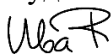


**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

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
2.	УСЛОВИЕ.....	5
3.	АЛГОРИТМ.....	6
3.1	Описание алгоритма и функционирования программы.....	6
3.1.1	Описание алгоритма нахождения чисел Ферма	6
3.1.2	Общий алгоритм	6
3.1.3	Начальные данные	6
3.1.4	Функция main	6
3.1.5	Функция I	7
3.1.6	Функция ee	7
3.2	Код программы.....	8
ПРИЛОЖЕНИЕ 1 СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ		11
ПРИЛОЖЕНИЕ 2 ВЫПОЛНЕНИЕ ПРОГРАММЫ		Ошибка! Закладка не определена.

1. НАИМЕНОВАНИЕ ПРОГРАММЫ

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

2. УСЛОВИЕ

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

3. АЛГОРИТМ

3.1 Описание алгоритма и функционирования программы

3.1.1 Описание алгоритма нахождения чисел Ферма

Формула, по которой мы находим числа Ферма: $F_n = 2^{2^n} + 1$, где $n \geq 0$

3.1.2 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.

3.1.3 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

3.1.4 Функция main

push rbp ; rbp в стек

mov rbp, rsp ; rsp->rbp, стек: адрес_возврата, сохраненное_rbp<-rbp

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

```

3.1.5 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  $2^{[rbp-8]}$ , 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 чисел Ферма

jmp l                 ; идем в начало цикла

```

3.1.6 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           ; печатаем 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
```

3.2 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<-rbp
```

```
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  $2^{[rbp-8]}$ , 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 чисел 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         ; печатаем 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

```

3.3 Текст работы

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

3.4 Выполнение работы

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

```
Числа Ферма длины до машинного слова:  
3  
5  
17  
257  
65537  
4294967297
```

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/>