Experiment List

Thadomal Shahani Engineering College

Computer Engineering Department

Subject: Microprocessors

Year: 2022-2023

Semester: IV

Class: C1, C2 & C3

r.No.	TOPICS
1.	Assembly program to display the contents of the flag register.
2.	Assembly programming for 8-bit addition, subtraction.
3.	Assembly programming for 16-bit addition, subtraction, multiplication and division
4.	Assembly program to accept a string, display a string, print a string, find the length of a string, check if given string is a palindrome.(menu based)
5.	Assembly program to sort numbers in ascending/ descending order.
6.	Assembly program to find minimum/ maximum no. from a given array.
7.	Assembly program to find factorial of number using procedure.
8.	Mixed Language program to find the GCD/LCM of two numbers.
9.	Mixed Language program to increment, decrement the size of the cursor and also to disable it.
10.	Assembly program to rotate a stepper motor in clockwise direction using 8086 MP and 8255

Gmshukla

Dr. Gauri Shukla

Subject Incharge

Dar

Experiment No 2

Objective:

Write an assembly language program for 8 bit addition, subtraction, multiplication and division.

Prerequisite:

TASM assembler

Algorithm for 8 bit addition:

1) Start
2) Initialize data segment through AX register in the DS register.
3) Display the message as "Enter the first number"
4) Read first digit in AL register through keyboard (e.g. AL=31h)
5) Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number.AL=01h
6) Move contents of AL register to a BL. (BL← AL so BL=01h)
7) Rotate the contents of BL register by 4 positions at left side. (BL=10h)
8) Read a second digit in AL register through keyboard AL=35h
9) Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number. AL=05h
10) Add the contents of BL and AL store the result in BL (BL←BL+AL so BL=15h)
11) Display the message as "Enter the second number"
12) Read first digit in AL register through keyboard AL=32h
13) Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number.AL=02h
14) Move contents of AL register to a CL. (CL← AL so CL=02h)
15) Rotate the contents of CL register by 4 positions at left side. (CL=20h)

16) Read a second digit in AL register through keyboard (AL=33h)

17) Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number. AL=03h

18) Add the contents of CL and AL store the result in CL (CL←CL+AL so CL=23h)

(Now both numbers are accepted as 15h and 23h)

19) Add the contents of BL and CL and result gets stored in BL (E.g ADD BL,CL so BL=38h)

20) Preserve the result of addition in some temporary variable say temp from BL.

21) Mask the first nibble by AND operation with number F0h (AND BL,F0h so BL=30h)

22) Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BL=33h)

23) Move the contents of BL to DL and display it on the screen

24) Move result from temporary variable to BL again (So BL=38h)

25) Mask the second nibble by AND operation with number 0Fh (AND BL,0Fh so BL=08h)

26) Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BL=38h)

27) Move the contents of BL to DL and display it on the screen

28) Stop

Algorithm for Input procedure:(To accept input from 0 to F)

1) Compare the contents of AL with 41h.

2) Jump to step no 4 if carry flag is set(digit is in the range of 0 to 9 so add only 30h)

3) Sub 07h to AL register(If digit is in the range from A to F then add 30h and 7h both)

4) Sub 30h to AL register

5) Return

Algorithm for Output procedure:

1) Compare the contents of BL with 0Ah
2) Jump to step no 4 if carry flag is set(digit is in the range of 0 to 9 so add only 30h)
3) Add 07h to BL register(If digit is in the range from A to F then add 30h and 7h both)
4) Add 30h to BL register
5) Return

Note:

While masking F or f is not case sensitive. But in input procedure 41h number is considered for comparison because 41h is ASCII hex value for 'A'. In output procedure '0A' is considered not 'a' is considered as small case a has 61h ASCII hex value.So this input and output procedure are applicable for only capital 'A' to 'F'

Algorithm for 8 bit subtraction:

1. Start
2. Initialize data segment through AX register in the DS register.
3. Display the message as "Enter the first number"
4. Read first digit in AL register through keyboard (e.g. AL=31h)
5. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number.AL=01h
6. Move contents of AL register to a BL. (BL← AL so BL=01h)
7. Rotate the contents of BL register by 4 positions at left side. (BL=10h)
8. Read a second digit in AL register through keyboard AL=35h
9. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number. AL=05h

10. Add the contents of BL and AL store the result in BL (BL←BL+AL so BL=15h)

11. Display the message as "Enter the second number"

12. Read first digit in AL register through keyboard AL=32h

13. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number.AL=02h

14. Move contents of AL register to a CL. (CL← AL so CL=02h)

15. Rotate the contents of CL register by 4 positions at left side. (CL=20h)

16. Read a second digit in AL register through keyboard (AL=33h)

17. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number. AL=03h

18. Add the contents of CL and AL store the result in CL (CL←CL+AL so CL=23h) (Now both numbers are accepted as 15h and 23h)

19. Subtract the contents of CL from BL and result gets stored in BL (E.g SUB BL,CL so BL=F2h)

20. Preserve the result in some temporary variable say temp from BL.

21. Mask the first nibble by AND operation with number F0h (AND BL,F0h so BL=30h)

22. Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BL=33h)

23. Move the contents of BL to DL and display it on the screen

24. Move result from temporary variable to BL again (So BL=38h)

25. Mask the second nibble by AND operation with number 0Fh (AND BL,0Fh so BL=08h)

26. Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BL=38h)

27. Move the contents of BL to DL and display it on the screen

28. Stop

Algorithm for Input procedure:(To accept input from 0 to f)

1. Compare the contents of AL with 41h
2. Jump to step no 4 if carry flag is set
3. Sub 07h to AL register
4. Sub 30h to AL register
5. Return

Algorithm for Output procedure:

1. Compare the contents of BL with 0Ah
2. Jump to step no 4 if carry flag is set
3. Add 07h to BL register
4. Add 30h to BL register
5. Return

Note:

While masking F or f is not case sensitive. But in input procedure 41h number is considered for comparison because 41h is ASCII hex value for 'A'. In output procedure '0A' is considered not 'a' is considered as small case a has 61h ASCII hex value.So this input and output procedure are applicable for only capital 'A' to 'F'.

MUL instruction-

- Multiplies unsigned byte/word from source by unsigned byte/word from AL/AX (multiplier) reg.
- Syntax-

  MUL source
- Result of 8bit X 8 bit will go in AX register and 16 bit X 16 bit will go in DX:AX register.

Algorithm for 8 bit Multiplication:

1. Start

Experiment No 3

Objective:

Write an assembly language program to add/subtract/multiply/divide two 16-bit numbers.

Prerequisite:

TASM assembler

Algorithm

1. Start
2. Initialize data segment through AX register in the DS register.
3. Display the 3 text message as "1. 16 bit addition 2 .16 bit subtraction 3. Exit Enter your choice"
4. Compare accepted choice with 03h.
5. If zero flag is set then goto step no.6 otherwise goto step no 7
6. Exit the program
7. Display the message as "Enter first 16 bit number"
8. Read first digit in AL register through keyboard (e.g. AL=32h)
9. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number.AL=02h
10. Move contents of AH with 00h. (AH← 00h so AX=0002h)
11. Rotate AX contents in left directions by 12 bits. (AX=2000h)
12. Move the contents of AX to BX(BX←AX so BX=2000h)
13. Read a second digit in AL register through keyboard AL=35h
14. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number. AL=05h
15. Move contents of AH with 00h. (AH← 00h so AX=0005h)
16. Rotate AX contents in left directions by 8 bits. (AX=0500h)
17. Add the contents of AX and BX (BX=BX+AX so BX=2500h)
18. Read a third digit in AL register through keyboard AL=31h

19. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number. AL=01h

20. Move contents of AH with 00h. (AH← 00h so AX=0001h)

21. Rotate AX contents in left directions by 4 bits. (AX=0010h)

22. Add the contents of AX and BX (BX=BX+AX so BX=2510h)

23. Read a fourth digit in AL register through keyboard AL=30h

24. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number. AL=00h

25. Move contents of AH with 00h. (AH← 00h so AX=0000h)

26. Add the contents of AX and BX (BX=BX+AX so BX=2510h)

27. Display the message as "Enter second 16 bit number"

28. Read first digit in AL register through keyboard (e.g. AL=37h)

29. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number.AL=07h

30. Move contents of AH with 00h. (AH← 00h so AX=0007h)

31. Rotate AX contents in left directions by 12 bits. (AX=7000h)

32. Move the contents of AX to CX(CX←AX so CX=7000h)

33. Read a second digit in AL register through keyboard AL=35h

34. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number. AL=05h

35. Move contents of AH with 00h. (AH← 00h so AX=0005h)

36. Rotate AX contents in left directions by 8 bits. (AX=0500h)

37. Add the contents of AX and CX (CX=CX+AX so CX=7500h)

38. Read a third digit in AL register through keyboard AL=31h

39. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number. AL=01h

40. Move contents of AH with 00h. (AH← 00h so AX=0001h)

41. Rotate AX contents in left directions by 4 bits. (AX=0010h)

42. Add the contents of AX and CX (CX=CX+AX so CX=7510h)

43. Read a fourth digit in AL register through keyboard AL=34h

44. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number. AL=04h

45. Move contents of AH with 00h. (AH← 00h so AX=0004h)

46. Add the contents of AX and BX (CX=CX+AX so CX=7514h)

47. Compare accepted choice from AL with 02h

48. If zero flag is set then goto step no 69 otherwise goto step no 49

49. Add the contents of BX and CX(BX=BX+CX so BX=9A24h)

50. Preserve the result in temporary variable as t of 16 bit (so t=9A24h)

51. Mask the first nibble by AND operation with number F000h (AND BX,f000h so BX=9000h)

52. Rotate the BX contents right by 12(in decimal so BX=0009h)

53. Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BX=0039h)

