**Правительство Российской Федерации**

**Федеральное государственное автономное образовательноеучреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»**

Факультет компьютерных наук

Департамент программной инженерии

# Отчет к домашнему заданию По дисциплине

**«Архитектура вычислительных систем»**

Работу выполнил:

Студент группы БПИ-197 Аскербеков Д.A.

**Москва 2020**

**Задача**

Разработать программу определения количества обобщённых чисел Ферма a^2 n +b^2 n , где n неотрицательное целое число при a=2, b=3 от 1 до беззнакового двойного машинного слова **Решение**

Чтобы определить максимальное число Ферма, не превышающее значение беззнакового двойного машинного слова () нужно пройтись циклом с постусловием в котором будет считаться число Ферма под номером после чего это число будет проверяться на переполнение и если наш unsigneddwordпереполнился, то число Ферма под номером превысило максимальное значение беззнакового двойного слова, цикл завершается и на экран выводится .

Реализуем этот алгоритм на языке программирования C++ для того, чтобы убедится, что алгоритм работает верно. Учтем, что беззнаковое двойное машинное слово – это unsignedintв C++.

Текст программы приведен ниже:

#include<iostream>

#include <cmath>

int main() {

unsigned intmaxValue = 4294967295;

inti = 0;

unsigned int number = 0;

do

{

number = pow(2, pow(2, i)) + pow(3, pow(2, i)) ;

i++;

} while (number != maxValue); //проверяем на переполнение

std::cout<< --i;

}

В C++ при переполнении беззнаковых целочисленных типов переменной присваивается максимальное значение этого типа, то есть в данном случае . Известно, что ни одно число Ферма не равно , а значит без каких либо проблем можно сравнивать numberс maxValue.

По итогу программа вывела в консоль 5, а значит описанный алгоритм работает верно и его можно переносить на язык ассемблера для компилятора FASM.

**Функция printf**

Для вывода результата будем использовать функцию printf, которая форматирует данные по заданному шаблону и выводит их на консоль.

intprintf(const char\*format,…);

Она принимает следующие аргументы:

* char\*format – указатель на С-строку, содержащую формат результата
* Следующие аргументы – данные, подлежащие форматированию

Функция принимает в себя неограниченное количество параметров и по этой причине она использует соглашение вызова cdecl, а следовательно отчистку стека от переданных аргументов выполняет вызывающая функция.

**Функция ExitProcess**

Для завершения работы программы будем использовать функцию ExitProcess(uintuExitCode), которая завершает работу программы.

Она принимает следующие аргументы:

* uintuExitCode – определяет код выхода для процесса и для всех потоков, которые завершают работу в результате вызова функции.

В процессе реализации алгоритма на языке ассемблера алгоритм был разбит на несколько функций:

**main():**

Главная функция программы. В ней вызывается функция для подсчета количества числе Ферма, удовлетворяющих нашему условию и функция printfдля вывода полученного результата в консоль. Также в ней содержится функция getchсчитывающая символ и функция ExitProcess, завершающая работу программы.

**uint pow(uintnum, uint pow)**

Функция принимающая в себя два аргумента:

* + uintnumber – возводимое в некоторую степень число
  + uintpower – степень, в которую будет возводится число number

Функция содержит в себе локальные переменные:

* inti – счетчик

Результатом этой функции является number в степени power, результат возвращается в регистр eax так как функция реализует соглашение вызова cdecl.

**uintfindCountFermaNum(uintmaxValue)**

Функция принимающая в себя один аргумент:

* + uintmaxValue – возводимое в некоторую степень число

Функция содержит в себе локальные переменные:

* + inti – счетчик
  + uintoldVal – предыдущее число Ферма
  + uintval – текущее число Ферма

Результатом этой функции являетсяколичество чисел Ферма меньших maxValue, результат возвращается в регистр eax так как функция реализует соглашение вызова cdecl.

**uintfindFermaNum(intn)**

Функция принимающая в себя аргумент:

* + int n – номерчисла Ферма

Результатом функции является число Ферма под номером n, результат возвращается в регистр eax так как функция реализует соглашение вызова cdecl.

