MANUELA PETRESCU

Manuela.petrescu@ubbcluj.ro

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

ASC - problems and exercises

Fac. Mathematics and Computer Science

# Conversions

1. Convert to hexadecimal: 10100011b, 00000011, 17o
2. Add the following numbers:
   1. 11001111 b + 10110111b
   2. 0AFh + 15h
3. Convert

-808(d) 🡪 (b) on 16bits

## Answers:

1. The first number is in binary, so we can apply grouping by four: 1010\_0011b=>A3h

The second umber is in decimal, so, 11 in base ten is B in hexa: 00B

The third number is in octal base (that’s ‘o’ not 0); so it will be 0Fh

1. Results:
   1. 1 1000 0110
   2. C4
2. Result: -808(d) is -328(h). 328 in binary is 0011\_0010\_1000, after applying complement to 2,

the result will be: 1111\_1100\_1101\_1000b

Complement to 2 conversions:

0000\_ 0011\_0010\_1000 (inverting bits)=> 1111\_1100\_1101\_0111(b) then adding 1 => 1111\_1100\_1101\_1000

## Exercises:

1. Compute:

28h + 0F6h

1A3h - 368h

0C33h - 0AF5h

144 + 1DFh

15

-176(d) = \_\_\_\_\_\_\_\_\_\_\_(h) = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(b) on 16 bits

-80 (d) = \_\_\_\_\_\_\_\_\_\_\_(h) = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(b) on 32 bits

1. The value of 00100111b is:
   1. 27h
   2. 63h
   3. 26h
   4. 17h
2. The value of 0110\_0011b is:
   1. 0D4
   2. D4h
   3. 104d
   4. 063h
3. The value of 1110\_0011b is: ?

Note: depends on interpretation: signed or unsigned

# Variables and Memory layout

1. What is the memory representation for?

a db 11\_22h

b dw 1234h

c dd 11\_22h

d dd 11\_22\_33\_44\_55h

e dw 11\_22\_34\_45\_56h

1. How will be the data represented in memory?

a dd 125;

b db 2

c db 1,2,3,4

d db '1234';

e db '1,2,3,4'

f db '1',',','2'

g dw 'a'

1. What will be stored in registries after running each instruction considering the following variable definition?

a db 11\_22h

b dw 1234h

c db 0Ah

d dd 11\_22h

mov al, [a];

mov al, [b];

mov ax, [a];

mov eax, [b];

mov ax, [a+1];

mov bx, [c-2];

1. Are the following definitions equivalent?

a) b0 db 256;

b1 dd 100h;

b) b2 db 1203h;

b3 db 3;

1. Do the instructions generate the same result?

v db 4;

x db 5;

mov ax, 4;

mov ax, v;

1. Do the instructions generate the same result

v db 4;

x db 5;

y db 66h;

z dw 11\_22h;

mov ax, 4;

mov eax, dword [v];

1. What is the result generated by the instructions?

Mov al, Aah

Mov ax, Aah;

1. How many bytes will be reserved in the memory?

R resb 4;

a resw 2;

c resd 3;

d resq 4

e rest 1

1. What values will be in BX register and in variable c after running the following code?

a db 22h

b db 22h;

c dw 2h;

d dw 10h;

mov ax, [a];

mov bx, 4;

add bx, ax;

mov ax, [b+1]

add ax, [c+1]

mov [c], ax

1. What value will be in AX register after running the following code, if variable a is a word, b and c are bytes and a=5, b=9,c=4?

mov ax, [a]

add ax, 20

mov bx, [b]

mov bh, 0

mov [b],bx

add bx, [c]

add ax,bx

1. Which of the following variable definitions are correct? (H or X for hexadecimal, D or T for decimal, Q or O for base 8 and B or Y for binary)

a0 db 10b;

a1 dw 0h12;

a2 db 123456h;

a3 db 1111\_0011\_1111\_0000b;

\_a4 dw 124h;

a5# db 33h;

?a6 dw 233h;

a7~ db 1233h;

1. What is the result of:
   1. Mov al, 11223344h
   2. Mov ax, 11223344h
   3. Mov ax, 11223344

## Answers:

**Base notations**: H or X for hexadecimal, D or T for decimal, Q or O for base 8 and B or Y for binary; If no base is specified, base 10 (decimal) is the default one.

