DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

Academic Year: 2015-16 ODD SEMESTER

Programme (UG/PG) : **UG-B.Tech**

Semester : **03**

Course Code :**CS1033**

Course Title : **MICROPROCESSOR & INTERFACING LAB**

Prepared By

S.KIRUTHIKA DEVI
(A.P(O.G), Department of Computer Science and Engineering)

FACULTY OF ENGINEERING AND TECHNOLOGY
SRM UNIVERSITY
(Under section 3 of UGC Act, 1956)
SRM Nagar, Kattankulathur- 603203
Kancheepuram District

LIST OF EXPERIMENTS & SCHEDULE

COURSE CODE/TITLE: CS1033 - MICROPROCESSOR & INTERFACING LAB

Exp. No.	Title	Week No.
A	8085 Programs	1 - 5
1	8-bit Addition, Subtraction, Multiplication and Division	1
2	16-bit Addition, Subtraction, Multiplication and Division	2
3	Largest number in a data array	3
4	Smallest number in a data array	3
5	BCD to Hexadecimal and vice-versa	4
6	BCD to Binary Conversion and vice-versa	4
7	Move a data block without overlap	5
8	Counters and Time Delay	5
B	8086 Programs	6-8
9	Basic arithmetic and Logical operations	6
10	Code conversion, sorting and searching	7
11	Data transfer operations	8
12	Password checking	8
13	Print RAM size and system date	8
C.	Peripherals and Interfacing Experiments	9-12
14	Traffic light control	9
15	Stepper motor control	10
16	Digital clock	11
17	Key board and Printer status	12

Course Coordinator HOD

HARDWARE AND SOFTWARE REQUIREMENTS

SYSTEM REQUIREMENTS

- 8085 microprocessor kit.
- Jubin's- 8085 simulator.
- MASM
- Stepper Motor
- Traffic Light Controller
- 7 Segment LED Display

Operating system : Windows XP , Windows 7 - 32 and 64 bit editions, Windows 2000

Service Pack 3, Windows Server 2003, Windows XP Service Pack 2

INTERNAL ASSESSMENT MARK SPLIT UP

Observation	: 20 Marks
Attendance	: 5 Marks
Mini Project with the Report (Max. 8 Pages & 3 Students per Batch)	: 20 Marks
Model Exam	: 15 Marks
TOTAL MARKS	: 60 Marks

EXERCISE NO.1A

ADDITION OF TWO 8 BIT NUMBERS

AIM

To perform addition of two 8 bit numbers using 8085.

ALGORITHM

1) Start the program by loading the first data into Accumulator.
2) Move the data to a register (B register).
3) Get the second data and load into Accumulator.
4) Add the two register contents.
5) Check for carry.
6) Store the value of sum and carry in memory location.
7) Terminate the program.

SOURCE CODE

	MVI	C, 00	Initialize C register to 00
	LDA	4150	Load the value to Accumulator.
	MOV	B, A	Move the content of Accumulator to B register.
	LDA	4151	Load the value to Accumulator.
	ADD	B	Add the value of register B to A
	JNC	LOOP	Jump on no carry.
	INR	C	Increment value of register C
LOOP:	STA	4152	Store the value of Accumulator (SUM).
	MOV	A, C	Move content of register C to Acc.
	STA	4153	Store the value of Accumulator (CARRY)
	HLT		Halt the program.

SAMPLE INPUT & OUTPUT

Input: 80 (4150)
80 (4251)

Output: 00 (4152)
01 (4153)

RESULT

Thus the program to add two 8-bit numbers was executed.

EX NO.1B

SUBTRACTION OF TWO 8 BIT NUMBERS

AIM

To perform the subtraction of two 8 bit numbers using 8085.

ALGORITHM

1. Start the program by loading the first data into Accumulator.
2. Move the data to a register (B register).
3. Get the second data and load into Accumulator.
4. Subtract the two register contents.
5. Check for carry.
6. If carry is present take 2's complement of Accumulator.
7. Store the value of borrow in memory location.
8. Store the difference value (present in Accumulator) to a memory location
9. Terminate the program.

SOURCE CODE

	MVI	C, 00	Initialize C to 00
	LDA	4150	Load the value to Acc.
	MOV	B, A	Move the content of Acc to B register.
	LDA	4151	Load the value to Acc.
	SUB	B	Subtract the value of register B to A
	JNC	LOOP	Jump on no carry.
	CMA		Complement Accumulator contents.
	INR	A	Increment value in Accumulator.
	INR	C	Increment value in register C
LOOP:	STA	4152	Store the value of A-reg to memory address.
	MOV	A, C	Move contents of register C to Accumulator.
	STA	4153	Store the value of Accumulator memory address.
	HLT		Terminate the program.

SAMPLE INPUT & OUTPUT

Input: 06 (4150)
02 (4251)

Output: 04 (4152)
01 (4153)

RESULT

Thus the program to subtract two 8-bit numbers was executed.

EX. NO.1C

MULTIPLICATION OF TWO 8 BIT NUMBERS

AIM

To perform the multiplication of two 8 bit numbers using 8085.

ALGORITHM

1) Start the program by loading HL register pair with address of memory location.
2) Move the data to a register (B register).
3) Get the second data and load into Accumulator.
4) Add the two register contents.
5) Check for carry.
6) Increment the value of carry.
7) Check whether repeated addition is over and store the value of product and carry in memory location.
8) Terminate the program.

SOURCE CODE

	MVI	D, 00	Initialize register D to 00
	MVI	A, 00	Initialize Accumulator content to 00
	LXI	H, 4150	
	MOV	B, M	Get the first number in B - reg
	INX	H	
	MOV	C, M	Get the second number in C- reg.
LOOP:	ADD	B	Add content of A - reg to register B.
	JNC	NEXT	Jump on no carry to NEXT.
	INR	D	Increment content of register D
NEXT:	DCR	C	Decrement content of register C.
	JNZ	LOOP	Jump on no zero to address
	STA	4152	Store the result in Memory
	MOV	A, D	Move the content of D register to Accumulator
	STA	4153	Store the MSB of result in Memory
	HLT		Terminate the program.

SAMPLE INPUT &OUTPUT

Input: FF (4150)
FF (4151)

Output: 01 (4152)
FE (4153)

RESULT

Thus the program to multiply two 8-bit numbers was executed.

