**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

**Лабораторна робота № 7**

з дисципліни «Системне програмування»

на тему

«Модульне програмування. Використання процедур»

Виконав: Перевірив:

студент групи ІМ-12 доцент Павлов В.Г.

Креславський Михайло Олегович

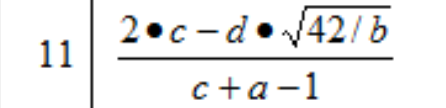
Номер у списку групи: 11

**Київ 2023**

**Мета роботи**

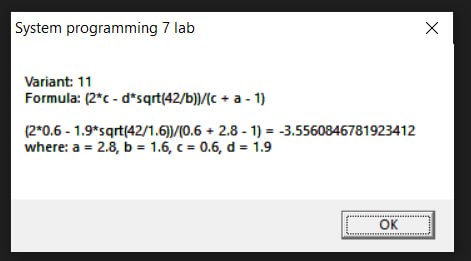
Вивчення прийомів модульного програмування, методів звернення до процедур і передачі в них параметрів.

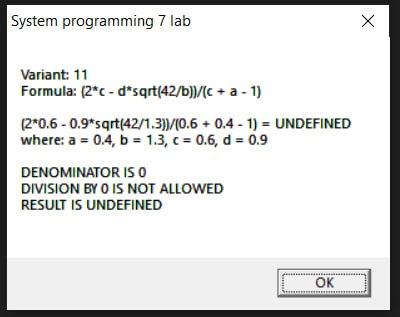
Формула:

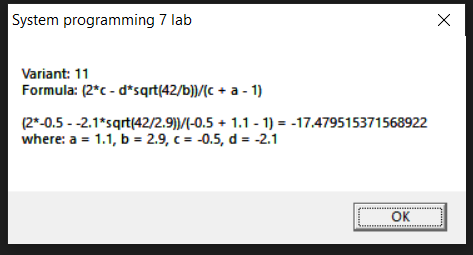


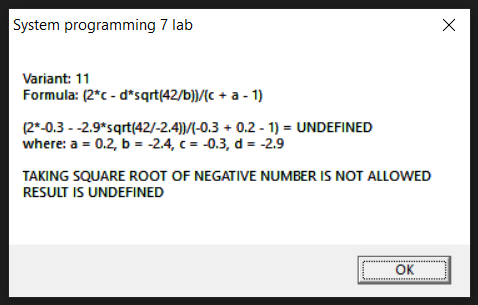
**Скріншоти виконання програми та контрольні розрахунки**

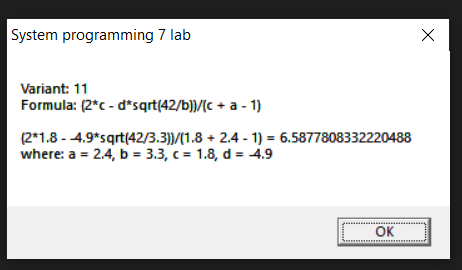
(зеленим підкреслено ті цифри, що співпадають, червоним – ті, що ні)

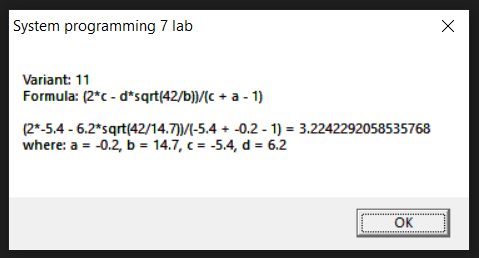
****

****

****

****

****

****

**Лістинг програми**

. .386

.model flat, stdcall

option casemap: none

include \masm32\include\masm32rt.inc

public arrayA, arrayC, one

extern calculateDenominator:proto

displayMessage macro windowMessage, windowTitle

invoke MessageBox, 0, offset windowMessage, offset windowTitle, 0

endm

.data

labTitle db "System programming 7 lab", 0

labMessageFormat db "Variant: 11", 10,

"Formula: (2\*c - d\*sqrt(42/b))/(c + a - 1)", 10, 10,

"(2\*%s - %s\*sqrt(42/%s))/(%s + %s - 1) = %s", 10,

"where: a = %s, b = %s, c = %s, d = %s", 10, 10, 0

errorZeroDenominatorFormat db "Variant: 11", 10,

"Formula: (2\*c - d\*sqrt(42/b))/(c + a - 1)", 10, 10,

"(2\*%s - %s\*sqrt(42/%s))/(%s + %s - 1) = UNDEFINED", 10,

"where: a = %s, b = %s, c = %s, d = %s", 10, 10,

"DENOMINATOR IS 0", 10,

"DIVISION BY 0 IS NOT ALLOWED", 10,

"RESULT IS UNDEFINED", 10, 10, 0

errorAreaOfDefinitionFormat db "Variant: 11", 10,

"Formula: (2\*c - d\*sqrt(42/b))/(c + a - 1)", 10, 10,

"(2\*%s - %s\*sqrt(42/%s))/(%s + %s - 1) = UNDEFINED", 10,

"where: a = %s, b = %s, c = %s, d = %s", 10, 10,

"TAKING SQUARE ROOT OF NEGATIVE NUMBER IS NOT ALLOWED", 10,

"RESULT IS UNDEFINED", 10, 10, 0

arrayA dq 2.8, 0.4, 1.1, 0.2, 2.4, -0.2

arrayB dq 1.6, 1.3, 2.9, -2.4, 3.3, 14.7

arrayC dq 0.6, 0.6, -0.5, -0.3, 1.8, -5.4

arrayD dq 1.9, 0.9, -2.1, -2.9, -4.9, 6.2

one dq 1.0

two dq 2.0

fourtyTwo dq 42.0

zero dq 0.0

.data?

result dq 1 dup(?)

numerator dt 16 dup(?)

denominator dt 16 dup(?)

final db 64 dup(?)

numberA db 32 dup(?)

numberB db 32 dup(?)

numberC db 32 dup(?)

numberD db 32 dup(?)

labMessage db 128 dup(?)

labMessageBuffer db 128 dup(?)

.code

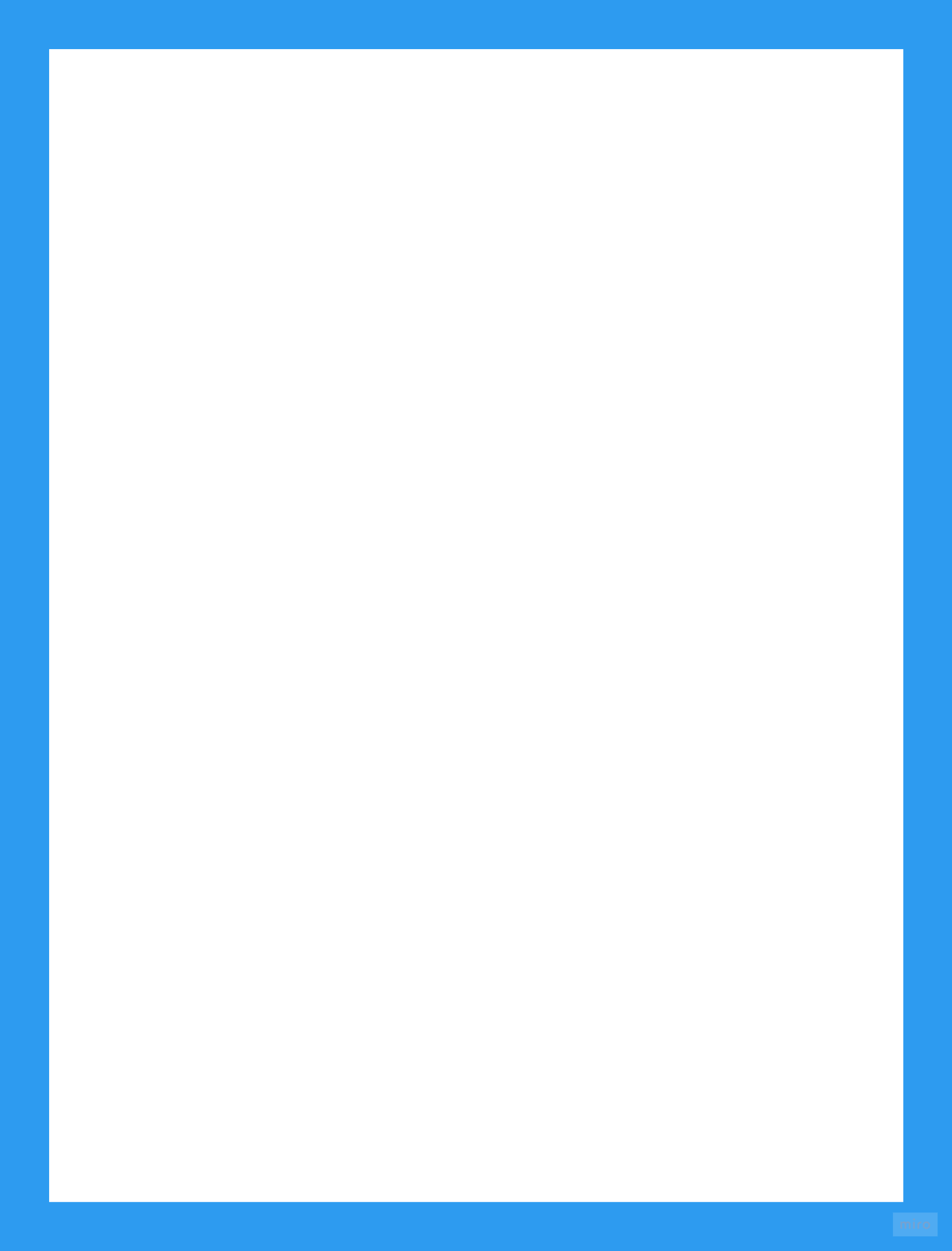
calculateFirstPart proc



fld qword ptr [eax + ebp \* 8]

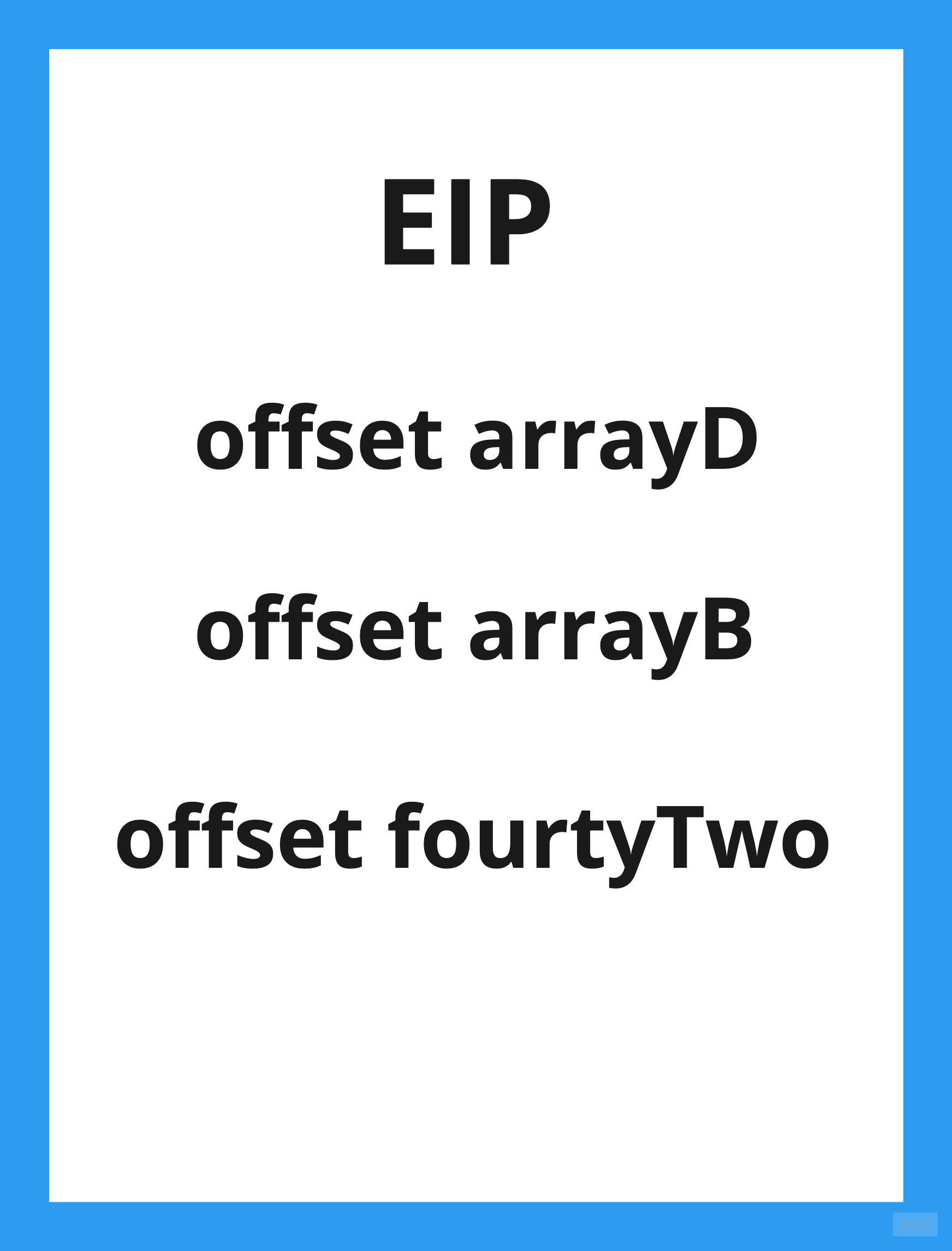
fmul qword ptr [ebx]

