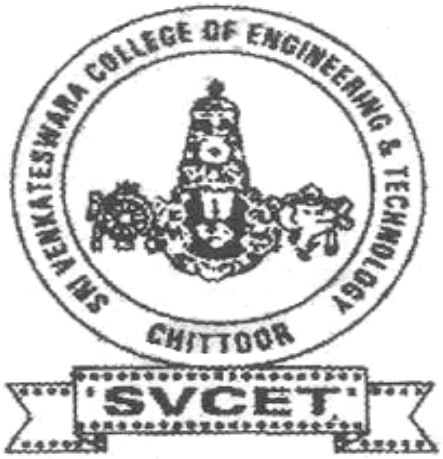
SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

Recognized by AICTE, NBA, NAAC and

Govt. of A.P. Affiliated by J.N.T.U.A., ANANTAPURAMU

R.V.S. Nagar, Tirupati Road, CHITTOOR- 517127



DEPARTMENT
OF
ELECTRONICS AND COMMUNICATION ENGINEERING

Microprocessors and Microcontrollers Lab (20AEC27)

Prepared by
Dr.D.SRIHARI M.E, Ph.D
Department of ECE

LIST OF EXPERIMENTS

(Minimum 12 experiments to be conducted)

PART-A

8086 Microprocessor Programs using MASM/TASM Software:

(Minimum 07 experiments to be conducted)

By using Arithmetic & Logical instructions:

1. Study of MASM/TASM Software.
2. ALP s (8086) for addition and subtraction.
3. ALP s (8086) for multiplication and Division.
4. ALP s (8086) to determine LCM and GCD of two 16-bit numbers.
5. ALP s (8086) to evaluate arithmetic expressions.
6. ALP s (8086) for sorting and searching.
7. Logic operations - Shift and rotate - Converting packed BCD to unpacked BCD, BCD to ASCII conversion.

By using String Instructions:

8. String operations - Move block, Reverse string, String comparison, Length of string.
9. ALP s (8086) for
   i) DOS interrupts
   ii) BIOS interrupts

PART-B

(Minimum 02 experiments to be conducted)

Interfacing Programs using 8086:

1. ALP s (8086) for generating ramp wave, triangular wave, and stair case wave forms using DAC.
2. ALP (8086) for traffic light controller.
3. ALP (8086) for stepper motor control.

PART-C

(Minimum 03 Experiments to be conducted)

8051 Microcontroller:

1. a) ALP (8051) to determine the addition.
   b) ALP (8051) to determine the subtraction.
2. a) ALP (8051) to determine the largest of N bytes.
   b) ALP (8051) to determine the smallest of N bytes.
3. a) ALP (8051) to multiply a 16-bit number by an 8-bit number.
   b) ALP (8051) to find square root of an 8-bit number.
4. a) ALP (8051) to determine LCM of two 8- bit numbers.
   b) ALP (8051) to determine GCD of two 8- bit numbers.
5. a) ALP (8051) to generate even numbers.
   b) ALP (8051) to generate odd numbers.
6. Timer/Counters (8051) in different modes.

Microprocessors and Microcontrollers Lab (20AEC27)

PART-A

S.NO	DATE	NAME OF THE EXPERIMENT	SIGNATURE OF THE FACULTY
1		Study of MASM/TASM Software	
2		ALPs (8086) for sum of 'n'8 bit numbers and Multi word addition.	
		ALPs (8086) for subtraction.	
3		ALPs (8086) for multiplication.	
		ALPs (8086) for Division.	
4		ALPs (8086) to determine LCM of two 16bit numbers	
		ALPs (8086) to determine GCD of two 16-bit numbers.	
5		ALPs (8086) to evaluate arithmetic expressions	
6		ALPs (8086) for sorting.	
		ALPs (8086) for searching.	
7		Logic operation- Converting packed BCD to unpacked BCD.	
		Logic operation- BCD to ASCII conversion.	
8		String operations - Move block, Reverse string,	
		String comparison, Length of string	
9		ALP s (8086) for i) DOS Interrupts	
		ii) BIOS Interrupts	
PART-B			
1		ALPs (8086) for generating ramp wave, triangular wave forms using DAC.	
		ALPs (8086) for generating stair case wave forms using DAC.	
2		ALP (8086) for traffic light controller	
3		ALP (8086) for stepper motor control.	

PART-C

S.NO	DATE	NAME OF THE EXPERIMENT	SIGNATURE OF THE FACULTY
1		ALP (8051) to determine the addition.	
		ALP (8051) to determine the subtraction.	
2		ALP (8051) to determine the largest of N bytes.	
		ALP (8051) to determine the smallest of N bytes.	
3		ALP (8051) to multiply a 16-bit number by an 8-bit number.	
		ALP (8051) to find square root of an 8-bit number	
4		ALP (8051) to determine LCM of two 8- bit numbers	
		ALP (8051) to determine GCD of two 8- bit numbers.	
5		ALP (8051) to generate even numbers	
		ALP (8051) to generate odd numbers.	
6		Timer/Counters (8051) in different modes. Mode 0 and Mode 1 operations	
		Mode 2 and mode 3 operations	

INDEX

PART-A

(Minimum 07 experiments to be conducted)

Date: **Exp. No. – 1**

Study of MASM/TASM Software

Turbo-Assembler (TASM)/ MASM (Macro Assembler)

Turbo Assembler (sometimes shortened to the name of the executable, TASM) is an assembler for software development published by Borland in 1989. **MASM,** It is an assembler that brings high-level language features to assembly language programming. It translates a single multi-argument source line o code into a sequence of machine instructions**.** TASM is an interactive means for assembling linking and debugging assembly language programs. Turbo- Assembler (TASM) is an integrated software package written by Borland Corporation for professional software developers.. It consists of an editor, an assembler, a linker and a debugger.

Assembling

The assembler is used to convert the assembly language instructions into machine code. It is used immediately after writing the Assembly Language program. The assembler starts by checking the syntax, or validity of the structure, of each instruction in the source file. If any errors are found, the assembler displays a report on these errors along with a brief explanation of their nature. However, if the program does not contain any errors, the assembler produces an object file that has the same name as the original file but with the "obj" extension.

Linking

The linker is used to convert the object file to an executable file. The executable file is the final set of machine code instructions that can directly be executed by the microprocessor. It is different than the object file in the sense that it is self-contained and re-locatable. An object file may represent one segment of a long program. This segment cannot operate by itself, and must be integrated with other object files representing the rest of the program, in order to produce the final self-contained executable file. In addition to the executable file, the linker can also generate a special file called the map file. This file contains information about the start, end, and the length of the stack, code, and data segments. It also lists the entry point of the program.

Executing

The executable file contains the machine language code. It can be loaded in the RAM and be executed by the microprocessor simply by typing, from the DOS prompt, Debug Filename.exe. If the program produces an output on the screen, or a sequence of control signals to control a piece of hardware, the effect should be noticed almost instantly.

All the steps and commands used to edit, assemble link and run a program using TASM is as follows:

1. Open TASM folder and click on DPMIRES icon and type EDIT to open the editor window.
2. Type the assembly language program and save the program as FILENAME.ASM.
3. Go to file menu and click on exit to move to DOS prompt.
4. Type **TASM FILENAME.ASM** command to assemble the program.
5. If any errors in the program use **EDIT FILENAME.ASM** command.
5. Type **TLINK FILENAME** command to link the file to turbo assembler.
6. Type **DEBUG FILENAME.EXE** to place the exe file in memory.
7. Type 'R' in the DOS prompt to see the register contents of 8086.
8. Type 'T' in the DOS prompt to observe the register contents used in the program, instruction by instruction until end of the program.
9. Type 'G' in the DOS prompt to run the entire program at a time.(if you use INT 03H in program)
10. To observe the results available in memory locations type **E 16-bit address** after executing the program.

The DOS "Debug" program is an example of simple debugger that comes with MS DOS. Hence, it is available on any PC. It was initially designed to give the user the capability to trace logical errors in executable files. It allows the user to take an existing executable file and unassembled it, i.e. convert it to assembly language. Also it allows the user to write assembly language instructions directly, and then convert them to machine language.

DOS commands:

R [register] It allows to show a contents of registers in the hexadecimal format;

SYNTAX: **R** [register]

T [trace] It allows to trace execution of one instruction;

SYNTAX: **T** [address] [value]

P [proceed] It allows to execute a group of instructions;

SYNTAX: **P** [address] [number]

Q [quit] It allows to quite a of DEBUG work;

SYNTAX: **Q**

E [enter] It allows to insert data into the memory, beginning from the given address;

SYNTAX: **E** [address]

D [dump] It allows to show the contents of memory in the hexadecimal format;

SYNTAX: **D** [range]

G [go] It allows to start execution of the given program in the memory;

SYNTAX: **G** [address]

A [assemble] It allows to transfer instructions, which are written in symbols into the machinery codes;

SYNTAX: **A** [address]

Date: **Exp. No. – 2**

ALPs (8086) for addition and subtraction

(A) Sum of given 'n' 8-bit numbers

AIM: To find out the sum of given 'n' 8-bit numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model.
2. Define the data segment with required variables.
3. Assign code segment.
4. Initialize DS register with the starting address of data segment.
5. Clear the BX & AX registers.
6. Take the count value into CL register.
7. Copy the offset address of num list into SI register
8. Move the content of SI to BL register
9. Add the content of AX and BX register
10. Increment the content of SI by one.
11. Decrement the content of CL by one
12. Perform repeated addition till the count value becomes zero. (Repeat from step 7)
13. Terminate the program by using breakpoint interrupt.
14. End

PROGRAM:

```
        . MODEL SMALL
        . DATA
        COUNT DB 06h
        NUMLIST DB 10h,22h,33h,44h,55h,66h
        . CODE
        MOV AX, @DATA
        MOV DS, AX
        XOR BX, BX
        XOR AX, AX
        MOV CL, COUNT
        MOV SI, OFFSET NUMLIST
AGAIN:  MOV BL, [SI]
        ADD AX, BX
        INC SI
        DEC CL
        JNZ AGAIN
        INT 03H
        END
```

INPUT:

OUTPUT:

RESULT:

(B) Multi word Addition

AIM: To perform the addition of two 32-bit numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program
2. Assign Data segment
3. Assign Code segment
4. LSW of 1st double word is moved to AX register.
5. LSW of 2nd double word is moved to BX register.
6. Add the contents of AX and BX registers and the result is stored in AX register.
7. Copy the LSW of result present in AX register into CX register.
8. MSW of 1st double word is moved to AX register.
9. MSW of 2nd double word is moved to BX register.
10. Add the contents of AX and BX registers along with carry (obtained from previous addition) and the result is stored in AX register.
11. Copy the MSW of the result present in AX register into DX register.
12. Terminate the program by using breakpoint interrupt.
13. End.

PROGRAM:

```
. MODEL SMALL
. DATA
. CODE
MOV AX, 0F000h
MOV BX, 1000h
ADD AX, BX
MOV CX, AX
MOV AX, 5678h
MOV BX, 1234h
ADC AX, BX
MOV DX, AX
INT 03H
END
```

INPUT 1:

INPUT 2:

OUTPUT:

RESULT:

(C) Multi Word Subtraction

AIM: To perform the subtraction of two 32-bit numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program
2. Assign Data segment
3. Assign Code segment
4. LSW of 1st double word is moved to AX register.
5. LSW of 2nd double word is moved to BX register.
6. Subtract the contents of AX and BX registers and the result is stored in AX register.
7. Copy the LSW of result present in AX register into CX register.
8. MSW of 1st double word is moved to AX register.
9. MSW of 2nd double word is moved to BX register.
10. Subtract the contents of AX and BX registers along with borrow (obtained from Previous subtract) and the result is stored in AX register.
11. Copy the MSW of the result present in AX register into DX register.
12. Terminate the program by using breakpoint interrupt.
13. End.

PROGRAM:

```
. MODEL SMALL
. DATA
. CODE
MOV AX, 0111h
MOV BX, 1000h
SUB AX, BX
MOV CX, AX
MOV AX, 5678h
MOV BX, 1234h
SBB AX, BX
MOV DX, AX
INT 03h
END
```

INPUT 1:

INPUT 2:

OUTPUT:

RESULT:

Date: **Exp. No.- 3**

ALPs (8086) for Multiplication and Division

(i) Multiplication of two unsigned 16-bit numbers

AIM: To perform the multiplication of two unsigned 16-bit numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program
2. Assign Data segment
3. Assign Code segment
4. Move the first 16 bit unsigned number into AX register.
5. Move the second 16 bit unsigned into BX register.
6. Perform unsigned multiplication between the values stored in AX and BX, observe the MSW of the result present in DX and LSW in AX
7. Terminate the program by using breakpoint interrupt.
8. End

PROGRAM:

```
. MODEL SMALL
. DATA
. CODE
MOV AX, 1234h
MOV BX, 1234h
MUL BX
INT 03h
END
```

INPUT 1:

INPUT 2:

OUTPUT:

RESULT:

(ii) Multiplication of two signed 16-bit numbers

AIM: To perform the multiplication of two signed 16-bit numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program
2. Assign Data segment
3. Assign Code segment
4. Move the first 16 bit signed number into AX register.
5. Move the second 16 bit signed number into BX register.
6. Perform signed multiplication between the values stored in AX and BX, observe the MSW of the result present in DX and LSW in AX
7. Terminate the program by using breakpoint interrupt.
8. End

PROGRAM:

```
. MODEL SMALL
. DATA
. CODE
MOV AX, 8000h → Negative number
MOV BX, 2000h → Positive number
IMUL BX
INT 03h
END
```

INPUT 1:

INPUT 2:

OUTPUT:

RESULT:

(iii) Division of 16-bit/8-bit number (unsigned)

AIM: To perform division of 16-bit by 08 bit number.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program
2. Assign Data segment
3. Assign Code segment
4. Move the 16 bit dividend into AX register.
5. Move the 8 bit divisor into BL register.
6. Perform division between the values stored in AX and BL, observe the quotient in AL and remainder in AH.
7. Terminate the program by using breakpoint interrupt
8. End

PROGRAM:

```
. MODEL SMALL
. DATA
. CODE
MOV AX, 1234h
MOV BL, 58h
DIV BL
INT 03h
END
```

INPUT 1:

INPUT 2:

OUTPUT:

RESULT:

(iv) Division of 16-bit/8-bit number (signed)

AIM: To perform the 16 bit /8 bit division of two signed numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program
2. Assign Data segment
3. Assign Code segment
4. Move the 16 bit dividend into AX register.
5. Move the 8 bit divisor into BL register.
6. Perform division between the values stored in AX and BL, observe the quotient in AL and remainder in AH.
7. Terminate the program by using breakpoint interrupt
8. End

PROGRAM:

```
. MODEL SMALL
. DATA
. CODE
MOV AX, -0002h
MOV BL, 80h
IDIV BL
INT 03h
END
```

INPUT:

OUTPUT:

RESULT:

(v) Division of 32-bit/16-bit number (unsigned)

AIM: To perform the division of 32-bit by 16 bit number.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program
2. Assign Data segment
3. Assign Code segment
4. Move the LSW of dividend into AX register.
5. Move the MSW of dividend into DX register.
6. Move the 16 bit divisor into BX register.
7. Perform division between the values stored in DX: AX and BX, observe the quotient in AX and remainder in DX.
8. Terminate the program by using Break Point Interrupt.
9. End

PROGRAM:

```
. MODEL SMALL
. DATA
. CODE
MOV AX, 2786h (LSW of dividend)
MOV DX, 2334h (MSW of dividend)
MOV BX, 3552h (divisor)
DIV BX
INT 03h
END
```

INPUT 1:

INPUT 2:

OUTPUT:

RESULT:

Date: **Exp. No. – 4**

ALPs (8086) for GCD and LCM of two 16-bit numbers

(A) ALP (8086) to determine GCD of two 16-bit binary numbers.

AIM: To find out GCD of two 16-bit binary numbers

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program
2. Define the data segment with required variables.
3. Assign Code segment.
4. Initialize DS register with the starting address of data segment.
5. Take 1st number into AX register and 2nd number into BX register.
6. Compare the content of AX and BX, if equal AX content is the final result else perform the subtraction until AX and BX content becomes equal.
7. Copy the result in AX to variable name GCD in data segment.
8. Terminate the program by using Break Point Interrupt.
9. End.

PROGRAM:

```
. MODEL SMALL
. DATA
NUM1 DW 003Ch
NUM2 DW 000Fh
GCD DW 01h DUP (?)
. CODE
MOV AX, @DATA
MOV DS, AX
MOV AX, NUM1
MOV BX, NUM2
BACK: CMP AX, BX
JE RESULT
JNC AHEAD
SUB BX, AX
JMP BACK
AHEAD: SUB AX, BX
JMP BACK
RESULT: MOV GCD, AX
INT 03h
END
```

INPUT1:

INPUT2:

OUTPUT:

RESULT:

(B) ALP (8086) to determine LCM of two 16-bit numbers.

AIM: To find out LCM of two 16-bit numbers

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program.
2. Define the data segment with required variables.
3. Assign Code segment.
4. Initialize DS register with the starting address of data segment.
5. Take 1st number into CX register and 2nd number into DX register.
6. Copy the content of CX and DX into Ax and BX, if equal AX content is the final result.
7. If AX is less than BX then add content of BX and DX else add the content of AX and CX.
8. Repeat the step5 and step6 until content of AX and BX equal.
9. Copy the result in AX to variable name LCM in data segment.
10. Terminate the program by using Break Point Interrupt.
11. End.

