Subj 1:

2. X dw -256, 256h

$256 = 2^8 = 1\,0000\,0000$

$C_2(256) = 1111\ 1111\ 0000\ 0000 = FF00$

$-256 =$

$\Rightarrow$ 00|FF in memory

256h $\Rightarrow$ 96|02 in memory

y dw 256|-256, 256h & 256

$256 = 2^8 = 0000\ 0001\ 0000\ 0000$

$-256 = C_2(256) = 1111\ 1111\ 0000\ 0000$

$256\ |\ -256 = 0000\ 0001\ 0000\ 0000\ |\ =$

$1111\ 1111\ 0000\ 0000$

$= 1111\ 1111\ 1\,0000\ 0000 = FF00h \Rightarrow$ 00|FF in memory

256h = 0000 0010 0101 0110 &

256 = 0000 0001 0000 0000 =

$=$ 0000 0000 0000 0000 = 0000h $\Rightarrow$ 00|00 in memory

z db $-z, y-x

db 'y'-'x', 'y-x'

$-z = 0 (the distance between the current position in memory and z in bytes

= 0000 0000 = 00

⇒ 00 in memory

y-x = 4 = 0000 0100 = 04h ⇒ 04 in memory

(the distance in bytes between y and x in memory)

'y'-'x' = 1 = 0000 0001 = 01h ⇒ 01 in memory

(the diff of ascii code)

'y-x' ⇒ places each character in memory

⇒ 'y' | '-' | 'x' | in memory

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

512>>2 = 128 = 2^7 = 10000000 = 80h = 80 in memory

$512 = 2^9$ = 10 0000 0000

-512 = C2(512) = 1111 1110 0000 0000 << 2 =

= 1111 1000 0000 0000 = F800 ⇒

⇒ we take a byte ⇒ 00 in memory

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

z-a = -6 = $C_2(6)$ = 1111 1111 1111 1010 = FFFA

6 = 0000 0000 0000 0110 ⇒ FA|FF in memory

!(z-a) = !(-6) = 0 ⇒ 00|00 in memory

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

$-b = 4 (the distance between the current pos in memory and the pos. of b in memory)

a-$ = 4 (the distance between the pos. of a in memory and the beginning of c)

($-b)+(a-$) = 4+4-8 = 0008h ⇒ 08|00|00|00 in memory

$-2*y+3 ⇒ multiplication of pointers is not valid ⇒ syntax error.

d db -128, 128 ^ (~128)

128 = 1000 0000

$C_2(128)$ = 1000 0000 = 0000 0080h on dd ⇒

⇒ 80|00|00|00 in memory

128 = 1000 0000 ^

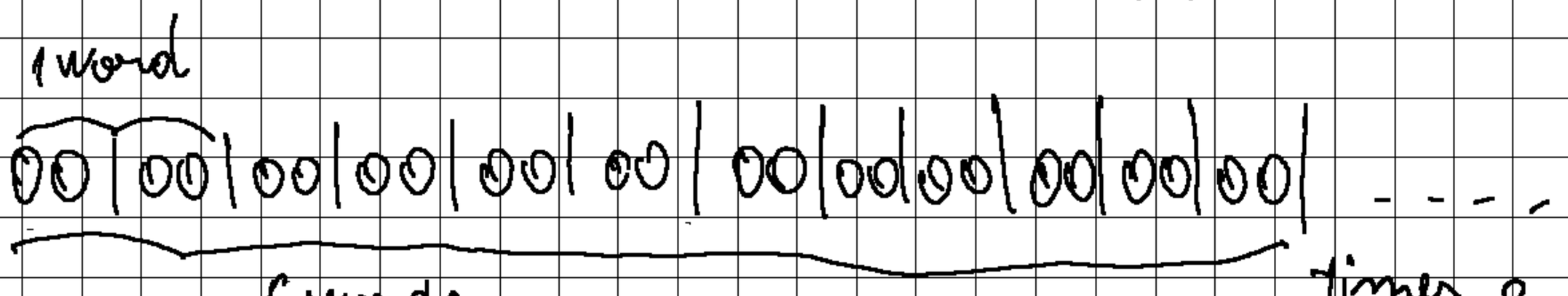
~128 = 0111 1111 = 1111 1111 = 0000 00FF

