2 tiputi de interpretari

1) interpretare cu semn

2) fara semn

De aceea orice numar poate ffi interpretat cu sau fara semn

De exemplu, pentru reprezentari ce nu depasesc un byte:

145 se reprezinta pe un byte: 1001 0001

in intepretare fara semn, numarul 1001 0001 este 145

in interpretare cu semn, numarul 1001 0001 este -111(nr negativ)

ex2

0010 0010 in ambele interpretari este 34

nr e [0, 127] => nr pozitiv in ambele interpretari

Pornind de la un numar in baza 2:

a) daca numarul are bitul de semn 0 => se converteste in baza 10 direct

b) daca numarul are bitul de semn 1 => se calculeaza prima data complementul iar dupa se convertsete complementul in baza 10 si se pune semnul -

Pentru calcularea numarului complementar:

1) se calculeaza complementul fata de 1 si se adauga 1(cod invers)”~”

2) se calculeaza complementul fata de 2 “-”

3) se calculeaza metoda complementara rapida

1) pentru adunare si scadere – pentru date de aceeasi dimensiune

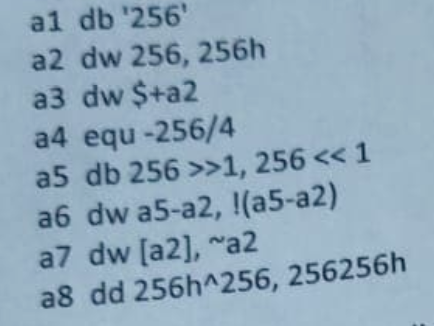
pentru date de dimensiuni diferite - conversie

2) pentru inmultire si impartire ai cate doua instructiuni pentru fiecare metoda de interpretare

instructiuni care opereaza atat cu neg cat si poz: add, sub, sbb, adc, idiv, imul, cmp

instructiuni care opereaza numai cu numere pozitive: div, mul

[-128, 127]



32 35 36 00 01 56 02 80 00 04 00 00 00 56 03 00 00 56 62 25 00

a1 a2 a5 a6 a8

a1 db “256”

a2 dw 256, 256h

256 = 100h 0100h

256h = 0256h

a3 dw $+a2

=> ARITMETICA DE POINTERI!!!!!!!!!!!!!!!!!!!!!!!

a4 equ -256/4

=> ASTA E CONSTANTA !!!!!

a5 db 256 >> 1, 256 << 1

256 >> 1

128 = 80 h

256 << 1

0 = 0

a7 ADRESARE DE MEMORIE IN DATA SEGMENT?????

a8 256h^256

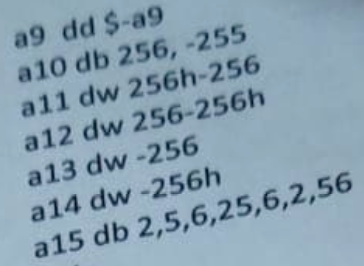
0000 0010 0101 0110

0000 0001 0000 0000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0000 0011 0101 0110

356h



a9 dd $-a9 = 0

32 35 36 00 01 56 02 80 00 04 00 00 00 56 03 00 00 56 62 25 00 00 00 00 00

a1 a2 a5 a6 a8 a9

00 01 56 01 AA FE 00 FF AA FD 02 05 06 19 06 02 38

a10 a11 a12 a13 a14 a15

a10 db 256, -255

-255 = 1h

255 = 1111 1111

1 0000 0001

a11 dw 256h -256

256h- 100h =156h

256 – 256h = -156 h

156h = 0001 0101 0110

1110 1010 1010 = FEAA

a13 = -256

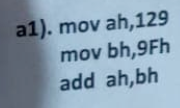
0000 0001 0000 0000

1111 1111 0000 0000

a14 = -256h

0000 0010 0101 0110

1111 1101 1010 1010 = FDAA



mov ah, 129

mov bh, 9fh

add ah, bh

ah = 129 (-127)

bh = 9fh = 9\*16 + 15 = 159(-97)

