Tema laborator 2

Pop Mihai-Daniel, Grupa 215/2

ADUNARI SI SCADERI (4 exercitii)

1. BYTE – Ex. 4 (a-b)+(c-b-d)+d

; Scrieți un program în limbaj de asamblare care să rezolve expresia aritmetică, considerând domeniile de definiție ale variabilelor

; adunari, scaderi, a,b,c,d byte, ex.4

; a - byte, b - byte, c - byte, d - byte

; (a-b)+(c-b-d)+d;

; ex. 1: a=40; b=10; c=50; d=20; Rezultat: (40-10)+(50-10-20)+20 = 30+20+20 = 70

bits 32 ;asamblare si compilare pentru arhitectura de 32 biti

; definim punctul de intrare in programul principal

global start

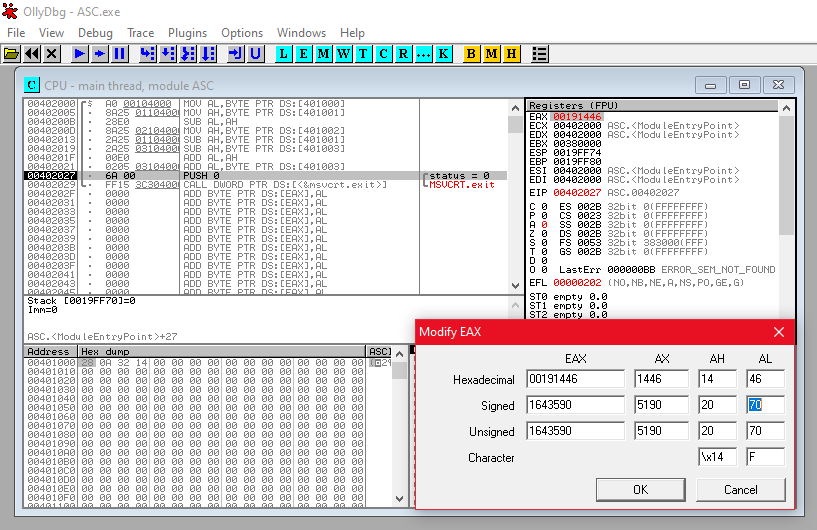
extern exit ; indicam asamblorului ca exit exista, chiar daca noi nu o vom defini

import exit msvcrt.dll; exit este o functie care incheie procesul, este definita in msvcrt.dll

; msvcrt.dll contine exit, printf si toate celelalte functii C-runtime importante

segment data use32 class=data ; segmentul de date in care se vor defini variabilele

a db 40

b db 10

c db 50

d db 20

segment code use32 class=code ; segmentul de cod

start:

mov AL, [a] ;AL = a

mov AH, [b] ;AH = b

sub AL, AH ;AL = AL - AH = a - b = 40 - 10 = 30

mov AH, [c] ;AH = c

sub AH, [b] ;AH = AH - b = c - b = 50 - 10 = 40

sub AH, [d] ;AH = AH - d = c-b-d = 40 - 20 = 20

add AL, AH ;AL = AL + AH = 30 + 20 = 50

add AL, [d] ;AL = AL + d = 50 + 20 = 70

push dword 0 ;se pune pe stiva codul de retur al functiei exit

call [exit] ;apelul functiei sistem exit pentru terminarea executiei programului

1. BYTE – Ex. 27 (a+b-c)-(a+d)

; adunari, scaderi, a,b,c,d byte, ex.27

; a - byte, b - byte, c - byte, d - byte

; (a+b-c)-(a+d);

; ex. 1: a=11; b=12; c=13; d=14; Rezultat: (11+12-13)-(11+14) = 10-25 = -15

bits 32 ;asamblare si compilare pentru arhitectura de 32 biti

; definim punctul de intrare in programul principal

global start

extern exit ; indicam asamblorului ca exit exista, chiar daca noi nu o vom defini

import exit msvcrt.dll; exit este o functie care incheie procesul, este definita in msvcrt.dll

; msvcrt.dll contine exit, printf si toate celelalte functii C-runtime importante

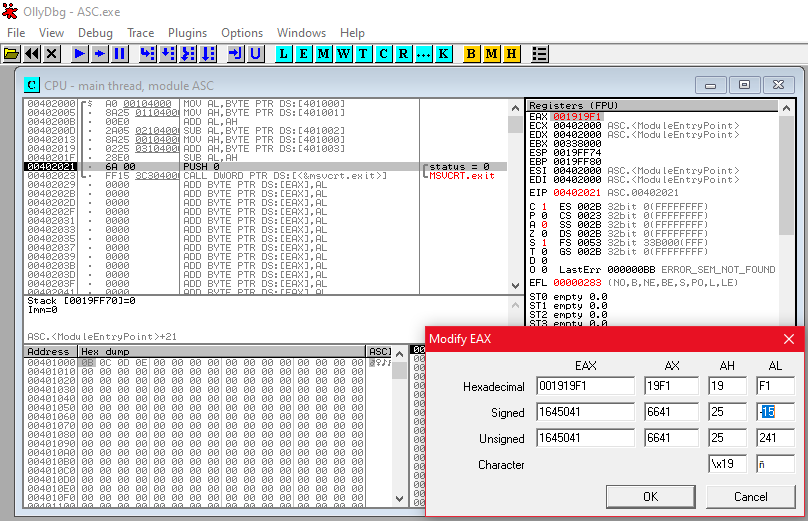
segment data use32 class=data ; segmentul de date in care se vor defini variabilele

a db 11

b db 12

c db 13

d db 14

segment code use32 class=code ; segmentul de cod

start:

mov AL, [a] ;AL = a

mov AH, [b] ;AH = b

add AL, AH ;AL = AL+AH = a+b = 11+12 = 23

sub AL, [c] ;AL = AL-c = 23-13 = 10

mov AH, [a] ;AH = a

add AH, [d] ;AH = AH+d = 11+14 = 25

sub AL, AH ;AL = AL-AH = 10-25 = -15

push dword 0 ;se pune pe stiva codul de retur al functiei exit

call [exit] ;apelul functiei sistem exit pentru terminarea executiei programului

1. WORD – Ex. 4 (b+b)-c-(a+d)

; adunari, scaderi, a,b,c,d word, ex.4

; a - word, b - word, c - word, d - word

; (b+b)-c-(a+d);

; ex. 1: a=250; b=555; c=696; d=123; Rezultat: (555+555)-696-(250+123) = 1110-696-373 = 414-373 = 41

bits 32 ;asamblare si compilare pentru arhitectura de 32 biti

; definim punctul de intrare in programul principal

global start

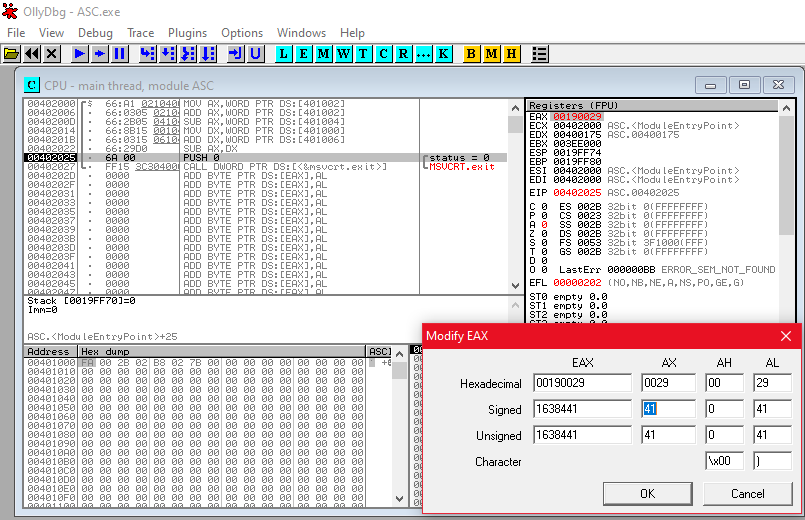
extern exit ; indicam asamblorului ca exit exista, chiar daca noi nu o vom defini

import exit msvcrt.dll; exit este o functie care incheie procesul, este definita in msvcrt.dll

