Федеральное агентство по образованию

ГОУ ВПО

Уфимский государственный авиационный технический университет

# Курсовая работа

# по дисциплине «Микропрограммирование»

«Графика на языке Assembler»

Выполнил:

студент гр. ПО-228

Елизарьев Д.И.

Уфа

2008

**1. Постановка задачи**

Необходимо разработать программу, выводящую на экран трехмерный куб, и позволяющую поворачивать его с помощью клавиш.

Программа реализована на языке “Assembler”. Для вывода графики на экран используется прямое обращение к видеобуферу. Для достижения плавности прорисовки изображения применяется синхронизация с вертикальной развёрткой монитора.

Повороты вокруг осей производятся по следующим формулам:

* Вокруг оси X:   
  
* Вокруг оси Y:  
  
* Вокруг оси Z:  
  

Для рисования линии используется алгоритм Брезенхэма.

Значения синуса и косинуса вычисляются при помощи таблицы синусов для углов от 0 до 90 градусов.

**2. Текст программы.**

DATAS SEGMENT

X DW 0 ;Промежуточная координата X

Y DW 0 ;Промежуточная координата Y

Z DW 0 ;Промежуточная координата Z

ANX DW 0 ;Текущий угол поворота вокруг оси X

ANY DW 0 ;Текущий угол поворота вокруг оси Y

ANZ DW 0 ;Текущий угол поворота вокруг оси Z

DeltaX DW 2 ;Приращение угла поворота вокруг оси X

DeltaY DW 2 ;Приращение угла поворота вокруг оси Y

DeltaZ DW 2 ;Приращение угла поворота вокруг оси Z

X2D DW 0 ;Проекция трехмерной точки на плоскость

Y2D DW 0 ;

X1 DW 0 ;Координаты

Y1 DW 0 ;начала и

X2 DW 0 ;конца

Y2 DW 0 ;линии

DelX DW 0 ;Промежуточные

DelY DW 0 ;переменные,

LenX DW 0 ;используемые

LenY DW 0 ;в процедуре

Leng DW 0 ;рисования

D DW 0 ;линии

COLOR DB 10 ;Цвет фигуры

FULLCIRCLE DW 360 ;Константа = 360 градусов

POINTS DW 8 ;Количество вершин

WID DW 320 ;Ширина экрана

;Таблица синусов углов от 0 до 90 градусов.

;Каждое значение синуса умножено на 512

SINES DW 0, 9, 18, 27, 36, 45

DW 54, 62, 71, 80, 89

DW 98, 106, 115, 124, 133

DW 141, 150, 158, 167, 175

DW 183, 192, 200, 208, 216

DW 224, 232, 240, 248, 256

DW 264, 271, 279, 286, 294

DW 301, 308, 315, 322, 329

DW 336, 343, 349, 356, 362

DW 368, 374, 380, 386, 392

DW 398, 403, 409, 414, 419

DW 424, 429, 434, 439, 443

DW 448, 452, 456, 460, 464

DW 468, 471, 475, 478, 481

DW 484, 487, 490, 492, 495

DW 497, 499, 501, 503, 504

DW 506, 507, 508, 509, 510

DW 511, 511, 512, 512, 512

;Координаты вершин куба

CUBE DW 20, 20, 20

DW 20, 20, -20

DW 20, -20, -20

DW 20, -20, 20

DW -20, -20, 20

DW -20, 20, 20

DW -20, 20, -20

DW -20, -20, -20

DATAS ENDS

CODES SEGMENT

ASSUME DS:DATAS, CS:CODES

FIND\_SIN PROC ;Нахождение синуса угла от 0 до 360 градусов

push ax

push cx

sub cx, cx

cmp ax, 181

jb SIN\_POS

mov cx, 8000h

sub ax, 180

SIN\_POS:

cmp ax, 91

jb GET\_SIN

neg ax

add ax, 180

GET\_SIN:

mov bx, ax

shl bx, 1

mov bx, sines[bx]

cmp cx, 8000h

jne NE1

neg bx

NE1:

pop cx

pop ax

ret

FIND\_SIN ENDP

FIND\_COS PROC ;Нахождение косинуса угла от 0 до 360 градусов

push ax

push cx

sub cx, cx

cmp ax, 91 ;если угол 90 и меньше,

jb COS\_POS ;то знак положительный

cmp ax, 269 ;если угол 270 и больше, то знак "плюс"

jg CP

mov cx, 8000h ;иначе ставим флаг в CX, что знак "минус"

sub ax, 90 ;делаем поправку на 90

cmp ax, 91

jb GET\_COS ;если < 91

neg ax ;иначе угол = 180 - угол

add ax, 180

jmp GET\_COS

CP:

sub ax, 270 ;угол = 270 - угол

jmp GET\_COS

COS\_POS:

neg ax

add ax, 90

cmp ax, 91

jb GET\_COS

neg ax

add ax, 360

GET\_COS: ;достаём значение косинуса из таблицы синусов

mov bx, ax

shl bx, 1

mov bx, sines[bx]

cmp cx, 8000h

jne NE2 ;если знак "минус",

neg bx ;то меняем знак

NE2:

pop cx

pop ax

ret

ENDP FIND\_COS

PUTPIXEL PROC ;Рисование точки в X2D, Y2D, цветом COLOR

push ax

push di

mov ax, 100 ;Высота экрана/2

sub ax, Y2D

push dx

mul WID ;Index = Y \* WIDTH

pop dx

add ax, X2D ;Index + X

add ax, 160

mov di, ax

mov al, COLOR

mov byte ptr ES:[di], al ;рисуем точку

pop di

pop ax

ret

ENDP PUTPIXEL

PROJECT PROC ;Проецирование трёхмерной точки на плоскость

push ax

mov ax, X

mov X2D, ax

mov ax, Y

mov Y2D, ax

pop ax

ret

ENDP PROJECT

ROTX PROC ;Поворот точки вокруг оси X

push cx

push ax

push bx

push dx

mov ax, ANX

CALL FIND\_COS ;

mov ax, bx ;

imul Y ;

mov cx, ax ;

mov ax, ANX ;

CALL FIND\_SIN ;YNEW = Y\*COS(ANX) - Z\*SIN(ANX)

mov ax, bx ;

imul Z ;

neg ax ;

add ax, cx ;

sar ax, 9

mov cx, Y

mov Y, ax

mov ax, ANX ;

CALL FIND\_SIN ;

mov ax, bx ;

imul cx

mov cx, ax ;

mov ax, ANX ;ZNEW = Y\*SIN(ANX) + Z\*COS(ANX)

CALL FIND\_COS ;

mov ax, bx ;

imul Z ;

add ax, cx ;

sar ax, 9

mov Z, ax

pop dx

pop bx

pop ax

pop cx

ret

ENDP ROTX

ROTY PROC ;Поворот точки вокруг оси Y

push cx

push ax

push bx

push dx

mov ax, ANY

CALL FIND\_COS ;

mov ax, bx ;

imul X ;

mov cx, ax ;

mov ax, ANY ;

CALL FIND\_SIN ;XNEW = X\*COS(ANY) - Z\*SIN(ANY)

mov ax, bx ;

imul Z ;

neg ax ;

add ax, cx ;

sar ax, 9

mov cx, X

mov X, ax

mov ax, ANY ;

CALL FIND\_SIN ;

mov ax, bx ;

imul cx ;

mov cx, ax ;

mov ax, ANY ;ZNEW = X\*SIN(ANY) + Z\*COS(ANY)

CALL FIND\_COS ;

mov ax, bx ;

imul Z ;

add ax, cx ;

sar ax, 9

mov Z, ax

pop dx

pop bx

pop ax

pop cx

ret

ENDP ROTY

ROTZ PROC ;Поворот точки вокруг оси Z

push cx

push ax

push bx

push dx

mov ax, ANZ

CALL FIND\_COS ;

mov ax, bx ;

imul X ;

mov cx, ax ;

mov ax, ANZ ;