1. What is the memory representation for?

a db 11\_22h ; because a is defined as a byte, only a byte will be allocated in the memory. The number in little endian representation is 22\_11h. Because only a byte can be stored, the result will be: 22h

b dw 1234h; 34\_12h (because of little endian)

c dd 11\_22h; 22\_11\_00\_00 because c is defined as a double word

d dd 11\_22\_33\_44\_55hh;

; 55\_44\_33\_22; d is a double word, only that space is allocated in the memory, so 11 is lost.

e dw 11\_22\_34\_45\_ 56h; 56\_45 – see previous comment

a dd 125; 7D 00 00 00 – 125 was in **decimal**, if no base is specified, the **default bas**e is base 10

b db 2; 02

c db 1,2,3,4; 01 02 03 04 – it’s considered an array of individual values

d db '1234'; 31 32 33 34 - it’s considered a string, multiple values,

e db '1,2,3,4'; 31 ‘,’ 32 ‘,’ 33 ‘,’ 34; where ‘,’ is the ASCII code for “,”

f db '1','2'; 31 32

g db 'a' ; 61

1. The memory will look like: 22 34 12 0A 22 11 00 00

mov al, [a]; AL = 22h

mov al, [b]; AL = 34h; b is a word, but AL only a byte, so it will read a byte from the memory address where b is saved. Due to little Indian representation, is 34, the less significant byte

mov ax, [a]; AX = 34 22h because a is a byte, ax a word, so it will read a word from the address of a

mov eax, [b]; EAX = 22 0A 12 34h because a double word from b in memory is: 34 12 0A 22, use little endian to get the value

mov ax, [a+1]; AX = 12 34; variable a starts in memory at 22, a+1 starts at 34, so it will be read 34 12, because of little Indian it will be 12 34

mov bx, [c-2]; BX = 0A 12; c starts at 22 11 00 00, c- 2 starts at 12 0A, bx is a word, so the result will be 0A 12

1. Are the following definitions equivalent?

a)

b0 db 256; 256d = 100h

b1 dd 100h;

Even if they have the same value, the type is different, so they are not equivalent.

b)

b2 db 1203h;

b3 db 3;

Same value is stored in the memory, the type is the same, so the definitions are equivalent.

1. Do the instructions generate the same result?

v dw 4;

mov ax, 4;

mov ax, v;

Without [], the value used is the memory address and not the value defined in variable v. They don’t generate the same result.

1. Do the instructions generate the same result?

v db 4;

x db 5;

y db 66h;

z dw 11\_22h; Memory layout: 04 05 66 22 11

mov ax, 4; AX = 00 04

mov eax, dword [v]; EAX = 22 66 05 04; Nope, ax has different values

1. Do the instructions generate the same result?

Mov al, Aah

Mov ax, Aah;

Syntax Error – the value Aah is not recognized as a number. To be recognized as a number, add “0” in front of it: mov al, 0Aah

1. How many bytes will be reserved in the memory?

R resb 4; 4 \* 1 (1 byte)

a resw 7; 7\* 2 (1 word = 2 bytes)

c resd 3; 3\*4 (double word = 4 bytes)

d resq 5 ; 5 \* 8 (quadword = 8 bytes)

e rest 1; reserves 10 bytes

1. The memory will look like this taking into account that for a byte we have 2 digits in hexa (a touple) and for a word we have 4 digits, 2 groups of 2. Also, because of little endian representation, the number are represented in reverse order (by groups of 2) from less significant to the most significant: 22 22 02 00 10 00

mov ax, [a]; ax = 22 22h because it goes in the memory to the address of a and reads the size of a register. AX is a word, so it reads a word