PROGRAM:

```
        . MODEL SMALL
        . DATA
        NUM1 DW 003Ch
        NUM2 DW 000Fh
        LCM DW 01h DUP (?)
        . CODE
        MOV AX, @DATA
        MOV DS, AX
        MOV AX, NUM1
        MOV BX, NUM2
        MOV CX, AX
        MOV DX, BX
BACK:   CMP AX, BX
        JE RESULT
        JNC AHEAD
        ADD AX, CX
        JMP BACK
AHEAD:  ADD BX, DX
        JMP BACK
RESULT: MOV LCM, AX
        INT 03h
        END
```

INPUT1:

INPUT2:

OUTPUT:

RESULT:

Date: **Exp. No – 5**

ALP (8086) to evaluate arithmetic expressions

AIM: To evaluate a given arithmetic expression $f = \frac{(a+b)\,(b+c)\,(c+d)}{(a+b+c+d)}$

APPARATUS: System with TASM software.

ALGORITHM:

1. Assign Model to the Program.
2. Define the data segment with required variables.
3. Assign Code segment.
4. Initialize DS register with the starting address of data segment through AX register.
5. Copy the variable data a & b into AL & BL and perform the addition between AL & BL.
6. Initialize the starting address of result temporarily with SI register and store the result 1200h location from the AL register.
7. Copy the variable data b & c into AL & BL and perform the addition between AL & BL.
8. Increment the SI register and store the result 1201h location from the AL register.
9. Copy the variable data c & d into AL & BL and perform the addition between AL & BL.
10. Increment the SI register and store the result 1202h location from the AL register.
11. Copy the variable data a into AL register and perform the addition between AL registers & b.
12. Add the content of AL and c then d to the AL register.
13. Copy the result into BL register from AL register & move the content in 1200h location to AL.
14. Multiply the content in AL with content available in 1201h location then multiply with content in 1202h location perform the division.
15. Terminate the program by using Break Point Interrupt.
16. End.

PROGRAM:

```
. MODEL SMALL
. DATA
A DB 01H
B DB 02H
C DB 03H
D DB 04H
. CODE
MOV AX, @DATA
MOV DS, AX
XOR AX, AX
MOV AL, A
MOV BL, B
ADD AL, BL
MOV SI, 1200H
MOV [SI], AL
MOV AL, B
MOV BL, C
ADD AL, BL
INC SI
MOV [SI], AL
MOV AL, C
MOV BL, D
ADD AL, BL
INC SI
MOV [SI], AL
MOV AL, A
ADD AL, B
ADD AL, C
ADD AL, D
MOV CL, AL
XOR AX, AX
```

```
MOV AL, [SI]
MOV BL, [SI-1]
MUL BL
MOV BL, [SI-2]
MUL BL
MOV BL, CL
DIV BL
INT 03H
END
```

INPUT:

OUTPUT:

RESULT:

Date: **Exp. No – 6**

ALPs (8086) for sorting and searching

(A) Sorting a string in an ascending order

AIM: To sort the given string in an ascending order.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the Program.
2. Define the data segment with required variables.
3. Assign code segment.
4. Initialize DS register with the starting address of data segment.
5. Specify the count value for external loop in DL register.
6. Move the offset address of array into SI register.
7. Move the value of DL register into CL register.
8. Move the content of SI register into the AL register.
9. Compare the number in AL register with subsequent number and exchange the position of the numbers depending on result of the comparison.
10. Repeat step7 until the value present in CL register is Zero.
11. Repeat the steps 5, 6,7and 8 until the DL becomes Zero and observe the results in ascending order.
12. Terminate the program by using break Point Interrupt.
13. End.

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
 array db 66H,07H,10H,56H,23H
 count db 04H
. CODE
MOV AX, @data
MOV DS, AX
MOV DL, count
a4: MOV SI, offset array
MOV CL, DL
a3: MOV AL, [SI]
CMP AL, [SI+1]
JC a2
XCHG AL, [SI+1]
XCHG AL, [SI]
a2: INC SI
DEC CL
JNZ a3
DEC DL
JNZ a4
INT 03H
END
```

INPUT:

OUTPUT:

RESULT:

(B) Sorting a string in a descending order

AIM: To sort the given string in a descending order.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the Program.
2. Define the data segment with required variables.
3. Assign code segment.
4. Initialize DS register with the starting address of data segment.
5. Specify the count value for external loop in DL register.
6. Move the offset address of array into SI register.
7. Move the value of DL register into CL register.
8. Move the content of SI into the AL register.
9. Compare the number in AL register with subsequent number and exchange the position of the numbers depending on result of the comparison.
10. Repeat step7 until the value present in CL register is Zero.
11. Repeat the steps 5, 6,7and 8 until the DL becomes Zero and observe the results in ascending Order.
12. Terminate the program by using break Point Interrupt.
13. End.

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
 array db 66H,07H,10H,56H,23H
 count db 04H
. CODE
MOV AX, @data
MOV DS, AX
MOV DL, count
a4: MOV SI, offset array
MOV CL, DL
a3: MOV AL, [SI]
CMP AL, [SI+1]
JNC a2
XCHG AL, [SI+1]
XCHG AL, [SI]
a2: INC SI
DEC CL
JNZ a3
DEC DL
JNZ a4
INT 03H
END
```

INPUT:

OUTPUT:

RESULT:

(C) Smallest number of a given 'n' numbers

AIM: To find the smallest number in a given array.

APPARATUS: System with TASM software.

ALGORITHM:

1. Assign Model to the Program.
2. Define the data segment with required variables.
3. Assign Code segment.
4. Initialize DS register with the starting address of data segment.
5. Move the offset address of array into SI register.
6. Copy the count value into CL register.
7. Move the content of SI register into AL register.
8. Compare the number in AL register with subsequent number and copy the smallest number into AL register depending on result of the comparison.
9. Repeat step7 until the value present in CL register is Zero.
10. Terminate the program by using Break Point Interrupt.
11. End.

PROGRAM:

```
    . MODEL SMALL
    . STACK
    . DATA
    LIST DB 02h, 09h, 03h, 06h, 08h, 07h
    . CODE
    MOV AX, @DATA
    MOV DS, AX
    MOV SI, OFFSET LIST
    MOV CL, 05h
    MOV AL, [SI]
UP: INC SI
    CMP AL, [SI]
    JB GO
    MOV AL, [SI]
GO: DEC CL
    JNZ UP
    INT 03h
    END
```

INPUT:

OUTPUT:

RESULT:

(D) Largest number of a given 'n' numbers

AIM: To find the largest number in a given array.

APPARATUS: System with TASM software.

ALGORITHM:

1. Assign Model to the Program.
2. Define the data segment with required variables.
3. Assign Code segment.
4. Initialize DS register with the starting address of data segment.
5. Move the offset address of array into SI register.
6. Copy the count value into CL register.
7. Move the content of SI register into AL register.
8. Compare the number in AL register with subsequent number and copy the largest number into AL register depending on result of the comparison.
9. Repeat step7 until the value present in CL register is Zero.
10. Terminate the program by using Break Point Interrupt.
11. End.

PROGRAM:

```
    . MODEL SMALL
    . STACK
    . DATA
    LIST DB 02h, 09h, 03h, 06h, 08h, 07h
    . CODE
    MOV AX, @DATA
    MOV DS, AX
    MOV SI, OFFSET LIST
    MOV CL, 05h
    XOR AX, AX
    MOV AL, [SI]
UP: INC SI
    CMP AL, [SI]
    JNB GO
    MOV AL, [SI]
GO: DEC CL
    JNZ UP
    INT 03h
    END
```

INPUT:

OUTPUT:

RESULT:

(E) Search for a given number

AIM: To perform searching for a byte in an array.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the Program.
2. Define the data segment with required variables.
3. Assign Code segment.
4. Initialize DS and ES register with the starting address of data segment.
5. Copy the count value into CL register.
6. Get the offset address of array into DI register and clear the direction flag.
7. Copy the number to be search in AL register and scan repeatedly.
8. If not found display the message BYTE NOT FOUND.
9. If found display the message BYTE FOUND.
10. Terminate the program by using break point interrupt.
11. End

PROGRAM:

```
        . MODEL SMALL
        .DATA
        array db 15H,07H,25H,12H
        count db 04H
        stg db 'byte found','$'
        stg1 db 'byte not found','$'
        .CODE
START:  MOV AX, @data
        MOV DS, AX
        MOV ES, AX
        MOV DI, offset array
        MOV CL, count
        MOV AL, 76H
      REPNE SCASB
        JZ a2
        MOV AH, 09H
        LEA DX,stg1
        INT 21H
        JMP a3
    a2: MOV AH, 09H
        LEA DX, stg
        INT 21H
    a3: INT 03H
END
```

INPUT:

OUTPUT:

RESULT:

Date: **Exp. No – 7**

Logical Operations

(A) Shift Logical Right

AIM: To perform the shift right operation.

