**Содержание**

[Введение 4](#_Toc419324114)

[1 Теоритическая часть 5](#_Toc419324115)

[1.1 Постановка задачи 5](#_Toc419324116)

[1.2 Обоснование выбора инструментальных средств 5](#_Toc419324117)

[2 Практическая часть 6](#_Toc419324118)

[2.1 Проектирование структурной схемы алгоритма 6](#_Toc419324119)

[2.1.1 Структурная схема алгоритма 6](#_Toc419324120)

[2.1.2 Описание структурной схемы алгоритма 6](#_Toc419324121)

[2.2 Разработка программы на языке Ассемблера 7](#_Toc419324122)

[2.2.1 Листинг программы 7](#_Toc419324123)

[2.2.2 Описание программы 17](#_Toc419324124)

[Заключение 18](#_Toc419324125)

[Список использованных источников 19](#_Toc419324126)

# **Введение**

Язык ассемблера  — машинно-ориентированный [язык низкого уровня](https://ru.wikipedia.org/wiki/%D0%9D%D0%B8%D0%B7%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) с командами, обычно соответствующими [командам машины](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8), который может обеспечить дополнительные возможности вроде [макрокоманд](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%B0); автокод, расширенный конструкциями [языков программирования высокого уровня](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), такими как выражения, макрокоманды, средства обеспечения [модульности программ](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)).

Язык ассемблера — система обозначений, используемая для представления в удобочитаемой форме программ, записанных в машинном коде. Язык ассемблера позволяет программисту пользоваться алфавитными мнемоническими кодами операций, по своему усмотрению присваивать символические имена [регистрам ЭВМ](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0) и памяти, а также задавать удобные для себя схемы [адресации](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8) (например, индексную или косвенную). Кроме того, он позволяет использовать различные системы счисления (например, [десятичную](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%81%D1%8F%D1%82%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) или [шестнадцатеричную](https://ru.wikipedia.org/wiki/%D0%A8%D0%B5%D1%81%D1%82%D0%BD%D0%B0%D0%B4%D1%86%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)) для представления числовых констант и даёт возможность помечать строки программы метками с символическими именами с тем, чтобы к ним можно было обращаться (по именам, а не по адресам) из других частей программы (например, для передачи управления)[.

Перевод программы на языке ассемблера в исполнимый [машинный код](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) (вычисление выражений, раскрытие макрокоманд, замена мнемоник собственно машинными кодами и символьных адресов на [абсолютные](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D1%80%D0%B5%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)#.D0.B0.D0.B1.D1.81.D0.BE.D0.BB.D1.8E.D1.82.D0.BD.D1.8B.D0.B9) или [относительные адреса](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D1%80%D0%B5%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)#.D0.BE.D1.82.D0.BD.D0.BE.D1.81.D0.B8.D1.82.D0.B5.D0.BB.D1.8C.D0.BD.D1.8B.D0.B9)) производится [ассемблером](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80) — программой-транслятором, которая и дала языку ассемблера его название.

# **1 Теоритическая часть**

## 1.1 Постановка задачи

Разработать драйвер клавиатуры, работающий в реальном режиме (под DOS), заменяющий стандартный драйвер, выполняющий все функции стандартного, а также (в соответствии с вариантом): осуществлять сдвиг и вращение графического объекта, выводимого на экран.

## 1.2 Обоснование выбора инструментальных средств

Для выполнения данной курсовой работы была выбрана среда разработки Turbo Assembler. Эта среда доказала свою пригодность для выполнения учебных практических работ.

