

Conversions

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

-808(d) → (b) on 16bits

FC D8
1111 1100 1101 0100

Handwritten conversion of -808 to 16-bit two's complement binary:

$$\begin{array}{r}
 65536 - \\
 \underline{808} \\
 64728 \\
 \hline
 8
 \end{array}
 \begin{array}{r}
 16 \\
 4045 \\
 \hline
 13=D
 \end{array}
 \begin{array}{r}
 16 \\
 252 \\
 \hline
 16 \\
 92 \\
 \hline
 12=C
 \end{array}
 \begin{array}{r}
 16 \\
 15 \\
 \hline
 1 \\
 0
 \end{array}
 \begin{array}{r}
 16 \\
 1 \\
 \hline
 0
 \end{array}$$

Answers:

1. The first number is in binary, so we can apply grouping by four: 1010_0011b=>A3h
The second number 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
2. Results:
 - a. 1 1000 0110
 - b. C4
3. 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

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
2. 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'
3. 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];
```
4. Are the following definitions equivalent?
 - a) b0 db 256;
b1 dd 100h;
 - b) b2 db 1203h;
b3 db 3;
5. Do the instructions generate the same result?
 - v db 4;
x db 5;
mov ax, 4;
mov ax, v;
 - d) v db 4;
x db 5;
y db 66h;
z dw 11_22h;

```
mov ax, 4;
mov eax, dword [v];
```

6. What values will be in AL register after running the following code?

```
mov al, Aah
```

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

```
R resb 4;
a resw 2;
c resd 3;
d resq 4
e rest 1
```

8. 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
```

9. 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
```

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

2.

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

3. 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

4. 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.

5. Do the instructions generate the same result?

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.

d)

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;
```

6. 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`
7. 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
```

8. The memory will look like this taking into account that for a byte we have 2 digits in hexa (a couple) and for a word we have 4 digits, 2 groups of 2. Also, because of little endian representation, the numbers 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
```

9. 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
```

Multiplications and divisions

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

```
mov ax, [a]
add ax, [b]
div [c]
```

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

```

mov ax,[a]
add ax, [b]
div 2

```

- c. a is variable of type byte where a =8

```

mov ah,[0]
mov al, [a]
mul -3

```

- d. 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

```

- e. 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

```

- f. 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

```

- g. 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

```

2. 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.

Answers:

1.

- a. A division to word implies DX:AX registers, DX register value was not set, so the result of the division cannot be specified.
- b. There will be an assembly error as division and multiplications cannot be performed with numbers
- c. Check 1.b.
- d. 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.
- e. 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
- f. 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 remainder will be 6 (stored in AH)
- g. 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 it will be AL = -3 = FD and AH = 0;

2. 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)

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. syntax 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.

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

Considering the variables, which 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
```


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

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
```

2. 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 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

2. Times does not allow to have a nonconstant argument, so because lb is not a constant, there will be an assembly error.

a) How to solve it: define lb using equ (it becomes constant)

Byte Shifts

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

- ~~0FFC8 >>~~ 65535 -
 1) ~~mov ax, 65432 >> 4; AH=0Fh~~ 5432
 2) ~~mov ax, -1 >> 4; FFh~~ 00103
 3) ~~mov ax, -1 >> 8; FFh~~
 4) ~~mov ax, 0BFFFFFh >> 12; 00h 0Bh~~
 5) ~~mov ax, 0AFFFFFh >> 4; FFh~~

F F F F -
 0 0 0011 0111
 F F 1100 1000

NOT
 shl !!
 like sar

Example of possible answers:

- a) 1), 3)
- b) 3), 4)
- c) 2), 3)
- ☒ d) 2), 3), 5)
- e) All 4 sequences of instructions have the same effect
- f) 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:

- a) 1), 3)
- b) 1), 2)
- c) 2), 3)
- d) 1), 2), 3)
- e) 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

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

- a)


```

a dw 0111011101010111b
mov bx, 0 ;
mov ax, [a] ;
and ax, 0001110000000000b
mov cl, 10
ror ax, cl ;
or bx, ax ;
      
```
- b)


```

a dw 1001101110111110b
mov bx, 0 ; in bx se calc rezultat
mov ax, [a] ;
and ax, 00000000000011110b
mov cl, 6
      
```

```

rol ax, cl
or bx, ax

```

LEA & XLAT & MOVSX Instruction

What is the effect of

- LEA EBX,[5]
- LEA AX,[si]

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 bits

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

Memory addressing

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

ss, ds, cs, es: baza + index * scale + numar; scale: 0, 1, 2, 4, 8
ss: stack segment, cs: code segment...

- code*
stack
data
extra
- a) mov eax, [eax*9 + 12] ;=> $eax + 8*eax + 12$ ✓
 - b) mov eax, [ss:ebx + eax + 3] => ✓
 - c) mov eax, [esi + 2*esp + 1] => care e registrul care nu poate fi folosit ca index? *cannot be used as index*
 - 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] ✓

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

```

instruction with 2 implicit operands and 2 exp. op.