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.

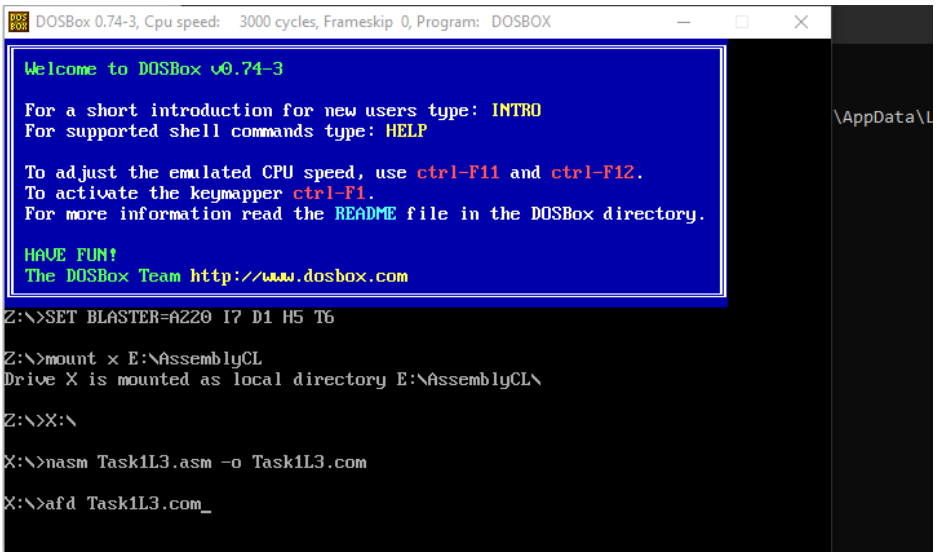
The Code is explained in the comments written with it line by line.

Lab1CL.asm | LAB2CL.asm | Task1LAB3.asm

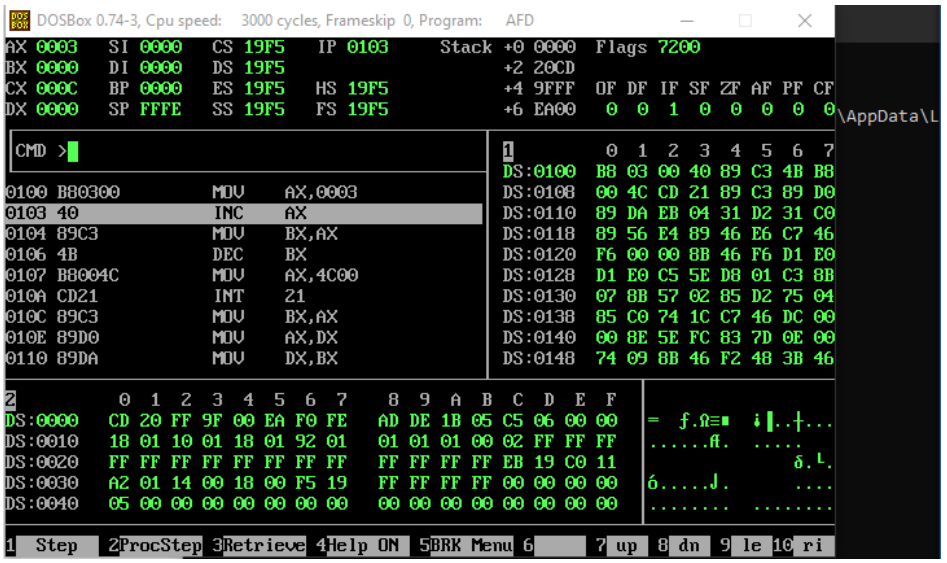
```
1   [org 0x0100]
2
3   mov ax,3        ; move 3 inside ax
4   inc ax          ; increment the value by 1 in ax
5
6   mov bx,ax       ; move the value of ax in bx
7
8   dec bx          ; now decrement the value by 1 in bx
9
10  mov ax, 0x4c00   ;to exit
11
12  int 0x21        ; interrupt
```

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.

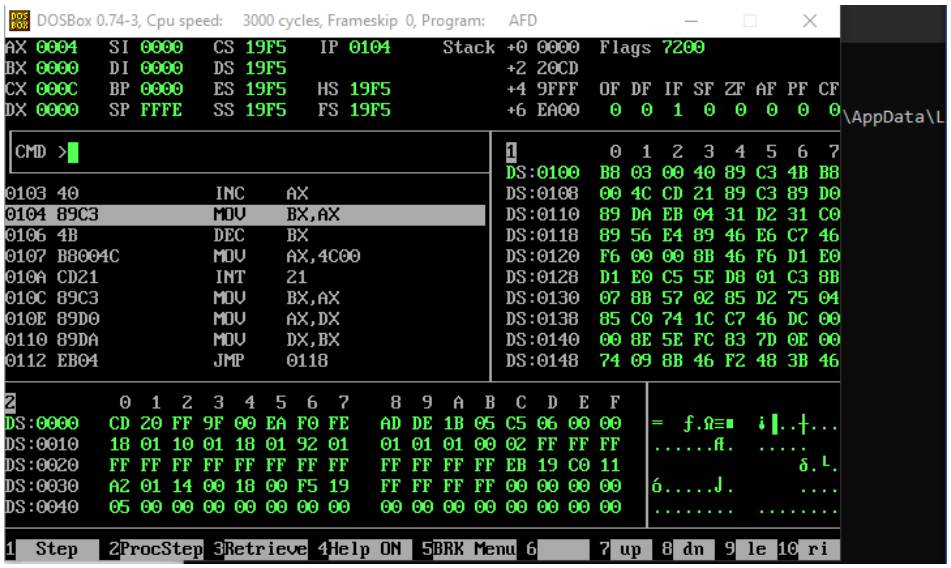
Then I compiled and opened the executable file for afd



