Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Конструирование программ»

**ОТЧЕТ**

к лабораторной работе №5

на тему:

**«**Целочисленные арифметические операции. Обработка массивов числовых данных**»**

БГУИР 1-40 04 01

|  |
| --- |
| Выполнил студент группы 253502  АЛЬХОВИК Данила Игоревич |
|  |
| (дата, подпись студента) |
| Проверил старший преподаватель кафедры проектирования информационно-компьютерных систем  ЯЧИН Николай Сергеевич |
|  |
| (дата, подпись преподавателя) |

Минск 2023

**Цель работы:** Вариант 2. Выполнить набор арифметических операций над двумя целыми числами, представленными в 10-ной системе счисления.

**Ход работы:** на рисунке 1 представлены изначальные значения регистров, на рисунке 2 – ход выполнения задачи, на рисунке 3 - соответствующие значения после выполнения программы.

Листинг 1 – Исходный код программы задания 1

section .data

EnterMessege db "Enter a: ", 0

EnterMessege2 db "Enter b: ", 0

resultFormat db "Result: %d", 10, 0

overflowMessage db "Overflow!", 10, 0

divisionByZeroMessage db "Division by zero!", 10, 0

formatInput db "%d", 0

section .bss

a resq 1

b resq 1

section .text

global \_start

extern printf, scanf

checkOverflow:

pushf ; Сохранить флаги в стеке

pop rax ; Загрузить флаги в регистр RAX

test rax, 0x800 ; Проверить флаг переполнения (OF) - бит 11

jo overflow ; Если флаг установлен, то перейти к обработчику переполнения

ret

overflow:

mov edi, overflowMessage ; Вывести сообщение о переполнении

mov eax, 0

call printf

jmp exit ; Возврат в ОС

addition:

push rbp ; Сохранить значение указателя на стеке

mov rbp, rsp ; Установить указатель на текущий стек

sub rsp, 16 ; Выделить место под локальные переменные

mov DWORD [rbp-4], edi ; Сохранить значение первого аргумента

mov DWORD [rbp-8], esi ; Сохранить значение второго аргумента

mov edx, DWORD [rbp-4] ; Загрузить значение первого аргумента в регистр EDX

mov eax, DWORD [rbp-8] ; Загрузить значение второго аргумента в регистр EAX

add eax, edx ; Сложить значения аргументов

mov esi, eax ; Сохранить результат в переменной, переданной по второму аргументу

add eax, edx ; Сложить значения аргументов

mov edi, resultFormat ; Загрузить формат строки в регистр EDI

mov eax, 0 ; Загрузить код возврата в регистр EAX

call printf ; Вызвать функцию printf

mov edx, DWORD [rbp-4] ; Загрузить значение первого аргумента в регистр EDX

mov eax, DWORD [rbp-8] ; Загрузить значение второго аргумента в регистр EAX

add eax, edx ; Сложить значения аргументов и сохранить результат в регистре EAX

call checkOverflow ; Проверить на переполнение

leave ; Восстановить указатель стека

ret ; Возврат из функции

subsrtact:

push rbp

mov rbp, rsp

sub rsp, 16

mov DWORD [rbp-4], edi

mov DWORD [rbp-8], esi

mov edx, DWORD [rbp-4]

mov eax, DWORD [rbp-8]

sub eax, edx ; Отнять значения аргументов

mov esi, eax

sub eax, edx ; Отнять значения аргументов

mov edi, resultFormat

mov eax, 0

call printf

mov edx, DWORD [rbp-4]

mov eax, DWORD [rbp-8]

sub eax, edx

call checkOverflow

leave

ret

multiply:

push rbp

mov rbp, rsp

sub rsp, 16

mov DWORD [rbp-4], edi

mov DWORD [rbp-8], esi

mov edx, DWORD [rbp-4]

mov eax, DWORD [rbp-8]

imul eax, edx ; Умножить значения аргументов

mov esi, eax

imul eax, edx ; Умножить значения аргументов

mov edi, resultFormat

mov eax, 0

call printf

mov edx, DWORD [rbp-4]

mov eax, DWORD [rbp-8]

imul eax, edx

call checkOverflow

leave

ret

division:

push rbp

mov rbp, rsp

sub rsp, 16

mov DWORD [rbp-4], edi

mov DWORD [rbp-8], esi

mov eax, DWORD [rbp-4]

cdq

idiv DWORD [rbp-8]

mov esi, eax

mov edi, resultFormat

mov eax, 0

call printf

mov eax, DWORD [rbp-4]

cdq

idiv DWORD [rbp-8]

pxor xmm0, xmm0 ; Обнулить регистр XMM0

cvtsi2ss xmm0, eax ; Преобразовать результат в формат с плавающей точкой

call checkOverflow

leave

ret

\_start:

;Вывод сообщения EnterMessege

mov rdi, EnterMessege

mov rax, 0

call printf

; Ввод a

mov rdi, formatInput

lea rsi, [a]

mov rax, 0

call scanf

;Вывод сообщения EnterMessege2

mov rdi, EnterMessege2

mov rax, 0

call printf

;Ввод b

mov rdi, formatInput

lea rsi, [b]

mov rax, 0

call scanf

;Вызов функции addition

mov rdi, [a]

mov rsi, [b]

call addition

;Вызов функции subsrtact

mov rsi, [a]

mov rdi, [b]

call subsrtact

;Вызов функции multiply

mov rdi, [a]

mov rsi, [b]

call multiply

;Вызов функции division

mov rdi, [a]

mov rsi, [b]

cmp rsi, 0

jne NO\_DIVISION\_BY\_ZERO

mov edi, divisionByZeroMessage

mov eax, 0

call printf

jmp exit

NO\_DIVISION\_BY\_ZERO:

mov rdi, [a]

mov rsi, [b]

call division

exit:

mov rax, 60

mov rdi, 0

syscall

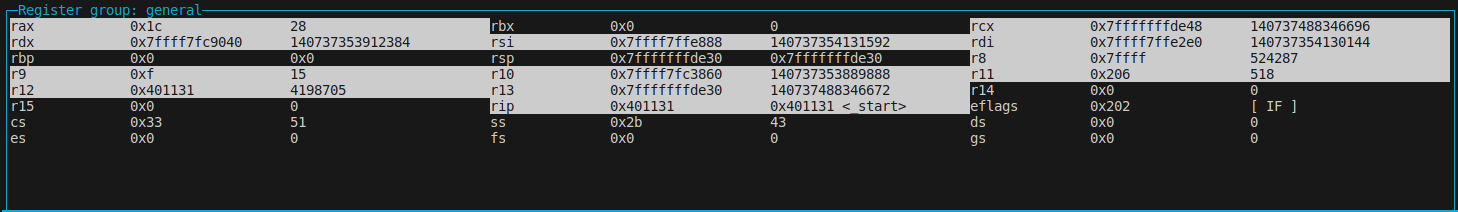


Рисунок 1 – Значения регистров перед выполнением

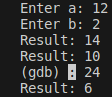


Рисунок 2 – Ход выполнения задачи

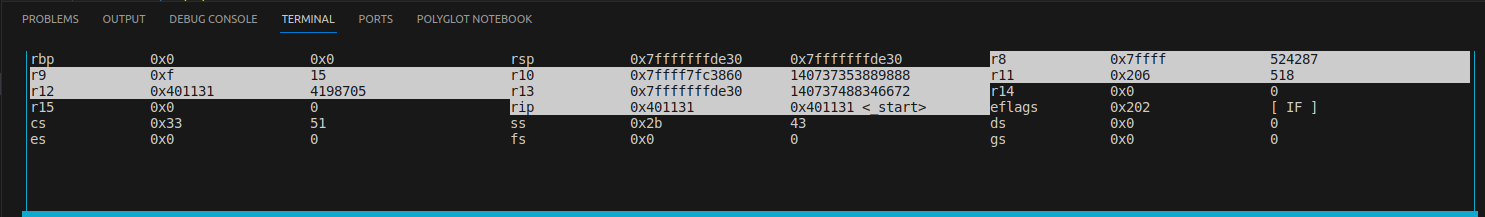


Рисунок 3 – Значения регистров после выполнения

**Вопросы:**

1.Проблемы ввода числа в ассемблерных программах (знак числа, переполнение при вводе)  
 Знак числа

Обработка знака

Переполнение

Ввод и конвертация

Обработка ошибок

2. Арифметические операции над целыми числами.

* Команда: ADD
* Команда: SUB
* Команда: IMUL (для знаковых чисел) или MUL (для беззнаковых чисел)
* Команда: IDIV (для знаковых чисел) или DIV (для беззнаковых чисел)
* Команды: INC и DEC
* Команда: IMUL с использованием дополнительного операнда
* Команды: AND, OR, XOR, NOT

3. Переполнение при арифметических операциях.

* Использование флагов переполнения (Overflow Flags):
* Использование условных переходов (Conditional Jumps):
* Контрольные точки и проверка диапазона (Range Checking):
* Использование расширенных типов данных: