МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра дискретной математики и информатики

ЛАБОРАТОРНАЯ РАБОТА №8

студента 2 курса 221 группы

направления 09.03.01 Информатика и вычислительная техника

факультета компьютерных наук и информационных технологий

Мусатова Федора Алексеевича

Саратов 2025

Вариант 7818

CONST = 450Ch

op1 = —

op2 = +

A = байт со знаком (-128 … 127)

B = байт со знаком (-128 … 127)

Надо найти максимум  
  
; Внешние процедуры и функции

extern ExitProcess : proc,

GetStdHandle : proc,

WriteConsoleA : proc,

ReadConsoleA : proc,

lstrlenA : proc

; Макросы для работы со стеком

STACKALLOC macro arg

push R15 ; Выравниваем стек

mov R15, RSP ; Возьмем регистр R15, в котором будет храниться указатель на "старый" стек

sub RSP, 8 \* 4 ; Поместим в выбранный регистр значение RSP

if arg ; Если число аргументов макроса не равно нулю, то освободим место и для них.

sub RSP, 8 \* arg

endif

and SPL, 0F0h ; Выравниваем стек по 16-байтовой границе

endm

STACKFREE macro

mov RSP, R15 ; Освобождаем выделенную память

pop R15 ; Занесем в регистр RSP значение, сохраненное в регистре R15

endm

NULL\_FIFTH\_ARG macro

mov qword ptr [RSP + 32], 0 ; Установка пятого аргумента в ноль

endm

STD\_OUTPUT\_HANDLE = -11

STD\_INPUT\_HANDLE = -10

.data

; Строки вывода

astr db 'A = ', 0

bstr db 'B = ', 0

ans db '450Ch - A + B = ', 0

maxs db 'Max = ', 0

newline db 0Ah, 0

outstr db 'Press any button to exit program...', 0

hStdInput dq 0

hStdOutput dq 0

sum dq 0

maxnumb dq 0

; Буферы для ввода/вывода

readStr db 64 dup(0)

bytesRead dd 0

numberStr db 22 dup(0)

; Ошибки

formaterror db 'Input format error', 0

aerror db 'Number A out of range', 0

berror db 'Number B out of range', 0

.code

Start proc

sub RSP, 8 \* 4

and SPL, 0F0h

; Получение дескрипторов

mov RCX, STD\_OUTPUT\_HANDLE

call GetStdHandle

mov hStdOutput, RAX

mov RCX, STD\_INPUT\_HANDLE

call GetStdHandle

mov hStdInput, RAX

; Ввод A

lea RAX, astr

push RAX

call OutputProc

call InputProc

cmp R10, 1

jz err ; Ошибка формата

cmp R10, 2

jz erra ; Ошибка диапазона

mov AL, BL ; Переносим результат в 8-битный регистр

movsx R8, AL ; Сохраняем A как 64-битное число

; Ввод B

lea RAX, bstr

push RAX

call OutputProc

call InputProc

cmp R10, 1

jz err ; Ошибка формата

cmp R10, 2

jz errb ; Ошибка диапазона

mov AL, BL ; Переносим результат в 8-битный регистр

movsx R9, AL ; Сохраняем B как 64-битное число

; Вычисление F = 450Ch - A + B

xor RAX, RAX

mov RAX, 450Ch ; 450Ch = 17676

sub RAX, R8

add RAX, R9

mov sum, RAX

; Нахождение максимума

cmp R8, R9

jg first

mov maxnumb, R9

jmp continue

first:

mov maxnumb, R8

continue:

; Вывод выражения

lea RAX, ans

push RAX

call OutputProc

; Вывод F

push sum

call OutputNum

; Вывод новой строки

lea RAX, newline

push RAX

call OutputProc

; Вывод "Max = "

lea RAX, maxs

push RAX

call OutputProc

; Вывод максимума

push maxnumb

call OutputNum

; Ожидание ввода

call ExpectInput

jmp final

err:

lea RAX, formaterror

push RAX

call OutputProc

jmp final

erra:

lea RAX, aerror

push RAX

call OutputProc

jmp final

errb:

lea RAX, berror

push RAX

call OutputProc

jmp final

final:

xor RCX, RCX

call ExitProcess

Start endp

; Процедура вывода строки

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

local bytesWritten: qword

STACKALLOC 1

; Получаем длину строки

mov RCX, string

call lstrlenA

; Вызываем WriteConsoleA

mov RCX, hStdOutput

mov RDX, string

mov R8, RAX

lea R9, bytesWritten

NULL\_FIFTH\_ARG

call WriteConsoleA

STACKFREE

ret 8

OutputProc endp

; Процедура ввода числа

InputProc proc uses RAX RCX RDX R8 R9

STACKALLOC 2

; Читаем строку

mov RCX, hStdInput

lea RDX, readStr

mov R8, 64

lea R9, bytesRead

NULL\_FIFTH\_ARG

call ReadConsoleA

; Обрабатываем строку

xor RCX, RCX

mov ECX, bytesRead

cmp ECX, 2

jl error ; Строка пустая

sub ECX, 2 ; Убираем CR/LF

cmp ECX, 0

jle error ; Строка пустая после удаления CR/LF

lea RSI, readStr

xor RBX, RBX ; Число

xor R10, R10 ; Флаг ошибки: 0 - нет ошибки, 1 - формат, 2 - диапазон

xor R11, R11 ; Флаг отрицательного числа

xor R12, R12 ; Счетчик цифр

; Проверка первого символа

mov AL, [RSI]

cmp AL, '-'

jz process\_negative

cmp AL, '+'

jz skip\_sign

; Проверка цифры

cmp AL, '0'

jl error

cmp AL, '9'

jg error

sub AL, '0'

mov RBX, RAX

inc R12

inc RSI

dec ECX

jmp process\_digits

skip\_sign:

inc RSI

dec ECX

jmp process\_digits

process\_negative:

inc RSI

dec ECX

cmp ECX, 0

jle error ; Только минус

inc R11 ; Флаг отрицательного числа

process\_digits:

xor RAX, RAX ; Аккумулятор

process\_digit:

cmp ECX, 0

jle done\_processing ; Выход, если символы закончились

mov AL, [RSI]

cmp AL, '0'

jl error

cmp AL, '9'

jg error

inc R12 ; Счётчик цифр

sub AL, '0'

; Преобразование строки в число

imul RBX, RBX, 10

jo error\_overflow

add RBX, RAX

jo error\_overflow

inc RSI

dec ECX

jmp process\_digit

done\_processing:

cmp R12, 0

je error ; Нет цифр

; Проверка диапазона

cmp R11, 0

jnz check\_negative

cmp RBX, 127

jg error\_overflow

jmp done

check\_negative:

cmp RBX, 128

jg error\_overflow

neg RBX

cmp RBX, -128

jl error\_overflow

done:

mov RAX, RBX

STACKFREE

ret 8 \* 2

error\_overflow:

mov R10, 2 ; Ошибка диапазона

STACKFREE

ret 8 \* 2

error:

mov R10, 1 ; Ошибка формата

STACKFREE

ret 8 \* 2

InputProc endp

; Процедура вывода числа

OutputNum proc uses RAX RCX RDX R8 R9 R10 R11, number: qword

STACKALLOC 1

xor R8, R8 ; Счётчик символов

mov RAX, number

cmp RAX, 0

jge pos

negate:

lea RDX, numberStr

mov byte ptr [RDX + R8], '-'

inc R8

neg RAX ; Делаем положительным

pos:

mov RBX, 10

xor RCX, RCX ; Количество цифр

read:

xor RDX, RDX

cqo ; Знаковое расширение

idiv RBX ; Деление с учётом знака

add RDX, '0' ; Цифра -> символ

push RDX

inc RCX ; Счётчик

cmp RAX, 0

jne read

; Обработка нуля

cmp RCX, 0

je zero\_case

write:

pop RDX

lea RAX, numberStr

mov byte ptr [RAX + R8], DL

inc R8

loop write

jmp done

zero\_case:

mov byte ptr [numberStr], '0'

inc R8

done:

lea RAX, numberStr

mov byte ptr [RAX + R8], 0 ; Нуль-терминатор

lea RAX, numberStr

push RAX

call OutputProc

STACKFREE

ret 8

OutputNum endp

; Процедура ожидания ввода

ExpectInput proc uses RAX RCX RDX R8 R9 R10 R11

STACKALLOC 1

lea RAX, outstr

push RAX

call OutputProc

mov RCX, hStdInput

lea RDX, readStr

mov R8, 1

lea R9, bytesRead

NULL\_FIFTH\_ARG

call ReadConsoleA

STACKFREE

ret

ExpectInput endp

end