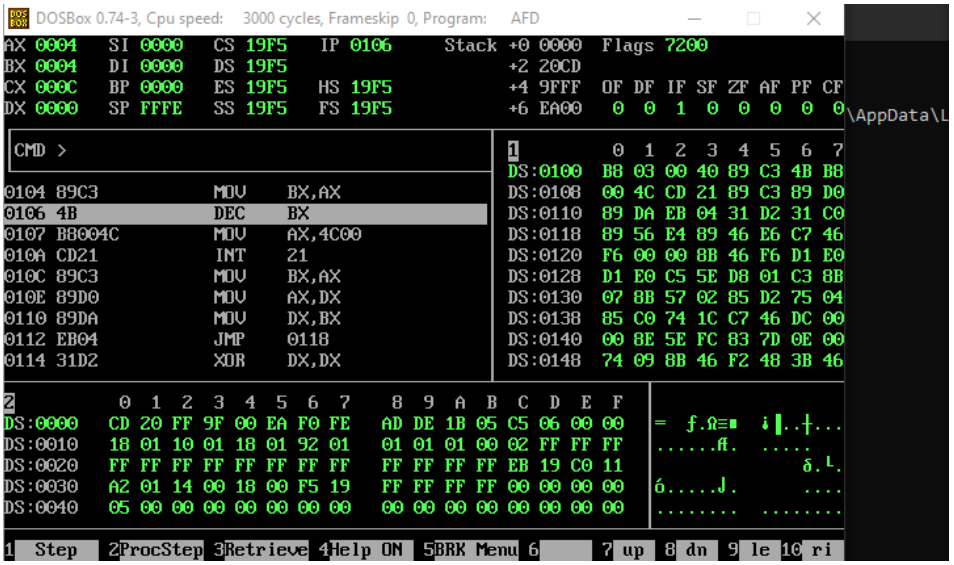
Now the first line as it says in the afd it puts 3 into AX



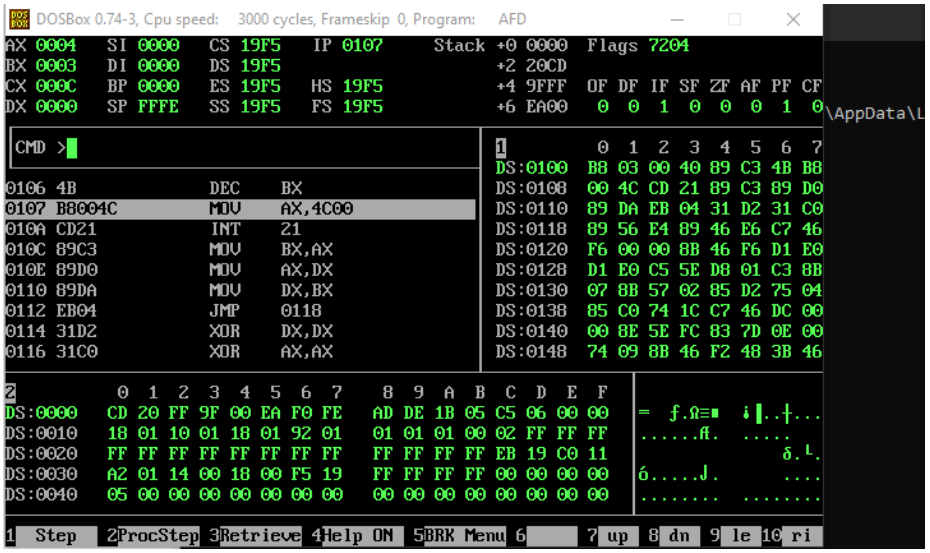
Now the vale with in ax register is incremented that means 1 is added with it.



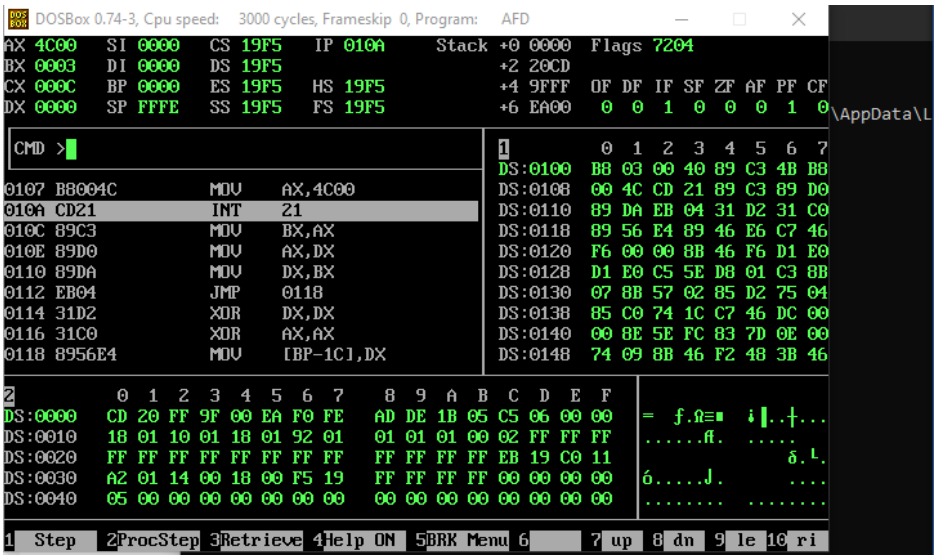
Now as the value of ax was incremented and it become 4 we copied ax ‘s value which was 4 to bx and now ax and bx has the same value which is 4.



