|  |  |
| --- | --- |
| 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**

Вариант 2.21

Студент гр. ИУ6-42Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.С. Твердюк**

(Подпись, дата) (И.О. Фамилия)

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

(Подпись, дата) (И.О. Фамилия)

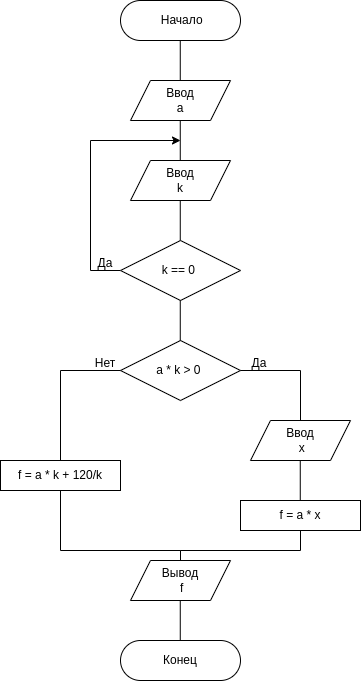
Москва, 2023

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

***Ход работы:***

***Задание.*** Вычислить целочисленное выражение:

Разработанная схема алгоритма приведена на рисунке 1.

Рисунок 1 - Схема алгоритма

**Программа LR3.asm:**

section .data

prsA db "Enter a:", 10

lenPrsA equ $-prsA

prsK db "Enter k(<> 0):", 10

lenPrsK equ $-prsK

prsX db "Enter x:", 10

lenPrsX equ $-prsX

ErrorMsg db "Varning! Devision by zero!", 10, "k should be equal to 0", 10, 10

lenError equ $-ErrorMsg

ExitMsg db "Result:", 10

lenExit equ $-ExitMsg

section .bss

InBuf resb 10

lenIn equ $-InBuf

OutBuf resb 10

a resw 1

k resw 1

x resw 1

f resw 1

section .text

global \_start

\_start:

; Enter a

mov rax, 1

mov rdi, 1

mov rsi, prsA

mov rdx, lenPrsA

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov rsi, InBuf

call StrToInt64

cmp ebx, 0

jne StrToInt64.Error

mov [a], ax

writeK:

; Enter K

mov rax, 1

mov rdi, 1

mov rsi, prsK

mov rdx, lenPrsK

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov rsi, InBuf

call StrToInt64

cmp ebx, 0

jne StrToInt64.Error

cmp ax, 0

je errorK

mov [k], ax

mov ax, [a]

mov dx, [k]

imul dx

cmp ax, 0 ; a \* k > 0 ?

JLE lessThanZero ; переход, если a \* k < 0

mov rax, 1

mov rdi, 1

mov rsi, prsX

mov rdx, lenPrsX

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov rsi, InBuf

call StrToInt64

cmp ebx, 0

jne StrToInt64.Error

xor dx, dx

mov dx, [a]

imul dx

mov [f], ax

jmp continue ; переход на конец ветвления

lessThanZero:

mov cx, [k]

mov ax, [a]

imul cx, ax

xor ax, ax

xor dx, dx

mov ax, 120

mov bx, [k]

cwd

idiv bx

add cx, ax

mov [f], cx

continue:

mov rax, 1

mov rdi, 1

mov rsi, ExitMsg

mov rdx, lenExit

syscall

mov rsi, OutBuf

mov ax, [f]

cwde

call IntToStr64

mov rdx, rax

mov rax, 1

mov rdi, 1

mov rsi, OutBuf

syscall

; exit

mov rax, 60

xor rdi, rdi

syscall

errorK:

mov rax, 1

mov rdi, 1

mov rsi, ErrorMsg

mov rdx, lenError

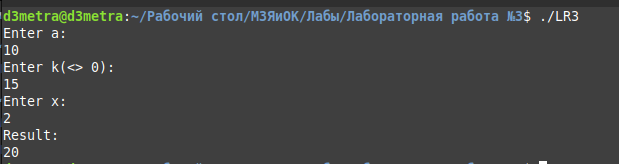
syscall

jmp writeK

%include "../lib64.asm"

Запустим программу. Результат выполнения программы представлен на рисунке 2.

Проведём тестирование программы. Результаты представлены в таблице 1.

Рисунок 2 - Выполнение программы

*Таблица 1 — Результаты тестирования программы*

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Полученный результат** |
| a = 0  k = 0  x = - | Devision by zero | Devision by zero |
| a = 50  k = 3  x = 20 | 1000 | 1000 |
| a = -15  k = 3  x = - | -5 | -5 |
| a = -100  k = -15  x = 60 | -6000 | -6000 |
| a = 14  k = -40  x = - | -563 | -563 |

***Контрольные вопросы***

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

При программировании ветвлений и циклов используются команды сравнения, условной и безусловной передачи управления: CMP, JMP, JZ, JG, JNG, JGE, JNL и др. В итерационных циклах используются те же команды, только меняется последовательность их применения.

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

cmp ax, 0 ; a \* k > 0 ?

JLE lessThanZero ; переход, если a \* k < 0

... ; вычисления при a \* k > 0

jmp continue ; переход на конец ветвления

lessThanZero: ; метка перехода

... ; вычисления при a \* k <= 0

continue: ; метка перехода

...

3. Чем вызвана необходимость использования команд безусловной передачи управления?

В программировании ветвлений команды безусловной передачи управления используются для того, чтобы после выполнения операций одной ветви переходить сразу к концу ветвления, из-за необходимости пропустить операции какой-то другой ветви. В программировании циклов эти команды позволяют осуществить повторение одной и той же последовательности команд нужное количество раз, пока команда условной передачи управления не выведет из цикла.

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

Для вывода:

mov rsi, OutBuf ; загрузка адреса буферы вывода

mov ax, [f] ; загрузка числа в регистр

cwde

call IntToStr64

mov rdx, rax ; длина результирующей строки

mov rax, 1 ; системная функция 1 (write)

mov rdi, 1 ; дескриптор файла stdout=1

mov rsi, OutBuf ; адрес выводимой строки

syscall

Для ввода:

mov rax, 0 ; системная функция 0 (read)

mov rdi, 0 ; дескриптор файла stdin=0

mov rsi, InBuf ; адрес вводимой строки

mov rdx, lenIn ; размер вводимой строки

syscall ; вызов системной функции

mov rsi, InBuf ; адрес введенной строки

call StrToInt64

cmp ebx, 0 ; проверка кода ошибки

jne StrToInt64.Error ; обнаружена ошибка

mov [a], ax ; запись числа в память

Сложность преобразований данных в том, что при вводе их нужно перевести из символьного представления в внутренний формат. Для целых чисел - это двоичный формат, отрицательные числа в дополнительном коде. После ввода получается строка, оканчивающаяся символом конца строки, но для перевода числа из символьного во внутреннее представление используется функция из библиотеки, которая принимает строку, оканчивающуюся нулем, поэтому также нужно использовать процедуру замены маркера конца строки нулем. При выводе стоит обратная задача. Нужно преобразовать числа в символьный формат и дальше вывод в консоль. Вывод: изучены средства и приемы программирования ветвлений и итерационных циклов на языке ассемблера. Разработанная программа на заданных исходных данных работает корректно.

***Вывод:*** изучены средства и приемы программирования ветвлений и итерационных циклов на языке ассемблера. Разработанная программа на заданных исходных данных работает корректно.