

Task 1:

Compose a code in assembly language to increment and decrement the values stored in registers. Show

your results from the debugger that how the values are being manipulated.

A) Store a constant; say 3 in AX and then increment it once. Store the incremented value of AX into

BX

B) Manipulate the values in the registers given in the task above such that the result in BX is the

same as that of original value AX using decrement mnemonic.

The Code is explained in the comments written with it line by line.

```

1  [org 0x0100]
2
3  mov ax,3      ; move 3 inside ax
4  inc ax        ; increment the value by 1 in ax
5
6  mov bx,ax     ; move the value of ax in bx
7
8  dec bx        ; now decrement the value by 1 in bx
9
10 mov ax, 0x4c00 ;to exit
11
12 int 0x21      ; interrupt

```

Now To run this code I first have to mount a virtual disk so it becomes easier for me to access the content inside it. For that reason im mounting x where my files of nasm afd and the code that I want to run is present.

Then I compiled and opened the executable file for afd

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

Welcome to DOSBox v0.74-3
For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>mount x E:\AssemblyCL
Drive X is mounted as local directory E:\AssemblyCL\
Z:\>X:\
X:\>nasm Task1L3.asm -o Task1L3.com
X:\>afd Task1L3.com_

```

Now the first line as it says in the afd it puts 3 into AX

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	0003	SI	0000	CS	19F5	IP	0103	Stack	+0 0000	Flags	7200
BX	0000	DI	0000	DS	19F5				+2 20CD		
CX	0000	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	IF	SF

CMD >

0100	B80300	MOV	AX,0003
0103	40	INC	AX
0104	89C3	MOV	BX,AX
0106	4B	DEC	BX
0107	B8004C	MOV	AX,4C00
010A	CD21	INT	21
010C	89C3	MOV	BX,AX
010E	89D0	MOV	AX,DX
0110	89DA	MOV	DX,BX

DS:0000	CD 20 FF 9F 00 EA F0 FE	AD DE 1B 05 C5 06 00 00	= f.Ω= i.+....
DS:0010	18 01 10 01 18 01 92 01	01 01 01 00 02 FF FF FFf.
DS:0020	FF FF FF FF FF FF FF FF	FF FF FF FF EB 19 C0 11δ. L.
DS:0030	A2 01 14 00 18 00 F5 19	FF FF FF FF 00 00 00 00	δ.....J.
DS:0040	05 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Now the value with in ax register is incremented that means 1 is added with it.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	0004	SI	0000	CS	19F5	IP	0104	Stack	+0 0000	Flags	7200
BX	0000	DI	0000	DS	19F5				+2 20CD		
CX	0000	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	IF	SF

CMD >

0103	40	INC	AX
0104	89C3	MOV	BX,AX
0106	4B	DEC	BX
0107	B8004C	MOV	AX,4C00
010A	CD21	INT	21
010C	89C3	MOV	BX,AX
010E	89D0	MOV	AX,DX
0110	89DA	MOV	DX,BX
0112	EB04	JMP	0118

DS:0000	CD 20 FF 9F 00 EA F0 FE	AD DE 1B 05 C5 06 00 00	= f.Ω= i.+....
DS:0010	18 01 10 01 18 01 92 01	01 01 01 00 02 FF FF FFf.
DS:0020	FF FF FF FF FF FF FF FF	FF FF FF FF EB 19 C0 11δ. L.
DS:0030	A2 01 14 00 18 00 F5 19	FF FF FF FF 00 00 00 00	δ.....J.
DS:0040	05 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Now as the value of ax was incremented and it become 4 we copied ax 's value which was 4 to bx and now ax and bx has the same value which is 4.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	0004	SI	0000	CS	19F5	IP	0106	Stack	+0 0000	Flags	7200
BX	0004	DI	0000	DS	19F5				+2 20CD		
CX	0000	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	IF	SF

CMD >

0104	89C3	MOV	BX,AX
0106	4B	DEC	BX
0107	B8004C	MOV	AX,4C00
010A	CD21	INT	21
010C	89C3	MOV	BX,AX
010E	89D0	MOV	AX,DX
0110	89DA	MOV	DX,BX
0112	EB04	JMP	0118
0114	31D2	XOR	DX,DX

DS:0000	CD 20 FF 9F 00 EA F0 FE	AD DE 1B 05 C5 06 00 00	= f.Ω= i.+....
DS:0010	18 01 10 01 18 01 92 01	01 01 01 00 02 FF FF FFf.
DS:0020	FF FF FF FF FF FF FF FF	FF FF FF FF EB 19 C0 11δ. L.
DS:0030	A2 01 14 00 18 00 F5 19	FF FF FF FF 00 00 00 00	δ.....J.
DS:0040	05 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Now we decrement the value of bx which had previously 4 in it is decremented and has 3 in it now.

For Terminating the code we put 0x4c00 inside ax and then interrupt the code that makes it to read the 0x4c00 and then ends the code.

Task 2:

Write instructions to perform the following operations.

a. Copy BL into CL

b. Copy DX into AX

c. Store 0x12 into AL

d. Store 0x1234 into AX

e. Store 0xFFFF into AX

Hint: Store some constant value before copying it somewhere else.

This is the code for the second task and it is self explanatory as ive stated what is happening in each line.

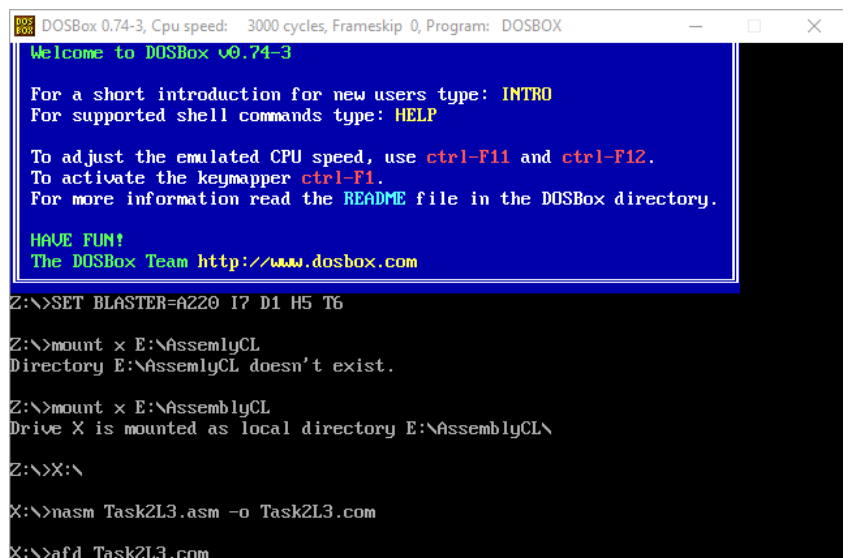
```

