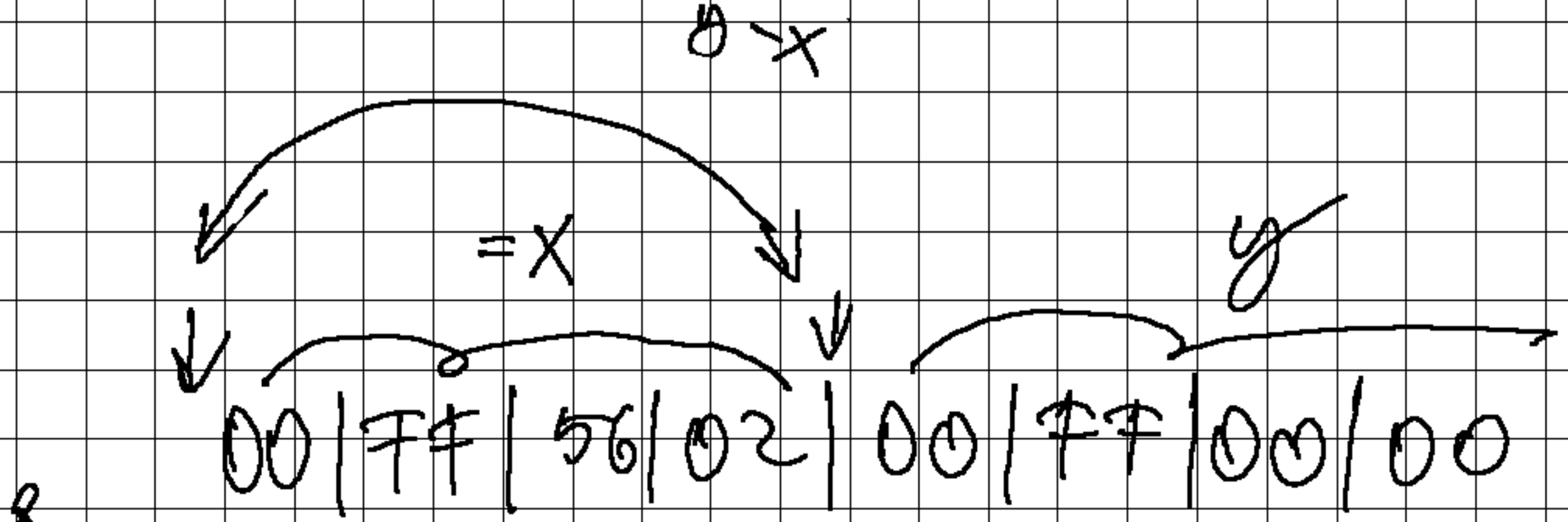
0 x
=x
y
00|FF|56|02|00|FF|00|00
8
=8
14
=6
00|04|01|'y'|'-'|'x'|80|00|
z-a=-6
=6
c
Cdd $-6=4

c) mov ax, ~(16h | 32)

mov bx, 2000h >> 4

imul bh

32	2	r0
16	2	r0
8	2	r0
4	2	r0
2	2	r0
1	2	r1
0		

32 = 0000|0000|0010|0000

16h = 0000|0000|0001|0110

16h | 32 = 0000|0000|0011|0110

~(16h | 32) = 1111|1111|1100|1001

= FFC9

AX = FFC9 = $16^3 \cdot 15 + 16^2 \cdot 15 + 16 \cdot 12 + 16^0 \cdot 9 = 65481$

AH AL

unsigned

$C_2(65481) = 110110 = -54$

2000h = 0010|0000|0000|0000 = 8192

2000h >> 4 = 0000|0010|0000|0000 (signed & unsigned)

CF = 0F = 0

bcs. 110 ∈ [-128, 127]