54. Move the contents of BL to DL and display it on the screen

55. Move result from temporary variable t to BX again (Now BX=9A24h)

56. Mask the second nibble by AND operation with number 0F00h (AND BX,0F00h so BX=0A00h)

57. Rotate the contents of BX to right by 8(in decimal)

58. Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BX=0041h(ASCII hex value of 'A'))

59. Move the contents of BL to DL and display it on the screen.

60. Move result from temporary variable to BX again (Now AX=9A24h)

61. Mask the third nibble by AND operation with number 00F0h (AND BX,00F0h so BX=0020h)

62. Rotate the contents of BX to right by 4(in decimal)

63. Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BX=00032h)

64. Move the contents of BL to DL and display it on the screen

65. Move result back from temporary variable to BX again (Now BX=9A24h)

66. Mask the fourth nibble by AND operation with number 000Fh (AND BX,000fh so BX=0004h)

67. Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BX=0004h)

68. Move the contents of BL to DL and display it on the screen.

69. Subtract the contents of CX from BX(BX←BX-CX so BX=AFFCh)

70. Preserve the result in temporary variable as t of 16 bit (so t=AFFCh)

71. Mask the first nibble by AND operation with number F000h (AND BX,F000h so BX=A000h)

72. Rotate the BX contents right by 12(in decimal so BX=000Ah)

73. Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BX=0041h (i.e. ASCII hex value for 'A'))

74. Move the contents of BL to DL and display it on the screen

75. Move result from temporary variable t to BX again (Now BX=AFFCh)

76. Mask the second nibble by AND operation with number 0f00h (AND BX,0F00h so BX=0F00h)

77. Rotate the contents of BX to right by 8(in decimal)

78. Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BX=0046h(i.e ASCII hex value of 'F'))

79. Move the contents of BL to DL and display it on the screen.

80. Move result from temporary variable to BX again (Now BX=AFFCh)

81. Mask the third nibble by AND operation with number 00F0h (AND BX,00F0h so BX=00F0h)

82. Rotate the contents of BX to right by 4(in decimal)

83. Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BX=0046h (i.e ASCII hex value of 'F'))

84. Move the contents of BL to DL and display it on the screen

85. Move result back from temporary variable to BX again (Now BX=AFFCh)

86. Mask the fourth nibble by AND operation with number 000fh (AND AX,000Fh so AX=000Ch)

87. Call Output procedure with BL register to make a digit back in ASCII hexadecimal range (BX=0043h(i.e ASCII hex value of 'C))

88. Move the contents of BL to DL and display it on the screen.

89. Stop

Algorithm for Input procedure :(To accept input from 0 to f)

6. Compare the contents of AL with 41h(Small case)
7. Jump to step no 4 if carry flag is set
8. Sub 07h to AL register
9. Sub 30h to AL register
10. Return.

Algorithm for Output procedure:

6. Compare the contents of BL with 0Ah
7. Jump to step no 4 if carry flag is set
8. Add 07h to AL register
9. Add 30h to AL register
10. Return.

Note:

While masking F or f is not case sensitive. But in input procedure 41h number is considered for comparison because 41h is ASCII hex value for 'A'. In output procedure '0A' is considered not 'a' is considered as small case a has 61h ASCII hex value.So this input and output procedure are applicable for only capital 'A' to 'F'

Similarly, the procedures for 8-bit subtraction, multiplication and division can be extended to 16-bit operations

Experiment No 4

Objective:

Write a menu driven assembly language program to accept ,display the string, find the length of string, reverse the string, check the string is palindrome or not and scan the string to check whether a accepted letter is present or not

Prerequisite:

TASM assembler

Description:

There are several string instructions, used to move large blocks of data or stings and perform operations on strings.

