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 № 3

Разработка программы ввода-вывода и обработки   
числовой информации

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**

Приобретение навыков: разработки программы ввода-вывода десятичных чисел со знаком, логической и арифметической обработки введённых чисел, контроль ситуаций переполнения и деления на 0.

**Задание**

Согласно варианту написать программу, выполняющую:

- ввод массива целых многозначных десятичных чисел со знаком;

- арифметическую и логическую обработку массива чисел (по вариантам);

- преобразование числового результата в ASCII-строку и отображение результата на экране.

В программе должен быть предусмотрен контроль ситуаций переполнения и деления на 0 с выводом на экран соответствующих сообщений.

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

**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

**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

В коде программы имеются следующие макросы:

* print - выводит указанную строку
* input - ввод с клавиатуры символов и сохранение в переданной строке

В коде программы имеются следующие процедуры:

* DIAPAZON – проверка диапазона вводимых чисел
* DOPUST – проверка допустимости вводимых символов
* AscToBin – перевод ASCII кодов чисел в двоичное число
* BinToAsc – обратная AscToBin

**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. Поскокуль 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 функции прерываний**

В программе используются следующие функции прерываний:

**int 10h** – функция стандартного видеосервиса ROM-BIOS, в программе она используется с ah=0(номер функции) – очистка экрана, установить поля BIOS, установить режим; al = 3 (номер режима).

mov ax, 03h ;Установка текстового видеорежима, очистка экрана

int 10h ;ah=0 (номер функции),al=3 (номер режима)

**int 21h –** вызов функций DOS.

09h – функция вывода строки

mov dx, offset srting

mov ah, 09h

int 21h

0Ah – извлечение буфера с пользовательским вводом

mov dx, offset srting

mov ah, 0Ah

int 21h

**Листинг программы**

;Лабораторная работа №3

;Вариант 13:

; Вычислить сумму положительных, произведение отрицательных и их разность,

; а также определить минимальное и максимальное число.

; Макрос вывода сообщений на экран

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

input macro srting

push ax

push dx

mov dx, offset srting

mov ah, 0Ah

int 21h

pop dx

pop ax

endm

d1 SEGMENT para public 'data'

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

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

kol db (?) ; Количество введённых символов

stroka db 7 dup (?) ; Буфер ввода чисел

number dw 5 dup (0) ; Массив 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 ; Kколичество чисел

lab3 db 'Laboratory work #3','$'

author db 13,10, 'Sapozhnikov Vladislav 19-V-1'

perevod db 10,13,'$'

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

divByZeroError db 'Divition by zero!', 10,10,'$'

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

sumPosText db 13,10, 'Sum of positive elements: ','$'

mulNegText db 13,10, 'Mul of negative element: ','$'

diffResText db 13,10,'Difference of sum and multiplication: ','$'

maxElText db 13,10, 'Max element: ','$'

minElText db 13,10, 'Min element: ','$'

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

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

flag\_err equ 1

d1 ENDS

st1 SEGMENT para stack 'stack'

dw 100 dup (?)

st1 ENDS

;Точка входа в программу

c1 SEGMENT para public 'code'

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

start:

mov ax, d1

mov ds, ax

mov ax, 03h ;Установка текстового видеорежима, очистка экрана

int 10h ;ah=0 (номер функции),al=3 (номер режима)

print lab3 ;вывод сообщения с номером работы

print author ;вывод автора работы

print perevod ;пропуск строки

;\*\*\*\*\*\*\*\*\*\*\* Ввод элементов с проверкой \*\*\*\*\*\*\*\*\*\*\*

xor di,di ;di - номер числа в массиве

mov cx, siz ;cx - размер массива

inputVal:

push cx

check:

print enter\_please ;Вывод сообщения о вводе строки

input in\_str ;Ввод числа в виде строки

print perevod ;Переход на новую строку

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 inputVal

jmp searchMinMax

inErr:

print inputError

jmp check

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

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

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

overFlowErr:

print overflow ;вывод сообщения о переполнении

jmp progend

zero:

print divByZeroError ;вывод сообщения о делении на ноль

jmp progend

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

resOutput:

print sumPosText

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

print mulNegText

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

print diffResText

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

print maxElText

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

print minElText

mov ax,min

call BinToAsc

print out\_str

mov cx,6 ;очистка буфера вывода

xor si,si

jmp progend

progend:

mov ax,4c00h

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:

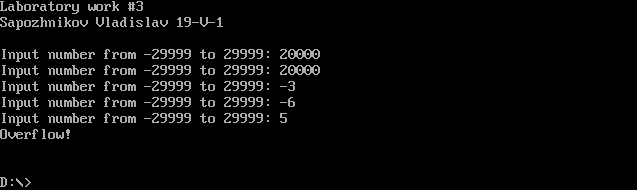
ret

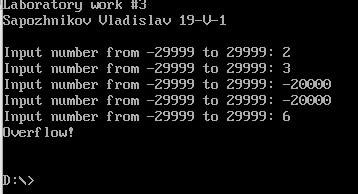
BinToAsc ENDP

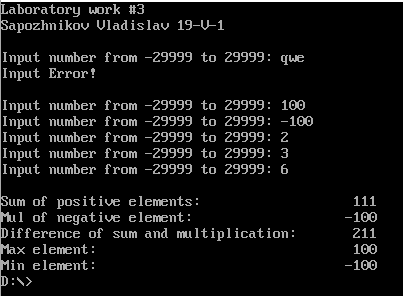
c1 ENDS

end start

**Результаты работы**







**Теоретическая часть**

**Вопрос 9. Функции прерывания 16h BIOS для ввода с клавиатуры.**

Клавиатура является основным способом взаимодействия с компьютером. Для ввода информации с клавиатуры можно использовать либо функции операционной системы, либо прямой опрос контроллера клавиатуры. Функции DOS имеет два серьезных недостатка:

1. Они не позволяют полностью реализовать возможности функциональных клавиш клавиш
2. Они предназначены для работы в режиме терминала. В процессе считывания символа они выполняют ряд дополнительных операций, что является не удобным для использования их в режиме отличного от терминального.

Функции BIOS обладают гораздо более широкими возможностями, которых достаточно для выполнения любых операций реальном режиме работы процессора. Вызов клавиатурных функций BIOS выполняется по прерыванию Int 16h.

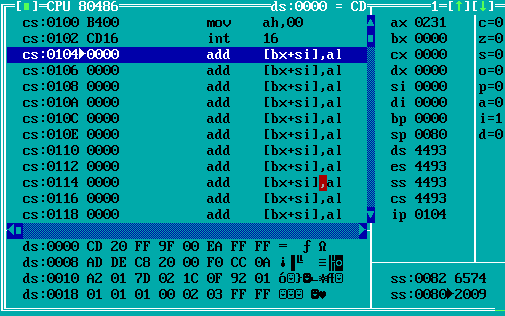
**Функция 00h: прочитать данные с клавиатуры**

Перед вызовом прерывания требуется записать в регистр AH значение функции: 00h

После выполнения функции в регистр будет помещена следующая информация:

* AH – скан-код символа
* AL – ASCII-код символа

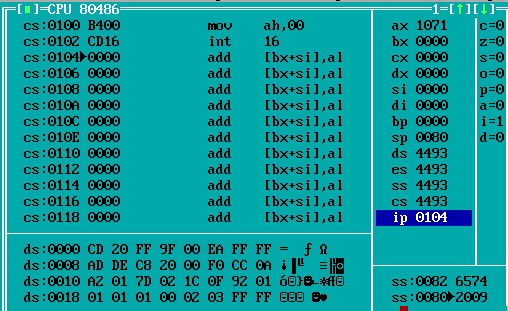
Нажатие клавиши 1 (основной клавиатуры, не numpad)



AH: 02h – скан код символа 1

AL: 31h - ASCII код символа 1

Нажатие клавиши q

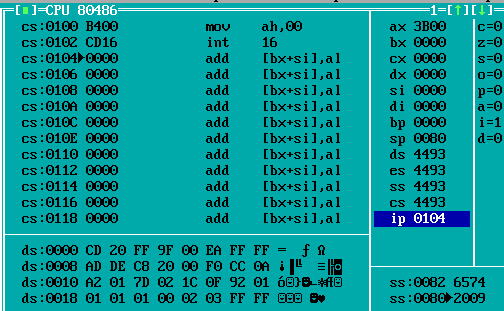


AH: 10h – скан код символа q

AL: 71h - ASCII код символа q

При нажатии управляющих и функциональных клавиш функция выдает ASCII-код со значением 0, благодаря чему их можно отличить от алфавитно-цифровых. Однако таким способом функциональные клавиши можно отличать лишь по скан-кодам.