EXERCISE NO.1D

DIVISION OF TWO 8 BIT NUMBERS

AIM

To perform the division of two 8 bit numbers using 8085

ALGORITHM

1) Start the program by loading HL register pair with address of memory location.
2) Move the data to a register (B register).
3) Get the second data and load into Accumulator.
4) Compare the two numbers to check for carry.
5) Subtract the two numbers.
6) Increment the value of carry.
7) Check whether repeated subtraction is over and store the value of product and carry in memory location.
8) Terminate the program.

SOURCE CODE

	LXI	H, 4150	
	MOV	B, M	Get the dividend in B – reg.
	MVI	C, 00	Clear C – reg for quotient
	INX	H	
	MOV	A, M	Get the divisor in A – reg.
NEXT:	CMP	B	Compare A - reg with register B.
	JC	LOOP	Jump on carry to LOOP
	SUB	B	Subtract A – reg from B- reg.
	INR	C	Increment content of register C.
	JMP	NEXT	Jump to NEXT
LOOP:	STA	4152	Store the remainder in Memory
	MOV	A, C	Move the Content of C register to Accumulator
	STA	4153	Store the quotient in memory
	HLT		Terminate the program.

SAMPLE INPUT & OUTPUT

Input: FF (4150)
FF (4251)

Output: 01 (4152) ---- Remainder
FE (4153) ---- Quotient

RESULT

Thus the program to divide two 8-bit numbers was executed.

QUESTIONS RELATED TO THE NEXT EXPERIMENT:

1. What is XCHG instruction?
2. What is DAD instruction?
3. Explain about SBB instruction.
4. Explain about SPHL instruction.
5. Difference between SHLD and STA.

EX. NO.2A
ADDITION OF TWO 16-BIT NUMBERS

AIM

To write an Assembly Language Program (ALP) for performing 16 bit addition.

ALGORITHM

1. Initialize the MSBs of sum to 0
2. Get the first number.
3. Add the second number to the first number.
4. If there is any carry, increment MSBs of sum by 1.
5. Store LSBs of sum.
6. Store MSBs of sum

SOURCE CODE

	LHLD	7601H	Get 1st no. in HL pair from memory 7601
	XCHG		Exchange cont. of DE HL
	LHLD	7603H	Get 2st no. in HL pair from location 7603
	MVI	C, 00H	Clear reg. C.
	DAD	D	Get HL+DE & store result in HL
	JNC	LOOP	If no carry move to loop/if carry then move to next step.
	INR	C	Increment reg C
LOOP:	MOV	A, C	Move carry from reg. C to reg.A
	STA	7502H	Store carry at 7502H
	SHLD	7500H	Store result in 7500H.
	HLT		

SAMPLE INPUT & OUTPUT

Input: 7601 77
7602 66
7603 44
7604 22

Output: 7502 BB
7503 88
7500 00

RESULT

Thus the program to add two 16-bit numbers was executed.

EX. NO.2B

SUBTRACTION OF TWO 16 BIT NUMBERS

AIM

To write an Assembly Language Program (ALP) for performing 16 bit subtraction.

ALGORITHM

1. Initialize the MSBs of difference to 0
2. Get the first number.
3. Subtract the second number from the first number.
4. If there is any borrow, increment MSBs of difference by 1.
5. Store LSBs of difference
6. Store MSBs of difference.

SOURCE CODE

	MVI	C,00H	Move immediate 00 value to C
	LHLD	5500H	Load HL pair with value from address
	XCHG		Exchange HL & DE values
	LHLD	5502	Load HL pair with value from address
	MOV	A,E	Move E to accumulator
	SUB	L	Subtract L
	JNC	LOOP 1	If no carry exists, go to loop 1
	CMA		Complement accumulator
	INR	A	Increment accumulator
LOOP 1	STA	5900	Store accumulator value in address
	MOV	A,D	Move D to accumulator
	SUB		Subtract H
	JNC	LOOP 2	If no carry exist go to LOOP 2
	CMA		Complement accumulator
	INR	A	Increment accumulator
	INC	C	Increment C
LOOP 2	STA	5901H	Store accumulator value in address
	MOV	A,C	Move C to accumulator
	STA	5902	Store accumulator value in address
	HLT		End program

SAMPLE INPUT & OUTPUT

Input: 5500 44
5501 22
5502 77
5503 66

Output: 5900 33
5901 44
5902 01

RESULT

Thus the program to subtract two 16-bit numbers was executed.

EXERCISE NO.2C

MULTIPLICATION OF TWO 16-BIT NUMBERS

AIM

To write an Assembly Language Program (ALP) for performing 16 bit multiplication.

ALGORITHM

1. Get the multiplier.
2. Get the multiplicand
3. Initialize the product to 0.
4. Product = product + multiplicand
5. Decrement the multiplier by 1
6. If multiplicand is not equal to 0, repeat from step (4) otherwise store the product.

SOURCE CODE

```
        LHLD    8500        Load HL pair with values from address
        SPHL                Exchange stack pointer & HL
        LHLD     8502H      Load HL pair with values from address
        XCHG                Exchange DE & HL values
        LXI H,  0000 H      Load immediate values in HL pair
        LXI B,  000 H       Load immediate value in BC pair
        Next     DAD SP     Add stack pointer to HL
        JNC      LOOP       If no carry exists go to loop
        INX      B          Increment BC pair
LOOP :  DCX      D          Decrement DE pair
        MOV      A,E        Move E to accumulator
        ORA      D          Perform OR with D & accumulator
        JNZ      Next       if not zero go to next
        SHL      D 8504     Store HL pair value in address
        MOV      L,C        Move C to L
        MOV      H,B        Move B to H
        SHLD     8506H      Store HL pair value in address
        HLT                 End Program
```

SAMPLE INPUT & OUTPUT

Input:
8500 01
8501 F0
8502 02
8503 F0

Output:
8504 02
8505 00
8506 02
8507 E1

RESULT

Thus the program to multiply two 16-bit numbers was executed.

EX. NO.2D

DIVISION OF TWO 16-BIT NUMBERS

AIM

To write an Assembly Language Program (ALP) for performing 16 bit division.

ALGORITHM

1. Get the dividend
2. Get the divisor
3. Initialize the quotient to 0.
4. Dividend = dividend – divisor
5. If the divisor is greater, store the quotient. Go to step g.
6. If dividend is greater, quotient = quotient + 1. Repeat from step (4) Store the dividend value as remainder.

