

Правительство Российской Федерации

Федеральное государственное автономное образовательное

учреждение 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-194 Назмутдинов Р.Р.

Москва 2020

Задача

Разработать программу определения количества чисел Ферма от 1 до беззнакового двойного машинного слова

Решение

Число Ферма – это числа вида $F_n = 2^{2^n} + 1$, где $n \geq 0$, чтобы определить максимальное число Ферма, не превышающее значение беззнакового двойного машинного слова ($2^{32} - 1$) нужно пройти цикл с постусловием в котором будет считаться число Ферма под номером i после чего это число будет проверяться на переполнение и если наш unsigned dword переполнился, то число Ферма под номером i превысило максимальное значение беззнакового двойного слова, цикл завершается и на экран выводится $i - 1$.

Реализуем этот алгоритм на языке программирования C++ для того, чтобы убедиться, что алгоритм работает верно. Учтем, что беззнаковое двойное машинное слово – это unsigned int в C++.

Текст программы приведен ниже:

```
#include <iostream>
#include <cmath>

int main() {
    unsigned int maxValue = 4294967295;
    int i = 0;
    unsigned int number = 0;
    do
    {
        number = pow(2, pow(2, i)) + 1;
        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 $2^{32} - 1$. Известно, что ни одно число Ферма не равно $2^{32} - 1$, а значит без каких либо проблем можно сравнивать number с 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.

```
int printf(const char* format,...);
```

Она принимает следующие аргументы:

- `char*format` – указатель на C-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(uint uExitCode)`, которая завершает работу программы.

Она принимает следующие аргументы:

- `uint uExitCode` – 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 чисел Ферма, удовлетворяющих нашему условию и функция `printf` для вывода полученного результата в консоль. Также в ней содержится функция `getch` считывающая символ и функция `ExitProcess`, завершающая работу программы.

`uint pow(uint num, uint pow)`

Функция принимающая в себя два аргумента:

- `uint number` – возводимое в некоторую степень число
- `uint power` – степень, в которую будет возводиться число `number`

Функция содержит в себе локальные переменные:

- `int i` – счетчик

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

`uint findCountFermaNum(uint maxValue)`

Функция принимающая в себя один аргумент:

- `uint maxValue` – возводимое в некоторую степень число

Функция содержит в себе локальные переменные:

- int i – счетчик
- uint oldVal – предыдущее число Ферма
- uint val – текущее число Ферма

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

uint findFermaNum(int n)

Функция принимающая в себя аргумент:

- int n – номер числа Ферма

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

Текст программы

format PE console

include 'win32a.inc'

entry start

```

;-----
; Студент: Назмутдинов Роман Ренатович
; Группа: БПИ194
; Вариант: 13
; Условие задачи:
; Разработать программу определения количества
; чисел Ферма от 1 до беззнакового двойного
; машинного слова
;-----

```

section '.data' data readable writable

strPrintNum db '%u', 0

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

maxDword dd 4294967295 ;Максимальное значение dword

NULL = 0

section '.code' code readable executable

;_____MAIN_____

start:

push [maxDword] ;Записываем в стек максимум dword

call findCountFermaNum ;Вызываем findCountFermaNum

add esp, 4 ;Удаляем переданные аргументы

push eax ;Записываем в стек

findCountFermaNum(maxDword)

push strCountFermaNum ;Записываем в стек шаблон

```

        call [printf]          ;Выводим найденное количество чисел Ферма
        add esp, 8             ;Удаляем переданные аргументы

        call [getch]           ;Считываем ввод символа

        push NULL              ;Записываем в стек код выхода
        call ExitProcess       ;Завершаем работу программы

;_____MAIN_____

; Описание:
; Возводит число number в степень power
;
; Аргументы:
; num - число, возводимое в какую-то степень
; power - степень в которую будет возведено number
;
; Возвращает
; number возведенное в степень power
;
;_____POW_____
pow:
        ;Пролог функции
        push ecx
        push ebp
        mov ebp, esp
        sub esp, 4             ;Выделяем место для лок. переменных

;-----Локальные-переменные-----
        i      equ  ebp-4             ;Счетчик
;-----Параметры-----
        num     equ  ebp+16           ;Возводимое в степень число
        power    equ  ebp+12          ;Степень
;-----

        mov eax, 1               ;result=1
        mov [i], dword 0          ;i=0
powLoop:
        mov ecx, [i]             ;ecx = i для сравнения
        cmp ecx, [power]         ;Сравниваем i со power
        jge finishPowLoop        ;В случае если i >= power выходим из цикла

        imul eax, dword [num]     ;Умножаем результат на число

        inc dword [i]             ;i++
        jmp powLoop              ;Возвращаемся в начало цикла

finishPowLoop:
        ;Эпилог функции
        mov esp, ebp
        pop ebp
        pop ecx

        ret

;_____POW_____

; Описание:
; Находит все числа Ферма меньше maxValуe и выводит их в консоль.
; в качестве результата возвращается количество найденных чисел Ферма
;
; Аргументы:

```

```

; maxValue - максимальное число с которым будут сравниваться числа Ферма
;
; Возвращает:
; Количество найденных чисел Ферма меньших максимальному значению
; unsigned dword
;
; _____FIND_COUNT_OF_FERMA_NUM_____

    findCountFermaNum:
        ;Пролог функции
        push ecx
        push ebp
        mov  ebp, esp
        sub  esp, 8                ;Выделяем место для лок. переменных

; -----Локальные-переменные-----
        i      equ  ebp-4          ;Счетчик
        oldVal equ  ebp-8          ;Предыдущее значение
        val     equ  ebp-12        ;Нынешнее значение
; -----Параметры-----
        maxVal equ  ebp+12        ;Максимально допустимое значение
; -----

        mov [i], dword 0          ;i = 0
        mov [oldVal], dword 0     ;oldValue = 0

    fermaLoop:
        push dword [i]            ;Записываем в стек i
        call findFermaNum         ;Вызываем findFermaNum
        add  esp, 4               ;Удаляем переданные аргументы

        mov [val], eax            ;val = findFermaNum(i)
        mov ecx, [val]            ;ecx = val
        cmp  ecx, [oldVal]        ;Сравниваем val со oldVal (проверка
переполнения)
        jb  finishFermaLoop      ;Если меньше oldVal, то выходим из цикла

        mov [oldVal], ecx         ;oldVal = val
        inc dword [i]             ;ecx++

        jmp fermaLoop            ;В начало цикла

    finishFermaLoop:
        mov  eax, [i]             ;Записываем результат в eax
        ;Эпилог функции
        mov  esp, ebp
        pop  ebp
        pop  ecx

        ret

; _____FIND_COUNT_OF_FERMA_NUM_____

; Описание:
; Находит число Ферма под номером n
;
; Аргументы:
; n - номер искомого числа Ферма
;
; Возвращает:
; Число Ферма под номером n
;
; _____FIND_FERMA_NUM_____

    findFermaNum:

```

```

;Пролог функции
push ebp
mov  ebp, esp

;-----Параметры-----
;          n          equ  ebp+8          ;Номер искомого числа Ферма
;-----

;Находим 2^n
push 2          ;Записываем в стек 2
push dword [n]  ;Записываем в стек n
call pow        ;Вызываем pow
add  esp, 8      ;Удаляем переданные аргументы
;Находим 2^(2^n)
push 2          ;Записываем в стек 2
push eax        ;Записываем в стек pow(n)
call pow        ;Вызываем pow
add  esp, 8      ;Удаляем переданные аргументы
inc  eax         ;прибавляем к результату 1

;Эпилог функции
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](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>)