НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп’ютерних систем

Лабораторна робота №4

з дисципліни

«Системне програмування»

Тема:

«Приклад організації взаємодії програми мовою C++ і програми на Асемблері»

Виконав студент 2 курсу

ФПМ групи КВ-71

Рибак Ю. О.

Перевірив:

Київ – 2018

**Варіант** : 19

Розробити функцію bool FBig2Sub(byte\* M1, byte\* M2, short len), де M1,M2 –

надвеликі цілі додатні числа (байтові масиви довжиною len). Операція – М1=М1-М2.

Функції Fbig2Sub присвоюється значення False в разі наявності позики і True при її

відсутності. Повинні використовуватись команди для 32-розрядних даних. Якщо

значення len не кратно 4, то для віднімання останніх байт використовувати команди для

8-розрядних даних.

**Текст програми:**

**С++:**

#include <stdio.h>

#define n 24

typedef unsigned char byte;

// функція віднімання двох наделиких чисел

// m1 = m1 - m2

// False - наявність позики

// True - відсутність позики

extern "C" bool FBig2Sub(byte \*m1, byte \*m2, short len);

void output(byte \*m, int len)

{

// output array in hexidemical format

byte\* last = m + len - 1;

int i = len;

while (i != 0) {

if ((i & 3) == 0)

putchar(' ');

unsigned char b;

b = \*last >> 4;

putchar((b < 10) ? ('0' + b) : ('A' + b - 10));

b = \*last & 0x0F;

putchar((b < 10) ? ('0' + b) : ('A' + b - 10));

last--;

i--;

}

putchar('\n');

}

int main()

{

byte m1[n], m2[n];

for (int i = 0; i < n; ++i) {

m1[i] = 0x11;

m2[i] = 0xFF;

}

printf("m1 = ");

output(m1, n);

printf("m2 = ");

output(m2, n);

bool pozika = FBig2Sub(m1, m2, n);

printf("m1 = ");

output(m1, n);

puts(pozika ? "Pozyki ne vidbulos" : "Pozyka vidbulasya");

getchar();

return 0;

}

**ASM:**

.model flat, C

.code

FBig2Sub proc

; save ebp for previsious function

push ebp

mov ebp, esp

mov esi, DWORD PTR[ebp + 8] ; array M1 (8 = ret.adress + ebp)

mov edi, DWORD PTR[ebp + 12] ; array M2 (12 = ret.adress + ebp + m1)

mov ax, WORD PTR [ebp + 16] ; length

; clear flag and save it in stack

clc

pushf

; test length, if it is less then 4

cmp ax, 4

jl loop\_8

; sub. of 4 bytes data

loop\_32:

; use 8 bit operation if ax < 4

cmp ax, 4

jl loop\_8

mov edx, DWORD PTR [edi]

popf

sbb [esi], edx

pushf

sub ax, 4

add edi, 4

add esi, 4

jmp loop\_32

loop\_8:

; loop while ax != 0

test ax, ax

jz end\_loop\_8

mov dl, [edi]

popf

sbb [esi], dl

pushf

inc esi

inc edi

dec ax

jmp loop\_8

end\_loop\_8:

; result returns using accamulator register (eax)

xor eax, eax ; eax = 0

popf

setnc al ; set al = 1 if cf == 0 or al = 0 if cf == 1

pop ebp

ret

FBig2Sub endp

end

**LST:**

Microsoft (R) Macro Assembler Version 14.16.27027.1 03/18/19 23:52:48

FBig2Sub.asm Page 1 - 1

.model flat, C

00000000 .code

00000000 FBig2Sub proc

; save ebp for previsious function

00000000 55 push ebp

00000001 8B EC mov ebp, esp

00000003 8B 75 08 mov esi, DWORD PTR[ebp + 8] ; array M1 (8 = ret.adress + ebp)

00000006 8B 7D 0C mov edi, DWORD PTR[ebp + 12] ; array M2 (12 = ret.adress + ebp + m1)

00000009 66| 8B 45 10 mov ax, WORD PTR [ebp + 16] ; length

; clear flag and save it in stack

0000000D F8 clc

0000000E 66| 9C pushf

; test length, if it is less then 4

00000010 66| 83 F8 04 cmp ax, 4

00000014 7C 1A jl loop\_8

; sub. of 4 bytes data

00000016 loop\_32:

; use 8 bit operation if ax < 4

00000016 66| 83 F8 04 cmp ax, 4

0000001A 7C 14 jl loop\_8

0000001C 8B 17 mov edx, DWORD PTR [edi]

0000001E 66| 9D popf

00000020 19 16 sbb [esi], edx

00000022 66| 9C pushf

00000024 66| 83 E8 04 sub ax, 4

00000028 83 C7 04 add edi, 4

0000002B 83 C6 04 add esi, 4

0000002E EB E6 jmp loop\_32

00000030 loop\_8:

; loop while ax != 0

00000030 66| 85 C0 test ax, ax

00000033 74 0E jz end\_loop\_8

00000035 8A 17 mov dl, [edi]

00000037 66| 9D popf

00000039 18 16 sbb [esi], dl

0000003B 66| 9C pushf

0000003D 46 inc esi

0000003E 47 inc edi

0000003F 66| 48 dec ax

00000041 EB ED jmp loop\_8

00000043 end\_loop\_8:

; result returns using accamulator register (eax)

00000043 33 C0 xor eax, eax ; eax = 0

00000045 66| 9D popf

00000047 0F 93 C0 setnc al ; set al = 1 if cf == 0 or al = 0 if cf == 1

0000004A 5D pop ebp

0000004B C3 ret

0000004C FBig2Sub endp

end

Microsoft (R) Macro Assembler Version 14.16.27027.1 03/18/19 23:52:48

FBig2Sub.asm Symbols 2 - 1

Segments and Groups:

N a m e Size Length Align Combine Class

FLAT . . . . . . . . . . . . . . GROUP

\_DATA . . . . . . . . . . . . . 32 Bit 00000000 Para Public 'DATA'

\_TEXT . . . . . . . . . . . . . 32 Bit 0000004C Para Public 'CODE'

Procedures, parameters, and locals:

N a m e Type Value Attr

FBig2Sub . . . . . . . . . . . . P Near 00000000 \_TEXT Length= 0000004C Public C

loop\_32 . . . . . . . . . . . L Near 00000016 \_TEXT

loop\_8 . . . . . . . . . . . . L Near 00000030 \_TEXT

end\_loop\_8 . . . . . . . . . . L Near 00000043 \_TEXT

Symbols:

N a m e Type Value Attr

@CodeSize . . . . . . . . . . . Number 00000000h

@DataSize . . . . . . . . . . . Number 00000000h

@Interface . . . . . . . . . . . Number 00000001h

@Model . . . . . . . . . . . . . Number 00000007h

@code . . . . . . . . . . . . . Text \_TEXT

@data . . . . . . . . . . . . . Text FLAT

@fardata? . . . . . . . . . . . Text FLAT

@fardata . . . . . . . . . . . . Text FLAT

@stack . . . . . . . . . . . . . Text FLAT

0 Warnings

0 Errors