# **2 Практическая часть**

## 2.1 Проектирование структурной схемы алгоритма

## 2.1.1 Структурная схема алгоритма

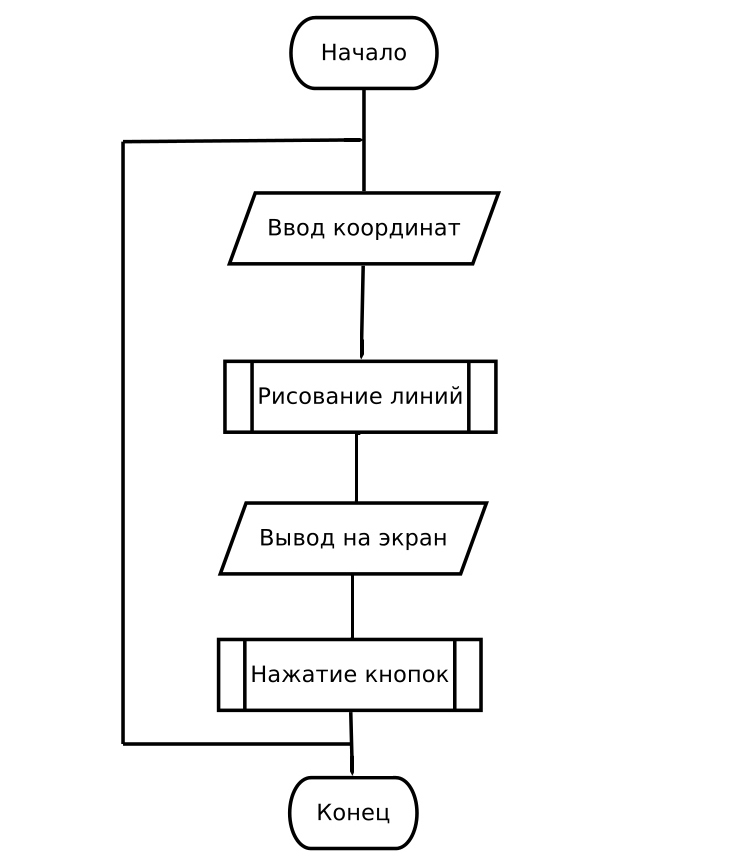


Рисунок 1 - Структурная схема

## 2.1.2 Описание структурной схемы алгоритма

Ввод координат. Из файла crane.obj получает координаты вершин фигуры и записываем их в стэк.

Рисование линий. Берём координаты для вершин и проводим линию, соединяющую их. Делам это для всех связанных вершин.

Вывод на экран. Отображаем на экране вершины и рёбра фигуры.

Нажатие кнопок. Отслеживаем нажатие определённой кнопки и выполняем смещение фигуры.

## 2.2 Разработка программы на языке Ассемблера

## 2.2.1 Листинг программы

Программа:

model tiny, C ; Этот параметр определяет модель сегментации памяти для программного модуля.

;Код, данные и стек объединены в одну группу с именем DGROUP и размером до 64Кб. СS=DS=SS=DGROUP.

;Второй параметр C указывает, что при вызове ассемблерной процедуры из другой программы будет задействован способ передачи параметров, принятый в языке C.

;Структура для хранения одной вершины

POINT3D struc

xx dw ?

yy dw ?

zz dw ?

POINT3D ends

;структура для хранения координат вершины на плоскости

POINT2D struc

xx dw ?

yy dw ?

POINT2D ends

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_МАКРОC ВЫЗОВА РИСОВАНИЯ ЛИНИИ ПО НОМЕРАМ ВЕРШИН

PRLINE macro num1, num2, colour

mov bx, num2

shl bx, 2 ;сдвиг влево

push colour

push [POINT2D ptr bx+si].yy

push [POINT2D ptr bx+si].xx ;записываем в стек координаты точки

mov bx, num1

shl bx, 2

push [POINT2D ptr bx+si].yy

push [POINT2D ptr bx+si].xx

call Line

add sp, 10

endm

.386 ; инициализация сопроцессора (нужно для команды fsincos)

.data

Rad dd 180. ;180 градусов - для преобразования в радианы

include crane.obj

;временные переменные для вычисления координат при вращении

X DW ?

Y DW ?

Z DW ?

firstcadr db ?

