НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии
Дисциплина: «Архитектура вычислительных систем»

МИКРОПРОЕКТ №1

Пояснительная записка

Выполнил:
студент группы БПИ198

Лямзин Дмитрий

Москва
2020

Содержание

1. Текст задания

Разработать программу, определяющую количество чисел Люка[2] в диапазоне от 1 до машинного слова.

2. Применяемые расчетные методы

2.1. Теория решения задания

Числа люка - это числа, из которых состоит математическая последовательность[1] натуральных чисел, в которой следующее число равно сумме двух предыдущих. Последовательность задаётся рекуррентной формулой $L_n = L_{n-2} + L_{n-1}$ с начальными значениями $L_0 = 2$, $L_1 = 1$.

2.2. Дополнительный функционал программы

Входные данные в программе отсутствуют. Подсчёт количества чисел Люка производится в подпрограмме без параметров под названием “CountLucasNumbers”. Для работы со стеком используется переменная “tmpStack” (тип данных - двойное слово). Счётчик количества чисел Люка располагается в переменной “countOfNumbers” (тип данных - двойное слово), которая хранится в регистре ecx. Для хранения промежуточных значений L_{n-2} и L_{n-1} используются переменные “previousNumber” и “currentNumber” соответственно, хранящиеся в регистрах bx и ax соответственно (тип данных обеих переменных - машинное слово). Чтобы вычислить следующее число Люка на каждой итерации цикла значение переменной “currentNumber” записывается в регистр dx, после чего происходит прибавление к регистру ax регистра bx с проверкой на переполнение, а значение, лежащее в регистре dx, переносится в bx. Счётчик количества чисел Люка, который перед началом цикла был равен 2 (первые два числа рассматриваются отдельно), увеличивается на 1 на каждой итерации и выводится в консоль после завершения цикла в результате переполнения регистра ax.

3. Тестирование программы

3.1. Корректные значения

Из-за того, что в программе отсутствуют входные данные, все значения можно считать корректными. Для проверки работоспособности программы и правильности нахождения чисел Люка на каждой итерации цикла выводится число Люка, полученное с помощью сложения двух предыдущих найденных чисел. Числа $L_0 = 2$, $L_1 = 1$ выводятся отдельно перед началом цикла (Рис. 1).

