; F=69+(1416-A+B) где 12h=18, 1369h=4969

; описываем используемые функции

extrn ExitProcess: proc,

GetStdHandle: proc,

WriteConsoleA: proc,

ReadConsoleA: proc,

lstrlenA: proc

.data

; Описываем макросы

; Выравниваем стек

STACKALLOC macro arg

push R15

mov R15, RSP

sub RSP, 8 \* 4

if arg

sub RSP, 8 \* arg

endif

and SPL, 0F0h

endm

; Освобождаем выделенную память

STACKFREE macro arg

mov RSP, R15

pop R15

endm

NULL\_FIFTH\_ARG macro arg

mov qword ptr [RSP + 32], 0

endm

; Макрозамены

STD\_OUTPUT\_HANDLE = -11

STD\_INPUT\_HANDLE = -10

; Объявляем глобальные переменные

; Дескрипторы ввода/вывода

hStdInput qword ?

hStdOutput qword ?

; Сумма которую находим

res qword ?

; Строки интерфейса пользователя

aOutputHandle db 'a = ', 0

bOutputHandle db 'b = ', 0

result db 'F = 18 + (4969 - A + B) = ', 0

invalid db 'Invalid character',0

continue db 0Ah, 'Press any key to exit...', 0

strErrL db 0Ah, 'Error Range', 0

; Процедуры

.code

; Главная процедура

Start proc

STACKALLOC 1 ; выделим место в стеке под аргументы

mov RCX, STD\_OUTPUT\_HANDLE ; поток вывода

call GetStdHandle

mov hStdOutput,RAX ; значение дескриптора

; Аналогично для потока ввода

mov RCX, STD\_INPUT\_HANDLE

call GetStdHandle

mov hStdInput, RAX

mov R8, 12h

; Выведем строку а =

lea RAX, aOutputHandle

push RAX

call StringWrite

call StringRead

; Проверим R10

cmp R10, 0

je M1

sub R8, RAX ; прибавим 'а' к сумме

; Выведем строку b =

lea RAX, bOutputHandle

push RAX

call StringWrite

call StringRead

; Проверим R10

cmp R10, 0

je M1

cmp RAX, 127

jge error1

; Вычтем 'в'

add R8,RAX

; Прибавим 120

add R8, 1369h

mov res, R8

; Выведем строку суммы

lea RAX, result

push RAX

call StringWrite

call PrintSignedNumber

jmp M1

error1:

lea RAX, strErrL

push RAX

call StringWrite

M1:

call inputAwaiting ; удержание консоли

xor RCX, RCX

call ExitProcess

Start endp

; Процедура вывода строки

StringWrite proc uses RAX RCX RDX R8 R9, string: qword

; Введем локальную переменную

local bytesWritten: qword

; Выделим место в стеке

STACKALLOC 1

; Получим длину строки

mov RCX, string

call lstrlenA

; Поместим аргументы в соотв. регистры

mov RCX, hStdOutput

mov RDX, string

mov R8, RAX

lea R9, bytesWritten

NULL\_FIFTH\_ARG

call WriteConsoleA

; Освободим стек

STACKFREE

; Возврат в основную программу

ret 8

StringWrite endp

; Процедура считывания с консоли

StringRead proc uses RBX RCX RDX R8 R9

; Введем локальные переменные

local readStr[8]: byte,

bytesRead: dword

; Выделим место в стеке

STACKALLOC 2

; Разместим аргументы в нужные регистры

mov RCX, hStdInput

lea RDX, readStr

mov R8, 64

lea R9, bytesRead

NULL\_FIFTH\_ARG

call ReadConsoleA

; Начнем вычисление строки

xor RCX, RCX

MOV ECX, bytesRead

sub ecx, 2

; Сделаем строку нуль-терминированной (ноль в конце)

mov readStr[RCX], 0

xor RBX, RBX ; Здесь будем хранить степени десятки

mov R8, 1

mov BL, readStr[0]

cmp BL, 0h

je error

xor RBX, RBX

rangeChecking:

cmp RBX, 32767

jge errorLen

dec RCX

; Проверяем знак числа

cmp RCX, -1

je scanningComplete

xor RAX, RAX

mov AL, readStr[RCX]

cmp RCX, 0

jne K

cmp AL, '-'

jne eval

cmp RBX,0

je error

neg RBX

jmp scanningComplete

; Проверка является ли символ десятичной цифрой

; Если нет, то переход на еrror

K:

cmp AL,'-'

je error

eval:

cmp AL, 30h

jl error

cmp AL, 39h

jg error

sub RAX, 30h

mul R8

add RBX, RAX

mov RAX, 10

mul R8

mov R8, RAX

jmp rangeChecking

error:

; Очистим R10

xor R10,R10

; Загрузим адрес строки с выводом ошибкой

lea RAX, invalid

push RAX

call StringWrite

STACKFREE

ret 8\*2

; Проверка на выход за границу числа

errorLen:

xor R10,R10

; Адрес строки с выводом ошибки

lea RAX, strErrL

push RAX

call StringWrite

STACKFREE

ret 8\*2

; Завершение сканирования

scanningComplete:

mov R10,1

mov RAX, RBX

STACKFREE

ret 8\*2

StringRead endp

; Процедура вывода знакового числа

PrintSignedNumber proc uses RAX RCX RDX R8 R9 R10 R11

; Введем локальную переменную

local numberStr[22] :byte

; Регистр счётчик

xor R8,R8

; число положительное или отрицательное

mov RAX, res

mov RBX, 63

btc res, RBX

jae digitDivision

; Если отрицательное, то первым символом должен стать '-'.

mov numberStr[0],2Dh

inc R8

neg RAX

digitDivision:

; Занесем в RBX 10 для осуществления деления.

mov RBX,10

; Сбросим значение RCX для записи длины строки.

xor RCX,RCX

stackIterating:

xor RDX, RDX

; Разделим RAX на RBX

div RBX

add RDX, '0'

; Поместим остаток в стек и увеличим RCX.

push RDX

inc RCX

; Проверим, если RAX стал нулем, то мы закончили деление

cmp RAX, 0

jne stackIterating

numberCollecting:

; переместим в регистр символ цифры из стека

pop RDX

mov numberStr[R8], DL

inc R8

; Повторим эти действия, пока RCX не станет нулем

loop numberCollecting

mov numberStr[R8], 0

lea RAX, numberStr

push RAX

call StringWrite

; Вернемся в место вызова, очищая стек от одного аргумента

ret 8

PrintSignedNumber endp

; Процедура ожидания ввода

inputAwaiting proc uses RAX RCX RDX R8 R9 R10 R11

; Введем локальные переменные

local readStr: byte,

bytesRead: dword

; Выровняем стек

STACKALLOC 1

; Передадим в регистры все необходимы параметры

lea RAX, continue

push RAX

call StringWrite

; Установим значения всех аргументов

mov RCX, hStdInput

lea RDX, readStr

mov R8, 1

lea R9, bytesRead

NULL\_FIFTH\_ARG

call ReadConsoleA

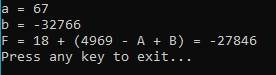
STACKFREE

ret

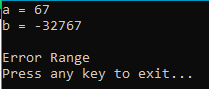
inputAwaiting endp

end

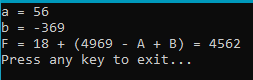
Ввод аргументов с получением отрицательного результата:



Ввод аргументов с получением ошибки размера хотя бы одного из аргументов:



Ввод аргументов с получением положительного результата:



Ввод аргументов с не цифровыми символами и получение соответствующей ошибки:

