1. **Приложение 1**

**Отрисовка горизонтальной линии, вариант 2**

.model small

.data

…

byt dw ?

tl DB 11111111b ; таблица

DB 01111111b ; для

DB 00111111b ; левого

DB 00011111b ; конца

DB 00001111b ; отрезка

DB 00000111b ;

DB 00000011b ;

DB 00000001b ;

.code

…

;========== find first byte

mov ax, y ; select y

mov dx, 80

mul dx ; ax = y \* 80

mov bx, x

shr bx, 1

shr bx, 1

shr bx, 1

add ax, bx ; select byte

mov byt, ax

;========== bit mask first byte / b1

mov ax, bx

shl ax, 1

shl ax, 1

shl ax, 1

mov cx, x

sub cx, ax

mov bit, cl

;========== set bit mask

mov dx, 3CEh ;указываем на адресный регистр

mov al, 8 ;номер регистра

out dx, al ;посылаем его

inc dx ;указываем на регистр данных

mov si, cx

mov al, tl[si]

out dx, al ;посылаем данные

;---чистим текущее содержимое задвижки

mov bx, byt

mov al, es:[bx] ;читаем содержимое в задвижку

mov al, 0 ;готовимся к очистке

mov es:[bx], al ;чистим задвижку

;---рисуем точку

mov al, 0FFh

mov es:[bx], al ;выводим точку

;========== draw middle bytes

;========== set bit mask

mov dx, 3CEh ;указываем на адресный регистр

mov al, 8 ;номер регистра

out dx, al ;посылаем его

inc dx ;указываем на регистр данных

mov al, 0FFh

out dx, al ;посылаем данные

;========== find length w/out first byte

mov ax, len

sub ax, cx

shr ax, 1

shr ax, 1

shr ax, 1

mov cx, ax

drawthisshit:

;---рисуем точку

inc bx

mov al, 0FFh

mov es:[bx], al ;выводим точку

loop drawthisshit

;========== find last byte

mov ax, y ; select y

mov dx, 80

mul dx ; ax = y \* 80

mov bx, x

add bx, len

shr bx, 1

shr bx, 1

shr bx, 1

add ax, bx ; select byte

mov byt, ax

;========== bit mask last byte / b2

mov ax, bx

shl ax, 1

shl ax, 1

shl ax, 1

mov cx, x

add cx, len

sub cx, ax

mov bit, cl

;========== set bit mask

mov dx, 3CEh ;указываем на адресный регистр

mov al, 8 ;номер регистра

out dx, al ;посылаем его

inc dx ;указываем на регистр данных

mov si, cx

mov al, tl[si]

not al

out dx, al ;посылаем данные

;---чистим текущее содержимое задвижки

mov bx, byt

mov al, es:[bx] ;читаем содержимое в задвижку

mov al, 0 ;готовимся к очистке

mov es:[bx], al ;чистим задвижку

;---рисуем точку

mov al, 0FFh

mov es:[bx], al ;выводим точку

…

**Отрисовка буквы G**

.286

.model small, stdcall

.data

color db 12

x dw 0

y dw 180

px dw ?

py dw ?

pc db ?