;углы поворота относительно осей

XAngle DW 0

YAngle DW 0

ZAngle DW 0

; смещение по координатам

DeltaX DW 0

DeltaY DW 0

DeltaZ DW 0

;переменные, используемые для расчета координат на эране

Xoff DW 400 ;пропорции по X

Yoff DW 400 ; и Y

Zoff DW 180 ;расстояние от глаз

Nx DW 160 ;середина экрана

Ny DW 100

.code ;сегмент кода

.startup ;точка входа

mov ax, 0013h ;vga 320x200x256

int 10h

mov ax, @DATA ;@data Физический адрес сегмента данных типа near

mov ds, ax ;инициируем сегмент данных

mov ax, 0a000h ;настроим дополнительный сегмент ES на участок видео памяти 0A000h

mov es, ax ;es - сегмент видео

mov FirstCadr, 0h

call MainProgram

MainLoop:

in al,60h ;проверка нажатия

;----------------------- СДВИГ ПО ОСИ Z

push ax

cmp al, 10h ;Если клавиша "A"

jne DONTROTSZ

mov ax, 1

sub DeltaZ, ax ;смещаем на 2 у.е.

jmp KEYPress

DONTROTSZ:

pop ax

push ax

cmp al, 12h ;Если клавиша "D"

jne DONTROTSZ2

mov ax, 1

add DeltaZ, ax ;смещаем на 2 у.е.

jmp KEYPress

DONTROTSZ2:

pop ax

;----------------------- СДВИГ ПО ОСИ X

push ax

cmp al, 1Eh ;Если клавиша "A"

jne DONTROTSX

mov ax, 1

sub DeltaX, ax ;смещаем на 2 у.е.

jmp KEYPress

DONTROTSX:

pop ax

push ax

cmp al, 20h ;Если клавиша "D"

jne DONTROTSX2

mov ax, 1

add DeltaX, ax ;смещаем на 2 у.е.

jmp KEYPress

DONTROTSX2:

pop ax

;----------------------- СДВИГ ПО ОСИ Y

push ax

cmp al, 11h ;Если клавиша "W"

jne DONTROTSY

mov ax, 1

sub DeltaY, ax ;смещаем на 2 у.е.

jmp KEYPress

DONTROTSY:

pop ax

push ax

cmp al, 1Fh ;Если клавиша "S"

jne DONTROTSY2

mov ax, 1

add DeltaY, ax ;смещаем на 2 у.е.

jmp KEYPress

DONTROTSY2:

pop ax

KEYPress:

call MainProgram ;отрисовка кадра

DONTROTD:

pop ax

cmp al, 01h ;ESCAPE

jne MainLoop ;если нет, повтор

mov ax,4c00h ;выход

int 21h ;в дос

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПОВОРОТ ТРЕХМЕРНЫХ КООРДИНАТ НА ОСНОВЕ ТЕКУЩИХ ЗНАЧЕНИЙ УГЛОВ ПОВТОРА

Rotation PROC uses si di, coord1:word, coord2:word, angle:word

;преобразуем градусы в радианы

fild angle ;угол записываем в st(0)

fldpi ;st(0)=пи st(0)->st(1)

fmulp ;угол\*пи st(0)=st(0)\*st(1)

fdiv Rad ;угол\*пи/180 st(0)=st(0)/Rad

;найдем sin и cos

fsincos ;st(1)=sin, st(0)=cos

;адреса координат

mov di, coord1

mov si, coord2

fild word ptr [di] ;первая координата (x) st(0)=x st(1)=cos st(2)=sin st(3)=angel

fmul st, st(1) ;x \* cos(a) st(0)=st(0)\*st(1)

fild word ptr [si] ;вторая координата (y) st(0)=y st(1)=x\*cos st(2)=cos st(3)=sin st(4)=angel

fmul st, st(3) ;y \* sin(a) st(0)=st(0)\*st(3)

fsubp ;x \* cos(a) - y \* sin(a) st(0)=st(1)-st(0) st(1)=cos st(2)=sin st(3)=angel

fild word ptr [di] ;x st(0)=x st(1)=x \* cos(a) - y \* sin(a) st(2)=cos st(3)=sin st(4)=angel

fmul st, st(3) ;x \* sin(a) st(0)=st(0)\*st(3)

fild word ptr [si] ;y st(0)=y

fmul st, st(3) ;y \* cos(a) st(0)=st(0)\*st(3)

faddp ;x \* sin(a) + y \* cos(a) st(0)=st(0)+st(1)

fistp word ptr [si] ;y = x \* sin(a) + y \* cos(a) word=st(0)

fstp st(1) ;уберем из стека cos

fstp st(1) ; и sin

fistp word ptr [di] ;x = x \* cos(a) - y \* sin(a)

ret ;в итоге получили новые координаты x,y и угол(в стеке)

