1. **What is the difference between the directives, ‘=’, ‘equ’, and ‘textequ’?**

EQU is used for assigning integer values to variables such as:

*Salary EQU 300h*

*Savings EQU Salary*

Along with this any symbol that is assigned integer values using EQU cannot be redefined in the same source file.

TEXTEQU is similar to EQU, it creates text macro such as:

Name TEXTEQU “Arham 22i-1552”

Also, a symbol defined by TEXTEQU can be re-defined at any time in the source file.

1. **Verify little endian order of saving variables in memory.**

In little Indian ordering the least significant byte of a multi-byte variable is stored at the lowest memory address while the most significant byte is stored at the highest memory address.

For e.g.

Num word 101112h will be stored as:

12 11 10 00 in memory.

**Q.3.**

**A screenshot of a computer

Description automatically generated**

dosseg

.model small

.stack 100h

.data

byte\_array db 'a','b','c','d','e'

word\_array dw 1234h,5678h,9ABCh

dword\_array dd 12345678h, 87654321h

string db "Byte",'$'

string\_2 db "Word",'$'

string\_3 db "Dword",'$'

string\_4 db "Values are in the order of :: offset -> lenght -> size -> type",'$'

array\_length db ?, ?, ?

array\_size db ?, ?, ?

.code

main proc

mov ax, @data

mov ds, ax

mov dx,offset string\_4

mov ah,9

int 21h

mov ah, 0Eh ;print new line sequence

mov al, 0Dh

int 10h

mov al, 0Ah

int 10h

mov dx,offset string ; prints byte

mov ah,9

int 21h

mov ah, 0Eh ;print new line sequence

mov al, 0Dh

int 10h

mov al, 0Ah

int 10h

mov dx,offset byte\_array

add dx,48

mov ah,2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

mov dl, byte ptr lengthof byte\_array

add dl, 48

mov ah, 2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

mov dl, byte ptr sizeof byte\_array

add dl, 48

mov ah, 2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

mov dl,byte ptr type byte\_array

add dl,48

mov ah,2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

; yahan sy word ka hy

mov dx,offset string\_2 ; prints word

mov ah,9

int 21h

mov ah, 0Eh ;print new line sequence

mov al, 0Dh

int 10h

mov al, 0Ah

int 10h

mov dx,offset word\_array

add dx,48

mov ah,2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

mov dl, byte ptr lengthof word\_array

add dl, 48

mov ah, 2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

mov dl, byte ptr sizeof word\_array

add dl, 48

mov ah, 2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

mov dl,byte ptr type word\_array

add dl,48

mov ah,2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

; yahan sy dword ka hy

mov dx,offset string\_3 ; prints dword

mov ah,9

int 21h

mov ah, 0Eh ;print new line sequence

mov al, 0Dh

int 10h

mov al, 0Ah

int 10h

mov dx,offset dword\_array

add dx,48

mov ah,2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

mov dl, byte ptr lengthof dword\_array

add dl, 48

mov ah, 2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

mov dl, byte ptr sizeof dword\_array

add dl, 48

mov ah, 2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

mov dl,byte ptr type dword\_array

add dl,48

mov ah,2

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

mov ah,4ch

int 21h

main endp

end main

**Q.4.**

**A screenshot of a computer

Description automatically generated**

dosseg

.model small

.stack 100h

.data

byte\_array db 1,2,3,4,5,6

rollnumber db "1552",'$'

array\_length db ?, ?, ?

array\_size db ?, ?, ?

.code

main proc

mov ax, @data

mov ds, ax

mov al, rollnumber + 3

sub al, '0'

and al, 1

cmp al, 0

je add\_even

jmp add\_odd

add\_even:

mov si, offset byte\_array

mov cx, lengthof byte\_array

mov bl, 1

add\_loop:

mov al, [si]

add al, bl

mov [si], al

add si, 2

loop add\_loop

jmp print\_array

add\_odd:

mov si, offset byte\_array + 1

mov cx, lengthof byte\_array - 1

mov bl, 2

add\_loop2:

mov al, [si]

add al, bl

mov [si], al

add si, 2

loop add\_loop2

print\_array:

mov si, offset byte\_array

mov cx, lengthof byte\_array

print\_loop:

mov dl, [si]

add dl, '0'

mov ah, 02h

int 21h

inc si

loop print\_loop

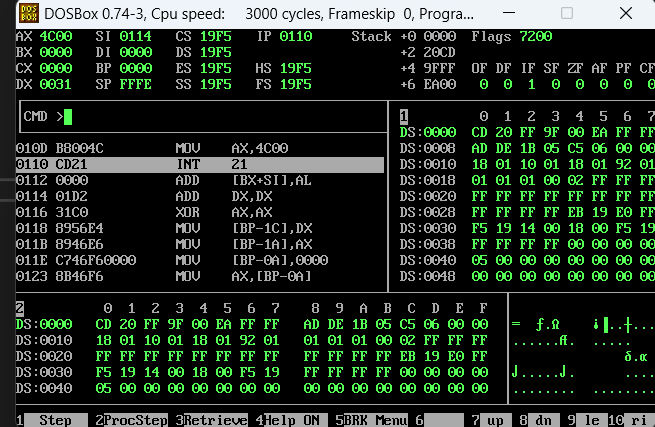
mov ah, 4Ch

int 21h

main endp

end main

**Q.5.**

****

[org 0x100]

