МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра вычислительные системы и технологии

Лабораторная работа № 4

Разработка интерфейсной программы консольного  
ввода-вывода с использованием макрокоманд и процедур

Вариант №13

ОТЧЕТ

по лабораторной работе

по дисциплине

Принципы и методы  
организации системных программных средств

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Викулова Е.Н.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сапожников В.О.

19-ИВТ-3

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

**Цель работы**

Изучение средств макропрограммирования и организации процедур. Управление терминалом через прерывания.

**Задание**

Написать программу на ассемблере, реализующую пользовательский интерфейс для программы арифметической обработки числовых данных (программа арифметической обработки уже разработана в лаб.раб. №3), включающую следующие элементы:

1. ввод исходных данных с соответствующими проверками и информированием пользователя об ошибках ввода;

2. реакция на функциональные клавиши;

3. вывод результатов на экран в удобном для пользователя виде;

4. реализация управления экраном в текстовом режиме:

- управление курсором, атрибутами, страницами;

- прокрутка, очистка;

- организация полей ввода текста, рамок («окон»), манипуляция окнами;

- создание меню, кнопок, переключателей.

Вариант 13: вычислить сумму положительных, произведение отрицательных и их разность, а также определить минимальное и максимальное число.

**Поставленные задачи**

1. Программа должна иметь запоминающиеся приветствие пользователя.
2. Переход между основными блоками программы должен осуществляться при нажатии соответствующих клавиш. (Пример: для продолжения нажмите Enter)
3. Пользователь должен иметь право сам выбирать место ввода в программе (Окно ввода можно двигать)
4. Пользователь должен иметь возможность прервать исполнение программы на любом из её этапов. (Пример: завершение работы при нажатии F4)
5. При ошибках ввода и ошибках арифметической обработки пользователь должен получать соответствующее сообщение в отдельном окне, при нажатии клавиши программа завершается.
6. Программа должна выполнять арифметическую обработку согласно заданию Вариант 13 Лабораторной работы №3

Замечание: программа должна избавиться от недоработок предыдущей вариации – Лабораторной работы №3: при вводе всех нулей произведение отрицательных чисел равнялось 1, а разность суммы и произведения -1 соответственно.

**Структура программы**

Разработана программа является многосегментной и имеет расширение exe. В программе имеет 3 сегмента. Сегмент кода d1, связанный с сегментным регистром cs; сегмент данных d1, связанный с сегментным регистром ds и сегмент стека, связанный с сегментным регистром ss.

ASSUME cs:c1, ds:d1, ss:st1

***макрокоманды***

printInWindow – макрокоманда вывода сообщений в определённую область. В

качестве параметров принимает сообщение для ввода, строку и столбец

куда следует установить курсор.

print – макрокоманда вывода сообщений в текущее положение курсора

input – макрокоманда ввода сообщения в текущем положение курсора

sleep – “засыпание” на заданное время при помощи функции **86h**

прерывания **Int 15h**

drawWindow – отрисовка области по заданным координатам и с заданными

цветами фона и букв при помощи функции **06h** прерывания **Int 10h**

***процедуры***

hideCursor – процедура переставляющая курсор за пределы консоли при

помощи функции **02h** прерывания **Int 10h**

leftShift – “сдвиг” координат окна влево

rightShift – “сдвиг” координат окна вправо

upShift – “сдвиг” координат окна вверх

downShift – “сдвиг” координат окна вниз

Такие процедуры как DIAPAZON, DOPUST, AscToBin, BinToAsc позаимствованы из учебных материалов к Лабораторной работе №3

**Алгоритмы обработки введенных значений**

Для упрощения обработки ситуации выхода за пределы допустимого диапазона было решено ограничить диапазон от -29999 до 29999. При такой обратке нам необходимо контролировать чтобы было введено 5 символом и первым числовым символом является цифра 2.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Проверка диапазона вводимых чисел -29999,+29999 \*

;\* Аргументы: \*

;\* Буфер ввода - stroka \*

;\* \*

;\* Результат: \*

;\* bh - флаг ошибки ввода \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DIAPAZON PROC

xor bh, bh

xor si, si

cmp kol, 05h ;Если ввели менее 5 символов, проверим их допустимость

jb dop

cmp stroka, 2dh ;Eсли ввели 5 или более символов проверим является ли

;первый минусом

jne plus ; Eсли 1 символ не минус, проверим число символов

cmp kol, 06h ;Eсли первый - минус и символов меньше 6 проверим

;допустимость символов

jb dop

inc si ;Иначе проверим первую цифру

jmp first

plus:

cmp kol,6 ;Bведено 6 символов и первый - не минус

je error1 ;Oшибка

first:

cmp stroka[si], 32h ; Cравним первый символ с '2'

jna dop ; Eсли первый <= '2' - проверим допустимость символов

error1:

mov bh, flag\_err ;Иначе bh = flag\_err

dop:

ret

DIAPAZON ENDP

Однако такой проверки недостаточно. Если строка имеет 5 символов и первый числовой символ это 2, то необходимо проверить все оставшиеся символы на вхождение в числовой диапазон. Для этого мы убеждаемся, что ASCII коды введенных символов принадлежат промежутку от 30h до 39h.

