ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ МОЛДОВЫ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ, ИНФОРМАТИКИ И МИКРОЭЛЕКТРОНИКИ

КАФЕДРА АВТОМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЕТ

По лабораторной работе №2

по дисциплине «Архитектура компьютеров»

Тема: Основы языка Ассемблер.

Выполнила: ст. гр. TI-155 Зверкова К.

Проверил: Колесник В.

Кишинев 2017

1. **Цель лабораторной работы**

Основные типы данных микропроцессора, системы счисления, перевод чисел в различные системы счисления, директивы резервирования данных, директивы сегментации.

1. **Произвести ассемблирование, исполнить в пошаговом режиме Debug и прокомментировать программу**

INCLUDE Irvine32.inc *; директива,которая подключает библиотеки irvine32.inc*

.data *; директива указывает начало сегмента данных*

alfa WORD 3 DUP(?) *; определяем массив из 3-х неопределённых слов со смещением alfa*

.code *; директива указывает начало сегмента кода*

main proc *; процедура с именем main*

mov ax,17 ; *заносим 17 в регистр AX*

mov ax,10101b ; *заносим* 10101b *в регистр AX*

mov ax,11b ; *заносим* 11b *в регистр AX*

mov ax,21o ; *заносим* 21o *в регистр AX*

mov alfa, ax ; *помещаем содержимое регистра AX в alfa*

mov cx, ax *; копируем значение регистра AX в регистр CX*

mov ax, bx *; копируем значение регистра BX в регистр AX*

mov ax, cx  *; восстанавливаем предыдущее значение AX из регистра CX*

xchg ax, bx *; меняем местами значения регистров AX и BX*

mov si, 2 *; помещаем значение 2 в регистр SI (source index – индекс источника)*

mov alfa[si], ax *; помещаем значение регистра AX во второе слово массива alfa*

mov esi, 2 *; помещаем значение 2 в регистр ESI*

mov ebx, offset alfa *; помещаем смещение alfa в регистр EBX*

lea ebx, alfa *; помещаем адрес (смещение) alfa в регистр EBX*

mov ecx, [ebx][esi] *; помещаем значение 2-го слова массива alfa в регистр ECX*

mov cx, alfa[2] *; помещаем значение 2-го слова массива alfa в регистр CX*

mov cx, [alfa+2] *; помещаем значение 2-го слова массива alfa в регистр CX*

mov di, 4 *; помещаем значение 4 в регистр DI (destination index- индекс приёмника)*

mov byte ptr [ebx][edi], 55h *; записываем 55h в 4-й байт alfa*

mov esi, 2 *; помещаем значение 2 в регистр ESI*

mov ebx, 3 *; помещаем значение 3 в регистр EBX*

mov alfa[ebx][esi], 33h *; помещаем 33h во 2-й байт 3 слова массива alfa*

mov alfa[ebx+esi], 33h *; помещаем 33h во 2-й байт 3 слова массива alfa*

mov [alfa+ebx+esi], 33h *; помещаем 33h во 2-й байт 3 слова массива alfa*

mov [ebx][esi]+alfa, 33h *; помещаем 33h во 2-й байт 3 слова массива alfa*

exit *; вызов процедуры выхода из программы из файла Irvine32.inc*

main ENDP *; завершение процедуры main* , *весь код помещенный после этой директивы, будет игнорироваться ассемблером;*

END main *;завершение программы/ точка входа в программу*

**Листинг**

00000000 .data

00000000 00000003 [ alfa WORD 3 DUP(?)

0000

]

00000000 .code

00000000 main proc

00000000 66| B8 0011 mov ax,17

00000004 66| B8 0015 mov ax,10101b

00000008 66| B8 0003 mov ax,11b

0000000C 66| B8 0011 mov ax,21o

00000010 66| A3 mov alfa,ax

00000000 R

00000016 66| 8B C8 mov cx,ax

00000019 66| 8B C3 mov ax,bx

0000001C 66| 8B C1 mov ax,cx

0000001F 66| 93 xchg ax,bx

00000021 66| BE 0002 mov si,2

00000025 66| 89 04 35 mov alfa[si],ax

00000000 R

0000002D BE 00000002 mov esi,2

00000032 BB 00000000 R mov ebx,offset alfa

00000037 8D 1D 00000000 R lea ebx,alfa

0000003D 8B 0C 1E mov ecx,[ebx][esi]

00000040 66| 8B 0D mov cx,alfa[2]

00000002 R

00000047 66| 8B 0D mov cx,[alfa+2]

