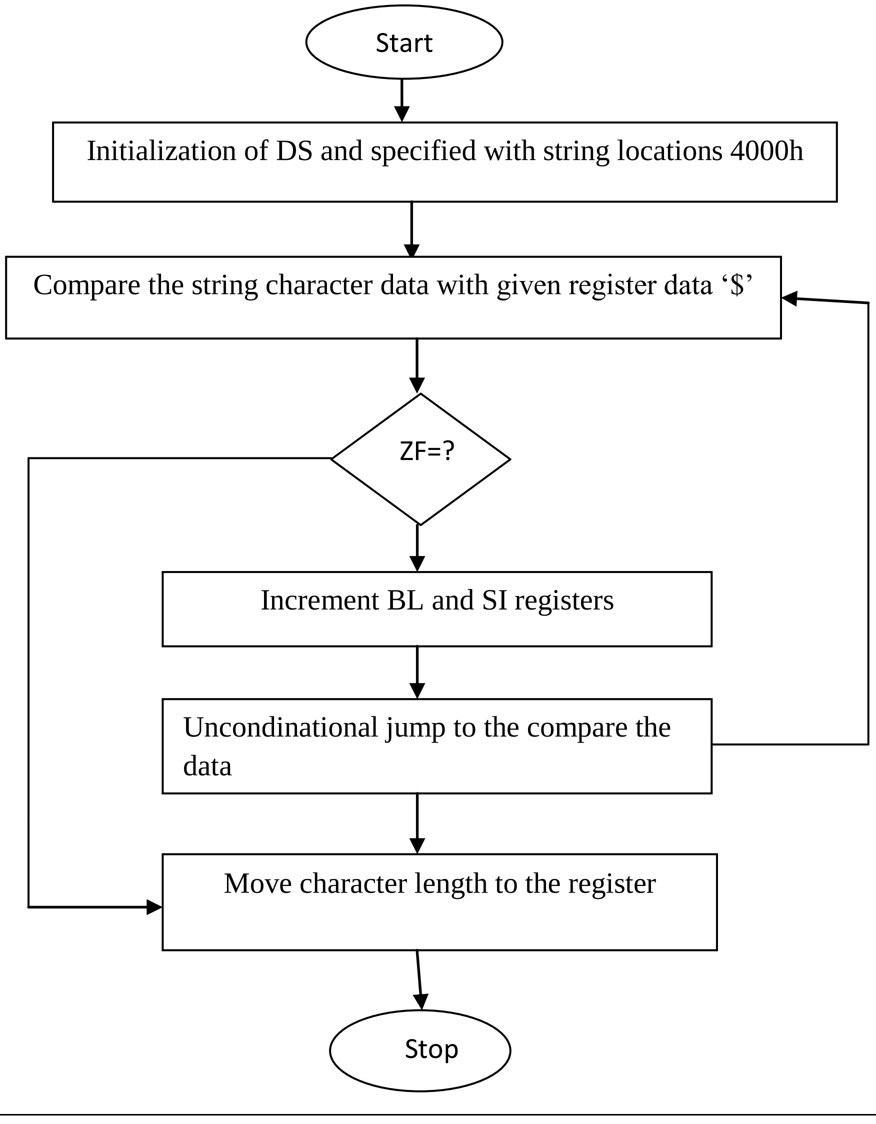
	1. Load the required BCD into the AL register, copy the same to AH register. 2. Convert the packed BCD to unpacked BCD. 3. Perform the ANDing operation on AL with 0Fh to obtain lower nibble. 4. Perform the rotate right operation on AH with rotations of 4 to obtain the higher nibble. 5. The result is in unpacked BCD form, to convert to ASCII, the lower and higher nibbles are to be ORed with 30h. 6. Store the resultant ASCII value to memory. **PROGRAM:** MOV AL,[3000] ;load al with the content of 3000 address MOV AH,AL ; load ah with the value of al AND AL,0FH ; perform AND operation to remove higher nibble of al register MOV CL,04H ; initialize the counter with 4 AND AH,F0H ; perform AND operation to remove lower nibble of ah register ROR AH,CL ; rotate right ah with the counter as no of iterations OR AX,3030 ; perform or operation on ax to convert to ascii form MOV SI,4000 ; load si with the address MOV [SI],AX ; store the ascii result to the si pointed address INT 03 ; invoke the breakpoint interrupt
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. For output device of ASCII standard the standard decimal is to be provided by using BCD to ASCII conversion.
Can you design new experiment with this set up	Yes, can be used in display of numbers in monitor with ASCII character set.
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	Moving a Block of 10bytes
Importance of Experiment	To write an ALP program to move a block of memory from one memory location to another memory location
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	To observe Moving a Block of 10bytes of decimal or hex numbers of the given **CLD:** Auto increment of the SI &DI starting addressing locations. **REP MOVSB:** Repeat the move string byte location to the destination loctation until the CX =0. INPUT: 3000 – 01 OUTPUT: 4000 – 01 3001 – 02 4001 – 02 3002 – 03 4002 – 03 3003 – 04 4003 – 04 3004 – 05 4004 – 05 3005 – 06 4005 – 06 3006 – 07 4006 – 07 3007 – 08 4007 – 08 3008 – 09 4008 – 09 3009 – 0A 4009 – 0A

Correlation of experimental outcome with theoretical concept	**FLOW CHART:**  START Load the data from source location to register byte by byte, initialize the counter Store the data from register to destination location byte by byte Repeat the same process until count value is equal to zero STOP **ALGORITHM:** 1. Move the string data to the starting address in the default DS-SI and DI of the ES. 2. Load the CX register with the no of string bytes to be moved. 3. Clear direction flag to have auto increment of the addresses for every successive byte transfer. 4. Repeat the moving string byte data from source to destination until the count is zero. **PROGRAM:** MOV SI,4000 ; load the si with the offset address of the source string MOV DI,5000 ; load the di with the offset address of the destination string MOV CX,000a ; load counter register with the number of elements of the string CLD ; clear direction flag to auto increment the string pointer address REP ; repeat prefix to continue the operation till counter is null MOVSB ; perform move cx string bytes INT 03 ; invoke the break point interrupt
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. used to transfer large data in data structures etc
Can you design new experiment with this set up	Yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	LENGTH OF THE STRING
Importance of Experiment	To write an ALP program to length of the given string byte using MASM software
Apparatus Required	1.Personal Computer 2.MASM/TASM Software Installed
Inference /Outcome	**CMP:** the compare instruction works same as subtraction except that the result is only going to effect the flags not the registers. **JZ:** jump if zero, if zero flag is set the instruction direction to a branched address. **JMP:** this is an unconditional jump instruction. INPUT: DS: SI - MICROPROCESSORS OUTPUT: D DS:SI -LEN – 0FH
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**  Start Initialization of DS and specified with string locations 4000h Compare the string character data with given register data '$' ZF=? Increment BL and SI registers Uncondinational jump to the compare the data Move character length to the register Stop

ALGORITHM:

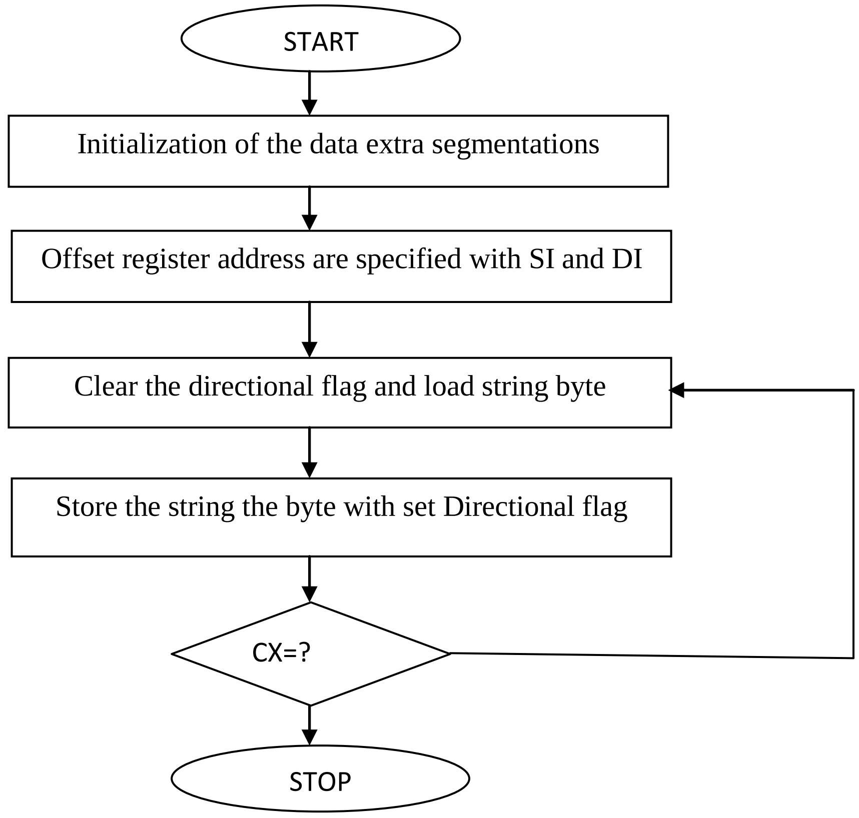
1. Store the required string whose length is to be found.
2. Load the address of the string into DS:SI, set count as zero.
3. Load the byte to AL and compare with end of string character $.
4. If zero flag set, we have reached the end of the string, else continue
5. Increment the address to next location and increment the count.
6. Go to step 2
7. Break the program to stop the execution.

PROGRAM:

```
        ASSUME CS:CODE,DS:DATA
        DATA SEGMENT
        ORG 4000H
        STR1 DB “MICROPROCESSORS$”
        LEN DB 01 DUP(0)
        DATA ENDS
        CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        LEA SI,STR1
        MOV AL,’$’
L2:     CMP AL,[SI]
        JZ L1
        INC BL
        INC SI
        JMP L2
L1:     MOV LEN,BL
        INT 03
        CODE ENDS
        END START
```

Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3.Display boards ,controlling signals in traffic light .
Can you design new experiment with this set up	Yes, for dynamic allocation of a memory, the length of the data has to be determined.
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	Reverse a given byte string
Importance of Experiment	To write an ALP program to reverse the given string byte using MASM software
Apparatus Required	1.Personal Computer 2.MASM/TASM Software Installed
Inference /Outcome	**LODSB:** The instruction LODSB is used to load the AL register with the byte pointed by the base and offset address pair DS:SI, after loading the SI content is incremented or decremented automatically according to the direction flag (DF). **STOSB:** The instruction STOSB is used to store the content of AL register to the address pointed by the ES:DI pair, after loading the DI content is incremented or decremented automatically according to the direction flag (DF). INPUT: DS: SI - MICROPROCESSORS OUTPUT: ES: DI – SROSSECORPORCIM
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**  START Initialization of the data extra segmentations Offset register address are specified with SI and DI Clear the directional flag and load string byte Store the string the byte with set Directional flag CX=? STOP

ALGORITHM:

1. Load the DS:SI and ES:DI with the source and destination addresses.
2. Load the count with the length of source string which is to be reversed and stored in destination.
3. Set the destination address at the end of string by adding the length to the starting address.
4. Clear direction flag and load the byte from source string using LODSB instruction.
5. Set direction flag and store the byte to destination string using STOSB instruction.
6. Decrement the length count and if not zero go to step 2
7. Break the program to stop the execution.

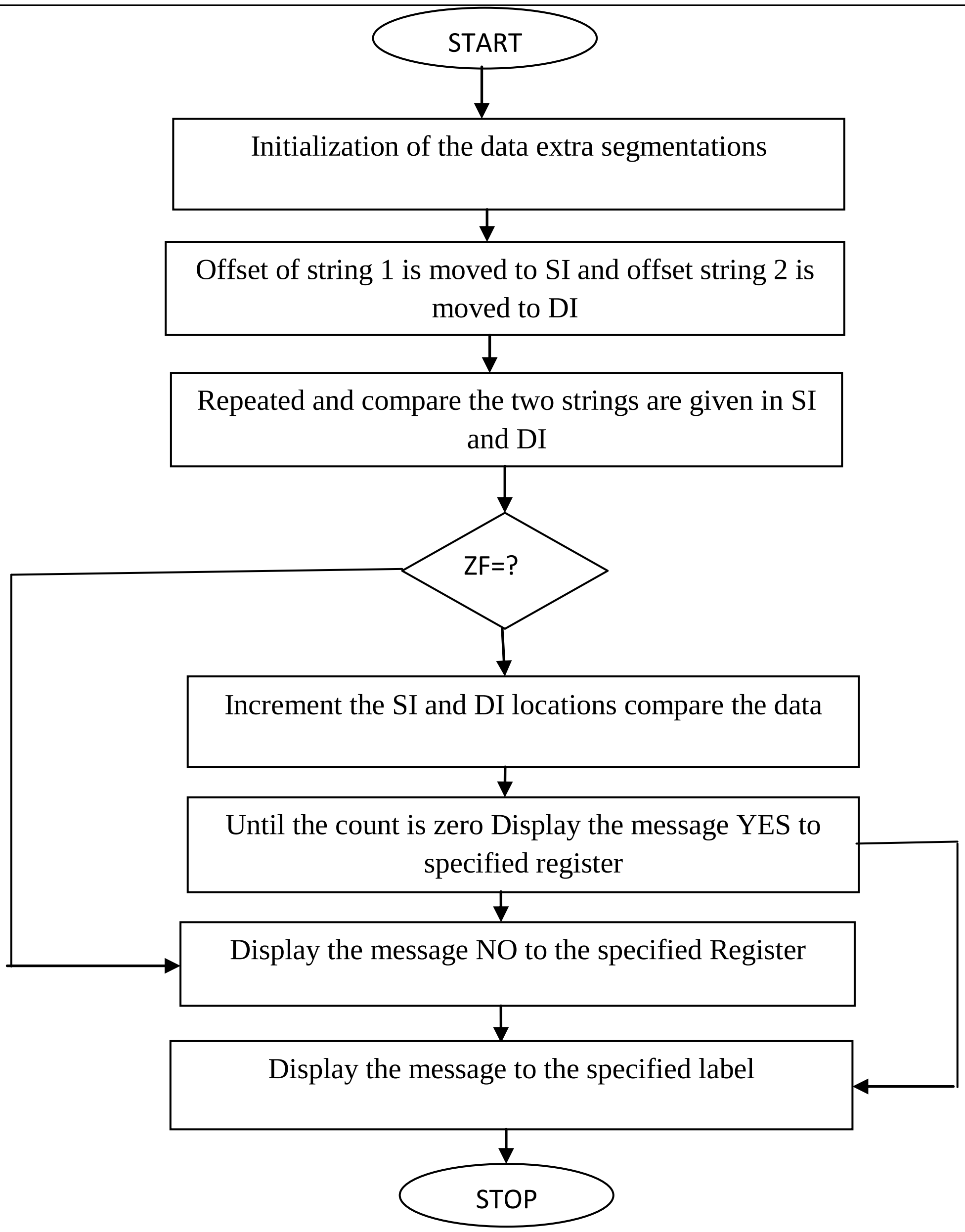
PROGRAM:

```
        ASSUME CS:CODE,DS:DATA,ES:DATA
        DATA SEGMENT
        ORG 4000H
        STR1 DB “MICROPROCESSORS”
        LEN EQU ($-STR1)
        ORG 5000H
        STR2 DB 20H DUP(00)
        DATA ENDS
        CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV ES,AX
        LEA SI,STR1
        MOV DI,OFFSET STR2
        ADD DI,LEN-1
        MOV CX,LEN
TI:     CLD
        LODSB
        STD
        STOSB
        LOOP TI
        INT 03
        CODE ENDS
        END START
```

Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3.Display boards ,controlling signals in traffic light .
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	STRING COMPARISION
Importance of Experiment	To write an ALP program to compare given two string bytes, and find out greater among them using MASM software
Apparatus Required	1.Personal Computer 2.MASM/TASM Software Installed
Inference /Outcome	**CMPSB:** The instruction CMPSB is used to compare each byte in their respective position of two strings**.** after comparison the compare takes decision to continue the next compare based on prefix REPE or REPNE as if the two string bytes are equal or not equal to repeat the next byte comparison. The two string addresses are loaded into the DS:SI and ES:DI pair. The comparison breaks only if the prefix fails or the count is zero which is decremented with each iteration of comparison. INPUT: DS: SI - RAMAKRISHNA ES: DI – RAMKUMAR OUTPUT: BX- 01H
Correlation of experimental outcome with theoretical concept	**FLOWCHARTS:**


START
Initialization of the data extra segmentations
Offset of string 1 is moved to SI and offset string 2 is moved to DI
Repeated and compare the two strings are given in SI and DI
ZF=?
Increment the SI and DI locations compare the data
Until the count is zero Display the message YES to specified register
Display the message NO to the specified Register
Display the message to the specified label
STOP

ALGORITHMS:

1. Two strings to be compared are placed in memory.
2. The address of two strings are loaded into DS:SI and ES:DI respectively.
3. Provide the length of the smallest string as the count.
4. Clear direction flag and perform the compare string byte repeatedly until the each byte in their corresponding positions are equal.
5. A string is said to be first in lexical order by determining if first string is greater than second, else the second string is first in lexical order.
6. The result indicating the first or second or equal status in lexical order with respect to first string is stored in memory.
7. Break the program to stop the execution.

PROGRAM:

```
ASSUME CS:CODE,DS:DATA,ES:DAT
                    DATA SEGMENT
                    ORG 3000H
                    STR DB “RAMAKRISHNA”
                    LEN EQU $-STR
                    DATA ENDS
                    DAT SEGMENT
                    ORG 4000H
                    STR1 DB “RAMKUMAR”
                    ORG 5000H
                    STAT DB 01 DUP(00)
                    DAT ENDS
                    CODE SEGMENT
```

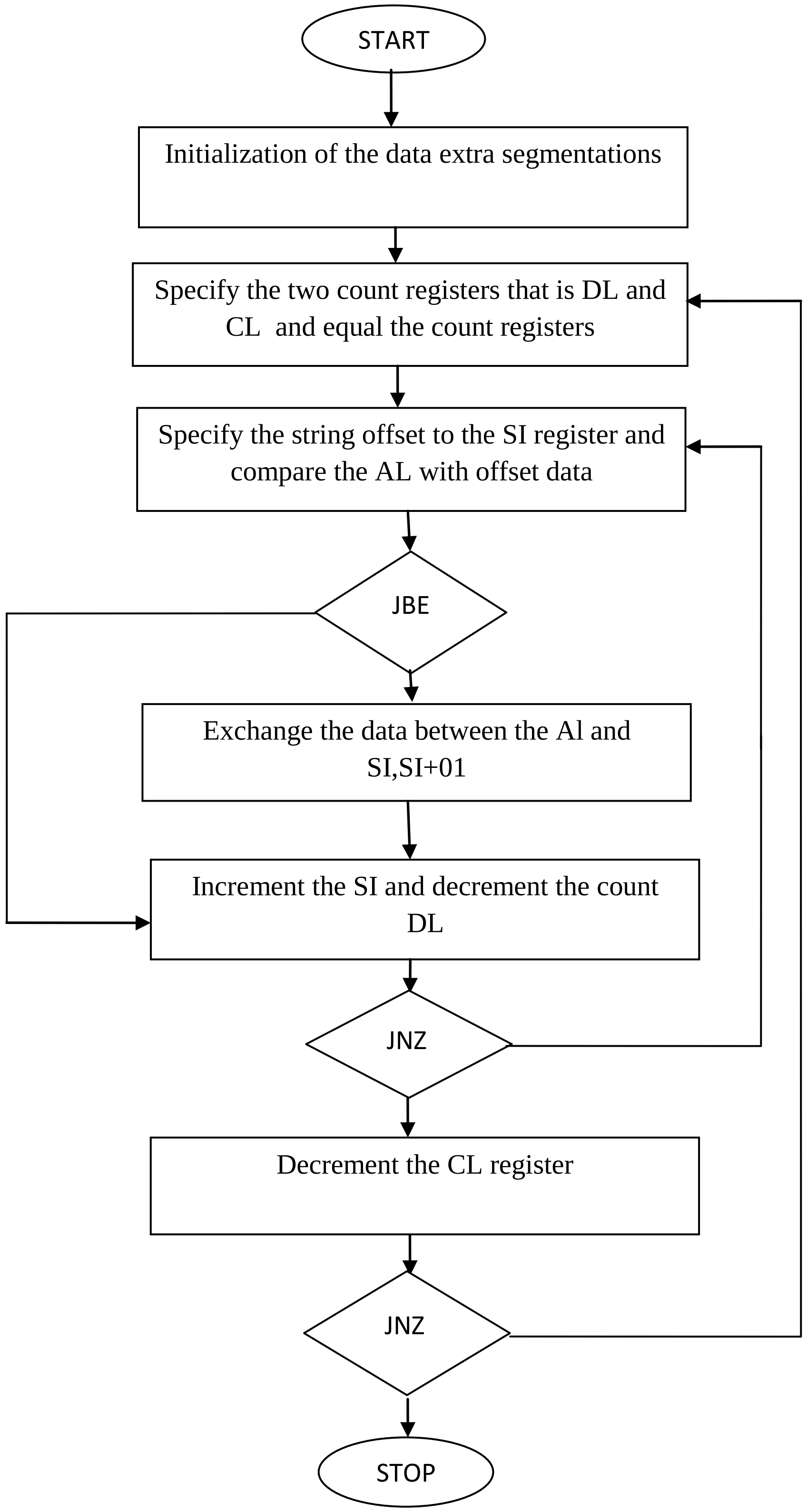
<table>
<tr><td></td><td><pre>
       START: MOV AX,DATA
              MOV DS,AX
              MOV AX,DAT
              MOV ES,AX
              MOV BL,00
              LEA SI,STR
              LEA DI,STR1
              MOV CX,LEN
              CLD
              REPE CMPSB
              JA TI
              JE EN
              INC BL
TI: INC BL
EN: MOV SI,OFFSET STAT
              MOV ES:[SI],BL
              INT 03
              CODE ENDS
              END START
</pre></td></tr>
<tr><td><b>Practical Application</b></td><td>1. Commercial applications like simple calculator, toys, and remote cars   etc.
2. And industrial wise large data processing and insurance companies.

3. Display boards, controlling signals in traffic light.

4.password checking</td></tr>
<tr><td><b>Can you design new experiment with this set up</b></td><td>Yes, in password checking program.</td></tr>
<tr><td><b>Is the experimental set up in working condition</b></td><td>yes</td></tr>
</table>

Signature of Faculty Member

Name of Experiment	ASCENDING ORDER
Importance of Experiment	To write an ALP program to sort the given string in ascending order using MASM software
Apparatus Required	Personal Computer MASM/TASM Software Installed
Inference /Outcome	**XCHG:** the instruction XCHG is used to exchange the content of two register of byte or word size. **CMP:** the compare instruction works same as subtraction except that the result is only going to affect the flags not the registers. **JBE:** this instruction checks for CF=1 or ZF=1, if true then it branches to the location specified by location or else it goes with the following instruction. INPUT: DS: SI - D2H, 3EH, 76H, 12H, E3H, 44H, 2AH, 69H OUTPUT: DS: SI – 12H, 2AH, 3EH, 44H, 69H, 76H, D2H, E3H
Correlation of experimental outcome with theoretical concept	FLOWCHART:


START
Initialization of the data extra segmentations
Specify the two count registers that is DL and CL and equal the count registers
Specify the string offset to the SI register and compare the AL with offset data
JBE
Exchange the data between the Al and SI,SI+01
Increment the SI and decrement the count DL
JNZ
Decrement the CL register
JNZ
STOP

ALGORITHM:

1. Load the outer count as length of the un-ordered numbers.
2. Load the inner count with the outer count.
3. Load the starting address of the un-ordered number string.
4. Compare the current location value with that of the next location value.
5. Check whether the first value is less than or equal to second value. If true goto step 7, else goto step 6.
6. Exchange the values between current and next locations.
7. Point to next location.
8. Decrement inner count and if non zero goto step 4 or else continue with next instruction.
9. Decrement outer count and if non zero goto step 2 or else continue with next instruction.

10. Terminate the program.

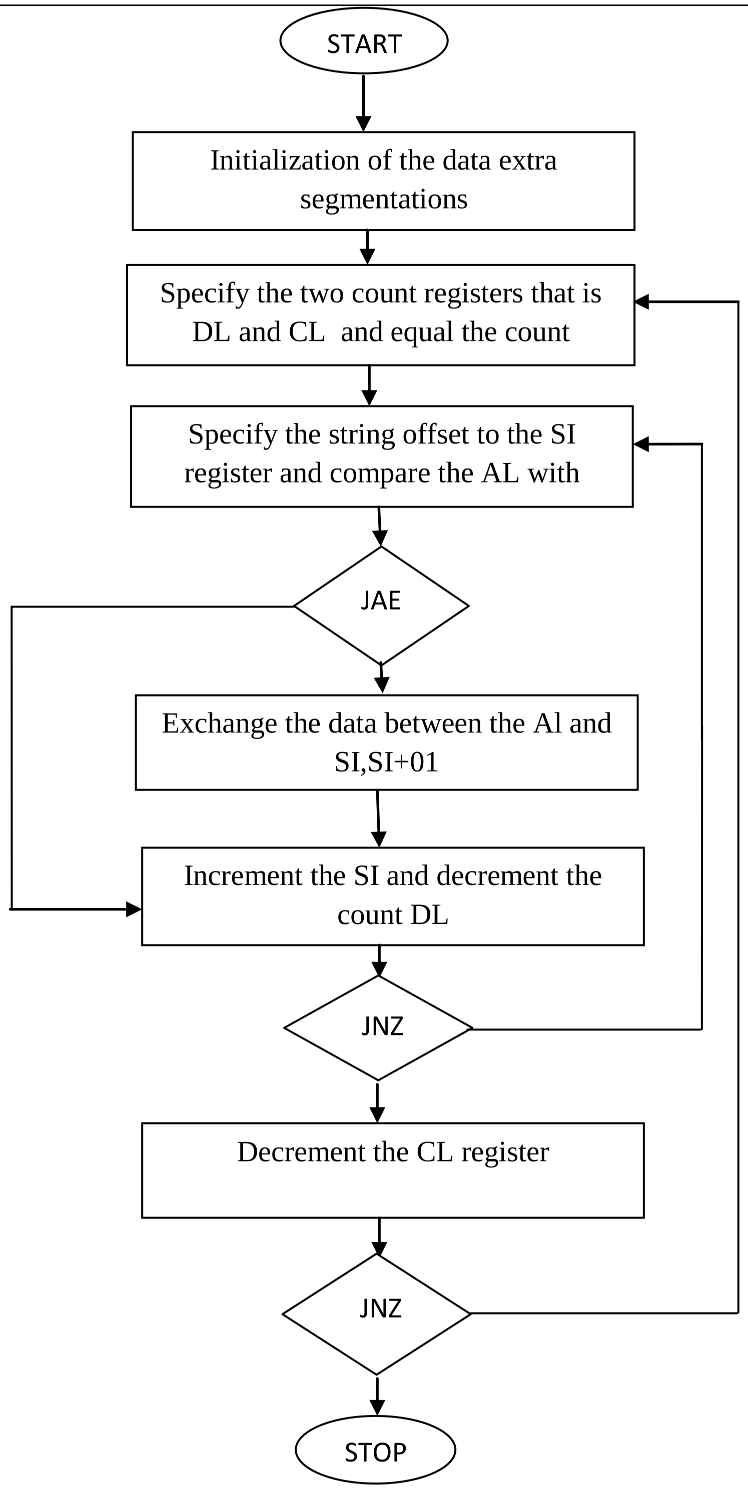
PROGRAM:

```
        ASSUME CS:CODE,DS:DATA
        DATA SEGMENT
        ORG 3000H
        STR DB 0D2H,3EH,76H,12H,0E3H,44H,2AH,69H
        LEN EQU ($-STR)
        DATA ENDS
        CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV CL,LEN-1
UPP:    MOV DL,CL
        LEA SI,STR
UP:     MOV AL,[SI]
        CMP AL,[SI+1]
        JBE TI
        XCHG AL,[SI+1]
        XCHG [SI],AL
TI:     INC SI
        DEC DL
        JNZ UP
        DEC CL
        JNZ UPP
        INT 03
        CODE ENDS
        END START
```

Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3.Display boards ,controlling signals in traffic light . 4. Can be used as part of sorting and searching in data structure.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	DESCENDING ORDER
Importance of Experiment	To write an ALP program to sort the given string in descending order using MASM software
Apparatus Required	Personal Computer MASM/TASM Software Installed
Inference /Outcome	**XCHG:** the instruction XCHG is used to exchange the content of two register of byte or word size. **CMP:** the compare instruction works same as subtraction except that the result is only going to affect the flags not the registers. **JAE:** this instruction checks for CF=0 or ZF=1, if true then it branches to the location specified by location or else it goes with the following instruction. INPUT: DS: SI - D2H, 3EH, 76H, 12H, E3H, 44H, 2AH, 69H OUTPUT: DS: SI – E3H, D2H, 76H, 69H, 44H, 3EH, 2AH, 12H To observe the sorting of the given data in descending order and also check with practical results.
Correlation of experimental outcome with theoretical concept	**FLOWCHART**


START
Initialization of the data extra segmentations
Specify the two count registers that is DL and CL and equal the count
Specify the string offset to the SI register and compare the AL with
JAE
Exchange the data between the Al and SI,SI+01
Increment the SI and decrement the count DL
JNZ
Decrement the CL register
JNZ
STOP

ALGORITHM:

1. Load the outer count as length of the un-ordered numbers.
2. Load the inner count with the outer count.
3. Load the starting address of the un-ordered number string.
4. Compare the current location value with that of the next location value.
5. Check whether the first value is greater than or equal to second value. If true goto step 7, else goto step 6.
6. Exchange the values between current and next locations.
7. Point to next location.
8. Decrement inner count and if non zero goto step 4 or else continue with next instruction.
9. Decrement outer count and if non zero goto step 2 or else continue with next instruction.
10. Terminate the program.

