**COA LAB ALL PROGRAMS (COMPILED BY** [**KALINGA ABHISEK**](https://instagram.com/kalinga_abhisek)**)**

**1. (Find the largest and smallest number in a given array of size N (8-bit numbers).**

assume cs:code 1,ds: data1

data1 segment

count db 04h ; count=array size

value db 09h,10h,05h,03h; array elements

res db ?

data1 ends; store the result in res

code1 segment

start : mov ax,data1

mov ds,ax

mov cl,count

dec cl

lea si,value

mov al,[si]

up:inc si

cmp al,[si]

jnc nxt

mov al,[si]

nxt:dec cl

jnz up ; jc if to make smallest number

lea di, res

mov [di],al

code1 ends

end start

**2. (Arrange the elements of a given array of size N in ascending and descending order**

**(8-bit numbers).**

assume cs:code, ds:data

data segment

count db 07

value db 09h,0fh,14h,45h,14h,24h,3fh

data ends

code segment

start:mov ax,data

mov ds,ax

lea di,count

mov ch,[di]

dec ch

up2:mov cl,ch

lea si,value

up1:mov al,[si]

cmp al,[si+1]

jc down ; jnc descending order

mov dl,[si+1]

xchg [si],dl

mov [si+1] ,dl

down:inc si

dec cl

jnz up1

dec ax

jnz up2

code ends

end start

**3. (Addition of two 16bit numbers using direct addressing mode.)**

mov ax,0000h

mov ds,ax

mov ax,[3000h]

mov bx,[3002h]

add ax,bx

mov [3004h],ax

**4.(Addition of two 16bit numbers using indirect addressing mode.)**

mov ax,2000h

mov ds,ax

mov si,0500h

mov ax,[si]

inc si

inc si

mov bx,[si]

add ax,bx

inc si

inc si

mov [si],ax

**5.(Addition of two 16bit numbers using index addressing mode.)**

mov ax,3000h

mov ds,ax

mov si,0500h

mov ax,[si+0]

mov bx,[si+2]

add ax,bx

mov [si+4],ax

**6.(Addition of two 16bit numbers using immediate addressing mode.)**

mov ax,5000h

mov bx,4500h

add ax,bx

**7.(Addition of two 16bit numbers using base index addressing mode.)**

mov ax,0000h

mov ds,ax

mov bx,3000h

mov si,0500h

mov cx,[bx+si]

mov dx,[bx+si+02]

add cx,dx

mov [bx+si+04],cx

**8. (Subtraction of two 16 bit numbers using direct addressing mode.)**

mov ax,0000h

mov ds,ax

mov ax,[3000h]

mov bx,[3002h]

sub ax,bx

mov [3004h],ax

**9.(Multiplication of two 16 bit numbers using direct addressing mode.)**

mov ax,0000h

mov ds,ax

mov ax, [3000]

mov bx,[3002]

mul bx

mov [3004],ax

mov ax,dx

mov [3006],ax

hlt

**10.(Multiplication of two 16-bit numbers without using MUL operator.)**

mov ax,0000h

mov ds,ax

mov ax,[3000h]

mov cl,02h

sal ax,cl

mov [3002h],ax

hlt

**11.(Division of two 16-bit numbers without using DIV operator.)**

mov ax,0000h

mov ds,ax

mov ax,[3000h]

mov cl,02h

shr ax,cl

mov [3002h],ax

hlt

**12.(Division of two 16 bit numbers using direct addressing mode.)**

mov ax,0000h

mov ds,ax

mov ax, [2008h]

mov bx,[200ah]

div bx

mov [200ch],ax

hlt

**13.(AND, OR, XOR, NOT two 16 bit numbers using direct addressing mode.)**

mov ax,0000h

mov ds,ax

mov ax,[2000h]

mov bx,[2002h]

and ax,bx

mov [2004h],ax

mov ax,[2000h]

mov bx,[2002h]

or ax,bx

mov [2006h],ax

mov ax,[2000h]

mov bx,[2002h]

xor ax,bx

mov [2008h],ax

mov ax,[2000h]

not ax

mov [200ah],ax

hlt

**14.(2’s complement of an 8 bit number.)**

mov ax,0000h

mov ds,ax

mov ax,[2000h]

not ax

mov [2002h],ax

add ax,01

mov [2004h],ax

**15.(Gray code of an 8-bit number.)**

mov ax,0000h

mov ds,ax

mov ax,[3000h]