⇒ FF|00|00|00 in memory

2 times 2 resw 6

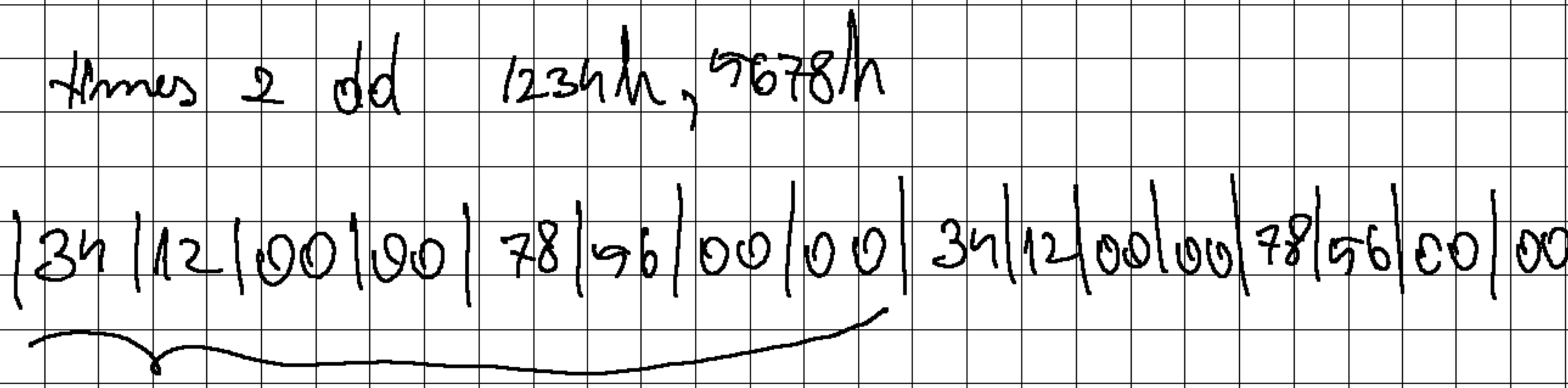
times 2 dd 1234h, 5678h

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



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

times 2 dd 1234h, 5678h



Subject 2:

2. a1 db '256'