ret

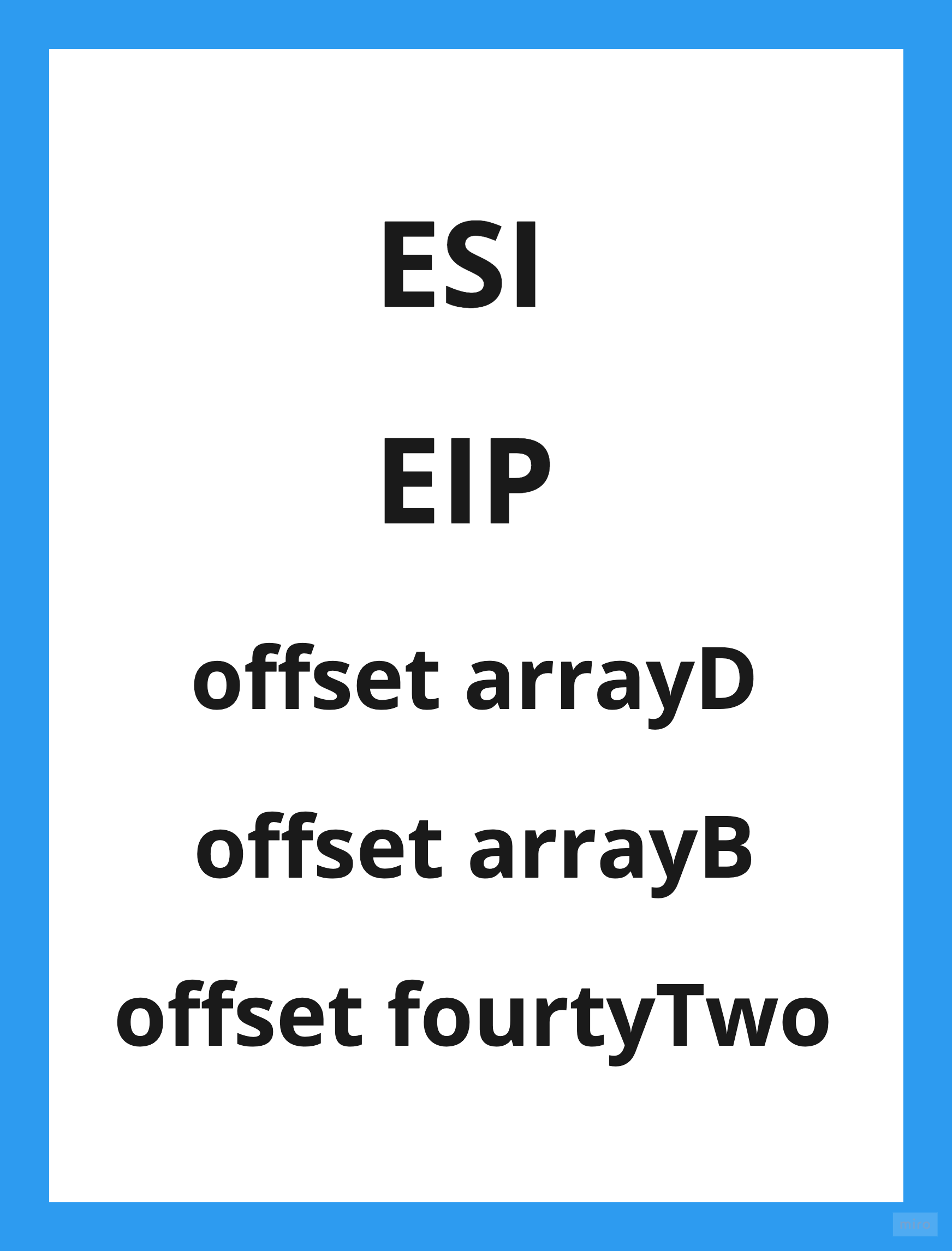


calculateFirstPart endp

calculateSecondPart proc



push esi



mov esi, esp

mov eax, [esi + 16]

mov ebx, [esi + 12]

mov edx, [esi + 8]

fld qword ptr [eax]

fdiv qword ptr [ebx + ebp \* 8]