- Memory string source - DS:SI
- Memory string destination - ES:DI
- String can be processed in forward and reverse direction.
- If DF=1 auto decrement for SI and DI
- If DF=0 auto increment for SI and DI

Algorithm:

1. Start
2. Initialize data segment through AX register in the DS register.
3. If choice =1 then goto step no.4 (accept the string)else goto the step no.14
4. Display the message "Enter the string"
5. Initialize the SI with 1000h(source index)
6. Initialize the DI with 2000h(destination index)
7. Accept the character through keyboard(AL=52h i.e ASCII hex value of 'm')
8. Move the AL contents to a location pointed by SI and DI.
9. Increment the CL register contents by 1

10. Increment SI and DI by 1

11. Repeat the step 7 to 10 till Enter key get pressed(i.e AL=0Dh ASCII hex value for enter key used to detect the end of the string.)

12. Decrement the CL by 1 (to avoid the enter key as a part of the string to obtain correct length value)

13. Preserve the length in temporary variable say count from CL

14. If choice = 2 then goto step no.15 (display the string)else goto step no. 21

15. Initialize SI again with 1000h (string source)

16. Move the content of location pointed by SI to DL

17. Display the character on the screen

18. Increment SI by 1and decrement the CL by 1

19. Repeat the step from 15 to 18 till Zero flag will come set (i.e CL reaches to zero).

20. Load the String length from count to CL register back.

21. If choice=3 (display the length of string) then goto step no. 22 else goto step no. 24

22. Move the contents of CL to AL and add 30h to AL

23. Move the AL contents to DL and display the length of string.

24. If choice=4 (Reverse the string) then goto step no. 24 else goto step no.30

25. Add CX (e.g CX=0005)to SI to make SI to point to the last letter of String.

26. Decrement to SI by 1 as SI will start from 0 index (e.x 1000 to 10004 are the locations if string is of 5 letters)

27. Move the letter pointed by SI to DL and display it on the screen

28. Decrement the SI by 1 and decrement the CL by 1

29. Repeat step 25 to 29 till CL reaches to zero if zero flag get set.

30. If choice = 5 then goto step no. 31 else goto the step no.43

31. Initialize SI with 1000h again

32. Initialize DI with 2000h again

33. Load CL with original length of string from count

34. Add DI with CX (e.x CX=0005)so DI will point to the last letter of string

35. Move the letter from location pointed by SI to AL

36. Move the letter from location pointed by DI to BL

37. Compare the AL and BL if zero flag is not set then goto step no. 38 else goto step no 39

38. Display the string is not palindrome.

39. Increment the SI by 1 and Decrement DI by 1

40. Decrement CL by 1.

41. Repeat step no. 34 to 38 till CL reaches to zero

42. If zero flag is set then display the string is palindrome.

43. If choice=6 then goto step no. 44 else goto step no 49

44. Accept the letter to be searched in AL.

45. Load CL with original length of string from count

46. Initialize the DI with 2000h

47. Use REPNE SCASB instruction this instruction scans the string for a letter stops when it finds the first occurrence of a letter. It decrements CX also by 1 for every scan. When this instruction stops CX will contain value as position-1 value.

48. Obtain the position of a letter by subtracting total length-CX value and display it on the screen.

49. Stop.

Experiment No 5

Objective:

Write an assembly language program to display the contents of 16 bit flag register.

Prerequisite:

TASM assembler

Description:

To display the contents of flag register pushf and pop instructions are used. Each bit of flag register is then masked off with 1 and all 0's (i.e. 1000 0000 0000 0000(16 bit) → 8000h) and based on the result of masking either 0 (30h) or 1 (31h) is get displayed on the screen. Each bit of the above 16 bit number gets shifted in right direction by 1 position before masking to obtain the next bit position of flag register. This whole procedure gets repeated 16 times.

Algorithm

1. Start
2. Initialize data segment through AX register in the DS register.
3. Display the flag bit names as "X X X X O D I T SF ZF x AF X PF X CF "
4. Push the contents of flag register to a stack
5. Pop the contents of stack to register to any 16 bit register (say BX =0000 0100 1000 1001)
6. Move the contents of BX to temporary variable say t
7. Move the 8000h number to AX.(AX← 8000h)
8. Move the count as 16(in decimal) to CX register (as 16 bit flag register)
9. Move the contents of temporary variable t to BX.
10. And the contents of BX and AX.

11. If zero flag is set then goto the step no 14 otherwise goto step no. 12

12. Move the 31h to DL register.

13. Make the unconditional jump to a step no. 15

14. Move the 30h to DL register.

15. Preserve the (8000h) number from AX in t1 temporary variable. (As while displaying 30h or 31 h AH register get modified as 02h function is moved of INT 21h).

16. Display the contents of DL register.

17. Move the contents of t1 to AX register back (As while displaying 30h or 31 h AH register get modified as 02h function is moved of INT 21h).