SOURCE CODE

```
LXI B, 0000H          Load immediate value in BC pair
LHLD 4500             Load HL pair with value from memory
XCHG                  Exchange HL & DE vales
LHLD 4502H            Load HL pair with value from memory
LOOP1     MOV A, L    Move L to accumulator
SUB E                 Subtract E
MOV L, A              Move accumulator to L
MOV A, H              Move H to accumulator
SUB D                 Subtract D with borrow
MOV H, A              Move accumulator to H
JM LOOP1              If minus go to LOOP1
INX B                 Increment BC pair
JMP LOOP2             Jump to LOOP2
LOOP2     DAD D       Add DE to HL
SHLD 4602H            Store HL pair at address
MOV L,C               Move C to L
MOV H,B               Move B to H
SHLD 4604             Store HL pair value at address
HLT                   End Program
```

SAMPLE INPUT & OUTPUT

```
Input:  4500   02
4501   02
4502   03
4503    03
Output :4602   02
 4603   02
4604   03
4605    03
```

RESULT

Thus the program to divide two 16-bit numbers was executed.

QUESTIONS RELATED TO THE NEXT EXPERIMENT:

1. Explain about CMP instruction.
2. Difference between INX and INR.
3. Difference between DCX and DCR.
4. What all are the conditional jump instruction in 8085.
5. What is LXI instruction?

EXP. NO: 3

LARGEST NUMBERS IN AN ARRAY OF DATA

OBJECTIVE

To find the largest number in an array of data using 8085 instruction set.

ALGORITHM

STEP 1: Load the address of the first element of the array in HL pair
STEP 2: Move the count to B – reg.
STEP 3: Increment the pointer
STEP 4: Get the first data in A – reg.
STEP 5: Decrement the count.
STEP 6: Increment the pointer
STEP 7: Compare the content of memory addressed by HL pair with that of A - reg.
STEP 8: If Carry = 0, go to step 10 or if Carry = 1 go to step 9
STEP 9: Move the content of memory addressed by HL to A – reg.
STEP 10: Decrement the count
STEP 11: Check for Zero of the count. If ZF = 0, go to step 6, or if ZF = 1 go to next step.
STEP 12: Store the largest data in memory.
STEP 13: Terminate the program.

SOURCE CODE

	LXI	H,4200	Set pointer for array
	MOV	B,M	Load the Count
	INX	H	
	MOV	A,M	Set 1st element as largest data
	DCR	B	Decrement the count
LOOP:	INX	H	
	CMP	M	If A- reg > M go to AHEAD
	JNC	AHEAD	
	MOV	A,M	Set the new value as largest
AHEAD:	DCR	B	
	JNZ	LOOP	Repeat comparisons till count = 0
	STA	4300	Store the largest value at 4300
	HLT		

SAMPLE INPUTS & OUTPUTS

Input:	05 (4200) ----- Array Size
	0A (4201)
	F1 (4202)
	1F (4203)
	26 (4204)
	FE (4205)
Output:	FE (4300)

RESULT

Thus the program to find the largest number in an array of data was executed

QUESTIONS RELATED TO THE NEXT EXPERIMENT:

1. List the data transfer instructions.
2. List out the logical instructions.
3. What is difference between JC and JNC?.
4. What is the use of CMP instruction?
5. Write about increment and decrement Instruction.

EXP. NO: 4

SMALLEST NUMBERS IN AN ARRAY OF DATA

OBJECTIVE:

To find the smallest number in an array of data using 8085 instruction set.

ALGORITHM:

STEP 1: Load the address of the first element of the array in HL pair
STEP 2: Move the count to B – reg.
STEP 3: Increment the pointer
STEP 4: Get the first data in A – reg.
STEP 5: Decrement the count.
STEP 6: Increment the pointer
STEP 7: Compare the content of memory addressed by HL pair with that of A - reg.
STEP 8: If carry = 1, go to step 10 or if Carry = 0 go to step 9
STEP 9: Move the content of memory addressed by HL to A – reg.
STEP 10: Decrement the count
STEP 11: Check for Zero of the count. If ZF = 0, go to step 6, or if ZF = 1 go to next step.
STEP 12: Store the smallest data in memory.
STEP 13: Terminate the program.

SOURCE CODE

	LXI	H,4200	Set pointer for array
	MOV	B,M	Load the Count
	INX	H	
	MOV	A,M	Set 1st element as largest data
	DCR	B	Decrement the count
LOOP:	INX	H	
	CMP	M	If A- reg < M go to AHEAD
	JC	AHEAD	
	MOV	A,M	Set the new value as smallest
AHEAD:	DCR	B	
	JNZ	LOOP	Repeat comparisons till count = 0
	STA	4300	Store the largest value at 4300
	HLT		

SAMPLE INPUTS & OUTPUTS

Input:	05 (4200) ----- Array Size
	0A (4201)
	F1 (4202)
	1F (4203)
	26 (4204)
	FE (4205)
Output:	0A (4201)

RESULT

Thus the program to find the smallest number in an array of data was executed

QUESTIONS RELATED TO THE NEXT EXPERIMENT:

1. Write about BCD system.
2. How will you convert BCD to hexadecimal?
3. What is the use if INX instruction?
4. Write various JMP operations?
5. How will you convert hexadecimal to BCD?

EX.NO. 5A

BCD TO HEX CONVERSION

AIM:

To convert two BCD numbers in memory to the equivalent HEX number using 8085 instruction set

ALGORITHM:

STEP 1: Initialize memory pointer to 4150 H
STEP 2: Get the Most Significant Digit (MSD)
STEP 3: Multiply the MSD by ten using repeated addition
STEP 4: Add the Least Significant Digit (LSD) to the result obtained in previous step
STEP 5: Store the HEX data in Memory

SOURCE CODE

```
LXI     H,4150
MOV     A,M     Initialize memory pointer
ADD     A       MSD X 2
MOV     B,A     Store MSD X 2
ADD     A       MSD X 4
ADD     A       MSD X 8
ADD     B       MSD X 10
INX     H       Point to LSD
ADD     M       Add to form HEX
INX     H
MOV     M,A     Store the result
HLT
```

SAMPLE INPUTS & OUTPUTS

Input: 4150 : 02 (MSD)
4151 : 09 (LSD)
Output: 4152 : 1D H

RESULT

Thus the program to convert BCD data to HEX data was executed.

