14AEC26 MICROPROCESSORS & MICROCONTROLLERS LAB

List of Experiments:

CYCLE –I

8086 Microprocessor Programs using Software:

1. ALPs (8086) for addition and subtraction.
2. a) ALPs (8086) for multiplication and Division.
   b) ALPs (8086) to determine GCD and LCM of two 16-bit numbers.
3. ALPs (8086) to evaluate arithmetic expressions.
4. ALPs (8086) for sorting and searching.
5. Logic operations - Shift and rotate - Converting packed BCD to unpacked BCD, BCD to ASCII conversion.
6. String operations - Move block, Reverse string, String comparison, Length of string.

CYCLE –II

Interfacing:

7. ALPs (8086) for generating ramp wave, triangular wave, and stair case wave forms using DAC.
8. ALP (8086) for traffic light controller.
9. ALP (8086) for stepper motor control.

8051 Microcontroller:

10. ALP (8051) to determine the largest and smallest of N bytes.
11. 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.
12. a) ALP (8051) to determine LCM of two 8- bit numbers.
    b) ALP (8051) to determine GCD of two 8- bit numbers.

Date: **Exp. No – 1**

ALPs (8086) for addition and subtraction

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

AIM: To write an ALP (8086) to find out the sum of given 'n' 8-bit numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Define the data segment with required variables.
3. Initialize DS register with the starting address of data segment.
4. Clear the AX and BX registers.
5. Take the count value into CL register.
6. Copy the offset address of numlist and move the first value into BL.
7. Perform repeated addition till the count value becomes zero.
8. Store the sum in the offset address defined for RESULT in data segment.
9. Terminate the program.
10. Stop

PROGRAM:

```
        . MODEL SMALL
        . STACK
        . DATA
        COUNT EQU 6d
        NUMLIST DB 19h,29h,39h,49h,59h,99h
        RESULT DW 01h DUP (?)
        . CODE
        MOV AX, @DATA
        MOV DS, AX
        XOR AX, AX
        XOR BX, BX
        MOV CL, COUNT
        MOV SI, OFFSET NUMLIST
AGAIN:  MOV BL, [SI]
        ADD AX, BX
        INC SI
        DEC CL
        JNZ AGAIN
        MOV DI, OFFSET RESULT
        MOV [DI], AX
        INT 21h
        END
```

INPUT: 19h, 29h, 39h, 49h, 59h, 99h

OUTPUT: 01B6H

RESULT: Thus the sum of given ‘n’ 8-bit numbers has been executed successfully and the result is verified.

(B) Multi word Addition

AIM: To write an ALP (8086) to perform the addition of two 32-bit numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. LSW of 1st operand is moved to AX register.
3. LSW of 2nd operand is moved to BX register.
4. Add the contents of AX and BX registers and the result is stored in AX register.
5. Copy the LSW of result present in AX register into CX register.
6. MSW of 1st operand is moved to AX register.
7. MSW of 2nd operand is moved to BX register.
8. Add the contents of AX and BX registers along with carry (obtained from previous addition) and the result is stored in AX register.
9. Copy the MSW of the result present in AX register into DX register.
10. Terminate the program.
11. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. 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 21h
END
```

INPUT 1: 5678F000h

INPUT 2: 12341000h

OUTPUT: DX: CX=68AD0000h

RESULT: Thus the program for addition of two double words has been executed successfully by using TASM and result is verified.

(C) Multi Word Subtraction

AIM: To write an ALP (8086) to perform the subtraction of two 32-bit numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. LSW of 1st operand is moved to AX register.
3. LSW of 2nd operand is moved to BX register.
4. Subtract the contents of BX from AX register and the result is stored in AX register.
5. Copy the LSW of result present in AX register into CX register.
6. MSW of 1st operand is moved to AX register.
7. MSW of 2nd operand is moved to BX register.
8. Subtract the contents of BX from AX registers along with borrow (obtained from previous subtraction) and the result is stored in AX register.
9. Copy the MSW of the result present in AX register into DX register.
10. Terminate the program.
11. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. 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 21h
END
```

INPUT 1: 56780111h

INPUT 2: 12341000h

OUTPUT: DX: CX=4443F111h

RESULT: Thus the program for subtraction of two double words has been executed successfully by using TASM & result is verified.

