

## **Пояснительная записка**

НИУ ВШЭ, ФКН

Образовательная программа “Программная инженерия”, 2 курс

Курс «Архитектура вычислительных систем»

Программирование на языке ассемблера. Микропроект.

Стегнина Варвара Валерьевна

Группа БПИ196

2020 г.

## Задание

**Вариант 21.** Разработать программу, определяющую число непересекающихся повторов троек битов '110' в заданном двойном машинном слове.

### Описание метода решения

Двойное машинное слово содержит 32 бита. Проверим все возможные расположения тройки битов на наличие там последовательности '110' (данные последовательности не могут пересекаться, об этом можно не беспокоиться). Начнем с последних трёх битов. Для проверки скопируем наше слово  $w1$  в переменную  $w2$  и видоизменим её: установим в  $w2$  в последние три бита последовательность '110'. Это можно сделать таким образом:

- 1)  $w2 = w2 \mid 000\dots110 = w2 \mid 6$
- 2)  $w2 = w2 \& 111\dots110 = w2 \& -2$

Теперь  $w2$  совпадает с  $w1$  по первым 29 битам, а в последних трёх установлена последовательность '110'. Теперь применим операцию xor:  $w2 = w2 \wedge w1$ . Если в  $w1$  в конце была последовательность '110', то  $w2$  равно  $w1$  и в результате получится 0. Иначе  $w2$  не будет равно 0. Теперь увеличим счетчик встретившихся последовательностей, если  $w2$  равно 0. Для проверки следующей тройки битов просто сдвинем  $w1$  побитово вправо и повторим действия. Так придется повторить 30 раз, так как всего последовательных троек битов в двойном слове  $32 - 3 + 1 = 30$ .

### Используемые переменные

db strDWord – строка с информацией для ввода машинного слова

db strScanInt – строка для считывания введенного числа

db strInt – строка для вывода числа

db strOutput – строка с информацией о выводе

dd double\_word – двойное слово, вводится пользователем,  $w1$  в описании метода решения

dd tmp\_word – изменяемая копия double\_word,  $w2$  в описании метода решения

### Используемые процедуры

- 1) Input

Используются переменные: strDWord, strScanInt

Результат записывается в double\_word

Получает от пользователя двойное слово.

- 2) NumberOf110

Используются переменные: double\_word, tmp\_word

Результат (число последовательностей) записывается в регистр edx.

Считает число последовательностей битов '110' в заданном двойном слове.

### Входные и выходные данные


На вход поступает целое число в пределах: -2 147 483 648 ... +2 147 483 647. Для отрицательных чисел последовательность ищется в дополнительном коде. Для чисел, выходящих за границы двойного слова, учитывается лишь та часть числа, которая умещается в пределы двойного слова.

Выходные данные: программа выведет число от 0 до 10 – число последовательностей '110'.

## Тестирование

1)  $6 = 110_2$


1 вхождение последовательности

 C:\Users\xiaom\assemb\microproject1\mp1.EXE

```
double word? 6
Number of bit sequences 110:
1
```

2)  $3 = 011_2$


0 вхождений последовательности

 C:\Users\xiaom\assemb\microproject1\mp1.EXE

```
double word? 3
Number of bit sequences 110:
0
```

3)  $-23 = 111111111111111111111111111101001_2$


1 вхождение последовательности

 C:\Users\xiaom\assemb\microproject1\mp1.EXE

```
double word? -23
Number of bit sequences 110:
1
```

4)  $1431655765 = 01010101010101010101010101010101_2$


0 вхождений последовательности

 C:\Users\xiaom\assemb\microproject1\mp1.EXE

```
double word? 1431655765
Number of bit sequences 110:
0
```

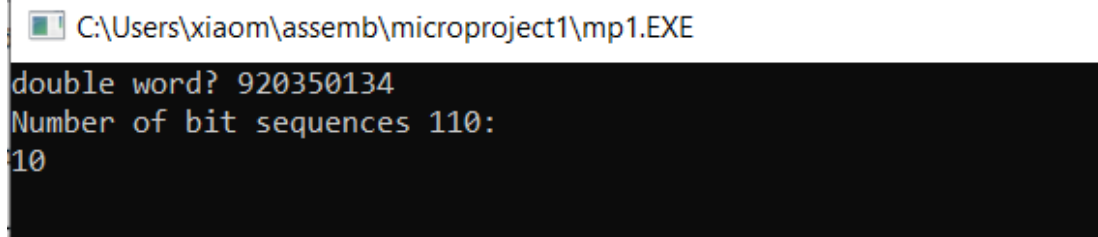
5)  $-2396746 = 11111111110110110110110110110110_2$

8 вхождений последовательности

 C:\Users\xiaom\assemb\microproject1\mp1.EXE

```
double word? -2396746
Number of bit sequences 110:
8
```

- 6)  $920350134 = 00110110110110110110110110110110_2$   
10 вхождений последовательности (максимальное возможное)



```
C:\Users\xiaom\assemb\microproject1\mp1.EXE
double word? 920350134
Number of bit sequences 110:
10
```

## Список источников

1. <http://natalia.appmat.ru/c&c++/assembler.html> - справочные материалы по assembler

## Текст программы

format PE console

entry start

include 'win32a.inc'

;-----

section '.data' data readable writable

strDWord db 'double word? ', 0

strScanInt db '%d', 0

strInt db '%d', 10, 0

strOutput db 'Number of bit sequences 110:', 10, 0

double\_word dd 0

tmp\_word dd 0

;-----

section '.code' code readable executable

start:

; 1) input

call Input

; 2) get number of sequences

call NumberOf110

; 3) output

push edx ;edx = number of '110'

push strOutput

call [printf]

add esp,4

pop edx

push edx

push strInt

call [printf]

add esp,8

finish:

call [getch]

push 0

call [ExitProcess]

-----

Input:

push strDWord

call [printf]

add esp, 4

push double\_word

push strScanInt

call [scanf]

add esp, 8

ret

-----

NumberOf110:

```
mov ecx, 30          ;ecx = number of bits in dd - length of sequence + 1 = 32 - 3 + 1
xor edx, edx         ;edx = 0
mov eax, [double_word]
mov [tmp_word], eax  ;tmp_word = double_word
```

bitLoop:

```
mov ebx, [tmp_word]  ;ebx = tmp_word
or ebx, 6            ;set 1 in the second and third bits from the end (6 = 000..0110)
and ebx, -2          ;set 0 in the last bit (-2 = 1111...110)
xor ebx, [tmp_word]  ;ebx ^ tmp_word (tmp_word = ....., ebx = ...110)
mov eax, [tmp_word]
shr eax, 1
mov [tmp_word], eax  ; tmp_word >> 1
cmp ebx, 0           ; ebx == 0 if tmp_word = ...110
jne continLoop       ; ebx != 0
inc edx              ; edx++
```

continLoop:

```
loop bitLoop
```

endNumberOf110:

```
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'
```