00000002 R

0000004E 66| BF 0004 mov di,4

00000052 C6 04 1F 55 mov byte ptr [ebx][edi],55h

00000056 BE 00000002 mov esi,2

0000005B BB 00000003 mov ebx,3

00000060 66| C7 84 1E mov alfa[ebx][esi],33h

00000000 R

0033

0000006A 66| C7 84 1E mov alfa[ebx+esi],33h

00000000 R

0033

00000074 66| C7 84 1E mov [alfa+ebx+esi],33h

00000000 R

0033

0000007E 66| C7 84 1E mov [ebx][esi]+alfa,33h

00000000 R

0033

exit

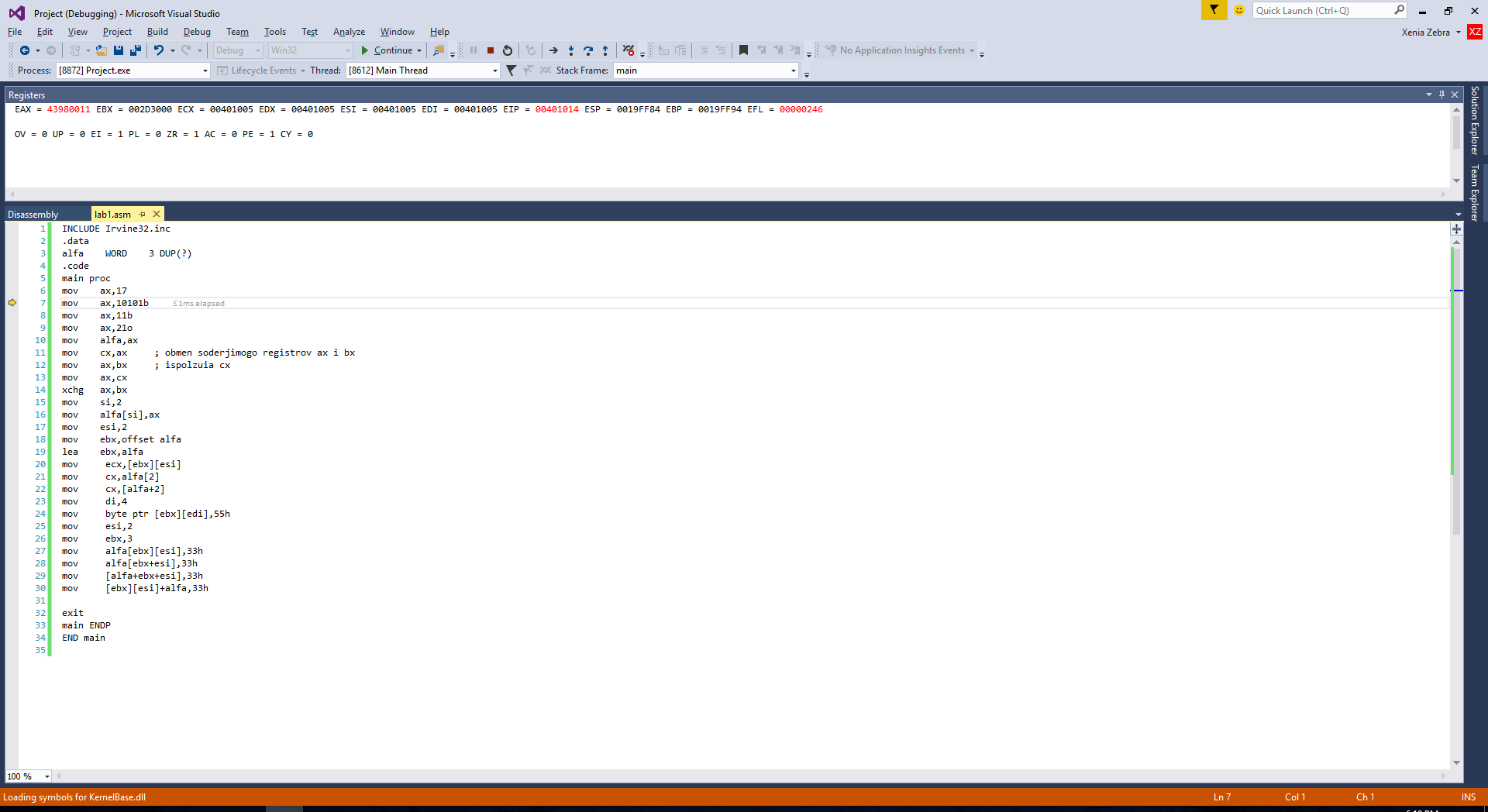
00000088 6A 00 \* push +000000000h

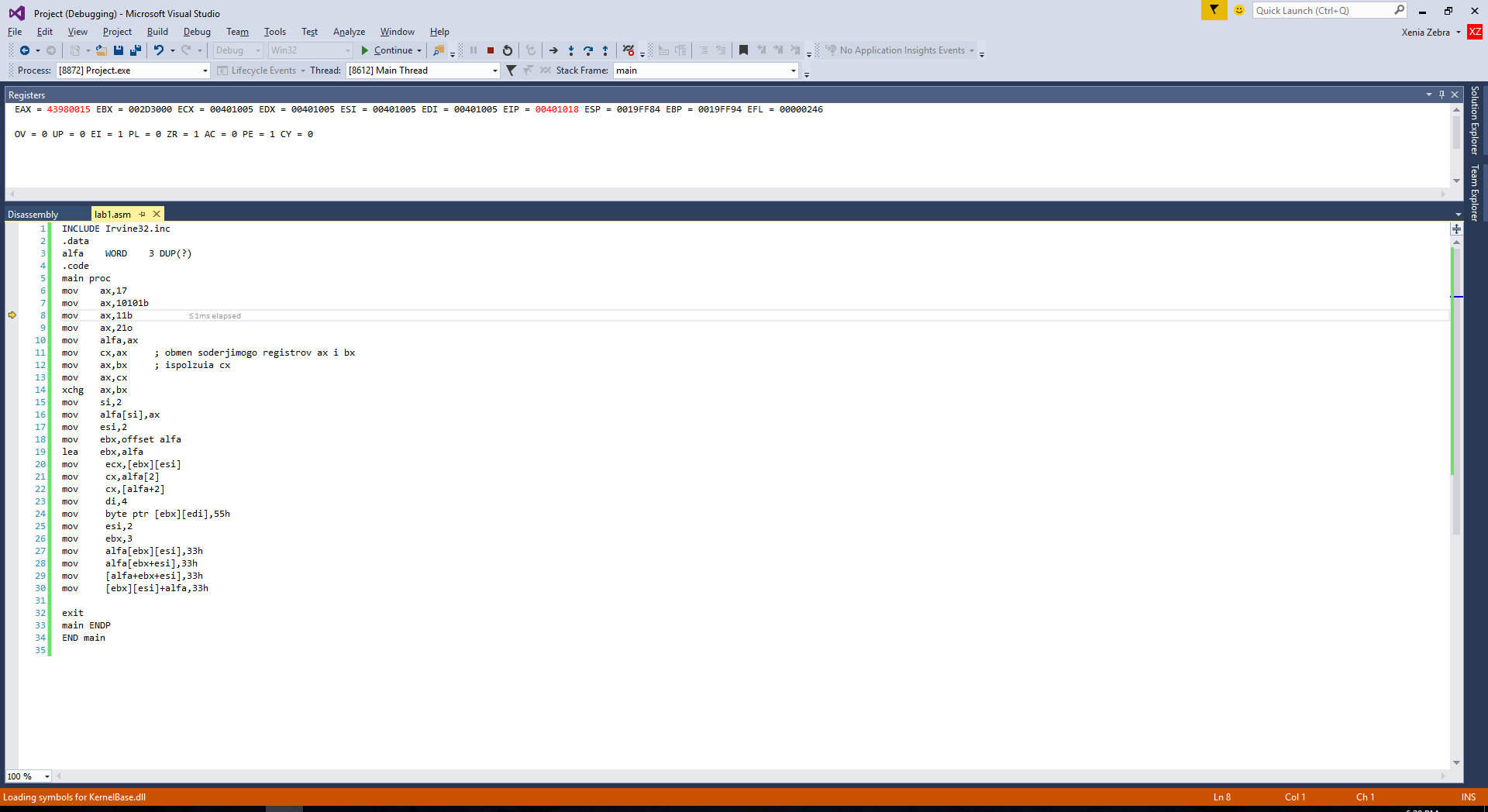
0000008A E8 00000000 E \* call ExitProcess

0000008F main ENDP

END main

**Дебаг**





1. **Вычислить арифметическое выражение: e=((a+b\*c-d)/f+g\*h)/i. Произвести ассемблирование и исполнить в пошаговом режиме в Debug.**

