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

ОТЧЕТ

по лабораторной работе № 4

«Разработка программы ввода-вывода и обработки числовой информации»

Вариант №19

по дисциплине

Принципы и методы
организации системных программных средств

РУКОВОДИТЕЛЬ:

_____ Викулова Е. Н.

СТУДЕНТ:

_____ Сухоруков В. А.

___19-В-2 .

Работа защищена «___» _____

С оценкой _____

Нижний Новгород 2021

Оглавление

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 19: Определить минимальный и максимальный элементы массива, среднее положительных чисел, сумму отрицательных.

Структура программы

Разработана программа является много сегментной и имеет расширение exe. При создании программы была использована модель small. В программе имеет 3 сегмента:

- Сегмент данных .data, связанный с сегментным регистром ds
- Сегмент стека .stack, связанный с сегментным регистром ss.
- Сегмент кода .code, связанный с сегментным регистром cs;

В коде программы имеются следующие макросы:

- **printInWindow** – макрокоманда вывода сообщений в определённую область. В качестве параметров принимает сообщение для ввода, строку и столбец куда следует установить курсор.
- **input** – макрокоманда ввода сообщения в текущем положение курсора
- **drawWindow** – отрисовка области по заданным координатам и с заданными цветами фона и букв при помощи функции 06h прерывания Int 10h
- **drawSnowman** – С помощью printInWindow рисует снеговика.

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
- leftShift – “сдвиг” координат окна влево
- rightShift – “сдвиг” координат окна вправо
- upShift – “сдвиг” координат окна вверх
- downShift – “сдвиг” координат окна вниз

Макросы

drawWindow- макрос рисования окна

```
;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              ;левый верхний угол - строка
     mov cl, xStart              ;левый верхний угол - столбец
     mov dh, yEnd                ;правый нижний угол - строка
     mov dl, xEnd                ;правый нижний угол - столбец
     mov bh, color               ;установка цвета фона и цвета букв
     int 10h                     ;прерывание отрисовки
endm
```

PrintInWindow- макрос рисования окна

```
;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
```

Input- макрос ввода строки символов

```
; Макрос ввода 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
```

DrawSnowman- макрос рисования снеговика

```
; Макорс рисования снеговика
drawSnowman macro
     printInWindow Snowman1, 6, 0
     printInWindow Snowman2, 7, 0
     printInWindow Snowman2, 8, 0
     printInWindow Snowman2, 9, 0
     printInWindow Snowman1, 10, 0
     printInWindow Snowman3, 11, 0
     printInWindow Snowman4, 12, 0
     printInWindow Snowman5, 13, 0
     printInWindow Snowman6, 14, 0
     printInWindow Snowman7, 15, 0
     printInWindow Snowman8, 16, 0
     printInWindow Snowman9, 17, 0
     printInWindow Snowman10, 18, 0
     printInWindow Snowman11, 19, 0
     printInWindow Snowman12, 20, 0
     printInWindow Snowman13, 21, 0
     printInWindow Snowman13, 22, 0
     printInWindow Snowman14, 23, 0
     printInWindow Snowman15, 24, 0
     printInWindow Snowman16, 25, 0
     printInWindow Snowman17, 26, 0
     printInWindow Snowman17, 27, 0
     printInWindow Snowman17, 28, 0
     printInWindow Snowman17, 29, 0
     printInWindow Snowman17, 30, 0
     printInWindow Snowman17, 31, 0
     printInWindow Snowman16, 32, 0
     printInWindow Snowman15, 33, 0
endm
```

Процедуры

Для изменения положения окна в консоли были использованы следующие процедуры:

leftShift PROC

```
;Процедура сдвига окна влево
leftShift PROC
     cmp mainWindowXStart, 0
     je retleftShift
     dec mainWindowXStart
     dec mainWindowXEnd
retleftShift:
     ret
ENDP
```

rightShift PROC

```
;Процедура сдвига окна вправо
rightShift PROC
     cmp mainWindowXStart, 0
     je rettightShift
     inc mainWindowXStart
     inc mainWindowXEnd
rettightShift:
     ret
ENDP
```

upShift PROC

```
;Процедура сдвига окна вверх
upShift PROC
     cmp mainWindowYStart, 0
     je retupShift
     dec mainWindowYStart
     dec mainWindowYEnd
retupShift:
     ret
ENDP
```

downShift PROC

```
;Процедура сдвига окна вниз
downShift PROC
     cmp mainWindowYStart, 0
     je retdownShift
     inc mainWindowYStart
     inc mainWindowYEnd
retdownShift:
     ret
ENDP
```

