| IDEAL  MODEL small  STACK 0f500h  p186  MAX\_BMP\_WIDTH = 320  MAX\_BMP\_HEIGHT = 200  SMALL\_BMP\_HEIGHT = 40  SMALL\_BMP\_WIDTH = 40  DATASEG  OneBmpLine db MAX\_BMP\_WIDTH dup (0) ; One Color line read buffer  ScreenLineMax db MAX\_BMP\_WIDTH dup (0) ; One Color line read buffer  ;BMP File data  FileHandle dw ?  Header db 54 dup(0)  Palette db 400h dup (0)  note1 dw 023A1h ; 1193180 /-> (hex) do  note2 dw 01FBEh ; 1193180 / -> (hex) re  note3 dw 01C47h ; 1193180 / -> (hex) mi  note4 dw 01AB1h ; 1193180 / -> (hex) fa  note5 dw 017C7h ; 1193180 / -> (hex) sol  note6 dw 0152Fh ; 1193180 / -> (hex) la  note7 dw 012DFh ; 1193180 / -> (hex) si  note8 dw 021A1h ; DO diaz  note9 dw 01DF6h ; RE diaz  note10 dw 01931h ;Fa diaz  note11 dw 01672h ;Sol diaz  note12 dw 013FFh ;La diaz  note13 dw 011D0h ; 1193180 /-> (hex) do  note14 dw 0FDFh ; 1193180 / -> (hex) re  note15 dw 0E24h ; 1193180 / -> (hex) mi  note20 dw 010D1h ; DO diaz  note21 dw 0EFBh ; RE diaz  clock equ es:6Ch  note dw ? ;note save  home db 'home1.bmp',0  play db 'keyboard.bmp',0  keyQ db 'keyQ.bmp', 0  keyW db 'keyW.bmp', 0  keyE db 'keyE.bmp', 0  keyR db 'keyR.bmp', 0  keyT db 'keyT.bmp', 0  keyY db 'keyY.bmp', 0  keyU db 'keyU.bmp', 0  keyI db 'keyI.bmp', 0  keyO db 'keyO.bmp', 0  keyP db 'keyP.bmp', 0  key2 db 'key2.bmp', 0  key3 db 'key3.bmp', 0  key5 db 'key5.bmp', 0  key6 db 'key6.bmp', 0  key7 db 'key7.bmp', 0  key9 db 'key9.bmp', 0  key0 db 'key0.bmp', 0  ErrorFile db 0  BmpLeft dw ?  BmpTop dw ?  BmpColSize dw ?  BmpRowSize dw ?  CODESEG  exit2:  call exitproc  start:  mov ax, @data  mov ds, ax  call SetGraphic  mov [BmpLeft],0  mov [BmpTop],0  mov [BmpColSize], 320  mov [BmpRowSize] ,200  showbmp2:  mov dx, offset home  call OpenShowBmp  ;delay  mov cx, 0fh  mov dx, 4240h  mov ah, 86h  int 15h    mov ah, 86h  int 15h  mov dx,offset play ;THE NORMAL PICTURE OF THE PIANO  call OpenShowBmp  jmp piano  piano1:  mov dx, offset play  call OpenShowBmp  call soundclose  jmp piano  STOPPER: ;STOP POSITIONS FOR SOUND OF THE 4TH OCTAVE  call procdo4  jmp piano1  STOPPER2:  call procre4  jmp piano1  STOPPER3:  call procmi4  jmp piano1  STOPPER8:  call procdodiaz4  jmp piano1  STOPPER9:  call procrediaz4  jmp piano1    piano: ;MAIN PIANO INPUT  mov ah, 7  int 21h    cmp al, 'q'  je dojmp  cmp al, 'w'  je rejmp  cmp al, 'e'  je mijmp  cmp al, 'r'  je fajmp  cmp al, 't'  je soljmp  cmp al, 'y'  je lajmp  cmp al, 'u'  je sijmp2  cmp al, 'i'  je stopper  cmp al, 'o'  je stopper2  cmp al, 'p'  je stopper3    cmp al,'2';diaz normal do  je dod  cmp al, '3' ;diaz normal re  je red  cmp al, '5' ;diaz normal fa  je fad  cmp al, '6' ;diaz normal sol  je sold  cmp al, '7' ;diaz normal la  je lad  cmp al,'9' ;DO DIAZ C4  je stopper8  cmp al,'0' ;RE DIAZ C4  je stopper9    jmp piano  dojmp: ;STOP POSITIONS FOR NORMAL NOTES OCTAVE C3 1-7  call procdo  jmp piano1  rejmp:  call