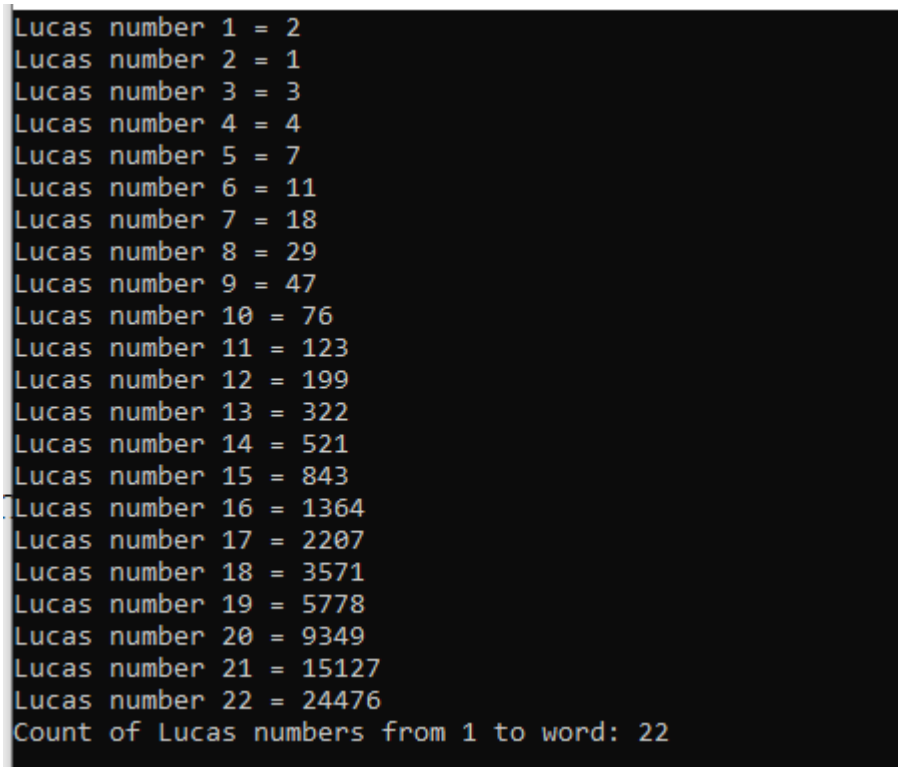

Рис. 1. Результат работы программы

Выведенные значения соответствуют числам из последовательности Люка, а при попытке получения 23-го числа возникает переполнение регистра ax, поскольку машинное слово ограничено размером 2-х байт. Отсюда можно сделать вывод о том, что программа работает корректно.

3.2. Некорректные значения

Проверка некорректных значений входных данных отсутствует из-за отсутствия самих входных данных.

ПРИЛОЖЕНИЕ 1

Список литературы

1. A000032 - OEIS. [Электронный ресурс] // URL: https://oeis.org/A000032, дата обращения: 31.10.2020
2. Lucas Number. [Электронный ресурс] // URL: https://mathworld.wolfram.com/LucasNumber.html, дата обращения: 31.10.2020

ПРИЛОЖЕНИЕ 2

Код программы

```
format PE console
entry start

include 'win32a.inc'

;--------------------------------------------------------------------—
section '.data' data readable writable

;variables for data output
countOut db 'Count of Lucas numbers from %d to word: %d',0
currentNumberOut db 'Lucas number %d = %d', 10, 0
countOfNumbers dd 0
previousNumber dw 0
currentNumber dw 0

tmpStack dd ?


;--------------------------------------------------------------------—
section '.code' code readable executable
start:

;counting the number of Lucas numbers from 1 to machine word
call CountLucasNumbers
```

```
finish:
call [getch]

push 0
call [ExitProcess]
;------------------------------------------------------------------------—
CountLucasNumbers:
    mov [tmpStack], esp

    ;zeroing variables
    xor ecx, ecx
    xor eax, eax
    xor ebx, ebx
    xor edx, edx
    inc ecx

    ;the first number = 2
    mov bx, 2
    mov [previousNumber], bx

    push ebx
    push ecx
    push currentNumberOut
    call[printf]
    add esp, 4
```

```
    mov ecx, 2
    mov [countOfNumbers], ecx

    ;the second number = 1
    mov ax, 1
    mov [currentNumber], ax

    push eax
    push ecx
    push currentNumberOut
    call[printf]
    add esp, 4
    mov ecx, 2

    ;switching to a loop
    jmp countNumbersLoop


countNumbersLoop:

    ;dx=0
    xor dx, dx
    mov ax, [currentNumber]
    mov bx, [previousNumber]

    ; saving current number in dx
```

```
mov dx, ax

; getting the next Lucas number
add ax, bx
mov [currentNumber], ax

;checking ax  for overflow
cmp ax, dx
jle endCount

; continue if there was no overflow
mov bx, dx
push eax
mov ecx, [countOfNumbers]

;incrementing count of numbers
inc ecx
push ecx
push currentNumberOut

; outputing current Lucas number
call[printf]
add esp, 4
mov [previousNumber], bx
mov ecx,[countOfNumbers]
inc ecx
```

```
    mov [countOfNumbers], ecx

    ; going to another iteration
    jmp countNumbersLoop

endCount:
    mov ecx, [countOfNumbers]
    push ecx
    mov ebx,1
    push ebx
    push countOut

    ; outputing final count of Lucas numbers from 1 to machine word
    call[printf]
    add esp, 4
    mov esp, [tmpStack]
    ret
;------------------------------third act - including HeapApi-----------------------—

section '.idata' import data readable
library kernel, 'kernel32.dll',\
msvcrt, 'msvcrt.dll',\
user32,'USER32.DLL'

include 'api\user32.inc'
include 'api\kernel32.inc'
```

```
import kernel,\
ExitProcess, 'ExitProcess',\
HeapCreate,'HeapCreate',\
HeapAlloc,'HeapAlloc'
include 'api\kernel32.inc'
import msvcrt,\
printf, 'printf',\
scanf, 'scanf',\
getch, '_getch'
```