Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Конструирование программ»

**ОТЧЕТ**

к лабораторной работе №6

на тему:

**«ЗАГРУЗКА И ВЫПОЛНЕНИЕ ПРОГРАММ. РАБОТА С ПАМЯТЬЮ.»**

БГУИР 6-05-0612-02 39

|  |
| --- |
| Выполнил студент группы 353503  ДЕРГУН Николай Олегович |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 9.** Написать программу для обработки строки вида «A1 x1 A2 x2 … xN-1 AN», где Ai – числа (i в диапазоне [1, N]), xi – одна из арифметических операций (для обработки арифметических операций сформировать оверлейные функции, в память одновременно может быть загружена только одна функция, загрузку функций оптимизировать).

# 2 Выполнение работы

Программа принимает на вход строчку из чисел (двухбайтовых знаковых целочисленных) и четырёх арифметических знаков, при этом числа и знаки должны чередоваться, первый элемент должен быть числом, количество пробелов между элементами строки произвольное, и должна быть задана хотя бы одна операция, если ввод не пустой.

Листинг кода 1 – Основная процедура программы main

main proc

mov ax, @data

mov ds, ax

call mng\_mem

; now there is free space to allocate

call copy\_cmd\_args

mov ah, 62h

int 21h

mov es, bx

mov al, es:[80h]

or al, al

jz no\_args\_\_

mov cmd\_args\_size, al

;cmd arg check

data\_to\_ds

mov ax, ds

mov es, ax

mov cx, -1

lea si, cmd\_args

mov dx, -1 ; what was the last variable

find\_param:

mov di, si

mov al, ' '

repe scasb

dec di

mov si, di

lodsb

dec si

cmp al, 0Dh

je params\_ended

cmp al, '-'

jne not\_minus

inc si

lodsb

sub si, 2

cmp al, ' '

je operator\_ch

jmp num\_ch

no\_args\_\_:

jmp no\_args\_

not\_minus:

cmp al, '0'

jae num\_ch

operator\_ch:

cmp dx, 1

je wrong\_args

cmp dx, -1

je wrong\_args

mov dx, 1

mov first\_op, 1

call check\_operator

cmp bx, -1

je wrong\_args

jmp find\_param

num\_ch:

cmp dx, 2

je wrong\_args

mov dx, 2

call check\_num

cmp bx, -1

je wrong\_args

jmp find\_param

no\_args\_:

jmp no\_args

wrong\_args:

lea dx, bad\_args\_provided

call printl

mov ah, 4Ch

int 21h

params\_ended:

cmp first\_op, 0

je wrong\_args

lea si, cmd\_args

find\_op\_h:

lodsb

dec si

mov bx, 1

cmp al, '\*'

je find\_fnum

mov bx, 4

cmp al, '/'

je find\_fnum

cmp al, 0Dh

je find\_op\_l\_

inc si

jmp find\_op\_h

params\_ended\_:

jmp params\_ended

find\_op\_l\_:

lea si, cmd\_args

find\_op\_l:

lodsb

dec si

mov bx, 2

cmp al, '+'

je find\_fnum

mov bx, 3

cmp al, '-'

je find\_fnum

cmp al, 0Dh

je results\_

inc si

jmp find\_op\_l

find\_fnum:

mov op, bl

mov byte ptr es:[si], ' '

push si

call sk\_sp\_r

sp\_sk\_1:

lodsb

dec si

cmp al, ' '

jne num\_sk\_1

dec si

jmp sp\_sk\_1

num\_sk\_1:

lodsb

dec si

cmp al, ' '

je fn\_read

dec si

jmp num\_sk\_1

results\_:

jmp results

params\_ended\_\_:

jmp params\_ended

fn\_read:

inc si

mov left, si

call read\_num

mov fnum, bx

pop si

;find\_snum

push si

mov di, si

call skip\_spaces

mov si, di

call read\_num

pop si

mov snum, bx

;calc

mov al, op

cmp al, 4 ;'/'

jne div\_check\_skip

xor ax, ax

cmp snum, ax

jne div\_check\_skip

lea dx, div\_by\_zero\_msg

call printl

mov ah, 4Ch

int 21h

div\_check\_skip:

; load overlay

call load\_overlay

call write\_num

jmp params\_ended\_\_

results:

call print\_result

no\_args:

mov ah, 4Ch

int 21h

main endp

Для работы с памятью в процедуре main используются процедуры mng\_mem и load\_overlay для освобождения памяти под оверлей и для использования оверлея (освобождение, загрузка, вызов) соответственно.

Листинг кода 2 – Процедуры для работы с памятью

mng\_mem proc

mov ax, es

mov bx, seg zseg

sub bx, ax

mov ah, 4ah

int 21h

jnc resize\_mem\_ok

mov ah, 09h

lea dx, [asErrResizeMem]

int 21h

mov ah, 4Ch

int 21h

resize\_mem\_ok:

ret

mng\_mem endp

; result in res variable

; bx = 1 if there was an overflow

load\_overlay proc

mov al, prev\_op

xor ah, ah

cmp al, op

je overlay\_call

cmp al, 0

je overlay\_alloc

; release memory

mov ah, 49h

mov bx, [pars].wStartSeg

mov es, bx

int 21h

overlay\_alloc:

mov bx, 1000h

mov ah, 48h

int 21h

jnc alloc\_ok

mov ah, 09h

lea dx, [asErrAllocMem]

int 21h

mov ah, 4Ch

int 21h

alloc\_ok:

mov [pars].wStartSeg, ax

mov [pars].wReloFactor, ax

mov word ptr [entry+2], ax

mov word ptr [entry], 100h

mov [stkseg], ss

mov [stkptr], sp

mov ax, ds

mov es, ax

mov al, op

mov dx, offset mul\_path

cmp al, 1

je path\_is\_found

mov dx, offset sum\_path

cmp al, 2

je path\_is\_found

mov dx, offset sub\_path

cmp al, 3

je path\_is\_found

mov dx, offset div\_path

path\_is\_found:

mov bx, offset pars

mov ax, 4b03h

int 21h

mov ax, \_DATA

mov ds, ax

mov es, ax

cli

mov ss, [stkseg]

mov sp, [stkptr]

sti

jnc exec\_ok

mov ah, 09h

lea dx, [asErrExec]

int 21h

mov ah, 4Ch

int 21h

exec\_ok:

overlay\_call:

mov ax, fnum

mov bx, snum

push ds

call dword ptr[entry]

pop ds

mov bl, op

mov prev\_op, bl

use\_results:

mov res, ax

ret

load\_overlay endp

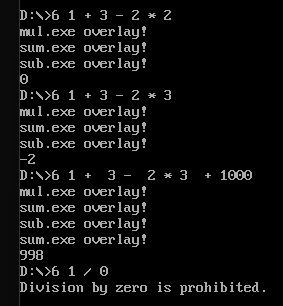


Рисунок 1 – Пример выполнения программы

# Вывод

В ходе выполнения лабораторной работы были получены навыки работы с памятью с использованием функций DOS. Ознакомились с использованием оверлеев.