**«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_____

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

# О Т Ч Е Т

## по лабораторной работе № 3

**Дисциплина:** Машинно-зависимые языки и основы компиляции

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

Вариант: 17

Студент гр. ИУ6-43Б **04.03.2022** **М.А. Мяделец**
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ **М.В. Широкова**
(Подпись, дата) (И.О. Фамилия)

Москва, 2022

## Программирование ветвлений и итерационных циклов

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

### Задание

Рассмотрим задание для лабораторной работы 3 (смотри рисунок 1).

**Лабораторная работ №3. Программирование ветвлений и циклов.**
Вычислить целочисленное выражение:

$$f = \begin{cases} a * y * \dfrac{(b-a)}{b} & \text{если } b - \text{четное} \\ a^5 - c^3 & \text{иначе} \end{cases}$$

*Рисунок 1 - Условие задания*

*Краткие моменты решения:*

1) Чтобы определить четность числа, можно применить побитовую операцию TEST, которая выполняет побитовое И без занесения результата. Тогда по значению флага ZF можно однозначно определить четность числа. Например, test 7, 1 = 1 (111 & 001 = 1), test 6, 1 = 0 (110 & 001 = 0).
2) Возведение в степень реализуем с помощью цикла, причем в пятую степень возведем, используя цикл-пока с помощью команд переходов, а в третью степень - используя счетный цикл посредством команд организации циклов (смотри блок-схему алгоритма на рисунке 2).

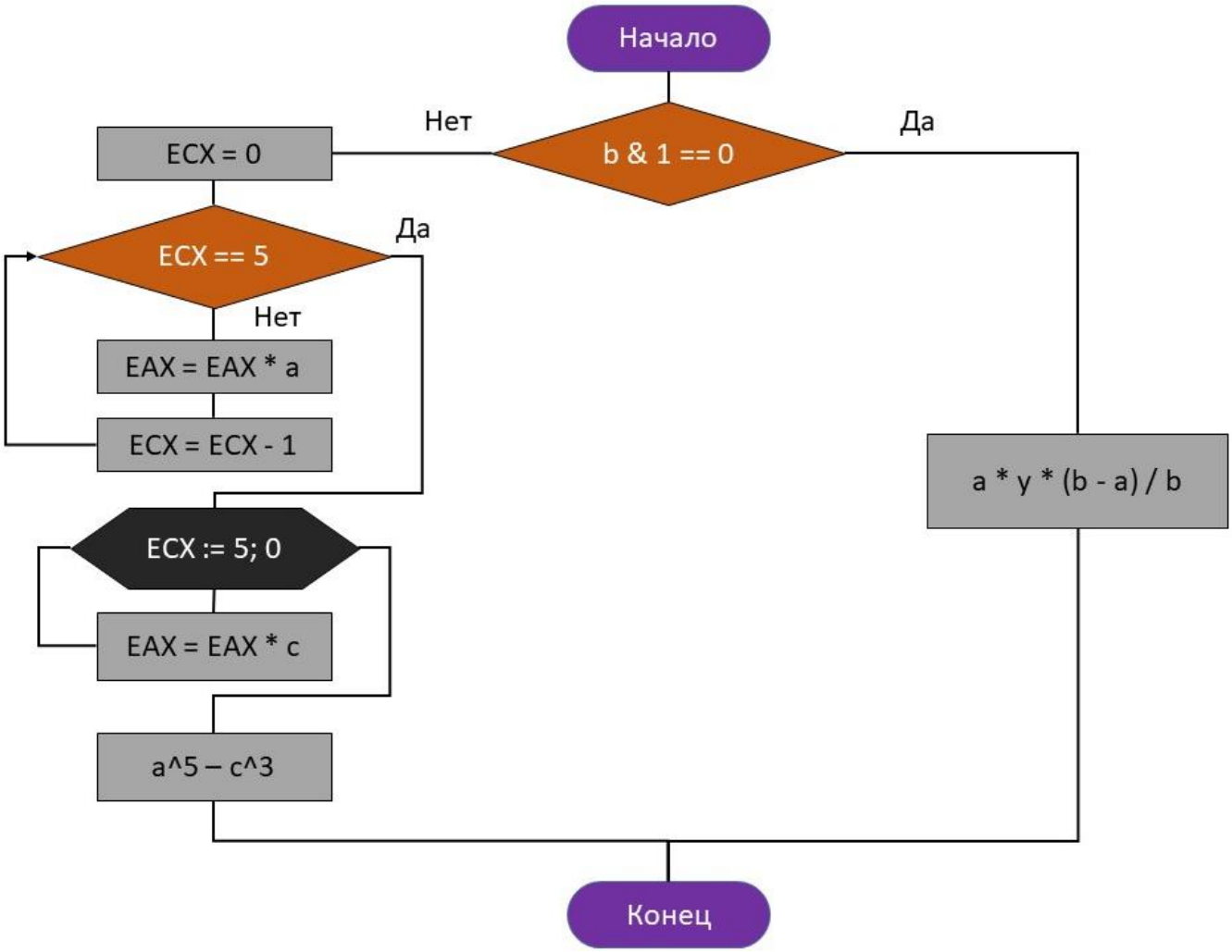
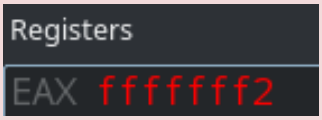
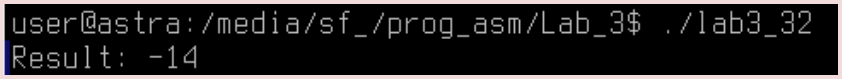
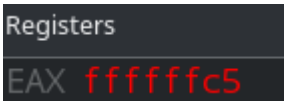
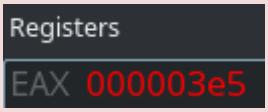
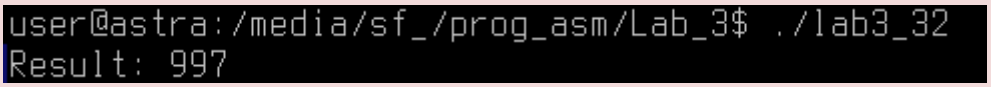
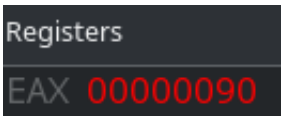
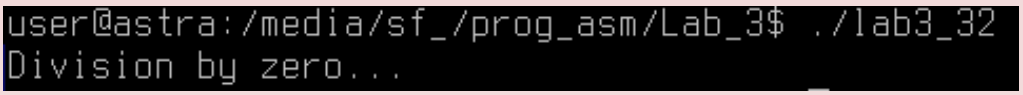