APPARATUS: System with TASM Software

ALGORITHM:

1. Assign model to the program.
2. Assign Code segment.
3. Move the data to be shifted into AL register.
3. Specify the number of shift operations to be performed in CL register.
4. Perform shift right operation and observe the result in AL register.
5. Terminate the program by using break point interrupt.
6. End

PROGRAM:

```
. MODEL SMALL
. CODE
MOV AL, 46H
MOV CL, 04H
SHR AL, CL
INT 03H
END
```

INPUT:

OUTPUT:

RESULT:

(B) Shift Logical Left

AIM: To perform the shift left operation.

APPARATUS: System with TASM Software

ALGORITHM:

1. Assign model to the program.
2. Assign Code segment.
3. Move the data to be shifted into AL register.
4. Specify the number of shift operations to be performed in CL register.
5. Perform shift left operation and observe the result in AL register.
6. Terminate the program by using break point interrupt.
7. End

PROGRAM:

```
. MODEL SMALL
. CODE
MOV AL, 46H
MOV CL, 04H
SHL AL, CL
INT 03H
END
```

INPUT:

OUTPUT:

RESULT:

(C) Rotate right without Carry

AIM: To perform the Rotate right without Carry.

APPARATUS: System with TASM Software

ALGORITHM:

1. Assign the model to the program.
2. Assign Code segment.
3. Move the data to be shifted into AL register.
4. Specify the number of rotate operations to be performed in CL register.
5. Perform rotate right operation without carry and observe the result in AL register.
6. Terminate the program by using break point interrupt.
7. End

PROGRAM:

```
. MODEL SMALL
. CODE
MOV AL, 68H
MOV CL, 04H
ROR AL, CL
INT 03H
END
```

INPUT:

OUTPUT:

RESULT:

(D) Rotate left without Carry

AIM: To perform the Rotate left without Carry.

APPARATUS: System with TASM Software

ALGORITHM:

1. Assign model to the program.
2. Assign code segment.
3. Move the data to be shifted into AL register.
3. Specify the number of rotate operations to be performed in CL register.
4. Perform rotate left operation without carry and observe the result in AL register.
5. Terminate the program by using Break point interrupt..
6. End.

PROGRAM:

```
. MODEL SMALL
. CODE
MOV AL, 60H
MOV CL, 04H
ROL AL, CL
INT 03H
END
```

INPUT:

OUTPUT:

RESULT:

(E) Rotate right with Carry

AIM: To perform the Rotate right with Carry.

APPARATUS: System with TASM Software

ALGORITHM:

1. Assign model to the program.
2. Assign Code segment.
3. Move the data to be shifted into AL register.
4. Specify the number of rotate operations to be performed in CL register.
5. Perform rotate right operation with carry and observe the result in AL register.
6. Terminate the program by using Break point interrupt..
7. End.

PROGRAM:

```
. MODEL SMALL
. CODE
MOV AL, 68H
MOV CL, 04H
RCR AL, CL
INT 03H
END
```

INPUT:

OUTPUT:

RESULT:

(F) Rotate left with Carry

AIM: To perform the Rotate left with Carry.

APPARATUS: System with TASM Software

ALGORITHM:

1. Assign Model to the program.
2. Assign Code segment.
3. Move the data to be shifted into AL register.
4. Specify the number of rotate operations to be performed in CL register.
5. Perform rotate left operation with carry and observe the result in AL register.
6. Terminate the program by using Break point interrupt..
7. End.

PROGRAM:

```
. MODEL SMALL
. CODE
MOV AL, 68h
MOV CL, 04h
RCL AL, CL
INT 03H
END
```

INPUT:

OUTPUT:

RESULT:

(G) Converting packed BCD to Unpacked BCD

AIM: To write an ALP to convert packed BCD number to Unpacked BCD number.

APPARATUS: System with TASM Software

ALGORITHM:

1. Assign Model to the program.
2. Assign Code segment.
3. Move the packed data into BL register and count value into CL register.
4. Copy the packed BCD data from BL to AL register.
5. Perform shift left & rotate right operations by the AL with the count specified by CL reg.
6. Copy the lower byte of unpacked BCD data to DL register.
7. Copy the packed BCD data from BL to AL register.
8. Perform shift right operations by the AL with the count specified by CL register.
9. Copy the higher byte of unpacked BCD data to DH register.
10. Terminate the program by using Break point Interrupt.
11. End.

PROGRAM:

```
. MODEL SMALL
. CODE
MOV BL, 57H
MOV CL, 04H
MOV AL, BL
SHL AL, CL
ROR AL, CL
MOV DL, AL
MOV AL, BL
SHR AL, CL
MOV DH, AL
INT 03H
END
```

INPUT: BL=57H **OUTPUT:** DX=0507H

RESULT:

(H) Converting BCD to ASCII number

AIM: To write an ALP to convert BCD number to ASCII number.

APPARATUS: System with TASM Software

ALGORITHM:

1. Assign Model to the program.
2. Assign Code segment.
3. Move the packed data into BL register and count value into CL register.
4. Copy the packed BCD data from BL to AL register.
5. Perform shift left & rotate right operations by the AL with the count specified by CL register.
6. Copy the lower byte of unpacked BCD data to DL register.
7. Copy the packed BCD data from BL to AL register.
8. Perform shift right operations by the AL with number of times specified by CL register.
9. Copy the higher byte of unpacked BCD data to DH register.
10. Perform XOR operation between DX register and 3030h
11. Terminate the program by using Break point interrupt.
12. End

PROGRAM:

```
. MODEL SMALL
. CODE
MOV BL, 57H
MOV CL, 04H
MOV AL, BL
SHL AL, CL
ROR AL, CL
MOV DL, AL
MOV AL, BL
SHR AL, CL
MOV DH, AL
XOR DX, 3030h
INT 03H
END
```

INPUT: BL=57H

OUTPUT: DX=3537H

RESULT:

Date: **Exp. No –8**

ALPs (8086) for String operations

(A) Move block from one segment to another segment

AIM: To write an ALP to move the block of data from one segment to another segment.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program.
2. Define the data segment with required variables.
3. Assign Code segment
4. Initialize DS register with the starting address of data segment.
5. Initialize ES register with the starting address of data segment.
6. Copy the number of bytes in the string to CL register.
7. Store the offset addresses of string and string1 into SI and DI registers respectively.
8. Clear the direction flow flag to select the auto increment of offset address.
9. Move the string bytes from source to destination until count becomes zero.
10. Terminate the program by using breakpoint interrupt
11. End

PROGRAM:

```
. MODEL SMALL
. DATA
STRING   DB   'SVCET','$'
STRING1   DB   05H   DUP (?),'$'
COUNT DB 05H
. CODE
MOV AX, @DATA
MOV DS, AX
MOV ES, AX
MOV CL, COUNT
MOV SI, OFFSET STRING
MOV DI, OFFSET STRING1
CLD
REP MOVSB
MOV AH, 09H
LEA DX, STRING1
INT 21H
INT 03H
END
```

INPUT:

OUTPUT:

RESULT:

(B) Reverse String

AIM: To perform the reverse of a string.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program.
2. Define the data segment with required variables.
3. Assign Code segment.
4. Initialize DS register with the starting address of data segment.
5. Initialize ES register with the starting address of data segment.
6. Copy the number of bytes in the string to CL register.
7. Copy the offset address of STG into SI register and offset address of STG1 into DI register.
8. Clear direction flag for auto increment of offset addresses.
9. Add the SI register with n-1 count value.
10. Copy the string characters in reverse order from source segment to destination segment.
11. Repeat step8 until the count becomes zero in CL register.
12. Terminate the program by using Breakpoint interrupt.
13. End.

PROGRAM:

```
        .MODEL SMALL
        .DATA
        STG DB 'SVCET','$'
        STG1 DB 05H DUP (?), '$,
        COUNT DB 05H
        .CODE
        MOV AX, @DATA
        MOV DS, AX
        MOV ES, AX
        MOV CL, COUNT
        MOV SI, OFFSET STG
        MOV DI, OFFSET STG1
        CLD
        ADD SI, 04h
    A1: MOVSB
        DEC SI
        DEC SI
        DEC CL
        JNZ A1
        MOV AH, 09H
        LEA DX, STG1
        INT 21H
        INT 03H
        END
```

INPUT:

OUTPUT:

RESULT:

(C) String Comparison:

AIM: To perform the comparison of strings.

APPARATUS: system with TASM software.

ALGORITHM:

1. Assign Model to the program.
2. Define the data segment with required variables.
3. Assign Code segment.
4. Initialize DS register with the starting address of data segment.
5. Initialize ES register with the starting address of data segment.
6. Copy the number of bytes in the string to CL register.
7. Store the offset address of STG1 & STG2 into SI and DI registers respectively.
8. Select the auto increment of offset addresses using direction flow flag.
9. Compare the string bytes of STG1 with STG2 until all the bytes are equal.(if not it won't repeat)
10. Display the result based on comparison.
11. Terminate the program by using breakpoint interrupt.
12. End.