INCLUDE Irvine32.inc *; директива,которая подключает библиотеки irvine32.inc*

; vicisliti arifmeticeskoe virajenie: e=((a+b\*c-d)/f+g\*h)/i

; razreadnosti a, d, f – slovo b, c, g, h, i –byte

; для деления на f необходимо расширить делимое до двойного слова

; использовать только частное, разрядность результата-байт

.data *; директива указывает начало сегмента данных*

a dw 5 *;опеределяем слово со значением 5 и смещением a*

b db 6 *;опеределяем байт со значением 6 и смещением b*

cd db 10 *;опеределяем байт со значением 10 и смещением cd*

d dw 5 *;опеределяем слово со значением 5 и смещением d*

f dw 6 *;опеределяем ckjdj со значением 6 и смещением f*

g db 10 *;опеределяем байт со значением 10 и смещением g*

h db 11 *;опеределяем байт со значением 11 и смещением h*

i db 10 *;опеределяем байт со значением 10 и смещением i*

interm dw ? *;опеределяем слово с неопределённым значением и смещением interm*

rez db ? *;опеределяем байт с неопределённым значением и смещением rez*

.code *; директива указывает начало сегмента кода*

main proc *; процедура с именем main*

mov eax,0 *; заносим 0 в регистр EAX*

mov al, b *; заносим значение b в регистр AL*

imul cd *; в ax coхранится результат b\*c*

add ax, a *; ax=b\*c+a*

sub ax, d *; ax=b\*c+a-d*

cwd *; расширили слово из ax, в двойное слово в dx:ax*

idiv f *; частное в ax и остаток в dx, ax=(a+b\*c-d)/f*

mov interm, ax *; interm=(a+b\*c-d)/f*

mov al, g *; заносим значение g в регистр AL*

imul h *; ax=g\*h*

add ax, interm *; ax=(a+b\*c-d)/f+g\*h*

idiv i *; частное вAL и остаток в AH*

mov rez, al *; помещаем значение регистра AL в rez*

exit *; вызов процедуры выхода из программы из файла Irvine32.inc*

main ENDP *; завершение процедуры main* , *весь код помещенный после этой директивы, будет игнорироваться ассемблером;*

END main *;завершение программы/ точка входа в программу*

*; проверка результата ((a+b\*c-d)/f+g\*h)/i=((5+6\*10-5)/6+10\*11)/10= 12*

**Листинг**

00000000 .data

00000000 0005 a dw 5

00000002 06 b db 6

00000003 0A cd db 10

00000004 0005 d dw 5

00000006 0006 f dw 6

00000008 0A g db 10

00000009 0B h db 11

0000000A 0A i db 10

0000000B 0000 interm dw ?

0000000D 00 rez db ?