; msvcrt.dll contine exit, printf si toate celelalte functii C-runtime importante

segment data use32 class=data ; segmentul de date in care se vor defini variabilele

a dw 250

 b dw 555

c dw 696

d dw 123

segment code use32 class=code ; segmentul de cod

start:

mov AX, [b] ;AX = b

add AX, [b] ;AX = AX+b = 555+555 = 1110

sub AX, [c] ;AX = AX-c = 1110-696 = 414

mov DX, [a] ;DX = a

add DX, [d] ;DX = DX+d = 250+123 = 373

sub AX, DX ;AX = AX-DX = 414-363 = 41

push dword 0 ;se pune pe stiva codul de retur al functiei exit

call [exit] ;apelul functiei sistem exit pentru terminarea executiei programului

1. WORD – Ex. 27 a+b-(c+d)+100h

; adunari, scaderi, a,b,c,d word, ex.27

; a - word, b - word, c - word, d - word

; a+b-(c+d)+100h;

; ex. 1: a=980; b=1220; c=450; d=540; Rezultat:

bits 32 ;asamblare si compilare pentru arhitectura de 32 biti

; definim punctul de intrare in programul principal

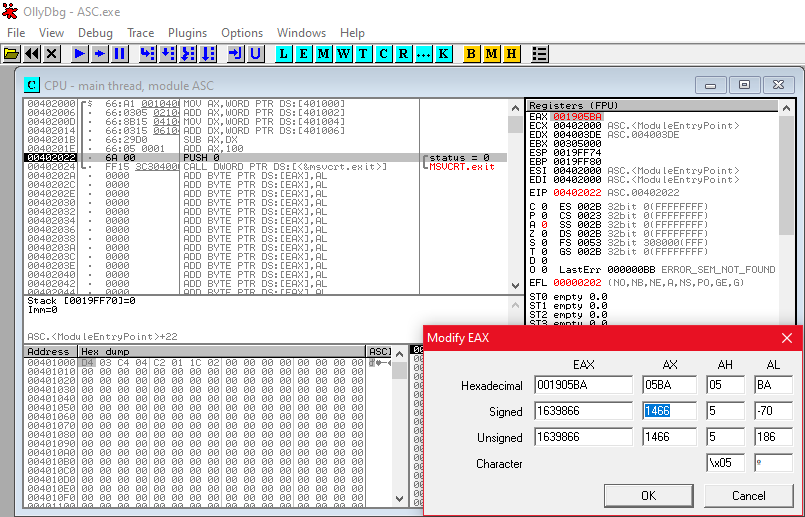
global start

extern exit ; indicam asamblorului ca exit exista, chiar daca noi nu o vom defini

import exit msvcrt.dll; exit este o functie care incheie procesul, este definita in msvcrt.dll

; msvcrt.dll contine exit, printf si toate celelalte functii C-runtime importante

segment data use32 class=data ; segmentul de date in care se vor defini variabilele

 a dw 980

b dw 1220

c dw 450

d dw 540

segment code use32 class=code ; segmentul de cod

start:

mov AX, [a] ;AX = a

add AX, [b] ;AX = AX+b = 980+1220 = 2200

mov DX, [c] ;DX = c

add DX, [d] ;DX = DX+d = 450+540 = 990

sub AX, DX ;AX = AX-DX = 2200-990 = 1210

add AX, 100h ;AX = AX+100h = 1210+256 = 1466

push dword 0 ;se pune pe stiva codul de retur al functiei exit

call [exit] ;apelul functiei sistem exit pentru terminarea executiei programului

INMULTIRI SI IMPARTIRI (4 exercitii)

1. BYTE/WORD – Ex. 4 –a\*a + 2\*(b-1) – d

; Scrieți un program în limbaj de asamblare care să rezolve expresia aritmetică, considerând domeniile de definiție ale variabilelor

; inmultiri, impartiri, a,b,c byte, d word, ex.4

; a - byte, b - byte, c - byte, d - word

; –a\*a + 2\*(b-1) – d;

; ex. 1: a=10, b=9, c=0, d=-94 Rezultat: -10\*10+2\*(9-1)-(-94) = -100+16+94 = 10

bits 32 ;asamblare si compilare pentru arhitectura de 32 biti

; definim punctul de intrare in programul principal

global start

extern exit ; indicam asamblorului ca exit exista, chiar daca noi nu o vom defini

import exit msvcrt.dll; exit este o functie care incheie procesul, este definita in msvcrt.dll

