Waleed Afzal

23p-0566

Section 3D

COAL Assignment # 02

Q1**:**[org 0x0100]

jmp start

missing: db 0

message: db "Missing Number is : "

msgLength: db 20

n: db 8 ; Range of array [1,N]

arr: db 1, 2, 6, 4, 6, 3, 7, 8 ; size = n (with duplicate)

new\_arr: db 0 ; size = n-1 (without duplicate)

clrscr:

push es

push ax

push di

mov ax, 0xb800

mov es, ax

xor di, di

nextloc:

mov word [es:di], 0x0720

add di, 2

cmp di, 0xFA0

jne nextloc

pop di

pop ax

pop es

ret

removeDuplicate:

push bp

mov bp, sp

push ax

push bx

push cx

push si

push di

xor ax, ax

xor cx, cx

mov si, [bp+6] ; Original array address

mov di, [bp+8] ; New array address

checkNum:

mov al, [si]

xor bx, bx ; Reset innerloop counter

innerloop:

cmp bx, cx

je notFound

cmp al, [di+bx]

je duplicate

inc bx

jmp innerloop

notFound:

mov bx, cx

mov [di+bx], al

inc cx

duplicate:

inc si ; Increment arr index

cmp cx, [bp+4] ; Compare SI with size n

jne checkNum

pop di

pop si

pop cx

pop bx

pop ax

pop bp

ret 6

findMissing:

push bp

mov bp, sp

push ax

push bx

push cx

push dx

push si

; Sum of the natural numbers til n

mov al, [bp+4] ; Load n = 8 in AL

mov ah, 0

mov bx, ax ; Store 8 in BX

inc bx ; Increment BX

mul bx ; Multiply BX with AX

shr ax, 1 ; Divide AX by 2

xor dx, dx

mov dl, al

mov si, [bp+6] ; Load new\_arr address in SI

xor cx, cx

mov cl, [bp+4] ; Load n = 8 in CX

dec cl ; Array size = n - 1

xor ax, ax

xor bx, bx

sumLoop:

mov bl, [si]

add al, bl

inc si

loop sumLoop

sub dl, al

mov [missing], dl

pop si

pop dx

pop cx

pop bx

pop ax

pop bp

ret 4

printScrn:

push bp

mov bp, sp

push ax

push bx

push cx

push dx

push es

push si

push di

mov ax, 0xb800

mov es, ax

xor di, di

mov di, 160 ; Printing Location

; Print Message

mov si, [bp+8] ; Load address of message

mov cx, [bp+6] ; Load message length = 20

mov ah, 0x47

nextchar:

mov al, [si]

mov [es:di], ax

add di, 2

inc si

loop nextchar

; Printing Missing Number

xor ax, ax

mov al, [bp+4] ; Missing number

mov bx, 10 ; Decimal base 10

xor cx, cx

nextdigit:

xor dx, dx

div bx

add dl, 0x30 ; Convert to ASCII

push dx

inc cx

cmp ax, 0

jne nextdigit

printDigit:

pop dx

mov dh, 0x47

mov [es:di], dx

add di, 2

loop printDigit

pop di

pop si

pop es

pop dx

pop cx

pop bx

pop ax

pop bp

ret 6

start:

call clrscr

push word new\_arr

push word arr

xor ax, ax

mov al, [n]

push ax

call removeDuplicate

push word new\_arr

xor ax, ax

mov al, [n]

push ax ; Value of n

call findMissing

push word message

xor ax, ax

mov al, [msgLength]

push ax

xor ax, ax

mov al, [missing] ; missing = 0x00FD = 253

push ax

call printScrn

mov ax, 0x4c00

int 0x21

A computer screen shot of a black screen

Description automatically generated

**Q2:**

[org 0x0100]

jmp start

arr: dw 1, 2, 3, 4

arrSize: dw 8

counting:

xor si, si

checkOneZero:

mov ax, [arr+si]

test ax, 1

jz countZero

mov cx, 0

mov dx, 16

countOne:

shr ax, 1

jnc skipOne

inc cx

skipOne:

dec dx

jnz countOne

mov [arr+si], cx ; Store count in array

jmp doneCounting

countZero:

mov cx, 0

mov dx, 16 ; 16-bit count

countZeroLoop:

shr ax, 1

jc skipZero

inc cx

skipZero:

dec dx

jnz countZeroLoop

mov [arr+si], cx ; Store count of zeros in array

doneCounting:

add si, 2

cmp word si, [arrSize]

jne checkOneZero

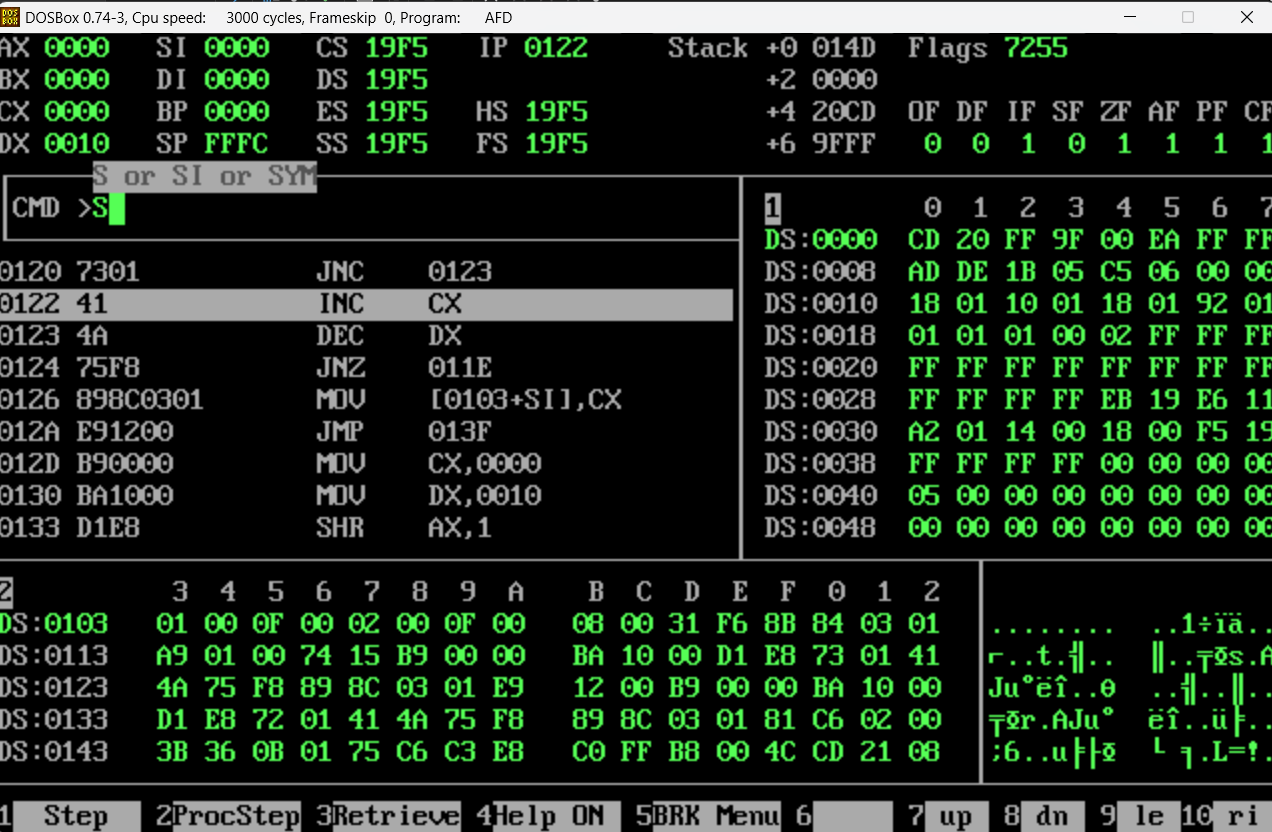
ret

start:

call counting

mov ax, 0x4c00

int 0x21

****



**Q3:**

[org 0x0100]

jmp start

SensorData1: dw 0x0599

SensorData2: dw 0x8855

totalAlerts: db 0

criticalAlerts: db 0

warningAlerts: db 0

infoAlerts: db 0

alerts: db "All Alerts: "

critical: db "Critical Alerts: "

warning: db "Warning Alerts: "

info: db "Info Alerts: "

clrscr:

push ax

push es

push di

mov ax, 0xb800

mov es, ax

mov di,0

mov ax, 0x0720

nextloc:

mov [es:di], ax

add di, 2

cmp di, 4000

jne nextloc

pop di

pop es

pop ax

ret

countOnes:

push bp

mov bp, sp

push ax

push dx

push cx

mov ax, [bp+4] ; Load SensorData1

mov cx, 16 ; 16-bits counter

xor dx, dx

mov byte [totalAlerts], 0

l1:

shr ax, 1

jnc skip

inc dx

skip:

loop l1

mov [totalAlerts], dl

pop cx

pop dx

pop ax

pop bp

ret 2

categorizeAlerts:

push bp

mov bp, sp

push ax

push bx

push cx

mov ax, [bp+4] ; Load SensorData1

xor cx, cx

mov byte [criticalAlerts], 0

mov byte [warningAlerts], 0

mov byte [infoAlerts], 0

mainloop:

shr ax, 1

jnc skipAlerts

cmp cx, 7

jbe infoCounts

cmp cx, 11

jbe warningCounts

cmp cx, 15

jbe criticalCounts

infoCounts:

inc byte [infoAlerts]

jmp skipAlerts

warningCounts:

inc byte [warningAlerts]

jmp skipAlerts

criticalCounts:

inc byte [criticalAlerts]

skipAlerts:

inc cx

cmp cx, 15 ; 16-bits (0-15) counter

jbe mainloop

pop cx

pop bx

pop ax

pop bp

ret 2

printMessage:

push bp

mov bp, sp

push es

push ax

push cx

push si

push di

mov ax,0xb800

mov es, ax

mov di, [bp+4]

mov ah, 0x07

printAlerts:

mov si, alerts

mov cx, 11 ; Message Length

nextcharA:

mov al, [si]

mov [es:di], ax

add di, 2

inc si

loop nextcharA

printAlertsVal:

add di, 2

mov al, [totalAlerts]

add al, 0x30

mov [es:di], ax

add di, 4

printCritical:

mov si, critical

mov cx, 16 ; Message Length

nextcharC:

mov al, [si]

mov [es:di], ax

add di, 2

inc si

loop nextcharC

printCriticalVal:

add di, 2

mov al, [criticalAlerts]

add al, 0x30

mov [es:di], ax

add di, 4

printWarning:

mov si, warning

mov cx, 15 ; Message Length

nextcharW:

mov al, [si]

mov [es:di], ax

add di, 2

inc si

loop nextcharW

printWarningVal:

add di, 2

mov al, [warningAlerts]

add al, 0x30

mov [es:di], ax

add di, 4

printInfo:

mov si, info

mov cx, 12

nextcharI:

mov al, [si]

mov [es:di], ax

inc si

add di, 2

loop nextcharI

printInfoVal:

add di, 2

mov al, [infoAlerts]

add al, 0x30

mov [es:di], ax

done:

pop di

pop si

pop cx

pop ax

pop es

pop bp

ret 2

start: call clrscr

push word [SensorData1]

call countOnes

push word [SensorData1]

call categorizeAlerts

mov ax, 160

push ax

call printMessage

push word [SensorData2]

call countOnes

push word [SensorData2]

