Experiment 04: Finding Negative Numbers from a given array

Learning Objective: Students should be able to Identify negative numbers from a given sign array using Assembly language.

Tools: TASM/MASM

Theory:

In computing, signed number representations are required to encode negative numbers in binary number systems. The early days of digital computing were marked by a lot of competing ideas about both hardware technology and mathematics technology (numbering systems). One of the great debates was the format of negative numbers, with some of the era's most expert people having very strong and different opinions. One camp supported two's complement, the system that is dominant today. Another camp supported ones' complement, where any positive value is made into its negative equivalent by inverting all of the bits in a word. A third group supported "sign & magnitude" (sign–magnitude), where a value is changed from positive to negative simply by toggling the word's sign (high–order) bit. In mathematics, negative numbers in any base are represented by prefixing them with a − sign. However, in computer hardware, numbers are represented in bit vectors only, without extra symbols. The four best–known methods of extending the binary numeral system to represent signed numbers are sign–and–magnitude, ones' complement, two's complement, and excess–K. Some of the alternative methods use implicit instead of explicit signs, such as negative binary, using the base −2. Corresponding methods can be devised for other bases,

whether positive, negative, fractional, or other elaborations on such themes. There is no definitive criterion by which any of the representations is universally superior. The representation used in most current computing devices is two's complement. To check whether the number is negative, perform AND operation with 80H, if the number is negative, the result will be Non-zero (i.e. MSB is 1) and if the number is positive then the result will be Zero (i.e. MSB is 0).

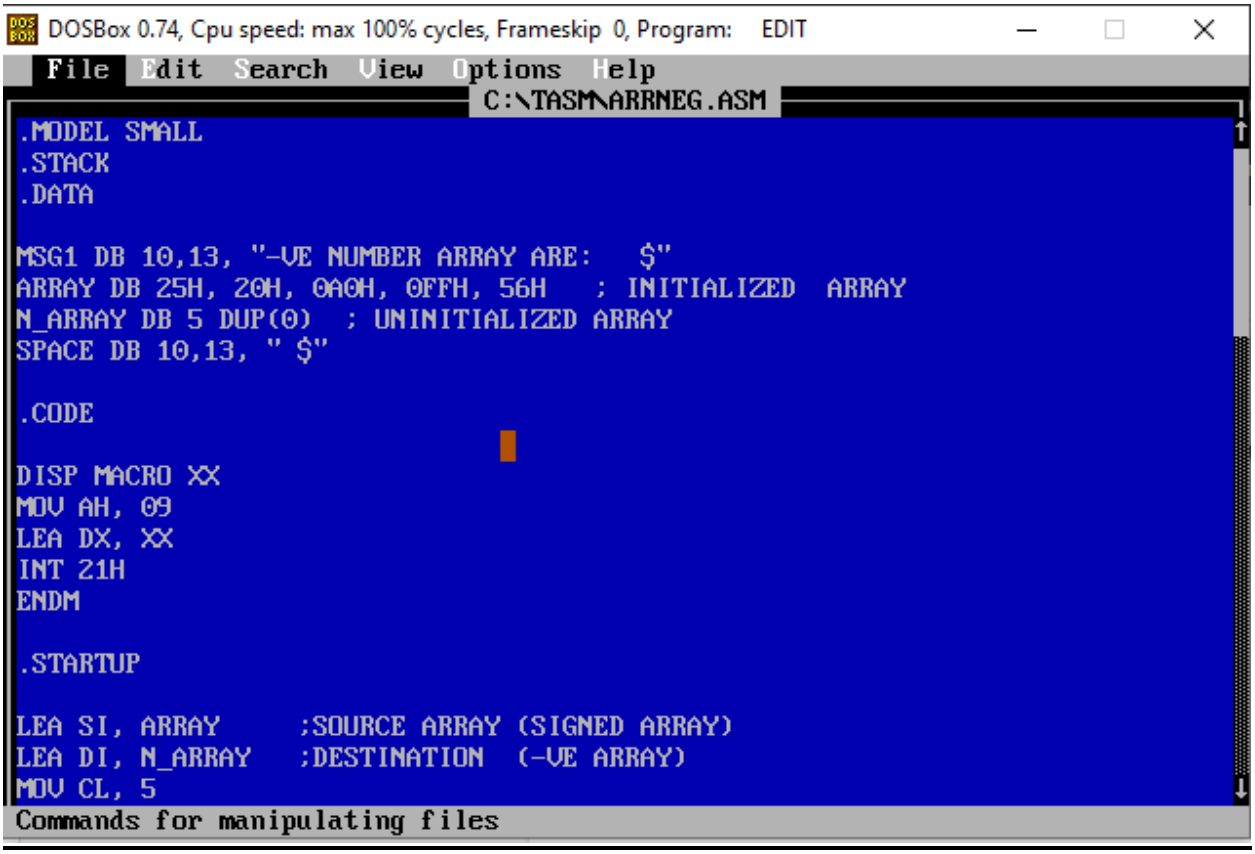
5. Procedure/Algorithm:

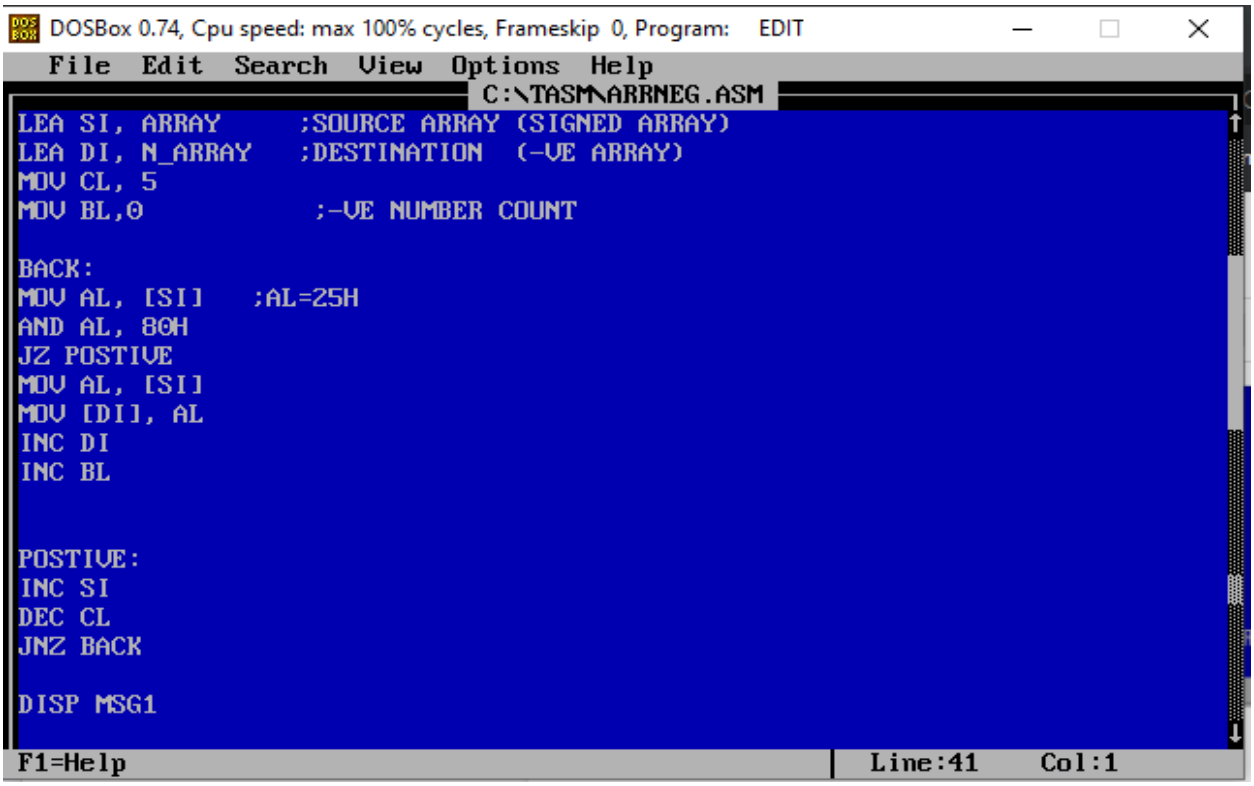
Algorithm:

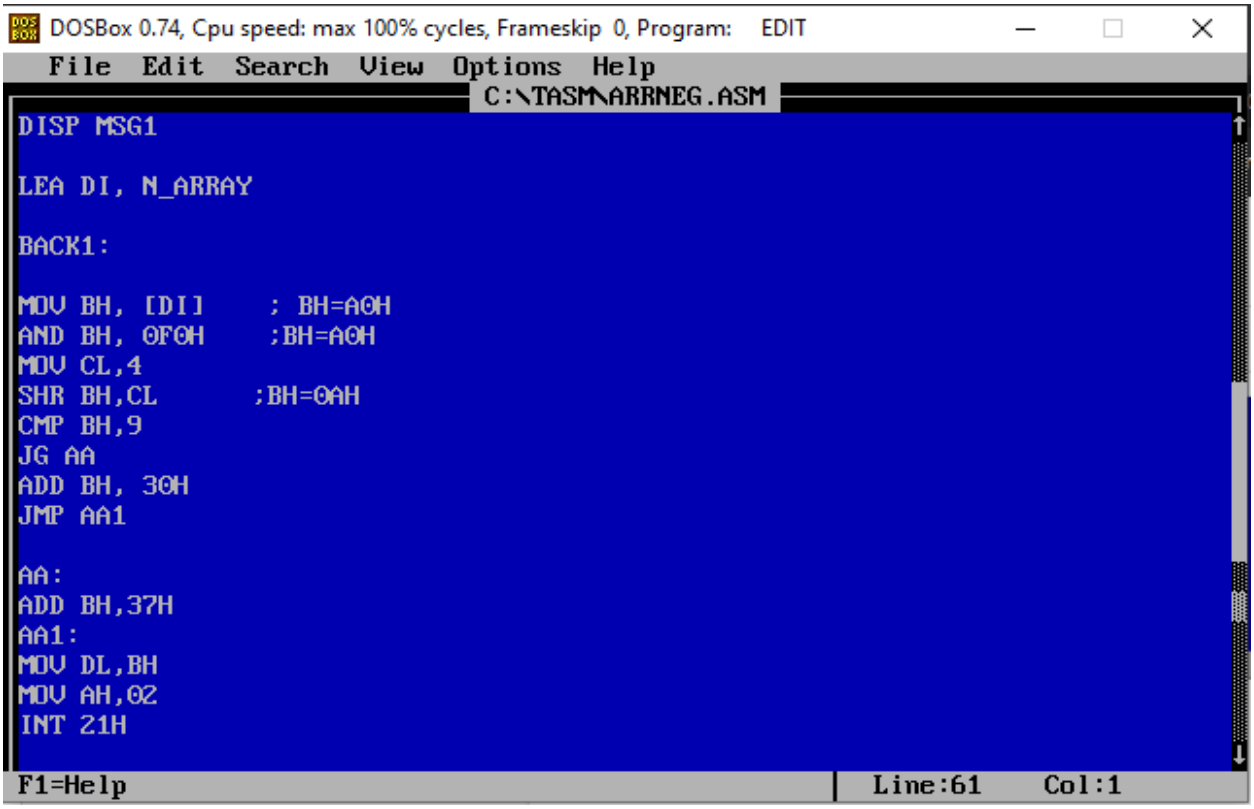
1. Initialize index register with the offset of an array of signed numbers
2. Initialize CX with array element count
3. Initialize negative number count to zero
4. Perform MSB test of the array element
5. If the set jump to step 6
6. Increment negative number count and continue
8. Point index register to the next element
9. Decrement the array element count from CX, if not zero jump to step 4, else continue
10. Display Negative number message and then display negative number count
12. EXIT