1  [org 0x0100]
2
3  mov bl,15 ;keeping a constant in bl
4
5  mov cl,bl ;moving or copying bl into cl
6
7  mov dx,16 ; keeping a constant in dx
8
9  mov ax,dx ; copying the values of dx into ax
10
11 mov al,0x12 ; storing 0x12 into al
12
13 mov ax, 0x1234 ;storing 1x1234 into ax
14
15 mov ax,0xFFFF ; storing 0xFFFF into ax
16
17
18 mov ax, 0x4c00 ;to exit
19
20 int 0x21 ; interrupt
21

```

Now im compiling and making the executable file for the afd to open.

After which I open up afd.



```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Welcome to DOSBox v0.74-3

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>mount x E:\AssemblyCL
Directory E:\AssemblyCL doesn't exist.
Z:\>mount x E:\AssemblyCL
Drive X is mounted as local directory E:\AssemblyCL\
Z:\>X:\
X:\>nasm Task2L3.asm -o Task2L3.com
X:\>afd Task2L3.com_

```

First of all im putting 15 into BL but in the terminal F is shown which is also right as the hexadecimal of 15 is F

After which im putting the value of BL into CL so now CL will also show us F which is 15.

Now here im putting 16 (hexadecimal = 12) into DX.

Now as the panel shows im copying the value of DX into AX so now after this code is executed AX will also show us 0010 (which is 16 in decimal)

Putting hexadecimal value 12 into AL
which is the lower half of the Ax register

And So the value will be stored in the
least significant place.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	0010	SI	0000	CS	19F5	IP	0109	Stack	+0 0000	Flags	7200
BX	000F	DI	0000	DS	19F5				+2 20CD		
CX	000F	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0010	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	0	0

CMD >

```

0107 89D0      MOV     AX,DX
0109 B012      MOV     AL,12
010B B83412    MOV     AX,1234
010E B8FFFF    MOV     AX,FFFF
0111 B8004C    MOV     AX,4C00
0114 CD21      INT     21
0116 31C0      XOR     AX,AX
0118 8956E4    MOV     [BP-1C],DX
011B 8946E6    MOV     [BP-1A],AX
  
```

DS:0000	CD 20 FF 9F 00 EA F0 FE	AD DE 1B 05 C5 06 00 00	= f.Ω= i.+....
DS:0010	18 01 10 01 18 01 92 01	01 01 01 00 02 FF FF FFf.
DS:0020	FF FF FF FF FF FF FF FF	FF FF FF FF EB 19 C0 11δ. L.
DS:0030	A2 01 14 00 18 00 F5 19	FF FF FF FF 00 00 00 00	6.....J.
DS:0040	05 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Now im putting the hexadecimal value
(1234) inside ax

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	0012	SI	0000	CS	19F5	IP	010B	Stack	+0 0000	Flags	7200
BX	000F	DI	0000	DS	19F5				+2 20CD		
CX	000F	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0010	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	0	0

CMD >

```

0109 B012      MOV     AL,12
010B B83412    MOV     AX,1234
010E B8FFFF    MOV     AX,FFFF
0111 B8004C    MOV     AX,4C00
0114 CD21      INT     21
0116 31C0      XOR     AX,AX
0118 8956E4    MOV     [BP-1C],DX
011B 8946E6    MOV     [BP-1A],AX
011E C746F60000 MOV     [BP-0A],0000
  
```

DS:0000	CD 20 FF 9F 00 EA F0 FE	AD DE 1B 05 C5 06 00 00	= f.Ω= i.+....
DS:0010	18 01 10 01 18 01 92 01	01 01 01 00 02 FF FF FFf.
DS:0020	FF FF FF FF FF FF FF FF	FF FF FF FF EB 19 C0 11δ. L.
DS:0030	A2 01 14 00 18 00 F5 19	FF FF FF FF 00 00 00 00	6.....J.
DS:0040	05 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Now im putting hexadecimal value (FFFF)
into AX replacing the previous value.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	1234	SI	0000	CS	19F5	IP	010E	Stack	+0 0000	Flags	7200
BX	000F	DI	0000	DS	19F5				+2 20CD		
CX	000F	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0010	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	0	0

CMD >

```

010B B83412    MOV     AX,1234
010E B8FFFF    MOV     AX,FFFF
0111 B8004C    MOV     AX,4C00
0114 CD21      INT     21
0116 31C0      XOR     AX,AX
0118 8956E4    MOV     [BP-1C],DX
011B 8946E6    MOV     [BP-1A],AX
011E C746F60000 MOV     [BP-0A],0000
0123 8B46F6      MOV     AX,[BP-0A]
  
```

DS:0000	CD 20 FF 9F 00 EA F0 FE	AD DE 1B 05 C5 06 00 00	= f.Ω= i.+....
DS:0010	18 01 10 01 18 01 92 01	01 01 01 00 02 FF FF FFf.
DS:0020	FF FF FF FF FF FF FF FF	FF FF FF FF EB 19 C0 11δ. L.
DS:0030	A2 01 14 00 18 00 F5 19	FF FF FF FF 00 00 00 00	6.....J.
DS:0040	05 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

In these two lines the code executes and exits as we put 0x4C00 into AX that means it has to end the code

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	FFFF	SI	0000	CS	19F5	IP	0111	Stack	+0 0000	Flags	7200
BX	000F	DI	0000	DS	19F5				+2 20CD		
CX	000F	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0010	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	0	0
										1	0
										2	0
										3	0
										4	0
										5	0
										6	0
										7	0

CMD >

010E	B8FFFF	MOV	AX,FFFF	DS:0100	B3 0F 88 D9 BA 10 00 89
0111	B8004C	MOV	AX,4C00	DS:0108	D0 B0 12 B8 34 12 B8 FF
0114	CD21	INT	21	DS:0110	FF B8 00 4C CD 21 31 C0
0116	31C0	XOR	AX,AX	DS:0118	89 56 E4 89 46 E6 C7 46
0118	8956E4	MOV	[BP-1C],DX	DS:0120	F6 00 00 8B 46 F6 D1 E0
011B	8946E6	MOV	[BP-1A],AX	DS:0128	D1 E0 C5 5E D8 01 C3 8B
011E	C746F60000	MOV	[BP-0A],0000	DS:0130	07 8B 57 02 85 D2 75 04
0123	8B46F6	MOV	AX,[BP-0A]	DS:0138	85 C0 74 1C C7 46 DC 00
0126	D1E0	SHL	AX,1	DS:0140	00 8E 5E FC 83 7D 0E 00
				DS:0148	74 09 8B 46 F2 48 3B 46

2

DS:0000	CD 20 FF 9F 00 EA F0 FE	AD DE 1B 05 C5 06 00 00	= f.Ω≡ i .+....
