

Федеральное государственное автономное образовательное
учреждение высшего образования
**"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
"ВЫСШАЯ ШКОЛА ЭКОНОМИКИ"**

Факультет Компьютерных наук
Департамент программной инженерии

Пояснительная записка

к домашнему заданию по дисциплине
“Архитектура вычислительных систем”

Симоновича Ивана Сергеевича

БПИ 193-2 19 Вариант

Москва 2020

Формулировка задания:

Разработать программу, определяющую число непересекающихся повторов троек битов '011' в заданном машинном слове

Входные данные:

На вход поступает 32-битное число в пределах от 0 до 4 294 967 295. Если ввести число отрицательное, то количество повторов будет считаться в дополнительном коде, а если ввести выходящее за границы, то будут считаться только те биты, что попали в пределы слова.

Если введено не целое число, то выведется сообщение о его некорректности в консоль.

Выходные данные:

В консоль выводится двоичное представление числа и количество непересекающихся повторов троек.

Решение задания:

Введенное число хранится в переменной i ; число $3_{10}=011_2$ хранится в переменной `targetSequence`, а его длина в двоичной системе в `targetSequenceLength`.

После успешного считывания числа при вводе пользователя запускается метод перевода в двоичную систему `systemTwo`, где в цикле выводятся все 32 бита числа i , после запускается метод `countTriples`, который не принимает значений и ничего не возвращает, но в котором происходит вывод информации о количестве непересекающихся повторов троек битов '011' и их подсчет соответственно. В этом методе в цикле с помощью побитовых сдвигов берется из введенного числа по тройке битов, начиная с левого края двоичного представления числа:

0110011110011011₂ -например, для данного числа в двоичной системе в самом начале цикла считаются 3 самых первых бита слева '011' (выделены красным).

Эта тройка битов сравнивается с числом $3_{10}=011_2$, если все биты в них попарно совпадают, то выполняется равенство:

$\text{temp} \oplus 011_2 = 0$ (где temp – это тройка битов при конкретной итерации цикла, \oplus - xor, логическое или)

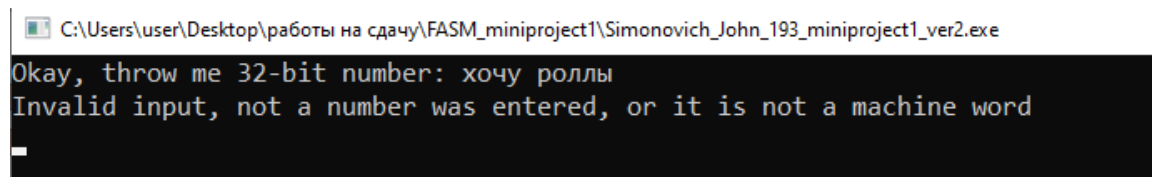
Если это условие выполнилось, то счетчик количества последовательностей '011' увеличивается на единицу и итератор цикла сдвигается на 3 шага (двигаемся на 3 шага вправо, если смотреть на число в 2 системе), так как мы уже проверили все 3 бита. Если же условие не выполнилось и 3 бита не совпадают, то мы двигаемся на один шаг (вправо) и не увеличиваем счетчик.

Цикл будет выполняться, пока итератор не будет отрицательным, то есть до нуля включительно. В начале цикла итератор равен максимально возможному кол-ву разрядов вводимого числа, которое равно 32 битам. В конце метода countTriples в консоль выводится информация о найденном количестве троек.