mov bx, 4; bx= 4

add bx, ax; **bx = 22 26;**

mov ax, [b+1]; ax = 00 02; it goes in the memory to the address of b, adds 1 and reads a word (because AX has the size of a word)

add ax, [c+1]; ax = 00 02h + 10 00h ; the value from the address c+1 in memory is 00 10, due to little endian-> 10 00h

mov [c], ax; **c = 10 02h**

1. The same concepts (from exercise 1 apply, so😊):

The memory will look like: 05 00 09 04

mov ax, [a]; ax = 00 05

add ax, 20h; ax = 00 25

mov bx, [b]; bx = 04 09

mov bh, 0; bx = 00 09

mov [b],bx; b = 09 00 !! c was overwritten now!!, c = 0

add bx, [c]; bx = 00 09

add ax,bx; **ax = 00 25 + 00 09 = 00 2E**

1. The characters from which a label can be constituted are the following:

* Letters, both A-Z and a-z;
* Numbers from 0 to 9;
* The characters \_, $, #, @, ~, . and ?

Only letters, \_ and ? are allowed as the first character of a tag. These rules apply to all case identifiers (symbolic names, such as variable names, labels, macros, etc.).

1. What is the result of:
   1. Mov al, 11223344h = > even if the number is large, al is a byte, so the size occupied will be 1 byte. Due to little endian, the number in the memory is 44 33 22 11; so the AL register will have the value of the last byte: 11
   2. Mov ax, 11223344h; same explanation as before, the number in little endian is 44 33 22 11, the last word is 22 11, so the value in AX register will be 22 11
   3. Mov ax, 11223344; h is missing from the number definition, by default is in base 10, si it will be transformed in base 16 and then the previous logic applies.

## Exercises:

1. Which is the effect of the execution of instruction "mov ax, [b-1]" and „mov ax, [b+1]” if we have defined:

a dw 1A2Bh

b db 3Bh

c db 9h;

Memory: 2B 1A 3B 09

[b+1] reads starting with the next touple in the memory. So, if the memory looks like:

2B 1A 3B, the [b+1] adress will be the adress where variable b starts +1, so in this case, it will be the adress of variable c!!

Similar, the adress of [b-1] will be the adress of the last byte from variable a. Taking into account little endian, it will point to 1A

1. Which is the effect of the execution of instruction "mov eax, [b-1]" and „mov eax, [b+1]” if we have defined:

a dw 1A2Bh

b db 3Bh, 4Bh

c dd 1A2B3C4Dh;

[b+1] reads starting with the next touple in the memory. So, if the memory looks like:

2B 1A 3B 4B 4D 3C... the [b+1] adress will be the adress where variable b starts +1, so in this case, it will be the adress of 4B.

1. a dw 1234h, 0aaaah

b db 2, 3, 0, 1

c dw 7, 8; 00 07 00 08

mov ax, [a+5]

mov bx, [a+3]

mov cx, [a+7]

mov dx, [b-2]

# Multiplications and divisions

1. What will be the result of running the following code where:
   1. a,b,c are variables of type word, where a =8, b=6 and c = 3

mov ax,[a]

add ax, [b]

div [c]

* 1. a,b are variables of type word where a =8, b=6

mov ax,[a]

add ax, [b]

div 2

* 1. a is variable of type byte where a =8

mov ah,[0]

mov al, [a]

mul -3

* 1. What will be the result of the following code if a is variable of type word, a = 513 and b=2

mov ax,[a]

mov bl, [b]

div bl

* 1. What will be the result of the following code if a and b are variables of type byte:

a db 6, b db 2

mov ax,[a]

mov bl, [b]

div bl

* 1. What will be the result of the following code if a and b are variables of type byte:

a db 6, b db -2

mov ax, 0

mov al, [a]

mov bl, [b]

div bl

* 1. What will be the result of the following code if a and b are variables of type byte:

a db 6, b db -2

mov ax, [0]

mov al, [a]

mov bl, [b]

idiv bl

1. What will be the result of running the following code where:

a,b,c are variables of type word

mov ax,[a]

mov dx, 0

add ax, [b]

div [c]

Which is the result if the entry values are: a=5, b=-9, c=2? Explain.