⇒ puts each character in memory ⇒ '2'|'5'|'6'| in memory

a2 dw 256, 256h

256 = 2^8 = 0000 0001 0000 0000 = 0100h ⇒ 00|01 in memory

256h ⇒ 56|02| in memory

a3 dw $5+a2

the sum of pointers is invalid

an lgn -256/4

it is not stored in memory

a5 db 256>>1, 256<<1

256>>1 = 128 = 2^7 = 1000 0000 = 80h ⇒ 80 in memory

256<<1 = 512 = 2^9 = 0000 0010 0000 0000

We only take a byte ⇒ 00h ⇒ |00| in memory

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

a5-a2 = 4 = 0004h => 04|00 in memory

!(a5-a2) = !(4) = 0 = 0000h => 00|00 in memory

a7 dw [a2], ~a2

the value of a2 can't be determined at assembly time

We can only do bit operations on scalar values

08 dd 256h ^ 256, 256256h

256h = 0000 0010 0101 0110 ^

256 = 2^8 = 0000 0001 0000 0000 =

= 0000 0011 0101 0110 = 0356 => 56|03|00|00 in memory

256256h => 56|62|25|00 in memory

a9 db $-a9

$-a9 = 0 => 00|00|00|00 in memory

a_{10} db 256, -255

$256 = 2^8 = 1\,0000\,0000 \Rightarrow$ 00h on byte

$\Rightarrow$ 100 in memory

$-255 = C_2(255) = 0000\,0001 = 01h \Rightarrow$ 101 in memory

$255 = 2^8 - 1 = 1111\,1111$

a_{11} dw 256h-256

$256h = 0010\,0101\,0110 = 2^1+2^2+2^4+2^6+2^9 =$

$= 2+4+16+64+512 = 598$

$256h - 256 = 798 - 256 = 342 = 0156h \Rightarrow 56|01$

342 | 16 | 16 | 16

32 | 21 | 1 | 0

22 | 16

16 | 9

6

a_{12} dw 256-256h

$256 - 256h = 256 - 598 = -342$

$-342 = C_2(342) = C_2(0000\,0001\,0101\,0110) =$

$= 1111\,1110\,1010\,1010 = FEAA \Rightarrow AA|FE$

in memory

a_{13} dw -256

$256 = 2^8 = 0000.0001.0000.0000$

$-256 = C_2(256) = 1111\ 1111\ 0000\ 0000 = FF00h$

$\Rightarrow$ 00|FF in memory

a_{14} dw -256h

$256h = 0000\ 0010\ 0101\ 0110$

$-256h = C_2(256h) = 1111\ 1101\ 1010\ 1010$

$= FDAA \Rightarrow$ AA|FD in memory

a_{15} db 25,256,2,76

25 | 16 | 16
16 | 1 | 0
9

$\Rightarrow 25d = 19h$

76 | 16 | 16
48 | 3 | 0
8

$\Rightarrow 76d = 38h$

$\Rightarrow$ 02|07|06|19|06|02|38| in memory

Subject 4:

2. a_1 db '256, -256'

takes each character and places it in memory

=> |'2'|'5'|'6'|','|'-'|'2'|'5'|'6'|

a_2 dw 256, 256h

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

=> 00|01|

in memory

256h => 76|02| in memory

a_2 dw $-$ a_2

$-$ a_2 = 4 (the distance between the position of a_2 and a_3 in bytes)

=> 04|00 in memory

an eqv -256/4

! it is not stored in memory

a_5 db 128>>1 , -128<<1

$128 = 2^7 = 1000\ 0000 >> 1 = 0100\ 0000 = 40h$

$\Rightarrow$ |40| in memory

$-128 = C_2(128) = 1000\ 0000 << 1 = 0000\ 0000$

$\Rightarrow$ |00| in memory

a_6 dw $a_2 - a_5$, ~(a_2-a_5)

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

~(a_2-a_5) we can't do bit operation only on scalar values.

a_7 old [a_2], !a_2

[a_2] the value of a_2 is unknown at assembly time

!$a_2 = 0 \Rightarrow$ 00|00|00|00 in memory

a_8 dd 256h^256 , 256256h

256h = 0000 0010 0101 0110 1

$256 = 2^8 =$ 0001 0000 0000 =

= 0000 0011 0101 0110 = 0000 0356h ⇒

⇒ 56|03|00|00 in memory

256256h ⇒ 56|62|25|00 in memory

a_9 dd ($-$a8) + ($a10 - $)

$ - $a8 = 8 ($ - $a8) + ($a10 - $) = 8 + 4 = 12 = C

$a10 - $ = 4 ⇒ 0C|00|00|00| in memory

a_{10} dw -255, 256

$255 = 2^8 - 1 =$ 1111 1111

$-255 = C_2(255) =$ 1111 1111 0000 0001 = FF01

⇒ 01|FF in memory

$256 = 2^8 =$ 1 0000 0000 = 0100h ⇒ 0001 in memory

a_{11} rusb 6 $\Rightarrow$ |00|00|00|00|00|00| in memory
6 bytes

a_{12} times 4 dw 256

$256 = 2^8 = 10000\,0000 = 0100h$

$\Rightarrow$ 00|01|00|01|00|01|00|01 in memory

a_{13} dw times 4 -128 $\Rightarrow$ syntax error, it should be a_{13} times 4 dw -128

times 2 rusw 2 $\Rightarrow$ 00|00|00|00|00|00|00|00
word
2 words
2 words

times 2 dd 12345678h

$\Rightarrow$ 78|56|34|12|78|56|34|12

Subject 6:

2. X dw -256, 256h

256 = 2^8 = 0000 0001 0000 0000

-256 = $C_2(256)$ = 1111 1111 00000000 = FF00h

=> 00|FF in memory

256h => 56|02 in memory

Y dw 256 |-256, 296h & 256

256 = 2^8 = 0000 0001 0000 0000 | =

-296 = $C_2(296)$ = 1111 1111 0000 0000

= 0000 0001 0000 0000 = 0100h => 00|01
in memory

256h = 0000 0010 0101 0110 &

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

=> 00|00 in memory

z db $-z, y-x

db 'y'-'x', 'y-x'