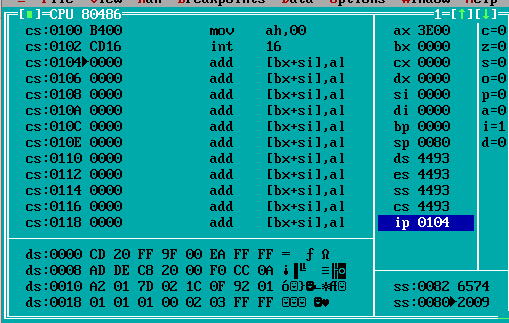
Нажатие клавиши F1



AH: 3Bh – скан код символа F1

AL: 00 - ASCII код для функциональных клавиш

Нажатие клавиши F4



AH: 3Eh – скан код символа F1

AL: 00 - ASCII код для функциональных клавиш

**Функция 00h: прочитать данные с клавиатуры**

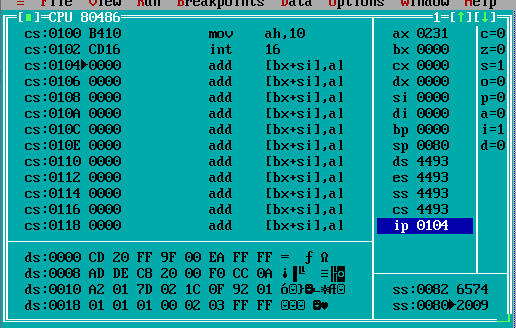
Функция 10h является усовершенствованным вариантом функции 00h. Она позволяет получить скан-коды функциональных клавиш F1-F12, а также клавиш управления курсором.

Перед вызовом прерывания требуется записать в регистр AH значение функции: 10h

После выполнения функции в регистр будет помещена следующая информация:

* AH – расширенный скан-код символа
* AL – ASCII-код символа

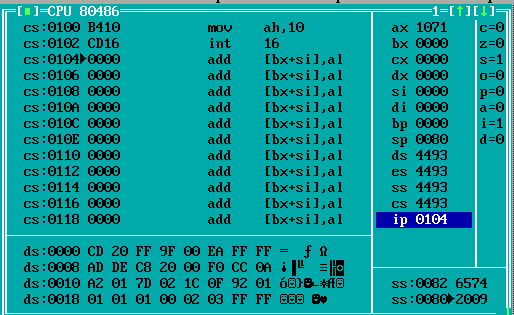
Нажатие клавиши 1 (основной клавиатуры, не numpad)



AH: 02h – скан код символа 1

AL: 31h - ASCII код символа 1

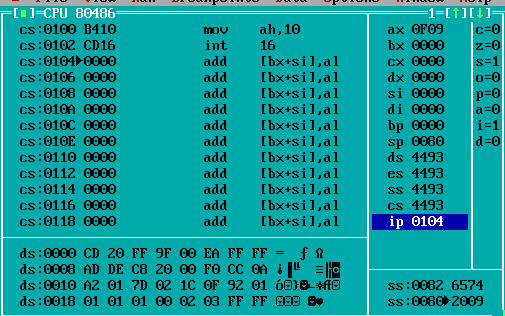
Нажатие клавиши q



AH: 10h – скан код символа q

AL: 71h – ASCII код символа q

Нажатие управляющей клавиши tab



AH: 0Fh – скан код клавиши enter

AL: 09h – ASCII код клавиши tab

Прерывание Int 16h имеет еще несколько функций:

* 01h – получить состояние клавиатуры
* 02h – получить состояние флагов клавиатуры
* 03h – управление режимом автоповтора
* 04h – вкл/выкл звуковой сигнал клавиш
* 05h – поместить символ в буфер клавиатуры
* 11h – получить состояние расширенной клавиатуры
* 12h – получить состояние флагов расширенной клавиатуры

Однако их рассмотрение выходит за рамки данного теоретического вопроса.

**Вывод**

В ходе выполнения данной работы были получены навыки разработок программ ввода-вывода десятичных чисел со знаком, логической и арифметической обработки введенных чисел со знаком, контроль ситуации переполнения и деления на 0.