; msvcrt.dll contine exit, printf si toate celelalte functii C-runtime importante

segment data use32 class=data ; segmentul de date in care se vor defini variabilele

a db 10

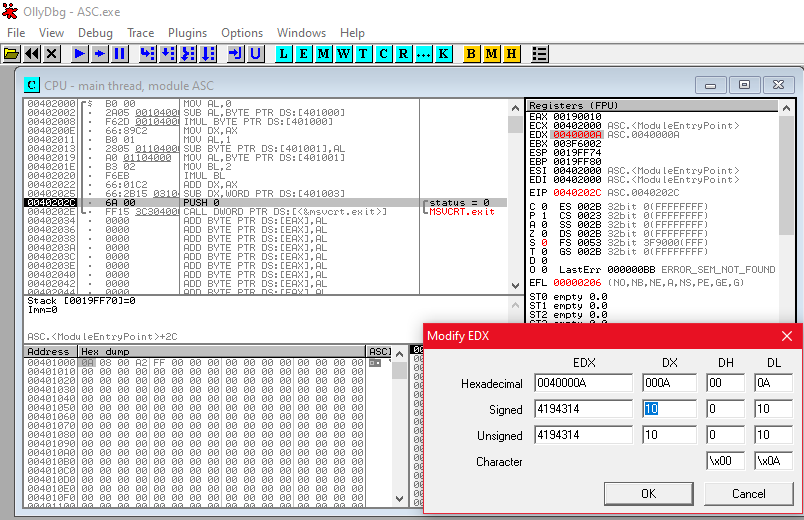
b db 9

c db 0

d dw -94

segment code use32 class=code ; segmentul de cod

start:

 mov AL, 0;

sub AL, [a] ;AL = AL-a = -a = -10

imul byte [a] ;AX = AL\*a = -10\*10 = -100

mov DX, AX ;DX = AX = -100

mov AL, 1 ;AL = 1

sub [b], AL ;b = b-AL = 9-1 = 8

mov AL, [b] ;AL = b = 8

mov BL, 2 ;BL = 2

imul BL ;AX = AL\*BL = 8\*2 = 16

add DX, AX ;DX = DX+AX = -100+16 = -84

sub DX, [d] ;DX = DX-d = -84-(-94) = 10

push dword 0 ;se pune pe stiva codul de retur al functiei exit

call [exit] ;apelul functiei sistem exit pentru terminarea executiei programului

1. BYTE/WORD – Ex. 27 d/[(a+b)-(c+c)]

; Scrieți un program în limbaj de asamblare care să rezolve expresia aritmetică, considerând domeniile de definiție ale variabilelor

; inmultiri, impartiri, a,b,c byte, d word, ex.27

; a - byte, b - byte, c - byte, d - word

; d/[(a+b)-(c+c)];

; ex. 1: a=15, b=25, c=-5, d=1000 Rezultat: 1000/[(15+25)-(-5+(-5))] = 1000/(40+10) = 1000/50 = 20

bits 32 ;asamblare si compilare pentru arhitectura de 32 biti

; definim punctul de intrare in programul principal

global start

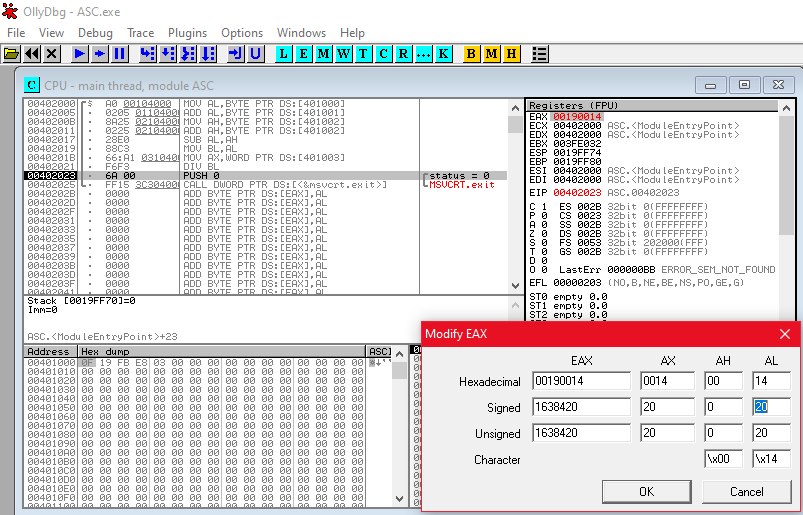
extern exit ; indicam asamblorului ca exit exista, chiar daca noi nu o vom defini