BX = 0200h

bh bl

imul Bh ⇒ AL · BH with sign = 02 · 99 = 110

signed

C9 = 201 = 11001001 - unsigned = 69426 unsigned

= 00110111 - signed

≈ 55

a1 db '256, -256'

each is considered a character on a byte

in memory: '2' | '5' | '6' | ',' | '-' | '2' | '5' | '6'

a2 dw 256, 256h

256 = 2^8 = 0000 0001 0000 0000 = 0100h

⇒ in memory we will have: 00|01 - little endian

256h ⇒ in memory 56|02

already in hexa

a2: 00|01|56|02|

a3 dw $-a2 = 4

(we had declared 2 words so far, so the distance from the current position in memory to a2 is 4 bytes)

in memory 00|04

an lgn -256/4 (does not occupy any memory) the assembler swaps an with the value at assembling time

a5 db 128 >> 1 , -128 << 1 a5 |80|00|

$128 = 2^7 = 10000000 = 80h$

$128 >> 1 = 1000\ 0000 >> 1 = 0100\ 0000 = 40h$

in memory |40|

$-128 << 1 = -256$ (can't be stored on a byte)

$= -256 + 256 = 0$

in memory we have |00|

a6 dw a2-a5, ~(a2-a5)

$a_2 - a_5 = -6$ (between a_2 and a_5 there are 6 bytes)

$6 = 0000\ 0000\ 0000\ 0110$

$C_2(6) = 1111\ 1111\ 1111\ 1010 = FFFA$

in memory: FA|FF

$\sim(a_2 - a_5) = 0000\ 0000\ 0000\ 0101 = 0005h$

in memory: 05|00

07 dd [a2], !a2

[a2] is not a value determinable at assembly time

!a2 - we can only do bit operations on with scalar values

08 dd 256h ^ 256, 256256h

256h = 0010.0101.0110 =

256 = 2^8 = 1.0000.0000

256h ^ 256 = 0000.0000.0000.0000.0000.0010.0101.0110 ^
0000.0000.0000.0000.0000.0001.0000.0000
0000.0000.0000.0000.0000.0011.0101.0110

= 00000356h

in memory: 56|03|00|00

256256h in memory: 56|62|25|00

08 dd ($-08)+(010-$)

$-08 = 8 (8 bytes from the current position in memory to the position of 08 in memory)

010-$ = 4 (4 bytes between the current pos in memory to the pos. of 010)

($-08)+(010-$) = 8+4 = 12 = 0Ch ⇒ 00|00|00|0C in memory

010 dw -275,256

275 = 1111 1111 $C_2(275)$ = 1111 1111 0000 0001

= FF01

in memory 01|FF

255	2	r1
127	2	r1
63	2	r1
31	2	r1
15	2	r1
7	2	r1
3	2	r1
1	2	r1
0		

$256 = 2^8$ = 1.0000.0000 = 0100h

in memory 00|01

011 resb 6

00|00|00|00|00|00 - reserves 6 empty bytes

012 times 4 dw 256 (places 256 4 times in memory)

$256 = 2^8 = 1.0000.0000 = 0100h$

00|01|00|01|00|01|00|01 - in memory

013 dw times 4 -128

Syntax error it should be

013 times 4 dw -128

times 2 resw 2 - reserves 2 words, empty times 2

00|00|00|00|00|00|00|00 so 4 in total

times 2 dd 12345678h (puts the number in memory times 2

78|56|34|12|78|56|34|12

one dd one dd

x dw -296, 256h

256 = 2^8 = 1.0000.0000 = 0100h

⇒ in memory 00|01

(little endian applies)

-256 = C2(296) = 1111 1111 0000 0000

= FF00

⇒ in memory 00|FF

256h ⇒ in memory: 56|02

X: 100|FF|56|02

y dw 296|-296, 256h & 256

296 = 0000 0001 0000 0000

-256 = 1111 1111 0000 0000

296|-296 = 1111 1111 0000 0000 = FF00

⇒ in memory: 00|FF

256h = 0000 0010 0101 0110

256 = 0000 0001 0000 0000

256h & 256 = 0000.0000.0000.0000

⇒ in memory: 00|00

y: 00|FF|00|00

```
z  db  $-z, y-x
   db 'y'-'x', 'y-x'
```

$-z = 0 (distance between the current pos in memory to the pos. of t in memory)