PROGRAM:

```
. MODEL SMALL
. DATA
STRG1   DB   'SVCET','$'
STRG2   DB   'SVPET', $'
RES   DB 'STRGS ARE EQUAL','$'
RES1 DB 'STRGS ARE NOT EQUAL','$'
COUNT EQU 03H
. CODE
MOV AX, @DATA
MOV DS, AX
MOV ES, AX
MOV CL, COUNT
LEA SI, STRG1
LEA DI, STRG2
CLD
REPE CMPSB
JNZ A2
MOV AH, 09H
LEA DX, RES
INT 21H
JMP A1
A2: MOV AH, 09H
LEA DX, RES1
INT 21H
A1: INT 03H
END
```

INPUT1:

INPUT2:

OUTPUT:

RESULT:

(D) Length of string

AIM: To write an ALP to find out the length of a given string.

APPARATUS: System with TASM software.

ALGORITHM:

1. Assign Model to the program.
2. Define the data segment with required variables.
3. Assign Code segment.
4. Initialize DS register with the starting address of data segment.
5. Move the value 0AH into AL register. (For accepting a characters from the keyboard and attached to a user designated buffer).
6. Load effective address of label byte string in DX register.
7. Use DOS interrupt.(and also DOS command P)
8. Move the value 02H into AL register. (To display a character)
9. Move the length attached to the variable A2 into DL register.
10. Convert the data in DL register to ASCII value by using OR function with 30H.
11. Use DOS interrupt to calculate the length of the given string and display.(and also DOS command P)
12. End.

PROGRAM:

```
. MODEL SMALL
. DATA
S1 LABEL BYTE
LX1 DB 30H
A2 DB?
. CODE
MOV AX, @DATA
MOV DS, AX
MOV AH, 0AH
LEA DX, S1
INT 21H
MOV AH, 02H
MOV DL, A2
OR DL, 30H
INT 21H
END
```

INPUT:

OUTPUT:

RESULT:

PART-B

(Minimum 02 experiments to be conducted)

Date: **Exp. No. --1**

(A) ALP (8086) for generating ramp waveform using DAC

AIM: To write an ALP to generate a ramp wave form using DAC.

APPARATUS: 1. 8086 Microprocessor kit.

2. DAC kit.

ALGORITHM:

1. Initialize the 8255 control port with 80h.
3. Copy the initial value as 00h for a ramp wave in AL register.
4. Send 00h through port A to DAC for digital to analog conversion.
5. Increment the content of AL by 1 up to FFh,
6. If the value reaches FFh then again start from 00h.
7. Repeat step 4, 5 & 6 to produce continuous ramp wave.

Port A Address: 70H

Port B Address: 72H

Port C Address: 74H

Control Port Address: 76H

Control Word Format of 8255

PROGRAM LOGIC:

```
        ORG 0400H
        MOV AL, 80H
        OUT 76H, AL
START:  MOV AL, 00H
BACK:   OUT 70H, AL
        INC AL
        CMP AL, FFH
        JBE BACK
        JMP START
```

Memory Address	Op-Code	Label	Mnemonics	operands	Comments
0400	B0		MOV	AL,80H	
0401	80				
0402	E6		OUT	76H,AL	
0403	76				
0404	B0	START	MOV	AL,00H	
0405	00				
0406	E6	BACK	OUT	70H,AL	
0407	70				
0408	FE		INC	AL	
0409	C0				
040A	3C		CMP	AL,FFH	
040B	FF				
040C	76		JBE	BACK (0406)	
040D	F8				
040E	EB		JMP	START (0404)	
040F	F4				

RESULT:

(B) ALP (8086) for generating Triangular waveform using DAC

AIM: To write an ALP to generate a triangular waveform using DAC.

APPARATUS: 1. 8086 Microprocessor kit.

2. DAC kit.

ALGORITHM:

1. Initialize the 8255 control port with 80h.
3. Copy the initial value as 00h for a ramp wave in AL register.
4. Send 00h through port A to DAC for digital to analog conversion.
5. Increment the content of AL by 1 up to FFh,
6. If the value reaches FFh then decrement the content of AL by 1 until reaches to 00h.
7. Repeat step 4,5 & 6 to produce continuous triangular wave.

Port A Address: 70H

Port B Address: 72H

Port C Address: 74H

Control Port Address: 76H

Control Word Format of 8255

PROGRAM LOGIC:

```
        ORG 0400H
        MOV AL, 80H
        OUT 76H, AL
        MOV AL, 00H
BACK1:  OUT 70H, AL
        INC AL
        CMP AL, FFH
        JB BACK1
BACK2:  OUT 70H, AL
        DEC AL
        CMP AL, 00H
        JA BACK2
        JMP BACK1
```

Memory Address	Op-Code	Label	Mnemonics	Operands	Comments
0400	B0		MOV	AL,80H	
0401	80				
0402	E6		OUT	76H,AL	
0403	76				
0404	B0	START	MOV	AL,00H	
0405	00				
0406	E6	BACK1	OUT	70H,AL	
0407	70				
0408	FE		INC	AL	
0409	C0				
040A	3C		CMP	AL,FFH	
040B	FF				
040C	72		JB	BACK (0406)	
040D	F8				
040E	E6	BACK2	OUT	70H,AL	
040F	70				
0410	FE		DEC	AL	
0411	C8				
0412	3C		CMP	AL,00H	
0413	00				
0414	77		JA	BACK2 (040E)	
0415	F8				
0416	EB		JMP	BACK1 (0406)	
0417	EE				

RESULT:

(C) ALP (8086) for generating stair case wave form using DAC

AIM: To write an ALP to generate a stair case waveform using DAC.

APPARATUS: 1. 8086 Microprocessor kit.
2. DAC kit.

ALGORITHM:

1. Initialize the 8255 control port with 80h.
3. Copy the initial value as 00h for a staircase wave in AL register.
4. Send 00h through port A to DAC for digital to analog conversion and wait for some delay.
5. Add the content of AL by 33h up to FFh,
6. If the value reaches FFh then wait for some delay and subtract the content of AL by 33h until reaches to 00h.
7. Repeat step 4, 5 & 6 to produce continuous triangular wave.

Port A Address: 70H

Port B Address: 72H

Port C Address: 74H

Control Port Address: 76H

Control Word Format of 8255

PROGRAM:

```
                    ORG 0400H
                    MOV AL, 80H
                    OUT 76H, AL
                    MOV AL, 00H
(0406) BACK1:       OUT 70H, AL
                    CALL DELAY (041E)
                    ADD AL, 33H
                    CMP AL, FFH
                    JB BACK1 (0406)
(0411) BACK2:       OUT 70H, AL
                    CALL DELAY (041E)
                    SUB AL, 33H
                    CMP AL, 00H
                    JA BACK2 (0411)
                    JMP BACK1 (0406)
 (041E)  DELAY:     MOV BX, 0111H
 (0421) BACK3:      DEC BX
                    JNZ BACK3 (0421)
                    RET
```

RESULT:

Date: **Exp. No – 2**

ALP (8086) for traffic light controller

AIM: To write an ALP for four road cross junction to control the traffic automatically.

APPARATUS: 1. 8086 Microprocessor kit.
2. Traffic Light Controller board.

ALGORITHM:

1. Initialize the 8255 control port with 80h.
3. Copy the content as 11h into AL and send through port C to stop east & west vehicles with Red light ON.
4. Copy the content as 44h into AL and send through port A to allow north & south vehicles with Green light ON with particular delay.
5. Copy the content as 22h into AL and send through port A & port C to allow all side vehicles with yellow light ON with particular delay.
6. Copy the content as 11h into AL and send through port A to stop north & south vehicles with red light ON.
7. Copy the content as 44h into AL and send through port C to allow east & west vehicles with Green light ON with particular delay.
8. Copy the content as 22h into AL and send through port A & port C to allow all side vehicles with yellow light ON with particular delay.
9. Repeat from step3 to 8 for traffic light controller to perform continuously.

