**I21-0707 Hamza Tariq Y**

**Tasks:**

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

“ = ”

is used to assign the values to the variables during assembled time which are to be used. The assembler during assembling the code replaces the variable with the value assigned by the **=** symbol. And the value of the variable assigned using **=** can also be redefined with in the program.

“ EQU ”

is more general which allows the numeric constants and as well as text. The text value can be changed after declaration. But one of it’s drawback is that the value assigned to any variable using **EQU** cannot be redefined any where in the program, we can say that it fixes the value.

“ TEXT EQU ”

Deals with text literals and strings and if the integer value is assigned to any variable

using **TEXT EQU** than it deals that integer as a Ascii value for any character and saves that as Character literak in memory **EXAMPLE** if we store **65** to any variable using **TEXT EQU** it considers that value as **ASCII** and returns Character A when Accessed .

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

.model small

.stack 100h

.data

var dd 12345678H

.code

MOV AX,@DATA

MOV DS,AX

mov si,offset var

mov ah,4ch

int 21h

end

Graphical user interface

Description automatically generated

1. Declare and initialize arrays of type byte, word, double word with non-zero elements of your choice.
2. Find out the type of array and save this value in a variable.
3. Find out the length of array and save this value in a variable.
4. Find out the size of the array and save this value in a variable.
5. Find out the offset of the array and view the array in memory.

.model small

.stack 100h

.data

var db 10H,20H,30H,40H,50H

var1 word 10H,20H,30H,40H,50H

var2 dword 10H,20H,30H,40H,50H

len\_var db ?

len\_var1 db ?

len\_var2 db ?

type\_var db ?

type\_var1 db ?

type\_var2 db ?

size\_var db ?

size\_var1 db ?

size\_var2 db ?

.code

MOV AX,@DATA

MOV DS,AX

mov len\_var,lengthof Var

mov type\_var,Type Var

mov size\_var,sizeof Var

mov bl,len\_var

mov bh,type\_var

mov cl,size\_var

mov len\_var1,lengthof Var1

mov type\_var1,Type Var1

mov size\_var1,sizeof Var1

mov bl,len\_var1

mov bh,type\_var1

mov cl,size\_var1

mov len\_var2,lengthof Var2

mov type\_var2,Type Var2

mov size\_var2,sizeof Var2

mov bl,len\_var2

mov bh,type\_var2

mov cl,size\_var2

mov si,offset var

mov si,offset var1

mov si,offset var2

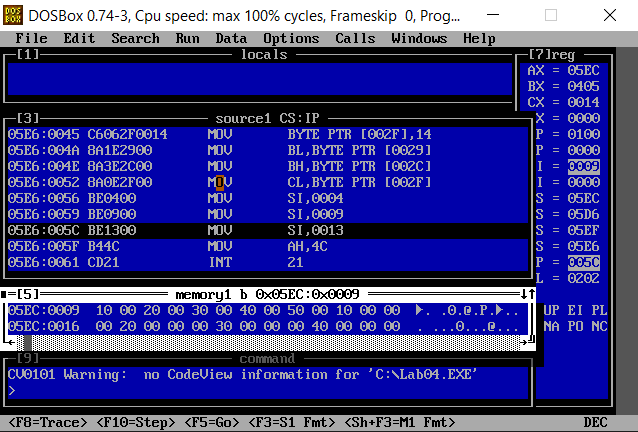
mov ah,4ch

int 21h

end

Graphical user interface

Description automatically generated



Graphical user interface

Description automatically generated

1. Using the array declared and initialized in above question, you need to add any number at the even/odd indices of the array. (even/odd determined based on your roll number.)

.model small

.stack 100h

.data

var db 10H,20H,30H,40H,50H

var1 word 10H,20H,30H,40H,50H

var2 dword 10H,20H,30H,40H,50H

.code

MOV AX,@DATA

MOV DS,AX

add [var+1],10H

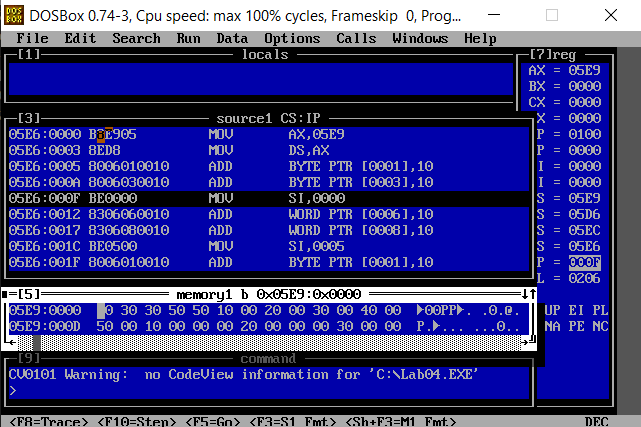
add [var+3],10H

mov si,offset var

mov ah,4ch

int 21h

end



1. Declare and initialize a variable and access its value using indirect addressing method and view its value in memory location.

.model small

.stack 100h

.data

var db 10h

.code

MOV AX,@DATA

MOV DS,AX

mov si,offset var

mov ah,4ch

int 21h

end

Graphical user interface

Description automatically generated

1. Using the array in question 3, replace an element in array at the index based on the last digit of your roll number by a variable. View the new array in memory.

.model small

.stack 100h

.data

var word 10H,20H,30H,40H,50H,60H,70H,80H

EX\_var word ?

.code

MOV AX,@DATA

MOV DS,AX

mov EX\_var,10H

mov bx,EX\_var

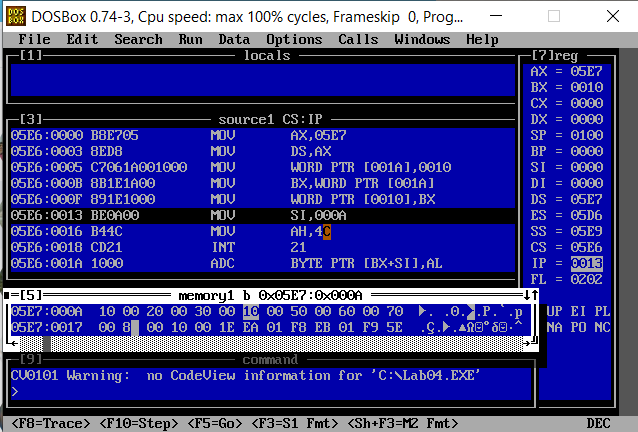
mov [var+6],bx

mov si,offset var

mov ah,4ch

int 21h

end



1. Declare and initialize a double or quad word type variable in memory, fetch its most significant 2 bytes and make it dividend, make its least significant byte a divisor and perform division. (e.g., if 12345678h is saved in memory, divide 1234h by 08h)

.model small

.stack 100h

.data

var Dq 13885664H

.code

MOV AX,@DATA

MOV DS,AX

mov bx,word ptr[var+2]

mov cl,byte ptr [var]

mov ax,bx

Div cl

mov ah,4ch

int 21h

end

Graphical user interface

Description automatically generated

1. Declare and initialize a double or quad word type variable in memory, fetch its most significant byte and make it one multiplier, make its least significant byte second multiplier, and perform multiplication. (e.g., if 12345678h is saved in memory, multiply 12h by 78h)

.model small

.stack 100h

.data

var DQ 1234567890123456H

.code

MOV AX,@DATA

MOV DS,AX

mov bl,byte ptr[var+2]

mov cl,byte ptr [var]

mov al,bl

mul cl

mov ah,4ch

int 21h

end