Тестирование программы:

Случаи некорректного ввода:

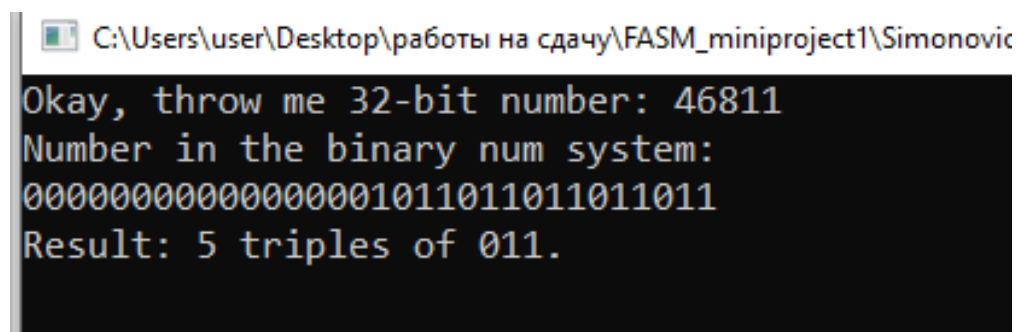
1. Не введено целое число



```
C:\Users\user\Desktop\работы на сдачу\FASM_miniproject1\Simonovich_John_193_miniproject1_ver2.exe
Okay, throw me 32-bit number: хочу роллы
Invalid input, not a number was entered, or it is not a machine word
```

Примеры корректного ввода:

1. 5 троек '011' при $46811_{10}=1011011011011011_2$



```
C:\Users\user\Desktop\работы на сдачу\FASM_miniproject1\Simonovich_John_193_miniproject1_ver2.exe
Okay, throw me 32-bit number: 46811
Number in the binary num system:
0000000000000000001011011011011011
Result: 5 triples of 011.
```

2. 4 тройки '011' при $26523_{10}=0110011110011011_2$

C:\Users\user\Desktop\работы на сдачу\FASM_miniproject1\Simonovich_John_

```
Okay, throw me 32-bit number: 26523
Number in the binary num system:
000000000000000000110011110011011
Result: 4 triples of 011.
```

3. 1 тройка '011' при $65533_{10}=011111111111111101_2$

Выбрать C:\Users\user\Desktop\работы на сдачу\FASM_miniproject1\Sim

```
Okay, throw me 32-bit number: 65533
Number in the binary num system:
000000000000000000111111111111101
Result: 1 triples of 011.
```

4. 2 тройки '011' при $27_{10}=011011_2$

C:\Users\user\Desktop\работы на сдачу\FASM_minipro

```
Okay, throw me 32-bit number: 27
Number in the binary num system:
000000000000000000000000000000011011
Result: 2 triples of 011.
```

5. 1 тройка '011' при $3_{10}=011_2$

C:\Users\user\Desktop\работы на сдачу\FASM_miniproj

```
Okay, throw me 32-bit number: 3
Number in the binary num system:
0000000000000000000000000000000011
Result: 1 triples of 011.
```

6. Для числа 5 нет троек

C:\Users\user\Desktop\работы на сдачу\FASM_miniproject1\Sir

[illegible][illegible]

Выбрать C:\Users\user\Desktop\работы на сдачу\FASM

```
Okay, throw me 32-bit number: -13
Number in the binary num system:
11111111111111111111111111110011
Result: 1 triples of 011.
```

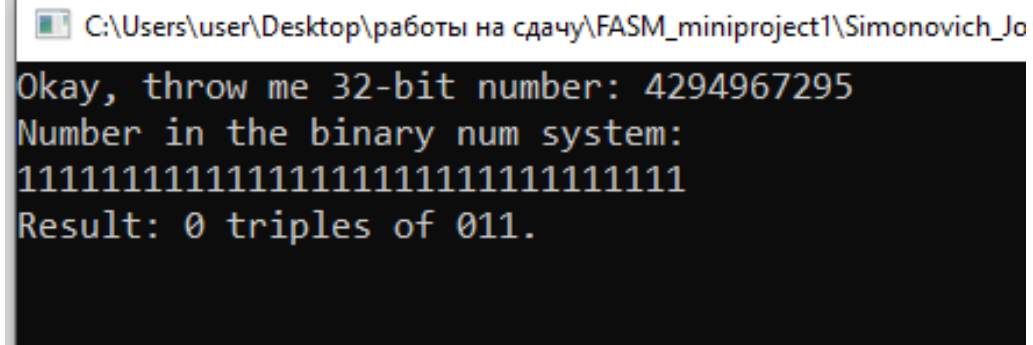
7. Максимальный результат 10 троек '011' при $460175067_{10}=00011011011011011011011011011011_2$

Можно поставить еще 1 в начало, но большего кол-ва троек добиться не получится

C:\Users\user\Desktop\работы на сдачу\FASM miniproject1\Simonovich_Joh

```
Okay, throw me 32-bit number: 460175067
Number in the binary num system:
00011011011011011011011011011011
Result: 10 triples of 011.
```

8. 0 троек '011' при $4294967295_{10} = 11111111111111111111111111111111_2$



```
C:\Users\user\Desktop\работы на сдачу\FASM_miniproject1\Simonovich_Jo
Okay, throw me 32-bit number: 4294967295
Number in the binary num system:
11111111111111111111111111111111
Result: 0 triples of 011.
```

Список используемых источников

1. FasmWorld Программирование на ассемблере FASM для начинающих и не только: <https://fasmworld.ru> (дата обращения: 27.10.2020).
2. FLAT ASSEMBLER 1.64 - МАНУАЛ ПРОГРАММЕРА:
<http://flatassembler.narod.ru/fasm.htm> (дата обращения: 27.10.2020).
3. Уроки Ассемблера-Ravesli, типы прыжков:
<https://ravesli.com/assembler-usloviya/> (дата обращения: 19.10.2020).
4. Самоучитель по Assembler- описание команд:
<https://i-assembler.ru/25/Text/Command.htm> (дата обращения: 19.10.2020).

Текст программы:

;Симонович Иван БПИ 193-2 Вариант 19

; Разработать программу, определяющую число непересекающихся повторов

; троек битов '011' в заданном машинном слове

format PE console

include 'win32a.inc'

entry start

section '.data' data readable writable

msgNumberInput db 'Okay, throw me 32-bit number: ', 0

```
msgNumberOutOfRange    db 'Invalid input, not a number was entered, or it is not a machine word',  
10, 0
```

```
msgResult               db 'Result: %d triples of 011.', 10, 0
```

```
msgTS                   db 'Number in the binary num system:', 10, 0
```

```
msgEnd                  db ", 10, 0
```

```
strScanInt              db '%d', 0 ;Формат ввода целого числа
```

```
targetSequence          dd 3 ; 011 в двоичной системе счисления
```

```
targetSequenceLength    dd 3 ; 3
```

```
i                       dd ? ; считанное число
```

```
tmp                     dd ? ; временная переменная
```

```
section '.code' code readable executable
```

```
start:
```

```
push msgNumberInput     ;добавляем информацию о вводе в стек
```

```
call [printf]           ;Выводим информация о вводе
```

```
add esp, 4 * 1          ;очистка стека
```

```
push i                  ;добавляем в стек переменную, куда будет записано число
```

```
push strScanInt ;добавляем в стек информацию о вводе
```

```
call [scanf] ;вызываем метод считывания числа и записываем в i
```

```
;push [i] ;добавляем в стек переменную, куда будет записано число
```

```
;push strScanInt ;добавляем в стек информацию о вводе
```

```
;call [printf]
```

```
;проверка что введено целое число в строке
```

```
cmp eax, 0
```

```
je @f
```

```
add esp, 4 * 2 ;очистка стека
```

```
push [i] ;добавляем в стек значение переменной i, введенное число
call systemTwo ;вызываем метод перевода в двоичную систему
add esp, 4 * 1 ;очистка стека
```

```
push [i] ;добавляем в стек значение переменной i, введенное число
call countTriples ;вызываем метод подсчета троек битов '011'
add esp, 4 * 1 ;очистка стека
```

```
jmp exit ; метод выхода
```