procre  jmp piano1  mijmp:  call procmi  jmp piano1  fajmp:  call procfa  jmp piano1  soljmp:  call procsol  jmp piano1  lajmp:  call procla  jmp piano1  sijmp2:  call procsi  jmp piano1  dod: ;do diaz ; ;DIAZ STOPPERS OCATVE C3  call procdod ;  jmp piano1 ;  red: ;re diaz ;  call procred ;  jmp piano1 ;  fad: ;fa diaz ;  call procfad ;  jmp piano1 ;  sold: ;sol diaz ;  call procsold ;  jmp piano1 ;  lad: ;la diaz ;  call proclad ;  jmp piano1 ;  proc sound ;sound toggle procedure 1  pusha  mov bp, sp  in al, 61h  or al, 00000011b  out 61h, al ; send control word to change frequency  mov al, 0B6h  out 43h, al  mov ax, [note]  out 42h, al ; Sending lower byte  mov al, ah  out 42h, al ; Sending upper byte  call Timer  call soundclose  mov dx,offset play  call OpenShowBmp  popa  ret  endp sound  proc Timer ;TIMER WITH 2 TICKS  pusha  mov ax,40h ;enable Timer  mov es,ax  mov ax, [clock]  FirstTick:  cmp ax, [clock]  mov cx, 6 ;ticks  je FirstTick  DelayLoop:  mov ax, [clock]  Tick:  cmp ax, [clock]  je Tick  loop DelayLoop  popa  ret  endp Timer  proc procmi4 ;mi in 4TH OCTAVE  mov dx, offset keyP  call OpenShowBmp  mov ax, [note15]  mov [note],ax  call sound  ret  endp procmi4  proc procre4 ;re in 4TH OCTAVE  mov dx, offset keyO  call OpenShowBmp  mov ax, [note14]  mov [note],ax  call sound  ret  endp procre4  proc procdo4 ;do in 4TH OCTAVE  mov dx, offset keyI  call OpenShowBmp  mov ax, [note13]  mov [note],ax  call sound  ret  endp procdo4  proc procsol ;sol NORMAL  pusha;  mov dx, offset keyT ;  call OpenShowBmp ;  mov ax, [note5] ;  mov [note],ax ;  call sound ;  popa ;  ret ;  endp procsol ;  proc procfa ;fa NORMAL  pusha ;  mov dx, offset keyR  call OpenShowBmp ;  mov ax, [note4] ;  mov [note],ax ;  call sound ;  popa ;  ret ;  endp procfa ;  proc procmi ;mi NORMAL  pusha ;  mov dx, offset keyE  call OpenShowBmp ;  mov ax, [note3] ;  mov [note],ax ;  call sound ;  popa ;  ret ;  endp procmi ;  proc procre ;re NORMAL  pusha ;  mov dx, offset keyW;  call OpenShowBmp ;  mov ax, [note2] ;  mov [note],ax ;  call sound ;  popa ;  ret ;  endp procre ;  proc procdo ;do NORMAL  pusha ;  mov dx, offset keyQ ;  call OpenShowBmp ;  mov ax, [note1] ;  mov [note],ax ;  call sound ;  popa ;  ret ;  endp procdo ;  proc procla ;la NORMAL  pusha ;  mov dx, offset keyY ;  call OpenShowBmp ;  mov ax, [note6] ;  mov [note],ax ;  call sound ;  popa ;  ret ;  endp procla ;  proc procsi ;si NORMAL  pusha ;  mov dx, offset keyU ;  call OpenShowBmp ;  mov ax, [note7] ;  mov [note],ax ;  call sound ;  popa ;  ret ;  endp procsi ;  proc procdod ;;DO DIAZ NORMAL  mov dx, offset key2 ;  call OpenShowBmp ;  mov ax, [note8] ;  mov [note],ax ;  call sound ;  ret ;  endp procdod ;  proc procred ;;RE DIAZ NORMAL  mov dx, offset key3 ;  call OpenShowBmp ;  mov ax, [note9] ;  mov [note],ax ;  call sound ;  ret ;  endp procred ;  proc procfad ;;FA DIAZ NORMAL  