Date: **Exp. No. – 2**

A) ALPs (8086) for multiplication and Division

(i) Multiplication of two unsigned 16-bit numbers

AIM: To write an ALP (8086) to perform the multiplication of two unsigned 16-bit numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Copy the first 16 bit operand into AX register.
3. Copy the second 16 bit operand into BX register.
4. Perform unsigned multiplication between the values stored in AX and BX, observe the MSW of the result present in DX and LSW in AX 5. Terminate the program.
5. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV AX, 1234h
MOV BX, 1234h
MUL BX
INT 21h
END
```

INPUT 1: 1234h

INPUT 2: 1234h

OUTPUT: DX:AX=014B5A90h

RESULT: Thus the program for multiplication of two 16-bit program executed successfully by using TASM & result is verified.

(ii) Multiplication of two signed 16-bit numbers

AIM: To write an ALP (8086) to perform the multiplication of two signed 16-bit numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Copy the first signed 16 bit operand into AX register.
3. Copy the second 16 bit operand into BX register.
4. Perform signed multiplication between the values stored in AX and BX , observe the MSW of the result present in DX and LSW in AX.
5. Terminate the program.
6. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV AX, 8000h → Negative number
MOV BX, 2000h → Positive number
IMUL BX
INT 21h
END
```

INPUT 1: 8000h

INPUT 2: 2000h

OUTPUT: DX: AX=F0000000h

RESULT: Thus the program for multiplication of two signed numbers has been executed successfully by using TASM and result is verified.

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

AIM: To write an ALP (8086) to perform the multiword division of 16-bit by 08 bit number.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Copy the 16 bit dividend into AX register.
3. Copy the 8 bit divisor into BL register.
4. Perform division between the values stored in AX and BL, observe the quotient in AL and remainder in AH.
5. Terminate the program.
6. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV AX, 1234h
MOV BL, 58h
DIV BL
INT 21h
END
```

INPUT 1: 1234h

INPUT 2: 58h

OUTPUT: AL (Q) = 34H

AL (R) =54H

RESULT: Thus the division of 16-bit by 8-bit number program executed successfully by using TASM and result is verified.

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

AIM: To write an ALP (8086) to perform the division of two signed numbers.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Copy the 16 bit dividend into AX register.
3. Copy the 8 bit divisor into BL register.
4. Perform division between the values stored in AX and BL, observe the quotient in AL and remainder in AH.
5. Terminate the program.
6. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV AX, -0002h
MOV BL, 80h
IDIV BL
INT 21h
END
```

INPUT: AX← -0002h
BL← 80h

OUTPUT: AL (Q) = 00h
AH(R) = FEh

RESULT: Thus the program for division of two signed numbers has been executed successfully by using TASM and result is verified.

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

AIM: To write an ALP (8086) to perform the multiword division of 32-bit by 16 bit number.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Copy the 32 bit dividend into DX: AX register.
3. Copy the 16 bit divisor into BX register.
4. Perform division between the values stored in DX: AX and BX, observe the quotient in AX and remainder in DX.
5. Terminate the program.
6. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV AX, 2786h (LSW of dividend)
MOV DX, 2334h (MSW of dividend)
MOV BX, 3552h (divisor)
DIV BX
INT 21h
END
```

INPUT 1: DX: AX=23342786h

INPUT 2: BX=3552h

OUTPUT: AX (Q) = A904h DX (R) =303Eh

RESULT: Thus the multiword division of 32-bit by 16 bit number program executed successfully by using TASM and result is verified.

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

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

AIM: To write an ALP (8086) to find out GCD of two 16-bit binary numbers

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Define the data segment with required variables.
3. Initialize DS register with the starting address of data segment.
4. Take 1st number into AX register and 2nd number into BX register.
5. 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.
6. Copy the result in AX to variable name GCD in data segment.
7. Terminate the program
8. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
NUM1 DW 003Ch
NUM2 DW 000Fh
GCD DW 01 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 21h
END
```

INPUT1: 003Ch

INPUT2: 000Fh

OUTPUT: 000Fh

RESULT: Thus the ALP for finding the GCD of two 16 –bit numbers was performed successfully.

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

AIM: To write an ALP (8086) to find out LCM of two 16-bit binary numbers

APPARATUS: system with TASM software.