@@: ;метка запускается, если был введен неверный размер

```
push msgNumberOutOfRange ;информация о выходе за рамки 32-битного числа
call [printf] ;вывод информации, что введено некорректное число
add esp, 4 * 1 ;очистка стека
```

```
jmp exit;метод выхода
```

systemTwo:: Вывод представления числа в двоичной системе счисления

```
push msgTS ;Вывод строки в консоль
call [printf]
add esp, 4 * 1
```

```
xor ebx, ebx ;обнуляем регистры
xor ecx, ecx
xor eax, eax
mov ecx, 32 ;записываем в количество циклов максимальное кол-во бит числа
sub ecx, 1;
```

simpleLoop::цикл

```
mov eax, [esp + 4] ; перемещаем считанное число [i] в eax
```


bt eax,ecx ; проверяем бит в eax на позиции ecx

jc @f ;перейти к @@, если проверяемый бит равен 1

mov [tmp], ecx; запоминаем ecx

push 0;

push strScanInt ;добавляем в стек информацию о вводе

call [printf]

add esp, 4 * 2 ;очистка стека

mov ecx, [tmp]; возвращаем значение ecx

jmp next;переходим к метке с итерацией

@@:

mov [tmp], ecx; запоминаем ecx

push 1

push strScanInt ;добавляем в стек информацию о вводе

call [printf]

add esp, 4 * 2 ;очистка стека

mov ecx, [tmp]; возвращаем значение ecx

next;;переход к следующей итерации

dec ecx; уменьшаем количество циклов на один

jns simpleLoop ;если значение без знака-, то опять запускаем цикл countTriplesLoop

push msgEnd ;Вывод строки переноса в консоль

```
call [printf]
```

```
add esp, 4 * 1
```

```
ret;завершаем метод
```

countTriples;;метод подсчета троек не возвращает и не принимает никаких значений

```
xor ebx, ebx ; обнуляем ebx
```

```
mov ecx, 32 ;записываем в количество циклов максимальное кол-во бит числа
```

```
sub ecx, [targetSequenceLength];отнимаем от общей длины длину последовательности'011'
```

countTriplesLoop;;цикл подсчета непересекающихся троек '011'

```
mov eax, [esp + 4] ; перемещаем считанное число [i] в ax
```

```
push ecx ;добавляем количество итераций цикла в стек
```

```
mov edx, 32 ;записываем в текущую длину максимальное кол-во бит числа
```

```
sub edx, ecx ;отнимаем от текущей длины количество циклов
```

```
sub edx, [targetSequenceLength];отнимаем от текущей длины размер  
последовательности'011'
```

```
mov ecx, edx ;обновляем количество циклов на текущую длину
```

;побитовый сдвиг влево-вправо, чтобы убрать старшие биты перед проверяемой тройкой
битов

```
shl eax, cl
```

```
shr eax, cl
```

```
pop ecx ;достаем количество итераций обратно
```

```
shr eax, cl ;побитовый сдвиг вправо, чтобы убрать младшие биты до проверяемой тройки
```

```
mov edx, [targetSequence];записываем в dx число 3='011'
```

```

xor    eax, edx;если все биты попарно равны, то результат xor будет=0,
;А это значит, что мы нашли последовательность = '011'
jnz    @@ ;если не ноль, то переходим к следующей метке @@ (которая ниже по коду)
inc    ebx ;увеличиваем счетчик троек
sub    ecx, [targetSequenceLength] ;отнимаем длину последовательности=3 от длины
jns    countTriplesLoop ;если значение без знака-, то опять запускаем цикл countTriplesLoop
@@: ;запускается если последовательности 011 нет,тогда двигаемся на 1 бит
dec    ecx; уменьшаем количество циклов на один
jns    countTriplesLoop ;если значение без знака-, то опять запускаем цикл countTriplesLoop

push    ebx ;добавляем счетчик троек в стек
push    msgResult ;добавляем информацию про результат
call    [printf] ;выводим информацию и счетчик
add     esp, 4 * 2 ;очистка стека

ret;завершаем метод

```

exit:

```

call [getch];ожидание ввода от пользователя
push    0
call    [ExitProcess];завершение программы

```

section '.idata' data readable import

```

library kernel32, 'kernel32.dll', \
    msvcrt, 'msvcrt.dll'
import kernel32, ExitProcess, 'ExitProcess'
import msvcrt, printf, 'printf', \
    scanf, 'scanf', \
    getch, '_getch'

```