**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 № 5**

з дисципліни

«Системне програмування»

на тему

“Арифметичні і логічні операції з цілими числами. Масиви.”

Виконав:

Перевірив:

студент групи ІП-93 доцент

Марченко Максим Олегович Павлов В.Г.

номер залікової книжки: 9316

Варіант 13

Київ 2020

Мета роботи : Вивчення арифметичних і логічних команд асемблера і здобуття навичок виконання розрахунків з елементами масивів.

Арифметичні операції в асемблері:

1. Додавання

Дія додавання виконується командою add <arg1>, <arg2>. Результат дії записується в <arg1>

1. Віднімання

Дія віднімання виконується командою sub <arg1>, <arg2>. Результат дії записується в <arg1>

1. Множення

Дія множення виконується командами:

* mul <arg> виконує множення <arg> та того, що знаходиться в регістрі EAX. Результат записується: перші біти - в EDX, останні в ЕАХ
* imul <arg1>, <arg2> виконує множення заданих двох аргументів, результат записує в <arg1>

1. Ділення

Дія ділення виконується командою div <arg> і виконує операцію ділення числа, перші біти якого - в EDX, останні в ЕАХ на arg. Результат цілочисельного ділення записується в EAX, остача в EDX.

Для того, щоб для дії ділення було можливо записати ділене належним форматом, використовується команда cdq, що форматує число, записане в еах до належного стану.

Також, в асмеблері існують команди для додання та віднімання одиниці до аргументу (inc <arg>, dec <arg>)

Логічні команди в асемблері:

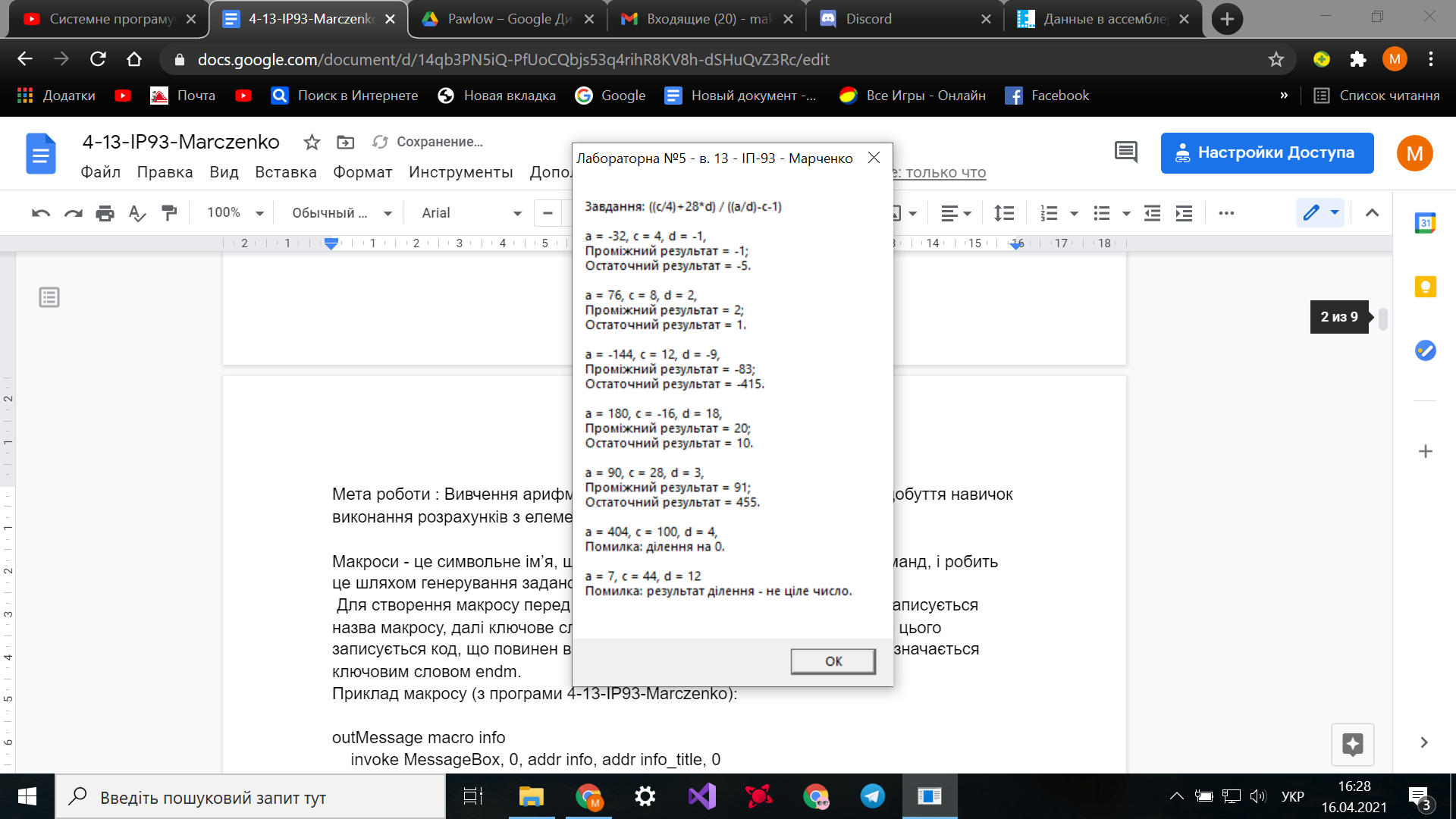
AND, OR, XOR, NOT - побітово виконують відповідні логічні умови, і записують в перший аргумент

TEST - виконує аналогічні операції до команди AND, проте знайдене значення не записується до першого аргументу, натомість стаючи його прапорцем.

Виконання дій над елементами масивів

Найкращий спосіб змінювати кожний елемент масиву окремо - це створити цикл на кількість ітерацій, рівною розміру циклу. Для того, щоб звернутися до і-тового елементу масиву потрібно застосувати запис <назва масиву> [<розмірність елементів в байтаx> \* i]. Таке звернення потрібне, тому що на відміну від мов високого рівня програмування, в асемблері звернення відбувається не до елементу від початку масиву, а до байту від початку масиву, тобто адреси змінної. Для присвоєння значень з та у масив, використовується команда mov.

Скріншот виконання програми



На скріншоті зображено виконання програми з 7 наборами вхідних даних, 5 з яких - коректні, 1 - спричинює ділення на 0, 1 - спричинює отримання нецілого результату ділення.

Для знаходження помилки ділення на 0, перед кожною операцією ділення знаменник порівнюється з 0, і, якщо вони рівні, то подальші обчислення не проводяться, а програма виводить повідомлення про помилку. Така перевірка необхідна при виконанні операції a/d (d == 0) та (c/4 + 28d) / (a/d - c - 1) ( a/d - c - 1 == 0)

Для знаходження помилки ділення з остачею, проводиться контрольне множення частки на дільник. Якщо знайдене число не рівне початковому діленому, то програма виводить повідомлення про помилку. Дана перевірка здійснюється тричі: a/d (а%d != 0), с/4 (с%4 != 0) та (c/4 + 28d) / (a/d - c - 1) ( c/4 + 28d % a/d - c - 1 != 0).

Контрольні розрахунки

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| А | С | D | А/D | A/D-C-1 | C/4 | 28\*D | C/4+28\*D | C/4+28\*D/(A/D-C-1) | Фінальні рез. |
| -32 | 4 | -1 | 32 | 27 | 1 | -28 | -27 | -1 | -5 |
| 76 | 8 | 2 | 38 | 29 | 2 | 56 | 58 | 2 | 1 |
| -144 | 12 | -9 | 16 | 3 | 3 | -252 | -249 | -83 | -415 |
| 180 | -16 | 18 | 10 | 25 | -4 | 504 | -500 | 20 | 10 |
| 90 | 28 | 3 | 30 | 1 | 7 | 84 | 91 | 91 | 455 |
| 404 | 100 | 4 | 101 | 0 | Помилка: ділення на 0 | | | | |
| 7 | 44 | 12 | 0.58 | Помилка : результат ділення - не ціле число | | | | | |

Дії для контрольних розрахунків проведені в тій ж послідовності, що й в програмі.

Як видно, результат виконання програми програми та контрольні розрахунки збігаються для всіх вхідних даних, включаючи випадки помилок, що підтверджує правильність програми.