bit db 8

.code

drawpixel proc near uses ax bx dx cx

mov ax, py ; select y

mov dx, 80

mul dx ; ax = y \* 80

mov bx, px

mov cl, bl

shr bx, 3

add bx, ax ; select byte

and cl, 07h ; bit mask

xor cl, 07h

mov ah, 01h

shl ah, cl

mov bit, ah

;---маскируем биты

mov dx, 3CEh ;указываем на адресный регистр

mov al, 8 ;номер регистра

out dx, al ;посылаем его

inc dx ;указываем на регистр данных

mov al, bit ;маска

out dx, al ;посылаем данные

mov al, es:[bx] ;читаем содержимое в задвижку

;---установка регистра маски карты для цвета

mov dx, 3C4h ;указываем на адресный регистр

mov al, 2 ;индекс регистра маски карты

out dx, al ;установка адреса

inc dx ;указываем на регистр данных

mov al, color ;код цвета

out dx, al ;посылаем код цвета

;---рисуем точку

mov al, 0FFh

mov es:[bx], al ;выводим точку

ret

drawpixel endp

; compare px;py pixel color w/ needed color

cmpclr proc near uses ax cx dx bx

; found byte offset

mov ax, py ; select y

mov dx, 80

mul dx ; ax = y \* 80

mov bx, px

mov cl, bl

shr bx, 3

add bx, ax ; select byte

; find pixel bit mask

and cl, 07h ; bit mask

xor cl, 07h

mov ah, 01h

shl ah, cl

mov bit, ah

mov pc, 0 ; by def color not equal

; mem start

mov ax, 0A000h

mov es, ax

; init

mov dx, 3ceh

mov al, 5

out dx, al

inc dx

mov al, 8

out dx, al

; clr compare mode

dec dx

mov al, 2

out dx, al

inc dx

mov al, color

out dx, al

; choose out pixel by mask

dec dx

mov al, 8

out dx, al

inc dx

mov al, bit

; read byte

mov ah, es:[bx] ;читаем содержимое в задвижку

and al, ah ; cmp one bit/pixel

jz ot ; it's not equal

mov pc, 1 ; yeah, it matches

ot:

ret

cmpclr endp

; Bresenham's line

line proc START\_X:word, START\_Y:word, END\_X:word, END\_Y:word

local DIAGONAL\_Y\_INCREMENT:word

local DIAGONAL\_X\_INCREMENT:word

local SHORT\_DISTANCE:word

local STRAIGHT\_X\_INCREMENT:word

local STRAIGHT\_Y\_INCREMENT:word

local STRAIGHT\_COUNT:word

local DIAGONAL\_COUNT:word

mov ax,START\_X ;если координаты начала и конца совпадают

cmp ax,END\_X

jnz ris

mov ax,START\_Y ;если нет - рисовать линию

cmp ax,END\_Y

jz LINE\_FINISHED

ris:

;---установка начальных инкрементов для каждой позиции точки

MOV CX,1;инкремент для оси x

MOV DX,1;инкремент для оси y

;---вычисление вертикальной дистанции

MOV DI,END\_Y;вычитаем координату начальной

SUB DI,START\_Y ;точки из координаты конечной

JGE KEEP\_Y ;вперед если наклон < 0

NEG DX ;иначе инкремент равен -1

NEG DI ;а дистанция должна быть > 0

KEEP\_Y:MOV DIAGONAL\_Y\_INCREMENT,DX

;---вычисление горизонтальной дистанции

MOV SI,END\_X;вычитаем координату начальной

SUB SI,START\_X ;точки из координаты конечной

JGE KEEP\_X ;вперед если наклон < 0

NEG CX ;иначе инкремент равен -1

NEG SI ;а дистанция должна быть > 0

KEEP\_X:MOV DIAGONAL\_X\_INCREMENT,CX

;---определяем горизонтальны или вертикальны прямые сегменты

CMP SI,DI ;горизонтальные длиннее?

JGE HORZ\_SEG;если да, то вперед

MOV CX,0;иначе для прямых x не меняется

XCHG SI,DI ;помещаем большее в CX

JMP SAVE\_VALUES;сохраняем значения

HORZ\_SEG: MOV DX,0;теперь для прямых не меняется y

SAVE\_VALUES:MOV SHORT\_DISTANCE,DI ;меньшее расстояние

MOV STRAIGHT\_X\_INCREMENT,CX ;один из них 0,

MOV STRAIGHT\_Y\_INCREMENT,DX ;а другой - 1.

;---вычисляем выравнивающий фактор

MOV AX,SHORT\_DISTANCE ;меньшее расстояние в AX

SHL AX,1;удваиваем его

MOV STRAIGHT\_COUNT,AX ;запоминаем его

SUB AX,SI ;2\*меньшее - большее

MOV BX,AX ;запоминаем как счетчик цикла

SUB AX,SI ;2\*меньшее - 2\*большее

MOV DIAGONAL\_COUNT,AX ;запоминаем

;---подготовка к выводу линии

MOV CX,START\_X ;начальная координата x

MOV DX,START\_Y ;начальная координата y

INC SI ;прибавляем 1 для конца

MOV AL, color;берем код цвета

;---теперь выводим линию

MAINLOOP: DEC SI ;счетчик для большего расстояния

JZ LINE\_FINISHED ;выход после последней точки

push bx

mov bh,0

mov ah,0ch

int 10h

pop bx

SKIP: CMP BX,0;если BX < 0, то прямой сегмент

JGE DIAGONAL\_LINE ;иначе диагональный сегмент

;---выводим прямые сегменты

ADD CX,STRAIGHT\_X\_INCREMENT ;определяем инкре-

ADD DX,STRAIGHT\_Y\_INCREMENT ;менты по осям

ADD BX,STRAIGHT\_COUNT ;фактор выравнивания

JMP MAINLOOP ;на следующую точку

;---выводим диагональные сегменты

DIAGONAL\_LINE: ADD CX,DIAGONAL\_X\_INCREMENT ;определяем инкре-

ADD DX,DIAGONAL\_Y\_INCREMENT ;менты по осям

ADD BX,DIAGONAL\_COUNT ;фактор выравнивания

JMP MAINLOOP ;на следующую точку

LINE\_FINISHED:

ret

line endp

; Starodubcev's 6/8 circle

circle proc

mov dx, 0

oe:

inc x

add dx, x

cmp dx, y

jl pixel

sub dx, y

dec y

pixel:

mov ax, x

add ax, 320

mov px, ax

mov ax, y

add ax, 240

mov py, ax

call drawpixel ; IV

mov ax, x

neg ax

add ax, 320

mov px, ax

mov ax, y

add ax, 240

mov py, ax

call drawpixel ; V

mov ax, x

add ax, 320

mov px, ax

mov ax, y

neg ax

add ax, 240

mov py, ax

call drawpixel ; I

mov ax, x

neg ax

add ax, 320

mov px, ax

mov ax, y

neg ax

add ax, 240

mov py, ax

call drawpixel ; VIII

mov ax, x

add ax, 240

mov py, ax

mov ax, y

neg ax

add ax, 320

mov px, ax

call drawpixel ; VI

mov ax, x

neg ax

add ax, 240

mov py, ax

mov ax, y

neg ax

add ax, 320

mov px, ax

call drawpixel ; VII

mov ax, x

cmp ax, y

jl oe

ret

circle endp

; Starodubcev's 1/8 circle

circle2 proc

mov dx, 0

mov x, 0

mov y, 40

oe2:

inc x

add dx, x

cmp dx, y

jl pixel2

sub dx, y

dec y

pixel2:

mov ax, x

add ax, 408

mov px, ax

mov ax, y

neg ax

add ax, 340

mov py, ax

call drawpixel ; I

mov ax, x

neg ax

add ax, 340

mov py, ax

mov ax, y

add ax, 408

mov px, ax

call drawpixel ; II

mov ax, x

cmp ax, y

jl oe2

ret

circle2 endp

; fill triangle between 2) 3) 4)

fillthisshit proc ZX:WORD, ZY:WORD

push 1024 ; when it pops then exit

push ZY ; first pixel

push ZX

recursive: ; main loop

pop ax ; pop x

cmp ax, 1024 ; its end of pixels

je ext ; then exit

pop bx ; pop y

mov px, ax

mov py, bx

call drawpixel ; color it

; check 4 pixels around

inc px ; right

call cmpclr

cmp pc, 1 ; coloured, skip it

je N1

push py ; or add it to stack

push px

N1: ; skip right pixel

dec px ; left now

dec px

call cmpclr ; ...

cmp pc, 1

je N2

push py

push px

N2:

inc px ; top

dec py

call cmpclr

cmp pc, 1

je N3

push py

push px

N3:

inc py ; btm

inc py

call cmpclr

cmp pc, 1

je N4

push py

push px

N4:

jmp recursive ; loop while there are pixels to colour

ext:

ret

fillthisshit endp

start:

mov ax, @data ; настройка сегмента данных

mov ds, ax ; через регистр ах

mov ax, 12h ; set mode to VGA 640x430x16

int 10h

mov ax, 0A000h ; set video memory start

mov es, ax

; mov bx, address ; select byte

mov dx, 3CEh ; port

mov ax, 0005h ; initialize

out dx, ax

mov dx, 3CEh ; port

mov ax, 0000h ; mode read-write 0

out dx, ax

; 1) C

; 2) |

; 3) -

; 4) \

; 5) )

; 6) fill triangle

;========== 3)

push 300

push 320

push 300

push 408

call line

;========== 2)

push 368

push 448

push 340

push 448

call line

;========== 4)

push 368

push 448

push 300

push 320

call line

;========== 1)

mov px, 320 ; top-left-bottom pixel fix

mov py, 60

call drawpixel

mov px, 320

mov py, 420

call drawpixel

mov px, 140

mov py, 240

call drawpixel

call circle

;========== 5)

call circle2

;========== 6)

push 320

push 400

call fillthisshit

xor ax, ax ;ожидание нажатия клавиши

int 16h

mov ax, 4C00h ;выход из графики с возвратом

int 21h ;в предыдущий режим

end start