DOPUST PROC

xor bh, bh

xor si, si

xor ah, ah

xor ch, ch

mov cl, kol ;В (cl) количество введенных символов

m11:

mov al, [stroka + si] ;B (al) - первый символ

cmp al, 2dh ;Является ли символ минусом

jne testdop ;Если не минус - проверка допустимости

cmp si, 00h ;Если минус - является ли он первым символом

jne error2 ;Если минус не первый - ошибка

jmp m13

testdop:

cmp al, 30h ;Является ли введенный символ цифрой

jb error2

cmp al, 39h

ja error2

m13:

inc si

loop m11

jmp m14

error2:

mov bh, flag\_err ;При недопустимости символа bh = flag\_err

m14:

ret

DOPUST ENDP

Для удобства работы с числами переводим их в двоичные. Для этого из кода каждой цифры вычитаем 30h, умножаем результат на вес разряда и складываем полученные произведения. Отрицательные числа представляем в дополнительном коде при помощи команды neg. Для вывода полученных результатов переводим числа в их ASCII коды, т.е. выполняем обратную операцию.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* ASCII to number \*

;\* Аргументы: \*

;\* B cx количество введенных символов \*

;\* B bx - номер символа начиная с последнего \*

;\* \*

;\* Результат: \*

;\* Буфер чисел - number \*

;\* B di - номер числа в массиве \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AscToBin PROC

xor ch, ch

mov cl, kol

xor bh, bh

mov bl, cl

dec bl

mov si, 01h ;В si вес разряда

n1:

mov al, [stroka + bx]

xor ah, ah

cmp al, 2dh ;Проверим знак числа

je otr ;Eсли число отрицательное

sub al, 30h

mul si

add [number + di], ax

mov ax, si

mov si, 10

mul si

mov si, ax

dec bx

loop n1

jmp n2

otr:

neg [number + di] ;Представим отрицательное число в дополнительном коде

n2:

ret

AscToBin ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Number to ASCII \*

;\* Аргументы: \*

;\* Число передается через ax \*

;\* \*

;\* Результат: \*

;\* Буфер чисел - out\_str \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BinToAsc PROC

xor si, si

add si, 05h

mov bx, 0Ah

push ax

cmp ax, 00h

jnl mm1

neg ax

mm1:

cwd

idiv bx

add dl,30h

mov [out\_str + si], dl

dec si

cmp ax, 00h

jne mm1

pop ax

cmp ax, 00h

jge mm2

mov [out\_str + si], 2dh

mm2:

ret

BinToAsc ENDP

**Алгоритмы арифметической обработки над двоичными числами**

Задача 1: нахождение минимального и максимального элемента.

Для нахождение максимального и минимального элементов формируем нулевую гипотезу: в max помещаем минимальное возможно допустимое значение, а в min максимальное. При выбранном диапазоне это составляет -29999 и 29999

min dw 29999 ;Минимальный элемент

max dw -29999 ;Максимальный элемент

Алгоритм

Цикл-Для каждого элемента массива:

Условие: элемент > max ?

Да: max = элемент

Нет: идем дальше

Условие: элемент < min ?

Да: min = элемент

Всё-цикл

;\*\*\*\*\*\*\*\*\* Нахождение минимального и максимального элементов \*\*\*\*\*\*\*\*\*\*\*\*\*\*

searchMinMax:

mov cx, siz ;cx - размер массива

mov si, offset number

xor ax, ax

searchLoop: ;для поиска минимальных и максимальных

mov ax,[si] ;элементов была принята нулевая гипотеза:

;max = -29999 (минимальный из возможных   
 элементов)

;min = 29999 (максимальный из возможных   
 элементов)

findMax:

cmp ax, max ;если найден элемент больше текущего   
 значения max,

jg foundMax ;то переход к перезаписи max

findMin:

cmp ax, min ;если найден элемент меньше текущего   
 значения min,

jl foundMin ;то переход к перезаписи min

jmp nextVal ;иначе переход к следующему элементу

foundMax:

mov max, ax ;перезапись max элемента

jmp findMin

foundMin:

mov min,ax ;перезапись min элемента

nextVal:

inc si

inc si

loop searchLoop

Задача 2: нахождение суммы положительных элементов

Алгоритм

Цикл-Для каждого элемента массива:

Условие: элемент > 0 ?

Да: аккумулятор += элемент

Нет: переход к следующему элементу

Проверка переполнения

Всё-цикл

;\*\*\*\*\*\*\*\*\*\* Нахождение суммы положительных элементов \*\*\*\*\*\*\*\*\*\*

searchPosSum:

mov cx, siz ;cx - размер массива

mov si, offset number

xor ax, ax

sumPositive:

mov ax,[si]

cmp ax, 0 ;сравнение с 0

jl negative ;если число меньше то переход

add sumPos,ax ;иначе сложение с переменной

jo overFlowErr ;если переполнение, то переход

negative:

inc si

inc si

loop sumPositive

Замечание: в данном случае переполнение – это выход из допустимого диапазона – затирается знаковый бит.

Задача 3: нахождение произведения отрицательных чисел

Алгоритм

Заносим в аккумулятор 1

Цикл-Для каждого элемента массива:

Условие: элемент < 0 ?

Да: аккумулятор \*= элемент

Нет: переход к следующему элементу

Проверка переполнения

Всё-цикл

Записываем значение аккумулятора в переменную результата

;\*\*\*\*\*\*\*\*\*\*\*\*\* Нахождение произведения отрицательных элементов \*\*\*\*\*\*\*\*\*\*\*\*\*

searchNegMul:

mov cx, siz ;cx - размер массива

mov si, offset number

mov ax, 1 ;для циклического умножения заносим 1

;поскольку умножение всегда просиходит с

регистром ax

minusEl:

mov bx,[si]

cmp bx, 00h

jge plusEl ;если положительный элемент - идем дальше

imul bx ;иначе умножаем

jo overFlowErr ;проверяем на переполнение

plusEl:

inc si

inc si

loop minusEl

mov mulNeg, ax ;заносим значение в переменную

Замечание: в данном случае переполнение – это получение в качестве результат двойного слова, т.е. результат занимает и ax, и dx, хотя алгоритм предназначен для работы со словами, т.е. только с регистром ax.

Задача 4: нахождение разности суммы положительных элементов и разницы отрицательных. Поскольку команда вычитания sub работает по следующему принципу: <Приемник> = <приемник> - <источник>, то чтобы не испортить результат предшествующих алгоритмов, производим действия через регистр.

Алгоритм

Заносим в переменную-буффер (регистр) значение суммы

Вычитаем значение произведения

Записываем полученное значение в результирующую переменную.