Висновки

Під час виконання лабораторної роботи я розібрався з принципом роботи арифметичних і логічних команд в мові асемблер, а також з роботою з елементами масивів.Команди додавання та віднімання записуються схожим чином: ключове слово команди (add/sub), далі через кому два аргументи - члени відповідної дії. Результат дії записується в перший аргумент. У програмі я використовував версію множення, що записується і виконується схожим чином, та використовує ключове слово mul. Для виконання ділення, необхідно спочатку записати ділене в регістр EAX, далі виконати команду cdq, що перезаписує перші біти цього числа до регістру EDX, далі виконати команду div, що приймає єдиний аргумент - дільник. Результат ділення записується в регістр EAX, а остача в EDX. Також, в асемблері наявні логічні команди, що працюють за схожим принципом з додаванням та відніманням, тобто записують свій результат до першого аргументу. У програмі використовувалася робота з масивами, зокрема, щоб зчитати та записати необхідні дані. Ці дії виконуються командою mov, у котру, відповідно другим та першим елементом записується вказівник на адресу даного елемента.

Додаток: Код програми 5-13-IP93-Marczenko

.686

.model flat, stdcall

option casemap:none

include <\masm32\include\windows.inc>

include <\masm32\include\dialogs.inc>

include <\masm32\include\user32.inc>

include <\masm32\include\kernel32.inc>

include <\masm32\include\masm32.inc>

includelib <\masm32\lib\user32.lib>

includelib <\masm32\lib\kernel32.lib>

includelib <\masm32\lib\masm32.lib>

.data

dividor dd 4

inputA dd -32, 76, -144, 180, 90, 82, 7

inputC dd 4, 8, 12, -16, 28, 40, 4

inputD dd -1, 2, -9, 18, 3, 2, 12

first\_result dd 5 dup(?)

output\_result dd 5 dup(?)

buffer db 86 dup(?)

output db 512 dup(?)

function0 db 'Завдання: ((c/4)+(28\*d)) / ((a/d)-c-1)',13,13,0

out\_results db "a = %d, c = %d, d = %d,",13,"Проміжний результат = %d;",13,"Остаточний результат = %d",13,13,0

out\_error\_0 db "a = %d, c = %d, d = %d",13,"Помилка: ділення на 0.",13,13,0

out\_error\_remainder db "a = %d, c = %d, d = %d",13,"Помилка: результат ділення - не ціле число.",13,13,0

header db 'Лабораторна №5 - в. 13 - ІП-93 - Марченко', 0

.code

start:

mov edi, 0

invoke wsprintf, addr output, addr function0

.WHILE edi <7

mov eax, inputA[4\*edi]

mov ebx, inputD[4\*edi]

mov ecx, inputC[4\*edi]

;; a/d

cmp ebx, 0

je error\_0 ;; d == 0

cdq

idiv ebx

mov edx,eax

imul edx, ebx

cmp edx, inputA[4\*edi]

jne error\_remainder ;; a%d != 0

;; a/d - c - 1

sub eax, ecx

dec eax

cmp eax, 0

je error\_0 ;; a/d - c - 1 == 0

mov esi, eax

;; c/4

mov eax, ecx

cdq

idiv dividor

mov edx, eax

imul edx, dividor

cmp edx, ecx

jne error\_remainder ;; c%4 != 0

;; 28\*d

imul ebx, 28

;; c/4 + 28d

add eax, ebx

mov ecx, eax

cdq

;; (c/4 + 28d) / (a/d - c - 1)

idiv esi

mov edx, eax

imul edx, esi

cmp edx, ecx

jne error\_remainder ;; (c/4 + 28d) % (a/d - c - 1) != 0

mov first\_result[4\*edi], eax

test eax, 1

jnz odd

mov esi, 2

cdq

idiv esi

jmp outif

odd:

imul eax, 5

outif:

mov output\_result[4\*edi], eax

invoke wsprintf, addr buffer, addr out\_results, inputA[4\*edi], inputC[4\*edi], inputD[4\*edi], first\_result[4\*edi], output\_result[4\*edi]

jmp cont

error\_0:

invoke wsprintf, addr buffer, addr out\_error\_0, inputA[4\*edi], inputC[4\*edi], inputD[4\*edi]

jmp cont

error\_remainder:

invoke wsprintf, addr buffer, addr out\_error\_remainder, inputA[4\*edi], inputC[4\*edi], inputD[4\*edi]

jmp cont

cont:

invoke szCatStr, addr output, addr buffer

inc edi

.ENDW

invoke MessageBox, 0, addr output, addr header, 0

end start