DIAPAZON PROC

Для 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.

```
DIAPAZON PROC
;проверка диапазона вводимых чисел -29999,+29999
;буфер ввода - stroka
;через bh возвращается флаг ошибки ввода
     xor bh,bh;
     xor si,si               ;номер символа в вводимом числе
                        ;если ввели менее 5 символов проверим их допустимость
     cmp kol,5
     jb dop
;если ввели 5 или более символов проверим является ли первый минусом
     cmp stroka,2dh
     jne plus                ;если 1 символ не минус, проверим число символов
;если первый - минус и символов меньше 6 проверим допустимость символов
     cmp kol,6
     jb dop
     inc si                  ;иначе проверим первую цифру
     jmp first

plus:   cmp kol,6            ;введено 6 символов и первый - не минус
     je error1               ; ошибка

first:  cmp stroka[si],32h   ;сравним первый символ с 2
     jna dop                 ;если первый <=2 -проверим допустимость символов
error1:   mov bh,flag_err    ;иначе bh=flag_err
dop: ret
DIAPAZON ENDP
```

DOPUST PROC

Проверки на допустимость 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
     ;проверка допустимости вводимых символов
     ;буфер ввода - stroka
     ;si - номер символа в строке
     ;через bh возвращается флаг ошибки ввода
          xor bh,bh
          xor si,si
          xor ah,ah
          xor ch,ch
          mov cl,kol         ;в ch количество введенных символов
     m11: mov al,[stroka+si] ; в al - первый символ
          cmp al,2dh         ;является ли символ минусом
          jne testdop        ;если не минус - проверка допустимости
          cmp si,0           ;если минус  - является ли он первым символом
          jne error2         ;если минус не первый -ошибка
          jmp m13
     ;является ли введенный символ цифрой
```

```
testdop:cmp al,30h
     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
```

AscToBin PROC

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`.

```
AscToBin PROC
;в cx количество введенных символов
;в bx - номер символа начиная с последнего
;буфер чисел - number, в di - номер числа в массиве
     xor ch,ch
     mov cl,kol
     xor bh,bh
     mov bl,cl
     dec bl
     mov si,1       ;в si вес разряда
n1:  mov al,[stroka+bx]
     xor ah,ah
     cmp al,2dh     ;проверим знак числа
     je otr         ;если 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
;представим отрицательное число в дополнительном коде
otr: neg [number+di]
n2:  ret
AscToBin ENDP
```

BinToAsc PROC

Для вывода полученных результатов переводим числа в их ASCII коды, т.е. выполняем обратную операцию.

```
BinToAsc PROC
;преобразование числа в строку
;число передается через ax
     xor si,si
     add si,5
     mov bx,10
     push ax
     cmp ax,0
     jnl mm1
     neg ax
mm1: cwd
     idiv bx
     add dl,30h
     mov [out_str+si],dl
     dec si
     cmp ax,0
     jne mm1
     pop ax
     cmp ax,0
     jge mm2
     mov [out_str+si],2dh
mm2: ret
BinToAsc ENDP
```

Алгоритмы арифметической обработки двоичных чисел

Нахождение минимального и максимального элемента.

Для нахождение максимального и минимального элементов формируем нулевую гипотезу: в max и min помещаем первое число.

Алгоритм

Цикл-Для каждого элемента массива:
- Условие: элемент > max ?
  - Да: max = элемент
  - Нет: идем дальше
- Условие: элемент < min ?
  - Да: min = элемент

Всё-цикл

```
;Поиск максимального и минимального элементов
          mov ax,number
          mov maxnum,ax
          mov minnum,ax

max:
          mov ax,number+si
          cmp ax,maxnum
          jle min
          mov maxnum,ax
          jmp min

min: cmp ax,minnum
          jge next
          mov minnum,ax
          jmp next

next:     inc si
          inc si
          loop max
```

Нахождение отрицательных элементов и расчёт их суммы

Алгоритм

Цикл-Для каждого элемента массива:
- Условие: элемент <0 ?
  - Да: Сумма_отрицательных += элемент
  - Нет: переход к следующему элементу
- Проверка переполнения

Всё-цикл

