ESCUELA POLITECNICA NACIONAL
FACULTAD DE INGENIERIA ELECTRICA

SISTEMA DE ADQUISICION Y PROCESAMIENTO DIGITAL DE LARINGOGRAMAS

LISTADOS DE LOS PROGRAMAS

Y

MANUAL DEL USUARIO

OLGA L. RIVERA R.

Julio, 1993

ANEXO:

LISTADO DE PROGRAMAS EN LENGUAJE C

ANEXO: LISTADO DE PROGRAMAS EN C.

ARCHIVO: BDD.I

```
/* ------------------------------------
   Variables globales
   ------------------------------------ */
C4CODE   cb1, cb2 ;
D4DATA  *data1, *data2 ;
F4FIELD *codigo2, *fecha2, *obs2, *datos2, *arch2;
F4FIELD *paciente1, *codigo1 ,*obs1;
T4TAG   *name, *class_list ;

/* ------------------------------------
   estructura de la base de paciente.dbf
   ------------------------------------ */
static F4FIELD_INFO paciente[] =
{
    /*
    campo          tipo  ancho   decimales
  */
    { "PACIENTE1",     'C',   30,     0 },
    {  "CODIGO1",      'C',    3,     0 },
    {     "OBS1",      'C',   40,     0 },
    {          0,        0,    0,     0 },
};

/* - definicion de TAGS para archivos indices paciente.mdx - */

static T4TAG_INFO tag_paciente[] =
{
    /*
       TAG       expresion      filtro  unico        descendente
    */
    { "CODIGO", "CODIGO1", "",   r4unique_continue,        0},
    {       0,        0,  0,                   0,        0 },
};

/* ------------------------------------
```

```
   estructura de la base de laringo.dbf
   ------------------------------------ */
static F4FIELD_INFO laringograma[] = {
    /*
       campo          tipo  ancho   decimales
    */
    { "CODIGO1",      'C',    3,     0 },
    {  "FECHA2",      'D',    8,     0 },
    {    "OBS2",      'C',   40,     0 },
    {  "DATOS2",      'M',   10,     0 },
    { "ARCHIVO",      'C',   10,     0 },
    {         0,        0,    0,     0 },
};

/* - definicion de TAGS para archivos indices laringo.mdx - */

static T4TAG_INFO tag_laringo[] =
{
    /*
       TAG       expresion          filtro  unico          descendente
    */
    { "CODIGO", "CODIGO1",              "",   0,                  0},
    {       0, 0,                      0,    0,                  0},
};
```

LISTADO DEL ARCHIVO: ESQUEMA.I

```
char basura;          /* para lectura de pulsos de tecla */

/* ---------------------------------------------
      procedimiento para el color de texto (c1) y color de
      fondo de texto (c2)
   --------------------------------------------- */
void color(int c1, int c2) {
      textcolor(c1);
      textbackground(c2);
}

/* ---------------------------------------------
       procedimiento para dibujar un cuadrado en pantalla.
       --------------------------------------------- */
void cuadro(int x1,int y1,int x2,int y2,char cadena[8])
{
      int i;
      gotoxy(x1,y1); printf("%c",cadena[0]);
      gotoxy(x2,y1); printf("%c",cadena[2]);
      gotoxy(x2,y2); printf("%c",cadena[4]);
      gotoxy(x1,y2); printf("%c",cadena[6]);
      for(i=x1+1;i<x2;i++) {
         gotoxy(i,y1); printf("%c",cadena[1]);
         gotoxy(i,y2); printf("%c",cadena[5]);
      }
      for(i=y1+1;i<y2;i++) {
         gotoxy(x2,i); printf("%c",cadena[3]);
         gotoxy(x1,i); printf("%c",cadena[7]);
      }
}

/* ---------------------------------------------
       procedimiento para definir una ventana en pantalla,
       con determinados colores.
       --------------------------------------------- */
```

```
void ventana(int x1,int y1,int x2,int y2,int color1,int color2) {
      window(x1,y1,x2,y2);
      color(color1,color2);
      clrscr();
      cuadro(1,1,x2-x1+1,y2-y1+1,"┌┐ │┘─└│");
}

/* ----------------------------------------------------
      procedimiento para borrar la pantalla entera.
      ---------------------------------------------------- */
void esquemar(void) {
      color(WHITE,BLACK);
      window(1,1,80,25);
      clrscr();
}

/* ----------------------------------------------------
      pantalla
      ---------------------------------------------------- */
void esquema1(void) {
      char t1[] = " Presione cualquier tecla para continuar ";
      esquemar();
      gotoxy((int)(80-strlen(t1))/2,22); printf("%s",t1);
}

/* ----------------------------------------------------
      pantalla
      ---------------------------------------------------- */
void esquema2(void) {
      int y=0,x1=5,y1=5,x2=75,y2=18;
      char t1[] = " ESCUELA POLITECNICA NACIONAL ";
      char t2[] = " FACULTAD DE INGENIERIA ELECTRICA ";
      char t3[] = " PROGRAMA PARA EL CONTROL DE LARINGOGRAMAS ";
      char t4[] = " por: OLGA RIVERA";
      char t5[] = " Quito, junio de 1993";
      ventana(x1,y1,x2,y2,WHITE,BLUE);
      gotoxy((int)(x2-x1-strlen(t1))/2,y+3); printf("%s",t1);
```

```
    gotoxy((int)(x2-x1-strlen(t2))/2,y+5); printf("%s",t2);
    gotoxy((int)(x2-x1-strlen(t3))/2,y+7); printf("%s",t3);
    gotoxy((int)(x2-x1-strlen(t4))/2,y+10); printf("%s",t4);
    gotoxy((int)(x2-x1-strlen(t5))/2,y+12); printf("%s",t5);
    basura=getch();
}
```

