|  |  |
| --- | --- |
| Gerb-BMSTU_01 | ***«*Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**по лабораторной работе № 3**

# Дисциплина: Машинно-зависимые языки и основы компиляции

**Название лабораторной работы:** Программирование ветвлений и   
итерационных циклов.

Студент гр. ИУ6-42Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Д. Шатский**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ C.C.Данилюк**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Вариант 2.23**

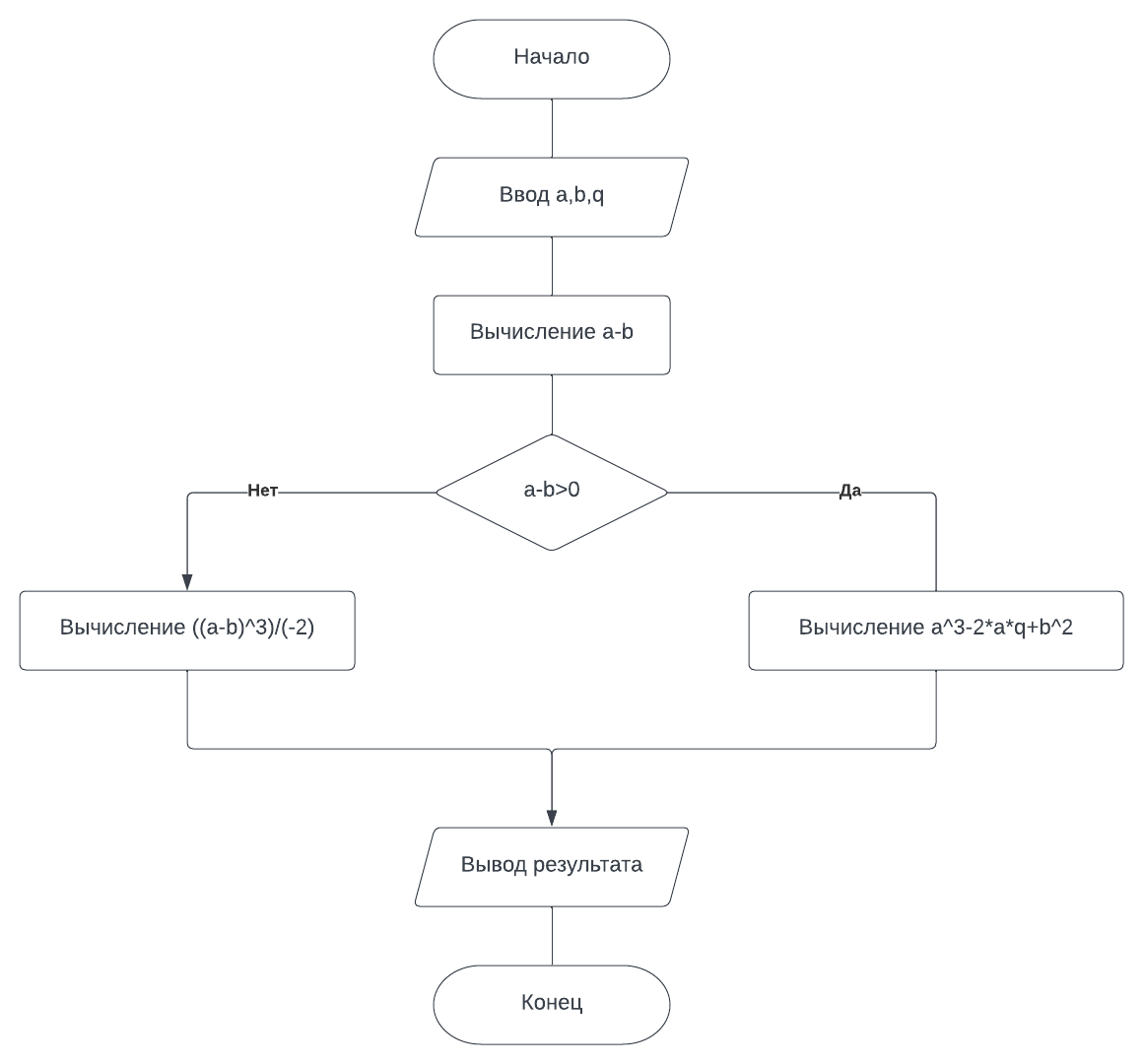
**Цель работы.** изучение средств и приемов программирования ветвлений и итерационных циклов на языке ассемблера.

**Задание.**

Вычислить целочисленное выражение:

**Схема алгоритма.**

Схему алгоритма можно увидеть на рисунке 1.



*Рисунок 1 – Алгоритм программы*

**Программа.**

section .data

EntrMsga db "Enter a:",10

lenEntra equ $-EntrMsga

EntrMsgb db "Enter b:",10

lenEntrb equ $-EntrMsgb

EntrMsgq db "Enter q:",10

lenEntrq equ $-EntrMsgq

ExitMsg db "Answer:",10

lenExit equ $-ExitMsg

section .bss

InBuf resb 10

lenIn equ $-InBuf

OutBuf resb 10

lenOut equ $-OutBuf

X resw 1

A resw 1

B resw 1

Q resw 1

section .text

global \_start

\_start:

mov rax, 1

mov rdi, 1

mov rsi, EntrMsga

mov rdx, lenEntra

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov esi,InBuf

call StrToInt64

cmp EBX, 0

jne StrToInt64.Error

mov [A],ax

mov rax, 1

mov rdi, 1

mov rsi, EntrMsgb

mov rdx, lenEntrb

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov esi,InBuf

call StrToInt64

cmp EBX, 0

jne StrToInt64.Error

mov [B],ax

mov rax, 1

mov rdi, 1

mov rsi, EntrMsgq

mov rdx, lenEntrq

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov esi,InBuf

call StrToInt64

cmp EBX, 0

jne StrToInt64.Error

mov [Q],ax

mov BX,[A]