Rotation endp

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ВРАЩАЕМ ТОЧКУ ВОКРУГ ВЫБРАННЫХ ОСЕЙ

RotatePoint PROC ;si указывает на структуру POINT3D

;читаем 3D координаты во временные переменные

mov ax, [POINT3D ptr si].xx

add ax, DeltaX

mov X, ax

mov ax, [POINT3D ptr si].yy

add ax, DeltaY

mov Y, ax

mov ax, [POINT3D ptr si].zz

add ax, DeltaZ

mov Z, ax

call Rotation, offset Y, offset Z, XAngle ;вращаем вокруг оси x

call Rotation, offset X, offset Z, YAngle ;вращаем вокруг оси y

call Rotation, offset X, offset Y, ZAngle ;вращаем вокруг оси z

ret

RotatePoint endp

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПРЕОБРАЗУЕМ 3D КООРДИНАТЫ ВЕРШИН В 2D КООРДИНАТЫ НА ЭКРАНЕ

;используем переменные X, Y, Z

;результат пишем по адресу [di], в элемент POINT2D

Conv3Dto2D PROC

mov ax, Xoff ; Xoff\*X / Z+Zoff = экранное X

mov bx, X

imul bx

mov bx, Z

add bx, Zoff ; расстояние от глаз

idiv bx

add ax, Nx ; центр экрана

mov [POINT2D ptr di].xx, ax ; сохраним экранное X

mov ax, Yoff ; Yoff\*Y / Z+Zoff = экранное Y

mov bx, Y

imul bx

mov bx, Z

add bx, Zoff ; расстояние от глаз

idiv bx

add ax, Ny ; центр экрана

mov [POINT2D ptr di].yy, ax ; сохраним экранное Y

ret

Conv3Dto2D ENDP

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ОСНОВНАЯ ПОДПРОГРАММА РАСЧЕТА И РИСОВАНИЯ

MainProgram PROC

;Заливка экрана

cmp FirstCadr, 1

jne draw

mov cx, CountL

xor di, di

FillScr:

push cx ;сохраняем значение счетчик

PRLINE LineMass[di][0], LineMass[di][2], 0

add di, 6

pop cx ;восстанавливаем значение счетчика

loop FillScr

draw:

;----------------------------------- ПЕРЕСЧЕТ фигуры

lea di, Figure2D ; адрес 2D координат на экране

lea si, Figure ; адрес 3D координат вершин

mov cx, CountV ; число вершин

ConvLoop:

call RotatePoint ; вращаем вершины

call Conv3Dto2D ; 3D в 2D

add si, size POINT3D ; на следующую 3D вершину

add di, size POINT2D ; на следующую 2D вершину

loop ConvLoop

;-----------------------------------ПРОРИСОВКА РЕБЕР

mov FirstCadr, 1h

mov cx, CountL

xor di, di

DrLine:

push cx ;сохраняем значение счетчик

PRLINE LineMass[di][0], LineMass[di][2], LineMass[di][4]

add di, 6

pop cx ;восстанавливаем значение счетчика

loop DrLine

ret

MainProgram ENDP

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_РИСУЕМ ЛИНИЮ (X1,Y1)-(X2,Y2) ЦВЕТОМ COLOR

Line proc uses di bx, x1:word, y1:word, x2:word, y2:word, color:byte

local i:word, \ ;для работы со сопроцессором

delta\_x:word, \ ;длина проекции на ось абсцисс

delta\_y:word, \ ;длина проекции на ось ординат

incx:word, \ ;приращение по X

incy:word ;приращение по Y

;определим длину проекции на ось абсцисс и шаг по оси X

mov ax, x2

sub ax, x1 ;ax=x2-x1;