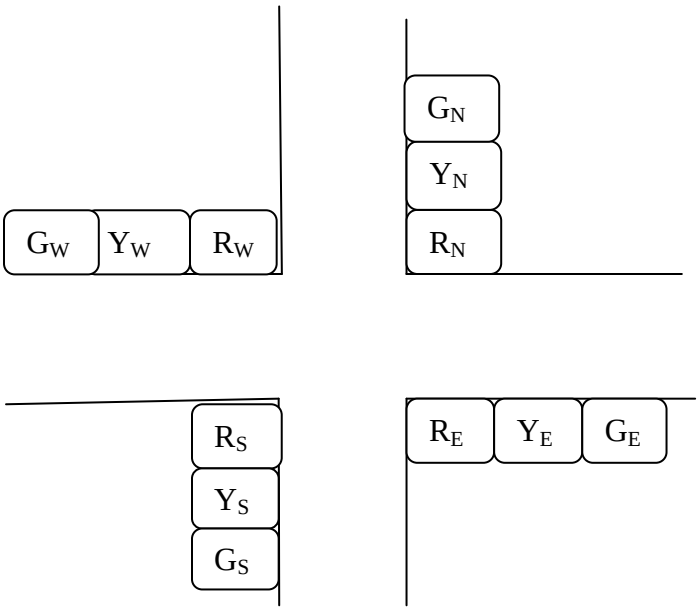
Port A Address: 70H

Port B Address: 72H

Port C Address: 74H

Control Port Address: 76H

Control Word Format of 8255



PA_0= R_N ; PA_1=Y_N; PA2=G_N ;PA_4=R_S ;PA_5=Y_S ;PA_6=G_S

PC_0= R_E ; PC_1=Y_E ;PC_2=G_E ;PC_4=R_W ;PC_5=Y_W ;PC_6=G_W

Consider PA_3 & PC_3 as 0

PROGRAM:

```
                ORG 0400H
                MOV AL, 80H
                OUT 76H, AL
BACK (0404):  MOV AL, 11H          ; RED ONLY
                OUT 74H, AL        ; FOR E-W
                MOV AL, 44H        ; GREEN ON
                OUT 70H, AL        ; FOR N-S
                CALL DELAY1 (042E) ; DELAY
                MOV AL, 22H        ; YELLOW ON
                OUT 70H, AL        ; ALL
                OUT 74H, AL        ; SIDES
                CALL DELAY2 (043B) ; DELAY
                MOV AL, 11H        ; RED ON
                OUT 70H, AL        ; FOR N-S
                MOV AL, 44H        ; GREEN ON
                OUT 74H, AL        ; FOR E-W
                CALL DELAY1 (042E)
```

```
                MOV AL, 22H
                OUT 70H, AL
                OUT 74H, AL
                CALL DELAY2 (042B)
                JMP BACK (0404)
DELAY1 (042E):  MOV BX, 000FH
 BACK2 (0431):  MOV CX, FFFFH
 BACK1 (0434):  DEC CX
                JNZ BACK1       (0434)
                DEC BX
                JNZ BACK2 (0431)
                RET
DELAY2 (043B):  MOV BX, 0005H
BACK4 (043E):    MOV CX, FFFFH
BACK3 (0441):    DEC CX
                JNZ BACK3 (0441)
                DEC BX
                JNZ BACK4 (043E)
                RET
```

RESULT:

Date: **Exp. No – 3**

ALP (8086) for stepper motor control

AIM: To write an ALP to run the stepper motor in anti-clockwise direction continuously.

APPARATUS: 1. 8086 Microprocessor kit.

2. Stepper Motor control board.

ALGORITHM:

1. Initialize the 8255 control port with 80h.
2. Copy 0Ah into AL register and send to port A then call delay program.
3. Copy 06h into AL register and send to port A then call delay program.
4. Copy 05h into AL register and send to port A then call delay program.
5. Copy 09h into AL register and send to port A then call delay program.
7. Repeat from step3 to step8 for stepper motor to rotate anti-clockwise direction continuously.

Port A Address: 70H

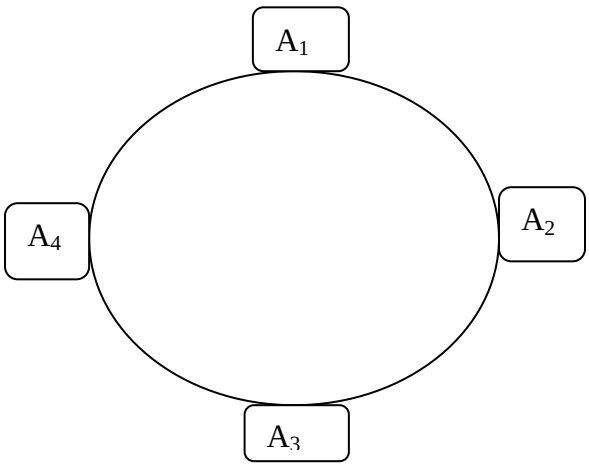
Port B Address: 72H

Port C Address: 74H

Control Port Address: 76H

Control Word Format of 8255

Consider PA_4 to PA_7 as 0



PA_7	PA_6	PA_5	PA_4	PA_3	PA_2	PA_1	PA_0	Data output to Port A to run the stepper motor in anti-clock wise direction
0	0	0	0	A_1	A_3	A_2	A_4	
0	0	0	0	1	0	1	0	0AH
0	0	0	0	0	1	1	0	06H
0	0	0	0	0	1	0	1	05H
0	0	0	0	1	0	0	1	09H

PROGRAM:

```
0400: MOV AL, 80H
      OUT 76, AL
0404: MOV AL, 0AH
      OUT 70, AL
      CALL 0423H
      MOV AL, 06H
      OUT 70, AL
      CALL 0423H
      MOV AL, 05H
      OUT 70, AL
      CALL 0423H
      MOV AL, 09H
      OUT 70, AL
      CALL 0423H
      JMP 0404H
0423: MOV CX, 0A0AH
0426: DEC CX
      JNZ 0426
      RET
```

INPUT: 0A, 06, 05, 09

OUTPUT: Rotates in anti clock wise direction

RESULT:

PART-C

(Minimum 03 Experiments to be conducted)

8051 Microcontroller:

Date: **Exp. No – 01**

a) ALP (8051) to determine the addition of N bytes

AIM: To write an ALP for 8051 to determine the addition of N bytes

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Access the memory location 4000h through DPTR.
2. Copy the number of bytes to be added into R0 register.
3. Move the number zero into R1 register
4. Copy the content of DPTR into A Register.
5. Add the content of A register and R1 register with carry and result is stored in R1 register
6. Increment DPTR content
7. Decrement the register R0 by one if not zero control transfer to step 4
8. Access the memory location 4200h through DPTR
9. Store the result from A register into memory location 4200
9. Terminate the program by using LCALL 0003

PROGRAM:

```
        3000: MOV DPTR, #4000
              MOV R0, #05
              MOV R1,#00
    ADDRESS: MOV X A, @DPTR
              ADDC R1, A
              INC DPTR
              DJNZ R0, ADDRESS
              MOV DPTR, #4200
              MOVX @DPTR, A
              LCALL 0003
INPUT:  4000:01
        4001:02
        4002:03
        4003:04
        4004:01
OUTPUT: 4200:0B
```

Date: **Exp. No – 01**

b) ALP (8051) to determine the addition of N bytes

AIM: To write an ALP for 8051 to determine the subtraction of N bytes

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Access the memory location 4000h through DPTR.
2. Copy the number of bytes to be added into R0 register.
3. Move the number zero into R1 register
4. Copy the content of DPTR into A Register.
5. Subtraction the content of A register and R1 register with borrow and result is stored in R1 Register.
6. Increment DPTR content
7. Decrement the register R0 by one if not zero control transfer to step 4
8. Access the memory location 4200h through DPTR
9. Store the result from A register into memory location 4200
9. Terminate the program by using LCALL 0003

PROGRAM:

```
        3000: MOV DPTR, #4000
              MOV R0, #05
              MOV R1,#00
    ADDRESS: MOV X A, @DPTR
              SUBB R1, A
              INC DPTR
              DJNZ R0, ADDRESS
              MOV DPTR, #4200
              MOVX @DPTR, A
              LCALL 0003
INPUT:  4000:01
        4001:02
        4002:03
        4003:04
        4004:01
OUTPUT: 4200:
```

Date: **Exp. No – 02**

A) ALP (8051) to determine the largest of N bytes

AIM: To write an ALP for 8051 to determine the largest of n-bytes.

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Access the memory location 4000h through DPTR and copy the content into A register.
2. Copy the number bytes in an array into R0 register.
3. Copy the content of A register into F0 Register.
4. Increment DPTR to access next memory location.
5. Decrement the count value in R0 and increment DPTR to copy the next byte into A register.
6. Compare the content of A and F0 register if not equal control transfer to new location.
7. Short jump to another location.
8. If carry is set then increment the DPTR. If not copy the content of A to F0 register.
9. Repeat from step 5 until R0 becomes zero.
10. Copy the largest value into A register from F0.
11. Copy the output memory location into DPTR 4200.
12. Copy the largest value from A to memory location.
13. Terminate the program by using LCALL 0003.

PROGRAM:

```
3000: MOV DPTR, #4000
      MOV R0, #05
      MOVX A, @DPTR
      MOV 0F0, A
      INC DPTR
      DEC R0
300A: MOVX A, @DPTR
      CJNE A, 0F0, 3010
      SJMP 3014
      JC 3014
      MOV 0F0, A
3014: INC DPTR
      DJNZ R0, 300A
      MOV A, 0F0
      MOV DPTR, #4200
      MOV @DPTR, A
      LCALL 0003
```

INPUT: 4000: 05

4001: 02

4002: 07

4003: 04

4004: 01

OUTPUT: 4200: 07

RESULT:

Date: **Exp. No – 02**

B) ALP (8051) to determine the smallest of N bytes

AIM: To write an ALP for 8051 to determine the smallest of n-bytes.

