**COAL Assignment 2**

Name: Ahmed Kasteer

Roll no: 20F-0336

Section 3D

Question 1

.data

bVal BYTE 100

bVal2 BYTE ?

wVal WORD 2

dVal DWORD 5

.code

**mov 45,45**

In the above Statement, the error is while using mov statement because we cannot use both constant value. Destination must be a register, Variable or a Memory Location. It should be like mov ax,45.

**mov 45, ds**

In the above Statement, the error is while using mov statement we cannot use constant value as destination. Destination must be register, Variable or Memory Location. It should be like mov ax, ds.

**mov esi, wVal**

In the above Statement, the error is while using mov statement we cannot use ESI as destination, As others have indicated, they have special uses with the string instructions. For real mode programming, the ES segment register must be used with DI and DS with SI as destination must be register, Variable. Memory Location should be like mov ax,45.

**mov eip, dVal**

In the above Statement, the error is while using mov statement we cannot use EIP as destination, EIP cannot be used as memory storing register. They have special purposes: The EIP register (prior to transferring program control) contains the address of the instruction following the CALL instruction. When this address is pushed on the stack, it is referred to as the return instruction pointer or return address., Variable/Memory Location. It should be like mov ax,45.

**mov 25, bVal**

In the above Statement, the error is while using mov statement. We cannot use both constant value, destination must be register, Variable or Memory Location. It should be like mov ax, bVal.

**mov bVal2, bVal**

In the above Statement, the error is while using mov statement. We cannot use both variable, one must be register or a Memory Location. It should be like mov ax, bVal.

Question 2

Using in indexed addressing:

.model small

.stack 100h

.data

arrayD dd 1,2,3

STRING db 'BEFORE REARRANGING : ', '$'

n\_line db 0AH,0DH,"$"

STRING2 db 'AFTER REARRANGING : ', '$'

.code

mov ax,@data

mov ds,ax

lea dx,STRING ;Moving string to dx

mov ah,09h ;Print the Strings

int 21h

mov [arrayD],ax

mov dx,[arrayD] ;Moving first index to dx

add dx,48 ;Adding 48 to Convert into value

sub dx,35

mov ah,02h ;Print the Value

int 21h

mov dx,[arrayD+4] ;Moving Second index to dx

add dx,48 ;Adding 48 to Convert into value

mov ah,02h ;Print the Value

int 21h

mov dx,[arrayD+8] ;Moving Third index to dx

add dx,48 ;Adding 48 to Convert Value

mov ah,02h ;Print the Value

int 21h

LEA DX,n\_line ;lea means least effective address

MOV AH,9 ;Print NEWline

INT 21H

lea dx,STRING ;Moving string to dx

mov ah,09h ;Print the Strings

int 21h

mov ax,[arrayD]

xchg ax,[arrayD+4] ;copy the first value into AX and exchange it with thevalue in the second position.

xchg ax,[arrayD+8] ;Exchange AX with the third array value and copy thevalue in AX to the first array position.

mov [arrayD],ax

mov dx,[arrayD] ;Moving first index to dx

add dx,48 ;Adding 48 to Convert into value

mov ah,02h ;Print the Value

int 21h

mov dx,[arrayD+4] ;Moving Second index to dx

add dx,48 ;Adding 48 to Convert into value

sub dx,35 ;Print the Value

mov ah,02h

int 21h

mov dx,[arrayD+8] ;Moving Third index to dx

add dx,48 ;Adding 48 to Convert into value

mov ah,02h ;Print the Value

int 21h

mov ah,4ch

int 21h

Graphical user interface, text

Description automatically generated

Using in Direct addressing:

.model small

.stack 100h

.data

arrayD dd 1,2,3

STRING db 'BEFORE REARRANGING : ', '$'

n\_line db 0AH,0DH,"$"

STRING2 db 'AFTER REARRANGING : ', '$'