18. Rotate the contents of AX by 1 positions in right direction.

19. Repeat step no 5 to 17 till count CX reaches to 0.

20. Stop.

Experiment No 6

Objective:

Write an assembly language program to sort numbers in ascending/ descending order.

Prerequisite:

TASM assembler

TASM Program for ascending order

```
DATA SEGMENT
MSG1 DB 10,13,"ARRAY LIMIT IS 10: $"
MSG2 DB 10,13,"ENTER THE NUMBER: $"
MSG3 DB 10,13,"THE SORTED ARRAY IS: $"
MSG4 DB 10,13,"$"
COUNT DB ?
TEMP DB ?
DATA ENDS

CODE SEGMENT
ASSUME DS:DATA,CS:CODE
START:
MOV AX,DATA
MOV DS,AX

LEA DX,MSG1
MOV AH,09H
INT 21H
```

```
MOV SI,4000H
MOV CX,10

LABEL1:

LEA DX,MSG2
MOV AH,09H
INT 21H

MOV AH,01H
INT 21H
CALL INPUT
ROL AL,04H
MOV BL,AL

MOV AH,01H
INT 21H
CALL INPUT
MOV AH,00H
ADD BL,AL

MOV [SI],BL
INC SI
LEA DX,MSG4
MOV AH,09H
INT 21H

LOOP LABEL1

MOV CX,9
mov bx,0000h

LABEL2:

MOV SI,4000H
MOV BL,[SI]
INC SI
MOV BH,9

LABEL3:

DEC BH
CMP BL,[SI]
JC LABEL4
MOV DL,BL
MOV BL,[SI]
DEC SI
MOV [SI],BL
INC SI
MOV [SI],DL
```

```
INT 21H

LOOP LABEL1

MOV CX,9
mov bx,0000h

 LABEL2:

 MOV SI,4000H
 MOV BL,[SI]
 INC SI
 MOV BH,9

 LABEL3:

 DEC BH
 CMP BL,[SI]
 JC LABEL4
 MOV DL,BL
 MOV BL,[SI]
 DEC SI
 MOV [SI],BL
 INC SI
 MOV [SI],DL
```

```
LABEL4:
MOV BL,[SI]
INC SI
CMP BH,00H
JNZ LABEL3

LOOP LABEL2

MOV SI,4000H
MOV CX,10

d:
LEA DX,MSG3
MOV AH,09H
INT 21H

MOV BL,00H
MOV BL,[SI]
AND BL,0F0H
ROR BL,04H
CALL OUTPUT
MOV BL,[SI]
AND BL,0FH
CALL OUTPUT
INC SI
LOOP d

MOV AH,4CH
INT 21H

INPUT PROC
    CMP AL,41H
    JC LABEL6
    SUB AL,07H
    LABEL6:
    SUB AL,30H

    RET
    ENDP

OUTPUT PROC
    CMP BL,0AH
    JC LABEL7
    ADD BL,07H
    LABEL7:
    ADD BL,30H
    MOV DL,BL
    MOV AH,02H
    INT 21H

    RET
    ENDP
```

```
MOV AH,4CH
INT 21H

INPUT PROC
        CMP AL,41H
        JC LABEL6
        SUB AL,07H
        LABEL6:
        SUB AL,30H

        RET
        ENDP

        OUTPUT PROC
            CMP BL,0AH
            JC LABEL7
            ADD BL,07H
            LABEL7:
            ADD BL,30H
            MOV DL,BL
            MOV AH,02H
            INT 21H

              RET
              ENDP
```

CODE ENDS

END START

Note: similar program can be written to arrange the integers in descending order

Experiment No 7

Objective:

Write a assembly language code to find the minimum/maximum number from an array

Prerequisite:

TASM assembler

Algorithm: (minimum no)

1. Start
2. Initialize data segment through AX register in the DS register.
3. Initialize the SI to 5000h
4. Initialize total elements of array as a count in CX(e.g 0005h)
5. Preserve the above count in c temporary variable.
6. Display the message as "Enter an array elements"
7. Read first digit in AL register through keyboard (e.g. AL=31h)
8. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number.AL=01h
9. Move AL contents to BL
10. Rotate BL contents by 4 in left direction.
11. Read second digit in AL register through keyboard (e.g AL=32h)
12. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number.AL=02h
13. Add BL and AL contents (BL← BL+AL)
14. Store the BL (current accepted number) to location pointed by SI
15. Increment SI by 1 to point to next location for the next number
16. Repeat step no. 7 to 15 till CX count reaches to 0.

