**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Образовательная программа бакалавриата «Программная инженерия»

|  |  |  |  |
| --- | --- | --- | --- |
|  | **РАЗРАБОТАТЬ** **ПРОГРАММУ ОПРЕДЕЛЕНИЯ КОЛИЧЕСТВА ЧИСЕЛ ФЕРМА ОТ 1 ДО БЕЗЗНАКОВОГО МАШИННОГО СЛОВА**  **Пояснительная записка** | | |
|  |  | |
| **Исполнитель**  студент группы БПИ197  \_\_\_\_\_\_/И. Ракичевич /  «30» октября 2020 г. | |
|  | | |
|  | |  |

**Москва 2020**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | | |
|  | **Пояснительная записка**  **Листов** | | | |
|  |  | | |
|  | | |
|  | | | |
|  | | |  |

**Москва 2020**

СОДЕРЖАНИЕ

[1. НАИМЕНОВАНИЕ ПРОГРАММЫ 4](#_Toc55141622)

[2. УСЛОВИЕ 5](#_Toc55141623)

[3. АЛГОРИТМ 6](#_Toc55141624)

[3.1 Описание алгоритма и функционирования программы 6](#_Toc55141625)

[3.1.1 Описание алгоритма нахождения чисел Ферма 6](#_Toc55141626)

[3.1.2 Общий алгоритм 6](#_Toc55141627)

[3.1.3 Начальные данные 6](#_Toc55141628)

[3.1.4 Функция main 6](#_Toc55141629)

[3.1.5 Функция l 7](#_Toc55141630)

[3.1.6 Функция ee 7](#_Toc55141631)

[3.2 Код программы 8](#_Toc55141632)

[ПРИЛОЖЕНИЕ 1 СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 11](#_Toc55141633)

[ПРИЛОЖЕНИЕ 2 ВЫПОЛНЕНИЕ ПРОГРАММЫ **Ошибка! Закладка не определена.**](#_Toc55141634)

# НАИМЕНОВАНИЕ ПРОГРАММЫ

Наименование программы: «Программа определения количества чисел Ферма от 1 до беззнакового машинного слова».

# УСЛОВИЕ

Разработать программу определения количества чисел Ферма от 1 до беззнакового машинного слова (Вариант 18).

# АЛГОРИТМ

## Описание алгоритма и функционирования программы

### Описание алгоритма нахождения чисел Ферма

Формула, по которой мы находим числа Ферма: 

### Общий алгоритм

Нахождение чисел ферма начинается с n = 0.

Находится первое число Ферма, сравнивается со значением беззнакового машинного слова, и, если оно меньше этого значения, то число Ферма выводится на экран, иначе программа завершается и на экран выводится количество чисел Ферма, удовлетворяющих условию.

### Начальные данные

format ELF64 ; Формат - запускаемый формат ELF64 - запускаемые программы на 64 битном Linux

section '.rodata' ; сегмент констант

welcome db 'Числа Ферма длины до машинного слова:', 10, 0

pfmt db '%llu',10,0 ; формат для печати 64битного числа: "%llf\n"

totfmt db 'Всего чисел Ферма до длины машинного слова: %llu',10, 0

public main ; наружу из программы экспортируем функцию main

extrn printf ; используем внешнюю функцию printf

section '.code' executable ; начало сегмента кода, в сегменте можно выполнять код

### Функция main

push rbp ; rbp в стек

mov rbp, rsp ; rsp->rbp, стек: адрес\_возврата, сохраненное\_rbp<-rpb

sub rsp, 16 ; создаем место под 2 локальные переменные (под 2, так как стек должен остаться выровненным на 16)

mov rdi, welcome ; 1 аргумент printf в rdi

xor rax, rax ; количество используемых векторных регистров -> 0

call printf ; печатаем приглашение

mov qword [rbp-8], 1 ; в [rbp-8] у нас будет степень двойки, в которую мы будем возводить двойку

mov qword [rbp-16], 0 ; счетчик чисел Ферма

### Функция l

mov rcx, [rbp-8] ; вытащили из стека нашу степень

cmp rcx, 64 ; сравнили ее с 64

je ee ; если равна, то все, идем в конец, потому что 2^64+1 >2^64-1

mov rsi, 1 ; заносим в rsi единицу

shl rsi, cl ; сдвигаем rsi на cl, то есть умножаем на 2^[rpb-8], то есть получаем 2^(2^n)