DS:0010	18 01 10 01 18 01 92 01	01 01 01 00 02 FF FF FFf.
DS:0020	FF FF FF FF FF FF FF FF	FF FF FF FF EB 19 C0 11δ. L.
DS:0030	A2 01 14 00 18 00 F5 19	FF FF FF FF 00 00 00 00	6.....J.
DS:0040	05 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Now we interrupt the code and send it back to the value of AX and that had 4c00 which will eventually end the code.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	4C00	SI	0000	CS	19F5	IP	0114	Stack	+0 0000	Flags	7200
BX	000F	DI	0000	DS	19F5				+2 20CD		
CX	000F	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0010	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	0	0
										1	0
										2	0
										3	0
										4	0
										5	0
										6	0
										7	0

CMD >

0111	B8004C	MOV	AX,4C00	DS:0100	B3 0F 88 D9 BA 10 00 89
0114	CD21	INT	21	DS:0108	D0 B0 12 B8 34 12 B8 FF
0116	31C0	XOR	AX,AX	DS:0110	FF B8 00 4C CD 21 31 C0
0118	8956E4	MOV	[BP-1C],DX	DS:0118	89 56 E4 89 46 E6 C7 46
011B	8946E6	MOV	[BP-1A],AX	DS:0120	F6 00 00 8B 46 F6 D1 E0
011E	C746F60000	MOV	[BP-0A],0000	DS:0128	D1 E0 C5 5E D8 01 C3 8B
0123	8B46F6	MOV	AX,[BP-0A]	DS:0130	07 8B 57 02 85 D2 75 04
0126	D1E0	SHL	AX,1	DS:0138	85 C0 74 1C C7 46 DC 00
012B	D1E0	SHL	AX,1	DS:0140	00 8E 5E FC 83 7D 0E 00
				DS:0148	74 09 8B 46 F2 48 3B 46

2

DS:0000	CD 20 FF 9F 00 EA F0 FE	AD DE 1B 05 C5 06 00 00	= f.Ω≡ i .+....
DS:0010	18 01 10 01 18 01 92 01	01 01 01 00 02 FF FF FFf.
DS:0020	FF FF FF FF FF FF FF FF	FF FF FF FF EB 19 C0 11δ. L.
DS:0030	A2 01 14 00 18 00 F5 19	FF FF FF FF 00 00 00 00	6.....J.
DS:0040	05 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Task 3:

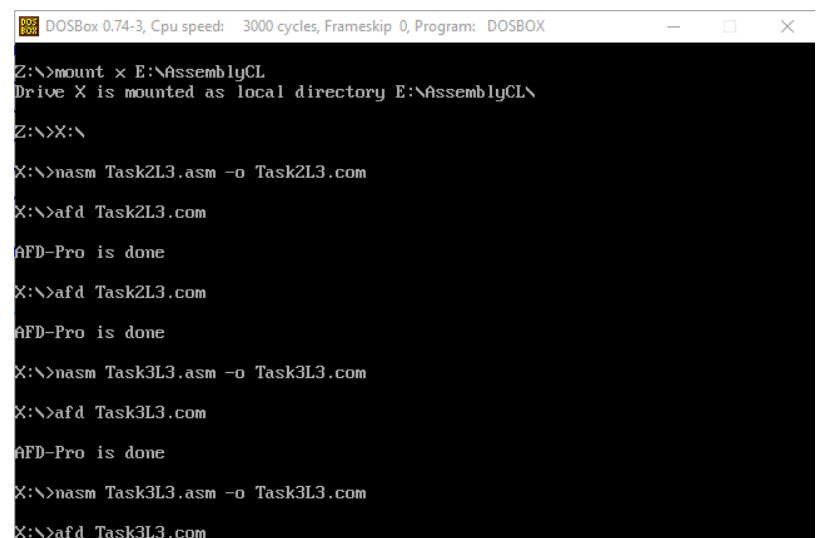
Write a program in assembly language that calculates the square of six by adding six to the accumulator

(AX). Only use ADD operation and determine how many times you need to add to get the required result.

The Code is Self-Explanatory as I've added comments with each line whatever is happening.

```
1  [org 0x0100]
2
3  mov ax,6 ;move 6 into ax
4  add ax,6 ; add 6 with 6 = 12
5  add ax,6 ; add 6 again with 12 = 18
6  add ax,6 ; add 6 with 18 = 24
7  add ax,6 ; add 6 with 24 = 30
8  add ax,6 ; add 6 with 30 = 36
9
10 mov ax,0x4c00
11
12 int 0x21
13
```

I'm doing the same stuff as explained above compiling and then making the executable file for the afd to run.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Z:\>mount x E:\AssemblyCL
Drive X is mounted as local directory E:\AssemblyCL\
Z:\>X:\
X:\>nasm Task2L3.asm -o Task2L3.com
X:\>afd Task2L3.com
AFD-Pro is done
X:\>afd Task2L3.com
AFD-Pro is done
X:\>nasm Task3L3.asm -o Task3L3.com
X:\>afd Task3L3.com
AFD-Pro is done
X:\>nasm Task3L3.asm -o Task3L3.com
X:\>afd Task3L3.com
```


So I placed 6 in ax and then add 6 with ax.