mov bx,[3002h]

mov bx,ax

shr ax,01

xor bx,ax

mov [3004h],ax

hlt

**16. i) Swapping of nibble of data1**

mov si, 1000h

mov al,[si]

mov cl,al

inc si

mov bl,[si]

mov dl,al

shr al,04

shl dl,04

or al,dl

inc si

mov [si],al

hlt

**ii) Swapping of nibble of data1 and Swapping of nibble of data1**

**and Y= (data1 and data2) or (data1 xor data2)**

mov si, 1000h

mov al,[si]

mov cl,al

inc si

mov bl,[si]

mov dl,al

shr al,04

shl dl,04

or al,dl

inc si

mov [si],al

mov al,cl

and al,bl

mov dl,al

xor cl,bl

or al,cl

inc si

mov [si],al

hlt

**17.(ADD, SUB, MUL of two 32-bit numbers using load/store addressing mode.)**

.global \_start

\_start:

ldr r4,#0x101f1000

ldr r1,[r4] @1st data

add r4,r4,#04

ldr r2,[r4] @2nd data

add r3,r1,r2

add r4,#04

str r3,[r4]

sub r3,r1,r2

add r4,#04

str r3,[r4]

mul r3,r1,r2

add r4,#04

str r3,[r4]

exit: b exit @so infinite loop at the end

**18.(Perform the logical operations (AND, OR, XOR and NOT) on two 32bit numbers using load/store addressing mode.)**

.global \_start

\_start:

ldr r4,#0x101f1000

ldr r1,[r4] @1st data

add r4,r4,#04

ldr r2,[r4] @2nd data

and r3,r1,r2

add r4,#04

str r3,[r4]

orr r3,r1,r2

add r4,#04

str r3,[r4]

eor r3,r1,r2

add r4,#04

str r3,[r4]

str r3,[r4]

mvn r3,r1

add r4,#04

str r3,[r4]

exit: b exit @so infinite loop at the end

**19.(Find the largest number in a given array of size N.)**

.global \_start

\_start:

@largest number from a given array

ldr r0,=count

ldr r1,[r0]

mov r4,#0x00

ldr r2,=array

back:ldr r3,[r2], #04

cmp r4,r3

bgt fwd

mov r4,r3

fwd:subs r1,r1, #01

bne back

str r4,[r2]

exit: b exit

.data

count:.word 0x05

array:.word 0x15,0x35,0x45,0x10,0x4f

**20.(Find the smallest number in a given array of size N.)**

.global \_start

\_start:

@smallest number from a given array

ldr r0,=count

ldr r1,[r0]

mov r4,#0xff

ldr r2,=array

back:ldr r3,[r2], #04

cmp r4,r3

blt fwd

mov r4,r3

fwd:subs r1,r1, #01

bne back

str r4,[r2]

exit: b exit

.data

count:.word 0x05

array:.word 0x15,0x35,0x45,0x10,0x4f

**21.(Program to separate even numbers and odd numbers from the given array elements.)**

.global \_start

\_start:

ldr r0,=count

ldr r1,[r0]

mov r2,#0x00

ldr r3,=array @ r2 = base address of array=array[0]

ldr r4,=even @r3=base address of even data locations =even[0]

ldr r5,=odd @r4=base address of even data locations =odd[0]

back: ldr r6, [r3],#4

ands r7,r6,#1

beq fwd

str r6,[r5],#4

b fwd1

fwd: str r6,[r4],#4

fwd1: subs r1,r1,#01

bne back

exit: b exit

.data

count: .word 0x07

array: .word 0x15, 0x35,0x32, 0x45, 0x10,0x4f,0x34,

even: .word 0, 0, 0, 0, 0

odd: .word 0, 0, 0, 0, 0

**22.(Find the sum and average of N 16-bit numbers.)**

mov si,2000h

mov cx,0005h

mov bx,cx

mov ax,0000h

back: add ax,[si]

jnc skip

inc dx

skip: inc si

inc si

dec cx

jnz back

mov [si],ax

inc si

inc si

mov [si],dx

div bx

inc si

inc si

mov [si],ax

inc si

mov [si],dx

**23.(Count no. of 1’s in an 8bit number.)**

mov ax,0000h

mov ds,ax

mov bx,3000h

mov al,[bx]

mov cl,08h

mov ch,00h

l2: shr al,01h

jnc l1

inc ch

l1: dec cl

jnz l2

inc bx

mov [bx],ch

hlt