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Access the initial memory location 4000h through DPTR and copy the content into A register.
2. Copy the number bytes in an array into R_0 register.
3. Copy the content of A register into F0 Register.
4. Increment DPTR to access next memory location.
5. Decrement the count value in R0 and increment DPTR to copy the next byte into A register.
6. Compare the content of A and F0 register if not equal control transfer to new location.
7. Short jump to another location.
8. If carry is not set then increment the DPTR. If set copy the content of A to F0 register.
9. Repeat from step 5 until R0 becomes zero.
10. Copy the smallest value into A register from F0.
11. Copy the output memory location into DPTR 4200.
12. Copy the smallest value from A to memory location.
13. Terminate the program by using LCALL 0003.

PROGRAM:

```
3000: MOV DPTR, #4000
      MOV R0, #05
      MOVX A, @DPTR
      MOV 0F0, A
      INC DPTR
      DEC R0
300A: MOVX A, @DPTR
      CJNE A, 0F0, 3010
      SJMP 3014
      JNC 3014
      MOV 0F0, A
3014: INC DPTR
      DJNZ R0, 300A
      MOV A, 0F0
      MOV DPTR, #4200
      MOVX @DPTR, A
      LCALL 0003
```

INPUT: 4000: 05

4001: 02

4002: 07

4003: 04

4004: 01

OUTPUT: 4200: 01

RESULT:

Date: **Exp. No – 3**

(A) ALP (8051) to multiply a 16-bit number by an 8-bit number

AIM: To write an ALP (8051) to multiply a 16-bit number by an 8-bit number.

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Start
2. Access the memory location 3500h through DPTR and copy the LSB of 16-bit number content into A register.
3. Copy the least significant byte in multiplicand from A into register R2.
4. Increment DPTR to access next memory location and copy the MSB into R1 through A register.
5. Increment DPTR and copy the 8-bit multiplier in R3 through A register.
6. Initialize the DPTR to store the product of multiplication.
7. Copy the content in R2 to B and perform multiplication between A and B.
7. Store the LSB of product in 4000h and increment DPTR.
8. Copy the content in B to R4 temporarily and R1 into B and R3 into A.
9. Perform the multiplication between A and B then add content of A and R4.
10. Store the next byte of product in 4001 from A and increment DPTR.
11. Copy the content in B to A and perform the addition between 00h and A.
12. Store the last byte of the product in 4002h
13. Terminate the program

PROGRAM:

```
3000: MOV DPTR, #3500
      MOVX A, @ DPTR
      MOV R2, A
      INC DPTR
      MOVX A, @ DPTR
      MOV R1, A
      INC DPTR
      MOVX A, @ DPTR
      MOV R3, A
      MOV DPTR, # 4000
      MOV B, R2
      MUL AB
      MOVX @ DPTR, A
      INC DPTR
      MOV R4, B
      MOV B, R1
      MOV A, R3
      MUL AB
      ADD A, R4
      MOVX @ DPTR, A
      INC DPTR
      MOV A, B
      ADDC A, # 00
      MOVX @ DPTR, A
      LJMP 16A5
```

INPUT: 3500=03; 3501=00; 3502=02 (0003x02)

OUTPUT: 4000=06; 4001=00; 4002=00 (000006)

RESULT:

(B) ALP (8051) to find square root of an 8-bit number

AIM: To write an ALP (8051) to find square root of an 8-bit number.

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Start
2. Access the memory location 3500h through DPTR and copy the 8-bit number into A register.
3. Copy the content A into internal memory location 45H temporarily.
4. Assume initial square root value is 00h and store it in R1 register.
5. Copy the content of R1 to A and A to B then perform the multiplication between A and B.
6. Compare content of A and internal memory location 45H, if equal content of R1 is a square root Number and copy into 4000h location through A register.
7. If not equal increment R1 and repeat step6.
8. Terminate the program
9. Stop

PROGRAM:

```
3000: MOV DPTR, #3500
      MOVX A, @ DPTR
      MOV 45H, A
      MOV R1, #00
3008: MOV A, R1
      MOV B, A
      MUL AB
      CJNE A, 45, 3017
      MOV DPTR, #4000
      MOV A, R1
      MOVX @DPTR, A
      LJMP 16A5
3017: INC R1
      SJMP 3008
```

INPUT: 3500=09

OUTPUT: 4000=03

RESULT:

Date: **Exp. No – 4**

(A) ALP (8051) to determine LCM of two 8- bit number

AIM: To write an ALP (8051) to determine LCM of two 8- bit number.

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Start
2. Access the memory location 3500h through DPTR & copy the one 8-bit number into A register.
3. Copy the content of A intoR1 and increment DPTR.
4. Copy the content of DPTR memory location 3501h into R2 through A register.
5. Copy the content of R1 to internal memory location 10h and R2 into 20h.
6. Copy data in 10h location to A and compare with data in 20h, if equal content in A is the LCM and move to 4000h location .
7. If not equal, clear the carry flag and subtract the data in 20h location from A then verify the carry flag.
8. If carry not generated , copy the data in 20h to A and add the content of A and R2 then store the result in 10h from A and perform step6.
9. If carry generated, copy the data in 10h to A and add the content of A and R1 then store the result in 10h location from A and perform step6.
10. Terminate the program
11. Stop

PROGRAM:

```
        ORG 3000
        MOV DPTR, #3500
        MOVX A,@DPTR
        MOV R1, A
        INC DPTR
        MOVX A, @DPTR
        MOV R2, A
        MOV 10, R1
        MOV 20, R2
(300C) BACK:  MOV A, 10
        CJNE A, 20, AHEAD (3013)
        SJMP RESULT (3026)
(3013) AHEAD:  CLR C
        SUBB A, 20
        JC FORWARD (301F)
        MOV A, 20
        ADD A, R2
        MOV 20, A
        SJMP BACK (300C)
(301F) FORWARD:  MOV A, 10
        ADD A, R1
        MOV 10, A
        SJMP BACK (300C)
(3026) RESULT:  MOV DPTR, #3600
        MOVX @DPTR, A
        LJMP 16A5
```

INPUT1: 3500=02

INPUT2: 3501=04

OUTPUT: 3600=04

RESULT:

(B) ALP (8051) to determine GCD of two 8- bit numbers

AIM: To write an ALP (8051) to determine GCD of two 8- bit numbers.

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Start
2. Access the memory location 3500h through DPTR & copy the one 8-bit number into A register.
3. Copy the content of A into 10h internal memory location and increment DPTR.
4. Copy the content of memory location 3501h into 20h internal memory location through A.
5. Copy data in 10h location to A and compare with data in 20h, if equal content in A is the GCD and move to 4000h location .
6. If not equal, clear the carry flag and subtract the data in 20h location from A then verify the carry flag.
7. If carry not generated, copy the data in A to 10h internal memory location and perform step6.
8. If carry generated, clear carry flag then subtract content of 10h location from A and store the result in 20h location through A and perform step6.
9. Terminate the program

PROGRAM:

```
            ORG 3000H
            MOV DPTR, #3500H
            MOVX A, @DPTR
            MOV 10H, A
            INC DPTR
            MOVX A, @DPTR
            MOV 20H, A
(300A) BACK:  MOV A, 10H
            CJNE A, 20H, AHEAD
            SJMP RESULT (3023)
(3011)  AHEAD:  CLR C
            SUBB A, 20H
            JNC AHEAD1 (301F)
            MOV A, 20H
            CLR C
            SUBB A, 10H
            MOV 20H, A
            SJMP BACK (300A)
(301F) AHEAD1:  MOV 10H, A
            SJMP BACK (300A)
(3023) RESULT:  MOV DPTR, #3600
            MOVX @DPTR, A
            LJMP 16A5H
```

INPUT1: 3500=3CH (60)

INPUT2: 3501= 24H (36)

OUTPUT: 3600= 0CH (12)

RESULT:

Date: **Exp. No – 5**

(A) ALP (8051) to generate even numbers

AIM: To write an ALP (8051) to generate even numbers.

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Start
2. Access the memory location 3500h through DPTR.
3. Copy zero value into R0 register.
4. Copy R0 content into A register.
5. Copy the Value of A register into external memory through DPTR.
6. Increment DPTR.
7. Increment the value of R0 twice.
8. Compare the value of R0 with 08 ;if not equal repeat from step 4.
9. Terminate the program by using LCALL 0003

PROGRAM:

```
3000:  MOV DPTR, #3500
       MOV R0, #00
BACK:  MOV A, R0
       MOVX @DPTR, A
       INC DPTR
       INC R0
       INC R0
       CJNE R0, #08, BACK
       LCALL 0003
```

INPUT: R0=00

OUTPUT: 3500= 00; 3501=02; 3502= 04; 3502 =06

RESULT:

Date: **Exp. No – 5**

(B) ALP (8051) to generate odd numbers

AIM: To write an ALP (8051) to generate odd numbers.