**Текстпрограммы**

format PE console

include 'win32a.inc'

entry start

section '.data' data readable writable

strPrintNum db '%u', 0

strCountFermaNum db 'Count of Ferma numbers: %d', 10, 0

maxDword dd 4294967295

NULL = 0

section '.code' code readable executable

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_MAIN\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

start:

push [maxDword]

call findCountFermaNum

add esp, 4

push eax

push strCountFermaNum

call [printf]

add esp, 8

call [getch]

push NULL

call ExitProcess

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_MAIN\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_POW\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

pow:

push ecx

push ebp

mov ebp, esp

sub esp, 4

;----------Локальные переменные------------

i equ ebp-4

;---------------Параметры ------------------

num equ ebp+16

power equ ebp+12

;------------------------------------------

mov eax, 1 ;result=1

mov [i], dword 0 ;i=0

powLoop:

mov ecx, [i]

cmp ecx, [power]

jge finishPowLoop

imul eax, dword [num]

inc dword [i]

jmp powLoop

finishPowLoop:

mov esp, ebp

pop ebp

pop ecx

ret

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_POW\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

;\_\_\_\_\_\_\_\_\_\_\_FIND\_COUNT\_OF\_FERMA\_NUM\_\_\_\_\_\_\_\_\_

findCountFermaNum:

push ecx

push ebp

mov ebp, esp

sub esp, 8

;----------Локальные переменные ------------

i equ ebp-4

oldVal equ ebp-8

val equ ebp-12

;---------------Параметры ------------------

maxVal equ ebp+12

;------------------------------------------

mov [i], dword 0 ;i = 0

mov [oldVal], dword 0 ;oldValue = 0

fermaLoop:

push dword [i]

call findFermaNum

add esp, 4

mov [val], eax ;val = findFermaNum(i)

mov ecx, [val] ;ecx = val

cmp ecx, [oldVal]

jb finishFermaLoop

mov [oldVal], ecx ;oldVal = val

inc dword [i] ;ecx++

jmp fermaLoop

finishFermaLoop:

mov eax, [i]

dec eax

mov esp, ebp

pop ebp

pop ecx

ret

;\_\_\_\_\_\_\_\_\_\_\_FIND\_COUNT\_OF\_FERMA\_NUM\_\_\_\_\_\_\_\_\_

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_FIND\_FERMA\_NUM\_\_\_\_\_\_\_\_\_\_\_\_\_\_

findFermaNum:

push ebp

mov ebp, esp

;----------------Параметры-----------------

n equ ebp+8

;------------------------------------------

push 2

push dword [n]

call pow

add esp, 8

push 2

push eax

call pow

add esp, 8

mov edx, eax

mov eax, 0

push 2

push dword [n]

call pow

add esp, 8

push 3

push eax

call pow

add esp, 8

add eax, edx

mov esp, ebp

pop ebp

ret

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_FIND\_FERMA\_NUM\_\_\_\_\_\_\_\_\_\_\_\_\_

section '.idata' data readable import

library kernel, 'kernel32.dll',\

msvcrt, 'msvcrt.dll'

import kernel,\

ExitProcess, 'ExitProcess'

import msvcrt,\

printf, 'printf',\

getch, '\_getch'

**Тестирование**

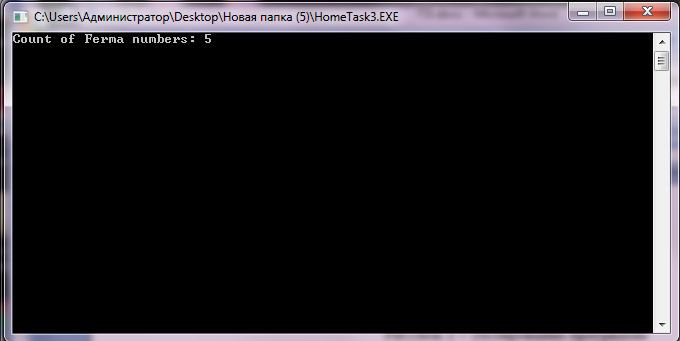


Рисунок 1 – Тестирование программы

Программа работает корректно и всегда выводит правильный результат (см. рис. 1)

**Список используемых источников**

1. Википедия (2020) «Число Ферма» (<https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE_%D0%A4%D0%B5%D1%80%D0%BC%D0%B0>)
2. SoftCraft«Программирование на языке ассемблера. Микропроект.Требования к оформлению. 2020-2021 уч.г.» (<http://softcraft.ru/edu/comparch/tasks/mp01/>)
3. natalia.appmat«Программирование на языке ассемблера» (<http://natalia.appmat.ru/c&c++/assembler.html>)
4. osinavi «Команды передачи управления»(<http://osinavi.ru/asm/4.html>)
5. vsokovikov.narod«Функция ExitProcess» (<http://vsokovikov.narod.ru/New_MSDN_API/Process_thread/fn_exitprocess.htm>)
6. Microsoft «Вызов функций C во встроенном коде на языке ассемблера» (<https://docs.microsoft.com/ru-ru/cpp/assembler/inline/calling-c-functions-in-inline-assembly?view=msvc-160>)