```
; Поиск отрицательных элементов и рассчет их суммы
          mov cx, siz          ; В (cx) - размер массива
```

```
          mov si, offset number
negFind:
          mov ax, [si]
          cmp ax, 0
          jge endNegFind
          add negSum, ax      ; Сумма отрицательных
          jo  OVR             ; Если произошло переполнение

endNegFind:
          inc si
          inc si
          loop negFind
```

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

Нахождение суммы положительных чисел

Алгоритм

Цикл-Для каждого элемента массива:
Условие: элемент > 0 ?
Да: Сумма_положительных += элемент
Количество_положительных+=1
Нет: переход к следующему элементу
Проверка переполнения
Всё-цикл

```
; Поиск положительных элементов  и рассчет их суммы
          mov cx, siz         ; В (cx) - размер массива
          mov si, offset number
          xor bx, bx

posFind:
          mov ax, [si]
          cmp ax, 0
          jl endPosFind
          add posSum, ax      ; Сумма положительных
          jo  OVR             ; Если произошло переполнение
          inc bx              ; Считаем количество положительных элементов


endPosFind:
          inc si
          inc si
          loop posFind
```

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

Нахождение среднего положительных чисел

Алгоритм

1. Заносим аккумулятор значение суммы положительных чисел.
2. Проверяем количество положительных чисел, если оно больше нуля, то переходим к пункту 3, иначе выходим из алгоритма.
3. Делим содержимое аккумулятора на количество положительных чисел, сохранённое в регистре bx.
4. Сохраняем полученное значение в переменной posAvg.