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

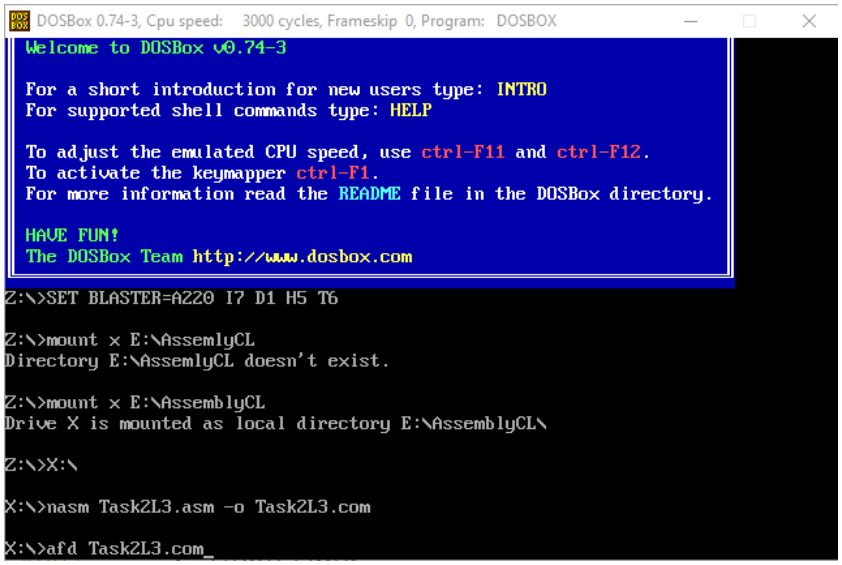
Hint: Store some constant value before copying it somewhere else.

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

```
[org 0x0100]

mov bl,15 ;keeping a constant in bl

mov cl,bl ;moving or copying bl into cl

mov dx,16 ; keeping a constant in dx

mov ax,dx  ; copying the values of dx into ax

mov al,0x12 ; storing ox12 into al

mov ax, 0x1234 ;storing 1x1234 into ax

mov ax,0xFFFF  ; storing 0xFFFF into ax


mov ax, 0x4c00  ;to exit

int 0x21     ; interrupt

```

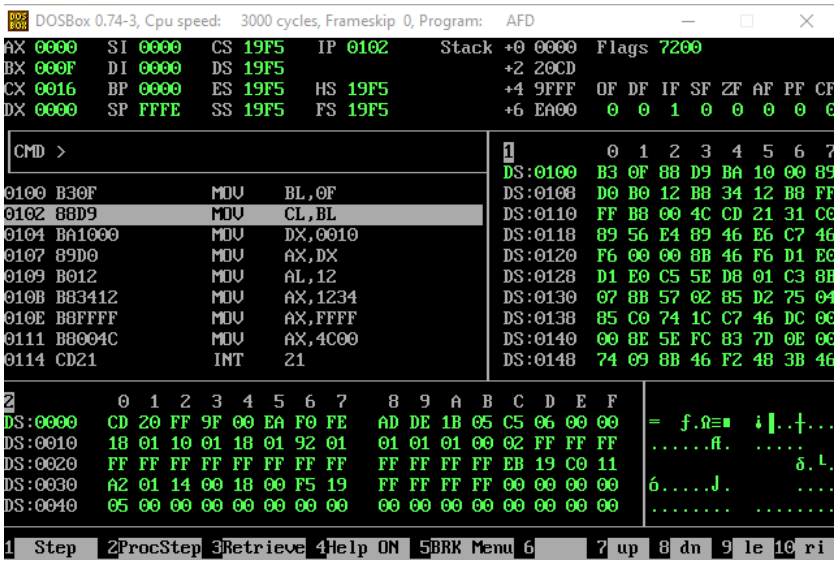
Now im compiling and making the executable file for the afd to open.

After which I open up afd.

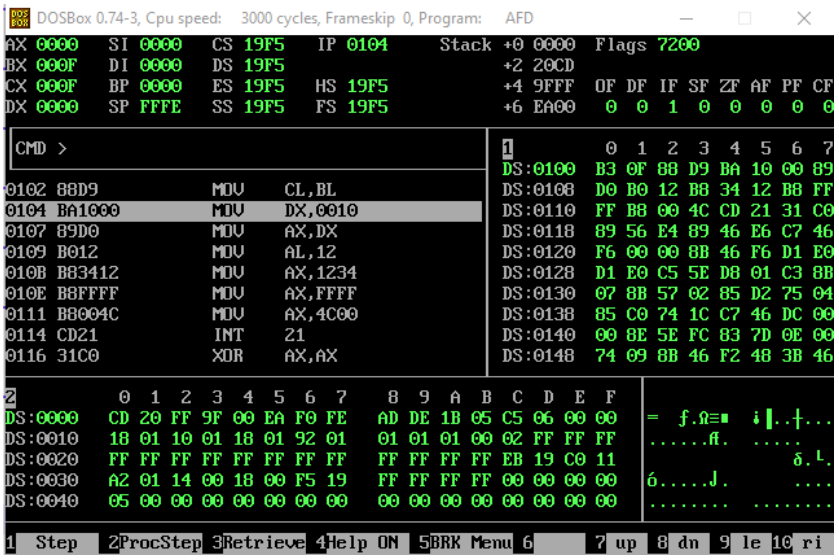


First of all im putting 15 into BL but in the terminal F is shown which is also right as the hexadecimal of 15 is F