⇒ in memory |00|

y-x = 4 (distance between the position in memory of y to x, which is 4 bytes)

⇒ in memory |04|

'y'-'x' = 1 (the difference of the ASCi code for the characters y and x which is 1)

in memory |01|

'y-x' (takes each character separately and puts them in memory)

in memory |'y'|'-'|'x'|

|00|04|01|'y'|'-'|'x'|

a db 512 >> 2 , -512 << 2

512 >> 2 = 128 = 2^7 = 1000.0000 = 80h

⇒ in memory |80|

-512 << 2 = -2048

2048 = 2 = 2^{11} = 1000 0000 0000

C(2048) = 1111 1000 0000 0000 on a word

⇒ on a byte we have 1001 in memory

a: |80|00|

b dw z-a, !(z-a)

z-a = -6 (we have 6 bytes between z and a)

6 = 0000 0000 0000 0110

C_2(6) = 1111 1111 1111 1010 = FFFA

⇒ in memory |FA|FF

!(z-a) = !(-6) = 0 ⇒ in memory |00|00|

b: |FA|FF|00|00|

c dd ($-b)+(d-$), $1-2*y+3

$-b = 4 (we have 4 bytes between the current position in memory and b)

d-$ = 4 (the distance between the current pos in memory and d is 4 because c is defined as a old, and only the first value is valid)

$-2*y+3 (syntax error, you can't multiply pointers.

($-b)+(d-$) = 8 ⇒ 08|00|00|00 in memory

we ignore $-2*y+3 because of syntax error.

d db -128, 128^(~128)

128 = 2^7 = 10000000 = 80h

C_2(128) = 10000000 = 80h ⇒ |80| in memory

128^(~128) = 10000000^ = 11111111 = FF in hexa
01111111

d: |80|FF| ⇒ |FF| in memory
|80| in memory

l times 2 resw 6

times 2 dd 1234h, 5678h

times 2 resw 6 (we reserve 6 empty words times 2, so 12 words in total)

In memory:

|00|00|00|00|00|00|00|00|00|00|00|00|
|00|00|00|00|00|00|00|00|00|00|00|00|

times 2 dd 1234h, 5678h

|34|12|00|00|78|56|00|00|34|12|00|00|78|56|00|00|

places the values as double words in memory two times (little endian applies)

II) a) 1. lea eax, [6+esp]

moves in eax the value of esp+6

(lea ignores the '[', ']' and takes the value)

2. mov eax, 6+esp

syntax error

3. movsx ax, [6+esp]

puts in ax, a byte with sign extended from [6+esp]

4. mov ebp, [6+ebp*2]

puts in ebp the value from the address of [6+ebp*2]

5. mov [6+ebp*2], 12

⇒ syntax error, we have specify the size for at least one operand

6. mov ebp, [ebx + esp]

index base ⇒ because esp can't be an index

moves in ebp the value from [ebx+esp]

index base

7. movsx [6+esp], eax

moves in [6+esp] the value from eax

8 mov [6+esp*2], eax

⇒ Syntax error because esp can't be an index

9. mov [6+ebp*2], [6+esp]

⇒ Syntax error, We can't have both explicit operands from memory

10. movzx eax, [6+ebp*2]

⇒ Syntax error, we have to specify the size of the right operand.

