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 (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

ОТЧЕТ

по лабораторной работе № 3

Дисциплина: МЗЯиОК

Название: Программирование ветвлений и итерационных циклов (14 вариант)

Студент ИУ6-43Б (Группа) 14.05.2024 (Подпись, дата) В.К. Залыгин (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) _____ (И.О. Фамилия)

Москва, 2024

Цель работы

Изучение средств и приемов программирования ветвлений и итерационных циклов на языке ассемблера.

Задание

Вычислить целочисленное выражение, представленное на рисунке 1.

$$f = \begin{cases} \dfrac{g - c}{a} + k & \text{если } g = m \\ m - g & \text{иначе} \end{cases}$$

Рисунок 1 – Формула выражения

Схема алгоритма

Схема алгоритма представлена на рисунке 2.

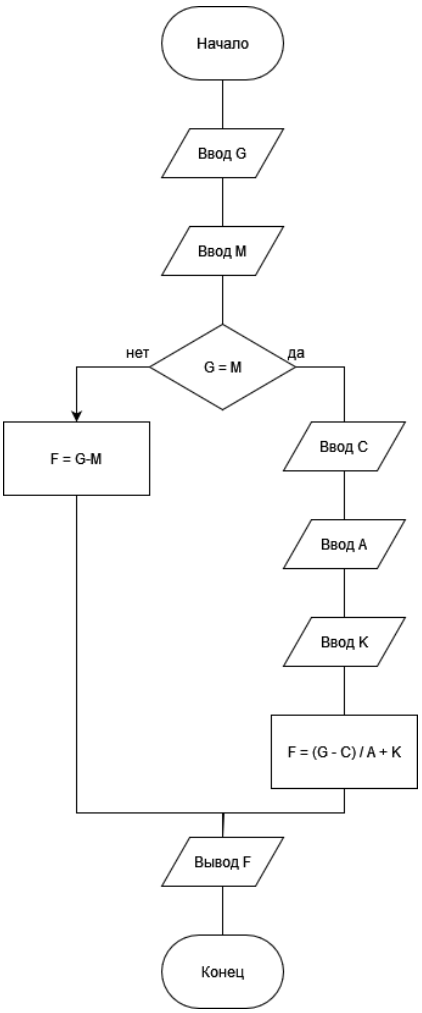

Рисунок 2 – Схема алгоритма

Код программы

Код программы представлен в листинге 1.

Листинг 1 – Код программы

```
    section .data
REQ_G     db     "Введите G:",10
REQ_G_LEN equ    $-REQ_G
REQ_M     db     "Введите M:",10
REQ_M_LEN equ    $-REQ_M
REQ_C    db     "Введите C:",10
REQ_C_LEN equ    $-REQ_C
REQ_A    db     "Введите A:",10
REQ_A_LEN equ    $-REQ_A
REQ_K    db     "Введите K:",10
REQ_K_LEN equ    $-REQ_K
RES    db     "Результат вычисления выражения:",10
RES_LEN    equ    $-RES
ERR    db     "Введены некорректные данные.
Завершение работы.",10
ERR_LEN    equ    $-ERR
THREE    dd    3
    section .bss
BUFFER    resb    10
BUFFER_LEN equ    $-BUFFER
G    resd    1
M    resd    1
C    resd    1
A    resd    1
K    resd    1
F    resd    1
section .text
global _start
_start:
    ; input the number G
    mov    rax,     1
    mov    rdi,     1
    mov    rsi,     REQ_G
    mov    rdx,     REQ_G_LEN
    syscall
    mov    rax,    0
    mov    rdi,    0
```

```
    mov    rsi,    BUFFER
    mov    rdx,    BUFFER_LEN
    syscall
    call    StrToInt64
    cmp    rbx,    0
    jne    .err
    mov    [G],    eax
    ; input the number M
    mov    rax,     1
    mov    rdi,     1
    mov    rsi,     REQ_M
    mov    rdx,     REQ_M_LEN
    syscall
    mov    rax,    0
    mov    rdi,    0
    mov    rsi,    BUFFER
    mov    rdx,    BUFFER_LEN
    syscall
    call    StrToInt64
    cmp    rbx,    0
    jne    .err
    mov    [M],    eax
    ; if g != m then compute f=m-g else continue
input vars
    mov    eax,    [M]
    cmp    eax,  [G]
    je    .coinp
    sub    eax,    [G]
    mov    [F],    eax
    jmp    .outp
.coinp:    mov    rax,     1    ; input the number
C
    mov    rdi,     1
    mov    rsi,     REQ_C
    mov    rdx,     REQ_C_LEN
    syscall
    mov    rax,    0
    mov    rdi,    0
    mov    rsi,    BUFFER
    mov    rdx,    BUFFER_LEN
    syscall
    call    StrToInt64
    cmp    rbx,    0
```

```
    jne    .err
    mov    [C],    eax
    ; input the number A
    mov    rax,     1
    mov    rdi,     1
    mov    rsi,     REQ_A
    mov    rdx,     REQ_A_LEN
    syscall
    mov    rax,    0
    mov    rdi,    0
    mov    rsi,    BUFFER
    mov    rdx,    BUFFER_LEN
    syscall
    call    StrToInt64
    cmp    rbx,    0
    jne    .err
     cmp    eax,    0
    je    .err
    mov    [A],    eax
    ; input the number K
    mov    rax,     1
    mov    rdi,     1
    mov    rsi,     REQ_K
    mov    rdx,     REQ_K_LEN
    syscall
    mov    rax,    0
    mov    rdi,    0
    mov    rsi,    BUFFER
    mov    rdx,    BUFFER_LEN
    syscall
    call    StrToInt64
    cmp    rbx,    0
    jne    .err
    mov    [K],    eax
    ; compute f = (g-c)/a+k
    mov    eax,    [G]
    sub    eax,    [C]
    xor    rdx,    rdx
    idiv    dword[A]
    add    eax,    [K]
    mov    [F],    eax
.outp    mov    rax,     1    ; output
    mov    rdi,     1
```

```
    mov    rsi,     RES
    mov    rdx,     RES_LEN
    syscall
    xor    rax,    rax
    mov    eax,    [F]
    mov    rsi,    BUFFER
    call    IntToStr64
    mov    rdx,    rax
    mov    rax,     1
    mov    rdi,     1
    syscall
    ; exit
    mov    rdi,    0
.exit    mov    rax,    60
    syscall
.err:    mov    rax,     1
    mov    rdi,     1
    mov    rsi,     ERR
    mov    rdx,     ERR_LEN
    syscall
    mov    rdi,    1
    jmp    .exit
%include "../lib.asm"
```

Тестирование

Результаты тестирования приведены в таблице 1.

Таблица 1 – Тестирование

№	Поток ввода	Ожидаемый результат	Поток вывода	Вердикт
1	5 4	1	1	Верно
2	0 1	-1	-1	Верно
3	0 0	3	3	Верно

	1 2 3			
4	0 0 0 0	Сообщение об ошибке.	Введены некорректные данные. Завершение работы.	Верно
5	a	Сообщение об ошибке.	Введены некорректные данные. Завершение работы.	Верно

Контрольные вопросы

1) Какие машинные команды используют при программировании ветвлений и циклов?

Loop, команды условной и безусловной передачи управления.

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

```
cmp   rbx,   0      ; сравнение значение регистра rbx с 0
jne   .err          ; если значение не равно 0, то перейти на метку .err
```

3) Чем вызвана необходимость использования команд безусловной передачи управления?

Необходимостью переключение регистра RIP на инструкцию, неследующую за текущей.

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

Сложность определяется тем, что формат вводимых данных отличается от формата представления данных в памяти. Вводятся строки, но хранятся числа и операции выполняются именно над ними.