fsqrt

ftst

fnstsw ax

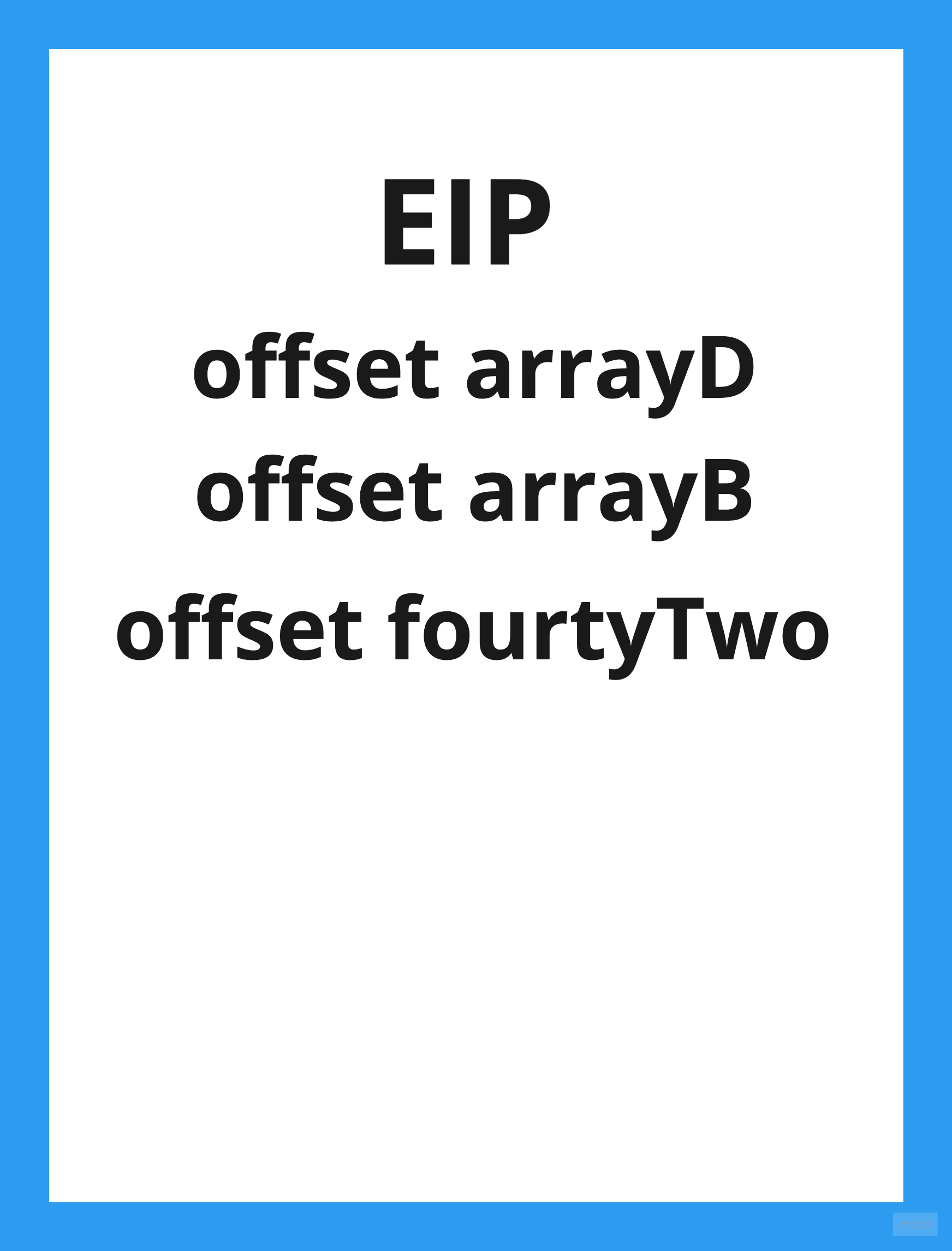
sahf

test ah, 01000000b

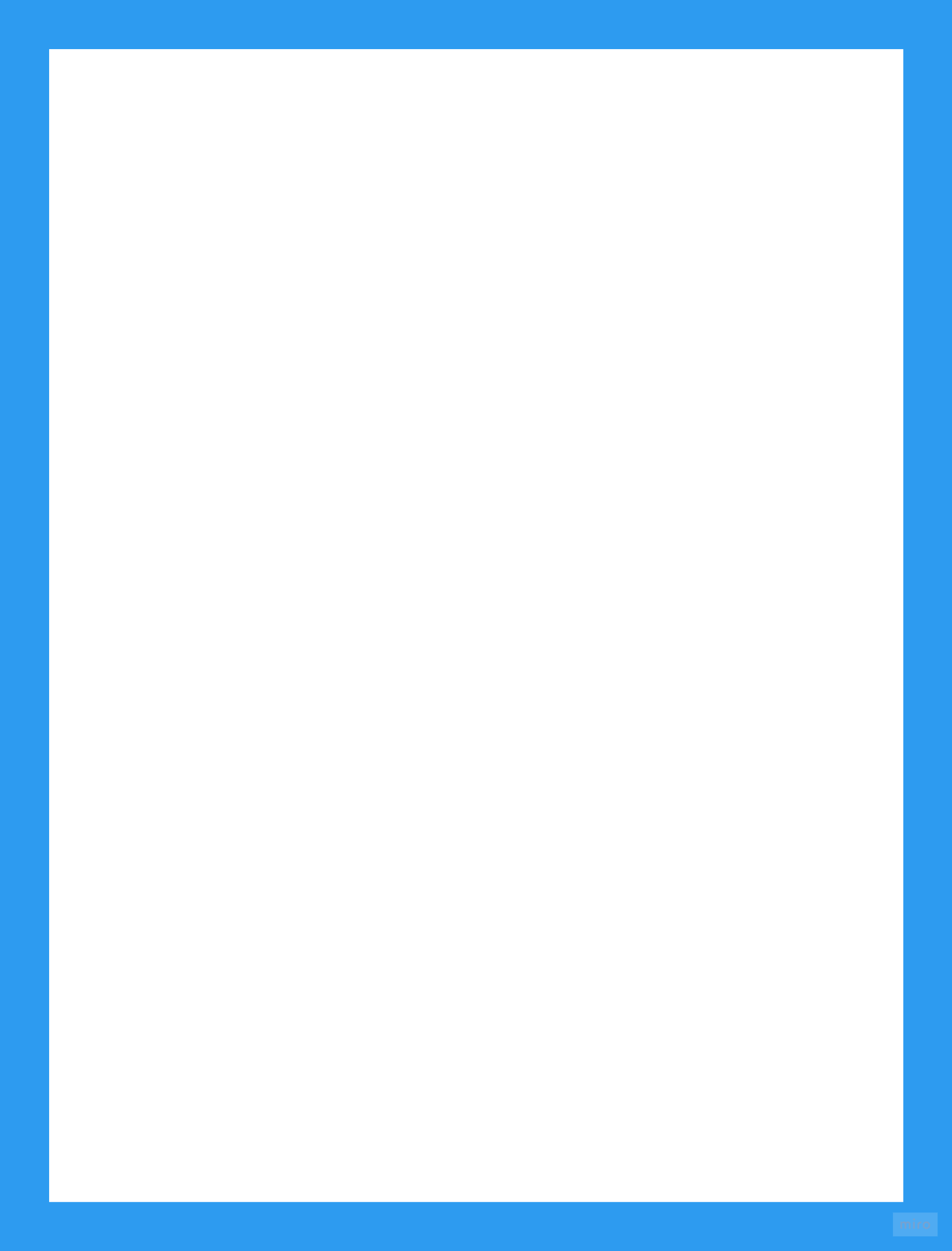
jnz errorAreaOfDefinitionError

fmul qword ptr [edx + ebp \* 8] ;; d\*sqrt(42/b)

pop esi