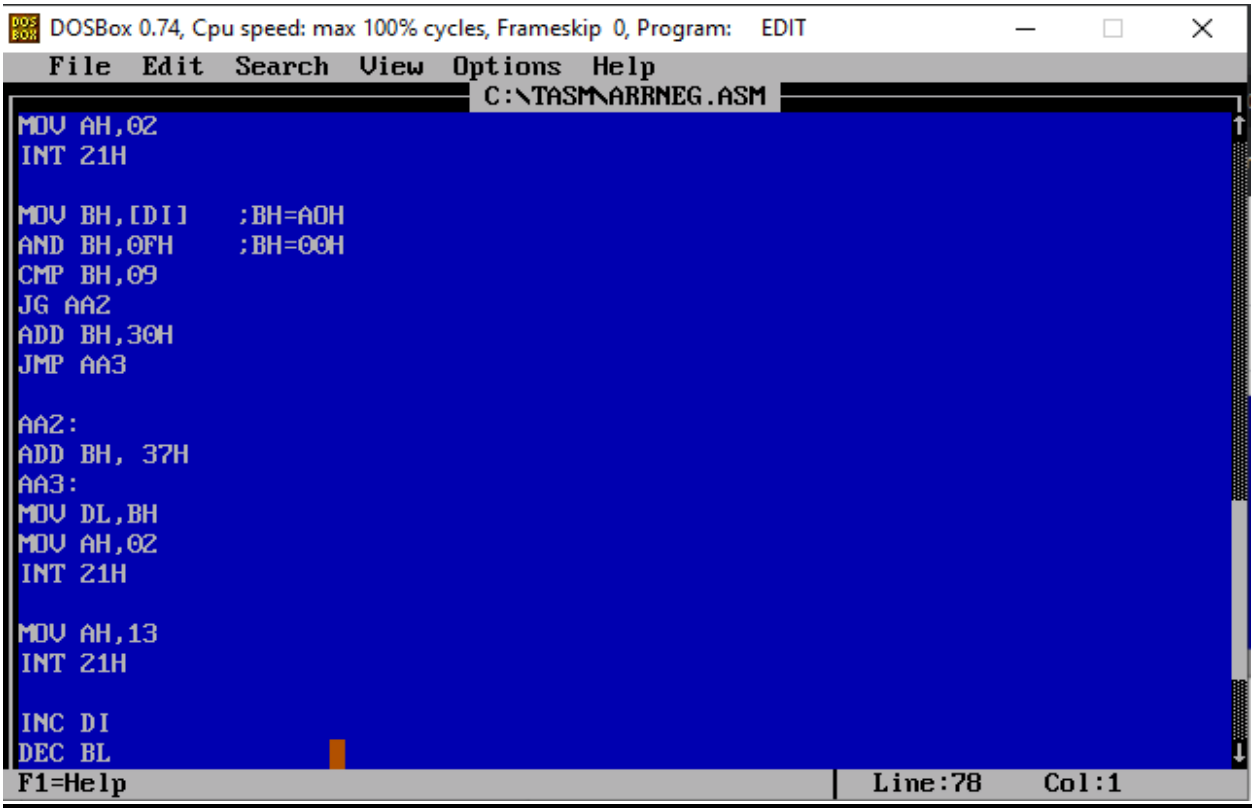
Application: Use of array in the Assembly Language programming to write a modular program.