1. What will be the result of:

Mov ah,0

Mov ax, 256;

Mov bl,1;

Div bl;

1. What will be the result of:

Mov ah,0

Mov ax, 256;

Div byte 1

1. What will be the result of:

Mov ah,0; ah=0

Mov ax, 22h;

Mov dx,0;

Mov cx, 3 ;

Div cx

Mov cx, 2;

Div cx ;

1. What will be the result of the following instructions:

mov al, 128 ;

cbw

mov bl,2;

div bl;

1. al = 40
2. al = 40d
3. al = 40,5
4. - execution error - T (overflow)
5. sintax error, cbw should not be used there
6. al = 40h
7. al = 0100\_0000

## Answers:

* 1. A division to word implies DX:AX registers, DX register value was not set, so the result of the division cannot be specified.
  2. There will be an assembly error as division and multiplications cannot be performed with numbers
  3. Check 1.b.
  4. There are some exceptional cases when the result of the division is too large for the register to store it. In this case: the results should be stored in AL register. AL register can store numbers between 0 and 255. The result of 513/ 2 = 256 > 255, so the result will not fit AL and it will get an overflow error.
  5. In AX will have a word starting from position of variable a in the memory, so we will have: AX = 02 06; divided by 2 -> overflow error
  6. Variable b has a negative value, so in BL = FE; if I divide 6 to FE, the result will be 0 (stored in AL) and the remainer will be 6 (stored in AH)
  7. In the line “mov ax, [0]” we will get an access violation (at execution time) as [0] should not have parenthesis. If you remove the parenthesis in will be AL = -3 = FD and AH = 0;

1. The instructions used are for unsigned numbers, but we got signed numbers (5+ -9 = -4), so the result will be: -4 : 2 = -2 (on registers will have AX = 1111\_1111\_1111\_1110)
2. What will be the result of:

Mov ah,0

Mov ax, 256; when the base is not specified, is decimal, 256d = 100h; so ; ;in ax will be: 0100

Mov bl,1; bl =1

Div bl; ax : bl = ah-(the rest of divison); al= quote

0100: 1= 100h -> so in AL there should be 100h – it does not fit the size of AL (which is a byte, in fits only 2 hexa numbers),=> Overflow error

1. What will be the result of:

Mov ah,0

Mov ax, 256;

Div byte 1 -> syntax error, you can not have multiplications or divisions with a constant!!

1. What will be the result of:

Mov ah,0; ah=0

Mov ax, 22h; ax = 00 22;

Mov dx,0; dx = 00 00;

Mov cx, 3 ; cx= 00 03

Div cx ; (dx:ax) 22 / 3 => ax: 7 DX=1 !!! DX register is modified, storing the remainder

Mov cx, 2; cx =00 02

Div cx ; DX:ax = 00 01 00 07: 2 => 10007:2, so when dividing, the remainder become the largest part of the number which was divided!!

1. What will be the result of the following instructions:

mov al, 128 ; 128d = 80 = 1000\_0000

cbw => ax: 1111\_1111\_1000\_0000 = FF 80: 88 40

mov bl,2; bl =2

div bl;

The result is a large value, so it will generate an overflow error.

## Exercises

1. What value will generate the following code:

mov ax,61440

mov bl,5

div bl

How about:

mov ax,61440

mov bl,5

idiv bl

Hint: negative numbers, divisions. How is represented 61440 in memory?