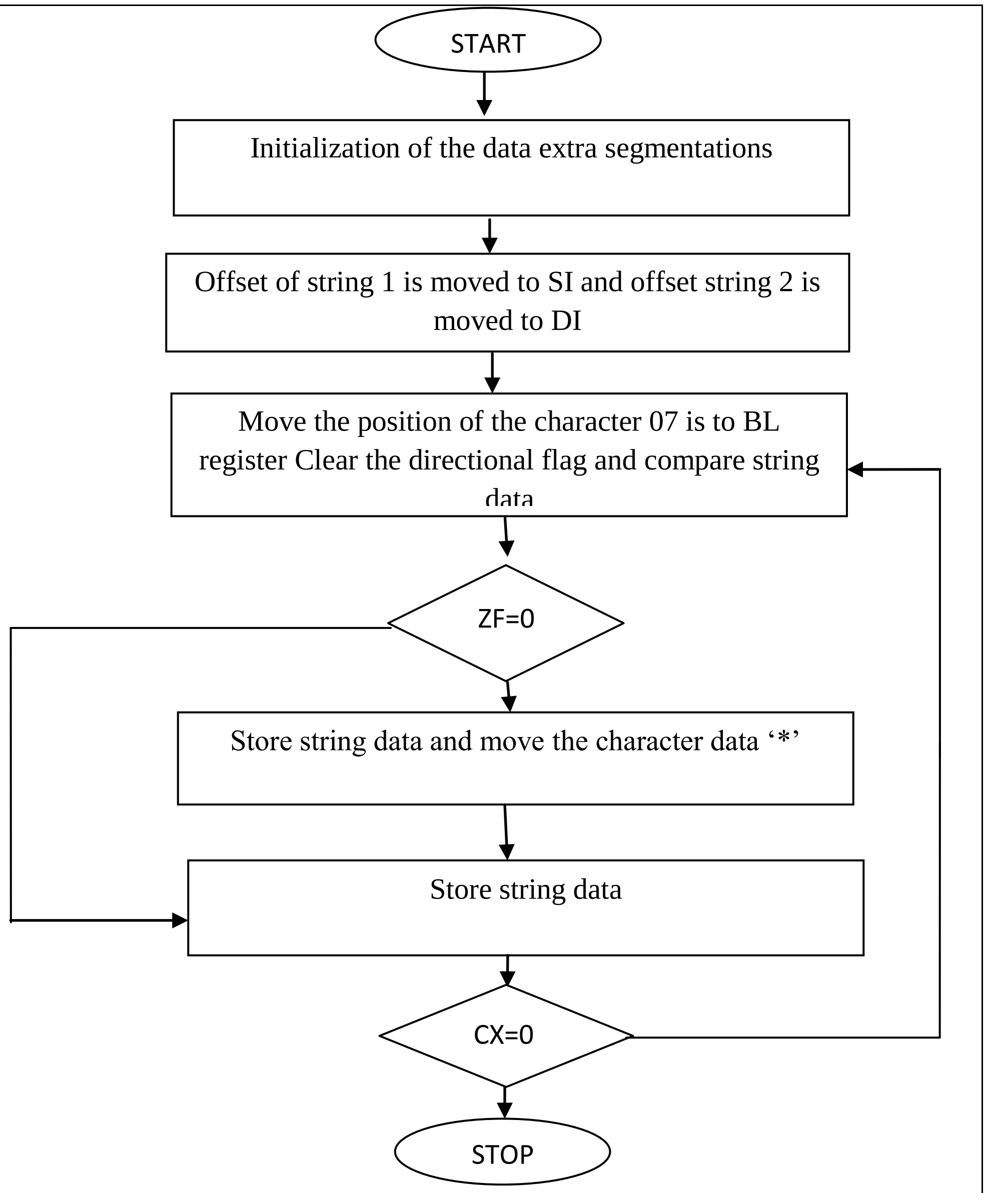
PROGRAM:

```
        ASSUME CS:CODE,DS:DATA
        DATA SEGMENT
        ORG 3000H
        STR DB 0D2H,3EH,76H,12H,0E3H,44H,2AH,69H
        LEN EQU ($-STR)
        DATA ENDS
        CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV CL,LEN-1
UPP:    MOV DL,CL
        LEA SI,STR
UP:     MOV AL,[SI]
        CMP AL,[SI+1]
        JAE TI
        XCHG AL,[SI+1]
        XCHG [SI],AL
TI:     INC SI
        DEC DL
        JNZ UP
        DEC CL
        JNZ UPP
        INT 03
        CODE ENDS
        END START
```

Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. Display boards, controlling signals in traffic light . 4. Can be used as part of sorting and searching in data structure.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	INSERT A CHARACTER IN TO GIVEN STRING
Importance of Experiment	To write an ALP program to insert a character in to a given string using MASM software
Apparatus Required	Personal Computer MASM/TASM Software Installed
Inference /Outcome	**LODSB:** The instruction LODSB is used to load the AL register with the byte pointed by the base and offset address pair DS:SI, after loading the SI content is incremented or decremented automatically according to the direction flag (DF). **STOSB:** The instruction STOSB is used to store the content of AL register to the address pointed by the ES:DI pair, after loading the DI content is incremented or decremented automatically according to the direction flag (DF). **LOOP**: The loop instruction uses count register CX as the no of iterations the loop has to run. It decrements CX register and if zero it breaks the branching **LEA:** This instruction loads the register with effective address of the label. INPUT: DS: SI - MICROPROCESSORS OUTPUT: DS: SI – MICROPRO*CESSORS To observe the insert a character in to given string and also check with practical results.
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**


START
Initialization of the data extra segmentations
Offset of string 1 is moved to SI and offset string 2 is moved to DI
Move the position of the character 07 is to BL register Clear the directional flag and compare string data
ZF=0
Store string data and move the character data '*'
Store string data
CX=0
STOP

ALGORITHM:

1. Load the starting address of the string to which the insertion is to be performed.
2. Load the position where the character is to be inserted.
3. Load the character to be inserted.
4. Store the character of current location of string to new string location.
5. Increment to next location of string and new string. Decrement the position, if not zero goto step 4 or else continue.
6. Store the inserting character to the new string.
7. Store the remaining characters from string to new string.
8. Stop the program

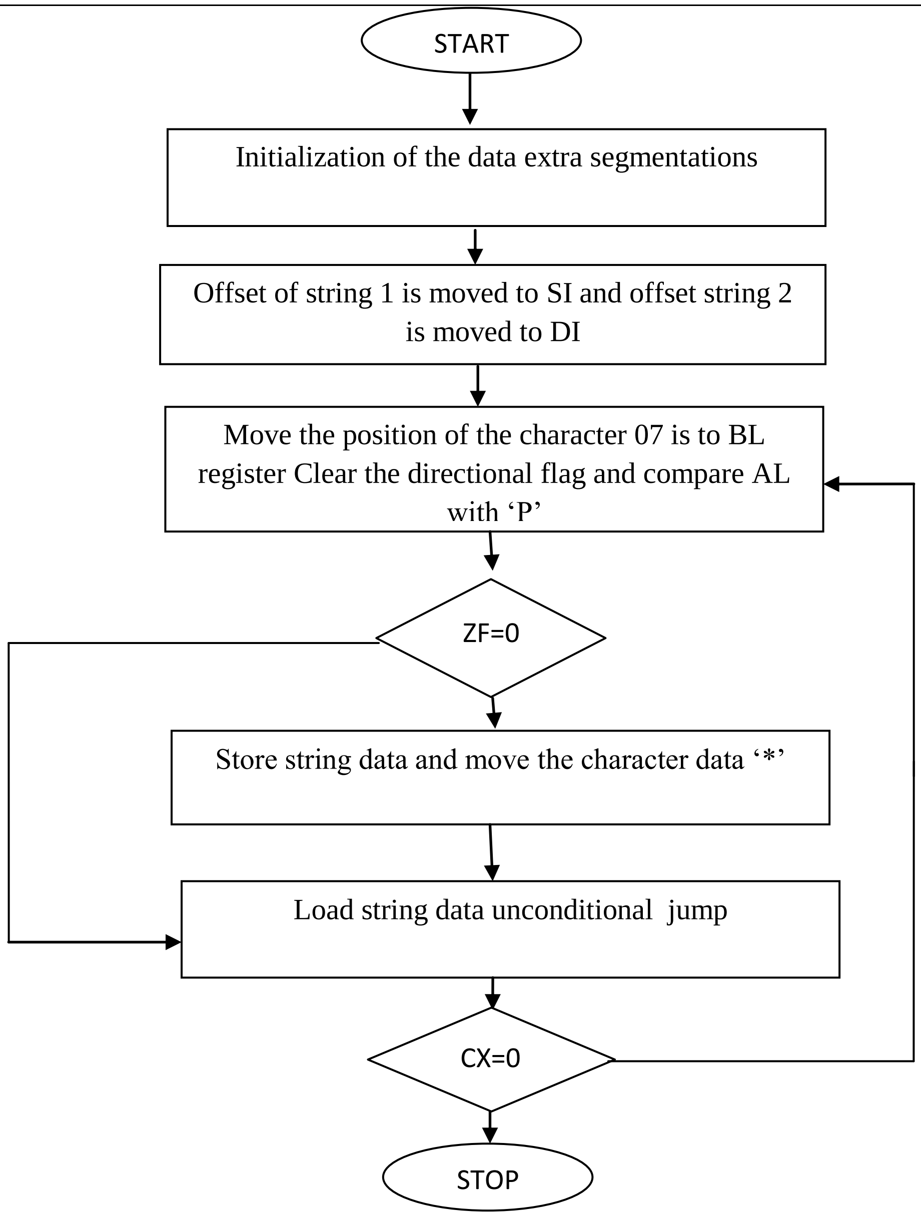
PROGRAM:

```
        ASSUME CS:CODE,DS:DATA,ES:DAT
        DATA SEGMENT
        ORG 3000H
        MSG DB “MICROPROCESSORS"
        LEN EQU ($-STR)
        DATA ENDS
        DAT SEGMENT
        ORG 4000H
        NEW DB 20H DUP(0)
        DAT ENDS
        CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV AX,DAT
```

	MOV ES,AX MOV CX,LEN LEA SI,MSG LEA DI,NEW MOV BL,07H CLD L2: LODSB CMP CL,BL JNZ L1 STOSB MOV AL,’*’ L1: STOSB LOOP L2 INT 03 CODE ENDS END START
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. Display boards, controlling signals in traffic light. 4. In database management, with addition of strings or characters.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	DELETE A CHARACTER FROM A GIVEN STRING
Importance of Experiment	To write an ALP program to delete a character from given string using MASM software
Apparatus Required	Personal Computer MASM/TASM Software Installed
Inference /Outcome	**LODSB:** The instruction LODSB is used to load the AL register with the byte pointed by the base and offset address pair DS:SI, after loading the SI content is incremented or decremented automatically according to the direction flag (DF). **STOSB:** The instruction STOSB is used to store the content of AL register to the address pointed by the ES:DI pair, after loading the DI content is incremented or decremented automatically according to the direction flag (DF). **LOOP**: The loop instruction uses count register CX as the no of iterations the loop has to run. It decrements CX register and if zero it breaks the branching **LEA:** This instruction loads the register with effective address of the label. Delete character: P INPUT: DS: SI - MICROPROCESSORS OUTPUT: DS: SI – MICROROCESSORS To observe the delete a character from given string and also check with practical results.. character: P
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**


START
Initialization of the data extra segmentations
Offset of string 1 is moved to SI and offset string 2 is moved to DI
Move the position of the character 07 is to BL register Clear the directional flag and compare AL with ‘P’
ZF=0
Store string data and move the character data ‘*’
Load string data unconditional jump
CX=0
STOP

ALGORITHM:

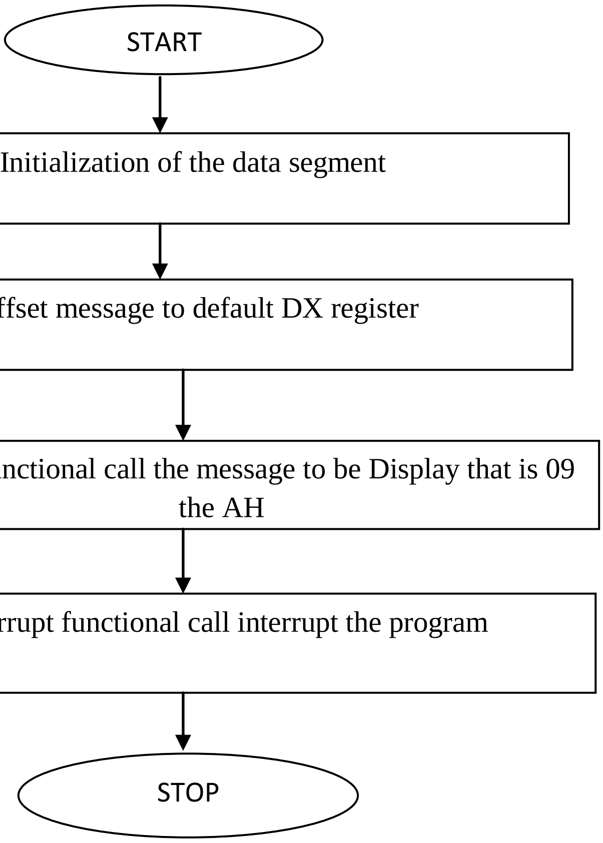
1. Load the starting of the string in which a character is to be deleted.
2. Load the length of the string as count.
3. Use LODSB instruction to obtain the character from the string.
4. Compare the obtained value with that of the deleting character, if same goto step 6 else continue.
5. Store the loaded character to new location using STOSB.
6. Point to next character in string and also to next location of the new string.
7. Decrement the count, if not zero goto step 3 or else continue.
8. Terminate the program.

PROGRAM:

```
        ASSUME CS:CODE,DS:DATA,ES:DAT
        DATA SEGMENT
        ORG 3000H
        MSG DB “MICROPROCESSORS"
        LEN EQU ($-STR)
        DATA ENDS
        DAT SEGMENT
        ORG 4000H
        NEW DB 20H DUP(0)
        DAT ENDS
        CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
```

	MOV AX,DAT MOV ES,AX MOV CX,LEN LEA SI,MSG LEA DI,NEW CLD UP: LODSB L2: CMP AL,’P’ JNZ L1 LODSB JMP L2 L1: STOSB LOOP UP INT 03 CODE ENDS END START
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. Display boards, controlling signals in traffic light. 4. Can be used as editing tool for keyboard program.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

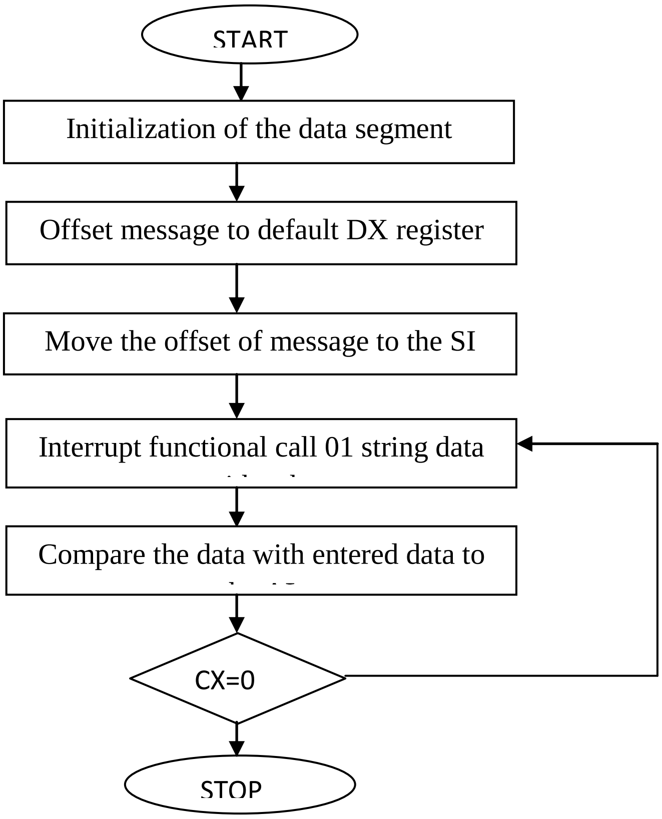
Signature of Faculty Member

Name of Experiment	DISPLAY MESSAGE ON THE SCREEN
Importance of Experiment	To write an ALP program to display the message on the screen using MASM software
Apparatus Required	Personal Computer MASM/TASM Software Installed
Inference /Outcome	**INT 21H:** the 256 interrupts of 8086 can be user defined with subroutine located in interrupt vector table. The INT21H is the interrupt number and entry in IVT for DOS services. **Function 09H**: for DOS services the function number 09H is used to display message to standard output display. The starting address of the message is stored in DX register. INPUT: MICROPROCESSORS OUTPUT SCREEN: MICROPROCESSORS To observe the display message string and also check with practical results.
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**  START Initialization of the data segment Offset message to default DX register Interrupt functional call the message to be Display that is 09 the AH Interrupt functional call interrupt the program STOP **ALGORITHM:** 1. Store The message to be displayed as DOS interrupt of display is

	invoked. 2. The message address is to be loaded into DX. 3. The DOS function number for the display message on screen is 09h, this is to be always present in AH register 4. Invoke DOS service interrupt by INT 21h for message display. 5. The function 4Ch for DOS service lets the program to terminate and return to DOS command prompt. **PROGRAM:** ASSUME CS:CODE,DS:DATA DATA SEGMENT MSG DB “knowledge is wealth”,’$’ DATA ENDS CODE SEGMENT START: MOV AX,DATA MOV DS,AX MOV DX,OFFSET MSG MOV AH,09H INT 21H MOV AH,4CH INT 21H CODE ENDS END START
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. Display boards, controlling signals in traffic light. 4. Used to develop complex OS operations and user programs to send their required data for display.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	READ STRING BYTE from KEYBOARD WITH ECHO
Importance of Experiment	To write an ALP program to read string byte from keyboard with echo using MASM software
Apparatus Required	Personal Computer MASM/TASM Software Installed
Inference /Outcome	**INT 21H**: the 256 interrupts of 8086 can be user defined with subroutine located in interrupt vector table. The INT21H is the interrupt number and entry in IVT for DOS services. Function 09H: for DOS services the function number 09H is used to display message to standard output display. The starting address of the message is stored in DX register. **Function 01H**: the function 01H is used to read a character from standard input i.e the keyboard and the ASCII value of the key is stored in the AL register along with the echo of the key on standard output, monitor. To observe the read string byte from keyboard with echo and also check with practical results. INPUT: enter a string with last character as: MICROPROCESSORS OUTPUT: MICROPROCESSORS (monitor screen entered by user)
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**


START
Initialization of the data segment
Offset message to default DX register
Move the offset of message to the SI
Interrupt functional call 01 string data
Compare the data with entered data to
CX=0
STOP

ALGORITHM:

1. Load the memory with message to request user to enter the string from keyboard.
2. Load AH with 09h function followed by DOS service interrupt INT 21h
3. Load AH with 01 function for reading a character from keyboard with echo, point in memory to the address of the string to be read.
4. Invoke the DOS service interrupt to read a character from keyboard.
5. Store the read in character to memory to a string address. Increment the string address.
6. Compare the character read into AL with the end character '%'. If not zero goto step 4 for reading next character.
7. Finally invoke the return to dos prompt function by 4Ch in AH register followed by INT 21h.

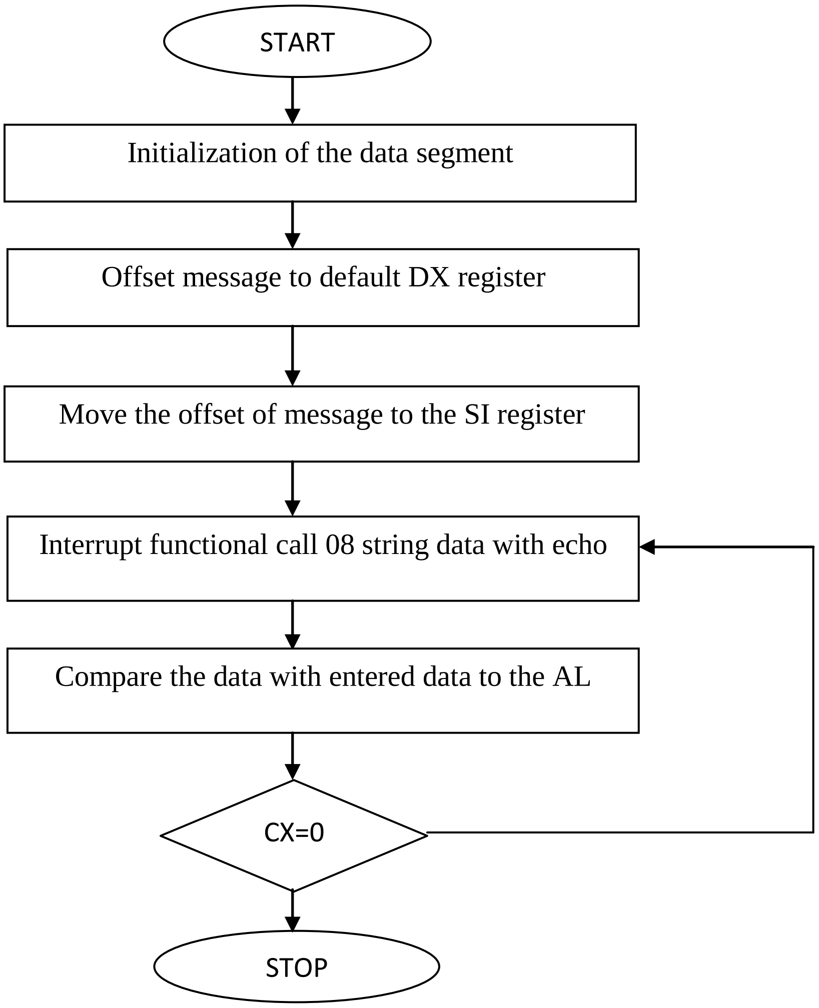
PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
MSG DB "enter a string with last character as:$",
STR DB 40H DUP(0)
DATA ENDS
CODE SEGMENT
START:MOV AX,DATA
        MOV DS,AX
        MOV AH,09H
        MOV DX,OFFSET MSG
        INT 21H
        MOV AH,01H
        MOV SI,OFFSET STR
        DEC SI
UP:     INT 21H
        INC SI
        MOV [SI],AL
        CMP AL,'%'
        JNZ UP
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. Display boards, controlling signals in traffic light. 4. Used to develop complex OS operations and user programs for every key typed from keyboard.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	READ STRING BYTE from KEYBOARD WITHOUT ECHO
Importance of Experiment	To write an ALP program to read string byte from keyboard without echo using MASM software
Apparatus Required	Personal Computer MASM/TASM Software Installed
Inference /Outcome	**INT 21H:** the 256 interrupts of 8086 can be user defined with subroutine located in interrupt vector table. The INT21H is the interrupt number and entry in IVT for DOS services. Function 09H: for DOS services the function number 09H is used to display message to standard output display. The starting address of the message is stored in DX register. **Function 08H:** the function 08H is used to read a character from standard input i.e the keyboard and the ASCII value of the key is stored in the AL register, there will be no echoing of the key to standard output. INPUT: enter a string with last character as: MICROPROCESSORS OUTPUT: DS:SI (monitor screen entered by user) To observe the read string byte from keyboard without echo and also check with practical results.
Correlation of experimental outcome with theoretical concept	**FLOWCHART:**


START
Initialization of the data segment
Offset message to default DX register
Move the offset of message to the SI register
Interrupt functional call 08 string data with echo
Compare the data with entered data to the AL
CX=0
STOP

ALGORITHM:

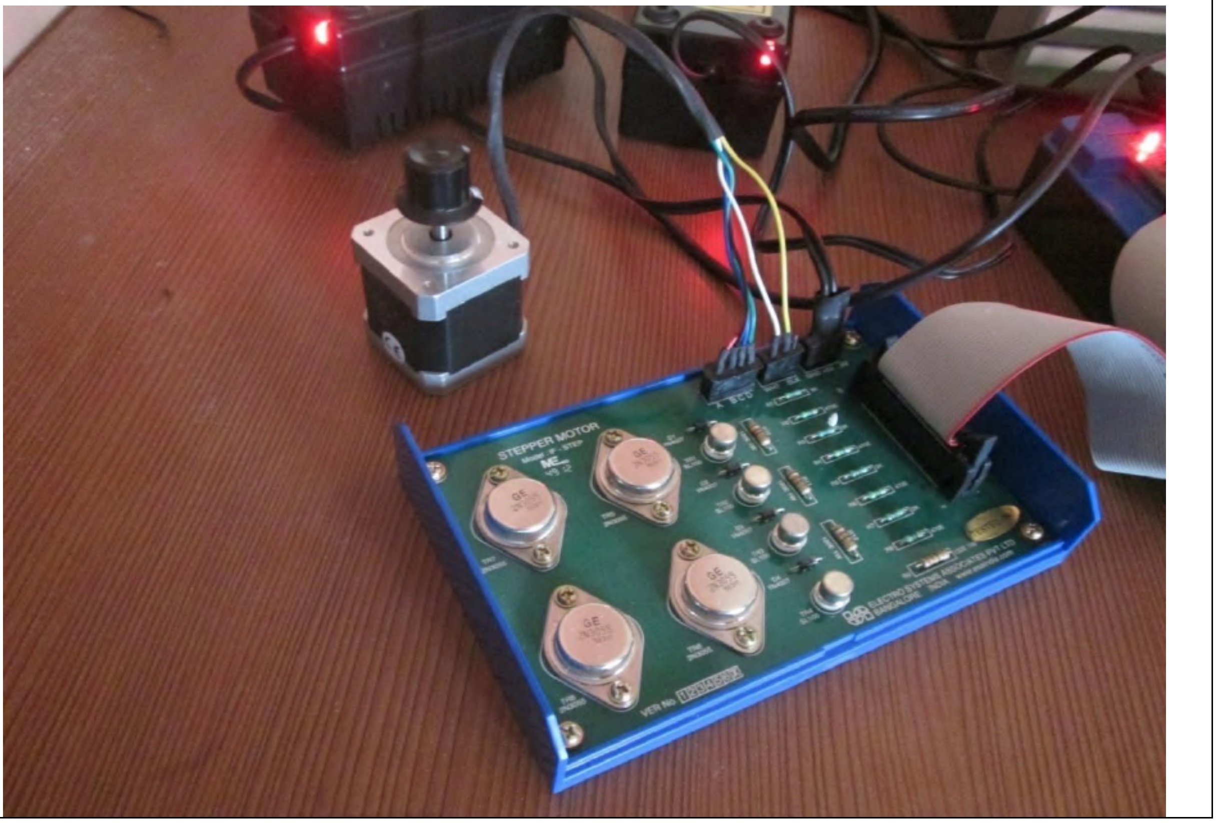
1. Load the memory with message to request user to enter the string from keyboard.
2. Load AH with 09h function followed by DOS service interrupt INT 21h
3. Load AH with 08h function for reading a character from keyboard without echo, point in memory to the address of the string to be read.
4. Invoke the DOS service interrupt to read a character from keyboard.
5. Store the read in character to memory to a string address. Increment the string address.
6. Compare the character read into AL with the end character '%'. If not zero goto step 4 for reading next character.
7. Finally invoke the return to dos prompt function by 4Ch in AH register followed by INT 21h.

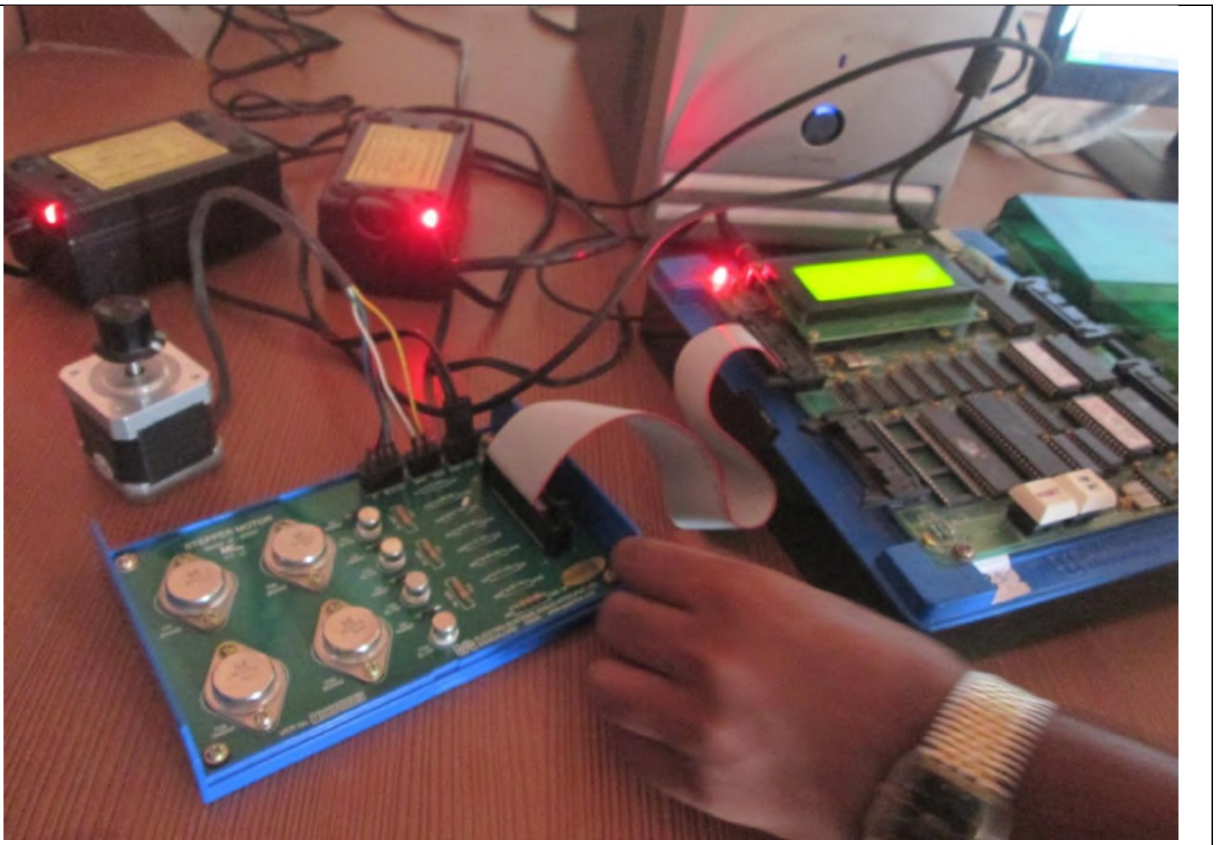
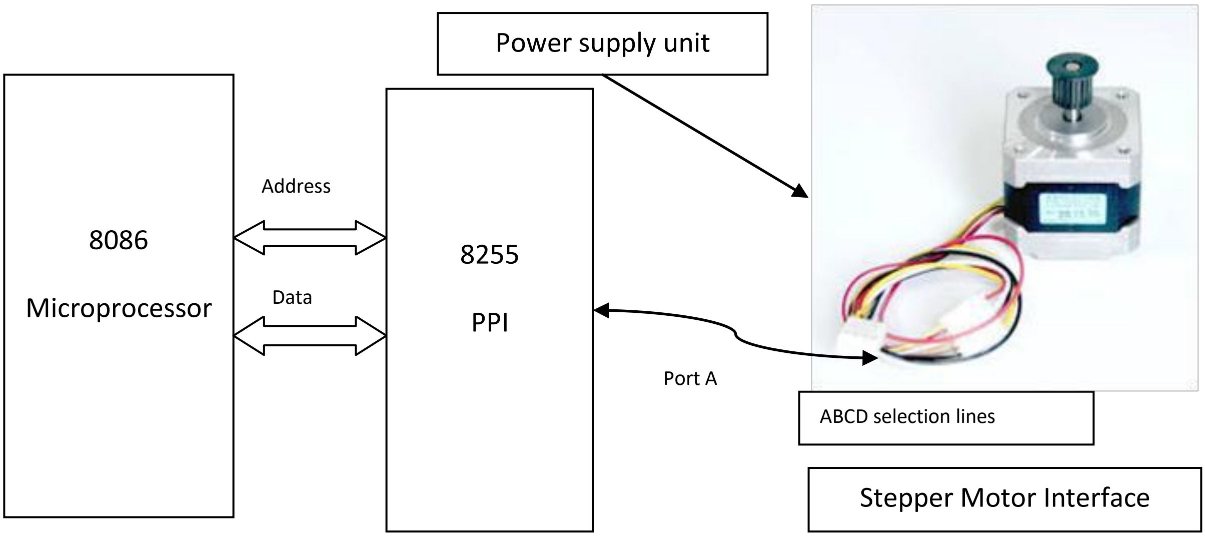
PROGRAM:

```
ASSUME CS:CODE,DS:DATA
                DATA SEGMENT
                MSG DB "enter a string with last character as $"
                STR DB 40H DUP(0)
                DATA ENDS
                CODE SEGMENT
                START:MOV AX,DATA
                        MOV DS,AX
                        MOV AH,09H
                        MOV DX,OFFSET MSG
                        INT 21H
                        MOV AH,08H
                        MOV SI,OFFSET STR
                        DEC SI
                UP:     INT 21H
                        INC SI
```

	MOV [SI],AL CMP AL,’%’ JNZ UP MOV AH,4CH INT 21H CODE ENDS END START
Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3. Display boards, controlling signals in traffic light. 4. Used to develop complex OS operations and user programs for every key typed from keyboard in a password program.
Can you design new experiment with this set up	Yes, as part of password program.
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	STEPPER MOTOR INTERFACE
Importance of Experiment	To write an ALP program to rotate the stepper motor clock and anti clockwise directions
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply 4.8255 Interface card 5.Stepper motor card
Inference /Outcome	To observe the stepper motor interface with 8086 and also check with practical results clock and antilock wise directions 

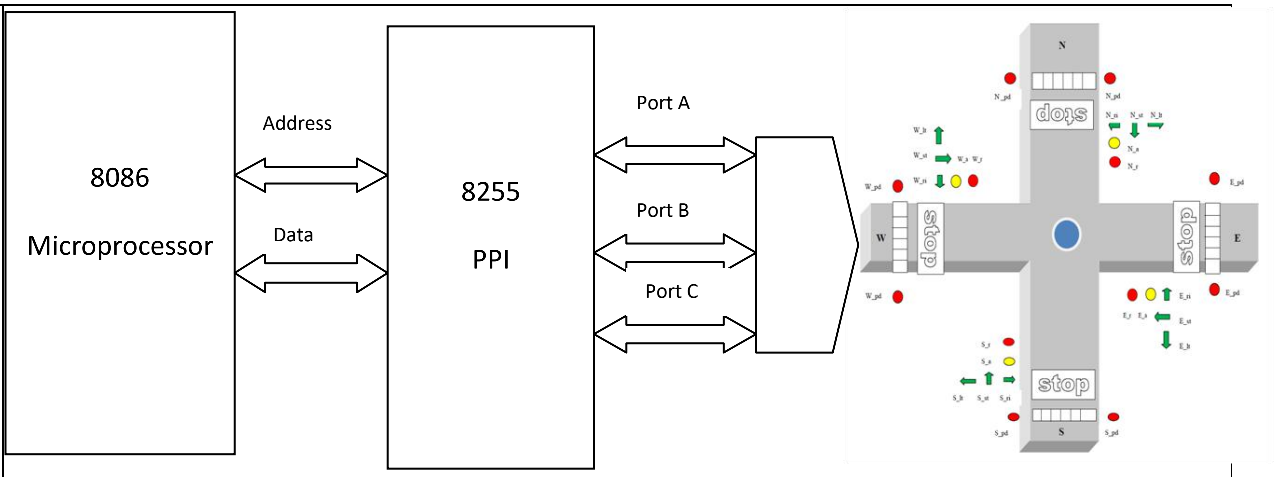
	  Power supply unit 8086 Microprocessor Address Data 8255 PPI Port A ABCD selection lines Stepper Motor Interface
Correlation of experimental outcome with theoretical concept	**PROGRAM:** MOV AL,80 MOV DX,0FFE6 OUT DX,AL MOV AL,11 MOV DX,0FFE0 AGAIN: OUT DX,AL MOV CX,0FFF BACK: LOOP BACK ROL AL,01 JMP AGAIN INT 03
Practical Application	1. Commercial applications like toys, and remote cars etc. 2. In robotics.
Can you design new experiment with this set up	yes

Is the experimental set up in working condition	yes

Signature of Faculty Member

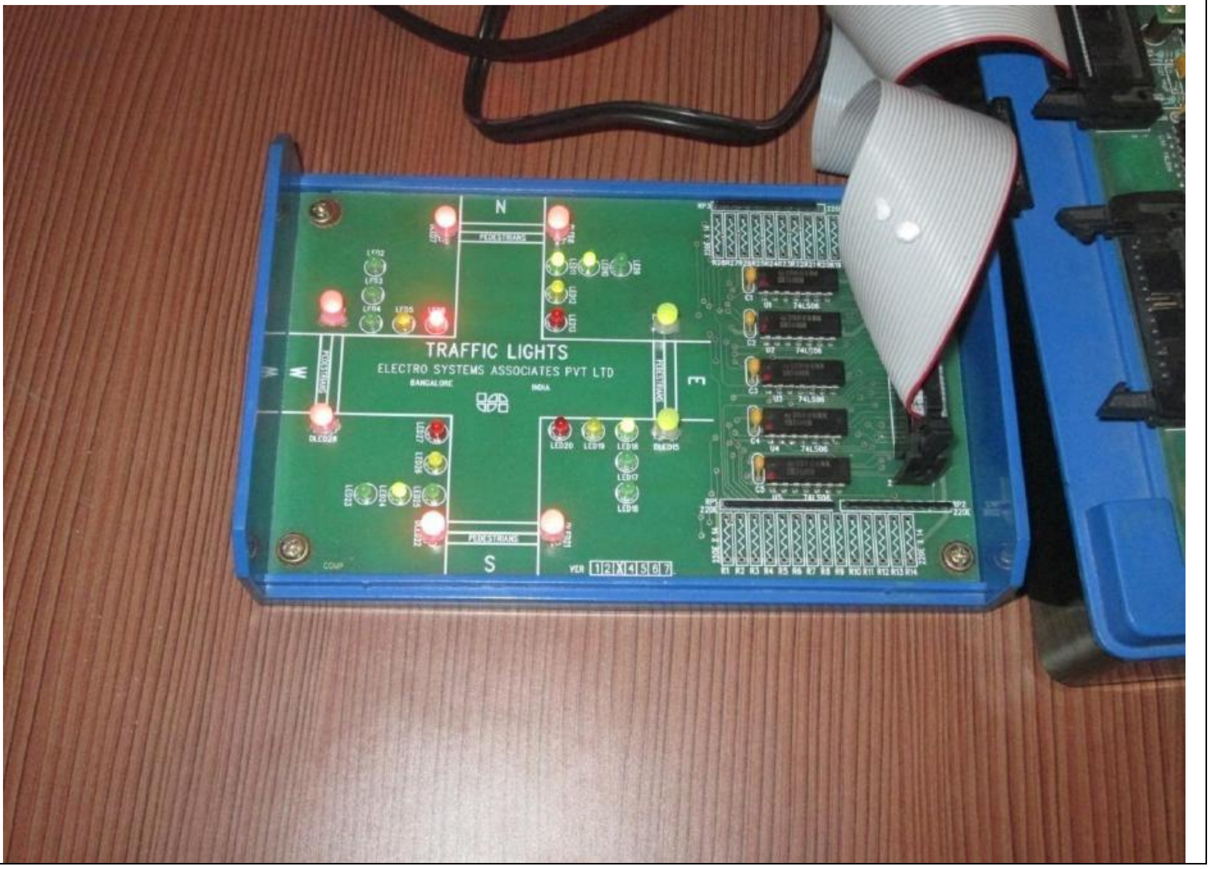
Name of Experiment	TRAFFIC LIGHT INTERFACE
Importance of Experiment	To write an ALP program to interface traffic light system to 8086 through 8255
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply 4.8255 Interface card 5.Traffic light card
Inference /Outcome	**8255 port map to traffic lights**

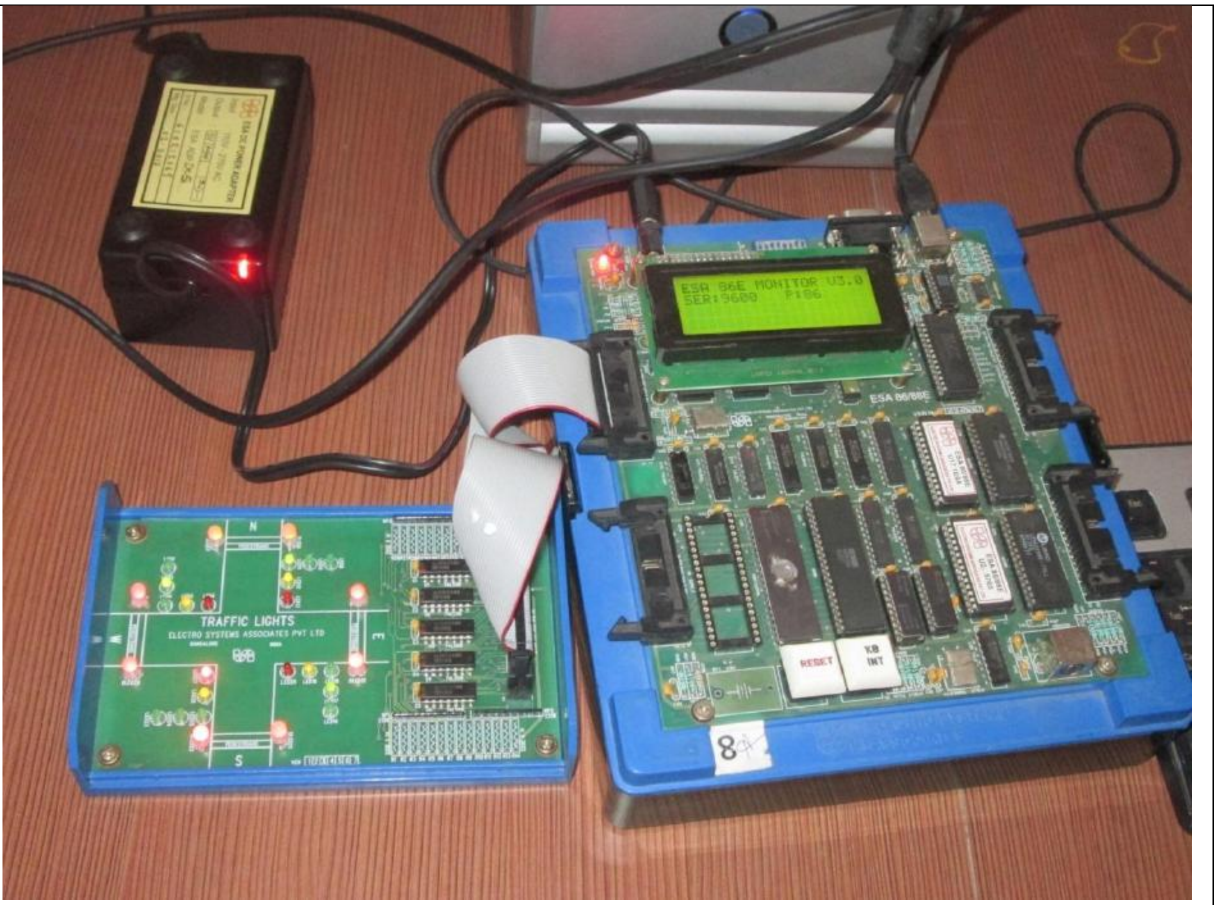
DIRECTION	LED	PORT LINE
SOUTH	RED(S_r)	PA3
	AMBER(S_a)	PA2
	LEFT(S_lt)	PA0
	STRAIGHT(S_st)	PC3
	RIGHT(S_ri)	PA1
	PEDESTRIAN(S_pd)	PC6
EAST	RED	PA7
	AMBER	PA6
	LEFT	PA4
	STRAIGHT	PC2
	RIGHT	PA5
	PEDESTRIAN	PC7
NORTH	RED	PB3
	AMBER	PB2
	LEFT	PB0
	STRAIGHT	PC1
	RIGHT	PB1
	PEDESTRIAN	PC4
WEST	RED	PB7
	AMBER	PB6
	LEFT	PB4
	STRAIGHT	PC0
	RIGHT	PB5
	PEDESTRIAN	PC5



TRAFFIC LIGHT MODEL

To observe the traffic light interface with 8086 through 8255 and also check with practical results.





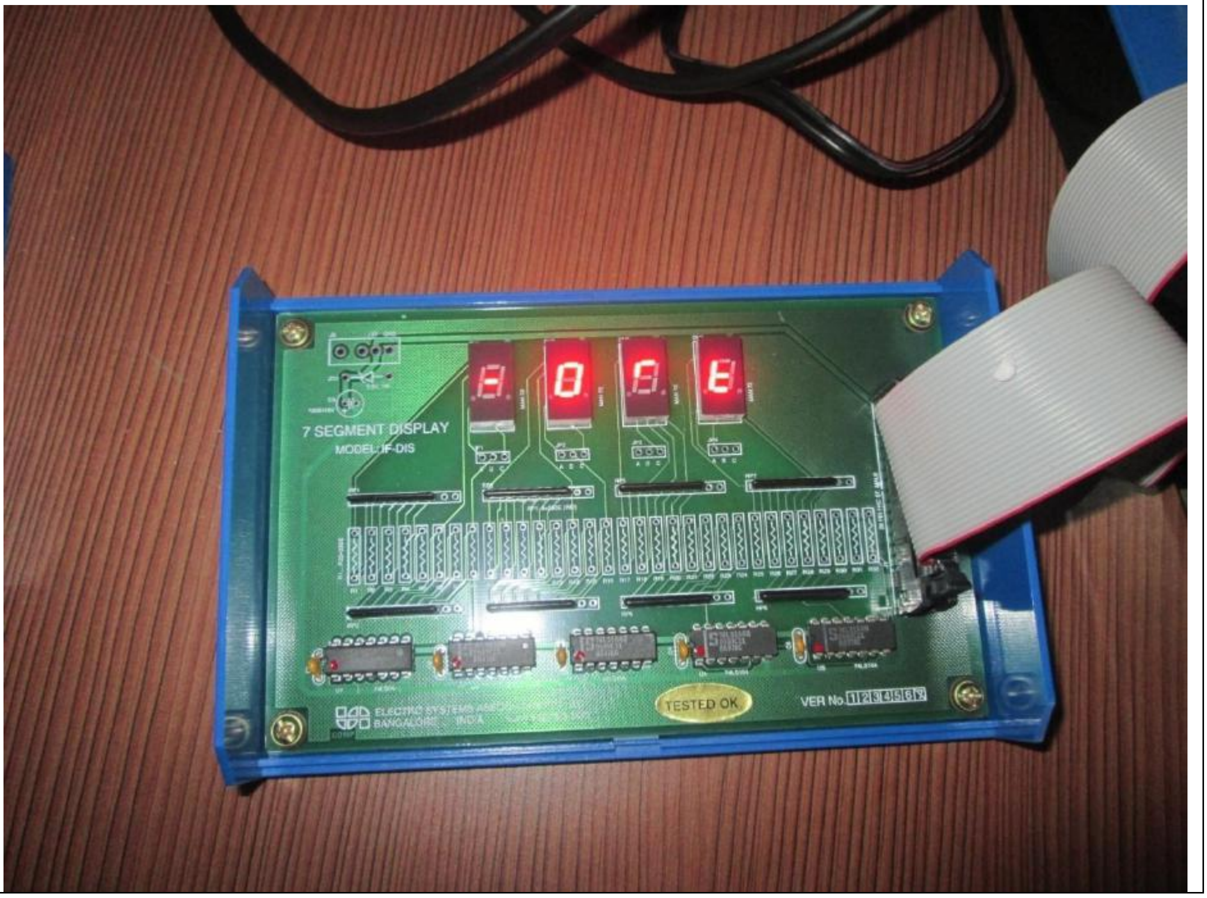
Correlation of experimental outcome with theoretical concept

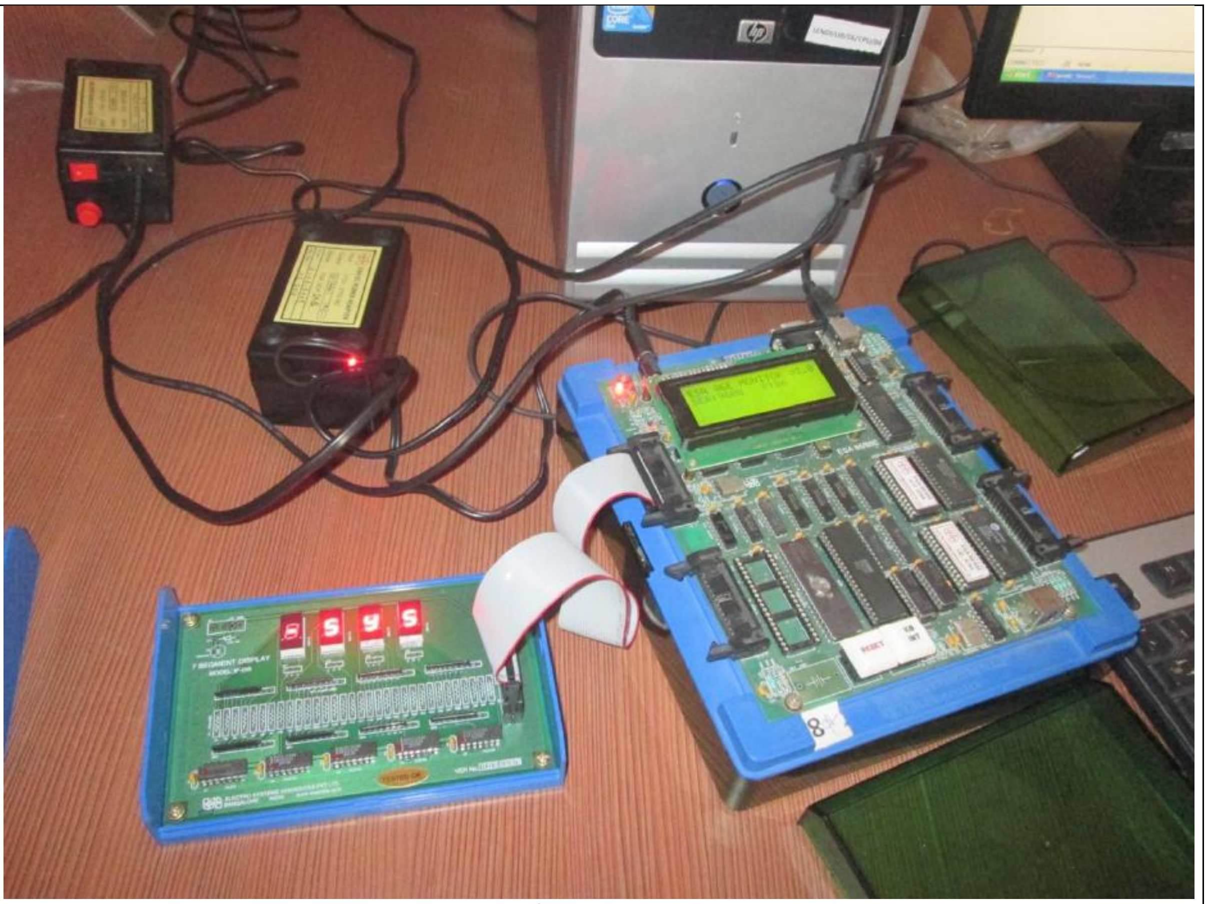
PROGRAM:

```
MOV DX,0FFE6
                MOV AL,80
                OUT DX,AL
                TOP: MOV BL,0A
                MOV SI,2000
                UP:  MOV DX,0FFE0
                MOV AL,[SI]
                OUT DX,AL
                MOV DX,0FFE2
                MOV AL,[SI]01
                OUT DX,AL
                MOV DX,0FFE4
                MOV AL,[SI]02
                OUT DX,AL
                DEC BL
                JZ TT
                MOV DL,BL
                AND DL,01
                JZ NT
                MOV BH,80
                CALL DELAY
                JMP ST
        NT:     MOV BH,10
                CALL DELAY
                ST: ADD SI,03
                JMP UP
        TT:     MOV BH,80
                CALL DELAY
                JMP TOP
                INT 03

DELAY:          MOV CX,0FFFF
                BACK: NOP
                NOP
                LOOP BACK
                DEC BH
                JNZ DELAY
                RET
```

Practical Application	1. Display boards, controlling signals in traffic light.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	SEVEN SEGMENT DISPLAY INTERFACE
Importance of Experiment	To write an ALP program to interface keyboard and display 8086 through
Apparatus Required	1. ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply 4.8255 Interface card 5. seven segment display interface card
Inference /Outcome	Input: “ELECTRO SYSTEMS “ OUTPUT 2500AD  7 SEGMENT DISPLAY MODEL: IF-DIS TESTED OK VER No. 123456



To observe seven segments displays interface and also check with practical results.

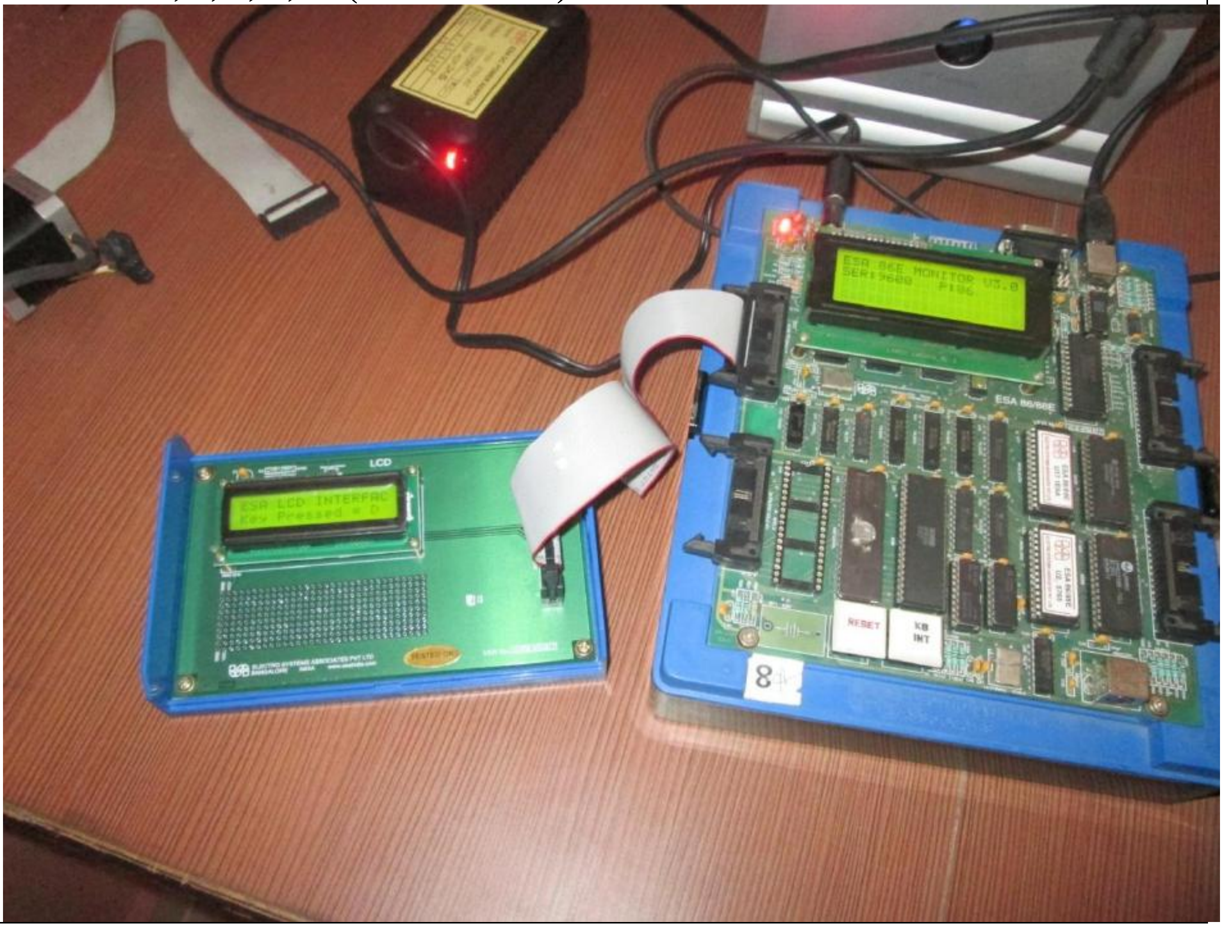
Correlation of experimental outcome with theoretical concept

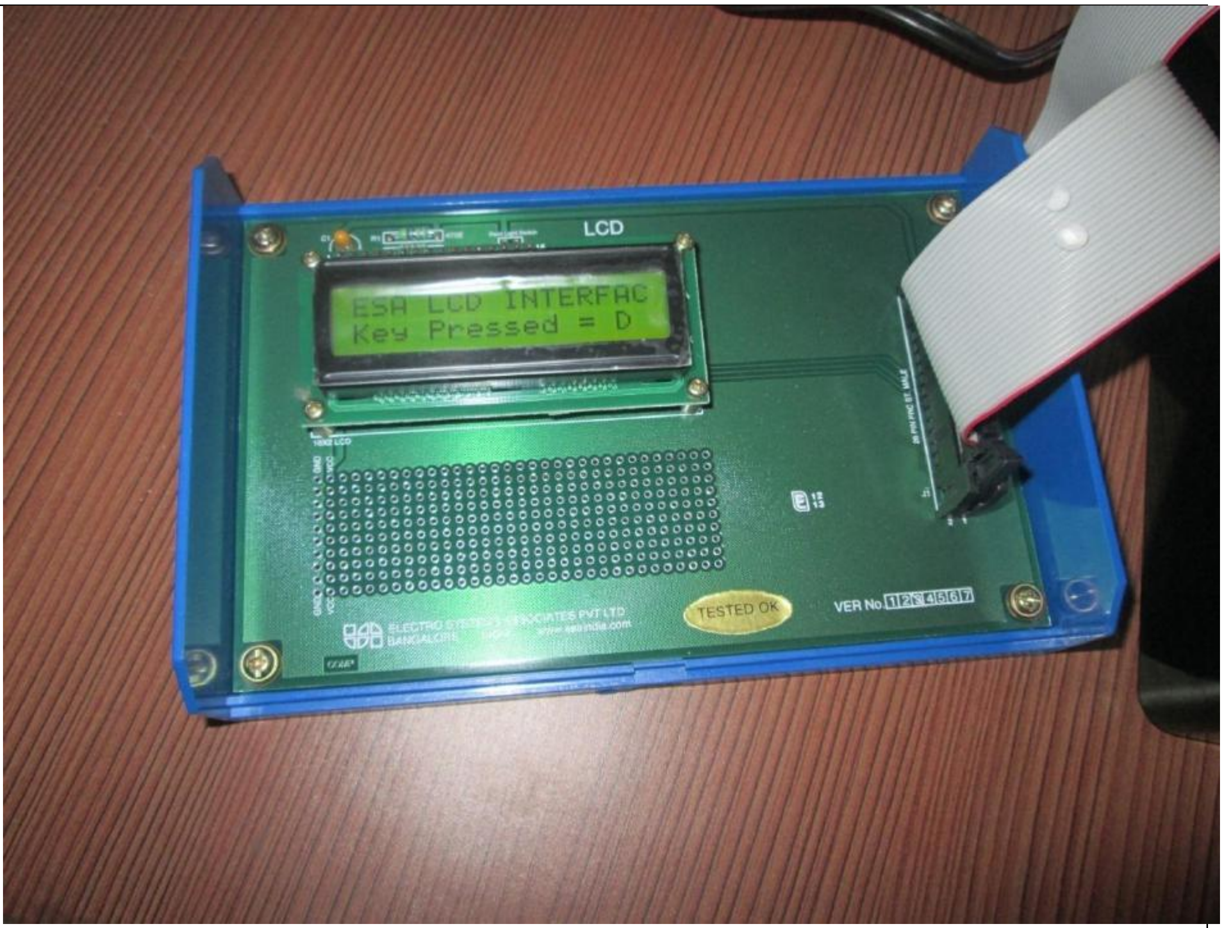
PROGRAM:

```
        ORG 2000H
        MOV AX, 0000H              ; Initialise segment reg.
        MOV DS,AX
        MOV DX,FFE6H               ;Configure all 8255 ports
        MOV AL, 80H                ; as output.
        OUT DX,AL

    LOOP4: MOV SI,2100H            ;Initialise pointer
        MOV CL, 05H                ;set counter for 5 groups.
    LOOP3: MOV CH,04H              ;4 characters/group
    LOOP2: MOV BL,08H              ;8 segments/character

        MOV AL,[SI]                ;get the display code
        INC SI                     ;Increment pointer
    LOOP1: ROL AL,1                ;get 1 data bit
        MOV DX,FFE2H
        OUT DX,AL                  ;o/p bit to portb
        MOV AH,AL
        MOV AL,01H                 ;o/p clock to
        MOV DX,FFE4H               ;shift register
        OUT DX,AL
        DEC AL
        OUT DX,AL
        MOV AL,AH
        DEC BL                     ;all bits over?
        JNZ LOOP1                  ;no, continue
        DEC CH                     ;all characters over?
        JNZ LOOP2                  ;no, continue
        CALL DELAY
        DEC CL                     ;all groups over?
        JNZ LOOP3                  ;no, continue
        JMP SHORT LOOP4

    DELAY: PUSH CX                 ;delay subroutine
        MOV CX,0
    L1: LOOP L1
```

	L2: LOOP L2 POP CX RET ;Display code table ORG 2100H STRING: DB 0FFH,0FFH,0FFH,0FFH DB 086H,0C7H,086H,0C6H DB 087H,0DEH,0C0H,0BFH DB 092H,091H,092H,0BFH DB 087H,086H,0C8H,092H END
Practical Application	Display boards, controlling signals in traffic light.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	LCD INTERFACING
Importance of Experiment	To write an ALP program to interface LCD interfacing
Apparatus Required	1. ESA 86/88E V3 TRAINER BOARD. 2. Key board 3. Power supply 4. Lcd Interface card
Inference /Outcome	INPUT: A, B, C, D, E... (Keyboard key pressing) OUT PUT: A, B, C, D, E...(monitor screen)  ESA LCD INTERFAC Key Pressed = D LCD ESA 86E MONITOR V3.0 SER:9600 P186 ESA 86/88E RESET KB INT



To observe the lcd interfacing with 8086 also show the key being pressed.

Correlation of experimental outcome with theoretical concept

PROGRAM:

```
ORG 2000H
            MOV SP,2200H
            JMP SHORT START
      MSG1:     DB   'ESA LCD INTERFACE',00H
      MSG2:     DB   'Key Pressed = ',00H

       START:   CALL INITLCD
            LEA DX,MSG1
            MOV SI,DX
            MOV AL,80H
            CALL CMDWR
            CALL DISPM
       AGAIN:   MOV AL,C0H
            CALL CMDWR
            LEA DX,MSG2
            MOV SI,DX
            CALL DISPM
            CALL FAR 0FE00:00A9H
            CMP AL,1BH
            JE EXIT
            CALL DATAWR
            JMP SHORT AGAIN
        EXIT:  INT 3

      INITLCD:   MOV DX,0FFE6H
            MOV AL,80H
            OUT DX,AL
            MOV AL,30H
            CALL CMDWR
            MOV AL,30H
            CALL CMDWR
            MOV AL,30H
            CALL CMDWR
            MOV AL,38H
```

```
          CALL CMDWR
          MOV AL,01H
          CALL CMDWR
          MOV AL,02H

          CALL CMDWR
          MOV AL,06H
          CALL CMDWR
          MOV AL,0CH
          CALL CMDWR
          RET

 DISPM:   MOV AL,[SI]
          CMP AL,00H
          JE END
          INC SI
          CALL DATAWR
          JMP SHORT DISPM
   END:   RET

 CMDWR:   PUSH AX
          MOV DX,0FFE2H
          MOV AL,02H
          OUT DX,AL

          MOV AL,00H
          OUT DX,AL

          MOV AL,04H
          OUT DX,AL

          MOV DX,0FFE0H
          POP AX
          OUT DX,AL
          CALL DELAY

          MOV DX,0FFE2H
          MOV AL,00H
          OUT DX,AL

          MOV AL,02H
          OUT DX,AL
          RET

DATAWR:   PUSH AX
          MOV DX,0FFE2H
          MOV AL,03H
          OUT DX,AL

          MOV AL,01H
          OUT DX,AL

          MOV AL,05H
          OUT DX,AL

          MOV DX,0FFE0H
          POP AX
          OUT DX,AL
          CALL DELAY

          MOV DX,0FFE2H
          MOV AL,01H
          OUT DX,AL

          MOV AL,03H
          OUT DX,AL
          RET

DELAY:    MOV CX,03FFH
```

	DY: LOOP DY RET END
Practical Application	Commercial applications like simple calculator, toys, and remote cars etc. Display boards, controlling signals in traffic light.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

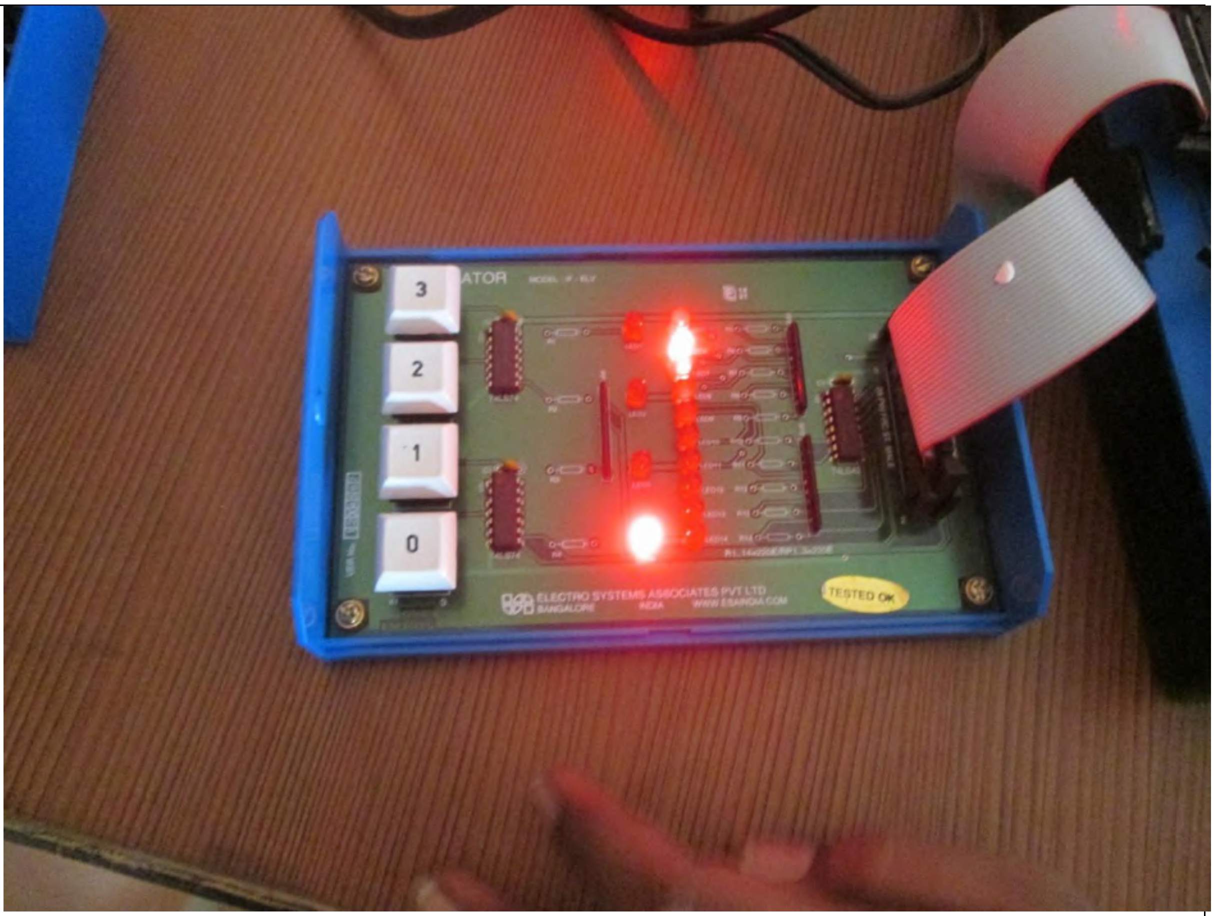
Name of Experiment	HEX KEYBOARD INTERFACING
Importance of Experiment	To write an ALP program to interface HEX KEYBOARD interfacing With 8086
Apparatus Required	1. ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply 4.Hex keyboard Interface card
Inference /Outcome	To observe the hex keyboard interfacing with 8086 and also check with practical results.
Correlation of experimental outcome with theoretical concept	**PROGRAM:** ORG 2000H MOV AX,0000H MOV ES,AX MOV DX,0FFE6H ; Configure 8255 in Mode0 MOV AL,90H ; PortA as i/p , PortB as o/p OUT DX,AL JMP SHORT START MES: DB 'KEY PRESSED = ',0H

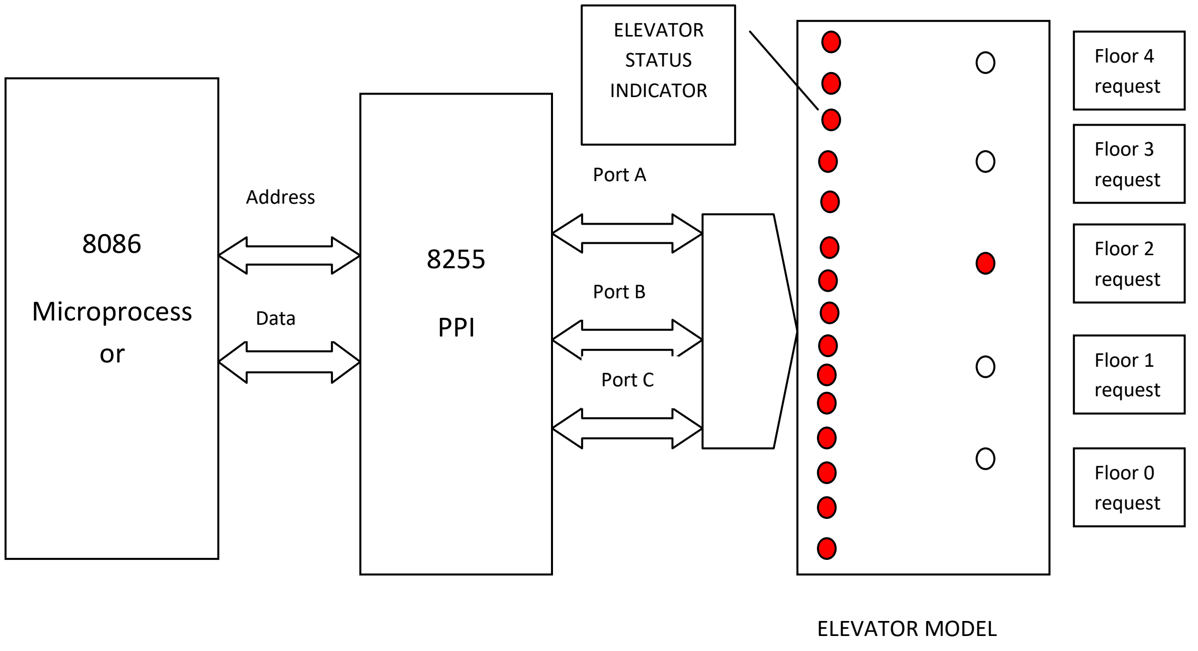
```
START:  CALL    FAR 0FE00:0031H  ;Newline
        LEA     DX,MES
        MOV     AX,DX
        CALL    FAR 0FE00:0013H  ;Call for displaying the message
        CALL    KSCAN
        CALL    DELAY
        JMP     SHORT START

KSCAN: MOV     CL,01H
              MOV     BH,0H
NEXT:   MOV     AL,CL
              MOV     DX,0FFE2H       ;Writing into Port B of 8255
              OUT     DX,AL
              MOV     DX,0FFE0H       ;Reading from Port A of 8255
              IN      AL,DX
             AND       AL,0FH
             MOV     AH,AL
             CMP     AL,0H
             JNE     KEYCODE
CONT:      ROL     CL,1
           CMP     CL,10H                  ;Compare upto Highest scan line
           JE      KSCAN
           ADD         BH,04H
           JMP     NEXT

KEYCODE:MOV     BL,0H
                    MOV     AL,AH
       SHIFT:  SHR     AL,1
                    CMP     AL,00H
                    JE      ROW
                    INC     BL
                   JMP     SHIFT
       ROW:  ADD     BH,BL
                  MOV AL,BH
CALL    FAR 0FE00:0052H
RET

DELAY:  PUSH    CX
MOV     CX,00H        ; Delay routine
DLY:    LOOP    DLY
POP     CX
RET

END
```

Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. Display boards, controlling signals in traffic light.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	ELEVATOR INTERFACE DESIGN
Importance of Experiment	To write an ALP program to interface elevator to 8086 through 8255 and study its function.
Apparatus Required	1.ESA 86/88E V3 TRAINER BOARD. 2.Key board 3.Power supply 4. elevator interface model.
Inference /Outcome	To design an elevator model and interface with 8086 through 8255 and observe its operation- students may learn the working model of the elevator.