LISTADO DEL ARCHIVO: PR01.C

```
#include <io.h>
#include <fcntl.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <graphics.h>
#include <d4all.h>        /* archivo de CODE BASE */
#include <process.h>
#include "esquema.i"      /* esquemas de pantallas */

/* - encabezados predefinidos - */
#ifdef __TURBOC__
    #pragma hdrstop
#endif

/* - longitud de la pila para compiladores BORLAND - */
#ifdef __TURBOC__
    extern unsigned _stklen = 15000 ;
#endif

#include "bdd.i"          /* definición de las bases de datos */

void menuxy(char *S,int px,int py)
{
    gotoxy(px,py);
    printf("%s",S);
}
/* ----------------------------------------------------
   procedimientos para dibujar el laringograma en pantalla
   ---------------------------------------------------- */
void PANTALLA()
{
    /* PANTALLA GRAFICA */

    int xmax=getmaxx()-29;
```

```
    int ymax=getmaxy()-29;
    int colorac=getcolor();

    outtextxy(92,32," GRAFICO DEL LARINGOGRAMA ");
    setlinestyle (0,0,3);
    setcolor(3);
    moveto(100 ,50);
    lineto(xmax,50);
    lineto(xmax,ymax);
    lineto(100,ymax);
    lineto(100 ,50);
    setlinestyle (0,0,0);

    setfillstyle(SOLID_FILL,3);
    bar(100,ymax-20,xmax,ymax);
    setcolor(0);
    outtextxy(120,ymax-8,"  1 -> 100%   2 -> 50%   3 -> 20%   <S/s> salir");
    setcolor(colorac);
}

void graficar(unsigned char *cad)
{
    int k,gdriver=DETECT,gmode,errorcode;
    int ymax;
    int limite=512,factor=1;
    char op1;
    initgraph(&gdriver,&gmode,"");
    errorcode=graphresult();
    if(errorcode != grOk)
    {
       printf ("NO EXISTE MANEJADOR DE GRAFICOS: %s\n",
       grapherrormsg(errorcode));
       return;
    }
    ymax = getmaxy()-29;
    do {
          clearviewport();
```

```
            PANTALLA();
            setcolor(getmaxcolor()-1);
            moveto(101,320);
            for(k=0; k<limite ; k++)
            lineto((k*factor)+101,ymax - 100 - cad[k] );
            op1=getch();
            switch(op1) {
               case '1': factor = 1; limite = 512; break;
             case '2': factor = 2; limite = 256; break;
             case '3': factor = 5; limite = 100; break;
               }
            } while((op1!='S')&&(op1!='s'));
               closegraph();
               return;
    }

/* ----------------------------------------------------
funcion....: fcrear()
objetivo...: crear un archivo de datos y su indice
argumentos.: fpaciente = nombre del archivo de pacientes
flaringo = nombre del archivo de laringograma
retorna....: 0 si fallo creacion del archivo.
1 si se creo el archivo.
notas.....: no verifica si existen los archivos o no
---------------------------------------------------- */

int fcrear(char *fpaciente,char *flaringo)
{
    /* crea archivo de pacientes paciente.dbf */
    cb1.safety = 0;
    data1 = d4create( &cb1, fpaciente, paciente, tag_paciente );
    d4close(data1);

    /* crea archivo de laringogramas laringo.dbf */
    cb2.safety = 0;
    data2 = d4create( &cb2, flaringo, laringograma, tag_laringo );
    d4close(data2);
```

```
    return(1);
}

/* ----------------------------------------------------
funcion....: fabrir()
objetivo...: abre archivos de datos y su indices
argumento..: fpaciente   archivo de pacientes
flaringo    archivo de laringogramas
retorna....: 0 si fallo apertura del archivo.
1 si se efectuo apertutura del archivo.
---------------------------------------------------- */

int fabrir(char *fpaciente, char *flaringo)
{
    int file1;
    /*comprueba existencia de archivos de datos*/
    if( (file1=open(fpaciente,O_RDONLY)) == -1 ){
        close(file1);
    return(0);
}
    else close(file1);
    if( ( file1=open(flaringo,O_RDONLY) ) == -1 ) {
        close(file1);
    return(0);
    }
    else close(file1);
    /* apertura del archivo de pacientes */
    data1 = d4open( &cb1, fpaciente );
    if(data1 == 0)
    return 0;
/* asignacion de campos a punteros */
codigo1   = d4field( data1, "CODIGO1");
paciente1 = d4field( data1, "PACIENTE1");
obs1      = d4field( data1, "OBS1" );

/* apertura del archivo de laringogramas */
data2 = d4open( &cb2, flaringo );
```

```
if(data1 == 0)
      return 0;
      /* asignacion de campos a punteros */
      codigo2 = d4field( data2, "CODIGO1");
      fecha2  = d4field( data2, "FECHA2");
      obs2    = d4field( data2, "OBS2" );
      datos2  = d4field( data2, "DATOS2" );
      arch2   = d4field( data2, "ARCHIVO" );
      return 1;
}

/* ------------------------------------------------------
funcion....: fcerrar()
objetivo...: abre un archivo de datos y su indice
argumento..: < ninguno >
retorna....: < sin retorno >
------------------------------------------------------ */
void fcerrar(void) {
