EXPERIMENT – 4

Program – 1

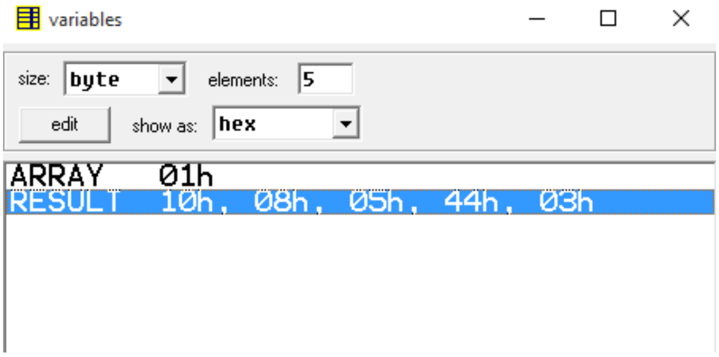
Aim: Divide 8-bit number stored in memory locations 4009H by data stored at memory location 4001H & store result of division at memory location 400AH. (Use Data Set -4)

Code:

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here

.model prog-4.1

.data
   array db 1,2,3,4,5
   result db 5 DUP(?)

.code

MOV DX, @data

MOV DS,DX

MOV CX, 5

MOV BX, 0


next:
   mov AL, 10h ; 10 will be divided by the array

   DIV array[bx] ; QUOTIENT SAVED IN al, AND REMAINDER IN AH, DIVIDEND SAVED IN AX

   mov result[bx], al

   inc bx

LOOP NEXT


Ret
```

Input/Output:

Input: array - 1,2,3,4,5 ; al – 10h
Output: result – 10h, 08h, 05h, 44h, 03h



size: byte
elements: 5
show as: hex

```
ARRAY    01h
RESULT   10h, 08h, 05h, 44h, 03h
```

EXPERIMENT – 4

Program – 2

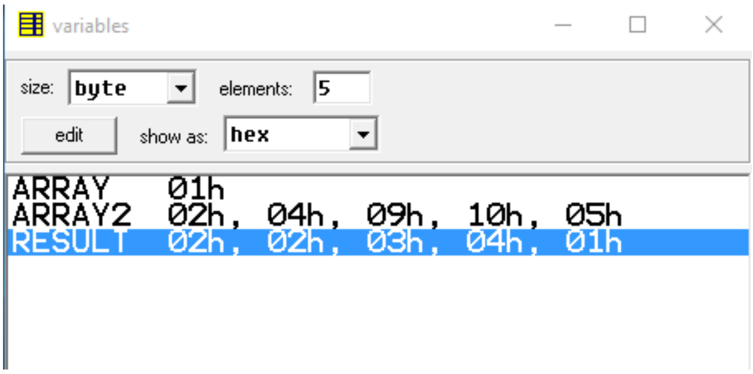
Aim: Divide 8-bit number stored in memory locations 4009H by data stored at memory location 4001H & store result of module operation at memory location 400AH. .(Use Data Set - 2,4)

Code:

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here

.model prog-4.2

.data
   array db 1,2,3,4,5
   array2 db 2,4,9,16,5
   result db 5 DUP(?)

.code

MOV DX, @data

MOV DS,DX

MOV CX, 5

MOV BX, 0


next:
   mov AL, array2[bx] ; array2 will be passed into the al everytime the loop rotates

   DIV array[bx] ; QUOTIENT SAVED IN al, AND REMAINDER IN AH, DIVIDEND
SAVED IN AX

   mov result[bx], al

   inc bx

LOOP NEXT
ret
```

Input/Output:

Input: array – 1, 2, 3, 4, 5; array2 – 02h, 04h, 09, 16h, 05h ;
Output: result – 02h, 02h, 03h, 04h, 01h



size: byte elements: 5
show as: hex

```
ARRAY    01h
ARRAY2   02h, 04h, 09h, 10h, 05h
RESULT   02h, 02h, 03h, 04h, 01h
```

EXPERIMENT – 4

Program – 3

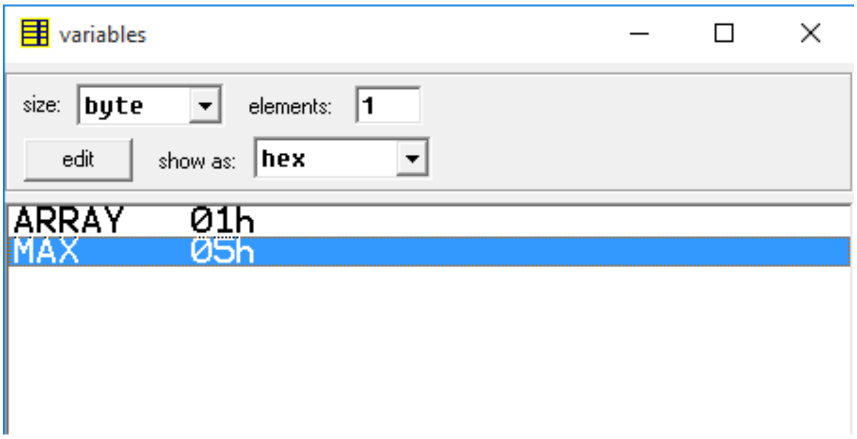
Aim: Write an assembly language program to find the largest number in an array.

Code:

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here

.model prog-4.3


.data
   array db 1,2,5,4,3
   max db 1 DUP(?)