;определим шаг по X (+1 если вперед, -1 если назад, 0 если не меняется)

mov incx, 0 ;пусть incx=0

test ax, ax ;ax=delta\_x

jz set\_delta\_x ;не меняется

jg set\_x\_1 ;jg>0 вперед?

dec incx ;назад, значит incx=-1

neg ax ;найдем ax=abs(delta\_x)

jmp set\_delta\_x ;на сохранение

set\_x\_1:

inc incx ;вперед, значит incx=1;

set\_delta\_x:

mov delta\_x, ax ;delta\_x = abs(x2-x1)

;определим длину проекции на ось ординат и шаг по оси Y

mov ax, y2

sub ax, y1 ;ax=y2-y1;

;определим шаг по Y (+1 если вперед, -1 если назад, 0 если не меняется)

mov incy, 0 ;пусть incy=0

test ax, ax ;ax=delta\_y

jz set\_delta\_y ;не меняется

jg set\_y\_1 ;вперед?

dec incy ;назад, значит incy=-1

neg ax ;найдем ax==abs(delta\_y)

jmp set\_delta\_y ;на сохранение

set\_y\_1:

inc incy ;вперед, значит incy=1;

set\_delta\_y:

mov delta\_y, ax ;delta\_y=abs(y2-y1)

;определим большее из проекций как основное направление

cmp ax, delta\_x ;ax=delta\_y

jge from\_y ;если проекция Y(ax)>=X(delta\_x) то y будет основным

cmp delta\_x, 0 ;проверим, чтобы не было delta\_x=0 (для точки),

jz Line\_ret ; иначе будет деление на 0

;delta\_x>delta\_y && delta\_x!=0

;основное направление - по оси X

fild delta\_y ;запись в стек сопроцессора delta\_y

fidiv delta\_x ;st=k=(float)(delta\_y/delta\_x) делим стек на delta\_x k-угол наклона линии

;for(int i=0;i<delta\_x;i++)

xor cx, cx ;cx=i

jmp cmp\_i\_x ;на проверку i<delta\_x

x\_loop: ;тело цикла

mov i, cx ;запишем переменную цикла в память (для сопроцессора)

fld st ;st(0)=st(1)=k

fimul i ;st(0)=k\*i

fimul incy ;st(0)=incy\*k\*i

call floor ;округлим до целого в большую сторону

fistp i ;сохраним в переменной, то есть i=st(0)... st(1)->st(0)

mov ax, i ;x относительно первой точки строки на экране

add ax, y1 ;добавим до ординаты начальной точки

mov dx, 320 ;получим индекс начала строки экрана в сегменте экрана

imul dx ;для этого умножим на длину в байтах одной строки

mov bx, ax ;сохраним bx=y=(y1+floor(incy\*k\*i))\*320

;посчитаем X

mov ax, incx ;X меняется ровно на шаг приращения,

imul cx ; умноженному на индекс точки

add ax, x1 ;добавим абциссу начальной точки ax=x=x1+incx\*i

add ax, bx ;сложим с индексом начала строки

mov di, ax ;будем адресовать через di

mov al, color ;цвет точки

mov es:[di], al ;рисуем

inc cx ;на следующую точку

cmp\_i\_x:

cmp cx, delta\_x ;дошли до конца?

jl x\_loop ;если i<delta\_x то переходим на x\_loop

jmp Line\_ret ;на выход

from\_y: ;вдоль оси Y

fild delta\_x

fidiv delta\_y ;st=k=(float)(delta\_x/delta\_y)