ALGORITHM:

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

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
NUM1 DW 003Ch
NUM2 DW 000Fh
LCM DW 01 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 21h
END
```

INPUT1: 003Ch

INPUT2: 000Fh

OUTPUT: 003Ch

RESULT: Thus the ALP for finding the LCM of two 16 –bit numbers was performed successfully.

Date: **Exp. No – 3**

ALP (8086) to evaluate arithmetic expressions

AIM: To write an ALP (8086) to evaluate given arithmetic expression is $f = \frac{(a+b)\,(b+c)\,(c+d)}{(a+b+c+d)}$

APPARATUS: System with TASM software.

ALGORITHM:

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

PROGRAM:

```
. MODEL SMALL
. STACK
. 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 21H
END
```

INPUT: a=01h, b=02h, c=03h, d=04

OUTPUT: AX = 050Ah

RESULT: Thus the program for given arithmetic expression was successfully executed.

Date: **Exp. No – 4**

ALPs (8086) for sorting and searching

(A) Sorting a string in an ascending order

AIM: To write an ALP (8086) to perform the string operation for sorting a string in an ascending order.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Define the data segment with required variables.
3. Initialize DS register with the starting address of data segment.
4. Specify the count value for external loop in DX and copy into CX register.
5. Take the offset address of list into SI register.
6. Copy the first number in list into the AL register.
7. Compare the number in AL register with subsequent number and exchange the position of the numbers depending on result of the comparison.
8. Repeat step7 until the value present in CX register is Zero.
9. Repeat the steps 6,7and 8 until the DX becomes Zero and observe the results in ascending order.
10. Terminate the program.
11. Stop.

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
LIST DB 53h, 10h, 24h, 12h
. CODE
MOV AX, @DATA
MOV DS, AX
MOV DX, 03h
AGAIN2: MOV CX, DX
MOV SI, OFFSET LIST
AGAIN1: MOV AL, [SI]
CMP AL, [SI+1]
JL PR1
XCHG [SI+1], AL
XCHG [SI], AL
PR1: ADD SI, 01h
LOOP AGAIN1
DEC DX
JNZ AGAIN2
MOV AH, 4Ch
INT 21h
END
```

INPUT: Enter string: 53h, 10h, 24h, 12h

OUTPUT: Sorted String: 10h, 12h, 24h, 53h

RESULT: Thus the program for sorting a string in an ascending order is executed successfully by using TASM and result is verified.

(B) Sorting a string in an descending order

AIM: To write an ALP (8086) to perform the string operation for sorting a string in an descending order.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Define the data segment with required variables.
3. Initialize DS register with the starting address of data segment.
4. Specify the count value for external loop in DX and copy into CX register.
5. Take the offset address of list into SI register.
6. Copy the first number in list into the AL register.
7. Compare the number in AL register with subsequent number and exchange the position of the numbers depending on result of the comparison.
8. Repeat step7 until the value present in CX register is Zero.
9. Repeat the steps 6,7& 8 until the DX becomes Zero and observe the results in descending order.
10. Terminate the program.
11. Stop.

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
LIST  DB 53h, 10h, 24h, 12h
. CODE
MOV AX, @DATA
MOV DS, AX
MOV DX, 03h
AGAIN2: MOV CX, DX
MOV SI, OFFSET LIST
AGAIN1: MOV AL, [SI]
CMP AL, [SI+1]
JG PR1
XCHG [SI+1], AL
XCHG [SI], AL
PR1: ADD SI, 01h
LOOP AGAIN1
DEC DX
JNZ AGAIN2
MOV AH, 4C
INT 21h
END
```

INPUT: Enter string: 53h, 10h, 24h, 12h

OUTPUT: Sorted String: 53h, 24h, 12h, 10h

RESULT: Thus the program for sorting a string in descending order is executed successfully by TASM and result is verified.

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

AIM: To write an ALP (8086) to determine the smallest number of a given array.

APPARATUS: System with TASM software.

ALGORITHM:

1. Start
2. Define the data segment with required variables.
3. Initialize DS register with the starting address of data segment.
4. Take the offset address of array into SI register.
5. Copy the count value into CL register and clear the AX register.
6. Take the first number in list into AL regiser.
7. Compare the number in AL register with subsequent number and copy the smallest number into AL register depending on result of the comparison.
8. Repeat step7 until the value present in CL register is Zero.
9. Terminate the program.
10. Stop.

PROGRAM:

```
        . MODEL SMALL
        . STACK
        . DATA
        LIST DB 02h, 09h, 03h, 06h, 08h, 07
        . 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]
        JB GO
        MOV AL, [SI]
  GO:   LOOP UP
        INT 21h
        END
```

INPUT: 02h, 09h, 03h, 06h, 08h, 07

OUTPUT: AL = 02H

RESULT: Thus the smallest number of a given 'n' number program was executed successfully using TASM and result is verified.

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

AIM: To write an ALP (8086) to determine the smallest number of a given array.

APPARATUS: System with TASM software.

ALGORITHM:

1. Start
2. Define the data segment with required variables.
3. Initialize DS register with the starting address of data segment.
4. Take the offset address of array into SI register.
5. Copy the count value into CL register and clear the AX register.
6. Take the first number in list into AL register.
7. Compare the number in AL register with subsequent number and copy the largest number into AL register depending on result of the comparison.
8. Repeat step7 until the value present in CL register is Zero.
9. Terminate the program.
10. Stop.

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:   LOOP UP
        INT 21h
        END
```

INPUT: 02h, 09h, 03h, 06h, 08h, 07

OUTPUT: AL = 09H

RESULT: Thus the largest number of a given 'n' number program was executed successfully using TASM and result is verified.

(E) Search for a given number

AIM: To write an ALP (8086) to search for a given number in an array.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Define the data segment with required variables.
3. Initialize DS and ES register with the starting address of data segment.
4. Copy the count value into CL register.
5. Get the offset address of array into SI register and clear the direction flag.
6. Copy the number to be search in AL register and scan repeatedly.
7. If not found display the message BYTE NOT FOUND.
8. If found display the message BYTE FOUND.
9. Terminate the program
10. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
COUNT EQU 3D
ARRAY DB 11h, 21h, 32h
MSG DB 'BYTE FOUND','$'
MSG1 DB 'BYTE NOT FOUND','$'
.CODE
MOV AX, @DATA
  MOV DS, AX
  MOV ES, AX
  MOV CL, COUNT
  MOV SI, OFFSET ARRAY
  CLD
  MOV AL, 32H
  REPNE SCASB
  JZ A2
  MOV AH, 09H
LEA DX, MSG1
 INT 21H
 JMP A3
A2: MOV AH, 09H
    LEA DX, MSG
    INT 21H
A3: INT 03H
    END
```

INPUT: 11H, 21H, 32H

OUTPUT: BYTE FOUND

RESULT: Thus the program for searching a given number was executed successfully using TASM and result is verified.

Date: **Exp. No – 5**

Logical Operations

(A) Shift Logical Right

AIM: To write an ALP (8086) to perform the shift logical right operation.

APPARATUS: System with TASM Software

ALGORITHM:

1. Start
2. Move the data to be shifted into AL register.
3. Specify the number of shift operations in CL register.
4. Perform shift logical right operation and observe the result in AL register.
5. Terminate the program.
6. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV AL, 46H
MOV CL, 04H
SHR AL, CL
INT 21H
END
```

INPUT: AL = 46h CL = 04h

OUTPUT: AL = 04h

RESULT: Thus the program for Shift right operation has been executed successfully by using TASM and result is verified.

(B) Shift Logical Left

AIM: To write an ALP (8086) to perform the shift logical left operation.

APPARATUS: System with TASM Software

ALGORITHM:

1. Start
2. Move the data to be shifted into AL register.
3. Specify the number of shift operations in CL register.
4. Perform shift logical left operation and observe the result in AL register.
5. Terminate the program.
6. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV AL, 46H
MOV CL, 04H
SHL AL, CL
INT 21H
END
```

INPUT: AL = 46h CL = 04h

OUTPUT: AL = 60h

RESULT: Thus the program for Shift left operation has been executed successfully by using TASM and result is verified.

(C) Rotate right without Carry

AIM: To write an ALP (8086) to perform the Rotate right without Carry operation.

APPARATUS: System with TASM Software

ALGORITHM:

1. Start
2. Move the data to be shifted into AL register.
3. Specify the number of rotate operations in CL register.
4. Perform rotate right operation without carry and observe the result in AL register.
5. Terminate the program.
6. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV AL, 68H
MOV CL, 04H
ROR AL, CL
INT 21H
END
```

INPUT: AL = 68h CL = 04h

OUTPUT: AL = 86h

RESULT: Thus the program for rotate right without carry operation has been executed successfully by using TASM and result is verified.