inc rsi ; увеличиваем на 1, в rsi очередное число Ферма, и это второй аргумент функции

mov rdi, pfmt ; первый аргумент функции - формат печати нашего числа ("%llu\n") в rdi

xor rax, rax ; количество используемых векторных регистров -> 0

call printf ; печатаем очередное число Ферма

shl qword [rbp-8],1 ; умножаем степень на 2

inc qword [rbp-16] ; увеличиваем счетчик числе Ферма

jmp l ; идем в начало цикла

### Функция ee

mov rdi, totfmt ; первый аргумент функции - формат печати

mov rsi, [rbp-16] ; второй - счетчик чисел Ферма

xor rax, rax ; количество используемых векторных регистров -> 0

call printf ; печаетаем число чисел Ферма

xor rax, rax ; код возврата программы - 0 (он в eax, а это часть rax)

leave ; восстанавливаем стек, аналог mov rsp, rbp | pop rbp

ret ; выходим из функции main

## Код программы

format ELF64 ; Формат - запускаемый формат ELF64 - запускаемые программы на 64 битном Linux

section '.rodata' ; сегмент констант

welcome db 'Числа Ферма длины до машинного слова:', 10, 0

pfmt db '%llu',10,0 ; формат для печати 64битного числа: "%llf\n"

totfmt db 'Всего чисел Ферма до длины машинного слова: %llu',10, 0

public main ; наружу из программы экспортируем функцию main

extrn printf ; используем внешнюю функцию printf

section '.code' executable ; начало сегмента кода, в сегменте можно выполнять код

main:

push rbp ; rbp в стек

mov rbp, rsp ; rsp->rbp, стек: адрес\_возврата, сохраненное\_rbp<-rpb

sub rsp, 16 ; создаем место под 2 локальные переменные (под 2, так как стек должен остаться выровненным на 16)

mov rdi, welcome ; 1 аргумент printf в rdi

xor rax, rax ; количество используемых векторных регистров -> 0

call printf ; печатаем приглашение

mov qword [rbp-8], 1 ; в [rbp-8] у нас будет степень двойки, в которую мы будем возводить двойку

mov qword [rbp-16], 0 ; счетчик чисел Ферма

l:

mov rcx, [rbp-8] ; вытащили из стека нашу степень

cmp rcx, 64 ; сравнили ее с 64

je ee ; если равна, то все, идем в конец, потому что 2^64+1 >2^64-1

mov rsi, 1 ; заносим в rsi единицу

shl rsi, cl ; сдвигаем rsi на cl, то есть умножаем на 2^[rpb-8], то есть получаем 2^(2^n)

inc rsi ; увеличиваем на 1, в rsi очередное число Ферма, и это второй аргумент функции

mov rdi, pfmt ; первый аргумент функции - формат печати нашего числа ("%llu\n") в rdi

xor rax, rax ; количество используемых векторных регистров -> 0

call printf ; печатаем очередное число Ферма

shl qword [rbp-8],1 ; умножаем степень на 2

inc qword [rbp-16] ; увеличиваем счетчик числе Ферма

jmp l ; идем в начало цикла

ee:

mov rdi, totfmt ; первый аргумент функции - формат печати

mov rsi, [rbp-16] ; второй - счетчик чисел Ферма

xor rax, rax ; количество используемых векторных регистров -> 0

call printf ; печаетаем число чисел Ферма

xor rax, rax ; код возврата программы - 0 (он в eax, а это часть rax)