call categorizeAlerts

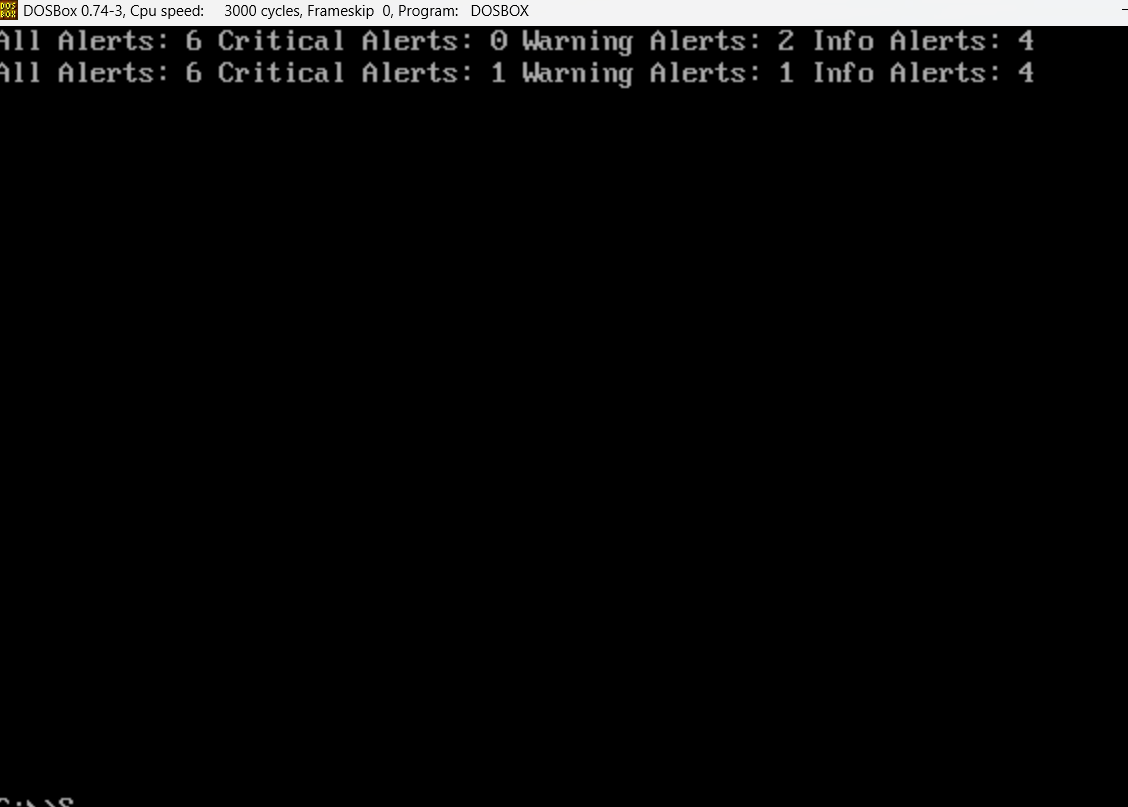
mov ax, 320

push ax

call printMessage

mov ax, 0x4c00

int 0x21

****

Q4:

[org 0x0100]

jmp start

size: dw 5 ; size = size + num = 10

num: dw 5 ; 23P-0566

sum: dw 0

clrscr:

push ax

push es

push di

mov ax, 0xb800

mov es, ax

mov di,0

mov ax, 0x0720

nextloc:

mov [es:di], ax

add di, 2

cmp di, 4000

jne nextloc

pop di

pop es

pop ax

ret

calculateSum:

push bp

mov bp, sp

push ax

push bx

push cx

push dx

xor ax, ax

mov ax, [bp+6]

add ax, [bp+4] ; Total size

mov bx,0 ; Outerloop counter

mov cx,0 ; Innerloop counter

mov dx,0

; Outer most loop

mainloop:

mov cx,0

inc bx

cmp bx, ax

jg exit

; Addition till size

innerloop:

inc cx

add dx, cx

cmp cx, bx

jl innerloop

; Addition from size till 1

innerloop2:

cmp cx, 1

je mainloop

dec cx

add dx, cx

jmp innerloop2

exit:

mov [sum], dx

pop dx

pop cx

pop bx

pop ax

pop bp

ret 4

displaySum:

push bp

mov bp, sp

push ax

push bx

push cx

push dx

push es

push di

mov ax, 0xb800

mov es, ax

mov di, 160 ; Printing from row 2 column 1

mov ax, [bp+4] ; Loading number into ax

mov bx, 10 ; Base 10 for Decimal number

mov cx, 0

; Converting decimal into hexadecimal

convert:

mov dx,0

div bx

add dl, 0x30

push dx

inc cx

cmp ax, 0

jnz convert

; Printing sum digits on screen

print:

pop dx

mov dh, 0x0F

mov [es:di], dx

add di, 2

loop print

pop di

pop es

pop dx

pop cx

pop bx

pop ax

pop bp

ret 2

start:

call clrscr

; Pushing parameters

push word [size]

push word [num]

; Calling subroutine

call calculateSum

; Pushing parameters

push word [sum]

; Calling subroutine

call displaySum

mov ax, 0x4c00

int 0x21

A screen shot of a computer

Description automatically generated

Q5

[org 0x0100]

jmp start

arr: db 1,3,2,5,4,7,9,8

length: db 8

swap: db 0

minimum: db 0

maximum: db 0

median: db 0

MinMsg: db "Minimum: "

MaxMsg: db "Maximum: "