EX.NO. 5B

HEX TO BCD CONVERSION

AIM

To convert given Hexa decimal number into its equivalent BCD number using 8085 instruction set

ALGORITHM

STEP 1: Initialize memory pointer to 4150 H
STEP 2: Get the Hexa decimal number in C - register
STEP 3: Perform repeated addition for C number of times
STEP 4: Adjust for BCD in each step
STEP 5: Store the BCD data in Memory

SOURCE CODE

```
        LXI     H,4150
        MVI     D,00
        XRA     A
        MOV     C,M
LOOP2:  ADI     01
        DAA
        JNC     LOOP1
        INR     D
LOOP1:  DCR     C
        JNZ     LOOP2
        STA     4151
        MOV     A,D
        STA     4152
        HLT
```

SAMPLE INPUTS & OUTPUTS

Input: 4150: FF

Output: 4151: 55 (LSB)
4152: 02 (MSB)

RESULT

Thus the program to convert HEX data to BCD data was executed.

QUESTIONS RELATED TO THE NEXT EXPERIMENT

1. What is HEX number?
2. Explain steps to convert HEX number to BCD number?
3. Explain various addressing modes of 8086 used in HEX to BCD conversion program?
4. Explain different assembler directives used in HEX to BCD conversion program?
5. Explain various number systems used in digital electronics?

EX.NO.6A

BINARY TO BCD CODE CONVERSIONS

AIM

To write an assembly language program to convert an 8 bit binary data to BCD using 8085 microprocessor kit.

ALGORITHM

STEP 1: Start the microprocessor
STEP 2: Clear 'D' and 'E' register to account for hundred's and ten's load the binary data in Accumulator
STEP 3: Compare 'A' with 64 if cy = 01, go step C otherwise next step
STEP 4: Subtract 64 from (64+1) 'A' register
STEP 5: Increment 'E' register
STEP 6: Compare the register 'A' with '0A', if cy=1, go to step 11, otherwise next step
STEP 7: Subtract (0AH) from 'A' register
STEP 8: Increment D register Step 9 : Go to step 7
STEP 10: Combine the units and tens to from 8 bit result
STEP 11: Save the units, tens and hundred's in memory
STEP 12 : Stop the program execution

SOURCE CODE:

```
        MVI   E,00
        MOV   D,E
        LDA   4200
HUND    CPI   64
        JC    TEN
        SUI   64
        INR   E
        JMP   HUND
TEN     CPI   0A
        JC    UNIT
        SUI   0A
        INR   D
        JMP   TEN
UNIT    MOV   4A
        MOV   A,D
        RLC
        RLC
        RLC
        RLC
        ADD
        STA
        HLT
```

SAMPLE INPUTS & OUTPUTS

Input: 4200 : 54

Output: 4250 : 84

RESULT

Thus the binary to BCD conversion was executed successfully

EX.NO.6B

BCD TO BINARY CODE CONVERSIONS

AIM

To write an assembly language program to convert BCD data to Binary data using 8085 microprocessor kit

ALGORITHM

STEP 1 : Start the microprocessor
STEP 2 : Get the BCD data in accumulator and save it in register ‘E’
STEP 3 : Mark the lower nibble of BCD data in accumulator
STEP 4 : Rotate upper nibble to lower nibble and save it in register ‘B’
STEP 5 : Clear the accumulator
STEP 6 : Move 0AH to ‘C’ register
STEP 7 : Add ‘A’ and ‘B’ register
STEP 8 : Decrement ‘C’ register. If zf = 0, go to step 7
STEP 9 : Save the product in ‘B’
STEP 10 : Get the BCD data in accumulator from ‘E’ register and mark the upper nibble
STEP 11 : Add the units (A-ug) to product (B-ug)
STEP 12 : Store the binary value in memory
STEP 13 : End the program

SOURCE CODE

```
LDA 4200
MOV E,A
ANI F0
RLC
RLC
RLC
RLC
MOV B,A
XRA A
MVI C,0A
REP
DCR C
JNZ
MOV B,A
MOV A,E
ANI 0F
ADD B
STA 4201
HLT
```

SAMPLE INPUTS & OUTPUTS

Input : 4200 : 84
Output: 4201 : 54

RESULT

Thus the BCD to binary conversion was executed successfully.

QUESTIONS RELATED TO THE NEXT EXPERIMENT:

1. What is a counter?
2. Explain how counters are used in loop instructions?
3. What is meant by time delay?
4. Explain how to calculate execution delay or delay sub-routine?
5. Difference between time delay in loop and nested loop?

EX.NO.7

COUNTER AND TIME DELAY (DECIMAL UPCOUNTER)

AIM

To write an ALP to implement a counter to count from ’00 – 99’ (UPCOUNTER) in BCD by Using a subroutine to generate a delay of one second between the counts.

ALGORITHM

STEP 1: Initiate the minimum number in Accumulator
STEP 2: Display in the DATA field
STEP 3: Add 01 to the present value Displayed
STEP 4: Use decimal conversion Instruction.
STEP 5: Repeat the steps 2-4.
STEP 6: Provide proper display between Each display.
STEP 7: Terminating Point.

SOURCE CODE

```
MVI A,00H
LOOP1: MOV H,A
CALL OUT
CALL DELAY
MOV A,H
ADI 01H
DAA
JMP LOOP1
HLT
DELAY: LXI B, FFFFH
WAIT:    DCX B
MOV A,C
ORA B
JNZ WAIT
RET
OUT:  MVI A,02H
CALL 0005H
MVI A,0CH
MVI C,00H
MOV D,H
CALL 0005H
RET
```

SAMPLE OUTPUT

0	0
0	1
.	.
.	.
9	8
9	9

RESULT

It counts from 00 to 99 with the given delay in DATA field.

QUESTIONS RELATED TO THE NEXT EXPERIMENT

1. What is overlapping?
2. What is meant by a data block?
3. What is overlapped block transfer?
4. What is the difference between overlapped and non-overlapped block transfer?
5. Say some of the data transfer instructions?

EXP. NO: 8

MOVE A DATABLOCK WITHOUT OVERLAP

OBJECTIVE

To write an Assembly Language Program to transfer a data block without overlap using 8085

ALGORITHM:

STEP 1: Load the DE pair with the destination address.
STEP 2: Load the HL pair with the count of elements in the data block.
STEP 3: Load element in the data block.
STEP 4: Increment the source address.
STEP 5: Copy the element to the accumulator and then transfer it to the destination address.
STEP 6: Increment destination address.
STEP 7: Decrement the count.
STEP 8: If Count = 0 then go to the next step else go to step 3.
STEP 9: Terminate the program.