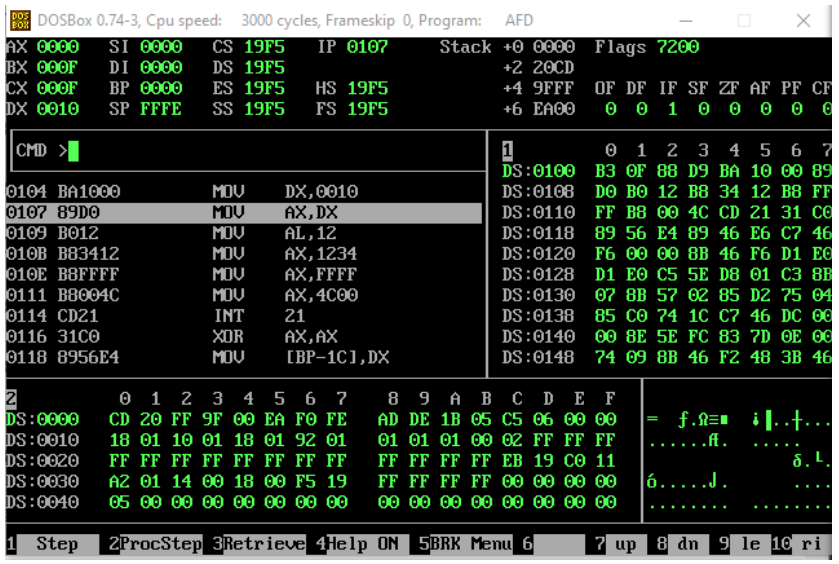
After which im putting the value of BL into CL so now CL will also show us F which is 15.



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

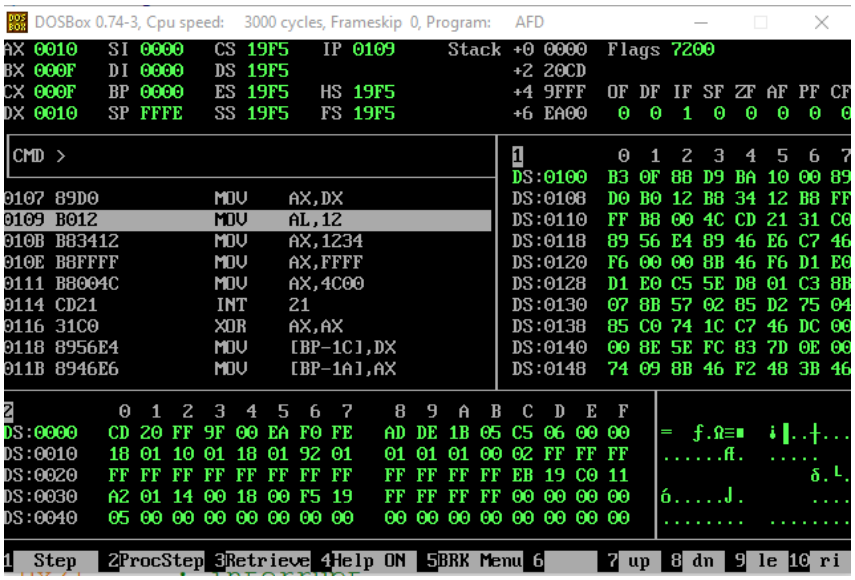


Now as the panel shows im copying the value of DX into AX so now after this code is executed AX will also show us 0010 (which is 16 in decimal)

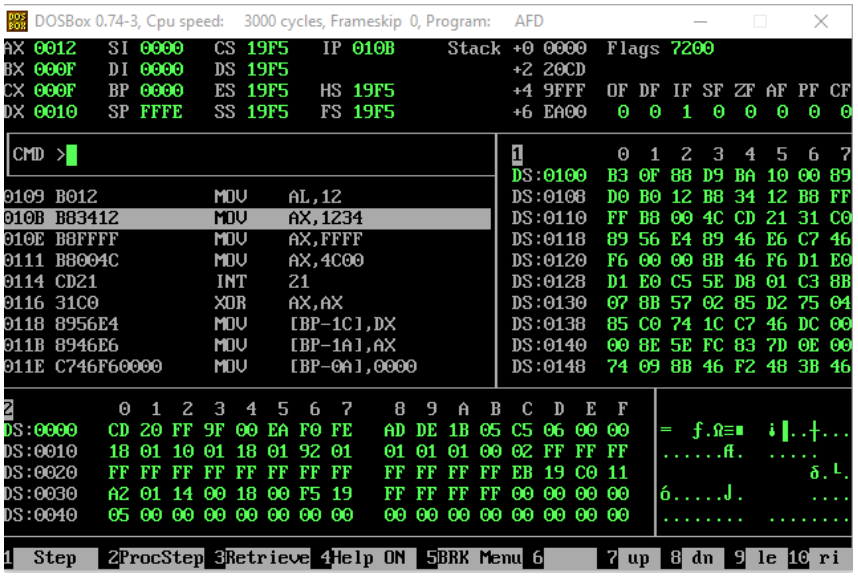


Putting hexadecimal value 12 into Al which is the lower half of the Ax register

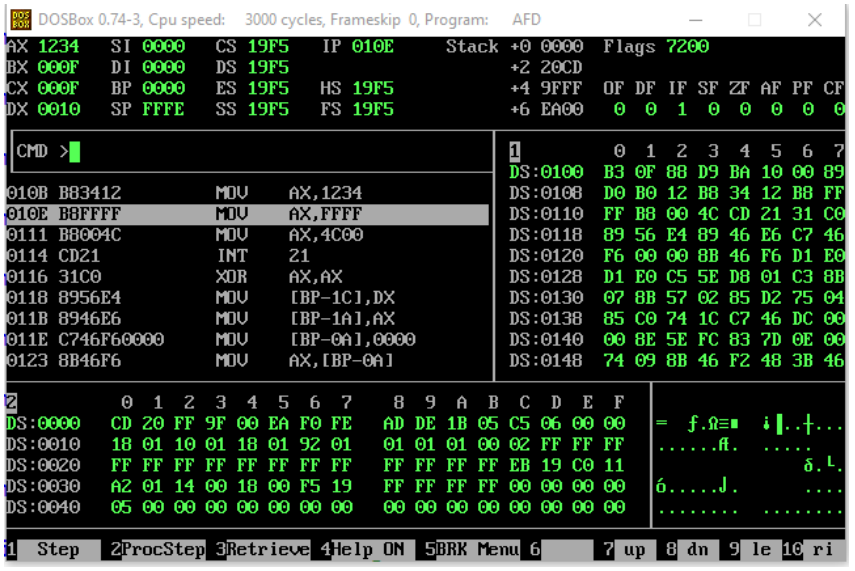
And So the value will be stored in the least significant place.