As you can see we previously added 6 with 6 in ax that becomes 12 which can be seen in the ax register it became C which is the hexadecimal of 12

Now we add 6 in the same AX which has 12 in it making the value in the register (12 (Hexadecimal)) which becomes 18.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	0006	SI	0000	CS	19F5	IP	0103	Stack	+0 0000	Flags	7200
BX	0000	DI	0000	DS	19F5				+2 20CD		
CX	0017	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	IF	SF

CMD >

0100	B80600	MOV	AX,0006	DS:0000	CD 20 FF 9F 00 EA F0 FE
0103	050600	ADD	AX,0006	DS:0008	AD DE 1B 05 C5 06 00 00
0106	050600	ADD	AX,0006	DS:0010	18 01 10 01 18 01 92 01
0109	050600	ADD	AX,0006	DS:0018	01 01 01 00 02 FF FF FF
010C	050600	ADD	AX,0006	DS:0020	FF FF FF FF FF FF FF FF
010F	050600	ADD	AX,0006	DS:0028	FF FF FF FF EB 19 C0 11
0112	B8004C	MOV	AX,4C00	DS:0030	A2 01 14 00 18 00 F5 19
0115	CD21	INT	21	DS:0038	FF FF FF FF 00 00 00 00
0117	C08956E489	ROR	B/[E456+BX+DI],89	DS:0040	05 00 00 00 00 00 00 00
				DS:0048	00 00 00 00 00 00 00 00

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	000C	SI	0000	CS	19F5	IP	0106	Stack	+0 0000	Flags	7204
BX	0000	DI	0000	DS	19F5				+2 20CD		
CX	0017	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	IF	SF

CMD >

0103	050600	ADD	AX,0006	DS:0000	CD 20 FF 9F 00 EA F0 FE
0106	050600	ADD	AX,0006	DS:0008	AD DE 1B 05 C5 06 00 00
0109	050600	ADD	AX,0006	DS:0010	18 01 10 01 18 01 92 01
010C	050600	ADD	AX,0006	DS:0018	01 01 01 00 02 FF FF FF
010F	050600	ADD	AX,0006	DS:0020	FF FF FF FF FF FF FF FF
0112	B8004C	MOV	AX,4C00	DS:0028	FF FF FF FF EB 19 C0 11
0115	CD21	INT	21	DS:0030	A2 01 14 00 18 00 F5 19
0117	C08956E489	ROR	B/[E456+BX+DI],89	DS:0038	FF FF FF FF 00 00 00 00
011C	46	INC	SI	DS:0040	05 00 00 00 00 00 00 00
				DS:0048	00 00 00 00 00 00 00 00

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	0012	SI	0000	CS	19F5	IP	0109	Stack	+0 0000	Flags	7214
BX	0000	DI	0000	DS	19F5				+2 20CD		
CX	0017	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	IF	SF

CMD >

0106	050600	ADD	AX,0006	DS:0000	CD 20 FF 9F 00 EA F0 FE
0109	050600	ADD	AX,0006	DS:0008	AD DE 1B 05 C5 06 00 00
010C	050600	ADD	AX,0006	DS:0010	18 01 10 01 18 01 92 01
010F	050600	ADD	AX,0006	DS:0018	01 01 01 00 02 FF FF FF
0112	B8004C	MOV	AX,4C00	DS:0020	FF FF FF FF FF FF FF FF
0115	CD21	INT	21	DS:0028	FF FF FF FF EB 19 C0 11
0117	C08956E489	ROR	B/[E456+BX+DI],89	DS:0030	A2 01 14 00 18 00 F5 19
011C	46	INC	SI	DS:0038	FF FF FF FF 00 00 00 00
011D	E6C7	OUT	IC71,AL	DS:0040	05 00 00 00 00 00 00 00
				DS:0048	00 00 00 00 00 00 00 00

Now we add 6 again in the AX register which has 18 now and making it 24 which can be seen in the AX register as 18 which is a hexadecimal value and its decimal is the same as 24.

Now we add 6 again with the previously value of 24 in the AX register making it 30 which can be seen in the register having 1E which's decimal value is 30.

Now you can see the last time we added 6 with the AX register having the value of 30 making it 36 whose hexadecimal value is 24 as being shown in the register.

And Like This we achieve the square root of 6 by adding 6 times 6 with 6 which was initially in AX register.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	0018	SI	0000	CS	19F5	IP	010C	Stack	+0 0000	Flags	7204
BX	0000	DI	0000	DS	19F5				+2 20CD		
CX	0017	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	IF	SF

CMD >

0109	050600	ADD	AX,0006	DS:0000	CD 20 FF 9F 00 EA F0 FE
010C	050600	ADD	AX,0006	DS:0008	AD DE 1B 05 C5 06 00 00
010F	050600	ADD	AX,0006	DS:0010	18 01 10 01 18 01 92 01
0112	B8004C	MOV	AX,4C00	DS:0018	01 01 01 00 02 FF FF FF
0115	CD21	INT	21	DS:0020	FF FF FF FF FF FF FF FF
0117	C08956E489	ROR	B/[E456+BX+DI],89	DS:0028	FF FF FF FF EB 19 C0 11
011C	46	INC	SI	DS:0030	FF FF FF FF 00 00 00 00
011D	E6C7	OUT	[C7],AL	DS:0038	05 00 00 00 00 00 00 00
011F	46	INC	SI	DS:0048	00 00 00 00 00 00 00 00

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	001E	SI	0000	CS	19F5	IP	010F	Stack	+0 0000	Flags	7204
BX	0000	DI	0000	DS	19F5				+2 20CD		
CX	0017	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	IF	SF

CMD >

010C	050600	ADD	AX,0006	DS:0000	CD 20 FF 9F 00 EA F0 FE
010F	050600	ADD	AX,0006	DS:0008	AD DE 1B 05 C5 06 00 00
0112	B8004C	MOV	AX,4C00	DS:0010	18 01 10 01 18 01 92 01
0115	CD21	INT	21	DS:0018	01 01 01 00 02 FF FF FF
0117	C08956E489	ROR	B/[E456+BX+DI],89	DS:0020	FF FF FF FF FF FF FF FF
011C	46	INC	SI	DS:0028	FF FF FF FF EB 19 C0 11
011D	E6C7	OUT	[C7],AL	DS:0030	A2 01 14 00 18 00 F5 19
011F	46	INC	SI	DS:0038	FF FF FF FF 00 00 00 00
0120	F60000	TEST	[BX+SI],00	DS:0040	05 00 00 00 00 00 00 00

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	0024	SI	0000	CS	19F5	IP	0112	Stack	+0 0000	Flags	7214
BX	0000	DI	0000	DS	19F5				+2 20CD		
CX	0017	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	IF	SF

CMD >

010F	050600	ADD	AX,0006	DS:0000	CD 20 FF 9F 00 EA F0 FE
0112	B8004C	MOV	AX,4C00	DS:0008	AD DE 1B 05 C5 06 00 00
0115	CD21	INT	21	DS:0010	18 01 10 01 18 01 92 01
0117	C08956E489	ROR	B/[E456+BX+DI],89	DS:0018	01 01 01 00 02 FF FF FF
011C	46	INC	SI	DS:0020	FF FF FF FF FF FF FF FF
011D	E6C7	OUT	[C7],AL	DS:0028	FF FF FF FF EB 19 C0 11
011F	46	INC	SI	DS:0030	A2 01 14 00 18 00 F5 19
0120	F60000	TEST	[BX+SI],00	DS:0038	FF FF FF FF 00 00 00 00
0123	8B46F6	MOV	AX,[BP-0A]	DS:0040	05 00 00 00 00 00 00 00

And in this screenshot in the same way I interrupt the code to end it.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX	4C00	SI	0000	CS	19F5	IP	0115	Stack	+0 0000	Flags	7214
BX	0000	DI	0000	DS	19F5				+2 20CD		
CX	0017	BP	0000	ES	19F5	HS	19F5		+4 9FFF	OF	DF IF SF ZF AF PF CF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6 EA00	0	0 1 0 0 1 1 0

CMD >			0	1	2	3	4	5	6	7
0112	B8004C	MOV	AX,4C00	DS:0000	CD	20	FF	9F	00	EA F0 FF
0115	CD21	INT	21	DS:0008	AD	DE	1B	05	C5	06 00 00
0117	C08956E489	ROR	B/[E456+BX+DI],89	DS:0010	18	01	10	01	18	01 92 01
011C	46	INC	SI	DS:0018	01	01	01	00	02	FF FF FF
011D	E6C7	OUT	[C7],AL	DS:0020	FF	FF	FF	FF	FF	FF FF FF
011F	46	INC	SI	DS:0028	FF	FF	FF	FF	EB	19 C0 11
0120	F60000	TEST	[BX+SI],00	DS:0030	A2	01	14	00	18	00 F5 19
0123	8B46F6	MOV	AX,[BP-0A]	DS:0038	FF	FF	FF	FF	00	00 00 00
0126	D1E0	SHL	AX,1	DS:0040	05	00	00	00	00	00 00 00
				DS:0048	00	00	00	00	00	00 00 00

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DS:0000	CD	20	FF	9F	00	EA	F0	FE	AD	DE	1B	05	C5	06	00	00
DS:0010	18	01	10	01	18	01	92	01	01	01	01	00	02	FF	FF	FF
DS:0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	EB	19	C0	11
DS:0030	A2	01	14	00	18	00	F5	19	FF	FF	FF	FF	00	00	00	00
DS:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

1 Step 2 ProcStep 3 Retrieve 4 Help ON 5 BRK Menu 6 7 up 8 dn 9 le 10 ri

This Screenshot is just to show that 24 is a hexadecimal number and the decimal value of it is 36 so we did the code right.

Conversion

ASCII: \$

Copy

Insert

Decimal: 36

Copy

Insert

Hexadecimal: 24

Copy

Insert

Binary: 100100

Copy

Insert

Octal: 44

Copy

Insert

Task 4:

For each of the following words identify the byte that is stored at lower memory address and the byte that is stored at higher memory address. Explain the reason of storage in

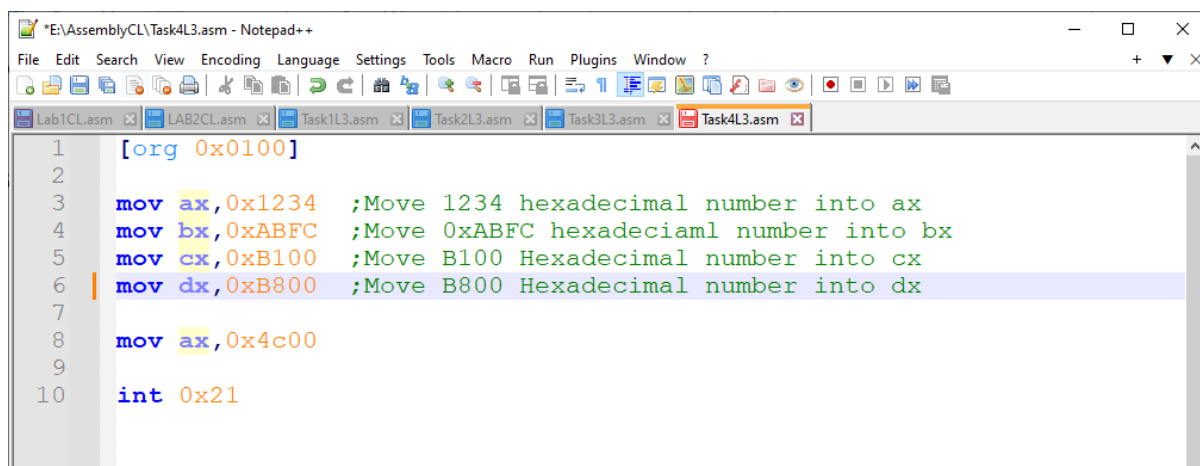
such manner with the help of your results.

a. 1234

b. ABFC

c. B100

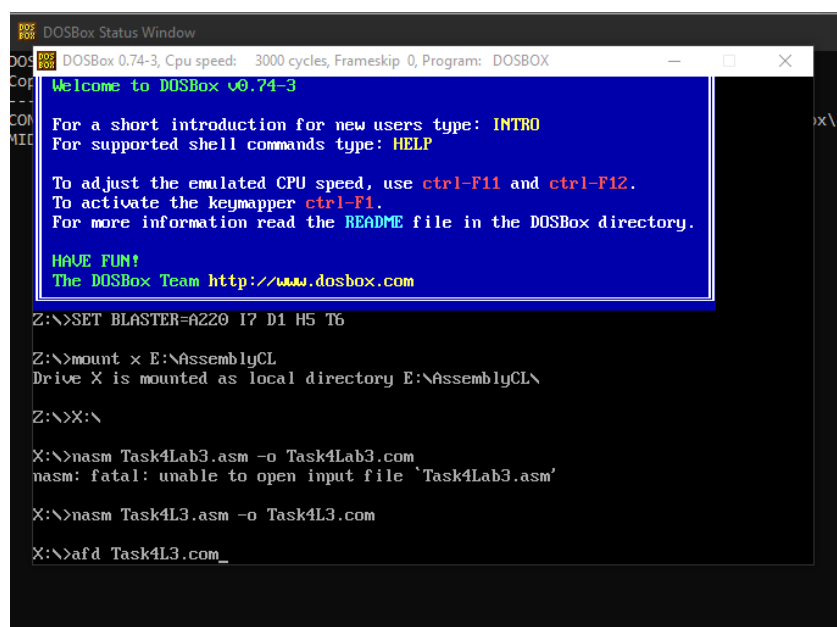
d. B800



```
*E:\AssemblyCL\Task4L3.asm - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Lab1CL.asm LAB2CL.asm Task1L3.asm Task2L3.asm Task3L3.asm Task4L3.asm
1 [org 0x0100]
2
3 mov ax,0x1234 ;Move 1234 hexadecimal number into ax
4 mov bx,0xABFC ;Move 0xABFC hexadecimal number into bx
5 mov cx,0xB100 ;Move B100 Hexadecimal number into cx
6 mov dx,0xB800 ;Move B800 Hexadecimal number into dx
7
8 mov ax,0x4c00
9
10 int 0x21
```