SOURCE CODE

LABEL	MNEMONIC	COMMENT
	LXI D,4500	Load destination address in DE pair
	LXI H,4100	Load the count in HL pair
	MOV C,M	Copy the count to register C
LOOP	INX H	Increment memory
	MOV A,M	Copy element to Accumulator
	STAX D	Store the element to the address in the DE pair
	INX D	Increment destination address
	DCR C	Decrement count
	JNZ LOOP	Jump on non-zero to the label LOOP
	HLT	Program ends

SAMPLE INPUTS & OUTPUTS

Input at	4100	:	04_H
	4101	:	06_H
	4102	:	07_H
	4103	:	12_H
	4104	:	03_H
Output at	4500	:	06_H
	4501	:	07_H
	4502	:	12_H
	4503	:	03_H

RESULT

Thus the program to move data without overlap was executed

QUESTIONS RELATED TO THE NEXT EXPERIMENT

1. List out the arithmetic instructions of 8086.
2. List out the logical instructions in 8086.
3. What is difference between ADD and ADC?
4. Explain XOR operation.
5. Write about registers in 8086.

B.8086 PROGRAMS

EXP NO:9

BASIC ARITHMETIC & LOGICAL OPERATIONS

OBJECTIVE

To perform the basic arithmetic and logical operations using the 8086 Microprocessor emulator

9A. ADDITION

ALGORITHM

Step 1. Allocate some space for the result in data segment
step 2. In code segment, store accumulator with some value
step 3. Store B register with some value
step 4. Add the register content with accumulator
step 5. Result is stored in accumulator
step 6. The result is stored in required memory location

SOURCE CODE

```
Start:  mov AX, 05H
mov BX, 03H
ADD AX,BX
end:  HLT
```

SAMPLE INPUTS& OUTPUTS

Before Execution:	After Execution:
AX = 0005H	AX = 0008H
BX = 0003H	

9B. SUBTRACTION

ALGORITHM

a) Start the program.
b) Allocate some space for the result in data segment
c) In code segment, store accumulator with some value
d) Store B register with some value
e) Subtract the register content from the accumulator
f) Result is stored in accumulator
g) The result is stored in required memory location
h) Stop the program.

SOURCE CODE

```
Start:  mov AX, 05H
mov BX, 03H
SUB AX,BX
end:  HLT
```

SAMPLE INPUTS & OUTPUTS

INPUT: 0005H ,0003H

OUTPUT: 0002H

9.C MULTIPLICATION

ALGORITHM

a) Start the program
b) Allocate some space for the result in data segment
c) In code segment,store accumulator with some value
d) Store B register with some value
e) Multiply the register content with accumulator
f) Result is stored in accumulator
g) The result is stored in required memory location
h) Stop the program.

SOURCE CODE

```
Start: mov AX, 05H
mov BX, 03H
MUL AX,BX
end: HLT
```

SAMPLE INPUTS & OUTPUTS

INPUT: 0006H, 0004H

OUTPUT: 0018H

9D.DIVISION

ALGORITHM:

a) Start the program.
b) Allocate some space for the result in data segment
c) Take 2 data as 2 inputs in 2 different registers
d) Perform the Division operation.
e) The quotient is stored in accumulator and the remainder is stored in D register
f) Store the remainder and quotient in required memory location.
g) Display the result.
h) Stop the program.

SOURCE CODE

```
Start: mov AX, 08H
mov BX, 02H
DIV AX,BX
end: HLT
```

SAMPLE INPUTS & OUTPUTS

INPUT: 0008H ,0002H
OUTPUT: 0004H

9E.LOGICAL AND OPERATION

ALGORITHM

Step 1. Allocate some space for the result in data segment
step 2. In code segment, store accumulator with some value
step 3. Store B register with some value
step 4. Perform AND operation on the register content with accumulator
step 5. Result is stored in accumulator
step 6. The result is stored in required memory location

SOURCE CODE

```
Start:  mov AX, 01H
        mov  BX, 01H
        AND AX,BX
End:  HLT
```

SAMPLE INPUTS & OUTPUTS

Before Execution: After Execution:
AX = 0001H AX = 0001H
BX = 0001H

9F. LOGICAL OR OPERATION

ALGORITHM

Step 1. Allocate some space for the result in data segment
step 2. In code segment, store accumulator with some value
step 3. Store B register with some value
step 4. Perform OR operation on register content with accumulator
step 5. Result is stored in accumulator
step 6. The result is stored in required memory location.

SOURCE CODE

```
Start:  mov AX, 01H
mov BX, 00H
OR AX,BX
end:  HLT
```

SAMPLE INPUTS & OUTPUTS

Before Execution:	After Execution:
AX = 0001H	AX = 0001H
BX = 0000H	

RESULT

The machine programs for basic arithmetic and logical operations were successfully implemented Using8086 emulator.

QUESTIONS RELATED TO THE NEXT EXPERIMENT

1.How to convert binary to BCD by giving the input in hexa?
2.What instruction is used to scan the character of string?
3.What is procedure ?
4.What is the use of data segment and how to get data as array?
5.How to display a msg?

EXP. NO: 10 A

CODE CONVERSIONS – BINARY TO BCD

OBJECTIVE

To convert a given binary to BCD.

ALGORITHM:

Step 1: Initialize the data to the data segment.
Step 2: Move the input to AX register.
Step 3: Move 64 to CL register
Step 4: Divide AL, CL value
Step 5: Increment memory by 1 and move AL value
Step 6: Move AH value to AL
Step 7: Move 0A to CL register
Step 8: Divide the AL, CL
Step 9: Rotate CL register 4 times
Step 10: Add AH, AL
Step 11: Store the resultant in memory location.

SOURCE CODE

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
        BIN     DW      01A9H
        BCD     DB      2  DUP  (0)
DATA ENDS
CODE SEGMENT
START:
        MOV   AX, DATA
        MOV   DS, AX
        MOV   AX, BIN
        MOV   CL, 64H
        DIV     CL
        MOV   BCD+1, AL
        MOV   AL, AH
        MOV   AH, 00H
        MOV   CL, 0AH
        DIV     CL
        MOV    CL, 04
        ROR    AL, CL
        ADD    AL, AH
        MOV   AH, 4CH
        INT     21H
CODE  ENDS
        END    START
```

OUTPUT

```
INPUT         :  01A9H
OUTPUT       :  0425
```

RESULT

Thus the program to convert a binary to BCD was executed.

EX. NO: 10 B

SORTING

OBJECTIVE

To sort the given number in ascending order using 8086.

ALGORITHM

Step 1: Get the input number from memory and move it to AL register
Step2: Move the count value to DX register (outer loop)
Step3: Decrement the value of DX by one and move it to CX register (inner loop)
Step4: Compare the AL and the next element in the memory
Step5: If CY=1 then AL is less than next element
Step6: If CY=0 then AL is greater than next element so exchange both value
Step7: Continue the step3 to step7 until CX and DX goes to zero.
Step8: Store the resultant value

SOURCE CODE