$-z = 0 ⇒ |00| in memory

y-x = 4 ⇒ |04| in memory

'y'-'x' = 1 ⇒ |01| in memory

'y-x' ⇒ 'y'|'-'|'x'| in memory

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

512>>2 = 128 = 2^7 = 10000000 = 80h ⇒ |80| in memory

-512 = C_2(512) = 1111 1110 0000 0000 << 2 =

512 = 2^9 = 0010 0000 0000

= 1111 1000 0000 0000 = F800h

⇒ on byte in memory |00|

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

z-a = -6 = C_2(6) = 1111 1111 1111 1010 = FFFA

6 = 0000 0000 0000 0110

⇒ FA|FF in memory

!(z-a) = !(-6) = 0 ⇒ 00|00 in memory

Subject 2022

2. a1 dd -103, a6

103	2 rm
51	2 rm
25	2 rm
12	2 ro
6	2 ro
3	2 rm
1	2 rm
0	

103 = 0110 0111

-103 = C2(103) = 1001 1001 = 99h

⇒ in memory on dd

⇒ 99 | ff | ff | ff

a6 = 10 (the offset of a6)

= 10|000 = 28h ⇒ 28|00|00|00 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 ⇒ 00|00 in memory on dw

= 0000h

3|4 = 0011 | 0100 = 0111 = 0007h ⇒ 07|00 in memory

5^6 = 0101 ^ 0110 = 0011 = 0003h ⇒ 03|00 in memory

1|2 = 0001 | 0010 = 0011 = 0003h ⇒ 03|00 in memory

3^4 = 0011 ^ 0100 = 0111 = 0007h ⇒ 07|00 in memory

5&6 = 0101 & 0110 = 0100 = 0004h ⇒ 04|00 in memory

1^2 = 0001 ^ 0010 = 0011 = 0003h ⇒ 03|00 in memory

3 & 4 = 0011 & 0100 = 0000 = 0000h => 00|00 in memory

5 | 6 = 0101 | 0110 = 0111 = 0007h => 07|00 in memory

a3 dw $-$$, $$-a3

$-$$ = 26 (the number of bytes from the current offset in memory and the start)

26	16	16
16	1	0
10		
=		
A		

26 = 1Ah => 1A|00 in memory

$$-a3 = -26 (the diff. between the start of memory and the position of a3 in memory)

-26 = C2(26) = C2(0000 0000 0001 1010) = FFE6h

=> E6|FF in memory

a4 db ~((-1)^0bbh), 1^0bbh

(-1)^0bbh = 1111 1111 ^ 1011 1011 = 0100 0100 =

-1 = 1111 1111 ~ = 1011 1011 = BB => |BB| in memory

0bbh = 1011 1011

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

=> |BA| in memory

a_5 dd -129 << 17h

129 = 128 + 1 = 1000 0001

$C_2(129)$ = 0111 1111 << 31 = 1000 0000 0000 0000 0000 0000 0000 0000

17 = 31

=> 80 00 00 00 h => 00 | 00 | 00 | 80 in memory

a_6 dd 'a1a2a3a4a5', (a_6 - a_5) << (a_5 - a_4)

'a1a2a3a4a5' => places each character in memory

=> | 'a' | '1' | 'a' | '2' | 'a' | '3' | 'a' | 'h' | 'a' | '5' | 0 | 0 |

1 byte

$a_6 - a_5 = 8$ 8 << 2 = 0000 1000 << 2

$a_5 - a_4 = 2$ = 0010 0000 = 20h

=> 20 | 00 | 00 | 00 in memory

a_7 times h dw a_2, a_2+1

a_2 = 8 (the offset of a_2)

=> 08 | 00 | 08 | 00 | 08 | 00 | 08 | 00

a_2

8 times h

a_2+1 => syntax error, can't add to pointers.

a8 dw !(a2-a1), !(a2-1)

a2-a1 = 8 a2-1 = 7

!(a2-a1) = !8 = 0 => 0000h => 0000 in memory

!(a2-1) = !7 = 0 => 0000h => 0000 in memory

movsx ebx, [ebx+ebp]

moves in ebx with sign extended a byte from the address [ebx+ebp]

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

index base

moves in ebx the value from [ebx+esp] if it's valid

moves at [ebx+esp] the value from ebx

lea eax, [ebx+esp]

index base

moves in eax the value of ebx+esp, after the sum

lea eax, [ebx+ebp]

moves in eax the value of ebx+ebp, after the sum

mov ax, a+b moves in ax, the value of the sum of the offsets a+b

a+b = 1

push eax <=> mov [ebp+4], eax

Subject 2024

2. X dw -129, 10+100h+1000b

129 = 128+1 = 1000 0001

-129 = C2(129) = 1111 1111 0111 1111 = FF7F => 7F|FF in memory

10+100h+1000b = 10+256+8 = 274

274 = 1000100102 = 112h => 12|01 in memory

y dw 1001h >> 1001b, 128h & 128

1001h = 0001 0000 0000 0001 >> 9 =

1001b = 9

= 0000 0000 0000 1000 = 0008h => 08|00

128h = 0000 0001 0010 1000 &

128 = 0000 0000 1000 0000

= 0000h => 00|00 in memory

= 0000 0000 000 0000 =

z dw z, $-z

z = 8 (the offset of z in memory)

= 1000 = 0008h => 08|00 in memory

$-z = -8 (the diff between the offset of the start of the data segment and the offset of z)

-8 = $_2(8) = 1111 1111 1111 1000 = FFF8 =>

=> F8|FF in memory

W dol X+y-z, W-y+x

the addition of pointers is not valid

h dw 101b, 101-h, 11h-11b, h-11

101b = 0005h => 05|00 in memory

101-h = 101-12 = 89 = 59h => 59|00 in memory

offset of h

11h-11b = 17-3 = 14 = E => 0E|00 in memory on word

h-11 = 12-11 = 1 = 0001h => 01|00 in memory

a db $$-$, h-11b

$$-$ = -20 = EC ⇒ |EC| in memory

20 = 14h

h-11b = 12-3 = 9 ⇒ 09h ⇒ |09| in memory

b dd a+b-0bh+2, h-b+0ah-0bh

! the addition of pointers is not valid

c db 3-b, z-w

3-b = 3-22 = -19

19 = 0001 0011

-19 = C2(19) = 1110 1101 = ED ⇒ |ED| in memory

z-w ⇒ syntax error because w is invalid

d dw -513, 128^(~128)

-513 = C2(-513) = 1111 1101 1111 1111 = FDFF ⇒ FF|FD in memory

513 = 0000 0010 0000 0001

128 = 2^7 = 0000 0000 1000 0000

~128 = 1111 1111 0111 1111

128^(~128) = FF7F ⇒ 7F|FF in memory

l dd abcdefgh, 'abcdefgh'
abcdefgh ⇒ syntax error, the symbols are not defined
'abcdefgh' ⇒ places each character in memory on 1 byte:
⇒ |'a'|'b'|'c'|'d'|'e'|'f'|'g'|'h'|0|

l dw w-1, [w-1]
w-1 ⇒ syntax error because w is not valid
[w-1] ⇒ syntax error because the value of w at assembly time is not valid

4 times 3 dw 'db'
⇒ in memory |'d'|'b'|'d'|'b'|'d'|'b'|

k dw 1+2b+3h+a, c+ach
! addition of pointers is not valid for both values, we have '+a' and 'c+' also 2b is not defined

m db a+aah, a+ah ⇒ should be aah
! invalid; addition of pointers is not valid
a+aah = 20+a = 30 = 1E ⇒ 1E|00|00|00 in memory

Sdd a-start, start-start1

this is wrong, syntax error because we already have named the entry point label as 'start'
also the expresions is not simple or relocatable
⇒ we get an error because we defined them in the code segment and some variables in the operation are from the data segment.

Subj. 1.