*Рисунок 2 - Блок-схема алгоритма*

## Таблица с результатами тестирования

*Таблица 1 - Результаты тестирования*

<table>
<tr><td colspan="2">Исходные данные</td><td>Значения функции<br>$f = \begin{cases} a * y * \frac{(b-a)}{b} & \text{если } b - \text{четное} \\ a^5 - c^3 & \text{иначе} \end{cases}$</td></tr>
<tr><td>a</td><td>-2</td><td rowspan="4">f(a, b, c, y) = (-2) * 7 * ((8 + 2) / 8) = (-14) * 1 = -14<br>  <br><br>  <br></td></tr>
<tr><td>b</td><td>8</td></tr>
<tr><td>c</td><td>3</td></tr>
<tr><td>y</td><td>7</td></tr>
<tr><td>a</td><td>-2</td><td rowspan="4">f(a, b, c, y) = (-2)^5 – 3^3 = -32 – 27 = -59<br>  <br><br>  <br></td></tr>
<tr><td>b</td><td>9</td></tr>
<tr><td>c</td><td>3</td></tr>
<tr><td>y</td><td>7</td></tr>
<tr><td>a</td><td>4</td><td rowspan="4">f(a, b, c, y) = 4^5 – 3^3 = 1024 – 27 = 997<br>  <br><br>  <br><br>Если a - положительное</td></tr>
<tr><td>b</td><td>-1</td></tr>
<tr><td>c</td><td>3</td></tr>
<tr><td>y</td><td>4</td></tr>
<tr><td>a</td><td>12</td><td rowspan="4">f(a, b, c, y) = 12 * 3 * ( (-4 - 12) / (-4) ) = 36 * 4 = 144<br>  <br><br>  <br></td></tr>
<tr><td>b</td><td>-4</td></tr>
<tr><td>c</td><td>2</td></tr>
<tr><td>y</td><td>3</td></tr>
<tr><td>a</td><td>1</td><td rowspan="4">Исключение: на ноль делить нельзя<br>  <br></td></tr>
<tr><td>b</td><td>0</td></tr>
<tr><td>c</td><td>2</td></tr>
<tr><td>y</td><td>3</td></tr>
</table>

**Вывод:** была сделана лабораторная работа 3, которая была нацелена на программирование ветвлений и итерационных циклов. Были изучены команды передачи управления, в частности команды условного и безусловного переходов. В ходе выполнения задания было принято решение возводить число в степень, используя циклы, поэтому в лабораторной работе также были применен цикл-пока и счетный цикл. Результаты тестирования корректны и оформлены в виде таблицы (смотри таблицу 1). Работа проведена успешно!

