**Computer Organization & Assembly Language**

**Lab 04**

**Topics:**

1. Directives ($, =, equ, textequ)
2. Arrays and its operations
3. Indirect Addressing
4. PTR operator

**Tasks:**

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

‘=’ it’s a directive that is used for assignment

EQU is more general in that it allows numeric constants as well as text constants. EQU also explicitly states that a text value can be changed after declaration. While the documentation is confusing, the statement "The name cannot be redefined later" only applies to the first form of EQU "name EQU expression", whereas the second form of EQU "name EQU <text>" is annotated with "The name can be assigned a different text later."

TEXTEQU, on the other hand, only deals with text literals. The use of the normal double quoted text, literals proceeded by % (which I've never seen), and the values of macros. The latter two types do not seem to be supported by EQU, but I have not tested that.

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

.model small

.stack 100h

.data

var dw 12345678h

.code

mov ax,@data

mov ds,ax

mov ah,4ch

int 21h

end

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

.model small

.stack 100h

.data

arr1 db 1, 2

len db ?

siz db ?

typ db ?

.code

mov esi, OFFSET arr1

MOV AX, @DATA

MOV DS, AX

MOV AL,Lengthof arr1

MOV len,AL

MOV AL, Type arr1

MOV typ , AL

MOV AL, Sizeof arr1

MOV siz , AL

mov ah,4ch

int 21h

end

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

arr1 db 1, 2 ,3 ,4

len db ?

siz db ?

typ db ?

.code

mov si, OFFSET arr1

MOV AX, @DATA

MOV DS, AX

mov AL,4

mov arr1[2],al

mov AL,6

mov arr1[4],al

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

arr1 db 1, 2 ,3 ,4

len db ?

siz db ?

typ db ?

.code

mov si, OFFSET arr1

MOV AX, @DATA

MOV DS, AX

mov arr1[si],AL

mov arr1[si+1],AL

mov arr1[si+2],AL

mov arr1[si+3],AL

mov ah,4ch

int 21h

end

1. Using the array in question 2, 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

arr1 db 1, 2 ,3 ,4

var db 6

.code

mov si, OFFSET arr1

MOV AX, @DATA

MOV DS, AX

mov al,var

mov arr1[1],al

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 dd 12345678h

.code

MOV AX, @DATA

MOV DS, AX

mov ax,word ptr[var+2]

mov bl,byte ptr[var]

div bl

mov ah,4ch

int 21h

end

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 dd 12345678h**

**.code**

**mov ax,@data**

**mov ds,ax**

**mov ax,word ptr[var+3]**

**mov bl,byte ptr[var]**

**mul bl**

**mov ah,4ch**

**int 21h**

**end**