.code
  MOV AX, @DATA

  MOV DS,AX

  MOV CX,5
  MOV BX,0
  MOV AL, ARRAY[BX] ; SET MAX=ARRAY[0]

  MOV MAX, AL

  CLC

  NEXT:
     MOV AL,MAX
     CMP AL,ARRAY[BX] ; PERFROMS SUBTRACTIONS
     INC BX
     JC MAXIMUM ; WHEN WE GET A LARGER NUMBER JUMP ONTO MAXIMUM
LABEL
  LOOP NEXT
  HLT

  MAXIMUM:
   MOV DL,ARRAY[BX]
   MOV MAX,DL    ; MAXIMUM IS SET TO THE LARGEST NUMBER
   DEC CX
   JMP NEXT
END


ret
```

Input/Output:

Input: array- 1,2,5,4,3
Output: max- 05h



size: byte
elements: 1
show as: hex

```
ARRAY    01h
MAX      05h
```

EXPERIMENT – 4

Program – 4

Aim: Write an assembly language program to count the numbers in an array (negative & positive)

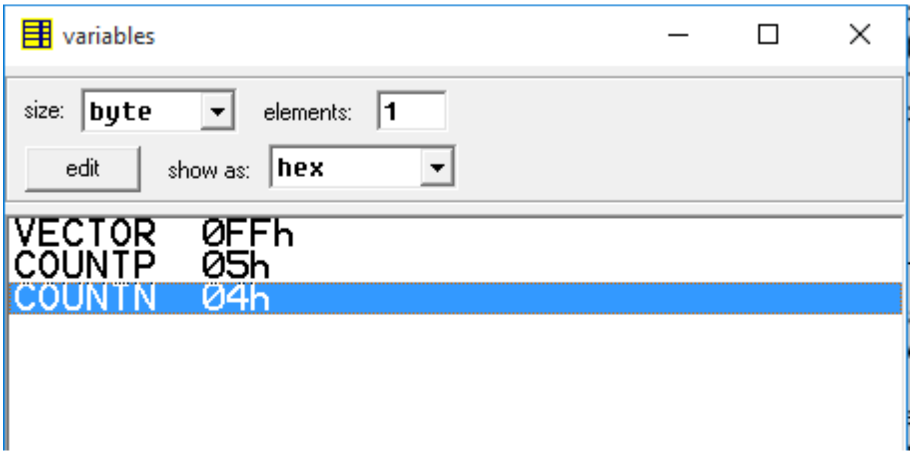
Code:

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
.MODEL PROG-4.4

.DATA
VECTOR DB -1,2,5,-3,6,-10,11,-6,12,'#'
COUNTP DB 0
COUNTN DB 0

.CODE
MOV AX,@DATA
MOV DS,AX

MOV AL,0
MOV BX,0
MOV CX,9

NEXT:
  CMP VECTOR[BX], 0H ; COMPARES FOR POSITIVE NUMBER
  JG POSITIVE ; CHECKS FOR SF (SIGN FLAG)
  INC BX
  INC COUNTN ; INCREMENTS NEGATIVE NUMBER

LOOP NEXT

HLT



POSITIVE:
     INC COUNTP  ; INCREMENTS POSITIVE VALUE
     INC BX
     DEC CX
```

```
    JNZ NEXT

END
ret
```

Input/Output:

Input: vector - 1,2,5,-3,6,-10,11,-6,12,'#'
Output: countp – 05h ; countn – 04h



```
VECTOR  0FFh
COUNTP  05h
COUNTN  04h
```

EXPERIMENT – 4

Program – 5

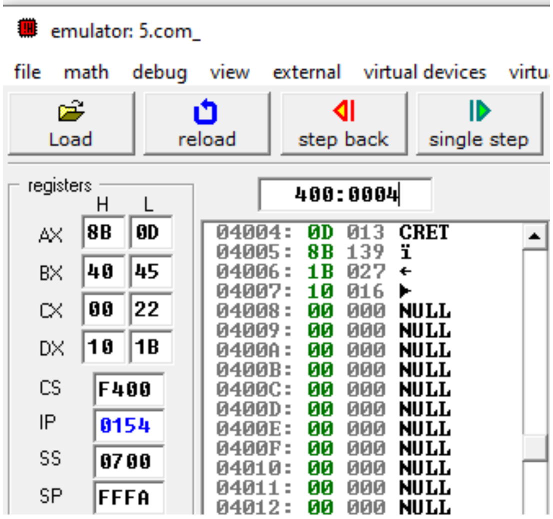
Aim: Write an assembly language program to multiply two 16-bit numbers in memory and store the result in memory.

Code:

```
mov bx,0400h
mov ds, bx


mov [00h],4029h ; directly moving values to memory location
 mov [02h],4045h ; directly moving the values to memory locations

mov ax,[00h]
mov bx,[02h]

mul bx;This will do the multiplication

mov [004h],ax
mov [006h],dx

ret
```

Input/Output:

Input: ax – 4029h ; bx – 4045h
Output: [4004-4005] – 0D8B ; [4006-4007] – 1B10



400:0004

	H	L
AX	8B	0D
BX	40	45
CX	00	22
DX	10	1B
CS	F400	
IP	0154	
SS	0700	
SP	FFFA	

```
04004: 0D 013 CRET
04005: 8B 139 ï
04006: 1B 027 ←
04007: 10 016 ►
04008: 00 000 NULL
04009: 00 000 NULL
0400A: 00 000 NULL
0400B: 00 000 NULL
0400C: 00 000 NULL
0400D: 00 000 NULL
0400E: 00 000 NULL
0400F: 00 000 NULL
04010: 00 000 NULL
04011: 00 000 NULL
04012: 00 000 NULL
```