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Start
2. Access the memory location 3500h through DPTR.
3. Copy 01 value into R0 register.
4. Copy R0 content into A register.
5. Copy the Value of A register into external memory through DPTR.
6. Increment DPTR.
7. Increment the value of R0 twice.
8. Compare the value of R0 with 09; if not equal repeat from step 4.
9. Terminate the program by using LCALL 0003

PROGRAM:

```
3000:  MOV DPTR, #3500
       MOV R0, #01
BACK:  MOV A, R0
       MOVX @DPTR, A
       INC DPTR
       INC R0
       INC R0
       CJNE R0, #09, BACK
       LCALL 0003
```

INPUT: R0=01

OUTPUT: 3500= 01; 3501=03; 3502= 05; 3502 =07

RESULT:

Date: **Exp. No – 6**

Timer/Counters in different modes

A) Timer in Mode -0 operation

AIM: To generate a square wave of frequency 1KHz at P1.0 of 8051 using timer 1 in mode 0..

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Program timer 1 in mode 0 by copy the value 00 into TMOD register
2. Clear both timer run bit as well as timer over flow bit.
3. Load 13-bit count value in timer 1 register i.e f1 into TH1 & 13 into TL1
4. Set the timer 1 run bit.
5. Check whether the timer 1 over flow is occurred or not .
6. Send to P1.0 and repeat from step 2

PROGRAM:

```
        MOV TMOD, #00H; Program timer 1 in mode 0
4003:   CLR TR1
        CLR TF1
        MOV TH1, #0F1H; load 13-bit count N in timer 1 register
        MOV TL1, #13H
        SETB TR1; run timer 1
400F:   MOV C, TF1
        JNC 400F
        MOV C, P1.0
        CPL C
        MOV P1.0, C
        SJMP 4003
```

RESULT:

Date: **Exp. No – 6**

Timer/Counters in different modes

Timer in Mode -1 operation

AIM: To generate a rectangular wave of frequency 1KHz with a duty cycle of 0.25 at P1.6 of 8051 using timer 0 in mode 1.

PROGRAM:

```
        MOV TMOD, #01H; Program timer 0 in mode 1
NEXT:   SETB P1.6
        CLR TR0
        CLR TF0
        MOV TH0, #0FFH; load count N  for timer 0 register
        MOV TL0, #1AH
        SETB TR0; run timer 0
BACK1:  MOV C, TF0
        JNC BACK1
        CLR P1.6
        CLR TR0
        CLR TF0
        MOV TH0,#0FDH; Count N2
        MOV TL0,#4DH;for T2
        SETB TR0
BACK2:  MOV C, TF0
        JNC BACK2
        SJMP NEXT
```

RESULT:

Date: **Exp. No – 6**

Timer/Counters in different modes

B) Timer in Mode -2 operation

AIM: To generate a square wave of frequency 2KHz at P1.3 of 8051 using timer 1(TH0) in mode2.

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Program timer 1 in mode 2 by copy the value 20H into TMOD register
2. Clear timer run bit .
3. Load 8-bit count value in timer 1 register i.e 1A into TH1 & 1A into TL1
4. Clear timer 1 overflow bit and Set the timer 1 run bit.
5. Check whether the timer 1 over flow is occurred or not .
6. Send to P1.3 and repeat from step 4

PROGRAM:

```
        MOV TMOD, #20H; Program timer 1 in mode 2
        CLR TR1
        MOV TH1,#1AH; load 8-bit count value in TH1
        MOV TL1,#1AH; load 8-bit count value in TL1
BACK1:  CLR TF1
        SETB TR1
BACK2:  MOV C, TF1
        JNC BACK2
        MOV C, P1.3
        CPL C
        MOV P1.3, C
        SJMP BACK1
```

RESULT:

Date: **Exp. No –6**

Timer/Counters in different modes

Timer in Mode -3 operation

AIM: To generate a square wave of frequency 5KHz at P1.5 of 8051 using timer 0(TH0) in mode3.

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Program timer 0 in mode 3 by copy the value 03H into TMOD register
2. Clear both timer 1 run bit and timer 1 over flow bit.
3. Load 8-bit count value in timer 0 register i.e A4 into TH0
4. Set the timer 1 run bit.
5. Check whether the timer 1 over flow is occurred or not.
6. Send to P1.5 and repeat from step 2

PROGRAM:

```
        MOV TMOD, #03H; Program timer 0 in mode 3
BACK1:  CLR TR1
        CLR TF1
        MOV TH0,#A4H; load 8-bit count value in TH0
        SETB TR1
BACK2:  MOV C, TF1
        JNC BACK2
        MOV C, P1.5
        CPL C
        MOV P1.5, C
        SJMP BACK1
```

RESULT:

I) .Procedure for Execution of Assembly language programs (8086) using TASM

1. Open TASM folder and click on DPMIRES icon and type EDIT to open the editor window.
2. Type the assembly language program and save the program as FILENAME.ASM.
3. Go to file menu and click on exit to move to DOS prompt.
4. Type **TASM FILENAME.ASM** command to assemble the program.
5. If any errors in the program use **EDIT FILENAME.ASM** command.
5. Type **TLINK FILENAME** command to link the file to turbo assembler.
6. Type **DEBUG FILENAME.EXE** to place the exe file in memory.
7. Type 'R' in the DOS prompt to see the register contents of 8086.
8. Type 'T' in the DOS prompt to observe the register contents used in the program ,instruction by instruction until end of the program.
9. Type 'G' in the DOS prompt to run the entire program at a time.(if you use INT 03H in program)
10. To observe the results available in memory locations type **E 16-bit address** after executing the program.

II). Procedure for execution of Interfacing programs (8086)

For generating op-codes using assembler:

1. Open the TASM folder and click on DPMIRES icon.
2. Type the DEBUG command to generate op-code for a program.
3. Type 'A' to assemble the program and type the program.
4. Type 'U' to un - assemble the program and to see the op-codes.

Interfacing 8086 with peripherals:

1. Switch on the power supply for 8086.
2. Press ENTER on the keyboard.
3. Press 'A' to enter the program directly on 8086 and then type 0400h and then ENTER.
4. Type the program.
5. Press 'G' and give the starting address of program and then ENTER.
6. Observe the result on corresponding device.

III). Procedure for 8051 Microcontroller

1. Switch on the power supply and press ENTER on the keyboard.
2. Press ‘A’ and then type ASM ORG 3000h then press ENTER.
3. Type the program and then press two times ENTER on keyboard.
4. To give the inputs on memory locations, press ‘M’ and type the input address.
5. Give the input values separated by **SPACE.**
6. Press two times ENTERS.
7. Press G and type the starting address of program then press ENTER to get EXECUTION COMPLETED.
8. Press RESET button & Press M and then type the output memory location to observe the result.

VIVA QUESTIONS

1. What is meant by .model small?
2. How data segment is assigned in program.
3. How code segment is assigned in program.
4. How stack segment is assigned in program
5. What is the difference between EQU & DB directive.
6. Mention the precautions to take while assigning the data to the variables.
7. What is the function of @?
8. What is the significance of $ at end of the string.
9. Mention the command to link the source file to the assembler.
10. Mention the command to place the machine codes into memory.
11. What happened if any instructions are used after end directive?
12. Define Assembler directive
13. Mention the classification of assembler directives.
14. What is the size of the segments if model small is assign to the program?
15. When the stack segment has to assign in the program.
16. What is the advantage of memory segmentation?
17. What are the data define and storage allocation directives.
18. Mention the significance of start directive
19. What are the rules & regulations for assembler?
20. Which DOS command has to use to verify the data in memory location?
21. Which DOS command has to use to proceed to the interrupt?
22. Which DOS command is used to exit from DOS?
23. What is the difference between DOS & BIOS?
24. Which command is used to verify all the register contents of 8086?
25. What is the function of P command?
26. What is the significance of G command?
27. What is the difference between MUL & I MUL instructions?
28. What is the difference between TEST & AND instructions?
29. What is the significance of CX register in logical instructions?
30. What is the significance of CX register in loop instructions?

31. What is the significance of CX register in string instructions?
32. What is the difference between REP & REPE?
33. What is the function of REP?
34. What is the function of REPE?
35. What is the significance of SI & DI Registers while executing string instructions?
36. What is the function of CLD instruction?
37. 0 to 9 BCD numbers can be represented in ASCII as?
38. To display the string attached to variable which identification number has to load in AH Register & which instruction has to use.
39. To accept the characters from keyboard which identification number has to load in AH reg & which Instruction has to use.
40. Which instruction has to use to display a character which is stored in DL register in the form of ASCII.
41. Source file has to save in the form of _____
42. To interface TLC to 8086 then 8255 the ports of 8255 operates in which mode.
43. To interface Stepper motor to 8086 through 8255, the ports of 8255 operates in which mode.
44. To interface DAC to 8086 through 8255, the ports of 8255 operates in which mode.
45. DAC 0800 gives output in terms of _____
46. Conversion formula for DAC.
47. Which type of addressing is used for ports of 8255?
48. What is the difference between CMP & SUB?
49. What is the forward & backward limit for conditional branch instructions?
50. What is the difference between Microprocessor & microcontroller?