00000000 .code

00000000 main proc

00000000 B8 00000000 mov eax,0

00000005 A0 00000002 R mov al, b

0000000A F6 2D 00000003 R imul cd

00000010 66| 03 05 add ax, a ; ax=b\*c+a

00000000 R

00000017 66| 2B 05 sub ax, d ; ax=b\*c+a-d

00000004 R

0000001E 66| 99 cwd

00000020 66| F7 3D idiv f ; ax=(a+b\*c-d)/f

00000006 R

00000027 66| A3 mov interm, ax ; interm=(a+b\*c-d)/f

0000000B R

0000002D A0 00000008 R mov al, g

00000032 F6 2D 00000009 R imul h ; ax=g\*h

00000038 66| 03 05 add ax, interm ; ax=(a+b\*c-d)/f+g\*h

0000000B R

0000003F F6 3D 0000000A R idiv i

00000045 A2 0000000D R mov rez, al

exit

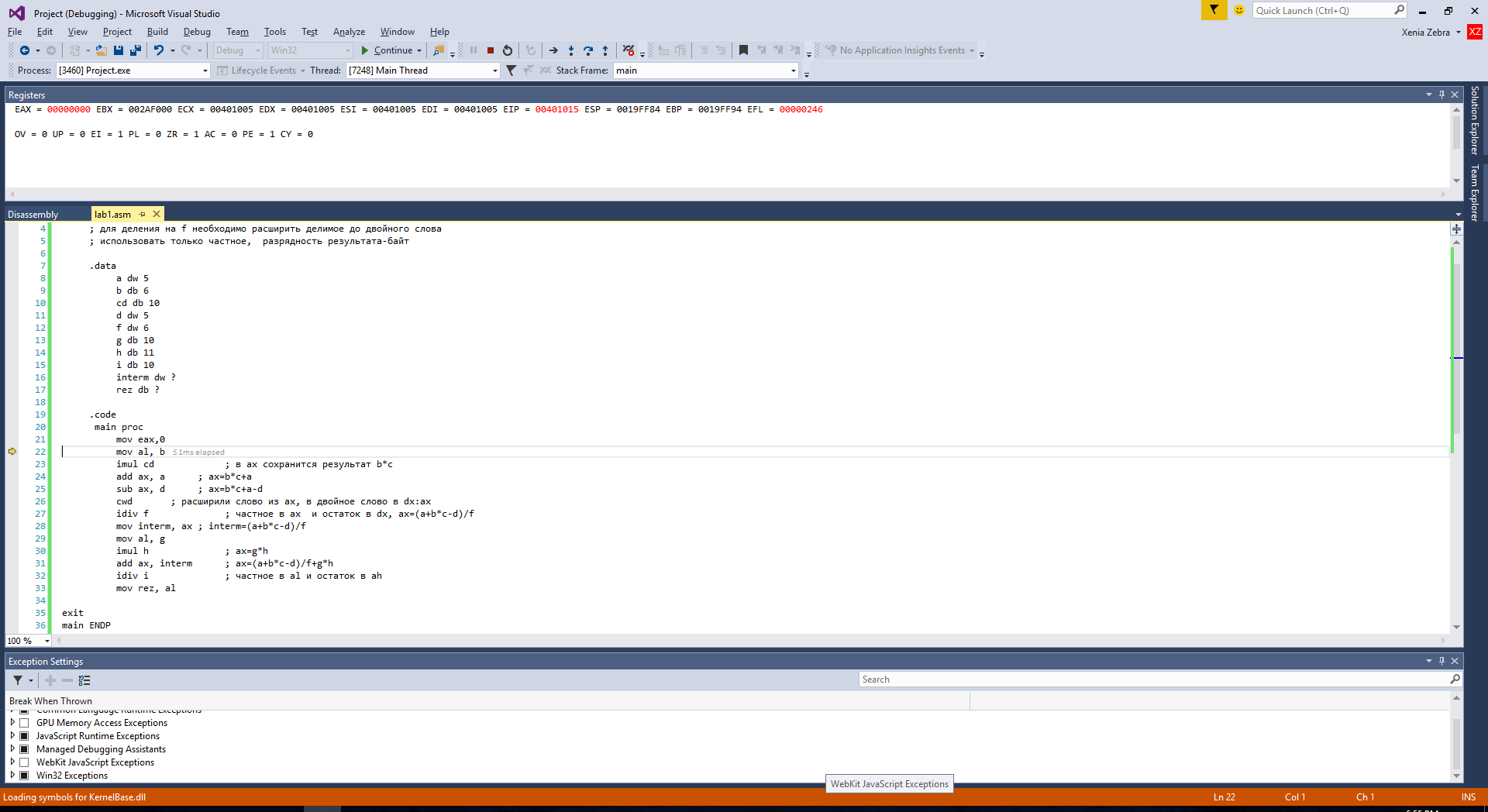
0000004A 6A 00 \* push +000000000h

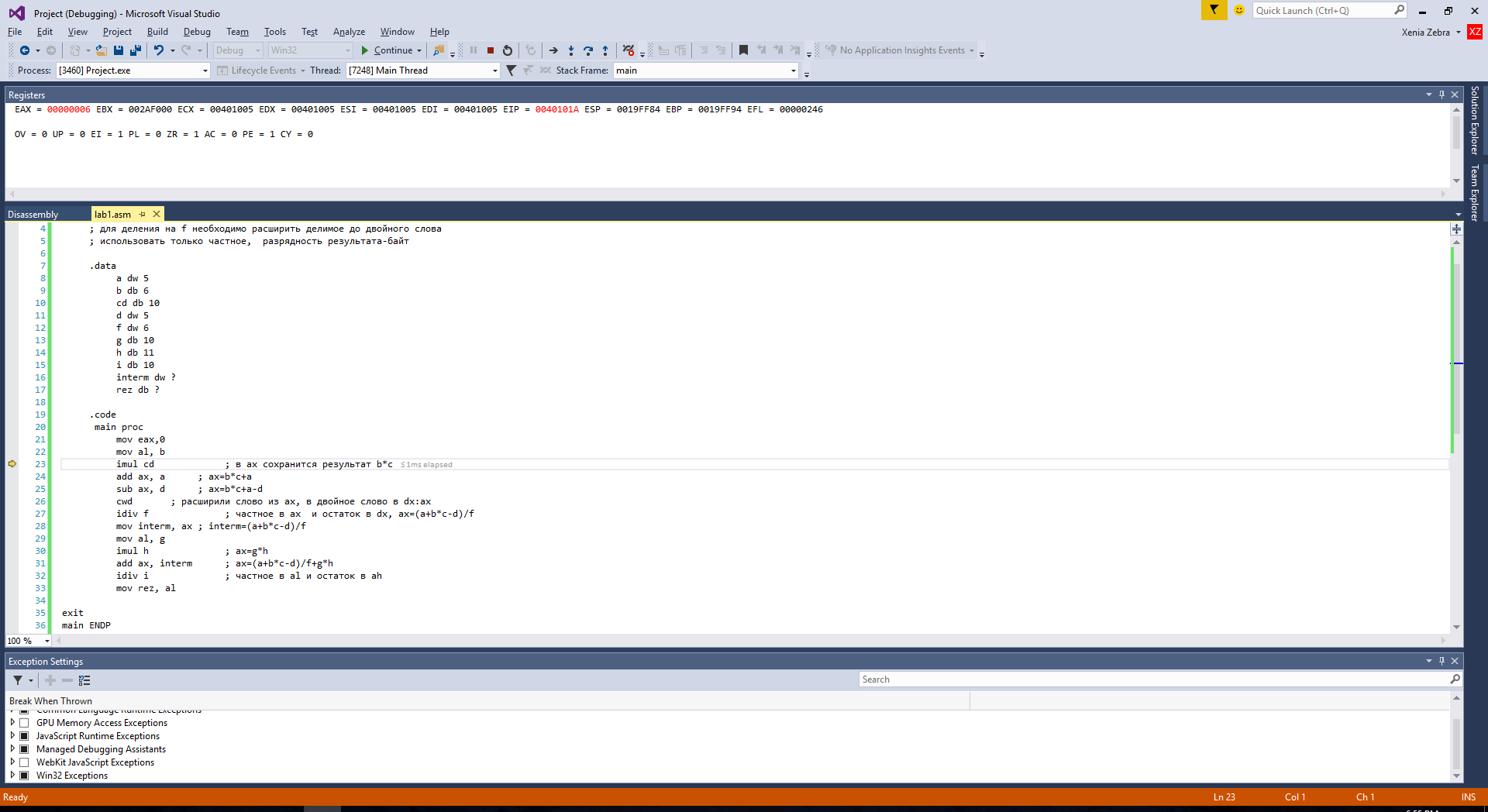
0000004C E8 00000000 E \* call ExitProcess