;\*\* Нахождение разницы суммы положительных и разницы отрицательных элементов ;\*\*\*\*

searchDiffSumMul: ;поскольку вычитание (sub) работает по принципу:

mov ax, sumPos ; <Приемник>=<Приемник>-<Источник>

sub ax, mulNeg ;чтобы не испортить результаты предыдущих   
 вычислений

mov diffRez, ax ;мы одну из переменных переносим в отдельный   
 регистр, где и сохраним

;результат, а затем запишем значение регистра в   
 переменную

jmp resOutput ;переход к выводу результатов

**Используемые прерывания**

В разработанной программе используются следующие прерывания и их функции:

***Int 10h***

*00h функция* – установить видео режима. AL – номер режима

*02h функция* – установить позицию курсора. DH, DL – строка, столбец

*06h функция* – прокрутки вверх. CH, CL – строка, колонка верхнего левого

угла (считая от 0); DH, DL – строка, колонка нижнего правого угла

(считая от 0)

***Int 15h***

*86h функция* – ожидание. CX, DX – сколько микросекунд надо ждать до

возвращения управления

***Int 16h***

*00h функция* – читать(ожидать) следующую нажатую клавишу

Выход: AL – ASCII код символа, AH – сканкод/расширенный ASCII

код  
 *Используемые ключевые клавиши*

|  |  |
| --- | --- |
| **Клавиша** | **ASCII-код + сканкод** |
| Enter | 1CODh |
| F4 | 3E00h |
| F5 | 3F00h |
| ← | 4B00h |
| → | 4D00h |
| ↑ | 5000h |
| ↓ | 4800h |

***Int 21h***

*09h функция* – Выдать строку

DS:DX – адрес строки, заканчивающейся символом ‘$’

*0ah функция* – Ввод строки в буфер

DS:DX – адрес входного буфера.

**Неудачные моменты в программе**

Неудачным местом в программе является “приветствие”. Поскольку при выводе сообщений при помощи функции **09h** прерывания **Int 21h** текст в любом случае будет выведен внутри консоли целиком, где бы не находился курсор делает невозможным организовать построчный вывод сообщений.

Идея: выводить в цикле текст так, чтобы n строк было за границами консоли и его не было видно и с каждый итерацией “поднимать” его на строку вверх.

Реализация с учетом ограничений: заведем множество переменных, где будет текст приветствия с разным кол-ом строк и в зависимости от итерации будем выводить нужную переменную.

***код***

***сегмент данных***

FullIntroduction1 db 15 dup (' '),'Episode IV', '$'

FullIntroduction2 db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK', '$'

FullIntroduction3 db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK',10, 13, '$'

FullIntroduction4 db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK',10, 13, 10, 20 dup (' '), 'The assembly language laboratory work is', '$'

FullIntroduction5 db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK',10, 13, 10, 20 dup (' '), 'The assembly language laboratory work is',10, 20 dup(' '),'underway. Students make their first', '$'

FullIntroduction6 db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK',10, 13, 10, 20 dup (' '), 'The assembly language laboratory work is',10, 20 dup(' '),'underway. Students make their first',10, 20 dup(' '),'successes in the battle with the evil', '$'

FullIntroduction db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK',10, 13, 10, 20 dup (' '), 'The assembly language laboratory work is',10, 20 dup(' '),'underway. Students make their first',10, 20 dup(' '),'successes in the battle with the Elena,10, 20 dup(' '), 'Nikolaevna...', '$'

***сегмент кода***

writeLoop:

drawWindow 20, countYStart, 60, countYEnd, titleBGColor

cmp countYStart, 24d

je print1IntroStr

cmp countYStart, 23d

je print2IntroStr

cmp countYStart, 22d

je print3IntroStr

cmp countYStart, 21d

je print4IntroStr

cmp countYStart, 20d

je print5IntroStr

cmp countYStart, 19d

je print6IntroStr

printInWindow FullIntroduction, countYStart, 21

cmp countYStart, 0

je waitToStart

continueLoop:

call hideCursor

dec countYStart

dec countYEnd

sleep 5

loop writeLoop

print1IntroStr:

printInWindow FullIntroduction1, countYStart, 21

jmp continueLoop

print2IntroStr:

printInWindow FullIntroduction2, countYStart, 21

jmp continueLoop

print3IntroStr:

printInWindow FullIntroduction3, countYStart, 21

jmp continueLoop

print4IntroStr:

printInWindow FullIntroduction4, countYStart, 21

jmp continueLoop

print5IntroStr:

printInWindow FullIntroduction5, countYStart, 21

jmp continueLoop

print6IntroStr:

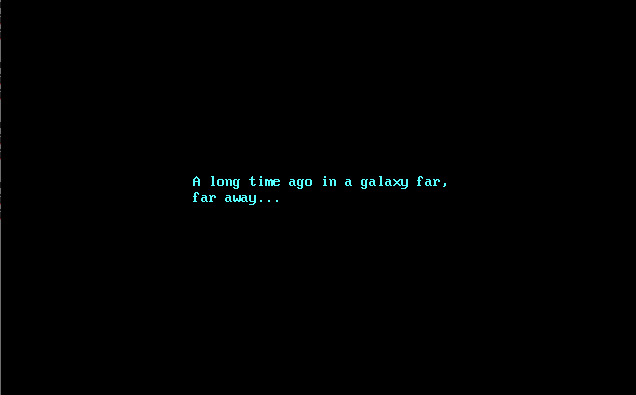
printInWindow FullIntroduction6, countYStart, 21

