

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

Факультет компьютерных наук
Департамент программной инженерии

**Отчет по домашнему заданию по дисциплине "Архитектура
вычислительных систем"**

Пояснительная Записка

Исполнитель:
Студент группы БПИ195(1)
Ни Эдуард
01.11.2020 г.

Москва 2020

Условие задания

Разработать программу, определяющей максимальное значение параметра числа линейной рекуррентной последовательности $t_n = t_{n-1} + t_{n-2} + t_{n-3} + t_{n-4}$ при $n \geq 4$ со стартовой последовательностью чисел $[0, 0, 0, 1]$, которое не выходит за пределы беззнакового двойного машинного слова.

Решение

В решении задачи следует пройти циклом до тех пор, пока значение t_n помещается в пределы беззнакового двойного машинного слова, т. е. $2^{32}-1$. В данной последовательности значение t_n всегда будет максимальным.

По условию, заданы начальные значения рекуррентной последовательности. Программа будет работать таким образом, что мы берем "окно", в которое помещается 4 элемента, сдвигая его на одну позицию вправо после проведенной итерации. Таким образом, мы всегда сможем вычислять следующее значение последовательности.

Составление программы

Для работы программы будем использовать регистры данных: EAX, EBX, ECX, EDX, – это 32-битные регистры.

Для вывода в консоль значений без знака будет использовать подстановку '%u'.

Текст программы

```
format
PE
CONSOLE
4.0
```

```
entry begin
```

```

include 'win32a.inc'

section '.data' data readable writable

;Initial values
t_1 dd 0
t_2 dd 0
t_3 dd 0
t_4 dd 1

t_n dd 0
cnt dd 4 ;count of sequence's elements

;output strings
slashn db 13, 10, 0
numerating db ' ', 0
el dd '%u', 0
strRes db 'Number of elements of the recurrent sequence: %d.', 0

section '.code' code readable executable

begin:

    mov ecx, 0
    call Rec_meth ;proc of recurrent seq.
    call FinishProgram ;Finishing program

Rec_meth:

lp:
    ;sum of values
    mov eax, [t_1]

```

```

add eax, [t_2]
add eax, [t_3]
add eax, [t_4]

mov dword [t_n], eax

;Shifting t1->t2,....
;rewrite in t_1
mov ecx, [t_1]
mov ebx, [t_2]

mov [t_1], ebx

;rewrite in t_2
mov ecx, [t_2]
mov ebx, [t_3]

mov [t_2], ebx

;rewrite in t_3
mov ecx, [t_3]
mov ebx, [t_4]

mov [t_3], ebx

;rewrite in t_4
mov ecx, [t_4]
mov ebx, [t_n]

mov [t_4], ebx

;Checking for subsequent overflow
mov edx, eax
add edx, [t_2]
jo lpEnd      ;if overflowed

```

```

        sub     edx, [t_2]

        cinvoke printf, e1, [t_n]
        cinvoke printf, slashn
        inc     [cnt] ;increment counter
        jmp     lp

lpEnd:

        cinvoke printf, slashn ;NewLine
        cinvoke printf, strRes, [cnt] ;output cnt

        ret     ;return type void to exit from the proc

FinishProgram:

        call    [getch]
        push    0
        call    [ExitProcess]

        ret

section '.idata' import data readable

library kernel, 'kernel32.dll',\
        msvcrt, 'msvcrt.dll',\
        user32, 'user32.dll'

import msvcrt,\
        printf, 'printf',\
        scanf, 'scanf',\
        getch, '_getch'

```

```
import kernel,\n\nExitProcess, 'ExitProcess'
```

Описание программы

1. Таблица данных.

```
section '.data' data readable writable\n\n;Initial values\n\t_1 dd 0\n\t_2 dd 0\n\t_3 dd 0\n\t_4 dd 1\n\n\t_n dd 0\n\tcnt dd 4 ;count of sequence's elements\n\n;output strings\n\tslashn db 13, 10, 0\n\tnumerating db ' ', 0\n\tel dd '%u', 0\n\tstrRes db 'Number of elements of the recurrent sequence: %d.', 0
```

Рисунок 1 - таблица данных

Задаем начальные значения последовательности t_1, \dots, t_4 , создаем t_n для хранения следующего значения последовательности, cnt – счетчик количества элементов последовательности.

Значения для вывода: slashn – переход на новую строку, el dd '%u' – подстановка для вывода беззнакового значения, strRes – вывод количества элементов последовательности.

2. Сумма элементов и сдвиг

```

;sum of values
mov eax, [t_1]
add eax, [t_2]
add eax, [t_3]
add eax, [t_4]

mov dword [t_n], eax

;Shifting t1->t2,....
;rewrite in t_1
mov ecx, [t_1]
mov ebx, [t_2]

mov [t_1], ebx

;rewrite in t_2
mov ecx, [t_2]
mov ebx, [t_3]

mov [t_2], ebx

;rewrite in t_3
mov ecx, [t_3]
mov ebx, [t_4]

mov [t_3], ebx

;rewrite in t_4
mov ecx, [t_4]
mov ebx, [t_n]

mov [t_4], ebx

```

Рисунок 2 - операции с элементами

Переносим t_1 в регистр EAX, суммируем с t_2 , t_3 , t_4 . Далее сдвигаем все элементы:
 $t_1=t_2$, $t_2=t_3$, $t_3=t_4$, $t_4=[EAX]$

3. Проверка на минимальное переполнение

```

;Checking for subsequent overflow
mov edx, eax
add edx, [t_2]
jo lpEnd ;if overflowed
sub edx, [t_2]

cinvoke printf, el, [t_n]
cinvoke printf, slashn
inc [cnt] ;increment counter
jmp lp

```

Рисунок 3 - обработка переполнения

Происходит проверка на минимальное переполнение за беззнаковый тип dword. jo – оператор условного перехода при переполнении. При переполнении цикл завершается, программа идет к метке конца процедуры, иначе выводится значение t_n .

Тестирование программы

```
C:\Users\22508\Desktop\All about deadlines\fasmw17325\Новая папка\miniproj_seq.EXE
1
2
4
8
15
29
56
108
208
401
773
1490
2872
5536
10671
20569
39648
76424
147312
283953
547337
1055026
2033628
3919944
7555935
14564533
28074040
54114452
104308960
201061985
387559437
747044834
1439975216
2775641472
Number of elements of the recurrent sequence: 38.
```

Рисунок 4- результат выполнения

Как видно из (рис. 4), программа выводит максимальное значение на итерации и прекращается при переполнении. В конце происходит вывод количества элементов последовательности до двойного машинного слова без знака.

Используемые источники

1. Ravesli, Ассемблер. Сегменты памяти и регистры [Электронный ресурс] //URL: <https://ravesli.com/assembler-segmenty-pamyati-i-registry/#toc-1>, свободный. (дата обращения 29.11.2020, режим доступа: свободный).
2. CyberForum.ru, Ассемблер. Синтаксис ассемблера [Электронный ресурс] //URL: <https://www.cyberforum.ru/assembler/thread1005284-page14.html>, свободный. (дата обращения 29.11.2020, режим доступа: свободный).
3. flat assembler, Ассемблер. Documentation and tutorials [Электронный ресурс] //URL: <https://flatassembler.net/docs.php?article=manual>, свободный. (дата обращения 29.11.2020, режим доступа: свободный).
4. YouTube. FASM. Данные в FASM. Стек, память, регистры. Биты, байты, слова [Электронный ресурс] //URL: <https://www.youtube.com/watch?v=V-97htBBtMI>, свободный. (дата обращения 29.11.2020, режим доступа: свободный).