sub BX,[B]

cmp BX,0

jle else

mov BX,[A]

mov AX,BX

imul BX

imul BX

mov CX,AX

mov AX,BX

mov BX,2

imul BX

mov BX,[Q]

imul BX

sub CX,AX

mov BX,[B]

mov AX,BX

imul BX

add CX,AX

mov AX,CX

jmp end

else:

mov AX,BX

imul BX

imul BX

mov BX,-2

idiv BX

end:

mov [X],AX

mov rax, 1; системная функция 1 (write)

mov rdi, 1; дескриптор файла stdout=1

mov rsi, ExitMsg ; адрес выводимой строки

mov rdx, lenExit; длина строки

syscall

mov esi, OutBuf ; загрузка адреса буфера вывода

mov ax, [X] ; загрузка числа в регистр

cwde ; развертывание числа из ax в eax

call IntToStr64

mov rdi, 1; дескриптор файла stdout=1

mov rsi, OutBuf ; адрес выводимой строки

mov rdx, lenOut ; длина строки

mov rax, 1; системная функция 1 (write)

; вызов системной функции

syscall

; read

mov rax, 0; системная функция 0 (read)

mov rdi, 0; дескриптор файла stdin=0

mov rsi, InBuf; адрес вводимой строки

mov rdx, lenIn; длина строки

; вызов системной функции

syscall

; exit

mov rax, 60; системная функция 60 (exit)

xor rdi, rdi; return code 0

syscall

; вызов системной функции

%include "../lib64.asm"

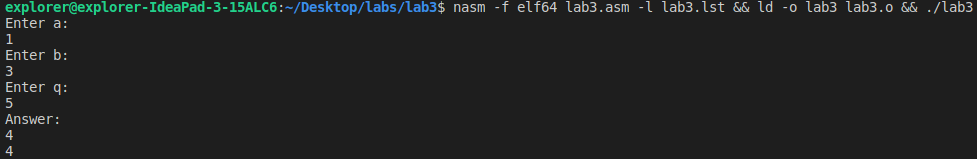
**Тесты.**

Тестирование и его результаты можно увидеть в таблице 1

Таблица 1 – результаты тестирования

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат |
| a=1 b=3 q=5 | 4 | 4 |
| a=-3 b=-5 q=7 | 40 | 40 |
| a=3 b=3 q=3 | 0 | 0 |
| a=0 b=0 q=0 | 0 | 0 |
| a=-5 b=-6 q=-7 | -159 | -159 |

Результат запуска программы на выполнение можно увидеть на рисунке 2:



*Рисунок 2 – Результат выполнения программы*

**Контрольные вопросы.**

**1. Какие машинные команды используют при программировании ветвлений и циклов?**

При программировании ветвлений и циклов используются команда

Сравнения CMP оператор безусловной передачи управления JMP и операторы условной передачи управления: JZ, JG, JNG, JGE, JNL и др.

Для создания итерационных циклов используются те же команды, только меняется последовательность их применения.

**2. Выделите в своей программе фрагмент, реализующий ветвление. Каково назначение каждой машинной команды фрагмента?**

cmp BX,0 ; выполнение сравнения

jle else ; если BX<0 то перейти на метку else

… ; команды если BX>0

jmp end ;пропустить команды, предназначенные для else, перейдя на метку end

else:

… ; команды если BX<0

end:

… ; продолжение выполнения вычислений

**3. Чем вызвана необходимость использования команд безусловной передачи управления?**

Команды безусловной передачи управления предназначены для того, чтобы после выполнения операций одной ветви переходить сразу к концу ветвления и не выполнять операции другой ветви. Также эти команды позволяют создать некоторые виды циклов.

**4. Поясните последовательность команд, выполняющих операции ввода-вывода в вашей программе. Чем вызвана сложность преобразований данных при выполнении операций ввода-вывода?**

Операция ввода:

mov rax, 0 ; системная функция 0 (read)

mov rdi, 0 ; дескриптор файла stdin=0

mov rsi, InBuf ; адрес вводимой строки

mov rdx, lenIn ; длина строки

syscall ; вызов системной функции

mov esi, InBuf ; загрузка адреса буфера вывода

call StrToInt64 ; вызов функции преобразования из строки в число

cmp ebx, 0 ; проверка кода ошибки

jne StrToInt64.Error ; при преобразовании обнаружена ошибка

mov [A], ax ; запись числа в память

Операция вывода:

mov esi, OutBuf ; загрузка адреса буфера вывода

mov ax, [X] ; загрузка числа в регистр

cwde ; развертывание числа из ax в eax

call IntToStr64 ; вызов функции преобразования из числа в строку

mov rax, 1 ; системная функция 1 (write)

mov rdi, 1 ; дескриптор файла stdout=1

mov rsi, ExitMsg ; адрес выводимой строки

mov rdx, lenExit ; длина строки

syscall ; вызов системной функции

Сложность преобразований данных при выполнении операций ввода-вывода вызвана тем, что при вводе чисел их необходимо преобразовывать из символьного представления в числовое, а при выводе все числа необходимо преобразовать в символьный формат.

**Вывод.** Былиизучены средства и приёмы программирования ветвлений и итерационных циклов на языке ассемблера. Программа работает корректно на заданных тестовых данных.