      d4close_all( &cb1 );
      d4close_all( &cb2 );
}

/* ------------------------------------------------------
funcion....: nuevopaciente
objetivo...: aumentar un nuevo paciente
argumento..: < ninguno >
retorna....: < sin retorno >
------------------------------------------------------ */
void nuevopaciente(void)
{
      char *mcodigo = NULL;   /* variable para el campo mcodigo */
      char *mnombre = NULL;   /* variable para el campo nombre */
      char *mobs    = NULL;   /* variable para el campo observación */
      int pos;                /* posición del registro en la bdd */
      clrscr();
      mcodigo = (char *) calloc(3,1);
      mnombre = (char *) calloc(30,1);
```

```
        mobs    = (char *) calloc(40,1);
        menuxy("Ingreso de nuevo paciente:",10,5);
        menuxy("-------------------------------------",10,6);
        basura=getc(stdin);
        menuxy("codigo.....: ",10,8);  gets(mcodigo);
        menuxy("nombre.....: ",10,10); gets(mnombre);
        menuxy("observaciones.....: ",10,12); gets(mobs);
        pos=d4seek(data1,mcodigo);
        if(pos!=0) {
        d4append_start(data1,0);
        m4assign(paciente1,mnombre);
        m4assign(codigo1,mcodigo);
        m4assign(obs1,mobs);
        d4append(data1);
        menuxy(".....grabado\n",10,18);
}
else {
        menuxy("Clave repetida, no se graba...",10,18);
        basura=getch();
        }
free(mcodigo);
free(mnombre);
free(mobs);
}

/* ----------------------------------------------------
funcion....: nuevolaringo
objetivo...: aumentar un nuevo laringograma
argumento..: el código del paciente
retorna....: < sin retorno >
---------------------------------------------------- */
void nuevolaringo(char *mcodigo)
{
        char *mfecha  = NULL;
        char *mobs    = NULL;
        char *mdatos  = NULL;
        char *fnombre = NULL;
```

```
FILE *fdata;
int i;

clrscr();
cuadro(1,1,79,24,"┌┐ │┘─└│");
mfecha  = (char *) calloc(9,1);
mobs    = (char *) calloc(51,1);
mdatos  = (char *) calloc(512,1);
fnombre = (char *) calloc(40,1);
menuxy("Ingreso de datos de nuevo laringogramas",10,5);
menuxy("──────────────────────────────────────────────────",10,6);
menuxy("codigo.....: ",10,8);  printf("%s",mcodigo);
basura = getc(stdin);
menuxy("fecha.....: ",10,10); gets(mfecha);
menuxy("observaciones.....: ",10,12); gets(mobs);
menuxy("Archivo laringograma...: ",10,14); gets(fnombre);
if((fdata=fopen(fnombre,"r"))==NULL) {
   menuxy("ERROR: no existe archivo de datos ",10,18);
   printf("<%s>",fnombre);
   basura = getch();
}
else {
   for(i=0;i<512;i++)
        mdatos[i]=getc(fdata);
        /* aumenta registro */
        d4append_start( data2, 0 );
        m4assign( codigo2 , mcodigo );
        m4assign( fecha2 , mfecha );
        m4assign( obs2 , mobs );
        m4assign( arch2 , fnombre );
        m4assign_n( datos2 , mdatos, 512 );
        d4append( data2 );
        menuxy(".....grabado presione cualquier tecla\n",10,18);
        basura=getch();
   }
   free(mfecha);
   free(mobs);
```

```
        free(mdatos);
    }

void feditareg(char *mcod)
{
    F4FIELD *field_ptr;
    char op;
    int encontrado=1;

    clrscr();
    d4seek(data2,mcod);
    field_ptr =  d4field_j( data2, 1 );
    if(strcmp(m4str(field_ptr),mcod) != 0)
        nuevolaringo(mcod);
    do {
        clrscr();
        menuxy("Edicion de paciente",10,5);
        field_ptr =  d4field_j( data2, 1 );
        if(strcmp(m4str(field_ptr),mcod) == 0) {
            menuxy("codigo paciente...: ",10,8); printf("%s",m4str(field_ptr));
            field_ptr =  d4field_j( data2, 2 ) ;
            menuxy("fecha.....: ",10,10); printf("%s",m4str(field_ptr));
            field_ptr =  d4field_j( data2, 3 ) ;
            menuxy("observaciones.....: ",10,12); printf("%s",m4str(field_ptr));
            field_ptr =  d4field_j( data2, 5 ) ;
            menuxy("Archivo .....: ",10,14); printf("%s",m4str(field_ptr));
            menuxy("(G)RAFICAR, (B)ORRAR, (S)ALIR (N)UEVO: ",10,18);
            op=getch();
            if( (op=='g') || (op=='G') ) {
                field_ptr =  d4field_j( data2, 4 );
                graficar((unsigned char *)m4str(field_ptr));
            }
            if( (op=='b') || (op=='B') ) d4delete(data2);
            if( (op=='s') || (op=='S') ) encontrado=0;
            if( (op=='n') || (op=='N') ) nuevolaringo(mcod);
            d4skip(data2,1);
            }
```

```
            else (encontrado=0);
            } while(encontrado);
        }

/* ----------------------------------------------------
   procedimientos para menu
   ---------------------------------------------------- */
void menu(char *S,int p) { gotoxy(25,p+7); printf("%s",S); }

int menuprincipal(void) {
    int y=1,x1=1,y1=1,x2=79,y2=24;
    char t1[] = " LARINGOGRAMAS ";
    char *tit;
    int i=1,op;

    ventana(x1,y1,x2,y2,WHITE,BLUE);
    gotoxy((int)(x2-x1-strlen(t1)+1)/2,y); printf("%s",t1);
    tit = "- MENU PRINCIPAL -";
    gotoxy((int)((80 - strlen(tit))/2),4); printf("%s",tit);
    menu("1.- Pacientes",i++);
    menu("2.- Adquisición de datos",i++);
    menu("3.- mantenimiento de datos",i++);
    menu("0.- Salir",i+2);
    menu("Ingrese su opción: ",i+4);
    scanf("%d",&op);
    return op;
}

/* ----------------------------------------------------
procedimientos para pacientes
---------------------------------------------------- */
void pacientes(void) {
    char mcodigo[3];
    int rini,i=2,pos;
    int opcion=1;
    F4FIELD *field_ptr,*mcod;
```

```
d4top(data1);
while(opcion!=0) {
    clrscr();
    cuadro(1,1,79,24,"┌┐ │┘─└│");
    d4top(data1);
    menuxy("Edicion de datos del paciente:",10,5);
    menuxy("──────────────────────────────────",10,6);
    menuxy("1. listado",10,8);
    menuxy("2. editar paciente",10,9);
    menuxy("3. nuevo paciente",10,10);
    menuxy("4. borrar paciente",10,11);
    menuxy("0. terminar ",10,14);
    menuxy("Ingrese su opción: ",10,16);
    scanf("%d",&opcion);
    switch(opcion) {
        case 1:
        clrscr();
        menuxy("LISTADO DE PACIENTES",3,2);
        menuxy("cod nombre                    observaciones",3,3);
        menuxy("──────────────────────────────────────────────────",3,4);
        cuadro(1,1,79,24,"┌┐ │┘─└│");
        i=5;
        for(d4top(data1);!d4eof(data1);d4skip(data1,1)) {
        field_ptr =  d4field_j( data1,1);
        menuxy(m4str(field_ptr),7,i);
        field_ptr =  d4field_j( data1,2);
        menuxy(m4str(field_ptr),3,i);
        field_ptr =  d4field_j( data1,3);
        menuxy(m4str(field_ptr),38,i++);
        if(i==20) {
            menuxy("Presione cualquier tecla para continuar...",10,23);
            basura = getch();
            clrscr();
            cuadro(1,1,79,24,"┌┐ │┘─└│");
            menuxy("LISTADO DE PACIENTES",2,3);
            menuxy("cod nombre                    observaciones",3,3);
            menuxy("──────────────────────────────────────────────────",3,4);
```

```
                i=5;
              }
          }
          menuxy("————————————————————————————————————————",3,22);
          menuxy("Fin del archivo de datos...",10,23);
          basura = getch();
          break;
          case 2:
              menuxy("ingrese codigo de paciente:",10,18);
              scanf("%s",&mcodigo);
              pos=d4seek(data1,mcodigo);
              if(pos==0)
                feditareg(mcodigo);
              else {
                menuxy("No existe ese código, presione cualquier tecla",10,20);
              basura=getch();
              }
        break;
        case 3: nuevopaciente(); break;
        case 4:
              menuxy("ingrese codigo de paciente a ser borrado:",10,18);
              scanf("%s",&mcodigo);
              pos=d4seek(data1,mcodigo);
              if(pos==0)
                 d4delete(data1);
                 pos=d4seek(data2,mcodigo);
              if(pos==0) {
                 do {
                      d4delete(data2);
                      d4skip(data2,1);
                      mcod = d4field(data2,"CODIGO1");
                 } while ( memcmp(f4ptr(mcod),mcodigo,3) == 0 );
                 }
              break;
      }
    }
}
```

```
void main() {

    int opcion;

    /* inicializa manejadores de archivos para el codebase */
    d4init(&cb1);           /* manejador del archivo de pacientes */
    d4init(&cb2);           /* manejador del archivo de laringogramas */
    /* abre los archivos, si no existen los crea */
    if(fabrir("paciente.dbf","laringo.dbf") == 0 ) {
        printf("archivos de datos no existen...\n");
        basura = getch();
        fcrear("paciente.dbf","laringo.dbf");
        printf("se generaron archivos de datos...\n");
        basura = getch();
        fabrir("paciente.dbf","laringo.dbf");
        printf("se abrieron archivos de datos...\n");
        basura = getch();
    }

    /* pantallas */
    esquema1();
    esquema2();
    esquemar();
    /* corrida del menu principal */
    do {
        opcion = menuprincipal();
        switch(opcion) {
            case 1:
    pacientes();
    break;
            case 2:
    clrscr();
    system("osc1.exe");
    break;
            case 3:
    /* empaquetar los datos */
    d4pack(data1);
```

```
        d4pack(data2);
        /* reordenar los datos */
        d4reindex(data1);
        d4reindex(data2);
        break;
           }
        } while(opcion!=0);
        clrscr();
        fcerrar();
}
```