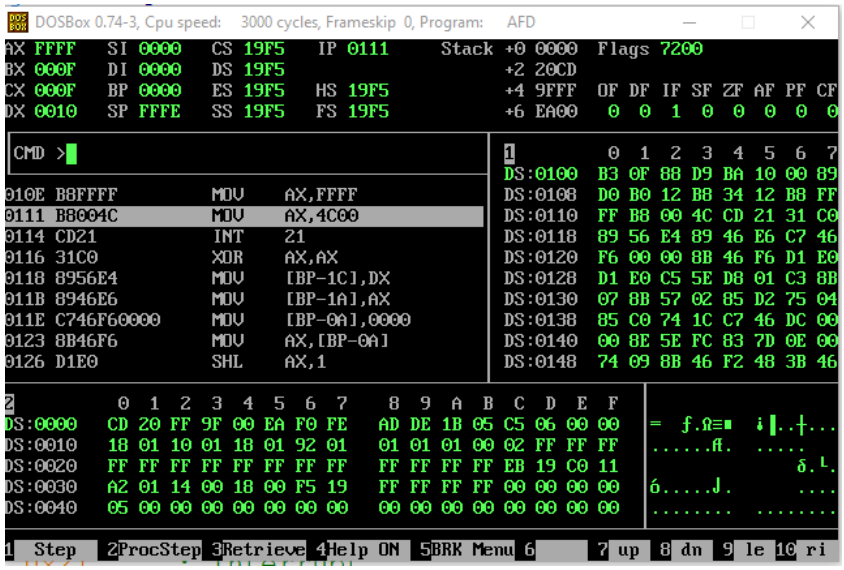
Now im putting the hexadecimal value (1234) inside ax



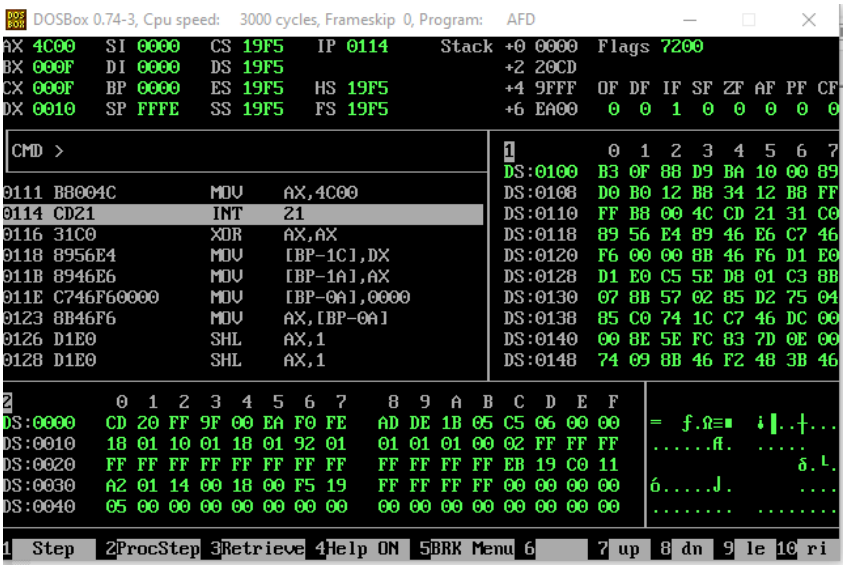
Now im putting hexadecimal value (FFFF) into AX replacing the previous value.



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



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



Task 3:

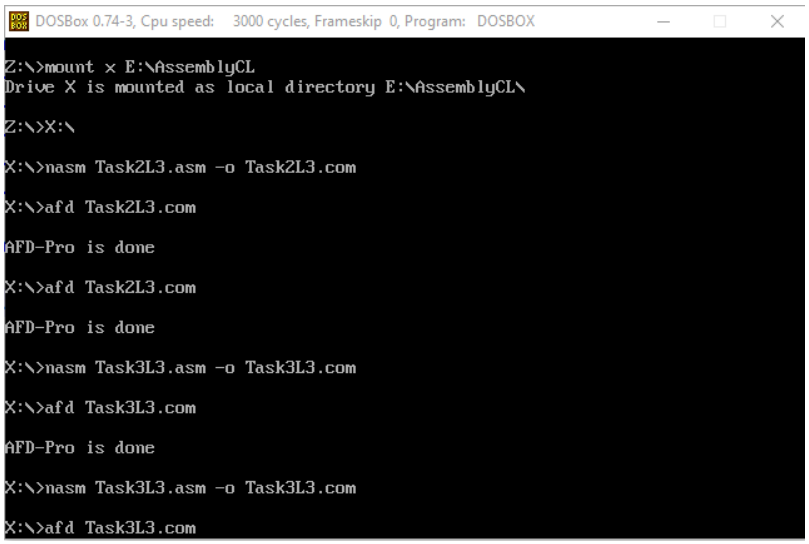
Write a program in assembly language that calculates the square of six by adding six to the accumulator