import exit msvcrt.dll; exit este o functie care incheie procesul, este definita in msvcrt.dll

; msvcrt.dll contine exit, printf si toate celelalte functii C-runtime importante

segment data use32 class=data ; segmentul de date in care se vor defini variabilele

a db 15

 b db 25

c db -5

d dw 1000

segment code use32 class=code ; segmentul de cod

start:

mov AL, [a] ;AL = a = 15

add AL, [b] ;AL = AL+b = 15+25 = 40

mov AH, [c] ;AH = c = -5

add AH, [c] ;AH = AH+c = -5+(-5) = -10

sub AL, AH ;AL = AL-AH = 40-(-10) = 50

mov BL, AL ;BL = AL = 50

mov AX, [d] ;AX = d = 1000

div BL ;AL = AX/BL = 1000/50 = 20

push dword 0 ;se pune pe stiva codul de retur al functiei exit

call [exit] ;apelul functiei sistem exit pentru terminarea executiei programului

1. BYTE/WORD – Ex. 4 (a-c)\*3+b\*b

; Scrieți un program în limbaj de asamblare care să rezolve expresia aritmetică, considerând domeniile de definiție ale variabilelor

; inmultiri, impartiri, a,b,c byte, d word, ex.27

; a - byte, b - byte, c - byte, d - byte

; e - word, f - word, g - word, h - word

; (a-c)\*3+b\*b;

; ex. 1: a=123, b=30, c=23, d=0 Rezultat: (123-23)\*3+30\*30 = 100\*3+900 = 1200

bits 32 ;asamblare si compilare pentru arhitectura de 32 biti

; definim punctul de intrare in programul principal

global start

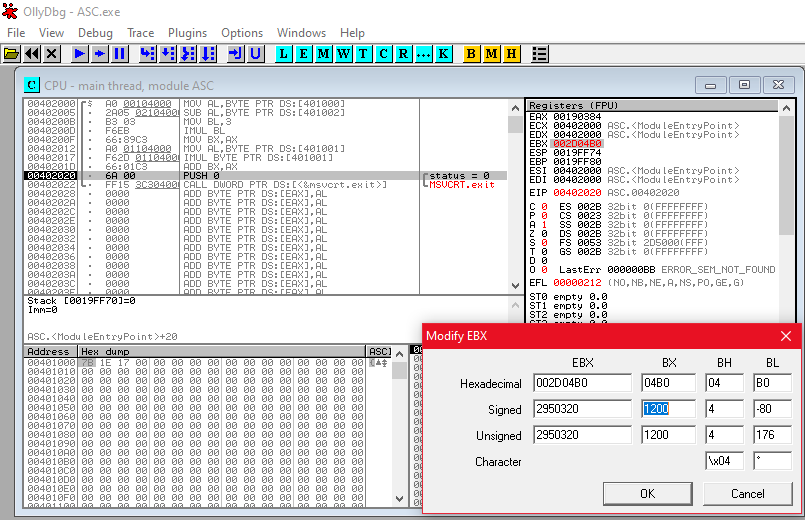
extern exit ; indicam asamblorului ca exit exista, chiar daca noi nu o vom defini

import exit msvcrt.dll; exit este o functie care incheie procesul, este definita in msvcrt.dll