ANEXO:

LISTADO DE PROGRAMAS EN LENGUAJE ASSEMBLY

ANEXO: LISTADO DE PROGRAMAS EN ASSEMBLY.

LISTADO DEL ARCHIVO: OSC1.ASM

```
PAGE 60,80
NAME OSC
TITLE LARINGOGRAMA DIGITAL CON MEMORIA
;****************************************
;* ESTE PROGRAMA ES USADO PARA ADQUIRIR *
;* DATOS CON UN ADC0804, A TRAVES DEL - *
;* PUERTO PARALELO DEL IMB PC MEDIANTE  *
;* LECTURA DE 512 MUESTRA DE SENAL ANA- *
;* LOGA, MOSTRARLOS EN LA PANTALLA,Y LUE *
;* GO ALMACENARLOS EN EL DISCO.         *
;* PROGRAMA RELIZADO POR OLGA RIVERA    *
;****************************************

SSEG    SEGMENT STACK
SSEG    ENDS
;************************************
CSEG    SEGMENT
ASSUME  CS:CSEG,SS:SSEG,DS:CSEG
START:  PUSH CS
        POP DS
        MOV WORD PTR RETURN+2, ES

ENTRADA: JMP INICIAL

;******* SOFTWARE DE INTERUPCIONES *******
FCALL   EQU 21H         ;INT DE FUNCIONES DEL DOS
VIDEO   EQU 10H         ;SERVICIO DE VIDEO DE ROM
KEYB    EQU 16H         ;SERVICIO DE TECLADO DE ROM
;********** FUNCIONES **************

PRINT   EQU 09H         ;FUNCION IMPRIMIR STRING
WPIXEL  EQU 0CH         ;ESCRIBIR UN PIXEL
SETMODE EQU 00H         ;COLOCAR MODO DE VIDEO
```

```
;****** DATOS MICELANEOS ************
LF          EQU 0AH
CR          EQU 0DH
;********* DIRECCIONES I\O ***********
OUTCM       EQU 037AH       ;DIRECC.DEL COMANDO DE SALIDA DEL IBM PC
OUTDT       EQU 0378H       ;DIRECC. SALIDA DE DATOS DEL IBM PC
INDT        EQU 0379H       ;DIRECC. DE ENTRADA DEL IBM PC
RETURN      DD  00000000H
;********* DATA ***********************
VALOR       DB 0
nHandle     DW 0
cBuffer     DB 32
nLongitud   DB ?
FILNAM      DB 32 DUP(0)
            DB 0,'$'
SALTE       DB CR,LF,'$'
TEXT4       DB 'Archivo : ','$'
TEXT1       DB 9,9,9,9,'** LARINGOGRAMA DIGITAL **','$'
TEXT2       DB '  BORRAR=>B  REDIBUJAR=>R  MEMORIZAR=>M  GRABAR=>G
              SALIR=>S',CR,LF,'$'
TEXT3       DB ' INGRESE NOMBRE DEL ARCHIVO :','$'
ERR2        DB ' ERROR DURING WRITING',CR,LF,'$'
Y0          DW  ?           ;Y COORDENADA
Y1          DW  ?           ;Y COORDENADA
SCREEN      DW  351
COLOR       DB  0FH         ;BLANCO
MDATA       DB  ?
VDATA       DW  ?           ;DATOS MARCADOS
ARRAY       DB  512 DUP(?)  ;512  NUMERO DE DATOS EN MEMORIA
TIME1       DW  5           ;RETARDO DE TIEMPO

;************************************
;*     PROGRAMA PRINCIPAL        *
;************************************

; ****** PROGRAMA PARA NOMBRAR EL ARCHIVO *******
```

```
INICIAL:    MOV DX,OFFSET TEXT3   NOMBRE DE ARCHIVO ?
            MOV AH,PRINT
            INT FCALL
            MOV CX,0

            MOV AH,0AH            ; TRAER CARACTERES DEL TECLADO
            MOV DX,OFFSET cBuffer ; DIRECCIONAR BUFFER DE TECLADO
            INT FCALL             ; REALICE ESTO
            MOV AH,0
            MOV AL,nLongitud      ; TRAER LONGITUD DE BUFFER
            MOV SI,AX
            MOV FILNAM[SI],0      ; DIRECCIONAR NOMBRE A FILNAM
            MOV FILNAM[SI+1],'$'


;******* MODO DE VIDEO **********
INIT:
            MOV AL,12H            ;VGA 640X480 16
            MOV AH,SETMODE        ;SET MODE OF VIDEO