3. a) mov lax, 200 ⇒ moves in lax the value 200

$$\begin{array}{r|l|l} 200 & 16 & 16 \\ \hline 16 & 12 & 0 \\ \hline 40 & & \\ 32 & & \\ \hline 8 & & \end{array}$$

200 = C8 = 1100 1000

(unsigned and signed
because 200 can be stored in dword)

mov ebx, 254h ⇒ moves in ebx the value 596

$254h = 2 \cdot 16^2 + 5 \cdot 16^1 + 4 \cdot 16^0 = 596 = 0010\ 0101\ 0100$

(signed and unsigned because 596 can be stored in dword)

idiv bl

BX = 0000 0010 0101 0100
BH BL

⇒ AL = AX / BL

⇒ AL = 200 / 84

⇒ AL = 2 = 2h

⇒ AH = 32 = 20h

AX = 0000 0000 1100 1000 = 200

BL = 0101 0100 = 54h = 84

At division CF, OF are not set

b) mov ax, 256h ⇒ moves the value 598 in ax

256h = 0000 0010 0101 0110 = 598 (signed & unsigned)

mov dx, -1 ⇒ moves in dx the value -1

-1 = FFFF = 1111 1111 1111 1111 = 65535 unsigned
≠ signed

add al dh, dh ⇒ adds to ah, the value of dh

AX = 0000 0010 0101 0110

AH = 2 AL

DX = 1111 1111 1111 1111

DH = FF DL

= -1

AH + DH = 2 + (-1) ⇒ 0000 0010 +
1111 1111

= 1 = 1 0000 0001

⇒ CF = 1 (we have a transport digit)

OF = 0 (2 + (-1) = 1 ∈ [-128, 127]

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

16h = 0001 0110 16h | 32 = 0001 0110 | 0001 0000

32 = 2^5 = 0010 0000 = 0011 0110

~(16h | 32) = ~(0011 0110) = 1100 1001 = C9

= FFC9 on AX

= 65481 unsigned

= C_2(FFC9) = <u>0000 0000</u> <u>0011 0111</u> = 0037h = 55
AH AL

signed: -55

mov bx, 2000h >>4 => moves in BX, 512

2000h = 0010 0000 0000 0000 >>4 =

= <u>0000 0010</u> <u>0000 0000</u> = 2^9 = 512 (signed and unsigned)
BH BL

imul BH => AX = AL * BH = -55 * 2 = -110
On signed

On unsigned: 65536 - 110 = 65426

CF = OF = 0 because b * b = b

d) mov ax, 21 << 7 => mov ax, 2688

21 << 7 = 10101 << 7 = 1010.1000.0000 = 2688

$21 = 2^4 + 2^2 + 2^0 =$ 10101 signed/unsigned

mov bh, 10h ^ 3

10h = 0001 0000 ^ = 0001 0011 = 13h = 19 signed / unsigned

3 = 0000 0011

sub bh, al

BH - AL = 19 - 128 = -109 signed

unsigned: -109 + 256 = 147

CF = OF = 1

l) mov eax, ebx <=> lea eax, [ebx]

Subject 2:

3. a1) mov ah, 129 ⇒ moves in ah, 129 without A sign

mov bh, 97h ⇒ moves in ah, 159 without sign

97 = 1001 1111 = 159

add ah, bh

ah+bh = 129+159 = 288 ∉[0,255] ⇒ CF=1

zF=0, result is not zero

SF=0, the sign of ah

signed: 129 = 129-256 = -127

159 = 159-256 = -97

ah+bh = -127+(-97) < -128 ⇒ OF=1

a2) mov ax, 128 ⇒ moves in ax 128

AH=0, AL=128 = 1000.0000

sar al, 7 ⇒ AL = 1111 1111 = FF = 255 unsigned
= -1 signed

imul ah ⇒ AX = AL*AH = -1*0 = 0

⇒ CF=0, OF=0

zF is not set at division or multiplication

SF=0 (the result is positive)

a3) mov ax, 256 ⇒ moves in ax, 256

mov bx, -1 ⇒ moves in Bx, -1

add ah, bh

AH AL

256 = 2^8 = 0000 0001 0000 0000

-1 = 1111 1111 1111 1111

AH+BL = 0000 0001 + ⇒ CF=1 (we have a carry digit)
1111 1111
1 0000 0000

+ + (-) = + ⇒ OF=0

SF=0 (0 is positive)