---

a1 dd -103, a6

103	2	r1
51	2	r1
25	2	r1
12	2	r0
6	2	r0
3	2	r1
1	2	r1
0		

$103 = 1100111 = 0000.0000.0000.0000.0000.0000.0110\ 0111$

$C_2(103) = 1111.1111.1111.1111.1111.1111.1001.1000$

$= FF|FF|FF|98$

⇒ 98|FF|FF|FF in memory

a2 dw 1&2, 3|4, 5^6, 1|2, 3^4, 5&6

dw 1^2, 3&4, 5|6

1&2 = 0001 & 0010 = 0000 ⇒ in memory |00|00|

3|4 = 0011 | 0100 = 0111 = 0007h on dw

⇒ in memory |07|00|

5^6 = 0101 ^ 0110 = 0011 = 0003h on dw

⇒ in memory |03|00|

1|2 = 0001 | 0010 = 0011 = 0003h on dw

⇒ in memory |03|00|

3^4 = 0011 ^ 0100 = 0111 = 0007h on dw

⇒ in memory |07|00|

5&6 = 0101 & 0110 = 0100 = 0004h on dw

⇒ in memory |04|00|

1 ^ 2 = 0001 ^ = 0011 = 0003h on dw
0010

⇒ in memory
03|00

3 & 4 = 0011 & = 0000 = 0000h on dw
0100

⇒ in memory
00|00

5 | 6 = 0101 | = 0111 = 0007h on dw
0110

⇒ in memory |07|00|

a3 dw $-$$, $$-a3

$-$$ = 26 (how many bytes there are between the current memory position and the start of the data segment

26 = 11010 = 001Ah ⇒ 1A|00 in memory

26	2 * 0
13	2 * 1
6	2 * 0
3	2 * 1
1	2 * 1
0	

$$-03 = -26 (the first position in mem. minus the first pos. in memory for a3)

C2(26) = 1111 1111 1110 0110 = FFE6

⇒ in memory |E6|FF|

on db ~((-1)^ 0bbh), 1^ 0bbh

1 = 0000 0001

-1 = C_2(1) = 1111 1111 = FF in hexa

bb = 1011 1011

(-1)^ 0bbh = 1111 1111^ 0100 0100

1011 1011

~(-1)^ 0bbh) = ~(0100 0100) = 1011 1011 = BB

in hexa

⇒ in memory |BB|

1^ 0bbh = 0000 0001^ 1011 1011 = 1011 1010 = BA

⇒ in memory |BA|

on |BB|BA|

os dd -129 << 17h, B1h >> 0111b

129 = 128+1 = 2^7+1 = 1000 0000 +1 = 1000 0001

C_2(129) = 1111 1111 1111 1111 1111 1111 0111 1111

17 = 0000 0000 0001 1111 = 31 in decimal

-129 << 17h = 1000 0000 0000 0000 0000 0000 0000 0000

= 80 00 00 00 ⇒ in memory

|00|00|00|80|

0B1h >> 0111b shifting to the right, div by 2

0B1h = 10110001

0111b = 7

0B1h >> 0111 = 177>>7 = 0000 0000 0000 0000

0000 0000 0000 0001

= 0000 0001 h

⇒ in memory |01|00|00|00|

for B1h >> 0111b we get a Syntax error because the symbol is not defined for 'B1h' it should be '0B1h'

a6 dd 'a1a2a3a4a5', (a6-a5) << (a5-a4)

'a1a2a3a4a5' ⇒ takes each character and places the ascii code in memory

in memory |'a'|'1'|'a'|'2'|'a'|'3'|'a'|'4'|'a'|'5'|

a6-a5 = 4 (the distance between a6 and a5 is 4 bytes)

a5-a4 = 2 (the distance between a5 and a4 is 2 bytes

(a6-a5) << (a5-a4) = 4 << 2 = 16 = 2^4 = 10000

in memory |01|00|00|00|

a_1 times h dw a_2, a_2+1

$a_2 = 8$ (we have declared 2 dd before a_2
so the offset of a_2 is 8 bytes)

$a_2+1 = 9$ (same here but we add 1

in memory 08|00|09|00

We put it 4 times so:

08|00|09|00|08|00|09|00|08|00|09|00|08|00|09|00

a8 dw !(a_2-a_1), !(a_2-1)

$a_2-a_1 = 8$ (we have 8 bytes between a_2 and a_1)

$a_2-1 = 7$ (the offset of a2 is 8, we sub 1 and
is 7)

!(a_2-a_1) = !(8) = 0 = 0000h

!(a_2-1) = !(7) = 0 = 0000h

⇒ in memory 00|00|00|00|

movsx ebx, [ebx+ebp]

moves in ebx a word from the adress of [ebx+ebp], with sign extended

xchg ebx, [ebx+esp] because esp can't be an index

index base

puts in ebx the value from [ebx+esp] if it's valid and moves the contents of ebx at the address of [ebx+esp]

lea eax, [ebx+esp]

puts in eax the value of ebx+esp

lea eax, [ebx+ebp]

puts in eax the value of ebx+ebp

mov ax, a+b

moves in ax the sum of the offsets of a plus b, if the offset of a is 2 and the offset of b is 3, it moves 2+3 in ax, so AX will be 5.

push eax, similar with mov [esp+4], eax

5) a)
```
mov ax, 0100h
mov bx, 1000+10b
idiv bl
```

0100h = 0000 0001 0000 0000b

= 2^8 = 256 (signed and unsigned)

mov ax, 0100h - moves in ax, the value 0100h = 256

mov bx, 1000 + 10b

1000	2
500	2
250	2
125	2
62	2
31	2
15	2
7	2
3	2
1	2
0	

1000 = 1111 1010 0000 = 03E8 h

1010b = 2 d = 2 h

1000+2 = 1002 = 1111 1010 1010 b (signed and unsigned)

= 03EA h

moves in bx the value 1002

idiv bl : AX : BL = AL (with sign)

256 : 234 = 1 r 22

AL AH

incapt in AL ⇒ OF = 0

b) mov ah, 0cdh
mov al, 0ebh
add ah, al

0cdh = 1100 1101 = 205 unsigned
= 205-256 = -51 signed

mov ah, 0cdh => moves in ah, 0cdh as represented above

mov al, 0ebh => moves in al the value 0ebh = 1110 1011 =

$2^7+2^6+2^5+2^3+2^1+2^0$ = 128+64+32+8+2+1
= 235 unsigned
= 235-256 = -21 signed

add ah, al
add ah+al = 205+235 = 440 > 255
=> CF = 1 (cant put it in ah)
=> in ah we have 440-256 = 184 unsigned
= 184-256 = -72 signed
=> -72 ∈ [-128,127] => OF = 0

c) mov ax, 1010h
mov bx, 1111b
mul bl

mov ax, 1010h ⇒ moves in ax the value
1010h = 0001 0000 0001 0000 b
= $2^{12} + 2^{4}$ = 4112 (signed and unsigned)

mov bx, 1111b ⇒ moves in bx the value
1111b = 000Fh = $2^0+2^1+2^2+2^3 = 1+2+4+8 = 15$
(signed and unsigned)

mul bl ⇒ AX = AL * BL = 16 · 15 = 240
$\in [0,256]$
AL = 00010000 = 2^4 = 16
⇒ CF = OF = 0
⇐
there is no carry digit

d) mov dh, 200
mov ch, 62h
sub dh, ch

mov dh, 200 ⇒ moves in dh the value
200 = 128+64+8 = $2^7+2^6+2^3$ = 1100.1000
= 200 unsigned
signed: 200-256 = -56 (signed)

mov ch, 62h ⇒ moves in ch the value
62h = 0110 0010 = $2^1+2^5+2^6$ = 2+32+64 = 98
(signed and unsigned)

sub dh, ch ⇒ substracts ch from dh
dh - ch = 200 - 98 = 102 = 66h
(signed and unsigned)
⇒ CF = 0 = 0
there is no carry digit.

102 ∈ [-128, 127]