1. What value will generate the following code:

mov ax,0F000

mov bl,5

div bl

How about:

mov ax,0F000

mov bl,5

idiv bl

1. What values will be in BX register and in variable c?

a db 22h

b db 22h;

c dw 2h ;

d dw 10h ;

mov ax, [a];

mov bx, 4;

add bx, ax;

mov ax, [b+1]

add ax, [c+1]

mov [c], ax

1. Mov ah,0

Mov al, 256; 256d=100h 2 al=00h !! so in AX there will be 00 00

Mov bl,1;

Div bl

1. What values will be in AX register?

a)

mov al,3

mov bl, -3

mul al

b)

mov al,3

mov bl,-3

mul bl

# Conversions- signed, unsigned

1. What will be the result of the following instruction:

mov al, 128

cbw

mov bl,2;

div bl;

1. al = 40
2. al = 40d
3. al = 40,5
4. execution error - T (overflow)
5. sintax error, cbw should not be used there
6. al = 40h
7. al = 0100\_0000

## Answers:

mov al, 128 ; 128d = 80 = 1000\_0000

cbw => ax: 1111\_1111\_1000\_0000 = FF 80: 88 40, because the first bit is 1, and we use conversion for signed numbers, it is considered a sign bit so the values in AH register will be filled with 1.

mov bl,2; bl =2

div bl; AX is a large number, when divided by two, it will not fit the size of the AL, so it will raise an execution error.

Exercise:

1. What value will generate the following code:

mov ax, 86h

cbw

mov bl,2

div bl

How about:

mov ax, 86h

cbw

mov bl,2

idiv bl

Hint: negative numbers, divisions. How is represented 86h in memory? If idiv is used how is considered the sign bit?

# ADC and SBB instructions

mov ax, [a]

mov bx, [a+2]

mov cx, [b]

mov dx, [b+2]

add ax, cx

adc bx, dx

# Array Lengths

1. Considering the variables, witch of the length definition is correct (returns the correct length value of the string A and B).

a db 1,2,3,4,5;

lenA1 equ $-a

b db 1,2,3,4,5;

c dw 1,2,3,4,5 ;

lenB equ $-b;

lenC equ $-c;

lenC2 equ $-c/2;

lenC3 equ $-b –c

1. What is the effect of; „b equ 123456h” in memory? How many bytes are allocated?

## Answers:

* lenA1 – correct, return the length of A
* lenB – incorrect, should be defined straight after b was defined, other ways it also takes into consideration the lengths of C
* LenC – counts the number of bytes defined, but C is type word, so it does not compute the length of C
* lenC2 – incorrect, Parenthesis are missing, the correct would be: lenC2 equ ($-c)/2;
* lenC3 – does not raise a syntax error, but is incorrect

1. This is a tricky question, do constant have allocated space in memory? No. Zero bytes are allocated in the memory.

## Times instruction

1. What is generated in the following sequence:

a times 2 db 0

b times 3 dw 5

c times 2 dw 7,9

d times 4 db ‘abc’, 0

e times 2 db 1,2,3

g times 4 equ 5

1. What is wrong in the following sequence:

a dw 1,2,3,4,56,0feh

la equ ($-a)/2

b dd 'abc'

lb dw ($-b)/4

sir\_a times la dw 0

sir\_b times lb dd -1

## Answers:

1. The TIMES directive allows multiple initializations to the same value. The memory will look like:

a times 2 db 1; 01 01

b times 3 dw 5; 05 00 05 00 05 00

c times 2 dw 7,9; 07 00 09 00 07 00 09 00

d times 4 db ‘abc’, 0; syntax error

e times 2 db 1,2,3 01 02 03 01 02 03

g times 4 equ 5; syntax error. EQU cannot be used here

1. Times does not allow to have a nonconstant argument, so because lb is not a constant, there will be an assembly error.
   1. How to solve it: define lb using equ (it becomes constant)

### Exercise:

What value will generate the following instruction?

b times 3 dw -5

### $ and $$

1. What is the value of the $ after the following definitons:.

a dd 0ab123456dfh

b times 3 dw 1ah

c times 6 db 'd'

1. What is the value of the $ after the following definitions:

a dw 0ab1210ab10h

b times 2 dw 1ah, 0hDc;

c equ 6

d times 6 db 'b'

Memory layout:

10 ab 1a 00 dc 00 1a 00 dc 00 62 62 62 62 62 62

Note: hexanumbers are not case senzitive:

0hAC = 0ACh= 0ACH = 0ACx

1. What will be the resulted values for len variables?

a db 1,2,3,4,5;

b db 1,2,3,4,5;

c dw 1,2,3,4,5 ;

lenA1 equ $-a;

lenA2 equ $-b;

lenA3 equ $-c;

lenB2 equ $-c/2;

lenB2 equ ($-c)/2;

d dd 1,2,3,4,5 ;

lenB4 equ ($-d); =

lenB4a equ ($-d)/4

lenB5 equ $-b – c;

Response:

a db 1,2,3,4,5; a is defined as byte, so for each element a touple will be defined, in total 5 bytes

b db 1,2,3,4,5; 5 bytes (same explanation as a)

c dw 1,2,3,4,5 ; 10 bytes (5 elements each one stored on a word, a word has two bytes, so 5\*2 = 10 bytes in total)

$ - the $ sign specifies the current point in the memory relative to the start of variable definitions (relative to “a” variable in this case)

lenA1 equ $-a; 20 = 5 (size occupied by variable a) + 5 (size occupied by b) + 10 (size occupied by variable c

lenA2 equ $-b; 15 = because we go only from variable b, so we will have: 5+10

lenA3 equ $-c; 10

lenB2 equ $-c/2; 100400 - 100400/2 = 50xxx!!!

lenB2 equ ($-c)/2; = 5

d dd 1,2,3,4,5 ;

lenB4 equ ($-d); = 4\* 5 elem = 20

lenB4a equ ($-d)/4

lenB5 equ $-b – c; = error

# Byte Shifts

1. Which instructions will have the same effect on AH register?

1. mov ax,65432>>4
2. mov ax, -1>>4;
3. mov ax, -1>>8;
4. mov ax, 0BFFFFFh>>12;
5. mov ax, 0AFFFFFh>>4;

Example of possible answers:

1. 1), 3)
2. 3), 4)
3. 2), 3)
4. 2), 3),5)
5. All 4 sequences of instructions have the same effect
6. Each of the 5 sequences of instructions will have a different effect on AH

Answer: d)

1. Which of the next sequences of instructions have the same effect on the EAX register?

1) or eax,-1; not eax; shr eax,1;

2) mov eax, 0; mov bl, 3 ^ 3; mul bl; sar eax,1;

3) or eax,-1; xor eax,-1; shl eax,1;

Example of possible answers:

1. 1), 3)
2. 1), 2)
3. 2), 3)
4. 1), 2), 3)
5. each of the above sequences will have a different effect on the EAX register

Answer: a), because for 2), EAX may not be zero before the sequence

1. What will be the result of the following instructions (the value from **bx** register)

a dw 0111011101010111b

mov bx, 0 ;

mov ax, [a] ;

and ax, 0001110000000000b

mov cl, 10

ror ax, cl ;

or bx, ax ;

1. What will be the result of the following instructions (the value from **bx** register)

a dw 1001101110111110b

mov bx, 0 ; bx register is used for the result

mov ax, [a] ;

and ax, 0000000000011110b

mov cl, 6

rol ax, cl

or bx, ax

1. What are the results after running the following code: (the values in ax,bx)

mov bx,0

mov al, 00011110b

shr al, 2 ;

adc bx,1

Hint: when shr is performed, the byte is saved in CF. Using ADC instruction, the value from CF is also added to the result

IN WORK:

1. What will I have in ch?

mov ch, -1 rol 6;

# Operators

What is the result of the following instruction?

mov al, (3&4)^(13^(\~13)

Explanations

& =AND

^ = XOR

~ = Complement to one

| = OR

! = NOT

% = reminder unsigned

%% = reminder signed

<< = shift

>> = shift

/ = unsigned div

// = signed div

In work!

1. Which instructions are correct:

not byte [a],3

not byte [a], byte [a]

not 3, byte [a]

not byte [a],3 & 6

not 3,3

# LEA & XLAT & MOVSX Instruction

What is the effect of

* LEA EBX,[5]
* LEA AX,[sir]

MOVSX- move with sign extended.