```
; Поиск среднего положительного
        mov dx,0
        mov ax,posSum
        cmp bx,0
        je endAvg
        div bx
        mov posAvg, ax
endAvg:
        jmp resOutput
```

Используемые функции прерываний

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
←	4B00h
→	4D00h
↑	5000h
↓	4800h

Int 21h

09h функция – Выдать строку

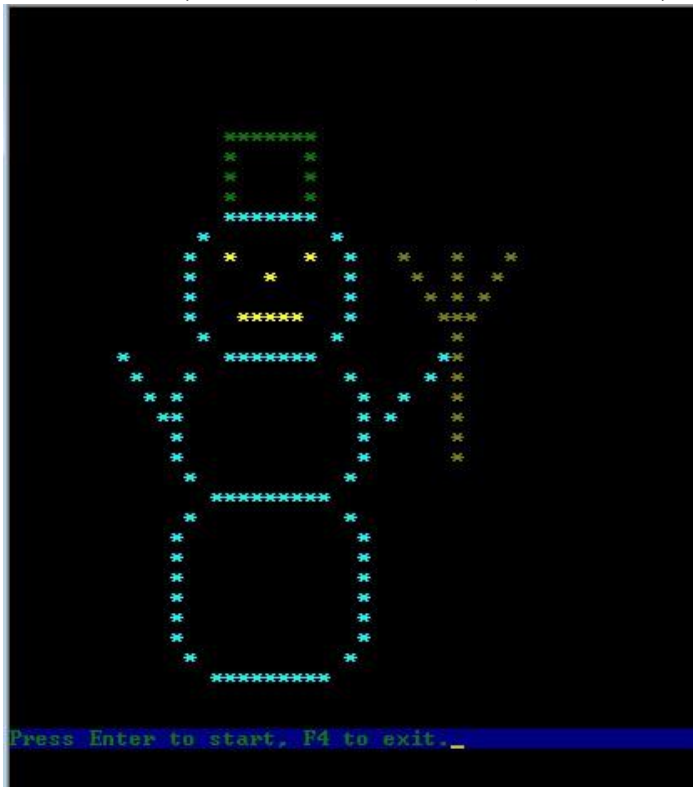
DS:DX – адрес строки, заканчивающейся символом ‘$’

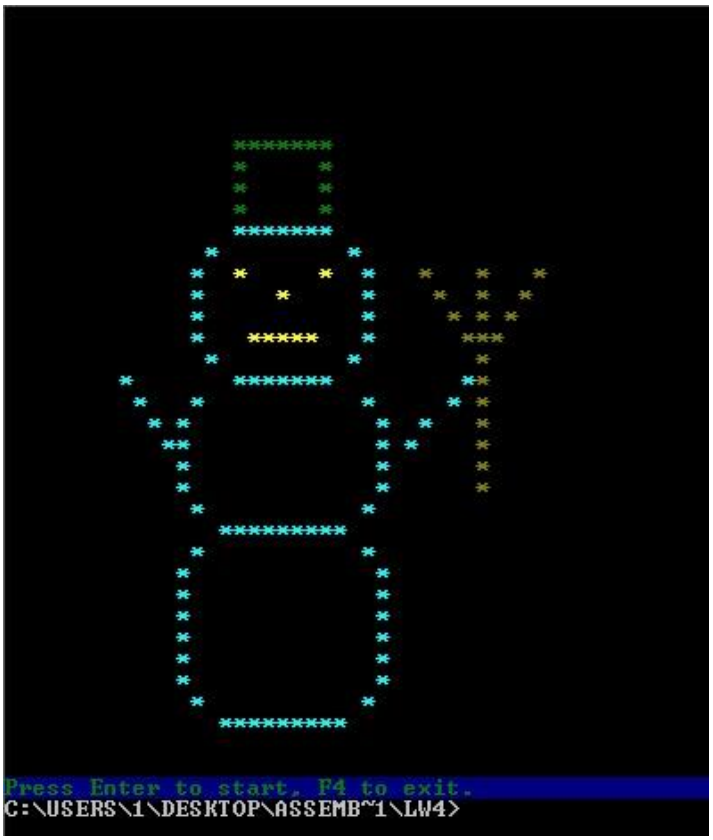
0ah функция – Ввод строки в буфер

DS:DX – адрес входного буфера.

Результаты работы

Стартовое окно – снеговик. С этого окна можно перейти к режиму передвижения главного окна, с помощью Enter, или выйти, с помощью F4.



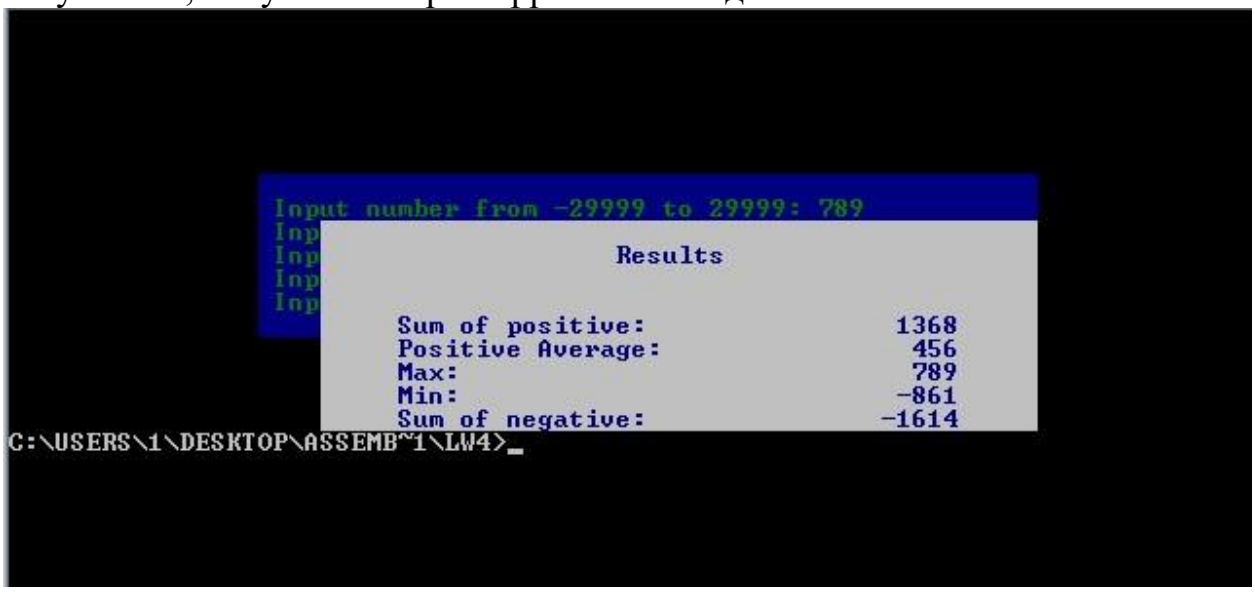


Использование стрелок для изменения положения окна ввода.

Обработка некорректного ввода.

Обработка переполнения.

Результаты, получаемые при корректном вводе.



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.