## Контрольные вопросы

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

При программировании циклов и ветвлений используют команду сравнения cmp, команды условного перехода je, jne, jl, jg…; безусловного перехода jmp. Для навигации используют метки.

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

В разработанной мной программе реализовано два вида циклов: цикл-пока с использованием команд переходов и счетный цикл с использованием команд организации циклической обработки.

*Цикл-пока.*

```
cycle:
        cmp ECX, [n]
        je break_cycle
        jmp cycle
        ...

break_cycle:
```

*Счетный цикл.*

```
mov ECX, [m]
jcxz end_loop
begin_loop:
        ...
        loop begin_loop
end_loop:
```

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

Тем, что необходимо пропускать некоторые операции, записанные последовательно. Например, в ветвлении пропустить ветку else, а в цикле осуществить выход из цикла.

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

Для осуществления операций ввода / вывода необходимо заполнить регистры определенными значениями, которые определять параметры системной функции. mov eax, 4 или mov eax, 3 задают системную функцию. В ebx пересылаем 0 или 1, тем самым указывания дескриптор файла stdin или stdout соответственно. ecx принимает адрес строки для ввода / вывода, edx содержит длину строки. Вызов системной функции происходит через прерывание int 80h. Также для ввода / вывода понадобиться преобразовать строку в число или число в строку, используя модули библиотеки StrToInt или IntToStr соответственно.

# ПРИЛОЖЕНИЕ

## Текст программы

```
            section    .data

    a       dd -2
    b       dd 8
    c       dd 3
    y       dd 7

    n       dd 5
    m       dd 3
    msg1    db "Result: "
    lenm1   equ $-msg1
msg0    db "Division by zero...", 10
    lenm0   equ $-msg0
    endline db 10

            section    .bss

    strres  resb 10

            section    .text
            global _start
    _start:

            test dword[b], 1        ; четное & 1 = 0, нечетное & 1 = 1
            jz else


                            ; b - нечетное, обрабатываем второе выражение
(ниже)
            mov EAX, 1      ; через цикл-пока считаем a^n, где n = 5
            mov ECX, 0
    cycle:
                    cmp ECX, [n]
                    je break_cycle
                    imul dword[a]
                    inc ECX
                    jmp cycle
```

```
break_cycle:

        mov EBX, EAX    ; a^n помещаем в EBX
                        ; через счетный цикл считаем c^m, где m = 3
        mov EAX, 1
        mov ECX, [m]
        jcxz end_loop
begin_loop:
                imul dword[c]
                loop begin_loop
end_loop:

        sub EBX, EAX    ; a^n - c^m
        mov EAX, EBX    ; результат в EAX

        jmp continue

else:                   ; обрабатываем первое выражение
        mov EAX, [b]
        cmp EAX, 0
        je exception    ; если b = 0, "генерируем исключение"

        sub EAX, [a]    ;          b - a
        cdq
        idiv dword[b]   ;         (b - a) / b
        imul dword[y]   ;     y * (b - a) / b
        imul dword[a]   ; a * y * (b - a) / b
        jmp continue

exception:              ; выводим "Division by zero..."
        mov eax, 4
        mov ebx, 1
        mov ecx, msg0
        mov edx, lenm0
        int 80h
        jmp exit

continue:               ; выводим "Result: <EAX>"
```

```
        push eax

        mov     eax, 4  ; Выводим "Result: "
        mov     ebx, 1
        mov     ecx, msg1
        mov     edx, lenm1
        int     80h

        pop eax
                        ; в EAX хранится целочисленный результат
        mov esi, strres ; заносим в esi буфер для результата строки

        call IntToStr   ; esi теперь хранит адрес строки "<EAX>", а EAX
- длину строки
        mov edi, eax

        mov eax, 4
        mov ebx, 1
        mov ecx, esi            ; esi содержит адрес строки "<EAX>"
        mov edx, edi            ; edi содержит длину строки "<EAX>"
        int 80h

exit:
        ; exit
        mov     eax, 1          ; системная функция 1 (exit)
        xor     ebx, ebx        ; код возврата 0
        int     80h             ; вызов системной функции

%include "./lib.asm"
```