Block diagram

Correlation of experimental outcome with theoretical concept

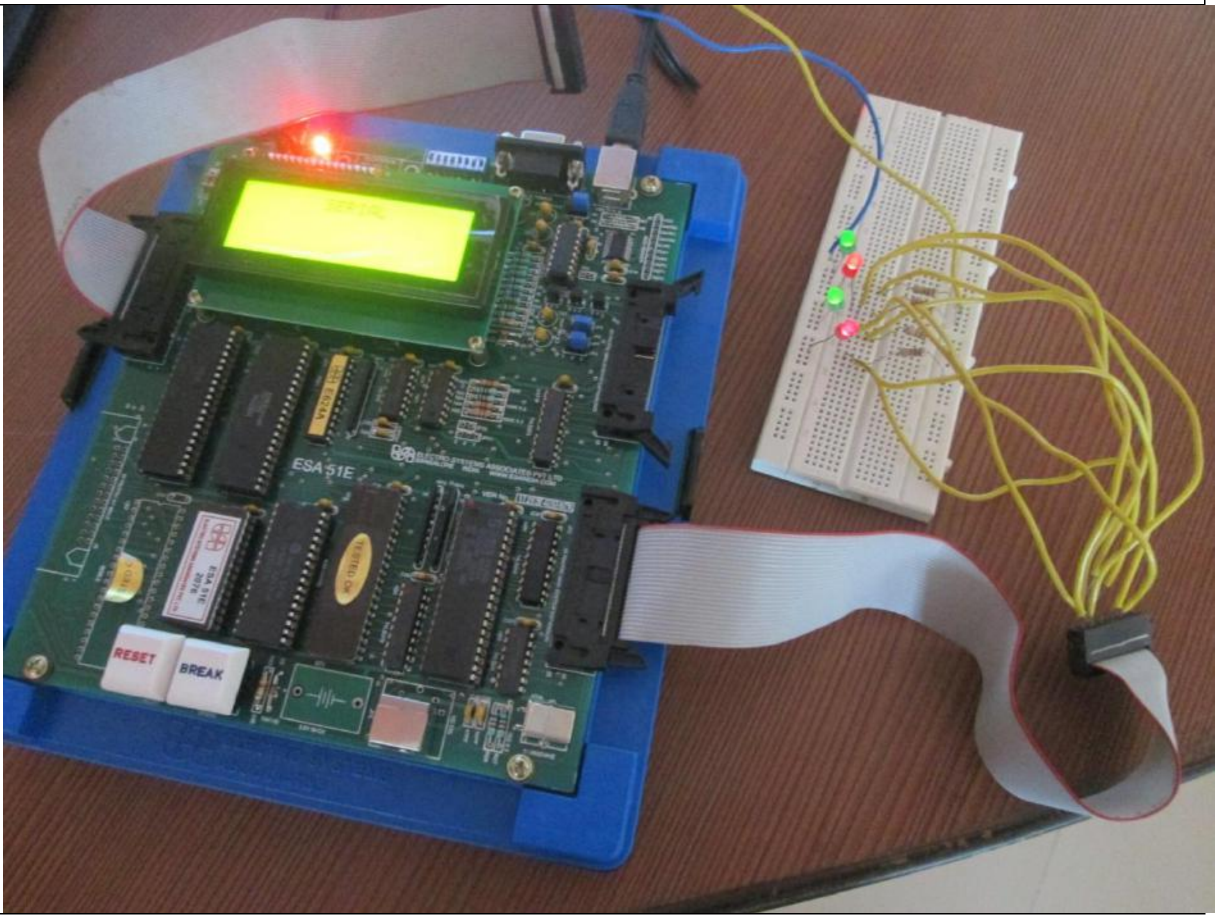
PROGRAM:

```
ORG     2000H
                    MOV     DX,0FFE6H          ; Configure 8255
                    MOV     AL,82H             ; PortA as o/p,PortB as i/p
                   OUT     DX,AL
                   XOR     AX,AX                          ; Initial stage is ground floor

LOOP1:             MOV     AL,AH               ; AH is the floor position
                   OR     AL,0F0H
                   MOV     DX,0FFE0H
                    OUT     DX,AL


                    MOV     DX,0FFE2H
LOOP2:             IN      AL,DX              ; Get request
                   AND     AL,0FH
```

```
                          CMP     AL,0FH
                          JZ      LOOP2

                           MOV     SI,00H
FINDF:                    ROR     AL,01H
                          JNC     FOUND              ;If requested floor found
                          INC     SI
                          JMP     SHORT   FINDF    ; Otherwise,continue search
FOUND:                    MOV     AL,[SI]2100H    ; Get requesting floor code
                          CMP     AL,AH              ; Compare with current floor
                          JA      GOUP                           ; If it need to go UP
                          JB      GODN                          ; If it need to g DOWN
CLEAR:                    MOV     AL,[SI]2104H
                          MOV     DX,0FFE0H
                          OUT     DX,AL
                          JMP     SHORT   LOOP1
GOUP:                     CALL    DELAY
                          INC     AH                            ; Elevator goes UP by one LED
                          XCHG    AL,AH
                          OR      AL,0F0H
                          MOV     DX,0FFE0H
                          OUT     DX,AL
                          AND     AL,0FH
                          XCHG    AH,AL
                          CMP     AL,AH
                          JNZ     GOUP
                          JMP     SHORT   CLEAR
GODN:                     CALL    DELAY
                          DEC     AH                            ; Elevator goes DOWN by one LED
                          XCHG    AH,AL
                          OR      AL,0F0H
                          MOV     DX,0FFE0H
                          OUT     DX,AL
                          AND     AL,0FH
                          XCHG    AL,AH
                          CMP     AL,AH
                          JNZ     GODN
                          JMP     SHORT   CLEAR

DELAY:                    MOV     CX,0800H           ; Delay between glow of successive LEDs
             HR1:    LOOP    HR1
             HR2:    LOOP    HR2
                            RET

       ORG     2100H

VALUE1:  DB      00H,03H,06H,09H     ; Position codes for floors
VALUE2:  DB      0E0H,0D3H,0B6H,79H  ; clear code+position dode for all floors
```

Practical Application	1. Commercially used in buildings for lateral transport of people and in construction to carry building material. 2. Used to carry large loads in to board in shipping.
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	8051 MICRO CONTROLLER PARALLEL PORT READING
Importance of Experiment	To write an ALP program to read parallel port and send the data to parallel port of the 8051 micro controller.
Apparatus Required	1. ESA 8051E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	To observe the 8051 micro controller parallel port reading and also check with practical results.
Correlation of experimental outcome with theoretical concept	

ALGORITHM:

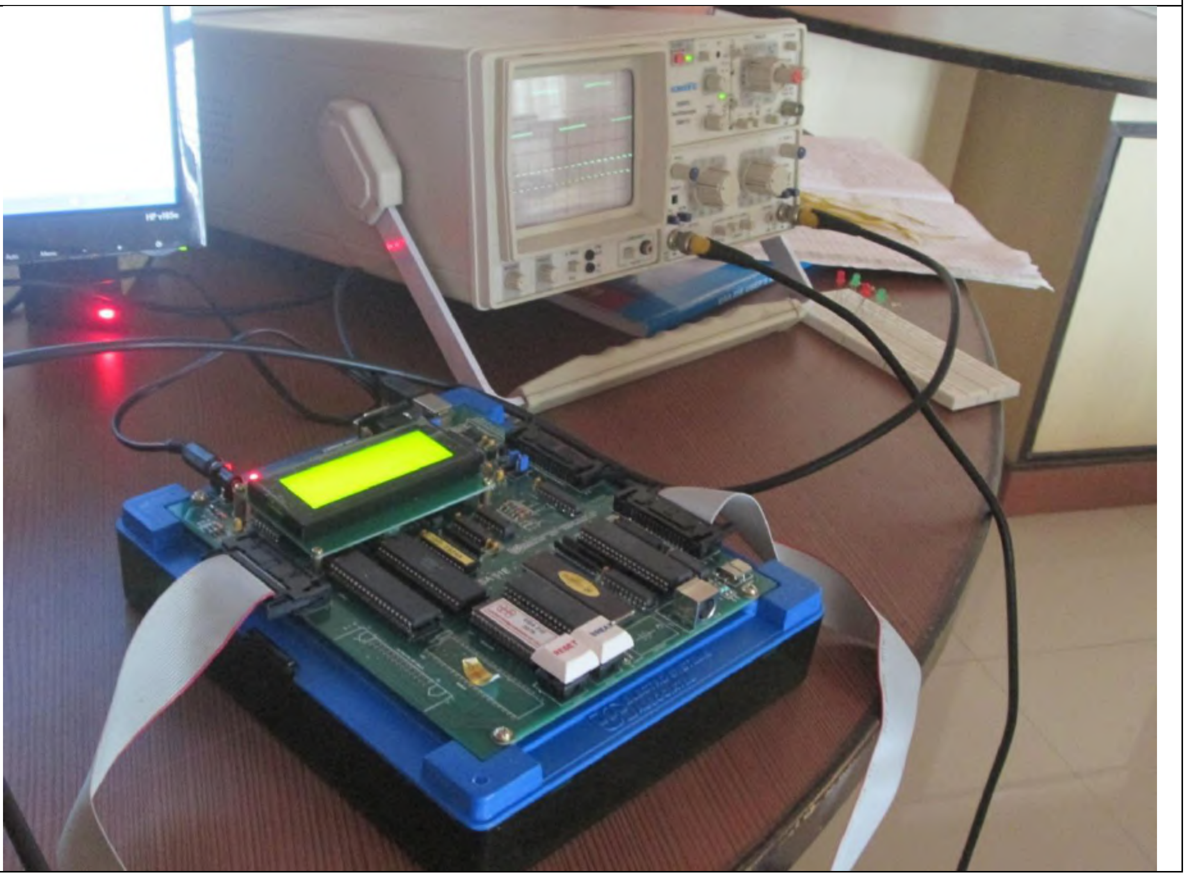
1. The lower nibble of port1 is made as inputs by giving 1bit to the corresponding pins.
2. The higher nibble is used as output lines to LED’s.
3. The closed or open switch to GND on input lines is read into the port.
4. The switch status is displayed on the output LED lines.
5. If the switch is closed the LED glows else LED is in off state.

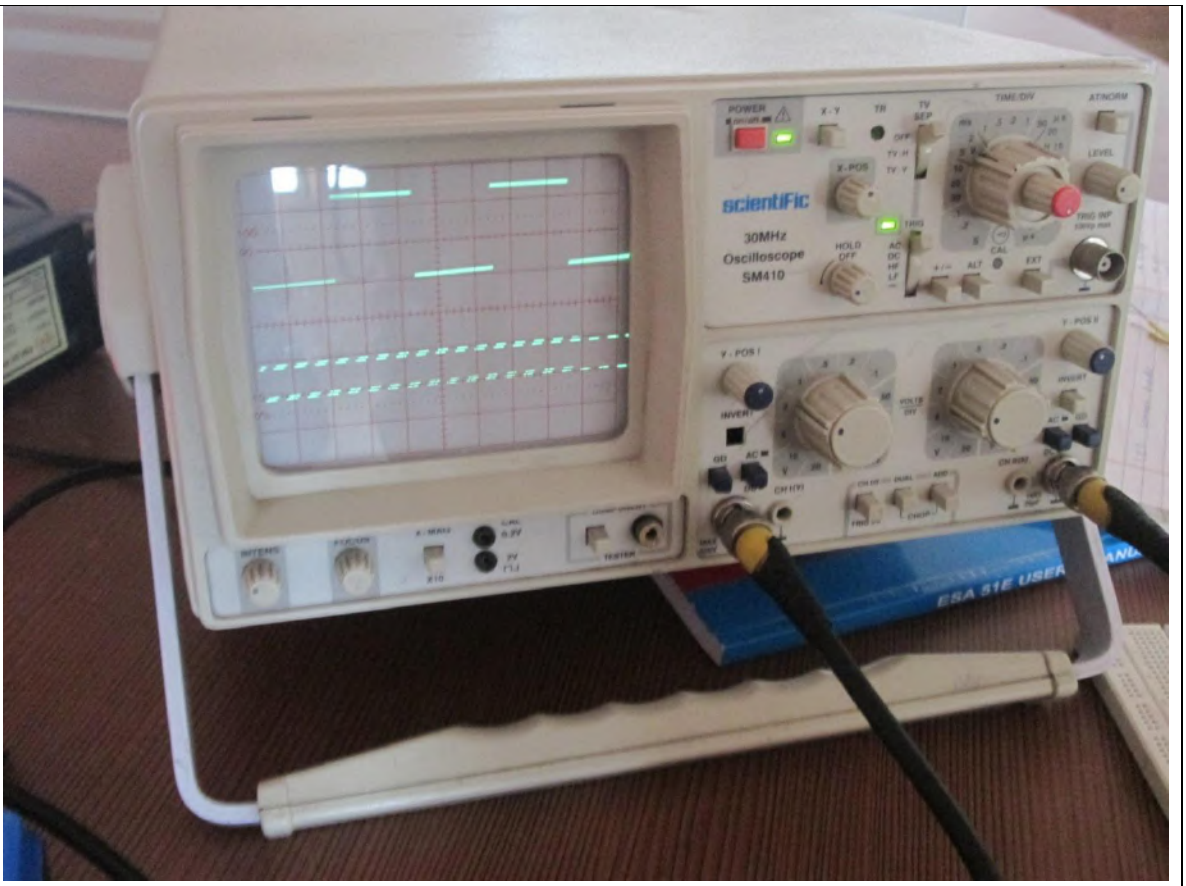
PROGRAM:

```
ORG 0H
JMP START
ORG 1000H
  START:     ORL P1,#0FH
             MOV A,P1
             SWAP A
             MOV P1,A
             SJMP START
   END
```

Practical Application	1. Commercial applications like simple calculator, toys, and remote cars etc. 2. And industrial wise large data processing and insurance companies. 3.Display boards ,controlling signals in traffic light .
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member

Name of Experiment	MODES OF TIMER OF 8051 CONTROLLER
Importance of Experiment	To write an ALP program to run the timer of 8051 in different modes
Apparatus Required	1.ESA 8051E V3 TRAINER BOARD. 2.Key board 3.Power supply
Inference /Outcome	To observe the 8051 micro controller parallel port reading and also check with practical results..
Correlation of experimental outcome with theoretical concept	



ALGORITHM:

1. The mode of timer 0 is mode 0 and timer 1 in mode 1.
2. Set the bits of port P1.0 and P1.1.
3. Set the timers overflow flags.
4. Load the timer count value to timer0 registers TL0 and TH0. Set the timer0 to run state by clearing the overflow flag and run flag.
5. Check for overflow flag to set, if not set goto step 7.
6. Load the timer count value to timer1 registers TL1 and TH1. Set the timer 1 to run state by clearing the overflow flag and run flag.
7. Check for overflow flag to set, if not set goto step 5.

PROGRAM:

```
UP:     MOV TMOD,#10H
        SETB P1.0
        SETB P1.1
        SETB TF0
        SETB TF1
TIMER0: JNB TF0,TIMER1
        CLR TR0
        CLR TF0
        CPL P1.0
        MOV TL0,#0FFH
        MOV TH0,#0F0H
        SETB TR0
TIMER1: JNB TF1, TIMERO
        CLR TR1
        CLR TF1
        CPL P1.1
        MOV TL1,#0FFH
        MOV TH1,#0F0H
        SETB TR1
        SJMP TIMER0
       END
```

Practical	1. Commercial applications like simple calculator, toys, and remote cars etc.

Application	2. Display boards, controlling signals in traffic light .
Can you design new experiment with this set up	yes
Is the experimental set up in working condition	yes

Signature of Faculty Member