17. Initialize SI again to 5000h and CX also with total number of elements.

18. Initialize AL with first element pointed by SI for the next comparison

19. Compare number pointed by SI from an array with AL register

20. If carry is generated (i.e. if number in AL > number pointed by SI) then goto step no. 22 else goto step no. 21

21. Make a unconditional jump to step no. 23

22. Move number pointed by SI to AL

23. Incremented SI by 1

24. Decrement CX by 1

25. Compare CX with 0000h (i.e. Repeat step no.19 to 25 till all numbers of array are not covered for the comparison)

26. If Zero flag is not set then jump to step no.19

27. Finally minimum number will be available in AL register.

28. Display the contents of AL register.

29. Stop.

Algorithm: (maximum no)

1. Start
2. Initialize data segment through AX register in the DS register.
3. Initialize the SI to 2000h
4. Initialize total elements of array as a count in CX(e.g 0005h)
5. Preserve the above count in c temporary variable.
6. Display the message as "Enter an array elements"
7. Read first digit in AL register through keyboard (e.g. AL=31h)

8. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number.AL=01h

9. Move AL contents to BL

10. Rotate BL contents by 4 in left direction.

11. Read second digit in AL register through keyboard (e.g AL=32h)

12. Call Input procedure to make a number from ASCII hexadecimal to a normal hexadecimal number.AL=02h

13. Add BL and AL contents (BL← BL+AL)

14. Store the BL (current accepted number) to location pointed by SI

15. Increment SI by 1 to point to next location for the next number

16. Repeat step no. 7 to 15 till CX count reaches to 0.

17. Initialize SI again to 2000h and CX also with total number of elements.

18. Initialize AL with first element pointed by SI for the next comparison

19. Compare number pointed by SI from an array with AL register

20. If carry is not generated (i.e. if number in AL < number pointed by SI) then goto step no. 22 else goto step no. 21

21. Make a unconditional jump to step no. 23

22. Move number pointed by SI to AL

23. Incremented SI by 1

24. Decrement CX by 1

25. Compare CX with 0000h (i.e. Repeat step no.19 to 25 till all numbers of array are not covered for the comparison)

26. If Zero flag is not set then jump to step no.19

27. Finally maximum number will be available in AL register.

Experiment No 8

Introduction to mixed language programming

Objective:

Write a mixed language code to add 2 numbers.

Prerequisite:

Turbo C++ Compiler

Description:

Inline Assembly:

Inline assembly means the mixed code gets compiled, assembled, loaded and linked using environment called as IDE (Integrated Development Environment). Turbo C++ has IDE environment.

ASM keyword

The **asm** keyword invokes the inline assembler and can appear wherever a C or C++ statement is legal. It cannot appear by itself. It must be followed by an assembly instruction, a group of instructions enclosed in braces, or, at the very least, an empty pair of braces. The term "**asm** block" here refers to any instruction or group of instructions, whether or not in braces.

Algorithm:

1. Start
2. Declare 3 variables as a,b,sum as integer
3. Read the values of a,b
4. Move contents of a to AX
5. Move the contents b to BX
6. Add AX and BX (AX← AX+BX)
7. Move result to sum from AX register
8. Display the sum.
9. Stop

Objective:

Write a mixed language code to calculate GCD/LCM of 2 numbers.

Prerequisite:

Turbo C++ Compiler

Algorithm: (GCD)

1. Start
2. Declare 3 numbers a,b,gcd as integers.
3. Move the a and b value to AX and BX respectively
4. Repeat till AX!=BX step no 5 to 7 otherwise goto 8
5. If AX > BX then goto step no 6 else goto step no 7
6. Subtract AX← AX-BX
7. Subtract BX←BX-AX
8. Move AX or BX to gcd variable
9. Display the gcd.
10. Stop

Algorithm: (LCM)

1. Start
2. Declare 3 numbers a, b, lcm as integers.
3. Move the a and b value to AX and BX respectively
4. Repeat step no 5 till AX/BX division not gives the remainder = 0
5. Add AX←AX+BX
6. As remainder 0 gets obtained AX value is LCM value. Move AX value to lcm
7. Display the lcm value.
8. Stop.

Experiment No 9

Objective:

Write a menu driven mixed language code to increment , decrement the size of cursor and disable the cursor.

Prerequisite:

Turbo C++ Compiler

Description:

To increment/ decrement the size of cursor INT 10H interrupt functions are used. These functions are called using CH and CL register as parameters. Where

CH =Minimum value of cursor (00h)

CL=Maximum limit of cursor (count)

01h function of INT 10H interrupt is used to increment and decrement the size of cursor. To increment cursor size count value gets incremented and moved to CL register and to decrement cursor size count value gets decremented and moved to CL register.

Disable the cursor, CL must be loaded with 20h and then call 01h function of INT 10h DOS interrupt.

Algorithm:

1. Start
2. Declare count value as char type
3. If choice =1 then goto step no 4 else goto step no 10
4. Check count >=4 then goto step no 5 else step no 6
5. Display the msg as "Reached maxium limit"
6. Increment count value by 1
7. Move count value to CL and move 00h to CH register.
8. Move 01h to AH register
9. Call function INT 10H DOS interrupt.
10. If choice =2 then goto step no. 11 else goto step no 17

11. Check count <=0 then goto step no 12 else goto step no 13
12. Display the message as "Reached minimum limit"
13. Decrement count by 1
14. Move count value to CL and move 00h to CH register.
15. Move 01h to AH register
16. Call function INT 10h DOS interrupt
17. If choice =3 then goto step no 18 else goto step no 21
18. Move 20h to CH register
19. Move 01h to AH register
20. Call function INT 10h interrupt.
21. Stop

Experiment No 10

Objective:

To rotate a stepper motor in clockwise and anticlockwise direction

Prerequisite:

8086 kit, stepper motor kit

Starting 8086kit:

Connect keyboard and power supply to 8086 kit and switch ON

Assemble (A [address]):

Default segment is CS. It will start at the current value of IP.
When done with assembly, press Enter and the hyphen prompt will return
a. Assembles mnemonics into the memory directly.
b. Creates machine executable code in memory beginning at cs:0100 [or specified address] from entered 8086 assembly instructions.
Process of assembly stops after you enter an empty line.

Go (G [=address] [addresses]). Executes current program starting at IP if [=address] is not specified, or at [=address] if specified. [addresses] is a space-delimited list of breakpoints.

The Go command (G), when used without a parameter, causes DEBUG to copy the contents of the register variables to the actual CPU registers and effectively jump to the instruction at CS:IP, giving full control to the program being debugged. The optional address parameter can be used to specify the starting address. The optional breakpoint list can be used to specify up to 10 temporary breakpoint addresses. To differentiate the address parameter from the breakpoint list, the address parameter must be preceded with an "=". Note that the breakpoint addresses must coincide with the start of a valid instruction. To resume execution after a breakpoint, use the Go command without a parameter.

Syntax: G [=address]

Expand (E):

Press CR key to authorise the command.

This command supports assembler

You can invoke any of the following by typing corresponding letter followed by CR key

A – Assemble

C - Compare memory

D – Hex dump

E – Examine/Modify

F – Fill memory

H – Hexadecimal Arithmatic

M – Move to Memory

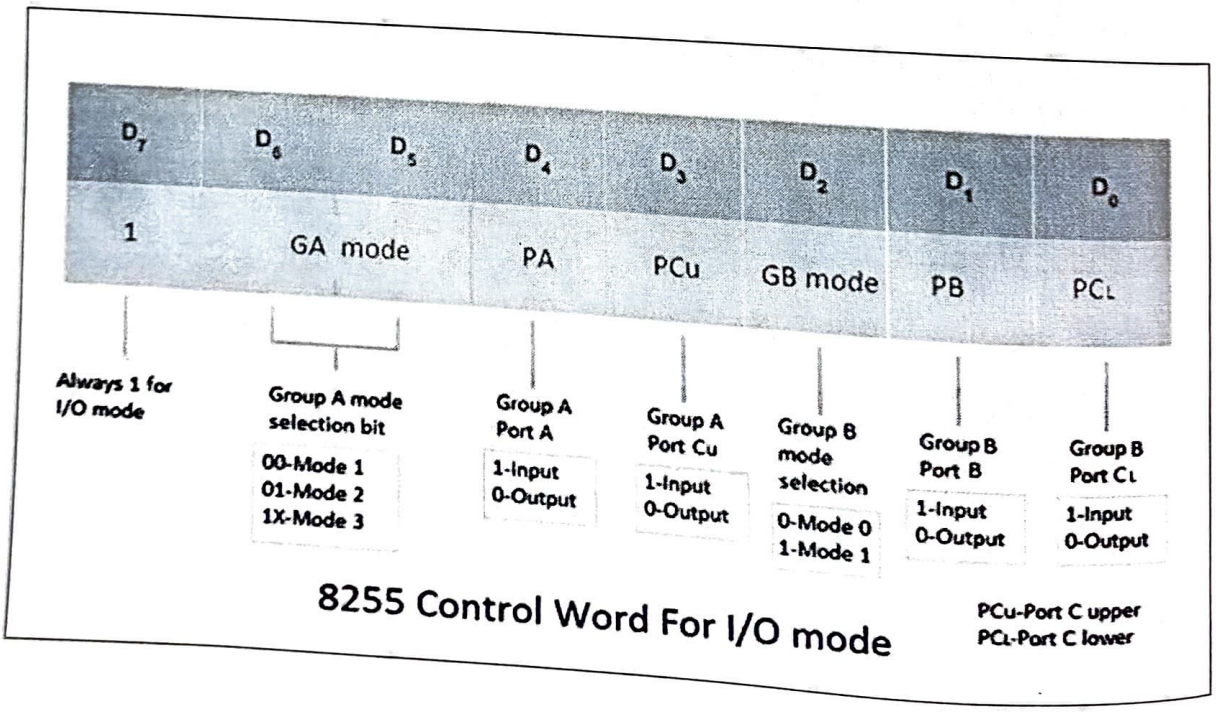
Q – Return to Command Mode of system monitor