(D) Rotate left without Carry

AIM: To write an ALP (8086) to perform the Rotate left without Carry operation.

APPARATUS: System with TASM Software

ALGORITHM:

1. Start
2. Move the data to be shifted into AL register.
3. Specify the number of rotate operations in CL register.
4. Perform rotate left operation without carry and observe the result in AL register.
5. Terminate the program.
6. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV AL, 60H
MOV CL, 04H
ROL AL, CL
INT 21H
END
```

INPUT: AL = 60h CL = 04h

OUTPUT: AL = 06h

RESULT: Thus the program for rotate left without carry operation has been executed successfully by using TASM and result is verified.

(E) Rotate right with Carry

AIM: To write an ALP (8086) to perform the Rotate right with Carry operation.

APPARATUS: System with TASM Software

ALGORITHM:

1. Start
2. Move the data to be shifted into AL register.
3. Specify the number of rotate operations in CL register.
4. Perform rotate right operation with carry and observe the result in AL register.
5. Terminate the program.
6. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV AL, 68H
MOV CL, 04H
RCR AL, CL
INT 21H
END
```

INPUT: AL = 68h CL = 04h

OUTPUT: AL = 06h

RESULT: Thus the program for rotate right with carry operation has been executed successfully by using TASM and result is verified.

(F) Rotate left with Carry

AIM: To write an ALP (8086) to perform the Rotate left with Carry operation.

APPARATUS: System with TASM Software

ALGORITHM:

1. Start
2. Move the data to be shifted into AL register.
3. Specify the number of rotate operations in CL register.
4. Perform rotate left operation with carry and observe the result in AL register.
5. Terminate the program.
6. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV AL, 68h
MOV CL, 04h
RCL AL, CL
INT 21H
END
```

INPUT: AL = 68h CL = 04h

OUTPUT: AL =83h

RESULT: Thus the program for rotate left with carry operation has been executed successfully by using TASM and result is verified.

(G) Converting packed BCD to Unpacked BCD

AIM: To write an ALP (8086) to convert from packed BCD to Unpacked BCD numbers..

APPARATUS: System with TASM Software

ALGORITHM:

1. Start
2. Move the packed data into BL register and count value into BH register temporarily.
3. Copy the packed BCD data from BL to AL register and count value from BH to CL registers.
4. Perform shift left & rotate right operations by the AL with the count number of times specified.
5. Copy the lower byte of unpacked BCD data to DL register.
6. Copy the packed BCD data from BL to AL register and count value from BH to CL registers.
7. Perform shift right operations by the AL with the count number of times specified.
8. Copy the higher byte of unpacked BCD data to DH register.
9. Terminate the program.
10. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV BL, 57H
MOV BH, 04H
MOV AL, BL
MOV CL, BH
SHL AL, CL
ROR AL, CL
MOV DL, AL
MOV AL, BL
MOV CL, BH
SHR AL, CL
MOV DH, AL
INT 21H
END
```

INPUT: Packed BCD = 57

OUTPUT: Unpacked BCD (DX) = 0507

RESULT: Thus the program for converting from packed BCD to unpacked BCD operation has been executed successfully by using TASM and result is verified.

(H) Converting BCD to ASCII number

AIM: To write an ALP (8086) to convert from BCD to ASCII numbers..

APPARATUS: System with TASM Software

ALGORITHM:

1. Start
2. Move the packed data into BL register and count value into BH register temporarily.
3. Copy the packed BCD data from BL to AL register and count value from BH to CL registers.
4. Perform shift left & rotate right operations by the AL with the count number of times specified.
5. Perform XOR and copy the lower byte of unpacked BCD data to DL register.
6. Copy the packed BCD data from BL to AL register and count value from BH to CL registers.
7. Perform shift right operations by the AL with the count number of times specified.
8. Perform XOR and copy the higher byte of unpacked BCD data to DH register.
9. Terminate the program.
10. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
. CODE
MOV BL, 57H
MOV BH, 04H
MOV AL, BL
MOV CL, BH
SHL AL, CL
ROR AL, CL
MOV DL, AL
MOV AL, BL
MOV CL, BH
SHR AL, CL
MOV DH, AL
XOR DX, 3030h
INT 21H
END
```

INPUT: BCD number: 57

OUTPUT: ASCII Number (DX) = 3537

RESULT: Thus the program for converting from BCD to ASCII has been executed successfully by using TASM and result is verified.

Date: **Exp. No –6**

ALPs (8086) for String operations

(A) Move a block one segment to another segment

AIM: To write an ALP (8086) to move the block of data from one segment to another segment.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Define the data segment with required variables.
3. Initialize DS register with the starting address of source segment.
4. Initialize ES register with the starting address of destination segment.
5. Copy the number of bytes in the string to CL register.
6. Store the offset address in source and destination segments in SI and DI registers respectively.
7. Select the auto increment/decrement of offset address using direction flag.
8. Move the string bytes from source segment to destination segment until all the bytes are moved.
9. Terminate the program.
10. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
STRING DB 'COMPUTER'
STRING1 DB 8 DUP (?)
. CODE
MOV AX, @DATA
MOV DS, AX
MOV ES, AX
MOV CL, 08H
MOV SI, OFFSET STRING
MOV DI, OFFSET STRING1
CLD
REP MOVSB
INT 21H
END
```