```
ASSUME CS: CODE, DS:DATA
DATA SEGMENT
        SERIES    DB    81H,82H,93H,95H,10H,56H,33H,99H,13H,44H
        COUNT     DW    10H
DATA ENDS
CODE SEGMENT
START:
        MOV  AX, DATA
        MOV  DS, AX
        MOV  DX, COUNT
        DEC  DX
GO:
        MOV   CX, DX
        LEA   SI, SERIES
NXT_BYTE:
        MOV   AL,[SI]
        CMP   AL,[SI+1]
        JB    NEXT
        XCHG    AL,[SI+1]
        XCHG    AL,[SI]
NEXT:
      INC    SI
      LOOP NXT_BYTE
      DEC   DX
      JNZ    GO
      MOV  AH, 4CH
      INT    21H
CODE ENDS
      END START
```

INPUT:

50000 81H
50002 82H
50004 93H
50006 95H
50008 10H
5000A 56H
5000C 33H
5000E 99H
50010 13H
50012 44H

OUTPUT:

50000 10H
50002 13H
50004 33H
50006 44H
50008 56H
5000A 81H
5000C 82H
5000E 93H
50010 95H
50012 99H

RESULT

Thus the program to Sort the given array in ascending order was executed successfully.

EX . NO: 10 C

SEARCHING A STRING

OBJECTIVE

To search the character in a string using 8086.

ALGORITHM

Step 1: Load the source index register with starting address.
Step 2: Initialize the counter with the total number of characters.
Step 3: Clear the direction flag for auto incrementing mode of transfer.
Step 4: Use the string manipulation instruction SCASW to search a character from string.
Step 5: If a match is found (z=1), display the MSG1. Otherwise, display the MSG2.

SOURCE CODE

```
ASSUME CS: CODE, DS: DATA, ES:DATA
DATA SEGMENT
        MSG   DB      'HELLO'
        CNT   EQU   $-MSG
        SRC   EQU   'E'
        MSG1 DB     10,13,'CHARACTER FOUND$'
        MSG2 DB     10,13,'CHARACTER NOT FOUND$'
DATA ENDS
CODE SEGMENT
START:
        MOV   AX, DATA
        MOV   DS, AX
        MOV   ES, AX
        LEA   SI, MSG
        MOV   AL, SRC
        MOV   CL, CNT
        MOV   CH, 00H
        CLD
UP:    SCASB
        JZ      DOWN
        LOOP  UP
        LEA   DX, MSG2
        MOV   AH, 09H
        INT    21H
        JMP    EXIT
DOWN:
        LEA   DX, MSG1
        MOV   AH, 09H
        INT    21H
EXIT:
        MOV   AH, 4CH
        INT    21H
CODE ENDS
        END START
```

OUTPUT :

INPUT: HELLO
SEARCH: E

OUTPUT:
CHARACTER FOUND

RESULT

Thus the program to search the character in a string was executed.

LIST OF QUESTION FOR NEXT EXPERIMENT

1. What is the operation of XLAT instruction?
2. Compare LEA and LES instruction.
3. List out the steps how PUSH AX instruction stores the value in the stack(AX=324B).
4. What is the purpose of XCHG instruction?
5. What is the use of POPF instruction?

EXP. NO: 11

DATA TRANSFER OPERATIONS

OBJECTIVE

To write a Program using 8086 for Copying 12 Bytes of Data from Source to Destination & Verify.

ALGORITHM:

STEP 1: Start the program
STEP 2: Clear the direction flag DF
STEP 3: Move source address to SI
STEP 4: Move destination address in DI
STEP 5: Increment the count and index register
STEP 6: Move Byte
STEP 7: Terminate the program

SOURCE CODE

Mnemonics	Operands	Comments
CLD		Clear direction flag DF
MOV	SI,0300	Source address in SI
MOV	DI,0202	Destination address in DI
MOV	CX,[SI]	Count in CX
INC	SI	Increment SI
INC	SI	Increment SI
MOV	SB	Move byte
LOOP	BACK	Jump to BACK until CX becomes Zero
INT		Interrupt program

SAMPLE INPUTS & OUTPUTS

INPUT DATA

0300 : 0B
0301 : 00
0302 : 03
0303 : 04
0304 : 05
0305 : 06
0306 : 15
0307 : 07
0308 : 12
0309 : 08
030A : 09
030B : 0A
030C : 0B
030D : 0E

OUTPUT DATA

0202 : 03
0203 : 04
0204 : 05
0205 : 06
0206 : 15
0207 : 07

RESULT

Thus the program Copying 6 Bytes of Data from Source to Destination was executed

FEW (MIN. 5) QUESTIONS RELATED TO THE NEXT EXPERIMENT

1. What are the DOS function calls?
2. How a CALL instruction will be executed?
3. What is the role of stack?
4. What is the difference between DOS and BIOS interrupts?
5. What is an interrupt vector table of 8086?

EXP. NO: 12

PASSWORD CHECKING

AIM

To write an ALP program for password checking using 8086.

ALGORITHM:

- Create a display micro
- Initialise a counter for max number of attempts available
- In the data segment store the password in a location
- In the code segment accept the string one by one and compare the stored value
- If the strings are equal display “valid password”
- If they are not equal then display invalid password
- Code ends

SOURCE CODE

```
disp macro x
        mov ah,09h
        lea dx,x
        int 21h
endm
data segment
        s db 13,10,"enter string:$"
        u db 13,10,"right password $"
        r db 13,10,"invalid $"
        m1 db '*$'
        m2 db 13,10,"try again $"
        pwd db "cmt $"
data ends
code segment
        assume cs:code,ds:data
        start:
                mov ax,data
                mov ds,ax
                mov ax,0003h
                int 10h
                mov bl,03h
        a1:
                mov cl,03h
                mov si,00h
                disp s
        a2:
                mov ah,08h
                int 21h
                cmp al,pwd[si]
                disp m1
                jne l1
                inc si
                loop a2
                disp u
```

```
        jmp l2
l1:     dec bl
        disp r
        disp m2
        cmp bl,00h
        je l2
        jne a1
l2:
        mov ax,4c00h
        int 21h
code ends
end start
```

OUTPUT