R – Display Register

S - Search string

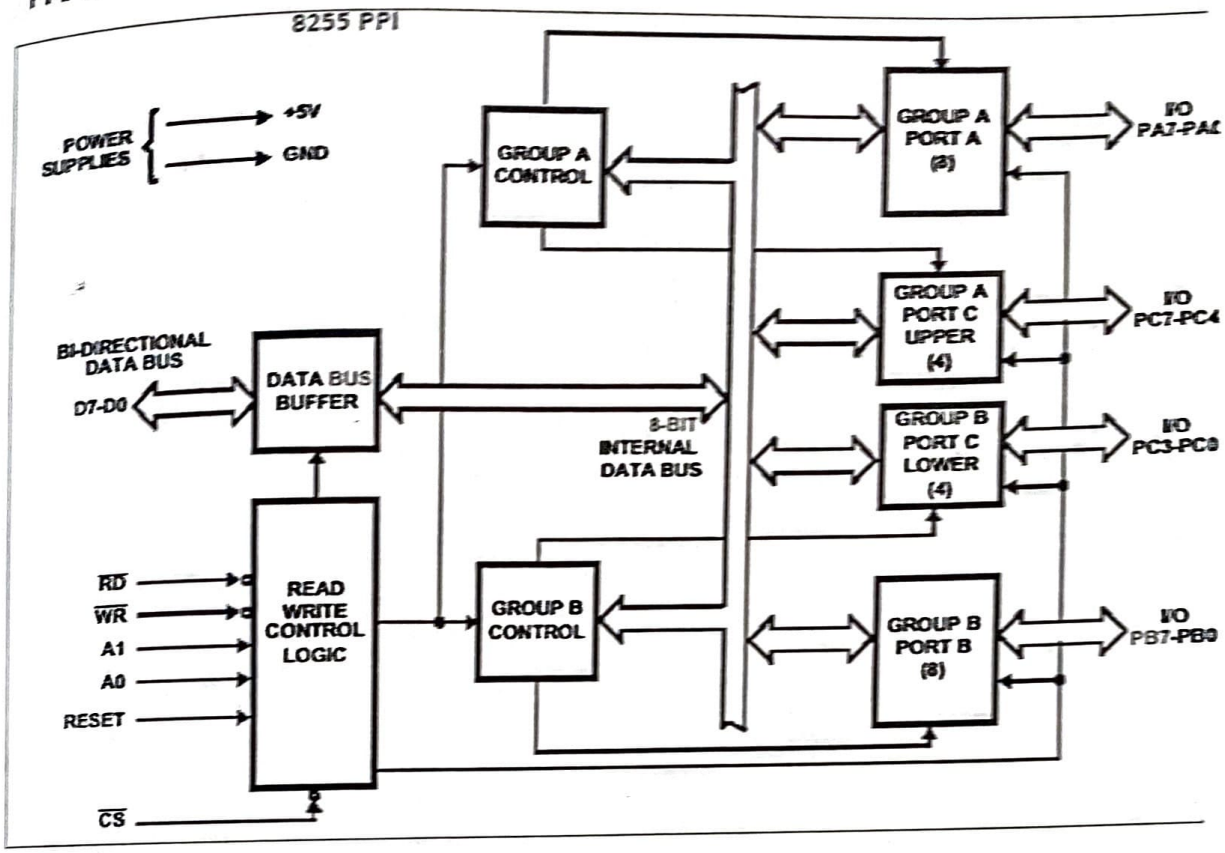
U – Unassemble

8255 I/O operating modes



8255 Control Word For I/O mode

PPI 8255 block diagram



The port address are as follows

Port A – 8000H

Port B – 8002H

Port C – 8004H

CWR - 8006H

Directly output on the port the phase code for stepper and rotate the motor in clockwise direction(06, 0C, 09, 03) and in reverse order to rotate in anticlockwise direction