In the Above code I simply stored the hexadecimal values in the registers Also explained well in the comments.

Now it's the exactly same step that we've done before to make the file ready and make an executable file for the afd to open it.



```
DOSBox Status Window
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Welcome to DOSBox v0.74-3
For a short introduction for new users type: INTRO
For supported shell commands type: HELP
To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.
HAVE FUN!
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>mount x E:\AssemblyCL
Drive X is mounted as local directory E:\AssemblyCL\
Z:\>X:\
X:\>nasm Task4Lab3.asm -o Task4Lab3.com
nasm: fatal: unable to open input file 'Task4Lab3.asm'
X:\>nasm Task4L3.asm -o Task4L3.com
X:\>afd Task4L3.com_
```

```

1  [org 0x0100]
2
3  mov ax,0x1234 ;Move 1234 hexadecimal number into ax
4  mov bx,0xABFC ;Move 0xABFC hexadecimal number into bx
5  mov cx,0xB100 ;Move B100 Hexadecimal number into cx
6  mov dx,0xB800 ;Move B800 Hexadecimal number into dx
7
8  mov ax,0x4c00
9
10 int 0x21

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

Register	Value	Segment	Value	Register	Value	Segment	Value
AX	1234	SI	0000	CS	19F5	IP	010C
BX	ABFC	DI	0000	DS	19F5	Stack	+0 0000
CX	B100	BP	0000	ES	19F5	Flags	7200
DX	B800	SP	FFFE	FS	19F5		

CMD >

Address	Op Code	Instruction	Comment
0109	B8 00 B8	MOV	DX, B800
010C	B8 00 4C	MOV	AX, 4C00
010F	CD 21	INT	21
0111	DA EB	ESC	15, BL
0113	04 31	ADD	AL, 31
0115	D2	DB	D2
0116	31 C0	XOR	AX, AX
0118	89 56 E4	MOV	[BP-1C], DX
011B	89 46 E6	MOV	[BP-1A], AX

Address	0	1	2	3	4	5	6	7
DS:0100	B8	34	12	BB	FC	AB	B9	00
DS:0103	B1	BA	00	B8	00	4C	CD	

Short Explanation :-

By typing the command m1 0100 in the CMD brings me to the address of 0100 and ive squared the 4 different commands with the first values B8 BB B9 And B1 representing op code and the next respective two values in each block shows the value stored in it and you may see that they are all swapped this is because of the architecture it is using Little Indian in which the least significant byte is stored before the most significant byte.

Longer Explanation :-

The Computer Architecture is following Little Indian and that's the reason that it is writing the least significant byte first and the most significant byte later as we can see in all the registers look at the address of the register in the bigger column on the right.

At 0100 address at 0th position B8 is the op code and then at 1th and 2nd position you can see that 1234 is swapped and at 1th 34 is stored and at 2nd 12 is stored which is because of the architecture following little Indian and writes the lsb before msb.

Similarly at 0103 which is shown as 3rd position you can see the op code for the next command that the number which we stored was ABFC but it is shown FC at 4th position and AB at 5th position

which is again because of the same reason that the architecture is following little Indian and storing the least significant byte at the starting and storing the most significant byte after it.

Similarly we can see the same behavior at 0106 which is shown as 6th position you can see that B9 is the op code and after which at 7th position the least significant byte is stored which is 00 and after that B1 is stored but we stored it as B100 but it is shown that the bytes swapped which is because of the architecture following little Indian.

Similarly the last value B800 can also be seen swapped in a way that 00 can be seen first and B8 can be seen after that.

-----THE END-----