|  |  |
| --- | --- |
| 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 циклов.**

Студент гр. ИУ6 - 41Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. А Тарасова**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. С. Данилюк**

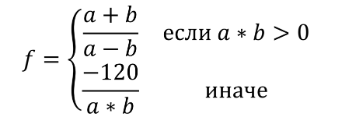
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Цель работы:** изучение средств и приемов программирования ветвлений

и итерационных циклов на языке ассемблера.

**Задание:** Вычислить целочисленное выражение:

****

**Код программы:**

%include "../lib.asm"

section .data

ExitMsg db "Press Enter to Exit",10

lenExit equ $-ExitMsg

EnterAMsg db "Enter the A which isn't 0: "

lenEnterA equ $-EnterAMsg

EnterBMsg db "Enter the B which isn't 0: "

lenEnterB equ $-EnterBMsg

ResMsg db "The result is: "

lenRes equ $-ResMsg

WrongAMsg db "A is 0!", 10

lenWrongA equ $-WrongAMsg

WrongBMsg db "B is 0!", 10

lenWrongB equ $-WrongBMsg

section .bss

A resd 1

B resd 1

OutBuf resb 10

InBuf resb 10

lenIn equ $-InBuf

section .text

global \_start

\_start:

rightA:

; write EnterAMsg

mov rax, 1

mov rdi, 1

mov rsi, EnterAMsg

mov rdx, lenEnterA

syscall

; read A

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov esi, InBuf

call StrToInt64

cmp ebx, 0

; проверка неравенства А нулю

cmp eax, 0

je rightA

mov [A], eax

rightB:

; write EnterBMsg

mov rax, 1

mov rdi, 1

mov rsi, EnterBMsg

mov rdx, lenEnterB

syscall

; read B

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov esi, InBuf

call StrToInt64

cmp ebx, 0

; проверка неравенства B нулю

cmp eax, 0

je rightB

mov [B], eax

; перемножение чисел для условия

mov ax, [A]

mov bx, [B]

imul bx

; начало сравнения

cmp ax, 0

jle less

mov ax, [A]

mov bx, [B]

sub ax, bx

mov cx, ax

mov ax, [A]

add ax, bx

cwd

idiv cx

jmp com

less:

mov ebx, eax

mov ax, -120

cwd

idiv bx

com:

mov ebx, eax ; переносим данные из EAX в EBX

; write ResMsg

mov rax, 1

mov rdi, 1

mov rsi, ResMsg

mov rdx, lenRes

syscall

mov eax, ebx ; переносим число обратно в EAX

; конвертируем частное из целого в строку

mov esi, OutBuf

cwde

call IntToStr64

; write

mov edx, eax

mov eax, 1

mov edi, 1

syscall

; write ExitMsg

mov rax, 1

mov rdi, 1

mov rsi, ExitMsg

mov rdx, lenExit

syscall

; read Enter

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

;exit

mov rax, 60

xor rdi, rdi

syscall

**Вывод программы (представлен на рисунках 1, 2):**

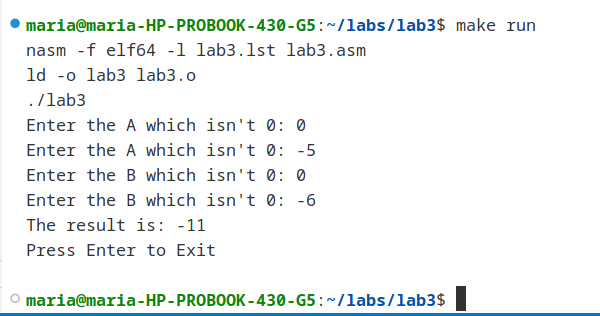
****

Рисунок 1 - результат программы

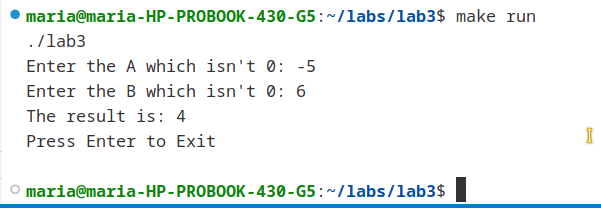
****

Рисунок 2 - результат программы

**Схема алгоритма (представлена на рисунке 3):**

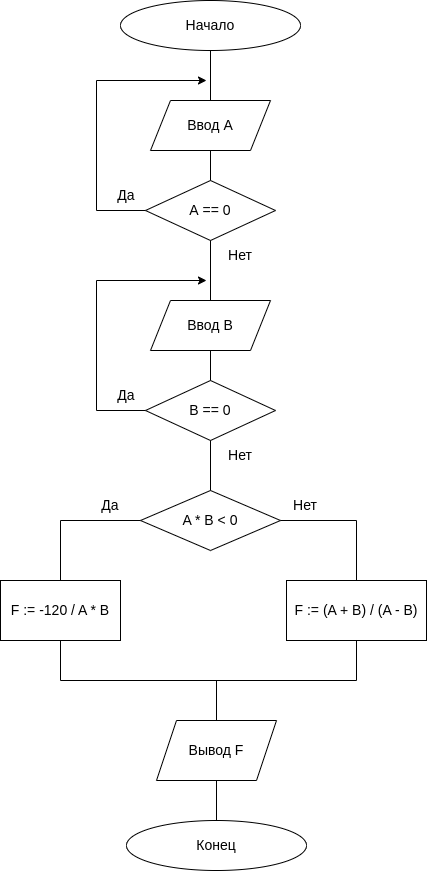
****

Рисунок 3 - Схема алгоритма

**Вывод:** в данной лабораторной работе мы разрабатывали программу, делающую те или иные вычисления с элементами ветвления и цикла.

**Контрольные вопросы:**

1. Какие машинные команды используют при программировании ветвле-

ний и циклов?

***cmp —*** сравнение данных в регистре с данными в другом регистре, переменной или литералом.

***jmp*** — переход на место, помеченное специальной «меткой», или на указанный адрес.

***J<условие>*** - условие, при котором происходит переход по указанному адресу/метке.

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

; перемножение чисел для условия

mov ax, [A]

mov bx, [B]

imul bx

; начало сравнения

cmp ax, 0 ; сравнение данных в ах с 0

jl less ; если число в ах < 0, переходим на метку less

mov ax, [A] ; если число в ах > 0, продолжаем вычисления в этом блоке

mov bx, [B]

sub ax, bx

mov cx, ax

mov ax, [A]

add ax, bx

cwd

idiv cx

jmp com ; по завершению вычислений переходим на метку com

less: ; при переходе по метке less делаем другие вычисления

mov ebx, eax

mov ax, -120

cwd

idiv bx

com: ; при переходе не метку com выводим результат

3. Чем вызвана необходимость использования команд безусловной передачи управления?

Когда проходит “if” и условие не выполняются, то программа продолжает работать дальше. И чтобы условие else не выполнилось, нужно этот фрагмент кода “перепрыгнуть” и тогда используются безусловные переходы.

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

Сначала мы вводим символ, который являются строкой. Он помещается в буфер преобразовывая в число, а после оно записывается в регистр. Вывод происходит наоборот, сначала число перевести в строку и после её напечатать.