00000051 main ENDP

END main

**Дебаг**





1. **Вычислить арифметическое выражение:  *z=(2+1/a)/(3+1/(b\*b))-1/(c\*c)***

INCLUDE Irvine32.inc *; директива,которая подключает библиотеки irvine32.inc*

; vicisliti arifmeticeskoe virajenie: z=(2+1/a)/(3+1/(b\*b))-1/(c\*c)

.data *; директива указывает начало сегмента данных*

var\_a db 1 ;опеределяем байт со значением 1 и смещением var\_a

var\_b db 4 ;опеределяем байт со значением 4 и смещением var\_b

var\_c db 6 ;опеределяем байт со значением 6 и смещением var\_c

interm1 dw ? ;опеределяем слово с неопределённым значением и смещением interm1

interm2 dw ? ;опеределяем слово с неопределённым значением и смещением interm2

rez db ? ;опеределяем байт с неопределённым значением и смещением rez

.code *; директива указывает начало сегмента кода*

main proc *; процедура с именем main*

mov eax, 0 *; заносим 0 в регистр EAX*

mov edx, 0 *; заносим 0 в регистр EDX*

mov ax, 1 ; *заносим 1 в регистр AX*

mov bl, var\_a *; заносим значение var\_a в регистр BL*

div bl *; частное в регистр AL, остаток в регистр AH*

add ax, 2 *;(2+1/a), суммируем 2 и значение в регистре AX*

mov interm1, ax *;interm1 = 3, помещаем значение регистра AX в interm1*

mov edx, 0 *; заносим 0 в регистр EDX*

mov al, var\_b *; заносим значение var\_b в регистр AL*

mul var\_b *;в AX сохранится результат var\_b \* var\_b*

mov interm2, ax *;interm2 = 10, помещаем значение регистра AX в interm2*

mov ax, 1 ; *заносим 1 в регистр AX*

cwd *; расширили слово из ax в двойное слово dx:ax*

div interm2 *; частное в AX и остаток в DX*

add ax, 3 *;(3+1/(b\*b)), суммируем 3 и значение в регистре AX*

mov interm2, ax *;interm2 = 3, помещаем значение регистра AX в interm2*

mov edx, 0 *; заносим 0 в регистр EDX*

mov ax, interm1 *; помещаем значение interm1 в регистр AX*

cwd *; расширили слово из ax в двойное слово dx:ax*

idiv interm2 *; частное в AX и остаток в DX*

mov interm1, ax *;(2+1/a)/(3+1/(b\*b)), interm1 = 1, значение регистра AX в interm1*

mov edx, 0 *; заносим 0 в регистр EDX*

mov al, var\_c *; заносим значение var\_c в регистр AL*

mul var\_c ;в AX сохранится результат var\_c \* var\_c

mov interm2, ax ;interm2 = 24, помещаем значение регистра AX в interm3

mov ax, 1 ; *заносим 1 в регистр AX*

cwd *; расширили слово из ax в двойное слово dx:ax*

idiv interm2 *;1/(c\*c), частное в AX и остаток в DX*

sub interm1, ax *;ax = 0, от interm1 отнимаем значение регистра AX*

mov ax, interm1 *; помещаем interm1 в регистр AX*

mov rez, al *; помещаем значение регистра AL в rez*

exit *; вызов процедуры выхода из программы из файла Irvine32.inc*

main ENDP *; завершение процедуры main* , *весь код помещенный после этой директивы, будет игнорироваться ассемблером;*

END main *;завершение программы/ точка входа в программу*

**Листинг**

00000000 .data

00000000 01 var\_a db 1

00000001 04 var\_b db 4

00000002 06 var\_c db 6

00000003 0000 interm1 dw ?

00000005 0000 interm2 dw ?

00000007 00 rez db ?