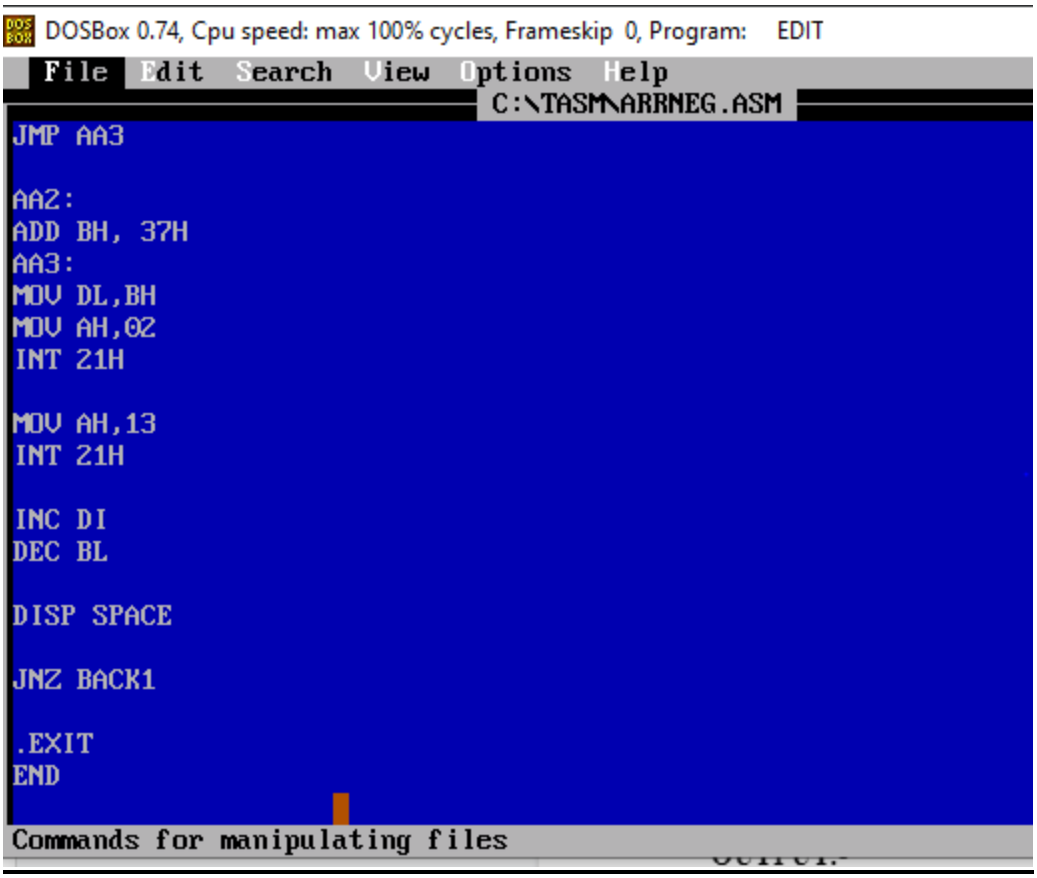
Design:


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\TASM\ARRNEG.ASM
.MODEL SMALL
.STACK
.DATA
MSG1 DB 10,13, "-VE NUMBER ARRAY ARE: $"
ARRAY DB 25H, 20H, 0A0H, 0FFH, 56H ; INITIALIZED ARRAY
N_ARRAY DB 5 DUP(0) ; UNINITIALIZED ARRAY
SPACE DB 10,13, " $"
.CODE
DISP MACRO XX
MOV AH, 09
LEA DX, XX
INT 21H
ENDM
.STARTUP
LEA SI, ARRAY ;SOURCE ARRAY (SIGNED ARRAY)
LEA DI, N_ARRAY ;DESTINATION (-VE ARRAY)
MOV CL, 5
Commands for manipulating files