            INT VIDEO             ;DO IT
;*********************************
            MOV DX,OFFSET TEXT4   ;TEXTO ARCHIVO:
            MOV AH, PRINT
            INT FCALL
                                  ; NOMBRE DEL ARCHIVO
            MOV DX,OFFSET FILNAM
            MOV AH, PRINT
            INT FCALL

            MOV DX,OFFSET SALTE   ; NUEVA LINEA
            MOV AH, PRINT
            INT FCALL

            MOV DX,OFFSET TEXT1   ; TITULO DE PANTALLA
            MOV AH,PRINT
            INT FCALL
```

```
        MOV AH,02H              ; SERVICIO POSICION DE CURSOR
        MOV DX,0203H            ; LINEA 02 COLUM 03
        INT 10H

        MOV DX,OFFSET TEXT2     ; LISTA DE MENUS
        MOV AH,PRINT
        MOV AL,0EH
        INT FCALL


CLS:  CALL SCALES

;       SUBRUTINA PARA ADQUISICION DE DATOS
;     ****************************************

ADC1:   MOV DI,00H      ;RESET CONTADOR DE MEMORIA
        CLI             ;INTERRUPCION DEL PC

ADC:    MOV DX,OUTCM        ;RESET DEL CONVERSOR AD
        MOV AL,11111011B    ;WR- ->BAJO
        OUT DX,AL
        MOV AL,11111101B    ;WR- ->ALTO
        OUT DX,AL       ;SELECION DE LOS BYTES MS
        MOV CX,TIME1
DELAY: DEC CX
        JNE DELAY

        MOV DX,INDT     ;DIRECC DEL PORTICO DE ENTRADA
        MOV CL,4
NEXT:   IN AL,DX        ;INGRESO DATOS ->AL
        SAR AL,CL       ;TEST AL,08H PARA BIT 3=H
        JC NEXT         ;NO LEA SALTE AL SIGUIENTE

;       MUESTREO DE DATOS

        IN AL,DX        ;INGRESE LA  MUUESTRA
        NOT AL          ;INVERT DATO
```

```
        ADD AL,80H      ;INVERT BIT 7
        MOV BL,AL
        AND BL,0F0H     ;BORRE BYTE LS
        MOV DX,OUTCM    ;DIRECC. PORTICO DE SALIDA
        MOV AL,11111111B    ;SELECT BIT 0 PARA SELEC. DEL MUX
        OUT DX,AL       ;PARA LSB
        MOV DX,INDT     ;DIRECC DE ENTRADA DE DATOS
        IN AL,DX        ;INGRESE MUESTRA
        NOT AL          ;INVERT DATO
        ADD AL,80H      ;INVERT BIT 7
        MOV CL,4        ;SHIFT COUNT
        SAR AL,CL
        AND AL,0FF      ;BORRAR BYTE MS
        OR BL,AL        ;DATO -> BL

;       ALMACENAR EN MEMORIA

        MOV OFFSET ARRAY[DI],BL
        INC DI          ;INCREMENTE LA DIRECC. DE MEM
        CMP DI,511      ;ULTIMA ADDRESS ?
        JLE ADC         ;SIGUIENTE MUESTRA
        STI             ;HABILITAR INTERRUP.
        CALL VIDEOUT
        CALL VDCANC

;************** COMANDOS ******************
CMMD:   MOV AH,01H      ;ENTRADA DESDE TABLERO
        INT KEYB
        JZ NOCHAR       ;NO CARACTER
ANALOG: MOV AH,00H      ;TOME UN CARACTER
        INT KEYB
        CMP AL,53H      ;SI S/s SALGA DEL PROGRAMA
        JE OUT
        CMP AL,73H
        JE OUT

        CMP AL,44H      ;SI D VISUALIZAR
```

```
                JE NOCHAR       ;NUEVO MUESTREO
                CMP AL,64H
                JE NOCHAR
                CMP AL,47H      ;SI G ESCRIBA EN EL DISCO
                JE  WRDISK
                CMP AL,67H
                JE  WRDISK
                CMP AL,52H      ;SI R REDUBUJAR ESCALA