section .data

rollnumber db 1

section .text

mov si, rollnumber

mov al, [si]

add al, '0'

mov dl, al

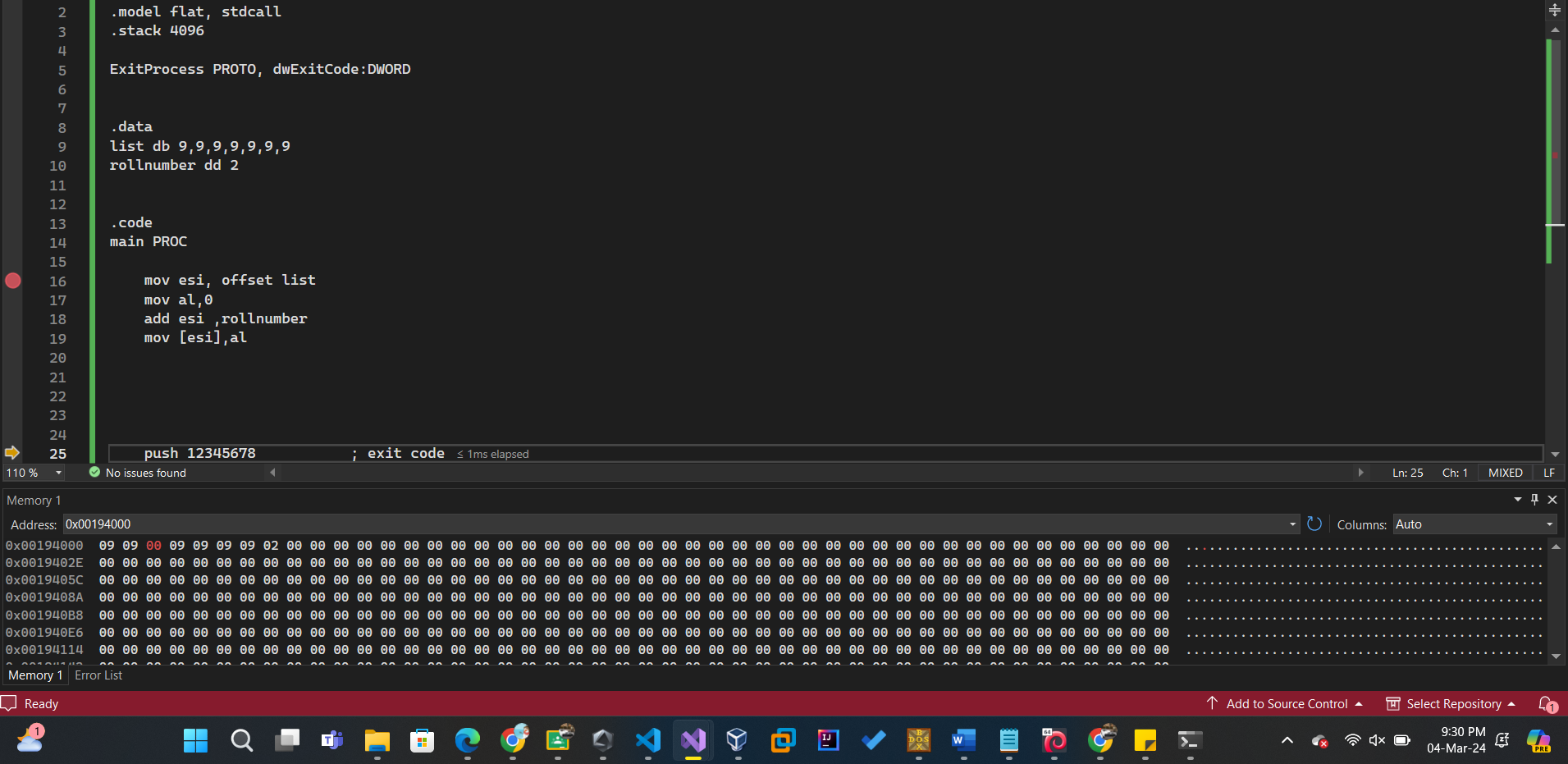
mov ah, 0x02

int 0x21

mov ax, 0x4c00

int 0x21

**Q.6.**

****

.386

.model flat, stdcall

.stack 4096

ExitProcess PROTO, dwExitCode:DWORD

.data

var dd 1552

.code

main PROC

mov esi, offset list

mov al,0

add esi ,rollnumber

mov [esi],al

push 12345678 ; exit code

call ExitProcess ; Or call ExitProcess@4

main ENDP

END main ;specifying the program's entry point