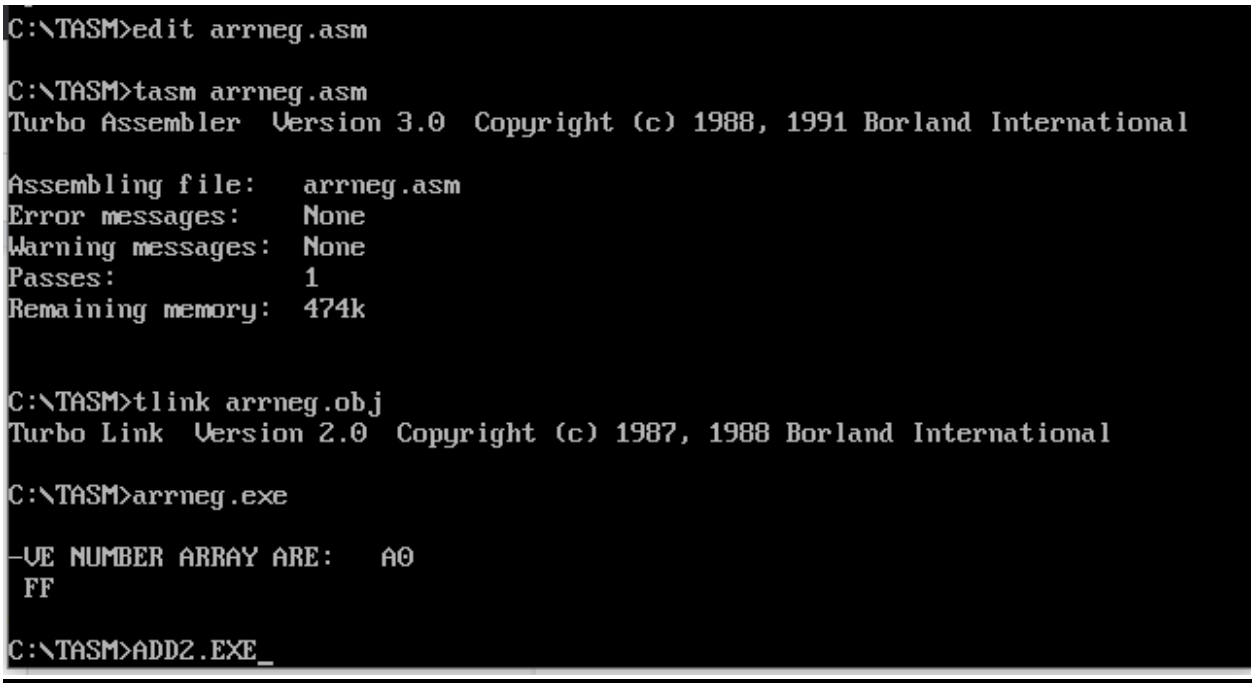

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\TASM\ARRNEG.ASM
LEA SI, ARRAY ;SOURCE ARRAY (SIGNED ARRAY)
LEA DI, N_ARRAY ;DESTINATION (-VE ARRAY)
MOV CL, 5
MOV BL,0 ;-VE NUMBER COUNT
BACK:
MOV AL, [SI] ;AL=25H
AND AL, 80H
JZ POSTIVE
MOV AL, [SI]
MOV [DI], AL
INC DI
INC BL
POSTIVE:
INC SI
DEC CL
JNZ BACK
DISP MSG1
F1=Help
Line:41 Col:1


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\TASM\ARRNEG.ASM
DISP MSG1
LEA DI, N_ARRAY
BACK1:
MOV BH, [DI] ; BH=A0H
AND BH, 0F0H ;BH=A0H
MOV CL,4
SHR BH,CL ;BH=0AH
CMP BH,9
JG AA
ADD BH, 30H
JMP AA1
AA:
ADD BH,37H
AA1:
MOV DL,BH
MOV AH,02
INT 21H
F1=Help
Line:61 Col:1


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\TASM\ARRNEG.ASM
MOV AH,02
INT 21H
MOV BH,[DI] ;BH=A0H
AND BH,0FH ;BH=00H
CMP BH,09
JG AA2
ADD BH,30H
JMP AA3
AA2:
ADD BH, 37H
AA3:
MOV DL,BH
MOV AH,02
INT 21H
MOV AH,13
INT 21H
INC DI
DEC BL
F1=Help
Line:78 Col:1


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: EDIT
File Edit Search View Options Help
C:\TASM\ARRNEG.ASM
JMP AA3
AA2:
ADD BH, 37H
AA3:
MOV DL,BH
MOV AH,02
INT 21H
MOV AH,13
INT 21H
INC DI
DEC BL
DISP SPACE
JNZ BACK1
.EXIT
END
Commands for manipulating files

OUTPUT:-


C:\TASM>edit arrneg.asm
C:\TASM>tasm arrneg.asm
Turbo Assembler Version 3.0 Copyright (c) 1988, 1991 Borland International
Assembling file: arrneg.asm
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 474k
C:\TASM>tlink arrneg.obj
Turbo Link Version 2.0 Copyright (c) 1987, 1988 Borland International
C:\TASM>arrneg.exe
-VE NUMBER ARRAY ARE: A0
FF
C:\TASM>ADD2.EXE_

Result and Discussion:

- **We develop a code to identify the negative number from a given sign number array for a 16 bits number.**
- **We understand how to use an array signed and unsigned create.**
- **We understand and use macro processors to display messages.**

Learning Outcomes: The student should have the ability to

LO1: Describe the string addressing mode.
LO2: Use of string instructions in the program to perform different string operations.
LO3: Write a program using an array to find negative numbers from a given signed array.

Course Outcomes: Upon completion of the course students will be able to make use of instructions of 8086 to build assembly and Mixed language programs.

Conclusion: We Understand the Array and its application in ALP.

Viva Questions:

1. How negative Numbers are represented in Computer systems?
2. Explain how the array is defined in ALP.
3. Explain the procedure to find negative numbers from the given array.

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				