**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №3**

**Дисциплина: Архитектура вычислительных систем**

**Тема: Арифметические команды центрального процессора**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: Осипов Олег Васильевич

**Белгород 2019**

**Цель работы:** изучение арифметических команд центрального процессора для работы с целыми числами.

**Задания для выполнения к работе**

1. Написать программу для вычисления значения арифметического выражения согласно варианту задания. Все переменные, используемые в программе, требуется использовать как знаковые и расширять до размерности двойного слова. Результат должен быть записан в регистр EAX. Если результат содержит остаток от деления, оставить его в регистре EDX. Подобрать набор тестовых данных (не менее 3).
2. Написать программу для сложения или вычитания целых беззнаковых чисел большой размерности (размерность и операция зависят от варианта задания). Младшие байты при этом хранить по младшему адресу. Подобрать наборы тестовых данных (не менее 3). Для выполнения этого задания изучить теоретический материал главы «Вычитание и сложение операндов большой размерности», начиная со страницы 176 учебника Юрова «Assembler».

**Вариант 13**

|  |  |  |  |
| --- | --- | --- | --- |
| 13 |  | *a*, *b*, *d* – byte  *x* – word | сложение  14 байт |

**Ход выполнения работы**

.386

.MODEL FLAT, STDCALL

OPTION CASEMAP: NONE

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

.DATA

a db 0 ; -128 ; 127

b db 0 ; -128 ; 127

d db 0 ; -128 ; 127

x dw 0 ; -32768 ; 32767

save db 8 dup(0)

.CODE

START:

XOR EAX, EAX

XOR EBX, EBX

XOR ECX, ECX

XOR EDX, EDX

MOVSX ECX, a ; запись с расширением в ECX байта из a

MOVSX EAX, x ; запись с расширением в EAX слова из x

IMUL ECX ; умножение EAX на ECX, результат в EAX

MOV EBX, EAX ; запись данных из EAX в EBX

MOVSX ECX, b ; запись с расширением в ECX байта из b

MOVSX EAX, x ; запись с расширением в EAX слова из x

IMUL EAX ; умножение EAX на EAX, результат в EAX

IMUL ECX ; умножение EAX на ECX, результат в EDX:EAX ADD EAX, EBX ; сложение EAX и EBX, результат в EAX

ADC EDX, 0 ; сложение EDX и переполнения

MOV dword ptr save(0), EDX ; перенос данных из EDX в память save(0)

MOV dword ptr save(4), EAX ; перенос данных из EAX в память save(4)

MOVSX ECX, d ; запись с расширением в ECX байта из d

MOVSX EAX, x ; запись с расширением в EAX слова из x

IMUL ECX ; умножение EAX на ECX, результат в EAX MOV ECX, EAX ; перенос данных из EAX в ECX

MOVSX EAX, x ; запись с расширением в EAX слова из x

IMUL EAX ; умножение EAX на EAX, результат в EAX

IMUL ECX ; умножение EAX на ECX, результат в EDX:EAX ADD EAX, dword ptr save(4) ; сложение EAX и участка памяти save(4)

ADC EDX, 0 ; сложение EDX и переполнения

ADD EDX, dword ptr save(0) ; сложение EDX и участка памяти save(0)

SUB EAX, 14\*14\*14 ; разность EAX выражения, результат в EAX

CMP EAX, 0

JG exit ; выход, если EAX > 0

CMP EDX, 0

JE exit ; выход, если EDX <> 0

CDQ ; конечный результат записан в расширенном регистре EDX:EAX

PUSH NULL

CALL ExitProcess ; завершение программы

END START

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *a* | *b* | *d* | *x* | EAX:EDX |
| 127 | 127 | 127 | 32767 | **FDFC0807EF4C9 = 4 486 142 553 178 569** |
| -128 | -128 | -128 | -32768 | **FFFE0003FF548 = 4 503 426 192 608 584** |
| 8 | -5 | 34 | -18 | **FFFCE7D4=**  **-202795 (в доп. коде** |

.386

.MODEL FLAT, STDCALL

OPTION CASEMAP: NONE

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

.DATA

one db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

two db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

res db 14 dup(0)

.CODE

START:

MOV EAX, dword ptr one[0]

ADD EAX, dword ptr two[0]

MOV dword ptr res[0], EAX

MOV EAX, dword ptr one[4]

ADC EAX, dword ptr two[4]

MOV dword ptr res[4], EAX

MOV EAX, dword ptr one[8]

ADC EAX, dword ptr two[8]

MOV dword ptr res[8], EAX

MOV AX, word ptr one[12]

ADC AX, word ptr two[12]

MOV word ptr res[12], AX

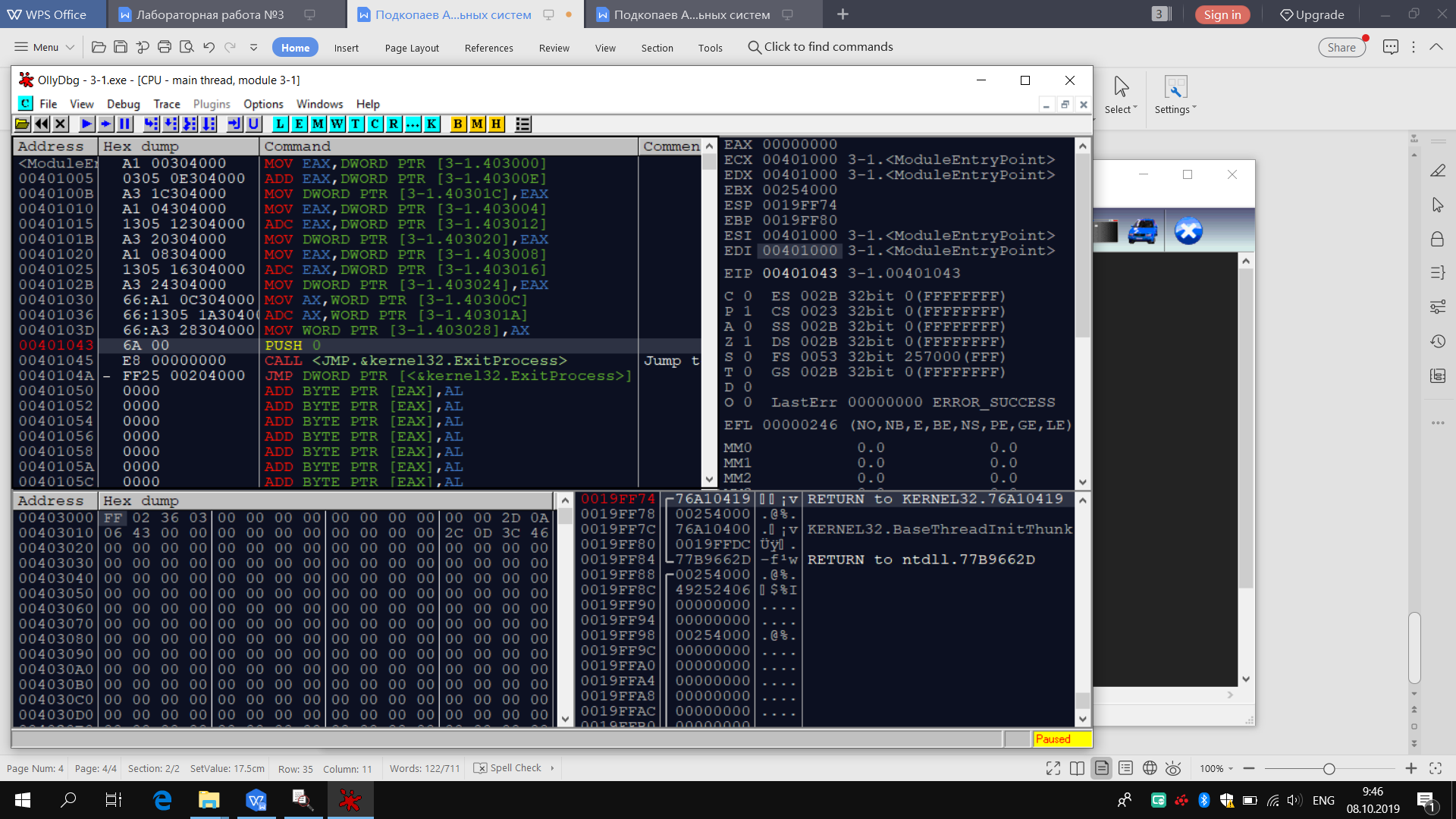
PUSH NULL

CALL ExitProcess

END START

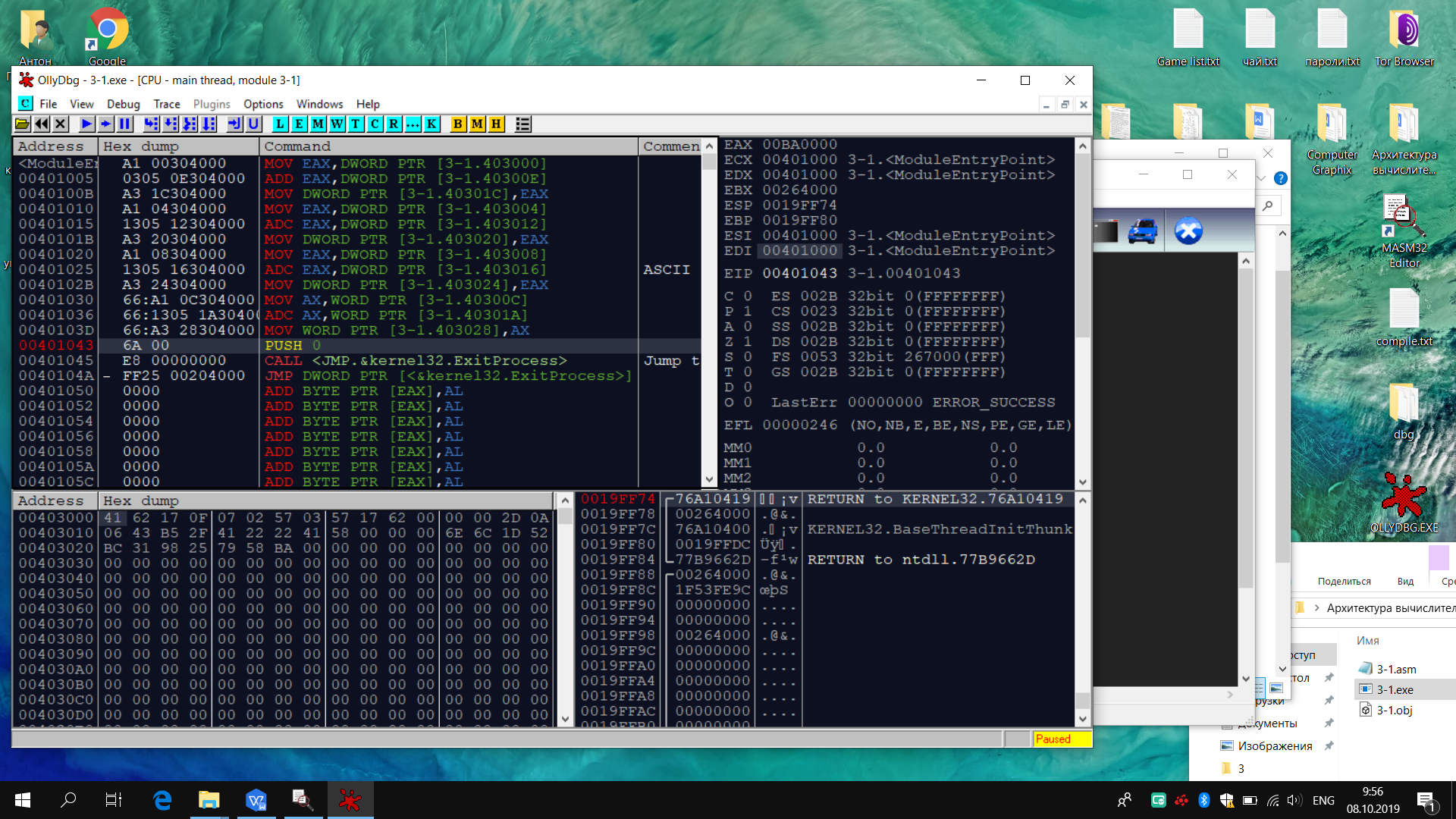
one db 0FFh, 2, 54, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

two db 45, 0Ah, 6, 67, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0



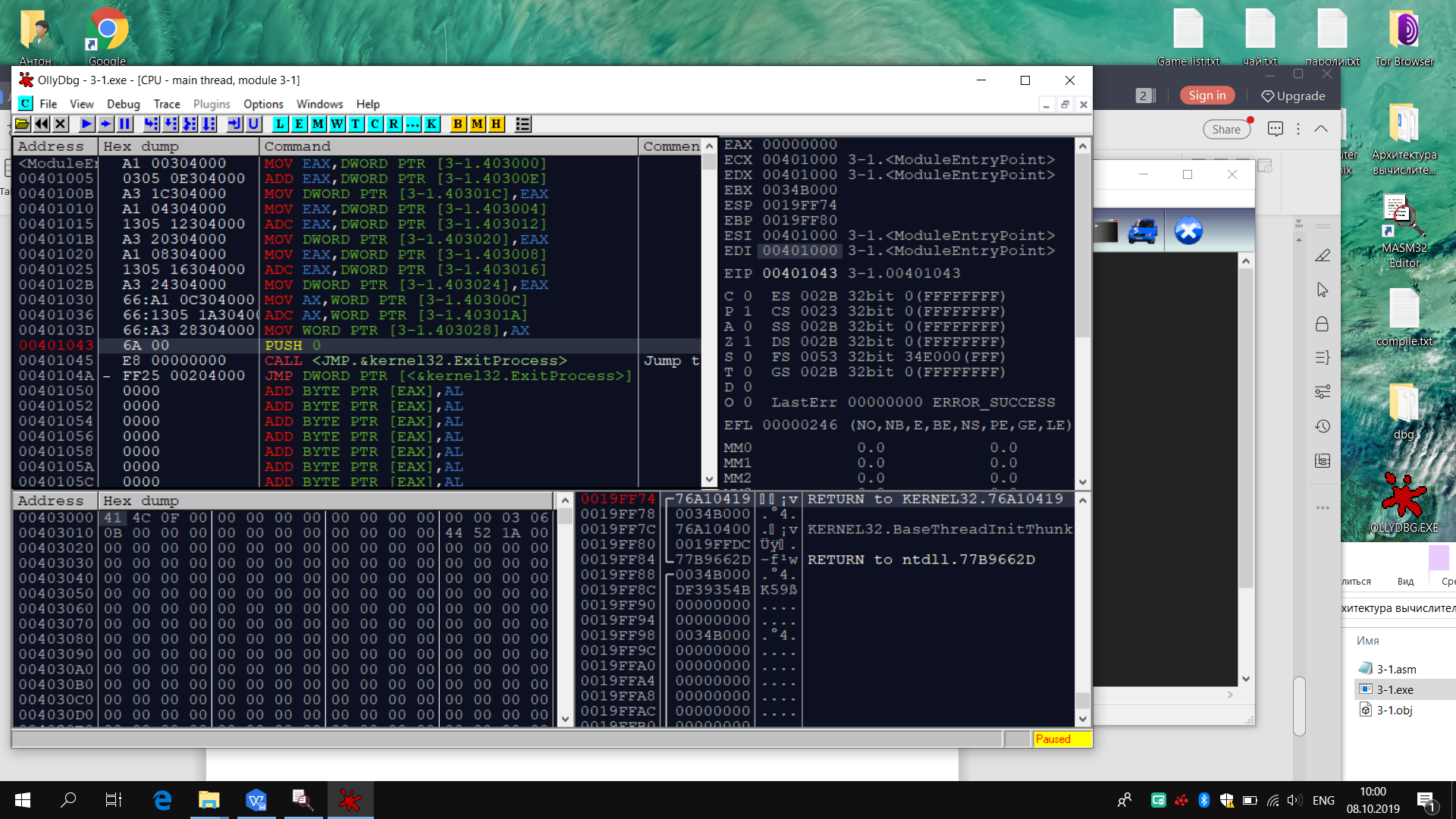
one db 65, 98, 23, 0Fh, 7, 2, 87, 3, 87, 23, 98, 0, 0, 0

two db 45, 0Ah, 6, 67, 0B5h, 02Fh, 65, 34, 34, 65, 88, 0, 0, 0



one db 65, 76, 0Fh, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

two db 3, 6, 0Bh, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0



.386

.MODEL FLAT, STDCALL

OPTION CASEMAP: NONE

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

.DATA

one db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh

two db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh

s db 0, 0, 0, 0, 0, 0, 0, 0

d db 0, 0, 0, 0, 0, 0, 0, 0

e db 0, 0, 0, 0, 0, 0, 0, 0

f db 0, 0, 0, 0, 0, 0, 0, 0

res db 16 dup(0)

.CODE

START:

XOR EAX, EAX

XOR EBX, EBX

MOV EAX, dword ptr one[0]

MOV EBX, dword ptr two[0]

MUL EBX

MOV dword ptr s[0], EAX

MOV dword ptr s[4], EDX

MOV EAX, dword ptr one[4]

MOV EBX, dword ptr two[0]

MUL EBX

MOV dword ptr d[0], EAX

MOV dword ptr d[4], EDX

MOV EAX, dword ptr one[0]

MOV EBX, dword ptr two[4]

MUL EBX

MOV dword ptr e[0], EAX

MOV dword ptr e[4], EDX

MOV EAX, dword ptr one[4]

MOV EBX, dword ptr two[4]

MUL EBX

MOV dword ptr f[0], EAX

MOV dword ptr f[4], EDX

MOV EAX, dword ptr s[0]

MOV dword ptr res[0], EAX

MOV EAX, dword ptr s[4]

MOV EBX, EAX

ADD EAX, dword ptr d[0]

MOV EBX, EAX

MOV EAX, dword ptr res[8]

ADC EAX, 0

MOV dword ptr res[8], EAX

MOV EAX, dword ptr res[12]

ADC EAX, 0

MOV dword ptr res[12], EAX

MOV EAX, EBX

ADD EAX, dword ptr e[0]

MOV EBX, EAX

MOV EAX, dword ptr res[8]

ADC EAX, 0

MOV dword ptr res[8], EAX

MOV EAX, dword ptr res[12]

ADC EAX, 0

MOV dword ptr res[12], EAX

MOV dword ptr res[4], EBX

MOV EAX, dword ptr res[8]

ADD EAX, dword ptr d[4]

MOV EBX, EAX

MOV EAX, dword ptr res[12]

ADC EAX, 0

MOV dword ptr res[12], EAX

MOV EAX, EBX

ADD EAX, dword ptr e[4]

MOV EBX, EAX

MOV EAX, dword ptr res[12]

ADC EAX, 0

MOV dword ptr res[12], EAX

MOV EAX, EBX

ADD EAX, dword ptr f[0]

MOV dword ptr res[8], EAX

MOV EAX, dword ptr res[12]

ADC EAX, 0

MOV dword ptr res[12], EAX

MOV EAX, dword ptr res[12]

ADD EAX, dword ptr f[4]

MOV dword ptr res[12], EAX

PUSH NULL

CALL ExitProcess

END START