ZF=1 (the result is 0)

a4) mov ah, 128|2 ⇒ moves in ah, 130

128 = 1000 0000 | = 10000010 = 130
0000 0010

mov bh, 90h >>3 ⇒ moves in bh, 18

90 = 1001 0000 >>3 = 0001 0010 = 2+2 = 18

Sub ah, bh

ah - bh = 130 - 18 = 112 signed and unsigned

1000 0010 -
0001 0010
0111 0000 ⇒ CF=0 (no carry digit)

- - (+) = + ⇒ OF=1 , SF=0 (positive number), ZF=0

Subject 3:

3. a) mov ax, 1000h ⇒ moves in ax, 4096

1000h = 0001 0000 0000 0000 = 2^{12} = 4096 (signed & unsigned)

mov bl, 1000b + 10b ⇒ moves in bl, 10

1000b + 10b = 1010b = 0Ah = 10 signed & unsigned

div bl ⇒ AL = AX / BL = 4096 / 10 = 409 ∉ [0, 255]

⇒ division overflow ⇒ runtime error

b) mov ah, 0bch ⇒ moves in ah, 0bch

bc = 1011 1100 = 188 unsigned

= 188 - 256 = -68 signed

mov al, 0deh ⇒ moves in al, 0deh

de = 1101 1110 = 222 unsigned

= 222 - 256 = -34 signed

add ah, al ⇒ adds al to ah

ah + al = 188 + 222 = 410 ∉ [0, 255] ⇒ CF = 1

= 410 - 256 = 154 unsigned

= 154 - 256 = -102 signed

∈ [-128, 127]

⇒ OF = 0

c) mov ax, 1001h ⇒ moves in ax, 1001h

1001h = 0001 0000 0000 0001 = 4097 Signed & unsigned

mov bx, 1111b ⇒ moves in bx, 15

1111b = F = 15 signed & unsigned

imul bl AL = 0000 0001 = 1

⇒ AX = AL * BL = 1 * 15 = 15 signed & unsigned

⇒ OF = CF = 0

byte * byte = byte

d) mov dh, 62h ⇒ moves in dh, 98

62h = 0110 0010 = 98 signed & unsigned

mov ch, 200

200 ∈ [0, 255] unsigned

200 = 200 - 256 = -56 Signed } = C8h

sub dh, ch

dh - ch = 98 - 200 = -102 ∉ [0, 255] ⇒ CF = 1

= 98 - (-56) = 154 ∉ [-128, 127] ⇒ OF = 1

Subject 6:

3. 1) lea eax, [6+ESP]

⇒ moves in eax the value of 6+ESP, lea ignores the square brackets

2) mov eax, 6+ESP

⇒ syntax error, we need to do the addition before the move because addition like that is only valid for offsets.

3) movsx ax, [6+ESP]

⇒ moves with sign extended in ax, a byte from the offset [6+ESP] in memory

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

moves in ebp a dword from the address [6+ebp*2]

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

⇒ Syntax error, we need to specify the size for at least one operand

6) mov ebp, [ebx+esp]

esp can't be an index

moves in ebp, a dword from [ebx+esp] (index: ebx, base: esp)

7) movsx [6+esp], eax

moves at the addres [6+Esp] the value from eax

8) mov [6+Esp*2], eax

⇒ Syntax error, esp can't be an index ⇒ ⇒ Invalid effective address.