CALL FIND\_SIN ;XNEW = X \* COS(ANZ) - Y \* SIN(ANZ)

mov ax, bx ;

imul Y ;

neg ax ;

add ax, cx ;

sar ax, 9

mov cx, X

mov X, ax

mov ax, ANZ ;

CALL FIND\_SIN ;

mov ax, bx ;

imul cx ;

mov cx, ax ;

mov ax, ANZ ;YNEW = X \* SIN(ANZ) + Y \* COS(ANZ)

CALL FIND\_COS ;

mov ax, bx ;

imul Y ;

add ax, cx ;

sar ax, 9

mov Y, ax

pop dx

pop bx

pop ax

pop cx

ret

ENDP ROTZ

WAITVRT PROC ;Ждёт вертикальную развёртку монитора.

mov dx,3dah ;3DAh - Номер порта экрана

Vrt:

in al,dx

test al,8

jnz Vrt ;Ждать пока развёртка начнётся

NoVrt:

in al,dx

test al,8

jz NoVrt ;Ждать, пока развёртка закончится

ret

ENDP WAITVRT

;Процедура рисования куба.

;Здесь последовательно вычисляются координаты двух соседних ;вершин, и проводится линия между ними.

;Всего 16 линий.

DRAWCUBE PROC

push cx

push ax

push bx

push dx

mov cx, POINTS

mov si, 0

DRC:

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X1, ax

mov Y1, bx

add si, 6

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X2, ax

mov Y2, bx

add si, 6

CALL LINE

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X1, ax

mov Y1, bx

add si, 6

CALL LINE

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X2, ax

mov Y2, bx

add si, 6

CALL LINE

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X1, ax

mov Y1, bx

add si, 6

CALL LINE

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X2, ax

mov Y2, bx

add si, 6

CALL LINE

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X1, ax

mov Y1, bx

add si, 6

CALL LINE

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X2, ax

mov Y2, bx

add si, 6

CALL LINE

mov si, 12

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X1, ax

mov Y1, bx

CALL LINE

mov si, 24

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X1, ax

mov Y1, bx

mov si, 0

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X1, ax

mov Y1, bx

mov si, 30

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X2, ax

mov Y2, bx

CALL LINE

mov si, 18

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X2, ax

mov Y2, bx

CALL LINE

mov si, 6

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X1, ax

mov Y1, bx

mov si, 36

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X2, ax

mov Y2, bx

CALL LINE

mov si, 42

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X1, ax

mov Y1, bx

mov si, 24

mov ax, CUBE[si]

mov bx, CUBE[si+2]

mov dx, CUBE[si+4]

mov X, ax

mov Y, bx

mov Z, dx

CALL ROTX

CALL ROTY

CALL ROTZ

CALL PROJECT

mov ax, X2D

mov bx, Y2D

mov X2, ax

mov Y2, bx

CALL LINE

pop dx

pop bx

pop ax

pop cx

ret

ENDP DRAWCUBE

;Алгоритм Брезенхэма для линии.

;Суть алгоритма заключается в том, что мы на каждом шаге ;увеличиваем координату X на единицу, и прибавляем к так ;называемой «Ошибке» значение DelY, которое равно “Y2 – Y1. Если ;ошибка превышает LenX, то увеличиваем координату Y (X) на ;единицу. Данный алгоритм пригоден только в том случае, если  
;X2 > X1 и расстояние по горизонтали (X2 – X1) больше расстояния ;по вертикали (Y2 – Y1). Иначе же, если X2 < X1, то к координате ;X не прибавляем единицу, а наоборот, отнимаем. Если ;вертикальное расстояние больше горизонтального, то переменные ;X и Y меняются ролями: на каждом шаге увеличиваем Y на единицу, ;а X увеличивается в зависимости от «Ошибки».