MedianMsg: db "Median: "

clrscr:

push es

push ax

push di

mov ax, 0xb800

mov es, ax

xor di, di

nextloc:

mov word [es:di], 0x0720

add di, 2

cmp di, 0xFA0

jne nextloc

pop di

pop ax

pop es

ret

BubbleSort:

push bp

mov bp, sp

sub sp, 2

push ax

push bx

push cx

push si

xor ax, ax

mov bx, [bp+4] ; Address of array

xor cx, cx

mov cl, [bp+6] ; length of array

dec cx

mainloop:

mov si, 0

mov word [bp-2], 0

innerloop:

mov al, [bx+si]

cmp al, [bx+si+1]

jbe noswap

xchg al, [bx+si+1]

mov [bx+si], al

mov word [bp-2], 1

noswap:

inc si

cmp si, cx

jne innerloop

cmp word [bp-2], 1

je mainloop

pop si

pop cx

pop bx

pop ax

add sp, 2

pop bp

ret 4

StatsOfArray:

push bp

mov bp, sp

push ax

push bx

push si

xor bx, bx

mov bl, [bp+4] ; Array length

mov si, [bp+6] ; Array address

push si

push bx

call BubbleSort

; Minimum

mov al, [si]

mov [bp+12], al

; Maximum

mov al, [si+bx-1]

mov [bp+10], al

test bl, 1 ; Compare if length is Odd

jnz isOdd

shr bl, 1

mov al, [si+bx]

mov ah, [si+bx-1]

add al, ah

shr al, 1

mov [bp+8], al

jmp exit

isOdd:

shr bl, 1

mov al, [si+bx]

mov [bp+8], al

exit:

pop si

pop bx

pop ax

pop bp

ret 4

printNumbers:

push bp

mov bp, sp

push es

push ax

push cx

push si

push di

mov ax, 0xb800

mov es, ax

mov di, 160

mov ah, 0x07

printMax:

mov si, MaxMsg

mov cx, 9

nextcharMax:

mov al, [si]

mov [es:di], ax

add di, 2

inc si

loop nextcharMax

MaxValue:

mov al, [bp+6]

add al, 0x30

mov [es:di], ax

add di, 4

printMin:

mov si, MinMsg

mov cx, 9

nextcharMin:

mov al, [si]

mov [es:di], ax

add di, 2

inc si

loop nextcharMin

MinValue:

mov al, [bp+8]

add al, 0x30

mov [es:di], ax

add di, 4

printMedian:

mov si, MedianMsg

mov cx, 8

nextcharMedian:

mov al, [si]

mov [es:di], ax

add di, 2

inc si

loop nextcharMedian

MedianValue:

mov al, [bp+4]

add al, 0x30

mov [es:di], ax

add di, 2

pop di

pop si

pop cx

pop ax

pop es

pop bp

ret 6

start: call clrscr

xor ax, ax

xor bx, bx

xor cx, cx

; Output variables

mov al, [minimum]

push ax

mov al, [maximum]

push ax

mov al, [median]

push ax

push word arr

mov al, [length]

push ax

call StatsOfArray

pop cx ; Median value

pop bx ; Maximum value

pop ax ; Minimum value

mov [minimum], al

mov [maximum], bl

mov [median], cl

push ax

push bx

push cx

call printNumbers

mov ax, 0x4c00

int 0x21

A screen shot of a computer

Description automatically generated

Q6

[org 0x0100]

jmp start

; Roll Number Digits

input1: dw 0

input2: dw 5

input3: dw 9

input4: dw 9

; Random values

output1: dw 4

output2: dw 7

output3: dw 9

clrscr: push ax

push es

push di

mov ax, 0xb800

mov es, ax

xor di, di

clrloop:

mov word [es:di], 0x0720

add di, 2

cmp di, 0xFA0

jne clrloop

pop di

pop es

pop ax

ret

subroutine1:

push bp

mov bp, sp

push ax

push bx

push cx

push si

push di

sub sp, 10 ; Make space for 5 local variables

mov word [bp-12], 1

mov word [bp-14], 2

mov word [bp-16], 3

mov word [bp-18], 4

mov word [bp-20], 5

; Perform Addition Operations

mov ax, [bp+10] ; Load first input

add ax, [bp-12] ; Add first local variable

mov [bp+16], ax ; Store result in the first output

mov bx, [bp+8] ; Load second input

add bx, [bp-14] ; Add second local variable

mov [bp+14], bx ; Store result in the second output

mov cx, [bp+6] ; Load third input

add cx, [bp-16] ; Add third local variable

mov [bp+12], cx ; Store result in the third output

mov si, [bp+4]

add si, [bp-18]

mov di, [bp+4]

add di, [bp-20]