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

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

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

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

Subject 1:

3. 1. mov ax, 0100h ⇒ moves in ax, 256

0100h = 0000 0001 0000 0000 = 2^8 = 256 (signed & unsigned)

mov bx, 1000+10b ⇒ moves in bx, 1002

1000+10b = 1002 = 0000 0011 1110 1010 = 03EAh (signed and unsigned)

1002	2 no
501	2 nu
250	2 no
125	2 nu
62	2 no
31	2 nu
15	2 nu
7	2 nu
3	2 nu
1	2 nu
0	

0000 0001 (AH) 0000 0000 (AL)

0000 0011 (BH) 1110 1010 (BL) = EA = 234 (unsigned)

= 234 - 256 = -22 signed

idiv bl ⇒ AL = AX / BL

⇒ AL = 256 / (-22) = -11, AH = -1h

CF and OF are not set at division

b) mov ah, 0cdh

0cdh = 1100 1101 = 205 unsigned

= 205 - 256 = -51 signed

mov al, 0ebh

0ebh = 1110 1011 = 235 unsigned

= 235 - 256 = -21 signed

add ah, al

ah+al = 205 + 235 = 440 $\notin [0, 255] \Rightarrow$ CF = 1

na have a carry digit

1100	1101 +
1110	1011
1 1011	1000

$\Rightarrow$ - + (-) = (-) $\Rightarrow$ OF = 0

= 440 - 256 = 184 unsigned

= 184 - 256 = -72 signed

c) mov ax, 1010h

1010h = 0001 0000 0001 0000 = 4112 (signed & unsigned)

AH = 16, AL = 16

mov bx, 1111b

1111b = f = 15 (signed & unsigned)

mul BL $\Rightarrow$ AX = AL * BL = 16 * 15 = 240

(signed & unsigned)

CF = OF = 0 because byte * byte can be stored on byte

d) mov dh, 200

200 = C8 = 1100 1000 unsigned

200 = 200-256 = -56 signed

mov ch, 62h

62h = 0110 0010 = 98 (Signed & unsigned)

Sub dh, ch

dh - ch = 200 - 98 = 102

1100 1000 - ⇒ CF = 0 (no carry digit)

0110 0010

0110 0110 ⇒ OF = 1 (-) - (+) = +

Subject 2:

3. mov eax, -2 ~~-2+256=254 - unsigned~~

2 = 0010 = 2h

-2 = $C_2(2)$ = 1111 1111 1111 1111 1111 1111 1111 1110 = FFFFFFFE

AX AX

signed

mov ebx, -1

-1 + 256 = 255 ⇒ unsigned

1 = 0001 = 1h

-1 = $C_2(1)$ = 1111 1111 1111 1111 1111 1111 1111 1111 = FFFFFFFF

BH BL signed

div BL ⇒ AL = AX / BL = 65534 / 255

= 256 ∉ [0, 255]

⇒ integer overflow

CF, OF are not set at division

ii) mov lax, 65409 ⇒ moves in lax the value 65409

65409 = FF81 (signed & unsigned for lax

FF81 = 1111 1111 1000 0001 for AX: unsigned: 65409

AH AL signed 65409 - 65536

"-1 on signed =-127

idiv ah

⇒ AL = AX / AH = -127 / -1 = 127 signed & unsigned

AH = 0

add al, al

AL + AL = 127+127 = 254 ∈[0,297] ⇒ CF = 0

no carry digit

254-256 = -2 signed

⇒ OF = 1

iii) mov lax, 255h & 255

255h = 0000 0010 0101 0101 &

255 = 0000 0000 1111 1111

AH AL

= 00000000 01010101

= 0055h = 85

signed & unsigned

mov bx, 256^256h

256 = 0000 0001 0000 0000 ^

256h = 0000 0010 0101 0110

BH BL

= 0000 0011 0101 0110

= 0356h = 854 d

in signed & unsigned

mul BH => AX = BH * AL = 3 * 85 = 255 $\in [0, 255]$

=> CF = OF = 0 because byte * byte can be stored on a byte

255 on unsigned & -1 on signed

iv) mov ax, 1282

128 = 2 = 1000 0000 + 1000 0010 = 130 = 82h

2 = 0000 0010 signed & unsigned

mov bh, 4ah >> 2

4ah = 0100 1010 >> 2 = 0001 0010 = 18 = 12h

signed & unsigned

sub ah, bh

ah - bh = 0 - 18 = -18 on signed, unsigned; -18 + 256 = 238

0000 0000 -
0001 0010
1110 1110

=> CF = 1 because we have a borrow from outside

(+) - (+) = (-) => OF = 0