4. The end result turns out to be 9. First eax has 4 stored with ecx as 2 for a counter. It then subtracts 2 from 4 so we're left with 2 in eax and 1 on the outside loop. Then 4 is into ecx to begin the second counter which increments eax by 4 giving us 6. because of the second loop. Now that ecx has one its subtracted from eax which is 6, giving us 5. 4 goes back into the second loop counter and is added to eax giving us 5+4=9. This ends the outer loop and the final result of 9.

$eax = 4 = 2+4 = 6-1 = 5+4 = 9$

2. 8, because DWORD are 4 bytes and there are two of them.

5.

5h	BB
6h	08
7h	a
8h	6
9h	22
Ah	11
Bh	22
Ch	11
Dh	DD
Eh	CC
Fh	00
10h	00
11h	66

needs to fill up us 00 because DWORD is 4 bytes and Var9 only has 2 bytes so the other 2 must fill out with 00.

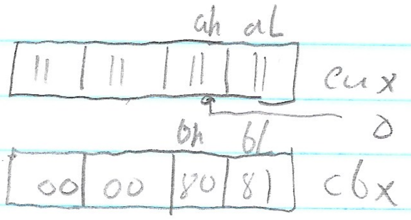
12h	55
13h	44
14h	33
15h	AA
16h	99
17h	88
18h	77

6. a) moving eax offset of var10
moves the memory location of var10
into eax because thats what offset
does, its just a pointer. eax has 11h

b) moving the type of var8 takes
the size of the type which is 2
because BYTE is 1 and DWORD
is 4 and stores the value into
eax.

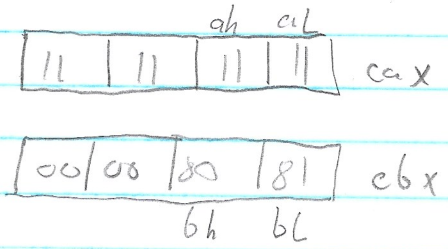
7.
```
 ; my code
 mov ecx, eax   ; moves size of strng into ecx for counting
start:
  Mov ebx, [edx] ; moves var in edx location into ebx
  push ebx       ; pushes stored ebx val into stack
  inc edx        ; increments edx location counter to grab next chr
loop start
mov ecx, eax ; moves stored strng size into ecx counter

Print:
  pop eax       ; pops last value in stack into eax.
  call writeChar; prints char stored in eax to screen
loop Print
```

8. a)

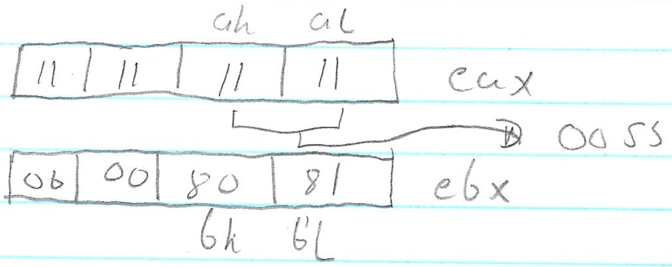


so then final is
11 1100 11

b)

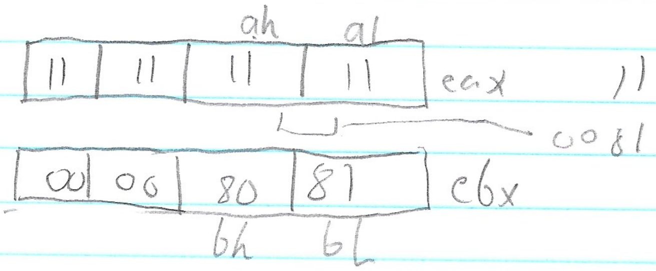


c)



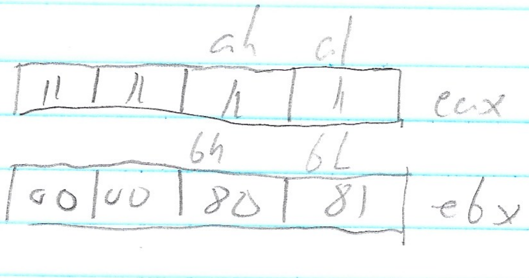
because ax is two bytes so then final in eax is
11 11 00 55

d)



moves what is in BL into ax which is 4 bytes so 0081 is input into ax.

e)



moving bh into eax takes the value in bh, 80, and fills it into eax.

80 being 1 byte and eax being 4 so the rest are filling out with zero because of movzx.

9. a)

CF	OF	SF	ZF
1	0	0	1

eax

ah

88	88	FF	FF

```
  1  1111  1111
+            1
 ------------
   0000  0000
```

b)

CF	OF	SF	ZF
0	1	1	0

ax

eax

00	00	66	77

```
   6    6      7    7          2    0    0    0
  0110 0110   0111 0111       0010 0000 0000 0000
+ 0010 0000   0000 0000
 ----------------------
  1000 0111   0111 0111
```

10. ;mycode   all comments are same excep xor from #7

```
 mov  ecx, eax
 mov  ebp, eax
 Start
   mov  eax, [edx]
   xor eax, 29      ; xors 29 with eax and stores in eax.
   push eax
   inc  edx
loop start
mov ecx, ebp
Print:
   pop eax
   call WriteChar
loop print
```