; msvcrt.dll contine exit, printf si toate celelalte functii C-runtime importante

segment data use32 class=data ; segmentul de date in care se vor defini variabilele

a db 123

 b db 30

c db 23

d dw 0

segment code use32 class=code ; segmentul de cod

start:

mov AL, [a] ;AL = a = 123

sub AL, [c] ;AL = AL-c = 123-23 = 100

mov BL, 3 ;BL = 3

imul BL ;AX = AL\*BL = 100\*3 = 300

mov BX, AX ;BX = AX = 300

mov AL, [b] ;AL = b = 30

imul BYTE [b] ;AX = AL\*b = 30\*30 = 900

add BX, AX ;BX = BX+AX = 300+900 = 1200

push dword 0 ;se pune pe stiva codul de retur al functiei exit

call [exit] ;apelul functiei sistem exit pentru terminarea executiei programului

1. BYTE/WORD – Ex. 27 [(e+f-g)+(b+c)\*3]/5

; Scrieți un program în limbaj de asamblare care să rezolve expresia aritmetică, considerând domeniile de definiție ale variabilelor

; inmultiri, impartiri, a,b,c byte, d word, ex.27

; a - byte, b - byte, c - byte, d - byte

; e - word, f - word, g - word, h - word

; [(e+f-g)+(b+c)\*3]/5;

; ex. 1: a=0, b=27, c=23, d=0, e=555, f=705, g=805, h=0

; Rezultat: [(555+705-805)+(27+23)\*3]/5 = (500+50\*3)/5 = (500+150)/5 = 650/5 = 130

bits 32 ;asamblare si compilare pentru arhitectura de 32 biti

; definim punctul de intrare in programul principal

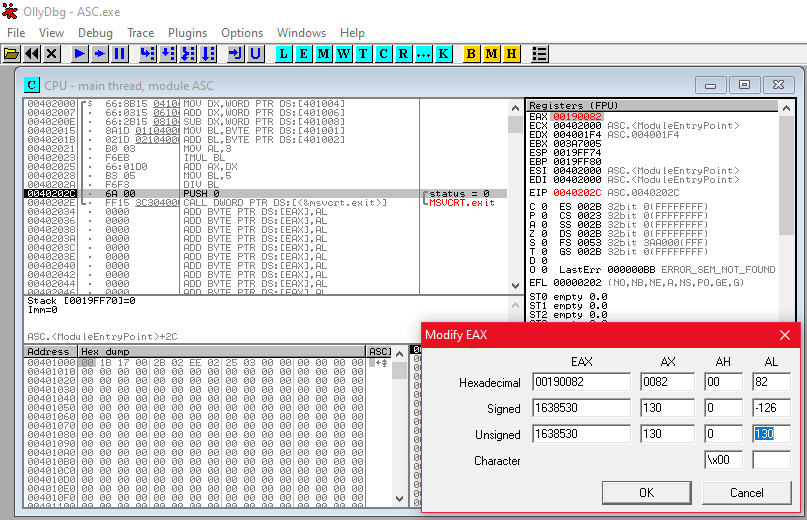
global start

extern exit ; indicam asamblorului ca exit exista, chiar daca noi nu o vom defini

import exit msvcrt.dll; exit este o functie care incheie procesul, este definita in msvcrt.dll

; msvcrt.dll contine exit, printf si toate celelalte functii C-runtime importante

segment data use32 class=data ; segmentul de date in care se vor defini variabilele

a db 0

b db 27

c db 23

d db 0

e dw 555

f dw 750

g dw 805

h dw 0

segment code use32 class=code ; segmentul de cod

start:

mov DX, [e] ;DX = e = 555

add DX, [f] ;DX = DX+f = 555+750 = 1305

sub DX, [g] ;DX = DX-g = 1305-805 = 500

mov BL, [b] ;BL = b = 27

add BL, [c] ;BL = BL+c = 27+23 = 50

mov AL, 3 ;AL = 3

imul BL ;AX = AL\*BL = 3\*50 = 150

add AX, DX ;AX = AX+DX = 150+500 = 650

mov BL, 5; BL = 5

div BL ;AL = AX/BL = 650/5 = 130

push dword 0 ;se pune pe stiva codul de retur al functiei exit

call [exit] ;apelul functiei sistem exit pentru terminarea executiei programului

Am o intrebare. La ultimul „div”, daca il pun „idiv” nu mai functioneaza bine/nu mai ajunge la rezultat/nu trece peste pasul respectiv in ollydbg, de ce?