.code

mov ax,@data

mov ds,ax

lea dx,STRING ;Moving string to dx

mov ah,09h ;Print the Strings

int 21h

mov arrayD,ax

mov dx,arrayD ;Moving first index to dx

add dx,48 ;Adding 48 to Convert into value

sub dx,35

mov ah,02h ;Print the Value

int 21h

mov dx,arrayD+4 ;Moving Second index to dx

add dx,48 ;Adding 48 to Convert into value

mov ah,02h ;Print the Value

int 21h

mov dx,arrayD+8 ;Moving Third index to dx

add dx,48 ;Adding 48 to Convert Value

mov ah,02h ;Print the Value

int 21h

LEA DX,n\_line ;lea means least effective address

MOV AH,9 ;Print NEWline

INT 21H

lea dx,STRING ;Moving string to dx

mov ah,09h ;Print the Strings

int 21h

mov ax,arrayD

xchg ax,arrayD+4 ;copy the first value into AX and exchange it with thevalue in the second position.

xchg ax,arrayD+8 ;Exchange AX with the third array value and copy thevalue in AX to the first array position.

mov arrayD,ax

mov dx,arrayD ;Moving first index to dx

add dx,48 ;Adding 48 to Convert into value

mov ah,02h ;Print the Value

int 21h

mov dx,arrayD+4 ;Moving Second index to dx

add dx,48 ;Adding 48 to Convert into value

sub dx,35 ;Print the Value

mov ah,02h

int 21h

mov dx,arrayD+8 ;Moving Third index to dx

add dx,48 ;Adding 48 to Convert into value

mov ah,02h ;Print the Value

int 21h

mov ah,4ch

int 21h

Graphical user interface, text

Description automatically generated

Question 3

We want to write a program that adds the following three bytes:

.data

myBytes BYTE 80h,66h,0A5h, 60

What is your evaluation of the following code?

**mov al, myBytes**

It will move 80h = 50 to the al memory location.

**add al, [myBytes]**

It will add 80h to the al memory location, which is already 80h and will become A0.

**add al, [myBytes+4]**

it will add 60 to the al memory location the 4th index in array, which is already A0 =160 and will become DC = 220.

What is your evaluation of the following code?

**mov ax, myBytes**

Statement didn’t Execute because ax is 16 bit register and mybytes is a 8 bit register so it will give an error No Value will be stored.

**add ax, [myBytes+1]**

Statement didn’t Execute because ax is 16 bit register and mybytes is a 8 bit register so it will give an error No Value will be stored.

**add ax, [myBytes+2]**

Statement didn’t Execute because ax is 16 bit register and mybytes is a 8 bit register so it will give an error No Value will be stored.

Question 4

Correct the following code snippest?

**movzx ax, myBytes**

This Statement will Store mybytes value 80h=50 to al memory location and extend 0 to ah register.

**mov bl, [myBytes+1]**

It will move 2nd Index value of array 66h to bl register as memory location size are same.

**add ax, bx**

This Statement will add bx, ax register but one mistake is that bx register must be set to 0 before previous statement to evaluate this line correctly.

**mov bl, [myBytes+2]**

It will move 3rd Index value of array 0A5h to bl register as memory location size are same.

**add ax, bx**

This Statement will add bx ,ax register but one mistake is that bx register must be set to 0 before Previous statement to evaluate this line correctly.

Question 5

**mov bx,0A69Bh**

This statement will store A69B to bx = 42651.

**movzx eax, bx**

This statement will store A69B to bx = 4265 and extend 0 to the other part of ax register upper 16 bit.

**movzx edx, bl**

This statement will store 9B to bl = 155 and will extend 0 to the other part.

**movzx cx, bl**

This statement will store 9B to cl = 155 and will extend 0 to the other part(ch).

Question 6

**mov bx,0D87Bh**

The above Statement will move D8 to bh and 78 to bl register.

**movsx eax, bx**

The above Statement will move D878 to ax register lower part and at upper part it will extend the F if MSB is greater than 7.

**movsx edx,bl**

The above Statement will move 78 to dl register and at upper part it will extend the F if MSB is greater than 7 and if not than 0.