LINE PROC

push cx

push ax

push bx

push dx

mov DelX, 1 ;Приращение X = 1

mov DelY, 1 ;Приращение Y = 1

mov ax, x2

cmp ax, x1

jge X2GX1 ;если X2 < X1

neg DelX ;DelX = -1

X2GX1:

mov ax, Y2

cmp ax, Y1

jge Y2GY1 ;Если Y2 < Y1

neg DelY ;DelY = -1

Y2GY1:

mov ax, X2

sub ax, x1

jns LENXG0

neg ax

LENXG0:

mov LenX, ax

mov ax, Y2

sub ax, Y1

jns LENYG0

neg ax

LENYG0:

mov LenY, ax

mov bx, LenX

cmp ax, bx

jg LenYGLenX

mov Leng, bx

jmp C1

LenYGLenX:

mov Leng, ax

C1:

cmp ax, bx

jg LYGLX ;Если ABS(X2-X1) > ABS(Y2-Y1)

mov ax, X1

mov bx, Y1

mov dx, LenX

neg dx

inc Leng

mov cx, Leng

shl LenX, 1

shl LenY, 1

CYCLE1:

mov X2D, ax ;X = X1

mov Y2D, bx ;Y = Y1

CALL PUTPIXEL ;Рисуем точку

add ax, DelX ;X = X + DelX

add dx, LenY ;D = D + 2\*(Y2-Y1)

cmp dx, 0 ;Если D > 0

jle DL01 ;

sub dx, LenX ;D = D - 2\*(X2-X1)

add bx, DelY ;Y = Y + DelY

DL01:

loop CYCLE1

jmp EXITLINE

LYGLX: ;Если ABS(X2-X1) <= ABS(Y2-Y1)

mov ax, X1

mov bx, Y1

mov dx, LenY

neg dx

inc Leng

mov cx, Leng

shl LenX, 1

shl LenY, 1

CYCLE2:

mov X2D, ax ;X = X1

mov Y2D, bx ;Y = Y1

CALL PUTPIXEL ;Рисуем точку

add bx, DelY ;Y = Y + DelY

add dx, LenX ;D = D + 2\*(X2-X1)

cmp dx, 0 ;Если D > 0

jle DL02

sub dx, LenY ;D = D - 2\*(Y2-Y1)

add ax, DelX ;X = X + DelX

DL02:

loop CYCLE2

EXITLINE:

pop dx

pop bx

pop ax

pop cx

ret

ENDP LINE

MAIN PROC

mov ax, datas

mov ds, ax ;Инициализация сегментов данных

mov ax, 0A000h ;A000h - сегмент видеобуфера

mov es, ax

mov ah, 00h ;Установка видеорежима

mov al, 13h ;Mode = 13h (320x200x256)

int 10h

MainLoop:

mov COLOR, 15 ;Рисование точки

CALL DRAWCUBE ;Рисуем куб

CALL WAITVRT ;Ждём развёртку экрана

mov COLOR, 0 ;

CALL DRAWCUBE ;Стираем куб

in al,60h ;Читаем скан-код клавиатуры

cmp al, 4bh ;Если клавиша "Left"

jne DONTROTL

mov ax, DeltaY

sub ANY, ax ;уменьшаем ANY на DeltaY

jns DONTROTD

add ANY, 360

jmp DONTROTD

DONTROTL:

cmp al, 4dh ;Если клавиша "Right"

jne DONTROTR

mov ax, DeltaY

add ANY, ax ;увеличиваем ANY на DeltaY

cmp ANY, 360

jb DONTROTD

sub ANY, 360

jmp DONTROTD

DONTROTR:

cmp al, 48h ;Если клавиша "Up"

jne DONTROTU

mov ax, DeltaX

add ANX, ax ;увеличиваем ANX на DeltaX

cmp ANX, 360

jb DONTROTD

sub ANX, 360

jmp DONTROTD

DONTROTU:

cmp al, 50h ;Если клавиша "Down"

jne DONTROTD ;

mov ax, DeltaX ;

sub ANX, ax ;уменьшаем ANX на DeltaX

jns DONTROTD ;

add ANX, 360 ;

DONTROTD:

cmp al, 01h ;если не Escape

jne MainLoop ;делаем цикл

EXIT:

mov ah, 00h ;Установка текстового видеорежима

mov al, 02h ;Mode - 02h

int 10h

mov ax, 4C00h ;Terminate

int 21h

ENDP MAIN

CODES ENDS

END MAIN