                JE RDRW
                CMP AL,72H
                JE RDRW
                CMP AL,4DH      ;SI M MEMORICE EN LA PANTALLA
                JE MEM
                CMP AL,6DH
                JE MEM
                CMP AL,42H      ;SI B BORRAR PANTALLA
                JE CLSCRE
                CMP AL,62H
                JE CLSCRE
                JMP ANALOG

;************************************
NOCHAR:   JMP ADC1          ;TO NEW FRAME
;******* REDUBUJAR ESCALAS **********
RDRW:     JMP CLS

;********** ESCRIBIR EN DISCO ******
WRDISK:   CALL WRDSK
          JMP CMMD

;******* LIMPIAR PANTALLA **********
CLSCRE:   JMP INIT

;*********** MEMORIA ***************
MEM:      MOV DI,00H        ; RESETEAR EL PUNTERO
          CALL VIDEOUT
          JMP ANALOG
```

```
;********** SALIR DEL PROGM.***********
OUT:        MOV AL,03H      ;80x 25
            MOV AH,SETMODE  ;COLOCAR MODO DE VIDEO
            INT VIDEO       ;HACER ESTO
            JMP RETURN      ;SALIR SEL PROGM.

;************* SALIDA DE VIDEO *************
VIDEOUT PROC NEAR
VD:         MOV DI,00H      ;CONTADDOR DE MEMORIA
            MOV CX,100      ;X COORDENADA
            MOV AX,SCREEN   ;TRACE ELVALOR DE LA LINEA
            MOV BL,OFFSET ARRAY[DI]   ;COPIAR LOS DATOS
            INC DI
            SUB AX,BX
            MOV Y0,AX       ;ALMACENE EL VALOR Y
;           IMPRESION EN PANTALLA
V1:         MOV DX,Y0       ;Y COORDENADA
            MOV AX,0C0FH    ;ESCRIBIR PIXEL COLOR BLANCO
            INT VIDEO
            INC CX          ;INCREMENT X COORD.
            CMP CX,600      ;ES ULTIMO PIXEL
            JGE VDOUT       ;SI ES ASI IR A SALIDA
                            ;CALCULAR NUEVA COORDENADA

            MOV AX,SCREEN   ;TRAZAR EL VALOR DE LA LINEA
            MOV BL,OFFSET ARRAY[DI]   ;COPIAR LOS DATOS
            INC DI
            SUB AX,BX

            MOV Y1,AX       ;ALMACENE EL NUEVO VALOR
V2:         MOV AX,Y1
            CMP AX,Y0       ;DELTA Y POSITIVO
            JGE V3          ;
            DEC Y0
;           IMPRIMIR EN PANTALLA
            MOV DX,Y0       ;Y COORDENADA
            MOV AX,0C0FF    ;ESCRIBIR PIXEL CON COLOR
```

```
                INT VIDEO
                JMP V2
V3:             MOV AX,Y1
                CMP AX,Y0       ;DELTA Y NEGATIVO
                JE V1
                INC Y0

;               IMPRIMIR EN PANTALLA
                MOV DX,Y0       ;Y COORDENADA
                MOV AX,0C0FH    ;ESCRIBIR PIXEL CON COLOR
                INT VIDEO
                JMP V3
VDOUT:          RET
VIDEOUT         ENDP

;************** BORRADO  DE PANTALLA **************
VDCANC PROC NEAR
                MOV DI,00H      ;CONTADOR DE MEM
                MOV CX,100      ;X COORDENADA

;               CALCULO DE COORDENADA

                MOV AX,SCREEN ;TRACE EL VALOR DE LA LINEA
                MOV BL,OFFSET ARRAY[DI]    ;COPIE EL DATO
                INC DI
                SUB AX,BX
                MOV Y0,AX       ;ALMACENE EL VALOR DE Y
;               BORRE EL PIXEL DE LA PANTALLA
