

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

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

**ПРОГРАММА ДЛЯ ВЫЧИСЛЕНИЯ ЧИСЛА ПРОСТЫХ ЧИСЕЛ
ФИБОНАЧЧИ В ДИАПАЗОНЕ ОТ 1 ДО МАШИННОГО СЛОВА**

Пояснительная записка

Исполнитель
студент группы **БПИ193**
Тимканов Имран
Алимович

31 октября 2020 г.

Формулировка задачи

Разработать программу, определяющую число простых чисел Фибоначчи в диапазоне от 1 до машинного слова.

Рекуррентная формула вычисление чисел Фибоначчи

$$F_n = F_{n-1} + F_{n-2}$$

Описание расчетного алгоритма программы

1. Общая структура программы:

```
start:
    mov [number], 65536 ; 2^16 = машинное слово

    ; count number of prime digits that are in fibbo sequence
    call getCount
finish:
    call [getch]

    push 0
    call [ExitProcess]
```

1.1 getCount– подсчет простых чисел Фибоначчи от 1 до введенного машинного слова

1.2 finish – вывод результатов в консоль

1. Подсчет простых чисел Фибоначчи от 1 до введенного машинного слова.

1.1 Нахождение текущего числа Фибоначчи по рекуррентной формуле

```
getCount:
    mov ebx, [lastFibbo] ; сохраним последнее число Фибоначчи в регистр ebx
    add ebx, [prelastFibbo] ; сохраним предпоследнее число Фибоначчи в регистр ebx
    mov [tmp], ebx ; перезаписываем в переменную временного хранилища

    mov eax, [number]
    cmp eax, [tmp] ; сравним рассматриваемое число Фибоначчи и максимальное число – если оно больше, то заканчиваем
    jl outputInfo ; вывод информации завершение
```

1.2 Проверка числа на простоту. Этот этап обусловлен проверкой все чисел от 2 до рассматриваемого числа – 1. В случае, если на этом промежутке найдется хоть один делитель числа, то текущее число Фибоначчи простым являться не будет.

```
primeCheckFunc:
    mov ecx, [lastFibbo]
    cmp [startNumber], ecx ; проверяем строго от 2 до самого числа – 1. Если хоть одно число будет делить – то исходное число не простое
    jb primeCheck
    jmp getCount ; повторим вновь

primeCheck:
    mov dx, 0
    mov ax, 0

    mov dx, word [lastFibbo+2]
    mov ax, word [lastFibbo] ; в регистр eax положим рассматриваемое на простоту число
    div [startNumber] ; поделить рассматриваемое число на текущий делитель. Остаток пойдет в регистр dx

    cmp dx, 0 ; Если остаток 0 --> число точно не простое, тогда переходим в метку "a"
    je a
    add [startNumber], 1
    jmp primeCheckFunc

a:
    sub [numberFibboPrime], 1 ; вычитаем единицу, которую прибавили априорно, поскольку убедились в обратном нашему предположению
    jmp getCount ; возвращаемся в getCount, где рассматриваем другое число
```

1.3. Вывод подсчитанной информации. Докладывает пользователю информацию в соответствии с требованиями – выводить число простых чисел Фибоначчи.

```
; -----
outputInfo:
    ; вывод ответа на задачу
    push [numberFibboPrime]
    push [number] ; число чисел
    push printNumber ; строка форматирования
    call [printf]
    add esp, 8
    call finish

; -----
```

Список переменных

```
section '.data' data readable writable
ScanInt db '%u', 10, 0
printNumber db "General number of prime & fibbo numbers in range [1;%u] = %u;", 0 ; строка ответа на задачу (ответ зависит от машинного слова)

numberFibboPrime dd 0 ; переменная – колво простых чисел среди чисел Фибоначчи в диапазоне [1; машинное слово]
startNumber dd 2 ; стартовое число для проверки на простоту

lastFibbo dd 1 ; предпоследнее рассматриваемое число Фибоначчи
prelastFibbo dd 1 ; последнее рассматриваемое число Фибоначчи
tmp dd 1


number dd ? ; машинное слово введенное пользователем
```

- 1) Tmp - временная переменная для хранения текущего числа фибоначчи

Область допустимых значений

Пользователь не принимает участие в ходе программы

Тестирование

 C:\Users\Имран Тимканов\Downloads\fasmw17325\FASM_PROJ\MiniProjectMain.EXE

```
General number of prime & fibbo numbers in range [1;65536] = 8;
```

Список используемых источников

1. <http://flatassembler.narod.ru/fasm.htm#2-1-13> (Описание данных, Инструкции FPU)
2. <http://flatassembler.narod.ru/fasm.htm#2-1-13>
3. <https://prog-cpp.ru/asm-operands/>

Приложение

1. Код программы (FASM)

```
33 ;-----
34 getCount:
35     mov ebx, [lastFibbo] ; сохраним последнее число Фибоначчи в регистр eax
36     add ebx, [prelastFibbo] ; сохраним предпоследнее число Фибоначчи в регистр ebx
37     mov [tmp], ebx ; перезаписываем в переменную временного хранилища
38
39     mov eax, [number]
40     cmp eax, [tmp] ; сравним рассматриваемое число Фибоначчи и максимальное число – если оно больше, то заканчиваем
41     jl outputInfo ; вывод информации завершение
42
43     mov edx, [lastFibbo] ; перезапись в регистр
44     mov [prelastFibbo], edx ; prelast — last
45     mov edx, [tmp]
46     mov [lastFibbo], edx ; last — новый
47
48
49     add [numberFibboPrime], 1 ; апостериорно добавим
50     mov [startNumber], 2 ; назовем первое число проверки
51     jmp primeCheckFunc
52 ;-----
53 primeCheckFunc:
54     mov ecx, [lastFibbo]
55     cmp [startNumber], ecx ; проверяем строго от 2 до самого числа – 1. Если хоть одно число будет делить – то исходное число не простое
56     jb primeCheck
57     jmp getCount ; повторим вновь
58 primeCheck:
59     mov dx, 0
60     mov ax, 0
61
62     mov dx, word [lastFibbo+2]
63     mov ax, word [lastFibbo] ; в регистр eax положим рассматриваемое на простоту число
64     div [startNumber] ; поделить рассматриваемое число на текущий делитель. Остаток пойдет в регистр dx
65
66     cmp dx, 0 ; Если остаток 0 —> число точно не простое, тогда переходим в метку "a"
67     je a
68     add [startNumber], 1
69     jmp primeCheckFunc
70 a:
71     sub [numberFibboPrime], 1 ; вычитаем единицу, которую прибавили априорно, поскольку убедились в обратном нашему предположению
72     jmp getCount ; возвращаемся в getCount, где рассматриваем другое число
73 ;-----
74 outputInfo:
75     ; вывод ответа на задачу
76     push [numberFibboPrime]
77     push [number] ; число чисел
78     push printNumber ; строка форматирования
79     call [printf]
80     add esp, 8
81     call finish
82 ;-----
83 section '.idata' import data readable ; подключаем все библиотеки
84     library kernel, 'kernel32.dll',\
85         msvcrt, 'msvcrt.dll'
86
87     import kernel,\
88         ExitProcess, 'ExitProcess'
89
90     import msvcrt,\
91         printf, 'printf',\
92         scanf, 'scanf',\
93         getch, '_getch'
94
```

```
1 ; Работу выполнил Тимканов Иман БПИ193
2
3
4 format PE console
5 entry start
6
7 include 'win32a.inc' ; Подключаем библиотеку
8
9 section '.data' data readable writable
10     ScanInt db '%u',10, 0
11     printNumber db "General number of prime & fibbo numbers in range [1;%u] = %u;",0 ; строка ответа на задачу (ответ зависит от машинного слова)
12
13     numberFibboPrime dd 0 ; переменная – колво простых чисел среди чисел Фибоначчи в диапазоне [1; машинное слово]
14     startNumber dd 2 ; стартовое число для проверки на простоту
15
16     lastFibbo dd 1 ; предпоследнее рассматриваемое число Фибоначчи
17     prelastFibbo dd 1 ; последнее рассматриваемое число Фибоначчи
18     tmp dd 1
19
20     number dd ?; машинное слово введенное пользователем
21
22 section '.code' code readable executable
23 start:
24     mov [number], 65536 ; 2^16 = машинное слово
25
26     ; count number of prime digits that are in fibbo sequence
27     call getCount
28 finish:
29     call [getch]
30
31     push 0
32     call [ExitProcess]
33 ;-----
```