ret 12



calculateSecondPart endp

programSeventhLab:

mov ebp, 0

.while ebp < 6

invoke FloatToStr2, arrayA[ebp \* 8], addr numberA

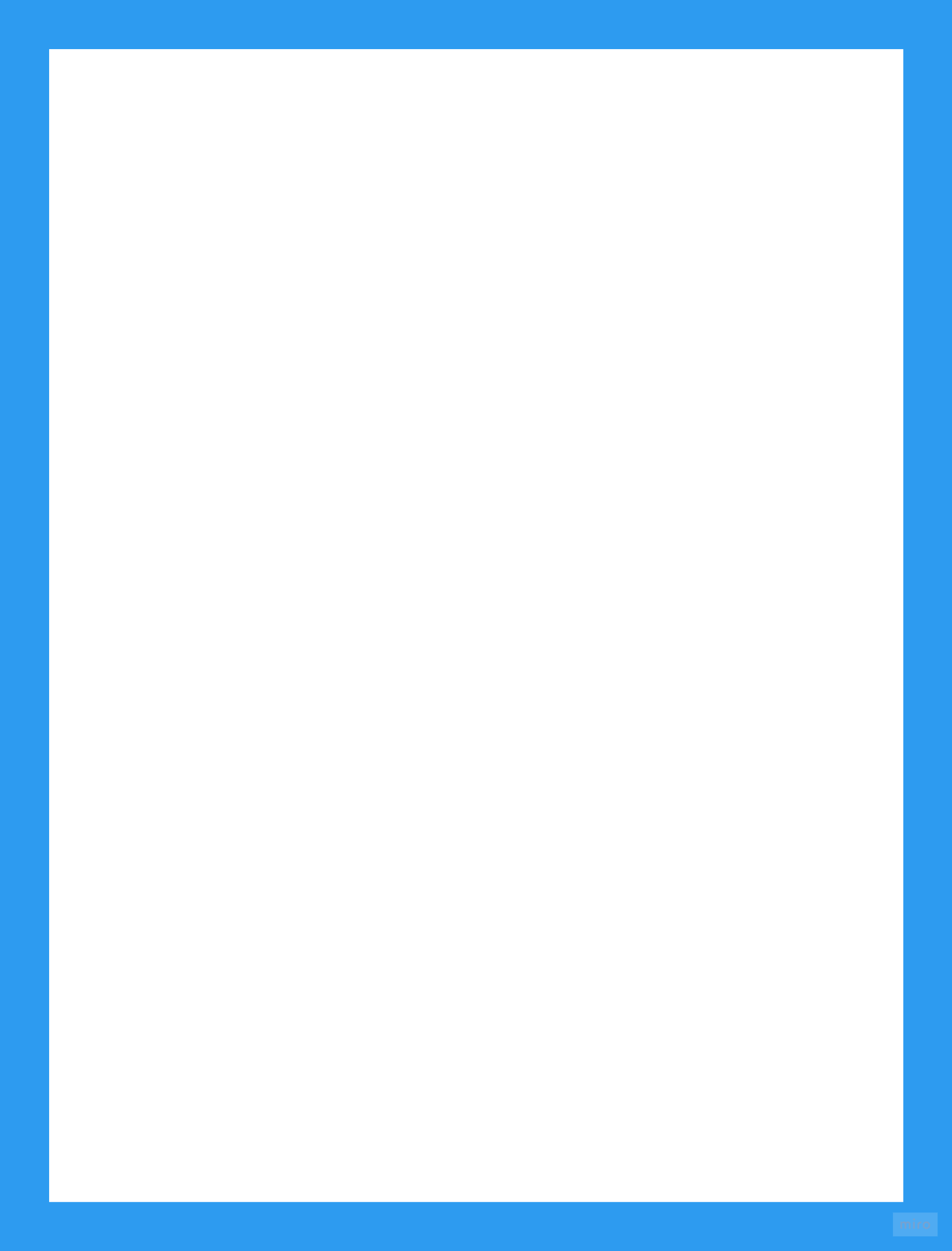
invoke FloatToStr2, arrayB[ebp \* 8], addr numberB