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

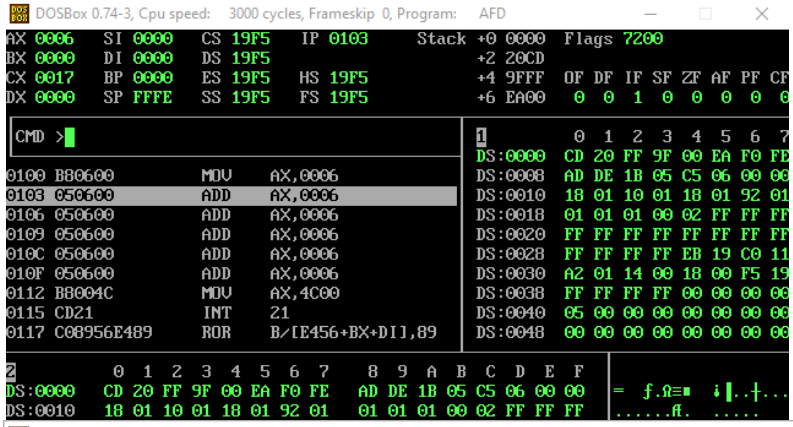
The Code is Self-Explanatory as I've added comments with each line whatever is happening.

```
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
```

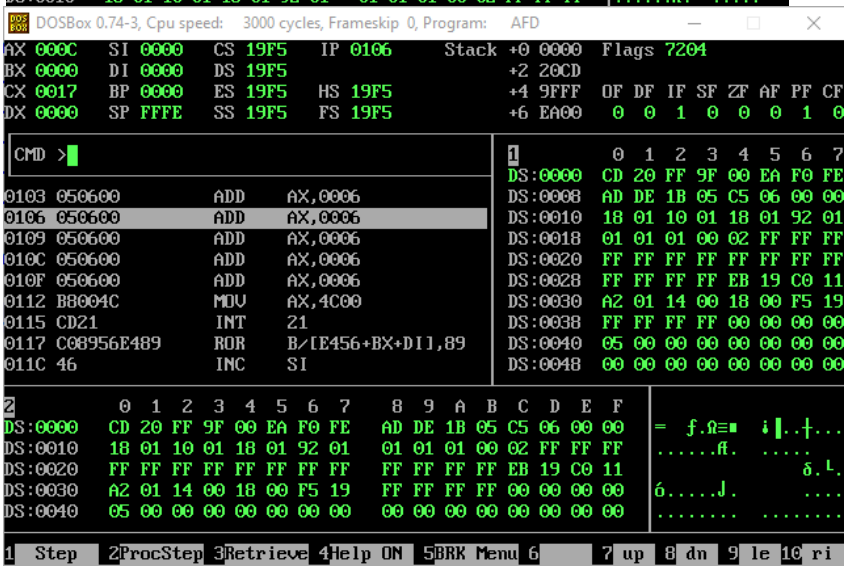
I'm doing the same stuff as explained above compiling and then making the executable file for the afd to run.



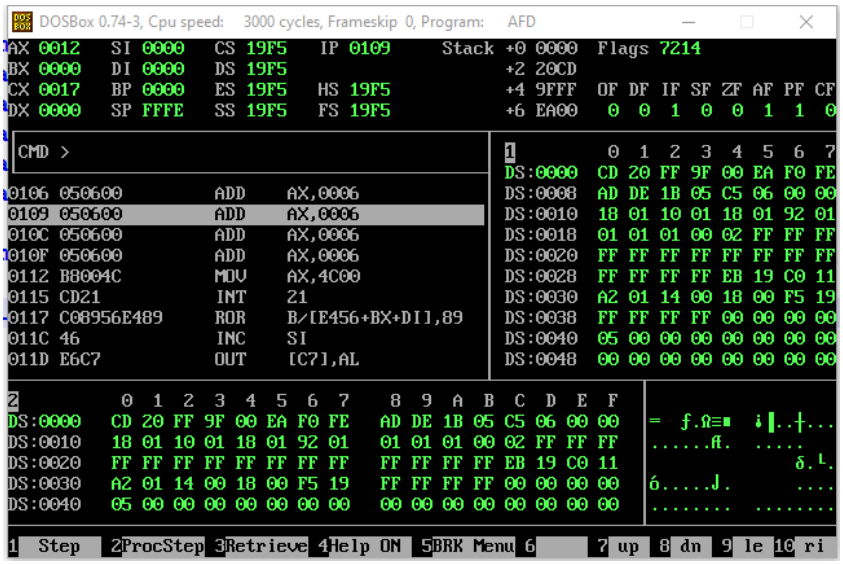
So I placed 6 in ax and then add 6 with ax.



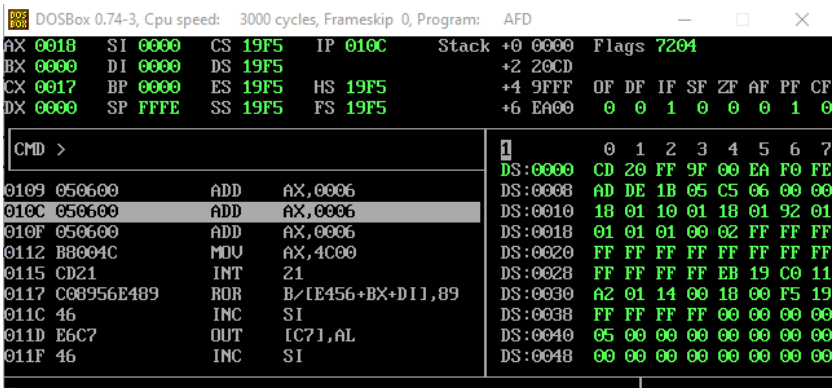
As you can see we previously added 6 with 6 in ax that becomes 12 which can be seen in the ax register it became C which is the hexadecimal of 12