159 = 128 + 16 +8 + 4+ 2+1

129+ 159 >256 => CF=1

-127 + -97 <128 => OF = 1

SF = 0

PF =0

AF =1

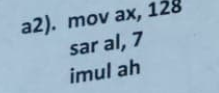
ZF =0

1000 0001

1001 1111

\_\_\_\_\_\_\_\_\_

1 0010 0000



mov ax, 128 ax = 0000 0000 1000 0000

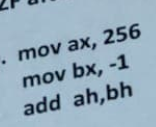
sar al, 7 al = 1111 1111

imul ah ax = al\*ah

ax = 0000 0000

CF = OF = 0 ZF = 1 AF = nedeterminat la inmultire si impartire

SF = 0 PF = 1

f

mov ax, 256 ax = 0000 0001 0000 0000

mov bx, -1 bx = 1111 1111 1111 1111

add ah, bh

ah = ah+bh

ah = 0000 0001

1111 1111

ah = 0000 0000

CF=1

OF=0

SF = 0

ZF = 1

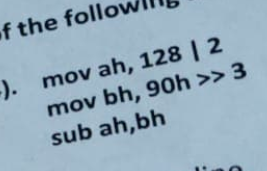
PF = 1

AF = 1

IF = ?

TF = ?

DF = ?



mov ah, 128|2

mov bh, 90h >> 3

sub ah, bh

128 =

1000 0000

0000 0010

\_\_\_\_\_\_\_\_\_

1000 0010

ah = 1000 0010

mov bh, 90h >> 3

90h = 1001 0000 >>3

bh = 0001 0010 = 18

1000 0010-

0001 0010

\_\_\_\_\_\_\_\_

0111 0000

SF = 0

CF = 0

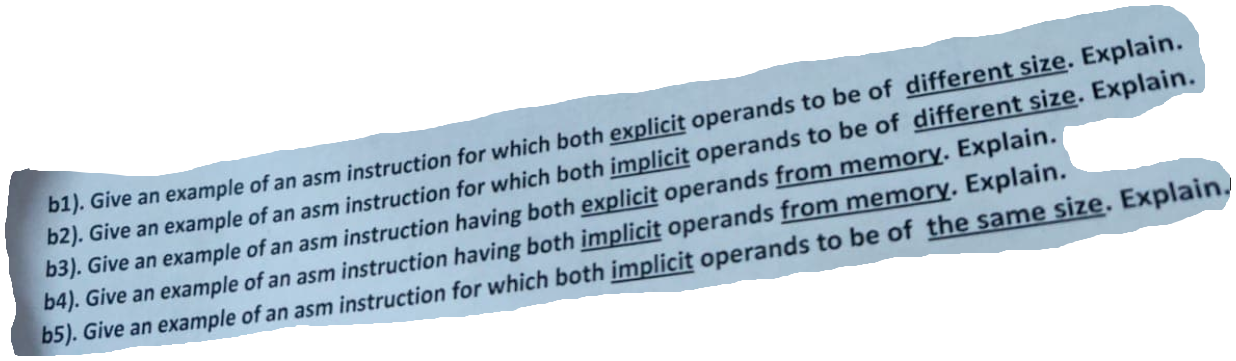
OF = 1

ZF = 0

AF = 0

PF = 0

DF IF TF?????



1)2 expliciti, dimensiuni diferite => mov ax, eticheta

movzx eax, bx

2)2 impliciti, dimensiuni diferite => cwde

3)-

4)movsb

5)2 impliciti de aceeasi dimensiune => cmpsb

bits 32   
  
global start   
  
extern exit, scanf, gets, printf, fopen, fclose, fread, fscanf, fprintf   
import exit msvcrt.dll   
import scanf msvcrt.dll  
import gets msvcrt.dll  
import printf msvcrt.dll  
import fopen msvcrt.dll  
import fclose msvcrt.dll  
import fread msvcrt.dll  
import fscanf msvcrt.dll  
import fprintf msvcrt.dll   
  
segment data use32 class=data  
 ; ...  
 nr dd 0  
 sir times 20 dd 0  
 sir2 times 20 dd 0  
 format db "%d", 0  
 zece dd 10  
  
segment code use32 class=code  
 start:  
 ; ...  
 push nr  
 push format  
 call [scanf]  
 add esp, 4\*2  
   
 mov ecx, [nr]  
 mov edi, sir  
 citire:  
 pushad  
 push edi  
 push format  
 call [scanf]  
 add esp, 4\*2  
 popad  
 times 4 inc edi  
 loop citire  
   
 mov ecx, [nr]  
 mov esi, sir  
 mov edi, sir2  
 pare:  
 lodsd  
 mov ebx, 0  
 cifre:  
 mov edx, 0  
 div dword[zece]  
 test edx, 1  
 jnp out2  
 add ebx, edx  
 out2:  
 cmp eax, 0  
 jne cifre  
 mov eax, ebx  
 stosd  
 loop pare  
   
   
 ; exit(0)  
 push dword 0   
 call [exit]