```
enter the password ***
right password
```

RESULT

Thus the ALP program for password checking using 8086 was executed

FEW (MIN. 5) QUESTIONS RELATED TO THE NEXT EXPERIMENT:

1. Explain the assembler directives.
2. What are the flags in 8086?
3. What is SIM and RIM instructions?
4. What is the difference between 8086 and 8088?
5. Which is the tool used to connect the user and the computer?

EXP. NO: 13

PRINT RAM SIZE AND SYSTEM DATE

OBJECTIVE

To write a program to Print RAM size and system date using 8086.

ALGORITHM

STEP 1: Create a display micro
STEP 2: C Initialise a Initialise the necessary register with the required values.
STEP 3: In Use a macro to display system date.
STEP 4: Terminate the program.

SOURCE CODE

```
Print RAM size:
PRINT MACRO MSG
MOV AH,09H
LEA DX,MSG
INT 21H
ENDM
DATA SEGMENT
ML1 DB 13,10,'SIZE OF RAM IS $'
M2 DB 'KILO BYTES $'
ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DAT
START:
MOV DX,DATA
MOV DS,DX
MOV AX,0003H
INT 10H
PRINT M1
INT 12H
MOV B1,64H
DIV B1
MOV CH,AH
ADD A1,'0'
MOV D1,A1
MOV AH,02H
INT 21H
MOV A1,CH
MOV AH,00H
MOV B1,0AH
DIV B1
ADD A1,'0'
ADD AH,'0'
MOV CH,AH
MOV AH,02H
INT 21H
```

```
    MOV D1,CH
    MOV AH,02H
    INT 21H
    PRINT M2
    MOV AX,4C00H
    INT 21H
    CODE ENDS
    END START
System Date:
    DISP MACRO X
            PUSH DX
            MOV AH,09H
            LEA DX,X
            POP DX
        ENDM
        PRINT MACRO
            MOV BH,0AH
            MOV AH,00H
            DIV BH
            ADD AL,'0'
            ADD AH,'0'
            MOV BH,AH
            MOV DL,AL
            MOV AH,02H
            INT 21H
        ENDM
        MYDATE SEGMENT
            S DB 13,10,'THE DATE IS:'$'
            C DB,'/$'
        MYDATAE ENDS
        MYCODE SEGMENT
        ASSUME CS:MYCODE,DS:MYDATE
START:      MOV AX,MYDATE
        MOV DS,AX
        MOV AX,0003H
        INT 10H
        DISP S
        MOV AH,2AH
        INT 21H
        MOV AL,DL
        MOV BL,DH
        PRINT
        DISP C
        MOV AL,BL
         PRINT
        DISP C
        MOV AX,CX
        MOV BX,03E8H
        MOV DX,0000H
        DIV BX
```

```
    MOV CX,DX
    MOV DL,AL
    ADD DL,'0'
    MOV AH,02H
    INT  21H
    MOV AX,CX
    MOV DX,0000H
    MOV BX,0064H
    DIV BX
    MOV CX,DX
    ADD AL,'0'
    MOV DL,AL
    MOV AH,02H
    INT 21H
    MOV AX,CX
    MOV BL,0AH
    DIV BL
    MOV BH,AH
    ADD AL,'0'
    MOV AH,02H
    INT  21H
    ADD BH,'0'
    MOV DL,BH
    MOV AH,02H
    INT 21H
    MOV AX,4C00H
    INT 21H
MYCODE ENDS
END START
```

SAMPLE INPUTS & OUTPUTS:

The Date is : 07-08-2015

The size of RAM is : 1 GB

RESULT

Thus the program to Print RAM size and system date using 8086 was executed

QUESTIONS RELATED TO THE NEXT EXPERIMENT

1. What is the role of stack?
2. What is the role of Call delay?
3. What is an interrupt vector table of 8086?
4. Which Segment is used to store interrupt and subroutine return address registers?
5. Which microprocessor accepts the program written for 8086 without any changes?

C.PHERIPHERALS AND INTERFACING EXPERIMENTS

EX. NO: 14

TRAFFIC LIGHT CONTROLLER

OBJECTIVE

To write and implement the program for traffic light controller using 8085.

ALGORITHM

STEP 1: Init PA &PB as output
STEP 2: Stop all four ends
STEP 3: GO STR signal of North & South, STOP signal of East &West
STEP 4: Alert signal for traffic
STEP 5: GO LEFT signal of North & South
STEP 6: STOP signal of North & South
STEP 7: GO STR signal of East & West
STEP 8: STOP signal of East &West

SOURCECODE:

Label	Mnemonics	Operands	Comments
	MVI	A,80H	Init PA &PB as output
	OUT	03H	
	MVI	A,11H	Stop all four ends
	OUT	00H	
	OUT	02H	
	CALL	DELAY1	
LOOP	MVI	A,44H	GO STR signal of North & South,STOP signal of East &West
	OUT	00H	
	CALL	DELAY1	
	MVI	A,22H	Alert signal for traffic
	OUT	00H	
	CALL	DELAY2	
	MVI	A,99H	GO LEFT signal of North & South
	OUT	00H	
	CALL	DELAY1	
	MVI	A,22H	Alert signal for traffic
	OUT	00H	
	CALL	DELAY2	
	MVI	A,11H	STOP signal of North & South
	OUT	00H	
	MVI	A,44H	GO STR signal of East & West
	OUT	02H	
	CALL	DELAY1	
	MVI	A,22H	Alert signal for traffic