MOVSX is an assembly instruction that is used to copy data from a source operand to a destination operand where the sign occupies the extra bits in the destination. If the source operand stored a negative number, then 1 is stored in the extra bits. For positive number, 0 is stored in the extra bits.

Destination should be a register on 16, 32 or 64 bițs

Exemple:

movzx al,[a]

movzx ax,[a]

movzx eax,word[a]

* The first one gives an error as AL register has only 8 bits (one byte)
* The second 2-a extends byte to word with zero-extension

Exercise:

1. What is the difference between the following instructions:

v db 9;

lea eax,[ebx+v-6]

mov eax, [ebx+v-6]

1. What will generate the following instructions:

lea eax, [eax \* 2]

lea eax, [eax \* 4 + eax]

# Memory addressing

Which one of the follwing address(es) is/are valid?

a) mov eax, [eax\*9 + 12]   
b) mov eax, [ss:ebx + eax + 3]   
c) mov eax, [esi + 2\*esp + 1]

d) mov eax, [esp + 2\*esi + 1]  
e) mov eax, [ebx + 3 \* eax + 1]  
f) mov eax, [edx + 4 \* eax + 2]  
g) mov eax, [edx + 9 \* eax]  
h) mov eax, [edx + eax \* 9]

Answer:

The memory addressing uses the following formula: {ss, ds, cs, es}: baze + index \* scale + number;

* The scale is a number equal with power of 2: 0, 1, 2, 4, 8.
* The base can be prefixed with ss, ds, cs, es, where ss represents the stack segment, cs the code segment, ds data segment, ...
* NOTE: one register can not be used as index

a) mov eax, [eax\*9 + 12] ;=> is correct because can be written as eax (base) + 8\*eax (scale) + 12   
b) mov eax, [ss:ebx + eax + 3] => ok  
c) mov eax, [esi + 2\*esp + 1] => which is the register that can not be used as index?  
d) mov eax, [esp + 2\*esi + 1]  
e) mov eax, [ebx + 3 \* eax + 1]; not ok  
f) mov eax, [edx + 4 \* eax + 2]; ok  
g) mov eax, [edx + 9 \* eax]; not ok  
h) mov eax, [edx + eax \* 9] ;= not ok

Exercises:

mov ax, [eax+eax+a+2] ; base, index and displacement

mov ax, [a+4+ebx\*2] ; index, scale (i.e. 2) and displacement

mov ax, [eax + ebx\*4 + 20];

mov ax, [eax + ebx\*5 + 20];

mov ax, [a+20]

# Implicit and explicit operands

1. What are implicit operands?
2. What are explicit operands?
3. How many operands and which type have the following instructions:

mov ax, bx

mul al

# Stack instructions

a dw 5

b dw 6;

c dd 112233h

mov ax, [a]

mov bx, [b]

mov ecx, [c]

push ecx

push ax

pop eax ;

push bx

pop ebx ;

What will be in eax, ebx after running the code?

# Jumps & loops

STILL IN WORK!

1. What will be in ax after running the following code?

mov ax, -4

mov bx, 3

cmp ax, bx

jl label1

inc ax

label1:

1. What will return the following code:
   1. mov ax,5

cmp 4,ax

* 1. mov bx,4

mov ax,4

cmp ax,bx

Hint: check cmp instruction format!

1. How many times is executed the code inside the loop?:

mov ax,5

mov ecx,1

sub ax, 2

sub ecx,2

repeta:

inc ax

loop repeta

a) once

b) sintax err

c) execution err

d) 256 ori

e) 65535 ori

f) infinit

g) FFFFFFFFori

1. How many times is executed the code inside the loop?:

mov ax,5

mov ecx,3

sub ax, 2

sub ecx,2

repeta:

inc ax

loop repeta

a) once

b) sintax err

c) execution err

d) 256 ori

e) 65535 ori

f) infinit

g) FFFFFFFFori