**mov bl,7Bh**

The above Statement will move 7B to bl register.

**movsx cx, bl**

The above Statement will move 7B from bl to cl register and extend sign to ch register.

Question 7

data

val1 DW 800h

val2 DW 500h

.code

**mov ax, val1**

This Statement will move 800h to ax register because memory size is same 16 bit.

**xchg ax, val2**

This Statement will exchange value of ax, val2 ax will become 500h and val2 will 800h

**mov val1, ax**

This Statement will move 500h to val1 from ax register because memory size is same 16 bit.

Question 8

.data

arrayB DB 10000h,20000h

.code

**mov eax, arrayB**

This Statement will move 10000h to eax register because size of double word and eax is 32 bit.

**mov eax, [arrayB+4]**

This Statement will move 20000h to eax register because size of double word and eax is 32 bit.

**mov eax, [arrayB+4]**

This Statement will move 20000h to eax register because size of double word and eax is 32 bit.

Question 9

.data

arrayD DD 10000h,20000h

.code

mov eax, arrayD ;eax = 10000h

mov eax, [arrayD+4] ;eax = 20000h

mov eax, [arrayD+4] ;eax = 20000h

Question 10

Rval = (- Xval – (-(-Yval + Zval)) + Xval ) – Yval

.model small

.stack 100h

.data

Xval db 10

Yval db 20

Zval db 30

Rval db ?

str dw 'EVALUATION IS : ','$'

.code

;Rval = (- Xval - (-(-Yval + Zval)) + Xval ) - Yval

mov ax,@data

mov ds,ax

mov al,Xval

mov bl,Yval

mov cl,Zval

neg bl

add bl,cl

neg bl

neg bl

mov al,Xval

neg al

add al,bl

mov bl ,Yval

sub al,bl

mov Rval,al

mov dl,Rval

lea dx,str

mov ah,09h

int 21h

add dl,48

mov ah,02h

int 21h

mov ah,4ch

int 21h

A picture containing text

Description automatically generated

9. Declare an uninitialized array of 50 signed doublewords named dArray.

dArray SDWORD 50 DUP(?)

10. Declare a string variable containing the word “TEST” repeated 500 times.

Str BYTE 500 DUP(“TEST”)

11. Declare an array of 20 unsigned bytes named bArray and initialize all elements to zero

bArray BYTE 20 DUP(0)

Question 12

.model small

.stack 100h

.data

bigEndian db 12h,34h,56h,78h

littleEndian dd 4 dup("?")

STRING db 'In BIG ENDIAN : ', '$'

n\_line db 0AH,0DH,"$"

STRING2 db 'In LITTLE ENDIAN : ', '$'

.code

mov ax,@data

mov ds,ax

lea dx ,STRING

mov ah,09h

int 21h

mov dl,10h

mov dl, [bigEndian]

mov ah,02h

int 21h

mov dl, [bigEndian+1]

mov ah,02h

int 21h

mov dl, [bigEndian+2]

mov ah,02h

int 21h

mov dl, [bigEndian+3]

mov ah,02h

int 21h

LEA dx,n\_line ;lea means least effective address

MOV ah,9 ;Print NEWline

INT 21H

mov al,[bigEndian+3]

mov BYTE PTR [littleEndian],al

mov al,[bigEndian+2]

mov BYTE PTR [littleEndian+1],al

mov al,[bigEndian+1]

mov BYTE PTR [littleEndian+2],al

mov al,[bigEndian]

mov BYTE PTR [littleEndian+3],al

lea dx ,STRING2

mov ah,09h

int 21h

mov dx, [littleEndian]

mov ah,02h

int 21h

mov dx, [littleEndian+1]

mov ah,02h

int 21h

mov dx, [littleEndian+2]

mov ah,02h

int 21h

mov dx, [littleEndian+3]

mov ah,02h

int 21h

mov ah,4ch

int 21h

Graphical user interface, text, application

Description automatically generated