leave ; восстанавливаем стек, аналог mov rsp, rbp | pop rbp

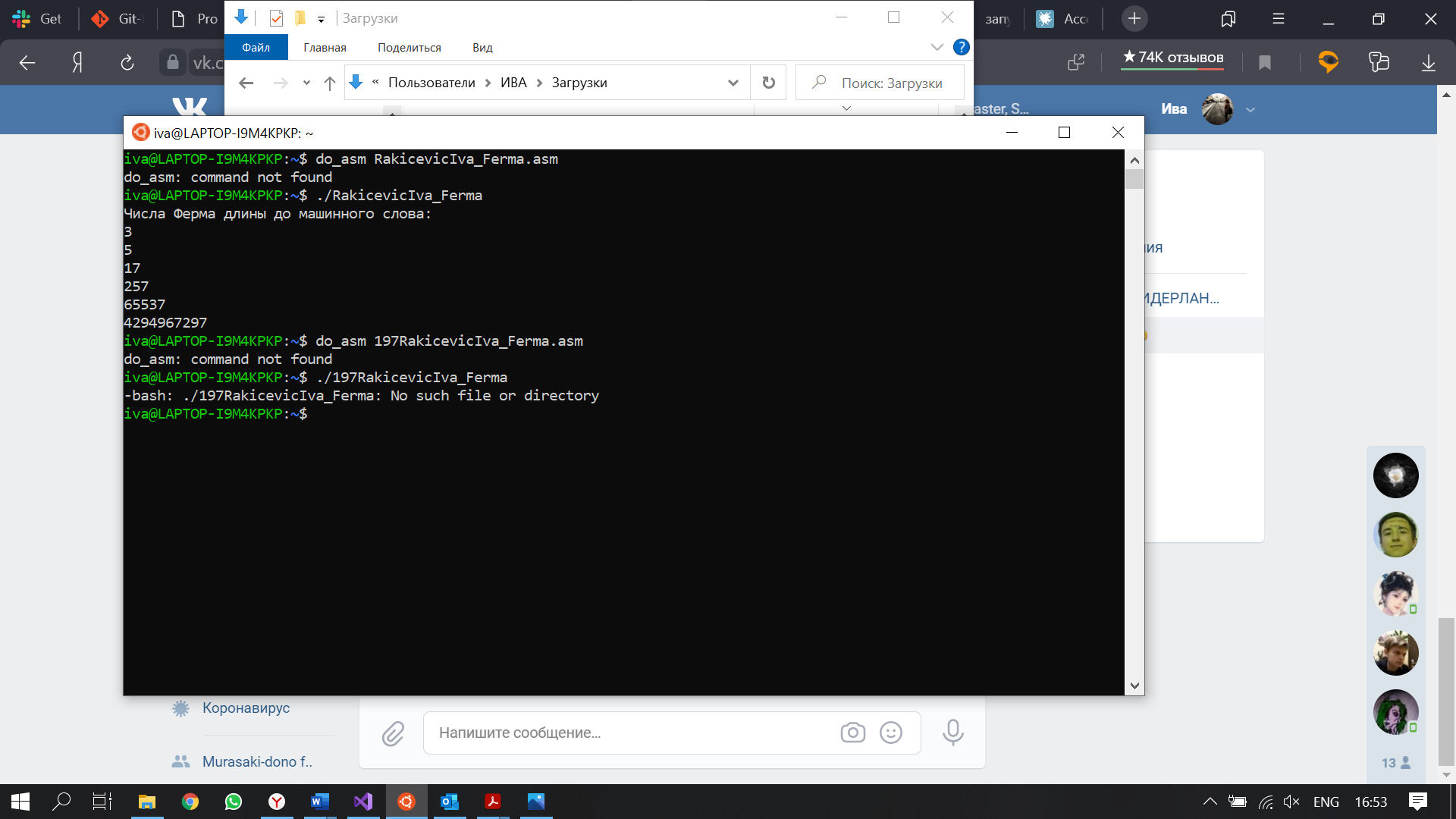
ret ; выходим из функции main

## Текст работы

Репозиторий на git: https://github.com/mikamurasaki2/Microproject1

### Выполнение работы

В оригинальном файле с выполненной задачей выводится еще и количество чисел Ферма. К сожалению, не успела заменить более поздний снимок на новый.



# ПРИЛОЖЕНИЕ 1 СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. SoftCraft (разноликое программирование), Архитектура вычислительных систем: <http://www.softcraft.ru/edu/comparch/>
2. Википедия: <https://ru.wikipedia.org/wiki/Число_Ферма>
3. Требования к оформлению. 2020-2021 уч.г.: <http://www.softcraft.ru/edu/comparch/tasks/mp01/>