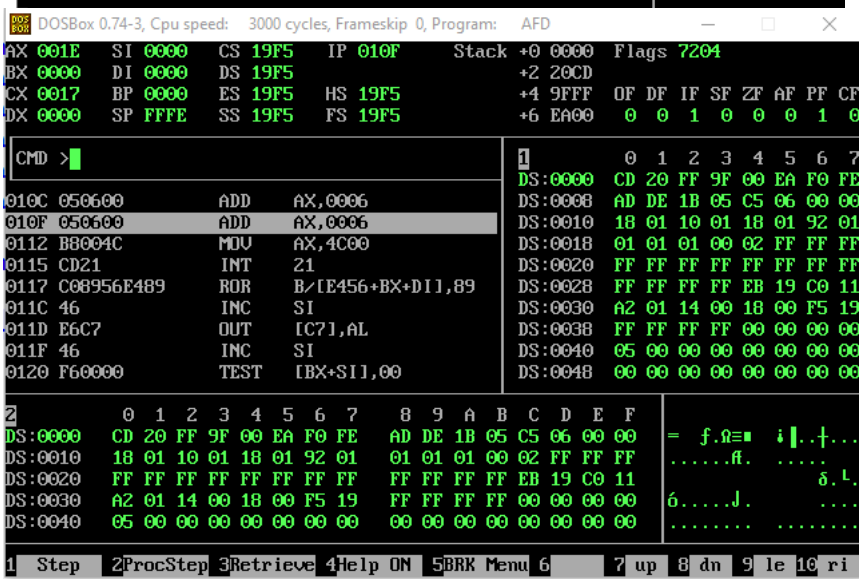
Now we add 6 in the same AX which has 12 in it making the value in the register (12 (Hexadecimal)) which becomes 18.



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.

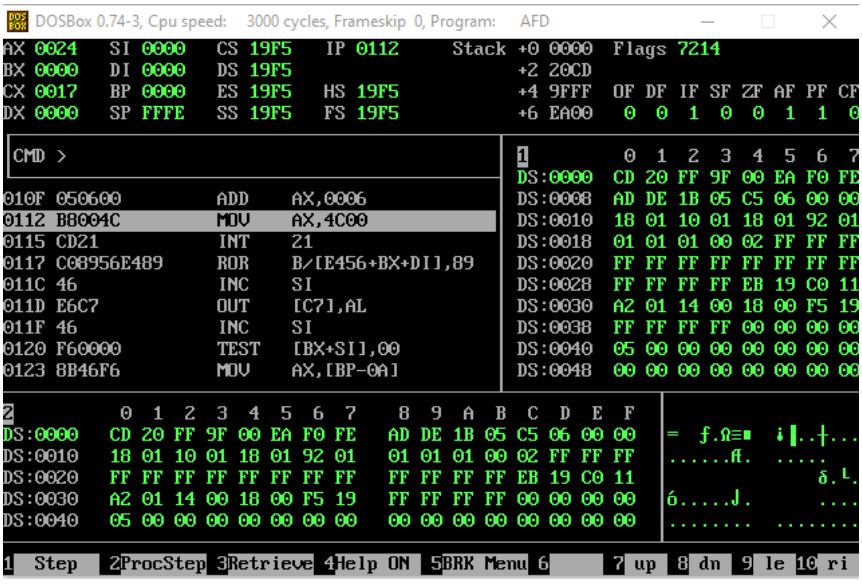


Now we add 6 again with the previously value of 24 in the AX register making it 30 which can be seen in the register having 1E which's decimal value is 30.

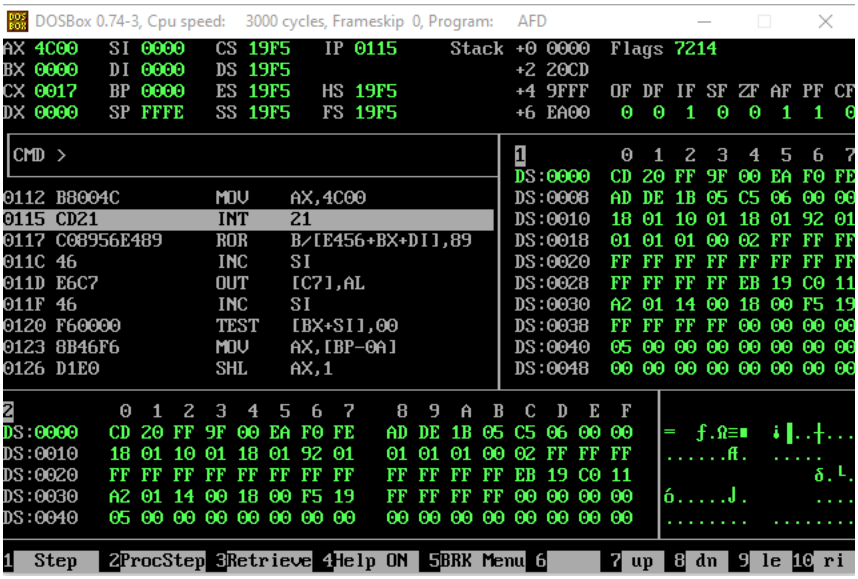


Now you can see the last time we added 6 with the AX register having the value of 30 making it 36 whose hexadecimal value is 24 as being shown in the register.

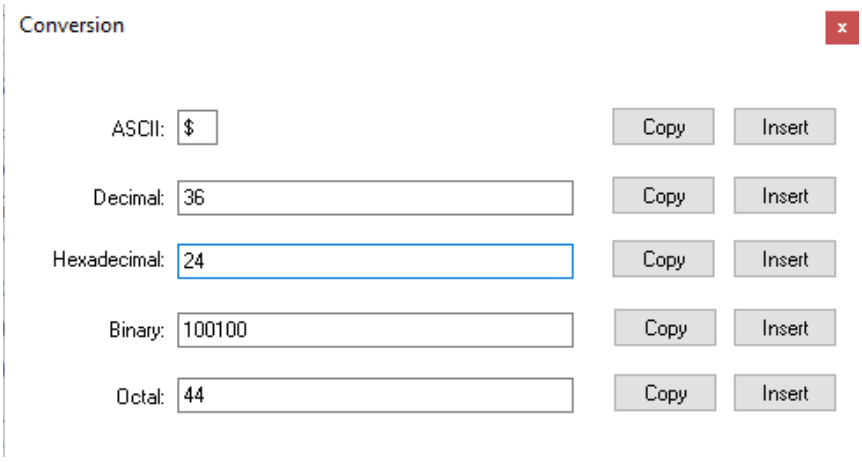
And Like This we achieve the square root of 6 by adding 6 times 6 with 6 which was initially in AX register.