INPUT: COMPUTER

OUTPUT: 43h, 4Fh, 4Dh, 50h, 55h, 54h, 45h,52h

RESULT: Thus the program to move a block of string from one memory location to another memory location is executed successfully.

(B) Reverse String

AIM: To write an ALP (8086) to reverse the string using TASM software.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Define the data segment with required variables.
3. Initialize DS register with the starting address of source segment.
4. Initialize ES register with the starting address of destination segment.
5. Copy the number of bytes in the string to CL register.
6. Copy the offset address of STG into SI register and offset address of ATG1 into DI register.
7. Clear direction flag for copying from lowest address to highest address.
8. Copy the string characters in reverse order from source segment to destination segment.
9. Repeat step8 until the count becomes zero in CL register.
10. Terminate the program
11. Stop

PROGRAM:

```
.MODEL SMALL
.STACK
.DATA
STG DB ‘SVCET’,’$’
STG1 DB 05H DUP (?), ‘$,
COUNT EQU 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: SVCET

OUTPUT: TECVS

RESULT: Thus the ALP for performing the reverse string operation has been successfully executed.

(C) String Comparison:

AIM: To write an ALP (8086) to compare the strings.

APPARATUS: system with TASM software.

ALGORITHM:

1. Start
2. Define the data segment with required variables.
3. Initialize DS register with the starting address of source segment where STRING1 present.
4. Initialize ES register with the starting address of destination segment where STRING 2 present.
5. Copy the number of bytes in the string to CL register.
6. Store the offset address of strings in source and destination segments in SI and DI registers respectively.
7. Select the auto increment/decrement of offset address using direction flag.
8. Compare the string bytes of STRING1 with STRING2 until all the bytes are compared
9. Display the result based on comparison.
10. Terminate the program.
11. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. DATA
STRG1 DB 'LAB','$'
STRG 2 DB 'LAB', $'
RESULT  DB 'STRGS ARE EQUAL','$'
RESULT1 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
REP CMPSB
JNZ LOOP1
MOV AH, 09H
LEA DX, RESULT
INT 21H
JMP A1
LOOP1: MOV AH, 09H
LEA DX, RESULT1
INT 21H
A1: MOV AH, 4CH
INT 21H
END
```

INPUT1: LAB

INPUT2: LAB

OUTPUT: STRGS ARE EQUAL

RESULT: Thus the program of string comparison is executed successfully and the result is verified.

(D) Length of string

AIM: To write an ALP (8086) to find out the length of given string.

APPARATUS: System with TASM software.

ALGORITHM:

1. Start
2. Define the data segment with required variables.
3. Initialize DS register with the starting address of source segment where STRING1 present.
4. Load effective address of label byte string in DX register.
5. Perform the operation to calculate the length of the given string and display.
6. Terminate the program.
7. Stop

PROGRAM:

```
. MODEL SMALL
. STACK
. 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: SVCET

OUTPUT: 5

RESULT: Thus the program to find the length of the string is executed successfully by TASM and result is verified.

Date: **Exp.No. --7**

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

AIM: To write an ALP (8086) to generate a ramp wave form using DAC.

APPARATUS: 1. 8086 Microprocessor kit.

2. DAC kit.

ALGORITHM:

1. Start
2. Initialize the 8255 port through CWR 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.
8. Terminate the program.
9. Stop

PROGRAM:

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

RESULT: Thus the ALP program of DAC using 8255 generate a ramp waveform that is successfully executed and result is observed in the CRO.

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

AIM: To write an ALP (8086) to generate a triangular waveform using DAC.

APPARATUS: 1. 8086 Microprocessor kit.

2. DAC kit.

ALGORITHM:

1. Start
2. Initialize the 8255 port through CWR 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.
8. Terminate the program.
9. Stop

PROGRAM:

```
        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
        RET
```

RESULT: Thus the ALP program of DAC using 8255 generate a triangular waveform that is successfully executed and result is observed in the CRO.

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

AIM: To write an ALP (8086) to generate a stair case waveform using DAC.

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

ALGORITHM:

1. Start
2. Initialize the 8255 port through CWR 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.
8. Terminate the program.
9. Stop

PROGRAM:

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

RESULT: Thus the ALP program of DAC using 8255 generate a staircase waveform that is successfully executed & result is observed in the CRO.

Date: **Exp. No – 8**

ALP (8086) for traffic light controller

AIM: To write an ALP (8086) in four road cross junction to control the traffic automatically.

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

ALGORITHM:

1. Start
2. Initialize the 8255 port through CWR 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.
10. Terminate the program.
11. Stop

PROGRAM:

```
        ORG 0400H
        MOV AL, 80H
        OUT 76H, AL
BACK:   MOV AL, 11H        ; READ ONLY
        OUT 74H, AL        ; FOR E-W
        MOV AL, 44H        ; GREEN ON
```

```
        OUT 70H, AL          ; FOR N-S
        CALL DELAY1          ; DELAY
        MOV AL, 22H          ; YELLOW ON
        OUT 70H, AL          ; ALL
        OUT 74H, AL          ; SIDES
        CALL DELAY2          ; 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
        MOV AL, 22H
        OUT 70H, AL
        OUT 74H, AL
        CALL DELAY2
        JMP BACK
DELAY1: MOV BX, 000FH
BACK2: MOV CX, FFFFH
BACK1: DEC CX
        JNZ BACK1
        DEC BX
        JNZ BACK2
        RET
DELAY2: MOV BX, 0005H
BACK4: MOV CX, FFFFH
BACK3: DEC CX
        JNZ BACK3
        DEC BX
        JNZ BACK4
        RET
```

RESULT: Thus an ALP (8086) in four road cross junction to control the traffic automatically completed successfully.

Date: **Exp. No – 9**

ALP (8086) for stepper motor control

AIM: To write an ALP to interface a stepper motor with 8086 and operate it in anti-clockwise continuously.

APPARATUS: 1. 8086 Microprocessor kit.

2. Stepper Motor control board.

ALGORITHM:

1. Start
2. Initialize the 8255 port through CWR with 80h.
3. Copy the content as 0Ah into AL and send through port A with some delay.
4. Copy the content as 06h into AL and send through port A with some delay.
5. Copy the content as 05h into AL and send through port A with some delay.
6. Copy the content as 09h into AL and send through port A with some delay.
7. Repeat from step3 to step8 for stepper motor to rotate anti-clockwise continuously.
8. Terminate the program.
9. Stop

PROGRAM:

```
        ORG 0400H
        MOV AL, 80H
        OUT 76H, AL
START:  MOV AL, 0AH
        OUT 70H, AL
        CALL DELAY
        MOV AL, 06H
        OUT 70H, AL
        CALL DELAY
        MOV AL, 05H
        OUT 70H, AL
        CALL DELAY
        MOV AL, 09H
        OUT 70H, AL
        CALL DELAY
        JMP START
DELAY:  MOV CX, 0000H
BACK:   DEC CX
        JNZ BACK
        RET
```

RESULT: Thus an ALP to interface a stepper motor with 8086 and operate it in anti-clockwise continuously is successfully completed.

Date: **Exp. No – 10**

ALP (8051) to determine the largest and smallest of N bytes

AIM: To write an ALP (8051) to determine the largest and smallest of N bytes.

APPARATUS: 8051 Microcontroller kit.

ALGORITHM:

1. Start
2. Access the initial memory location 3500h through DPTR and copy the content into A register.
3. Copy the number bytes in an array from A to register R1.
4. Increment DPTR to access next memory location and copy the next byte into A register.
5. Assume the largest in R2 and smallest in R3 is stored and initial value will get from A.
6. Decrement the count value in R1 and increment DPTR to copy the next byte into A register.
7. Clear carry flag and perform the subtraction content R2 from A.
7. If no carry is generated then content of A is largest and copy into R2 register.
8. If carry is generated add A and R2 then clear the carry flag.
9. Subtract the content of R3 from A and no carry is generated go to the next number in array.
10. If carry generated add the content of A and R3 then copy A into R3 and jump to the next number in array.
11. Copy the largest number in 4000h from R2 and smallest number in 4001h from R3.
12. Terminate the program.
13. Stop

PROGRAM:

```
        ORG 3000H
        MOV DPTR, #3500H
        MOVX A, @ DPTR
        MOV R1, A
        INC DPTR
        MOVX A, @ DPTR
```

```
        MOV R2, A
        MOV R3, A
        DEC R1
BACK:   INC DPTR
        MOVX A, @ DPTR
        CLR C
        SUBB A, R2
        JNC AHEAD
        ADD A, R2
        CLR C
        SUBB A, R3
        JNC NEXT
        ADD A, R3
        MOV R3, A
        SJMP NEXT
AHEAD:  ADD A, R2
        MOV R2, A
NEXT:   DJNZ R1, BACK
        MOV DPTR, #3600H
        MOV A, R2
        MOVX @ DPTR, A
        INC DPTR
        MOV A, R3
        MOVX @ DPTR, A
        LJMP 16A5H
```

INPUT: 3500=04h (length of array)

3501=10h 3502=04h 3503=15h 3504=30h

OUTPUT: 3600=30h (largest number) 3601=04h (smallest number)

RESULT: Thus the program for finding the largest and smallest number in an array was executed successfully on 8051 microcontroller.

Date: **Exp. No – 11**

(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
14. Stop

PROGRAM:

```
        ORG 3000H
        MOV DPTR, #3500H
        MOVX A, @ DPTR
        MOV R2, A
        INC DPTR
        MOVX A, @ DPTR
        MOV R1, A
        INC DPTR
        MOVX A, @ DPTR
        MOV R3, A
        MOV DPTR, # 3600H
        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, # 00H
        MOVX @ DPTR, A
        LJMP 16A5H
```

INPUT: 3500=34h (LSB of 16-bit number), 3501=12h (MSB of 16-bit number)
3502=12h (8-bit number)

OUTPUT: 3600=A8h (LSB of 24-bit result), 3601=47h, 3602=01h (MSB of 24-bit result)

RESULT: Thus the multiplication of 16-bit by 8-bit number program was executed successfully and result is verified.

(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 3600h location through A register.
7. If not equal increment R1 and repeat step6.
8. Terminate the program
9. Stop

PROGRAM:

```
        ORG 3000H
        MOV DPTR, #3500H
        MOVX A, @ DPTR
        MOV 45H, A
        MOV R1, #00H
BACK:   MOV A, R1
        MOV B, A
        MUL AB
        CJNE A, 45H, NEXT
        MOV DPTR, #3600H
        MOV A, R1
        MOVX @DPTR, A
        LJMP 16A5H
NEXT:   INC R1
        SJMP BACK
```

INPUT: 3500=64h (100d)

OUTPUT: 3600=0Ah (10d)

RESULT: Thus the square root of 8-bit number program was executed successfully and result is

verified.

Date: **Exp. No – 12**

(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 3000H
MOV DPTR, #3500H
MOVX A,@DPTR
MOV R1, A
INC DPTR
MOVX A, @DPTR
MOV R2, A
MOV 10H, R1
MOV 20H, R2
BACK: MOV A, 10H
CJNE A, 20H, AHEAD
SJMP RESULT
AHEAD: CLR C
SUBB A, 20H
JC FORWARD
MOV A, 20H
ADD A, R2
MOV 20H, A
SJMP BACK
FORWARD: MOV A, 10H
ADD A, R1
MOV 10H, A
SJMP BACK
RESULT: MOV DPTR, #3600H
MOVX @DPTR, A
LJMP 16A5H
```

INPUT1: 30h

INPUT2: 20h

OUTPUT: 60h

RESULT: Thus the LCM of given 8-bit numbers program was executed successfully and result is verified.

(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
10. Stop

PROGRAM:

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

INPUT1: 3Ch

INPUT2: 24h

OUTPUT: 0Ch

RESULT: Thus the GCD of given 8-bit numbers program was executed successfully and result is verified.