00000000 .code

00000000 main proc

00000000 B8 00000000 mov eax, 0

00000005 BA 00000000 mov edx, 0

0000000A 66| B8 0001 mov ax, 1

0000000E 8A 1D 00000000 R mov bl, var\_a

00000014 F6 F3 div bl

00000016 66| 83 C0 02 add ax, 2 ;(2+1/a)

0000001A 66| A3 mov interm1, ax ;interm1 = 3

00000003 R

00000020 BA 00000000 mov edx, 0

00000025 A0 00000001 R mov al, var\_b

0000002A F6 25 00000001 R mul var\_b

00000030 66| A3 mov interm2, ax ;interm2 = 10

00000005 R

00000036 66| B8 0001 mov ax, 1

0000003A 66| 99 cwd

0000003C 66| F7 35 div interm2

00000005 R

00000043 66| 83 C0 03 add ax, 3 ;(3+1/(b\*b))

00000047 66| A3 mov interm2, ax ;interm2 = 3

00000005 R

0000004D BA 00000000 mov edx, 0

00000052 66| A1 mov ax, interm1

00000003 R

00000058 66| 99 cwd

0000005A 66| F7 3D idiv interm2

00000005 R

00000061 66| A3 mov interm1, ax ;(2+1/a)/(3+1/(b\*b)), interm1 = 1

00000003 R

00000067 BA 00000000 mov edx, 0 ; ??????? 0 ? ??????? EDX

0000006C A0 00000002 R mov al, var\_c

00000071 F6 25 00000002 R mul var\_c

00000077 66| A3 mov interm2, ax ;interm2 = 24

00000005 R

0000007D 66| B8 0001 mov ax, 1

00000081 66| 99 cwd

00000083 66| F7 3D idiv interm2 ;1/(c\*c)