; Restore stack for local variables

add sp, 10

pop di

pop si

pop cx

pop bx

pop ax

pop bp

ret 8

subroutine2:

push bp

mov bp, sp

push ax

sub sp, 4 ; Make space for 2 local variables

mov word [bp-4], 5

mov word [bp-6], 10

mov ax, [bp+10] ; Load first input

add ax, [bp+8] ; Add second input

mov [bp+14], ax ; Store result in second output

mov ax, [bp+4] ; Load fourth input

add ax, 6 ; Add 6

sub ax, 1 ; Subtract 1

mov [bp+12], ax ; Store result in third output

mov ax, [bp+8] ; Load second input

add ax, [bp-6] ; Add second local variable

mov [bp+16], ax ; Store result in first output

; Restore stack for local variables

add sp, 4

pop ax

pop bp

ret 8

display:

push bp

mov bp, sp

push es

push di

push ax

push bx

push cx

push dx

push si

mov ax, 0xb800

mov es, ax

mov di, [bp+4] ; Load printing location

mov ax, [bp+6] ; Load number into ax

mov bx, 10 ; Decimal Number Base 10

xor cx, cx

; Converting num in Decimal

getNum:

xor dx, dx

div bx

add dx, '0' ; Convert remainder to ASCII

push dx

inc cx

cmp ax, 0

jnz getNum

; Printing num in Decimal

printNum:

pop dx

mov dh, 0x4f

mov [es:di], dx

add di, 2

loop printNum

pop si

pop dx

pop cx

pop bx

pop ax

pop di

pop es

pop bp

ret 4

start: ; Output variables with random values

push word [output1]

push word [output2]

push word [output3]

; Roll Number Digits

push word [input1]

push word [input2]

push word [input3]

push word [input4]

; Call subroutine1 to initialize values

call subroutine1

; Pop outputs into AX, BX, CX

pop cx

pop bx

pop ax

; Clear screen before displaying results

call clrscr

push ax

push word 160 ; Print location for output 1

call display

push bx

push word 320 ; Print location for output 2

call display

push cx

push word 480 ; Print location for output 3

call display

; Input values for subroutine2

mov word [input1], 32 ; Lahore (input 1)

mov word [input2], 46 ; Faisalabad (input 2)

mov word [input3], 16 ; Murree (input 3)

mov word [input4], 21 ; Islamabad (input 4)

push word [input1]

push word [input2]

push word [input3]

push word [input4]

; Call subroutine2 for temperature calculations

call subroutine2

; Pop outputs into AX, BX, CX

pop cx

pop bx

pop ax

; Store outputs

mov [output3], cx

mov [output2], bx

mov [output1], ax

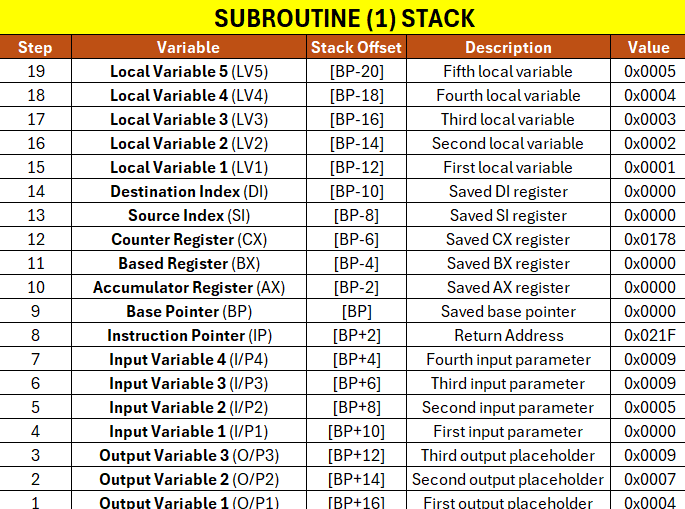
mov ax, 0x4c00

int 0x21

A black screen with white text

Description automatically generated

Stack Attachment:



A table with text and numbers

Description automatically generated