```
                        02H
                        DELAY2
                                    GO Left signal of East
        MVI     A,99H   &West
        OUT     02H
        CALL    DELAY1
        MVI     A,22H   Alert signal for traffic
        OUT     02H
        CALL    DELAY2
                                    STOP signal of East
        MVI     A,11H   &West
        OUT     02H
        JMP     LOOP    Jump to loop
DELAY1: MVI     B,25H   Delay of 10 sec.
LP3:    MVI     C,0FFH
LP2:    MVI     D, 0FFH
LP1:    DCR     D
        JNZ     LP1
        DCR     C
        JNZ     LP2
        DCR     B
        JNZ     LP3
        RET
DELAY2: MVI     B,05H   Delay of 2 sec
LP6:    MVI     C,0FFH
LP5:    MVI     D,0FFH
LP4:    DCR     D
        JNZ     LP4
        DCR     C
        JNZ     LP5
        DCR     B
        JNZ     LP6
        RET
```

SAMPLE INPUTS & OUTPUTS

Traffic Signal Timing observed for four lane.

RESULT :

Thus the program for traffic light controller using 8085 is implemented and executed successfully

FEW (MIN. 5) QUESTIONS RELATED TO THE NEXT EXPERIMENT:

1. What is stepper motor?
2. What are the applications of stepper motor?
3. What are the values be given to rotate motor in clock wise direction?
4. What are the values be given to rotate motor in anti clock wise direction?
5. Whether Delay is used in the program of stepper motor are not and why?

EX. NO: 15

STEPPER MOTOR

OBJECTIVE

To write and implement the program for stepper motor using 8085

ALGORITHM:

STEP 1: For running stepper motor clockwise and anticlockwise directions Drive the stepper motor circuitry and introduce delay

STEP 2: Get the first data from the lookup table.

STEP 3: Initialize the counter and move data into accumulator.

STEP 4: Decrement the counter is not zero repeat from step(iii)

STEP 5: Repeat the above procedure both for backward and forward directions.

SOURCE CODE

LABEL	MNEMONICS	OPCODE	COMMENTS
			Initialize port A as output
	MVI	A,80	port.
	OUT	3	OB
START	MVI	AFA	
	OUT	0	Output code for step o.
	CALL	DELAY	delay between two steps.
			Location reserve for
	MVI	A, F6	current Delay
	OUT	OO	Output code for step 1.
	CALL	DELAY	delay between two steps.
	MVI	A, F5	
	OUT	OO	Output code for step 2.
	CALL	DELAY	between two steps.
	MVI	A, F9.	
	OUT	OO	Output code for step 3.
	CALL	DELAY	delay between two steps.
	JUMP	START	
DELAY:	LXI	D 00 00	Generates a delay.
	CALL	DELAY	
	LXI	D 00 00	Generates a delay.
	CALL	DELAY	
	RET		

SAMPLE INPUTS & OUTPUTS

Changing the following contents will change the motor speed.

ADDRESS	DATA			
2030	11 00 20	AND	2036 TO SIMILAR	11 00 20
CHANGE	11 00 10	TO		11 0010
CHANGE	11 00 05	TO		11 0005
CHANGE	11 00 03	TO		11 0003.

The motor direction depends upon codes FA, F6 , F5 AND F9.Change in following codes will change the motor direction.

ADDRESS	DATA		
2005	3E F9	TO	3E FA
200C	3E F5	TO	3E F6
2012	3E F6	TO	3E F5
2019	3E FA	TO	3E F9.

RESULT

Thus the program for stepper motor using 8085 is implemented and executed successfully

QUESTIONS RELATED TO THE NEXT EXPERIMENT:

1. What is Digital Clock?
2. What are the applications of Digital Clock?
3. What is the formula for frequency?
4. Why clock is required?
5. What pins are used in 8085 to connect the clock?

EX. NO:16

DIGITAL CLOCK

AIM

To write an ALP program for displaying the Digital clock.

ALGORITHM

- Create the display macro for string
- Initialise the necessary register with the required values.
- Use a macro to display clock value.
- End the code.

SOURCE CODE

```
                assume cs: code
                code segment
extern get_time:  near
.model small
.stack 100h
.data
time_buf db "00:00:00$"
code
main proc
                mov ax,@data
                mov ds, ax
                lea bx, time_buf
                call get_time
                lea dx, time_buf
                mov ah, 09h
                int 21h
                mov ah, 4ch
                int 21h
main endp
end main
```

OUTPUT

10:49:00

RESULT

Thus the program for displaying the digital clock was executed.

QUESTIONS RELATED TO THE NEXT EXPERIMENT:

1.What is macros?
2.What is TEST instruction?
3.What is LEA instruction?
4.What are status keys in keyboard?
5.What operands we can declare?

EX. NO:17
KEYBOARD STATUS

OBJECTIVE

To write an ALP program to display the keyboard status using 8086.

ALGORITHM

Step1: Load the AH register with 02H and call int 11H.Now the 8bits will be set/reset according to the key position

Step2: The one on every bit will indicate different keys on keyboard

Step3: Extract each bit by using bitwise AND operation and accordingly design code to display the status.

SOURCE CODE:

```
PRINT MACRO MSG
        MOV AH, 09H
        LEA DX, MSG
        INT 21H
ENDM
AA MACRO
        MOV AL,Z
ENDM
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
        MZ7   DB    13,10,'INSERT ON $'
        MZ6   DB    13,10,'CAPSLOCK ON $'
        MZ5   DB    13,10,'NUM LOCK ON $'
        MZ4   DB    13,10,'SCROLL LOCK ON $'
        MZ3   DB    13,10,'ALT KEY DOWN $'
        MZ2   DB    13,10,'CTRL KEY DOWN $'
        MZ1   DB    13,10,'LEFT SHIFT KEY DOWN $'
        MZ0   DB    13,10,'RIGHT SHIFT KEY DOWN $'
        Z     DB    1
DATA ENDS
CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:
        MOV AX, DATA
        MOV DS, AX          ;INITIALIZING
        MOV AX, 003H
        INT 10H
        MOV AX, 0000H
        MOV DX, 0000H
        MOV AH, 01H
        INT 21H
        MOV AH, 01H
        INT 21H
        MOV AH, 02H         ; GETTING KEYBOARD STATUS
        INT 16H
```

```
        MOV Z, AL
        TEST AL, 80H        ; TESTING FOR KEY STATUS
         JZ LAL
        PRINT MZ7
LA1:
        AA
        TEST AL, 40H
        JZ LA2
        PRINT MZ6
LA2:
        AA
        TEST AL, 20H
        JZ LA3
        PRINT MZ5
LA3:
        AA
        TEST AL, 10H
        JZ LA4
        PRINT MZ4
LA4:
        AA
        TEST AL, 08H
        JZ LA5
        PRINT MZ3
LA5:
        AA
        TEST AL, 04H
        JZ LA6
        PRINT MZ2
LA6:
        AA
        TEST AL, 02H
        JZ LA7
        PRINT MZ1
LA7:
        AA
        TEST AL, 01H
        JZ LA8
        PRINT MZ0
LA8:
        AA
        MOV AX, 4C00H
        INT 21H

CODE ENDS

END START
```

OUTPUT

```
F:\2IT16>KEY
INSERT ON
CAPSLOCK ON
NUM LOCK ON
SCROLL LOCK ON
LEFT SHIFT KEY DOWN
```

RESULT:

Thus the program to display the keyboard status was executed.