00000005 R

0000008A 66| 29 05 sub interm1, ax ;ax = 0

00000003 R

00000091 66| A1 mov ax, interm1

00000003 R

00000097 A2 00000007 R mov rez, al

exit

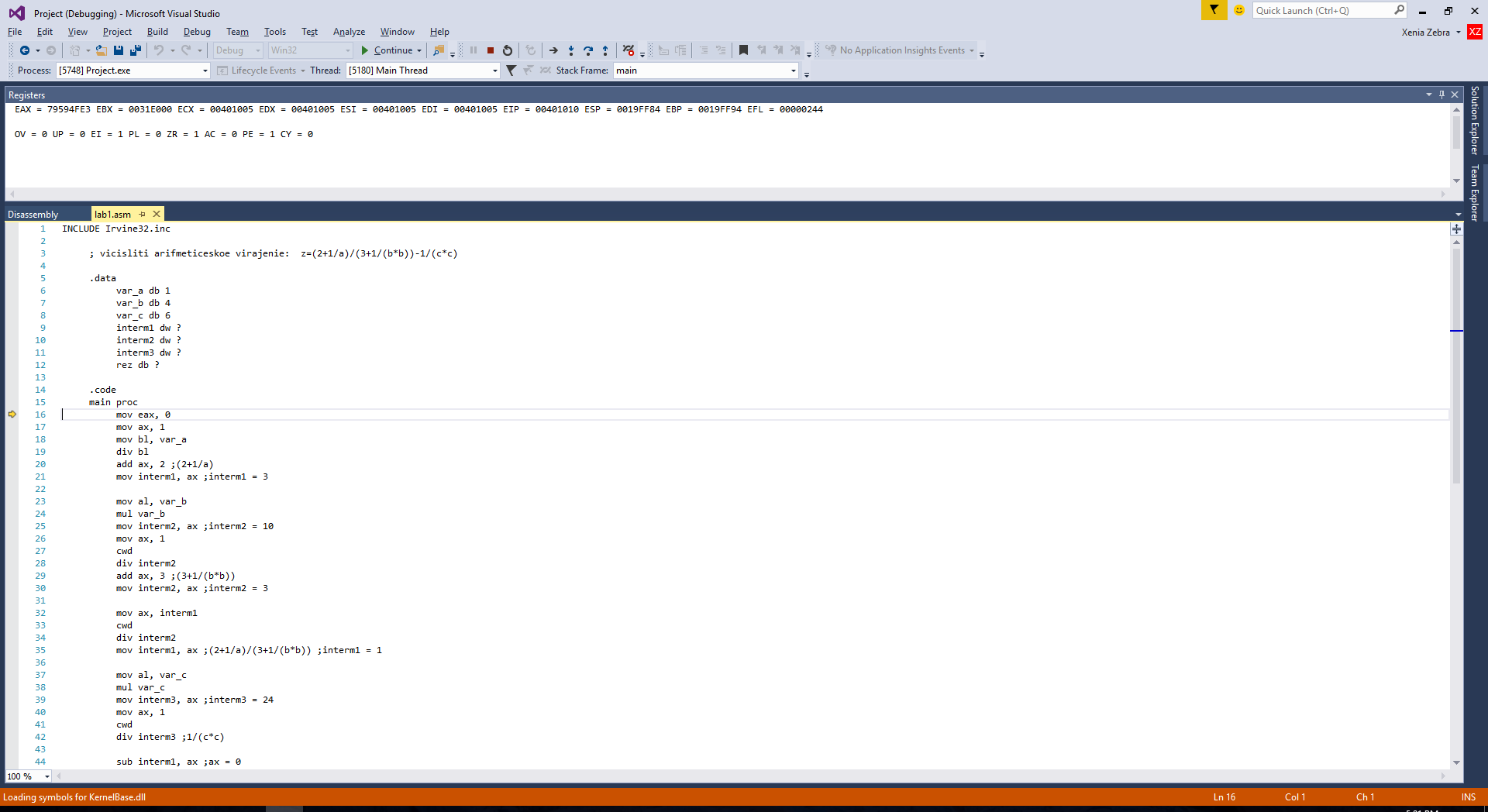
0000009C 6A 00 \* push +000000000h

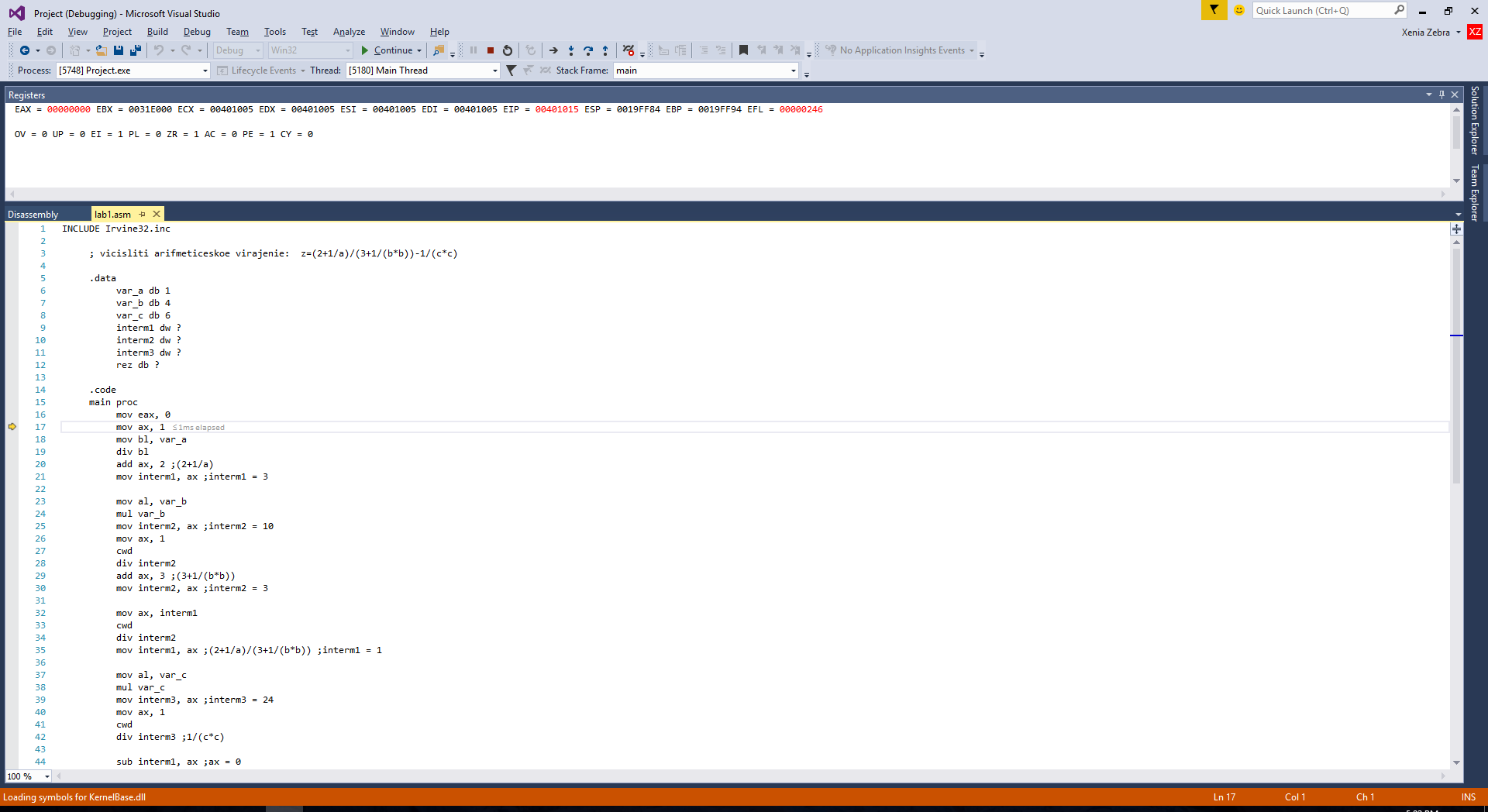
0000009E E8 00000000 E \* call ExitProcess

000000A3 main ENDP

END main

**Дебаг**





**Вывод:**

В ходе данной лабораторной работы были освоены и изучены основные типы данных микропроцессора, системы счисления, перевод чисел в различные системы счисления, директивы резервирования данных, директивы сегментации. На основе знаний, которые были получены, было осуществлено программа на языке ассемблера, выполняющее вычисление арифметического примера.