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



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.



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

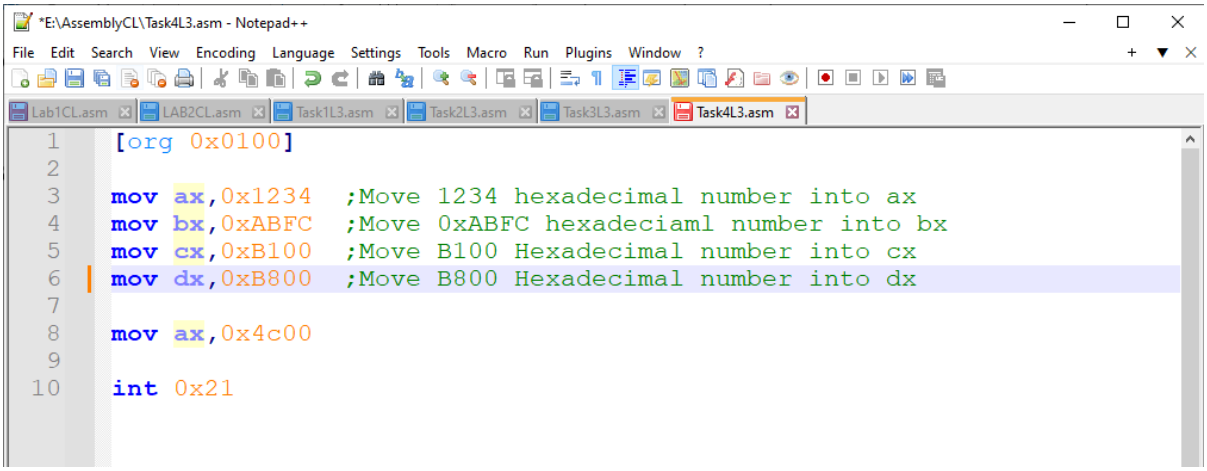
such manner with the help of your results.

a. 1234

b. ABFC

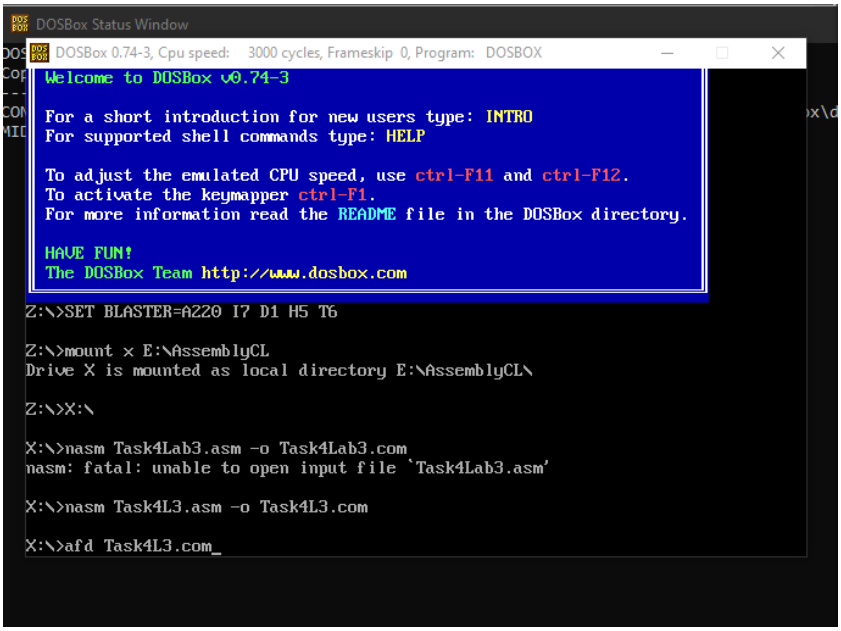
c. B100

d. B800

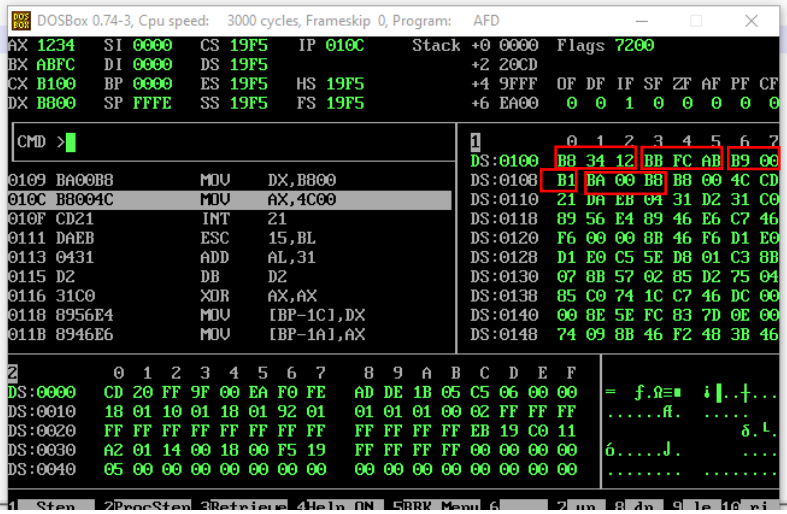


In the Above code I simply stored the hexadecimal values in the registers Also explained well in the comments.

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.



```
1  [org 0x0100]
2
3  mov ax,0x1234   ;Move 1234 hexadecimal number into ax
4  mov bx,0xABFC   ;Move 0xABFC hexadeciaml 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
```



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 4rth 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 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 been seen swapped in a way that 00 can be seen first and B8 can be seen after that.

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