mov dx, offset key5 ;  call OpenShowBmp ;  mov ax, [note10] ;  mov [note],ax ;  call sound ;  ret ;  endp procfad ;  proc procsold ;;SOL DIAZ NORMAL  mov dx, offset key6;  call OpenShowBmp ;  mov ax, [note11] ;  mov [note],ax ;  call sound ;  ret ;  endp procsold ;  proc proclad ;;LA DIAZ NORMAL  mov dx, offset key7 ;  call OpenShowBmp ;  mov ax, [note12] ;  mov [note],ax ;  call sound ;  ret ;  endp proclad ;  proc procdodiaz4 ;;DO DIAZ 4OCT  mov dx, offset key9  call OpenShowBmp ;  mov ax, [note20] ;  mov [note],ax ;  call sound ;  ret ;  endp procdodiaz4 ;  proc procrediaz4 ;;RE DIAZ 4OCT  mov dx, offset key0  call OpenShowBmp ;  mov ax, [note21] ;  mov [note],ax ;  call sound ;  ret ;  endp procrediaz4 ;  proc soundclose ;soundclose  in al, 61h  and al, 11111100b  out 61h, al  ret  endp soundclose  proc OpenShowBmp near  push cx  push bx  call OpenBmpFile  cmp [ErrorFile],1  je @@ExitProc  call ReadBmpHeader  call ReadBmpPalette  call CopyBmpPalette  call ShowBMP  call CloseBmpFile  @@ExitProc:  pop bx  pop cx  ret  endp OpenShowBmp  proc OpenBmpFile near  mov ah, 3Dh  xor al, al  int 21h  jc @@ErrorAtOpen  mov [FileHandle], ax  jmp @@ExitProc  @@ErrorAtOpen:  mov [ErrorFile],1  @@ExitProc:  ret  endp OpenBmpFile  proc CloseBmpFile near  mov ah,3Eh  mov bx, [FileHandle]  int 21h  ret  endp CloseBmpFile  proc ReadBmpHeader near  push cx  push dx  mov ah,3fh  mov bx, [FileHandle]  mov cx,54  mov dx,offset Header  int 21h  pop dx  pop cx  ret  endp ReadBmpHeader  proc ReadBmpPalette near  push cx  push dx  mov ah,3fh  mov cx,400h  mov dx,offset Palette  int 21h  pop dx  pop cx  ret  endp ReadBmpPalette  proc CopyBmpPalette near  push cx  push dx  mov si,offset Palette  mov cx,256  mov dx,3C8h  mov al,0  out dx,al  inc dx  CopyNextColor:  mov al,[si+2]  shr al,2  out dx,al  mov al,[si+1]  shr al,2  out dx,al  mov al,[si]  shr al,2  out dx,al  add si,4  loop CopyNextColor  pop dx  pop cx  ret  endp CopyBmpPalette  proc ShowBMP  push cx  mov ax, 0A000h  mov es, ax  mov cx,[BmpRowSize]  mov ax,[BmpColSize] ; row size must dived by 4 so if it less we must calculate the extra padding bytes  xor dx,dx  mov si,4  div si  mov bp,dx  mov dx,[BmpLeft]  @@NextLine:  push cx  push dx  mov di,cx ; Current Row at the small bmp (each time -1)  add di,[BmpTop] ; add the Y on entire screen  mov cx,di  shl cx,6  shl di,8  add di,cx  add di,dx  mov ah,3fh  mov cx,[BmpColSize]  add cx,bp ; extra bytes to each row must be divided by 4  mov dx,offset ScreenLineMax  int 21h  cld ; Clear direction flag, for movsb  mov cx,[BmpColSize]  mov si,offset ScreenLineMax  rep movsb ; Copy line to the screen  pop dx  pop cx  loop @@NextLine  pop cx  ret  endp ShowBMP  proc SetGraphic  mov ax,13h ; 320 X 200  int 10h  ret  endp SetGraphic  proc exitproc ;exit procedure any time  mov ax,2  int 10h  mov ax, 4c00h  int 21h  endp exitproc  END start |
| --- |