invoke FloatToStr2, arrayC[ebp \* 8], addr numberC

invoke FloatToStr2, arrayD[ebp \* 8], addr numberD

finit

call calculateDenominator

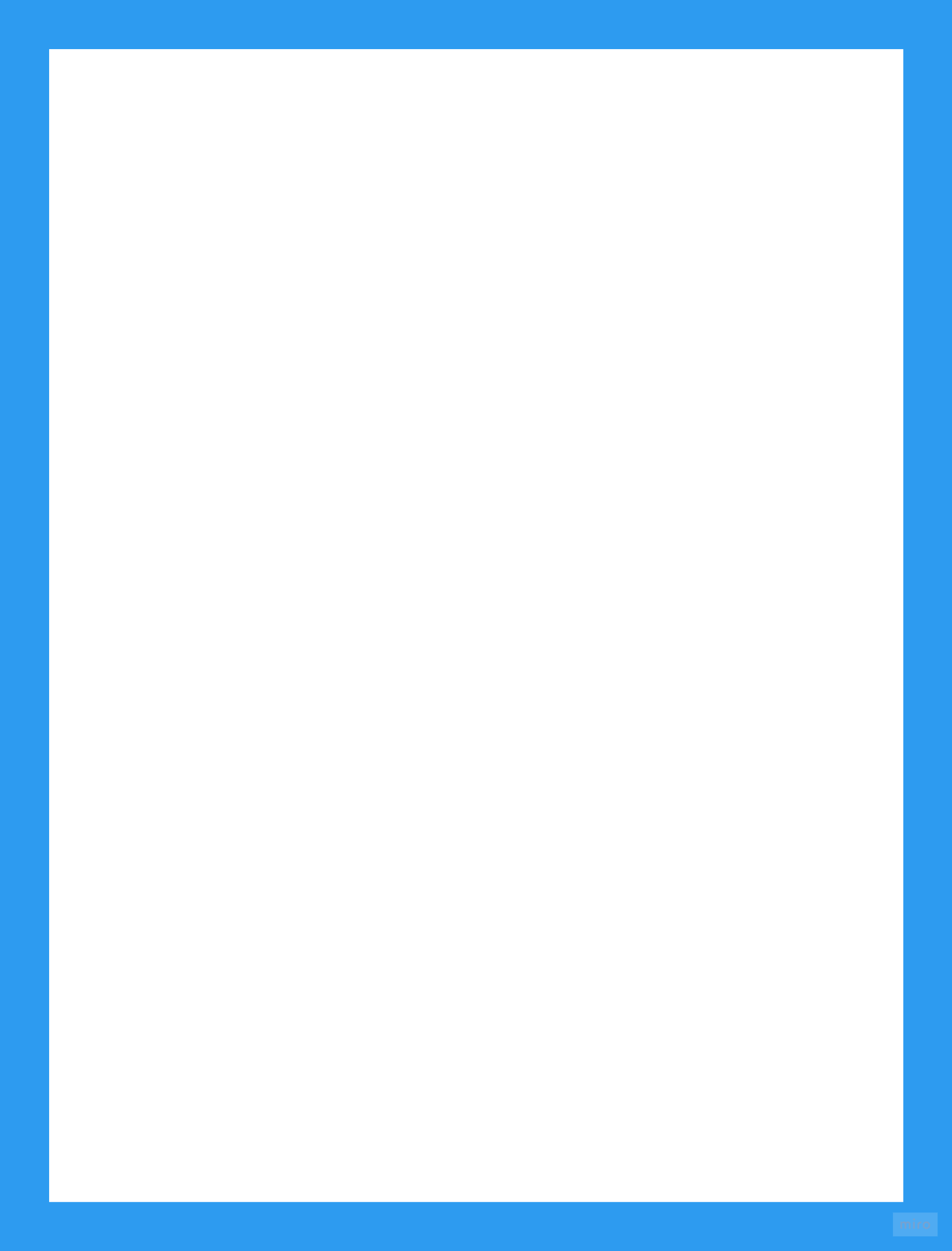


fstp denominator ;; save denominator

mov eax, offset arrayC

mov ebx, offset two

call calculateFirstPart



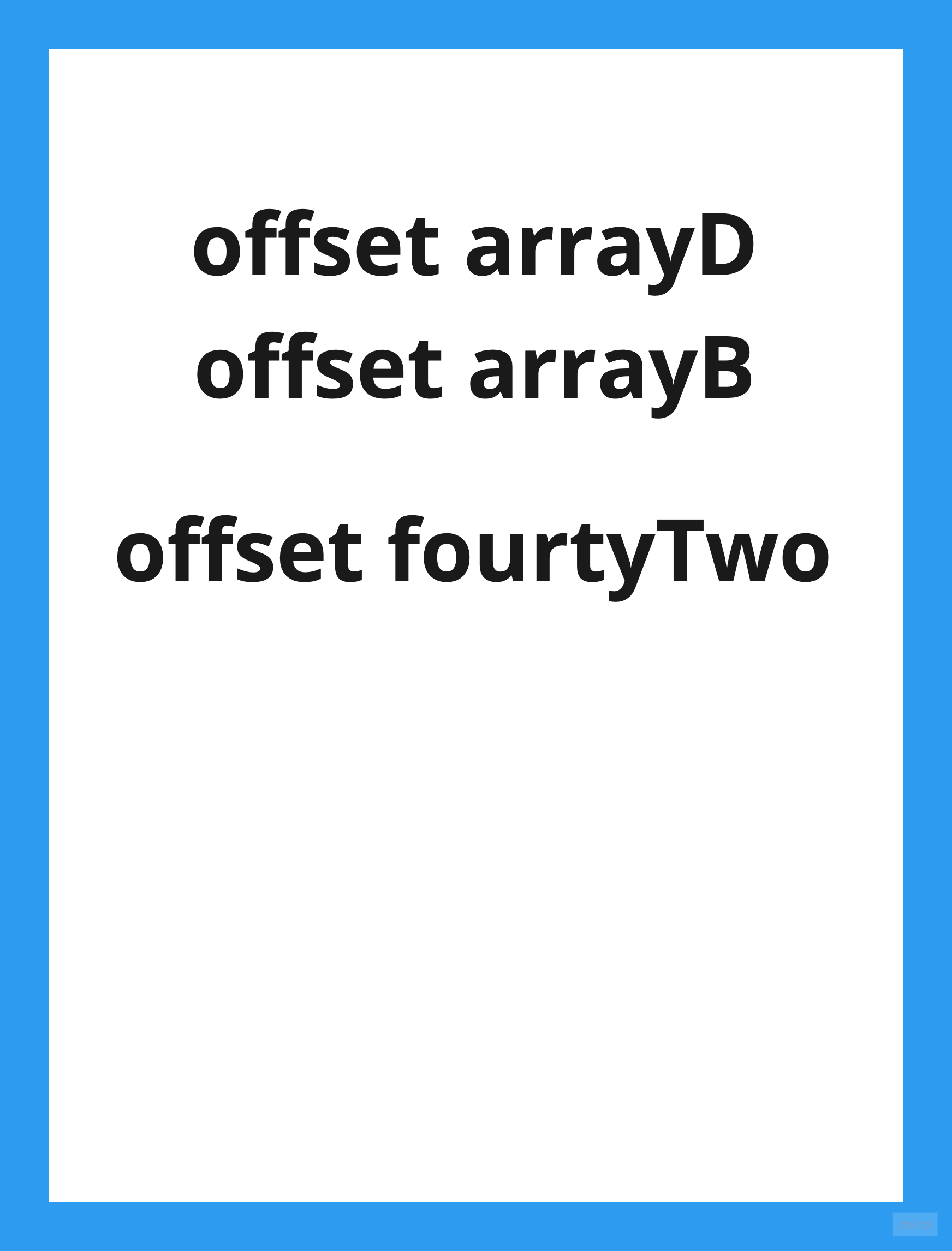
push offset fourtyTwo



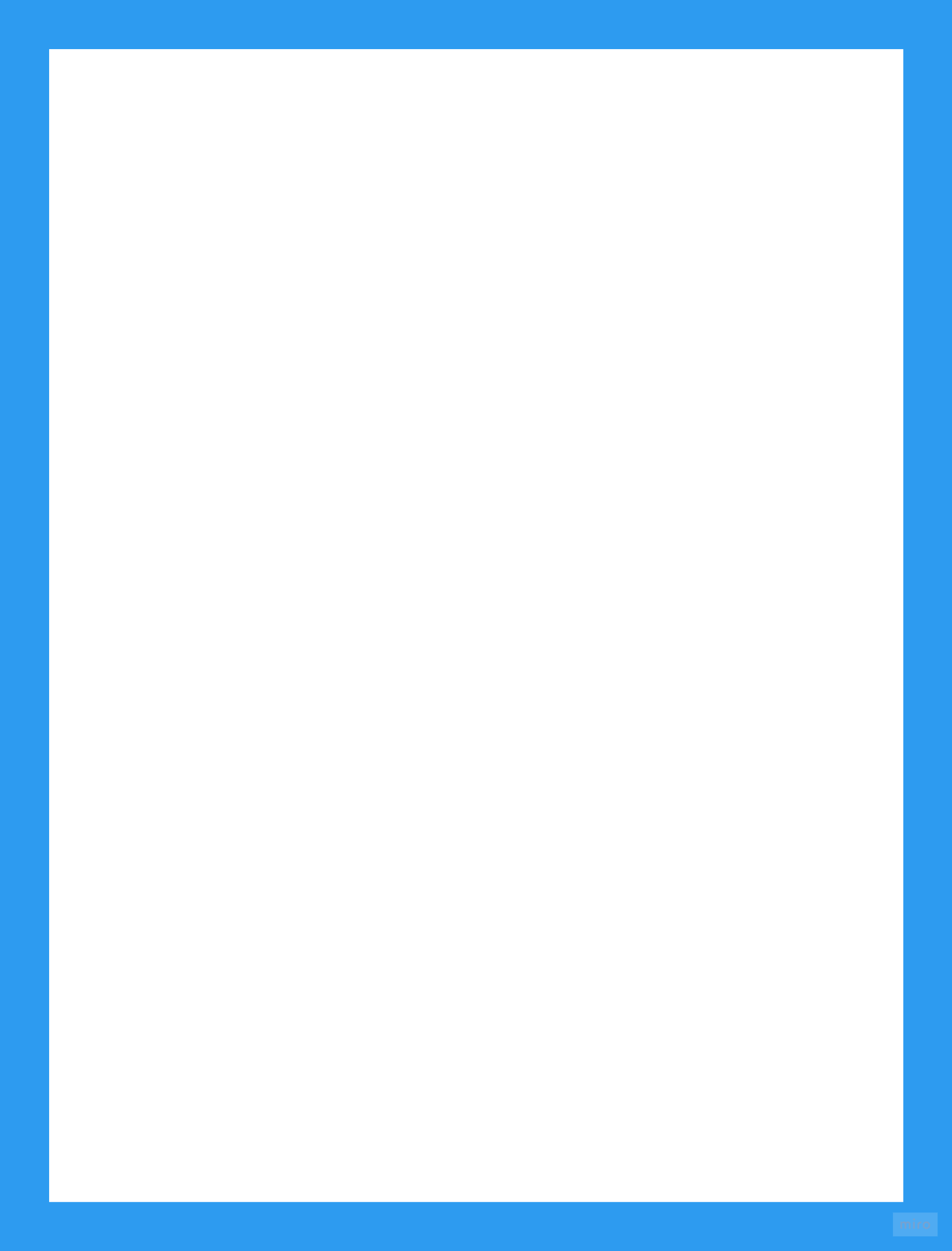
push offset arrayB



push offset arrayD



call calculateSecondPart



fxch

fsub st(0), st(1) ;; 2\*c - d\*sqrt(42/b)

fstp numerator ;; save numerator

fld tbyte ptr [numerator] ; st(1)

fld tbyte ptr [denominator] ; st(0)

ftst

fstsw ax

sahf

jz zeroDenominatorError

jnz noError

inc ebp

noError:

fdivp st(1), st(0)

fstp qword ptr [result]

invoke FloatToStr2, result, addr final

invoke wsprintf, addr labMessageBuffer, addr labMessageFormat,

addr numberC, addr numberD, addr numberB,

addr numberC, addr numberA,

addr final,

addr numberA, addr numberB, addr numberC, addr numberD

invoke szCatStr, addr labMessage, addr labMessageBuffer

displayMessage labMessage, labTitle

jmp nextTick

zeroDenominatorError:

invoke wsprintf, addr labMessageBuffer, addr errorZeroDenominatorFormat,

addr numberC, addr numberD, addr numberB,

addr numberC, addr numberA,

addr numberA, addr numberB, addr numberC, addr numberD

invoke szCatStr, addr labMessage, addr labMessageBuffer

displayMessage labMessage, labTitle

jmp nextTick

errorAreaOfDefinitionError:

invoke wsprintf, addr labMessageBuffer, addr errorAreaOfDefinitionFormat,

addr numberC, addr numberD, addr numberB,

addr numberC, addr numberA,

addr numberA, addr numberB, addr numberC, addr numberD

invoke szCatStr, addr labMessage, addr labMessageBuffer

displayMessage labMessage, labTitle

jmp nextTick

nextTick:

mov labMessage, 0h

inc ebp

.endw

invoke ExitProcess, 0

end programSeventhLab

**Лістинг файлу з процедурою**

.386

.model flat, stdcall

option casemap :none

public calculateDenominator

extern arrayA:qword, arrayC:qword, one:qword

.code

calculateDenominator proc

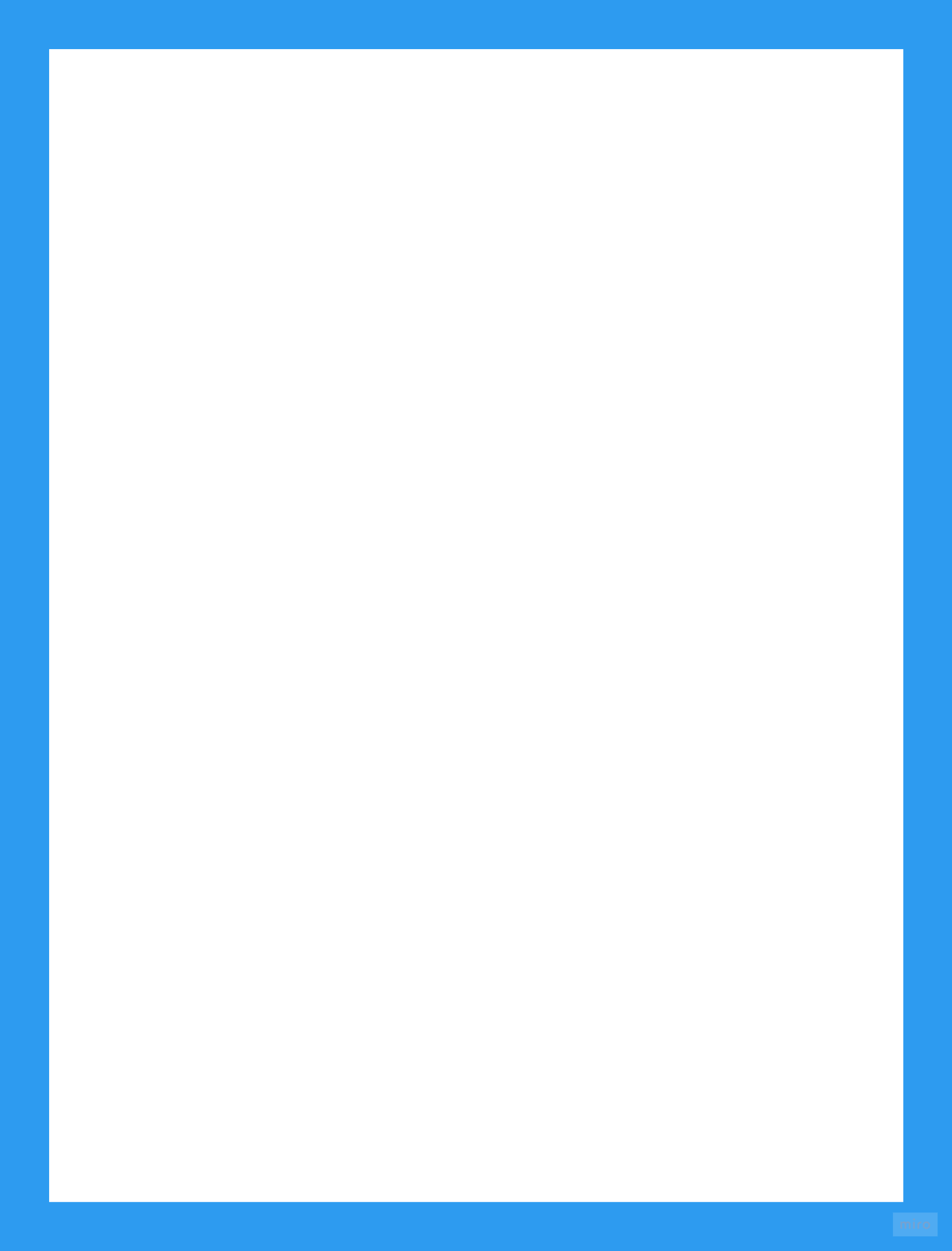


fld arrayC[ebp \* 8]

fadd arrayA[ebp \* 8]

fsub one

ret



calculateDenominator endp

end

**Висновок**

У цій лабораторній роботі я дослідив та практично використав різні методи передачі параметрів у процедуру у MASM32.

Використання процедур дозволяє економити простір програми і поліпшити її читабельність. Замість повторного копіювання однакового коду ми можемо просто викликати процедуру.

Також я використав зовнішні процедури з іншого файлу (EXTERN, PUBLIC), що ще більше розширило можливості повторного використання коду. За допомогою цього підходу можна написати процедуру в одному файлі і використовувати її в багатьох інших, що дозволяє ще більше зекономити простору та підвищити читабельність коду.

Я також вивчав роботу стеку під час використання процедур і зрозумів, що при передачі параметрів через стек ми зберігаємо адреси параметрів у стеці та звертаємося до них за допомогою базового вказівника, який ми зберігаємо у стеці на початку використання процедури і прирівнюємо його до вказівника стеку (esp). Також я виявив, що дані зберігаються в стеці у комірках по 4 байта.