LAB 2

Db x 13- x offset has value 0 (0x0)

Db y 12- y has value 1 e la dist de 1 byte de x (0x1)

Db z, 11, 12, 13, 14, “x”

ADD EAX, y (se adauga 1)

ADD EAX, [y] (se adauga val)

Dw x, 13 (ocupa 2 bytes)

Dd x, 13 (4 bytes)

Endianness-cum e repr orice mai mare decat db

Big-normal

Little endian (asa avem noi)

X 13

Y 00 0C-NU ASA E REPR

Y 0C 00-rasturnam

(accepta 2 operanzi)

-amandoi operazi trebuie sa aiba acelasi tip!!!!!

**ADD**

Db x 10,11,12,13

ADD EAX, [x]-o sa adauge tot din cauza registrului

-daca vrem sa adaugam doar 10 de ex

ADD, AL, [x]

-ADD BYTE [x],2

X 12 11 12 13

ADD WORD [x],2

Db x 0x10 0x11 0xAB

10-0x1011

0x1110+2

0x1112

Se inverseaza si se ia 12

**SUB**

-la fel ca la ADD

Ex1

MOV AL, [c]

MOV AH, [a]

ADD AH, [d]

SUB AL, AH

MOV AH, [b]

ADD AH, [d]

ADD AL, AH

push dword 0

call [ExitProcess]

ex 2

MOV AX, 10

MUL BYTE [a]

ADD AX, [d]

MOV DX, AX

MOV AX, [b]

MUL BYTE [c]

SUB DX, AX

push dword 0

call [ExitProcess]

(accepta 1 operand)

**MUL**

**Text

Description automatically generated with medium confidence**

**Graphical user interface, text

Description automatically generated**

* Inmultire fara semn
* MUL BYTE[X]->AX
* 04 00 (Ox0400 da e rasucit)
* MUL WORD[X]->AX\*X->DX\*X
* MUL DWORD[X]->EAX\*X->EDX\*EAX

**DIV**

**Text, letter

Description automatically generated**

-DIV BYTE[X]

- AX: [X] -> AL R AH

- EAX:[X] -> AX R DX(ce e inaintea lui AX)

**NEG**

**INC**

-nu putem avea comb mem mem sau const const

Ex ADD [x],[y]

ADD 2,[y]

-X RES B

W

D

Q

Pb 1 de la adunari scaderi

LAB 3

Push eax⬄ mov [esp], eax

Add esp, 4

A dq 1234|5678 h

B dq abcd|ef12 h

A+b

Mov eax, [a]

Mov edx, [a+4]

Mov ebx, [b]

Mov ecx, [b+4]

Add eax, ebx

Add edx, ecx

Mul al->ax

Div ax->dx:ax

X db 2

Mov al, ff h **(255)-max pe 2 biti**

Mul byte [x]

AH AL

01 DC(de ex ceva in hexa)=~255\*2510

Ff fe(-2)=~-1\*2

Imul byte [x]

Uneori nu conteaza semnul!!!

Mov al, 1

Imul 2

Mul 2

Conversii

Tin cont de semn(umple cu 1 sau 0 in functie de bitul de semn daca e setat)

CBW AL->AX

AL FFh

AX FFFF h

AL 2h

AX 2h

CWD AX->DX:AX

CWDE AX->EAX

CDQ EAX->EDX:EAX

AL FFh

Vrem sa fie aceeasi chestier in AX 255

Mov ax, 0

Mov al, 255

Mov ah, 0

LAB 4

**BITWISE OPERATORS**

**Graphical user interface, text, application, email

Description automatically generated**

XOR-exclusiv

0 0 0

0 1 1

1 0 1

1 1 0

Or 1,2=3 0011

1 0001

2 0010

TEST x,y modifica registrul EFLAGS

AND x,y

TEST [x],0-nu stocheaza rez

ZF=1(FLAG urile sunt 1 cand e adev ce repr)

C++

If(x!=0){

…

}

Else{

…

}

Assembly

TEST [x],0

Jz ELSE

…

ELSE:

…

SH

A picture containing diagram

Description automatically generated

Functioneaza cu const pe 8 biti sau registru cl

SHL left

SHR right

AL 11110000

SHR AL, 2

00111100

SHL AL, 2

11000000

ROL

ROR

AL 11110000

ROL 1

AL 11100001

ROL 1

11000011

RCL AL, 1

AL 1111 0000

CF =1

AL 1110000CF (1)

RCL AL, 1

1100 00 CF 1

RCR

B dw 10 (vrem sa facem cv cu bitii din mijloc)

000011110000 xor

Shl primii 4

Shr ult 4

Shl

[flavius.ilinoiu@gmail.com](mailto:flavius.ilinoiu@gmail.com)

pt lab 4 pb 2

LAB 5

TEST D&S ZF/PF

CMP D-S ZF/PF

JUMP LABEL

LABEL:

…

IF(..)

GO TO END:

END:..

CMP EAX,EBX

JL LABEL (EAX<EBX)

LABEL:.. = ADRESA DE MEMORIE (SE TRADUCE INTR UN OFFSET)!!!!

[JL, JG] SIGNED

[JB JA] UNSIGNED

JLG <= JAE >=

JE JBE

JZ

JP

JS

LAB 5

Esi adresa de inceput a string ului

Edi -||-

Lods b/w/d – ia de la adresa [esi] si il pune in registrul coresp al/ax/eax

Esi + 1/2/4

Stos b/w/d

Al/ax/eax in edi

Movs

Repeat

Lodsb

Stosb

(movsb)

Loop repeat

Mov esi , a

Mov edi, b

Muta din esi in al

Din al in

(e) flag-ds <- porneste de la capat in functie de ds

Scasb/w/d comparatia intre al/ax/eax cu [esi] -scaneaza sursa

Cmpsb/w/d cmp(diferenta) intre esi si edi

Rep movsb

Rep stosb

Rep lodsb

Stc

Clc

Std

Cld-de ex cand vrem sa parcurgem sirul de la st la dr

Stz

Clz

Tema lab 6 pb 10