;for (int i=0;i<delta\_y;i++)

xor cx, cx ;cx=i

jmp cmp\_i\_y ;на проверку i<delta\_y

y\_loop: ;тело цикла

mov ax, incy ;Y меняется ровно на шаг приращения,

imul cx ; умноженному на индекс точки

add ax, y1 ;добавим абциссу начальной точки ax=y=y1+incy\*i

mov dx, 320 ;получим индекс начала строки экрана в сегменте экрана

imul dx ; для этого умножим на длину в байтах одной стоки

mov bx, ax ;сохраним bx=y=(y1+incy\*i)\*320

;посчитаем X

mov i, cx ;запишем переменную цикла в память (для сопроцессора)

fld st ;st=st(1)=k

fimul i ;st=k\*i

fimul incx ;st=incx\*k\*i

call floor ;округлим до целого в большую сторону

fistp i ;сохраним в переменной

mov ax,i ;относительный номер строки на экране

add ax, x1 ;ax=x=x1+floor(incx\*k\*i)

add ax, bx ;сложим с индексом начала строки

mov di, ax ;будем адресовать через di

mov al, color ;цвет точки

mov es:[di], al ;рисуем!

inc cx ;на следующую точку

cmp\_i\_y:

cmp cx, delta\_y ;дошли до конца?

jl y\_loop

Line\_ret:

fistp i ;удалим из сопроцессора k

ret

Line endp

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ОКРУГЛЕНИЕ ДО ЦЕЛОГО В БОЛЬШУЮ СТОРОНУ

floor proc

local CtrlOld:word, CtrlNew:word

fstcw CtrlOld ;сохраним управляющее слово

fclex ;сбросим исключения

mov CtrlNew,0763h ;установим необходимое значение управляющего слова (округление в большую сторону)

fldcw CtrlNew ;загружаем управляющее слово

frndint ;округляем st до целого

fclex ;сбросим исключения

fldcw CtrlOld ;восстановим старое управляющее слово

ret

floor endp

END

Координаты фигуры:

Figure POINT3D < 0, 0, 0>

POINT3D < 70, 120, 0>

POINT3D < 80, 90, 0>

POINT3D <110, 90, 0>

POINT3D <80, 70, 0>

POINT3D <100, 40, 0>

POINT3D <70, 60, 0>

POINT3D <40, 40, 0>

POINT3D <60, 70, 0>

POINT3D <30, 90, 0>

POINT3D <60, 90, 0>

POINT3D <70, 80, 10>

POINT3D <70, 80, -10>

CountV EQU 13 ;число вершин

Figure2D POINT2D CountV dup (<>) ;экранные координаты вершин

LineMass dw 1, 2, 12

dw 2, 3, 12

dw 3, 4, 12

dw 4, 5, 12

dw 5, 6, 12

dw 6, 7, 12

dw 7, 8, 12

dw 8, 9, 12

dw 9, 10, 12

dw 10, 1, 12

dw 1, 11, 10

dw 2, 11, 10

dw 3, 11, 10

dw 4, 11, 10

dw 5, 11, 10

dw 6, 11, 10

dw 7, 11, 10

dw 8, 11, 10

dw 9, 11, 10

dw 10, 11, 10

dw 1, 12, 10

dw 2, 12, 10

dw 3, 12, 10

dw 4, 12, 10

dw 5, 12, 10

dw 6, 12, 10

dw 7, 12, 10

dw 8, 12, 10

dw 9, 12, 10

dw 10, 12, 10

CountL EQU 30

## 2.2.2 Описание программы

При запуске программа получает из файла с расширением .obj координаты точек фигуры. Далее при помощи клавиш клавиатуры W,A,S,D осуществляется сдвиг фигуры по экрану.

# **Заключение**

В ходе выполнения данной курсовой работы было проведено изучение принципов построения графических изображений на языке Ассемблера, а так же операции сдвиг.

# **Список использованных источников**

1. Мочалов А.Ю. Системное программное обеспечение: Уч. – СПб.: Питер, 2010. – 397с.
2. Рощин А.В. Системное программное обеспечение. Особенности программирования 32-разрядных процессоров: Уч. пос. – М.: МГУПИ, 2008. – 144с.
3. Юров В.И. Assebmler: Доп. МО РФ в кач. уч. Для вузов: Уч. Для вузов. – СПб.: Питер, 2011. – 636.
4. Интернет-форум Cyberforum.ru http://www.cyberforum.ru/asm-beginners/thread1097645.html