jmp continueLoop

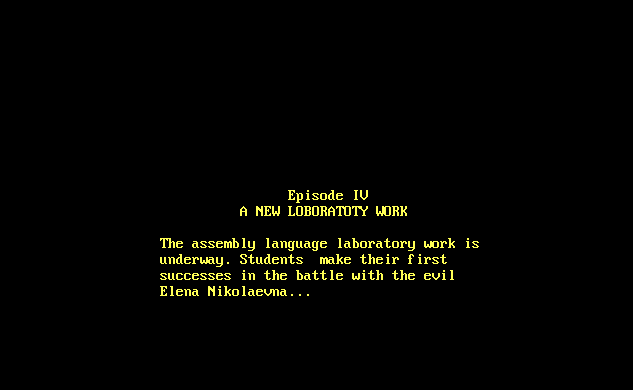
**Результаты работы**

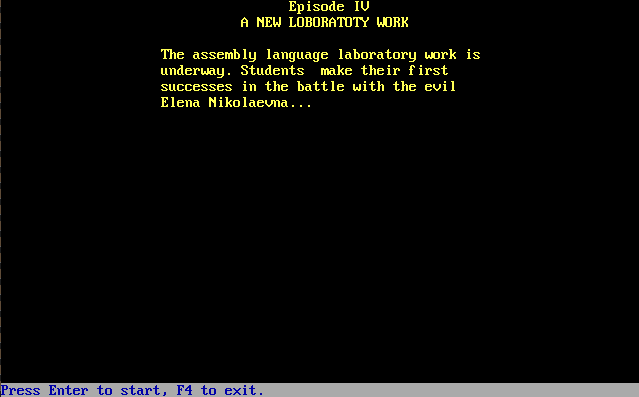
**Замечание:** если на скрине внизу видна серая полоса – то скрин сделан в момент мерцания текста (т.е. это дефект скрина, а не программы ☺)

*Приветствие*

******

******

******

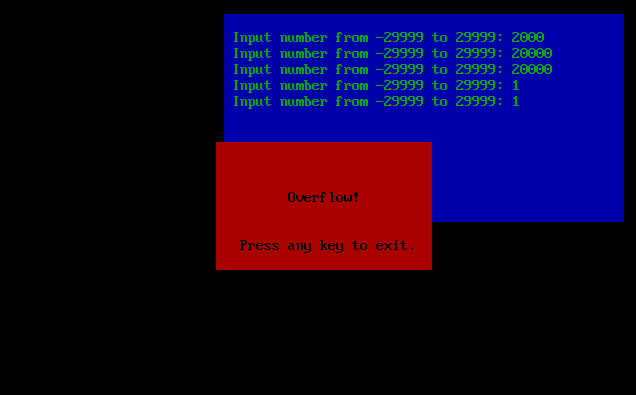
******

*Перемещение окна ввода*

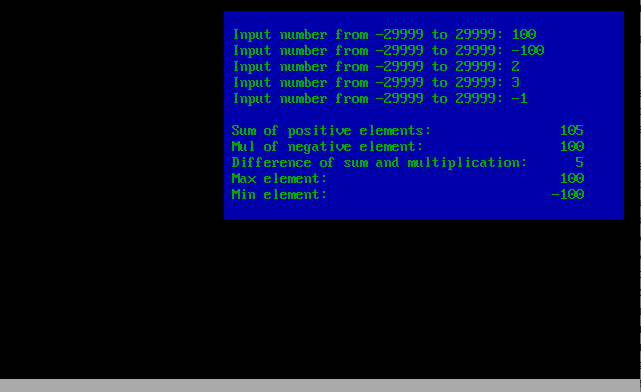
**

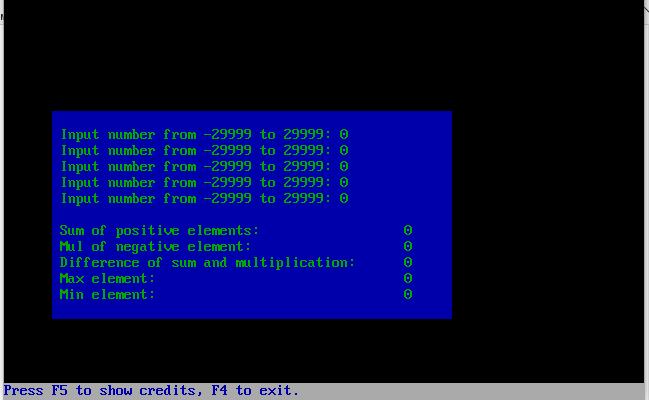
******

*Обработка ошибок*

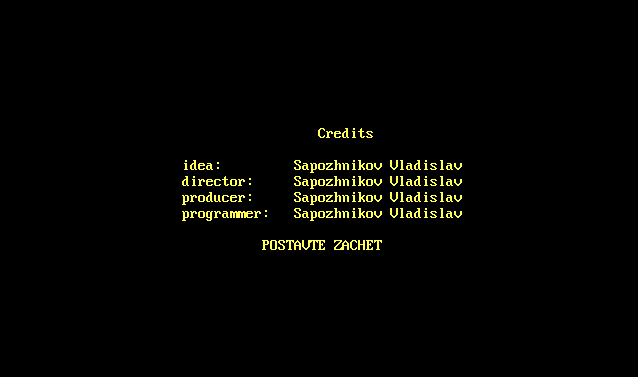
**

*Результаты арифметической обработки*



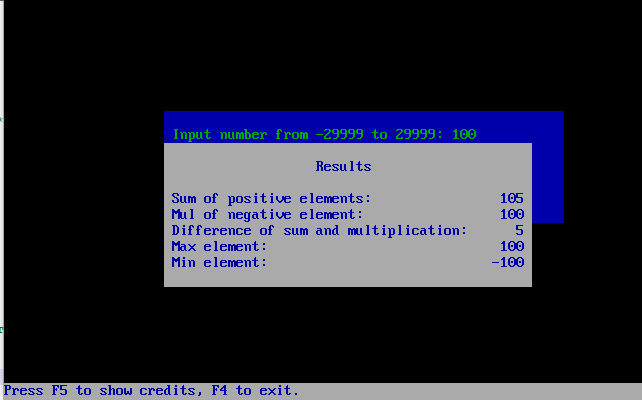


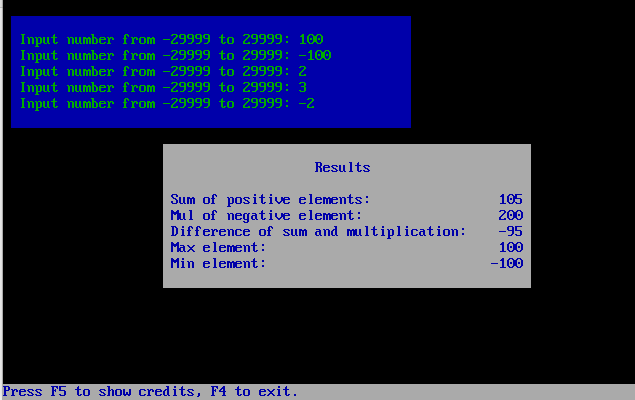
*Титры*



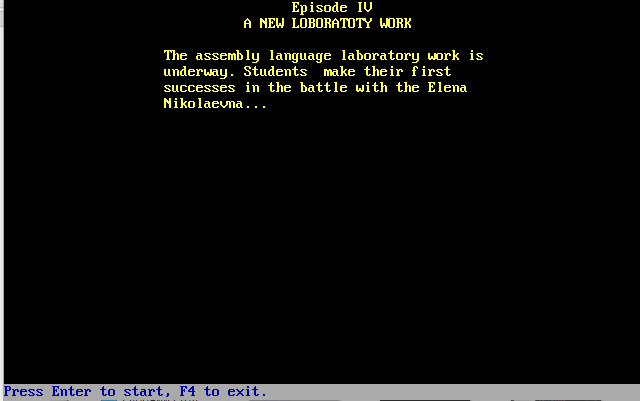
**Исправления**

*Вывод результатов в отдельном окне*

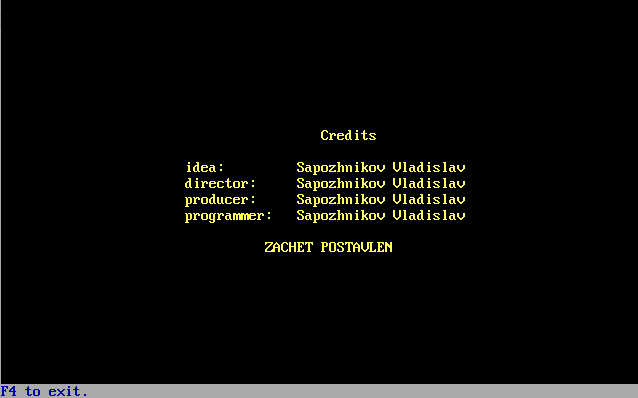




*Елена Николаевна больше не злая*



*Теперь зачет поставлен*



**Вывод:** данная программа имеет ряд ограничений:

1. Она может быть прервана только в определенные моменты (когда об сообщается)

Для решения данной проблемы необходимо отлавливать нажатие определённой клавиши на протяжении работы всей программы.

1. Нет возможности промотать вступление.  
   Для решения данной проблемы в моменты вступления необходимо отлавливать нажатие спец. Клавиши, что невозможно поскольку мы все задержки в программе полностью её блокируют и в эти момент нет возможности вести какие-либо обработки (в однопоточной программе коей она и является)
2. Цикл вывода “приветствия” ужас и требует переработки, однако здравых идей больше нет, скорее всего не хватает знаний.

**Листинг программы**

;Лабораторная работа №4

;Вариант 13 (Здадание из лабораторной №3):

; Вычислить сумму положительных, произведение отрицательных и их разность,

; а также определить минимальное и максимальное число.

d1 SEGMENT para public USE16 'data'

titleBGColor equ 00001110b ;черный фон - желтые буквы

startBGColor equ 00001011b ;черный фон - голубые буквы

waitBgColor equ 11110001b ;серый фон - синие буквы

mainBgColor equ 00010010b ;синий фон - зеленые буквы

errorBgColor equ 01000000b

errorBgColorWait equ 11000000b

videoSeg equ 0b800h

;Давно в далекой далекой галактике....

LongTimeAgo db 'A long time ago in a galaxy far,',10, 24 dup (' '), 'far away...$'

;Ассемблерные войны

Assembler db 'ASSEMBLER$',10,13

Separator db '===========$', 10, 13

;введение с разным кол-ом строк для построчного вывода

FullIntroduction1 db 15 dup (' '),'Episode IV', '$'

FullIntroduction2 db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK', '$'

FullIntroduction3 db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK',10, 13, '$'

FullIntroduction4 db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK',10, 13, 10, 20 dup (' '), 'The assembly language laboratory work is', '$'

FullIntroduction5 db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK',10, 13, 10, 20 dup (' '), 'The assembly language laboratory work is',10, 20 dup(' '),'underway. Students make their first', '$'

FullIntroduction6 db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK',10, 13, 10, 20 dup (' '), 'The assembly language laboratory work is',10, 20 dup(' '),'underway. Students make their first',10, 20 dup(' '),'successes in the battle with the evil', '$'

FullIntroduction db 15 dup (' '),'Episode IV', 10, 30 dup (' '), 'A NEW LOBORATOTY WORK',10, 13, 10, 20 dup (' '), 'The assembly language laboratory work is',10, 20 dup(' '),'underway. Students make their first',10, 20 dup(' '),'successes in the battle with the Elena,10, 20 dup(' '), ' Nikolaevna...', '$'

countYStart db 24d ;окно вступления - начальная строка

countYEnd db 57d ;окно вступления - конечная строка

WaitToStartText db 'Press Enter to start, F4 to exit.$'

WaitToMoveWiindow db 'Use the arrows to move window, press Enter to Start, F4 to exit.$'

WaitAnyKeyToExit db 'Press any key to exit.$'

WaitToReloadOrExit1 db 'Press F5 to show credits, F4 to exit.$'

WaitToExit db 'F4 to exit.$'

Result db 'Results$'

credits db 17 dup (' '), 'Credits',10, 10, 23 dup (' '), 'idea: Sapozhnikov Vladislav', 10, 23 dup (' '), 'director: Sapozhnikov Vladislav', 10, 23 dup (' '), 'producer: Sapozhnikov Vladislav', 10, 23 dup (' '), 'programmer: Sapozhnikov Vladislav', 10, 10, 33 dup (' '), 'ZACHET POSTAVLEN', 10,13, '$'

mainWindowXStart db 6d

mainWindowYStart db 7d

mainWindowXEnd db 55d

mainWindowYEnd db 19d

coursorX db 0

coursorY db 0

inputError db 'Input Error!', 10,10,'$'

overflow db 'Overflow!', 10,10,'$'

sumPosText db 'Sum of positive elements: ','$'

mulNegText db 'Mul of negative element: ','$'

diffResText db 'Difference of sum and multiplication: ','$'

maxElText db 'Max element: ','$'

minElText db 'Min element: ','$'

out\_str db 6 dup (' '),'$'

enter\_please db 'Input number from -29999 to 29999: $'

flag\_err equ 1

in\_str label byte ; Cтрока символов (не более 6)

razmer db 7 ; Размер буфера (6 символов и знак)

kol db (?) ; Количество введеных символов

stroka db 7 dup (?) ; Буфер ввода чисел

number dw 5 dup (0) ; Mассив чисел

sumPos dw 0 ; Сумма положительных чисел

mulNeg dw 0 ; Произведение отрицательных

diffRez dw 0 ; Разность суммы и произведения

min dw 29999 ; Минимальный элемент

max dw -29999 ; Максимальный элемент

siz dw 5 ; Kоличество чисел

d1 ENDS

st1 SEGMENT para stack USE16 'stack'

dw 100 dup (?)

st1 ENDS

;Макрос вывода в окне

; string - текст для вывода

; row - строка вывода

; column - колонка вывода

printInWindow macro string, row, column

push ax

push dx

mov ah,2

mov dh,row

mov dl,column

mov bh,0

int 10h

mov ah, 09h

mov dx, offset string

int 21h

pop dx

pop ax

endm

;Макрос вывода сообщений на экран

; string - строка для вывода

print macro srting

push ax

push dx

mov dx, offset srting

mov ah, 09h

int 21h

pop dx

pop ax

endm

; Макрос вывода строки символов

; string - строка для ввода

input macro srting

push ax

push dx

mov dx, offset srting

mov ah, 0Ah

int 21h

pop dx

pop ax

endm

;Макрос ожидания при помощи функции 86h прерывания Int 15h

; time - время в миллисекундах

sleep macro time

mov al, 0

mov ah, 86h

mov cx, time

int 15h

endm

;Макрос рисования окна

; xStart - левый верхний угол - столбец

; yStart - левый верхний угол - строка

; xEnd - правый нижний угол - столбец

; yEnd - правый нижний угол - строка

drawWindow macro xStart, yStart, xEnd, yEnd, color

mov ah, 06

mov al, 00

mov ch, yStart

mov cl, xStart

mov dh, yEnd

mov dl, xEnd

mov bh, color

int 10h

endm

.386

c1 SEGMENT para public USE16 'code'

ASSUME cs:c1, ds:d1, ss:st1

;Процедура прятания курсора

;устанавливает курсор за пределами окна

hideCursor PROC

mov ah,2

mov dh,26

mov dl,81

mov bh,0

int 10h

ret

ENDP

;Процедура сдвига окна влево

leftShift PROC

cmp mainWindowXStart, 0

je retleftShift

dec mainWindowXStart

dec mainWindowXEnd

retleftShift:

ret

ENDP

; Процедура сдвига окна вправо

rightShift PROC

cmp mainWindowXEnd, 79

je retrightShift

inc mainWindowXStart

inc mainWindowXEnd

retrightShift:

ret

ENDP

; Процедура сдвига окна вверх

upShift PROC

cmp mainWindowYStart, 0

je relupShift

dec mainWindowYStart

dec mainWindowYEnd

relupShift:

ret

ENDP

; Процедура сдвига окна вниз

downShift PROC

cmp mainWindowYEnd, 23

je downshitRet

inc mainWindowYStart

inc mainWindowYEnd

downshitRet:

ret

ENDP

;Старт программы

start:

mov ax, videoSeg

mov es, ax

mov ax, d1

mov ds, ax

;\*\*\*\*\*\*\*\*\*\* Вывод фразы: a long time ago in a galaxy far far away \*\*\*\*\*\*\*\*\*\*\*

drawWindow 0, 0, 80, 25, startBGColor ;на всю консоль черное окно, голубые буквы

printInWindow LongTimeAgo, 11, 24 ;вывод фразы в данном окне

call hideCursor ;прячем курсор

sleep 40 ;ождиание

mov ax, 03h ;очистка экрана

int 10h

drawWindow 0, 0, 80, 25, titleBGColor ;на всю консоль черное окно, желтые буквы

printInWindow Separator, 11, 34 ;вывод

printInWindow Assembler, 12, 35

printInWindow Separator, 13, 34

call hideCursor ;прячем курсор

sleep 60 ;ождиание

mov ax, 03h ;очистка экрана

int 10h

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Циклическая отрисовка вступления \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;Циклическая отрисовка окна с текстом вступления

;окно двигается снизу вверх, кол-во строк меняется

writeLoop:

drawWindow 20, countYStart, 60, countYEnd, titleBGColor

cmp countYStart, 24d

je print1IntroStr

cmp countYStart, 23d

je print2IntroStr

cmp countYStart, 22d

je print3IntroStr

cmp countYStart, 21d

je print4IntroStr

cmp countYStart, 20d

je print5IntroStr

cmp countYStart, 19d

je print6IntroStr

printInWindow FullIntroduction, countYStart, 21

cmp countYStart, 0

je waitToStart

continueLoop:

call hideCursor

dec countYStart

dec countYEnd

sleep 5

loop writeLoop

print1IntroStr:

printInWindow FullIntroduction1, countYStart, 21

jmp continueLoop

print2IntroStr:

printInWindow FullIntroduction2, countYStart, 21

jmp continueLoop

print3IntroStr:

printInWindow FullIntroduction3, countYStart, 21

jmp continueLoop

print4IntroStr:

printInWindow FullIntroduction4, countYStart, 21

jmp continueLoop

print5IntroStr:

printInWindow FullIntroduction5, countYStart, 21

jmp continueLoop

print6IntroStr:

printInWindow FullIntroduction6, countYStart, 21

jmp continueLoop

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Переход от вступления к двиганию окна \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;Ожидание нажатие F4/Enter

waitToStart:

drawWindow 0, 24, 80, 24, waitBgColor

printInWindow WaitToStartText, 24, 0

call hideCursor

mov ah, 00

int 16h

cmp ax, 3E00h

je programEnd

cmp ax, 1C0Dh

je moveWindow

loop waitToStart

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Двигаем окно \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

moveWindow:

mov ax, 03h ;очистка экрана

int 10h

drawWindow mainWindowXStart, mainWindowYStart, mainWindowXEnd, mainWindowYEnd, mainBgColor

drawWindow 0, 24, 80, 24, waitBgColor

printInWindow WaitToMoveWiindow, 24, 0

call hideCursor

mov ah, 00 ;ожидание ввода клавиши

int 16h

cmp ax, 3E00h ;сравнение с ASCII-кодом клавиши F4

je programEnd

cmp ax, 1C0Dh ;сравнение с ASCII и скан кодом клавиши Enter

je startInput

cmp ax, 4B00h ;сравнение с ASCII и скан кодом клавиши 'Стрелочка влево'

je moveLeft

cmp ax, 4D00h ;сравнение с ASCII и скан кодом клавиши 'Стрелочка вправо'

je moveRight

cmp ax, 4800h ;сравнение с ASCII и скан кодом клавиши 'Стрелочка вниз'

je moveUp

cmp ax, 5000h ;сравнение с ASCII и скан кодом клавиши 'Стрелочка вверх'

je moveDown

loop moveWindow

moveLeft:

call leftShift

jmp moveWindow

moveRight:

call rightShift

jmp moveWindow

moveUp:

call upShift

jmp moveWindow

moveDown:

call downShift

jmp moveWindow

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Вводим значения \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

startInput:

mov ax, 03h ;очистка экрана

int 10h

drawWindow mainWindowXStart, mainWindowYStart, mainWindowXEnd, mainWindowYEnd, mainBgColor

mov al, mainWindowXStart

mov coursorX, al

mov al, mainWindowYStart

mov coursorY, al

inc coursorX

xor di,di ;di - номер числа в массиве

mov cx, siz ;cx - размер массива

inputValues:

push cx

inc coursorY

;Вывод сообщения о вводе строки

printInWindow enter\_please, coursorY, coursorX

input in\_str ;Ввод числа в виде строки

call diapazon ;Проверка диапазона вводимых чисел (-29999,+29999)

cmp bh, flag\_err ;Сравним bh и flag\_err

je inErr ;Если равен 1 сообщение об ошибке ввода

call dopust ;Проверка допустимости вводимых символов

cmp bh, flag\_err ;Сравним bh и flag\_err

je inErr ;Если равен 1 сообщение об ошибке ввода

call AscToBin ;Преобразование строки в число

inc di

inc di

pop cx

loop inputValues

jmp searchMinMax

inErr:

drawWindow 27, 9, 53, 16, errorBgColor

drawWindow 27, 15, 53, 16, errorBgColorWait

printInWindow inputError, 12, 34

printInWindow WaitAnyKeyToExit, 15, 30

call hideCursor

mov ah, 00

int 16h

jmp programEnd

;\*\*\*\*\*\*\*\*\*\*\*\* Нахождение миниамльного и максимального элементов \*\*\*\*\*\*\*\*\*\*\*\*

searchMinMax:

mov cx, siz ;cx - размер массива

mov si, offset number

xor ax, ax

searchLoop: ;для поиска минимальных и максимальных

mov ax,[si] ;элементов была принята нулевая гипотеза:

;max = -29999 (минимальный из возможных   
 элементов)

;min = 29999 (максимальный из возможных   
 элементов)

findMax:

cmp ax, max ;если найден элемент больше текущего   
 значения max,

jg foundMax ;то переход к перезаписи max

findMin:

cmp ax, min ;если найден элемент меньше текущего значения min,

jl foundMin ;то переход к перезаписи min

jmp nextVal ;иначе переход к следующему элементу

foundMax:

mov max, ax ;перезапись max элемента

jmp findMin

foundMin:

mov min,ax ;перезапись min элемента

nextVal:

inc si

inc si

loop searchLoop

cmp max, 0

je checkMinZero

jmp searchPosSum

checkMinZero:

cmp min, 0

je resOutput

jmp searchPosSum

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Нахождение суммы положительных элементов \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

searchPosSum:

mov cx, siz ;cx - размер массива

mov si, offset number

xor ax, ax

sumPositive:

mov ax,[si]

cmp ax, 0 ;сравнение с 0

jl negative ;если число меньше то переход

add sumPos,ax ;иначе сложение с переменной

jo overFlowErr ;если переполнение, то переход

negative:

inc si

inc si

loop sumPositive

;\*\*\*\*\*\*\*\*\*\*\*\* Нахождение произведения отрицательных элементов \*\*\*\*\*\*\*\*\*\*\*\*\*

searchNegMul:

mov cx, siz ;cx - размер массива

mov si, offset number

mov ax, 1 ;для циклического умножения заносим 1

;поскольку умножение всегда происходит с   
 регистром ax

minusEl:

mov bx,[si]

cmp bx, 00h

jge plusEl ;если положительный элемент - идем дальше

imul bx ;иначе умножаем

jo overFlowErr ;проверяем на переполнение

plusEl:

inc si

inc si

loop minusEl

mov mulNeg, ax ;заносим значение в переменную

; Нахождение разницы суммы положительных и разницы отрицательных элементов

searchDiffSumMul: ;поскольку вычитание (sub) работает по принципу:

mov ax, sumPos ; <Приемник>=<Приемник>-<Источник>

sub ax, mulNeg ;чтобы не испортить результаты предудыщих вычислений

mov diffRez, ax ;мы одну из перенных переносим в отдельный регистр,   
 где и сохраним

;результат, а затем запишем значение регистра в   
 переменную

jmp resOutput ;переход к выводу результатов

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Вывод ошибок \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

overFlowErr:

drawWindow 27, 9, 53, 16, errorBgColor

drawWindow 27, 15, 53, 16, errorBgColorWait

printInWindow overflow, 12, 36

printInWindow WaitAnyKeyToExit, 15, 30

call hideCursor

mov ah, 00

int 16h

jmp programEnd

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Вывод полученных результатов \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

resOutput:

inc coursorY

inc coursorY

printInWindow sumPosText, coursorY, coursorX

mov ax, sumPos

call BinToAsc

print out\_str

mov cx,6 ;очистка буфера вывода

xor si,si

clear1:

mov [out\_str+si],' '

inc si

loop clear1

inc coursorY

printInWindow mulNegText, coursorY, coursorX

mov ax,mulNeg

call BinToAsc

print out\_str

mov cx,6 ;очистка буфера вывода

xor si,si

clear2:

mov [out\_str+si],' '

inc si

loop clear2

inc coursorY

printInWindow diffResText, coursorY, coursorX

mov ax,diffRez

call BinToAsc

print out\_str

mov cx,6 ;очистка буфера вывода

xor si,si

clear3:

mov [out\_str+si],' '

inc si

loop clear3

inc coursorY

printInWindow maxElText, coursorY, coursorX

mov ax,max

call BinToAsc

print out\_str

mov cx,6 ;очистка буфера вывода

xor si,si

clear4:

mov [out\_str+si],' '

inc si

loop clear4

inc coursorY

printInWindow minElText, coursorY, coursorX

mov ax,min

call BinToAsc

print out\_str

mov cx,6 ;очистка буфера вывода

xor si,si

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Вывод полученных результатов \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

drawWindow 0, 24, 80, 24, waitBgColor

printInWindow WaitToReloadOrExit1, 24, 0

pressWait1:

call hideCursor

mov ah, 00

int 16h

cmp ax, 3E00h

je programEnd

cmp ax, 3F00h

je showCredits

loop pressWait1

showCredits:

drawWindow 0, 0, 80, 24, titleBGColor

printInWindow credits, 8, 23

drawWindow 0, 24, 80, 24, waitBgColor

printInWindow WaitToExit, 24, 0

call hideCursor

pressWait2:

mov ah, 00

int 16h

cmp ax, 3E00h

je programEnd

loop pressWait2

programEnd:

mov ax, 03h ;очистка экрана

int 10h

mov ax, 4c00h ;завершение работы

int 21h

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Проверка диапазона вводимых чисел -29999,+29999 \*

;\* Аргументы: \*

;\* Буфер ввода - stroka \*

;\* \*

;\* Результат: \*

;\* bh - флаг ошибки ввода \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DIAPAZON PROC

xor bh, bh

xor si, si

cmp kol, 05h ; Если ввели менее 5 символов, проверим их допустимость

jb dop

cmp stroka, 2dh ; Eсли ввели 5 или более символов проверим является ли первый минусом

jne plus ; Eсли 1 символ не минус, проверим число символов

cmp kol, 06h ; Eсли первый - минус и символов меньше 6 проверим допустимость символов

jb dop

inc si ; Иначе проверим первую цифру

jmp first

plus:

cmp kol,6 ; Bведено 6 символов и первый - не минус

je error1 ; Oшибка

first:

cmp stroka[si], 32h ; Cравним первый символ с '2'

jna dop ; Eсли первый <= '2' - проверим допустимость символов

error1:

mov bh, flag\_err ; Иначе bh = flag\_err

dop:

ret

DIAPAZON ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Проверка допустимости вводимых символов \*

;\* Аргументы: \*

;\* Буфер ввода - stroka \*

;\* si - номер символа в строке \*

;\* \*

;\* Результат: \*

;\* bh - флаг ошибки ввода \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DOPUST PROC

xor bh, bh

xor si, si

xor ah, ah

xor ch, ch

mov cl, kol ; В (cl) количество введенных символов

m11:

mov al, [stroka + si] ; B (al) - первый символ

cmp al, 2dh ; Является ли символ минусом

jne testdop ; Если не минус - проверка допустимости

cmp si, 00h ; Если минус - является ли он первым символом

jne error2 ; Если минус не первый - ошибка

jmp m13

testdop:

cmp al, 30h ;Является ли введенный символ цифрой

jb error2

cmp al, 39h

ja error2

m13:

inc si

loop m11

jmp m14

error2:

mov bh, flag\_err ; При недопустимости символа bh = flag\_err

m14:

ret

DOPUST ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* ASCII to number \*

;\* Аргументы: \*

;\* B cx количество введенных символов \*

;\* B bx - номер символа начиная с последнего \*

;\* \*

;\* Результат: \*

;\* Буфер чисел - number \*

;\* B di - номер числа в массиве \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AscToBin PROC

xor ch, ch

mov cl, kol

xor bh, bh

mov bl, cl

dec bl

mov si, 01h ; В si вес разряда

n1:

mov al, [stroka + bx]

xor ah, ah

cmp al, 2dh ; Проверим знак числа

je otr ; Eсли число отрицательное

sub al, 30h

mul si

add [number + di], ax

mov ax, si

mov si, 10

mul si

mov si, ax

dec bx

loop n1

jmp n2

otr:

neg [number + di] ; Представим отрицательное число в дополнительном коде

n2:

ret

AscToBin ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Number to ASCII \*

;\* Аргументы: \*

;\* Число передается через ax \*

;\* \*

;\* Результат: \*

;\* Буфер чисел - out\_str \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BinToAsc PROC

xor si, si

add si, 05h

mov bx, 0Ah

push ax

cmp ax, 00h

jnl mm1

neg ax

mm1:

cwd

idiv bx

add dl,30h

mov [out\_str + si], dl

dec si

cmp ax, 00h

jne mm1

pop ax

cmp ax, 00h

jge mm2

mov [out\_str + si], 2dh

mm2:

ret

BinToAsc ENDP

c1 ENDS

end start