V1C:            MOV DX,Y0       ;Y COORDENNADA
                MOV AX,0C00H    ;ESCRIBIR PIXEL EN NEGRO
                INT VIDEO
                INC CX          ;INCREMENT X COORD.
                CMP CX,600      ;ES EL ULTIMO PIXEL
                JGE OUTCANC     ;SI ES ASI SALGA DEL PROCEDIMIENTO
;               CALCULE LA NUEVA COORDENADA
                MOV AX,SCREEN ;TRACE EL VALOR DE LA LINEA
                MOV BL,OFFSET ARRAY[DI]    ;COPIE EL DATO
```

```
            INC DI
            SUB AX,BX
            MOV Y1,AX       ;ALMACENE EL NUEVO VALOR
V2C:        MOV AX,Y1
            CMP AX,Y0       ;DELTA Y POSITIVO
            JGE V3C         ;
            DEC Y0
;           BORRE EL PIXEL DE LA PANTALLA
            MOV DX,Y0       ;Y COORDENADA
            MOV AX,0C00H    ;ESCRIBA PIXEL CON COLOR
            INT VIDEO
            JMP V2C
V3C:        MOV AX,Y1
            CMP AX,Y0       ;DELTA Y NEGATIVO
            JE V1C
            INC Y0

;           BORRAR EL PIXEL DE LA PANTALLA

            MOV DX,Y0       ;Y COORDENADA
            MOV AX,0C00H    ;ESCRIBA PIXEL CON COLOR
            INT VIDEO
            JMP V3C
OUTCANC:    RET
VDCANC      ENDP
;******* ESCRITURA EN EL DISCO *********
WRDSK  PROC NEAR
            MOV AH,3CH      ; FUNCION ABRIR ARCHIVO
            MOV CX,32       ; ATRIBUTO DEL ARCHIVO
            MOV DX,OFFSET FILNAM ; UBICAR EL NONBRE DEL ARCHIVO
            INT FCALL
            MOV nHandle,AX  ; CODIGO DE ASIGNACION DEL DOS
            MOV BX,AX       ; COLOCAMOS EL CODIGO EN BX
            MOV AH,40H      ; FUNCION ESCRIBIR EN EL ARCHIVO
            MOV CX,512      ; NUMERO DE BITS A ESCRIBIR
            MOV DX,OFFSET ARRAY  ; DIRECCION DE LOS DATOS
            INT FCALL       ; HACER ESTO
```

```
                MOV AH,3EH      ; CERRAR ARCHIVO
                MOV BX,nHandle  ; CUAL CODIGO DESEA CERRAR
                INT FCALL       ; EJECUTE LA FUNCION
                MOV AH,2        ; FUNCION DE ESCRIBIR
                MOV DL,7        ; CARACTER DE PITO
                INT FCALL
                RET
WRDSK  ENDP
;******** ESCALAS DE PANTALLA ************
SCALES PROC NEAR
                MOV AL,03H      ;COLOR DE PIXEL
                MOV BH,00H      ;VIDEO PAG

                MOV DX,50       ;Y 1RA COORDENADA
INI:            MOV CX,100      ;X 1RA COORDENADA
LSUP:           MOV AH,WPIXEL   ;ESCRIBA PRIMER PIXEL
                INT VIDEO       ;HACER ESTO
                INC CX
                CMP CX,601
                JL  LSUP
                ADD DX,50
                CMP DX,451
                JL INI
                MOV CX,100      ;X 1RA COORDENADA
INI2:           MOV DX,50       ;Y 1RA COORDENADA
LSUP2:          MOV AH,WPIXEL   ;ESCRIBIR PRIMER PIXEL
                INT VIDEO       ;HACER ESTO
                INC DX
                CMP DX,451
                JL  LSUP2
                ADD CX,50
                CMP CX,601
                JL INI2

;******** ESCALAS DE PANTALLA **********

                MOV CX,100      ;X 1RA COORDENADA
```

```
INI3:     MOV DX,248      ;Y 1RA COORDENADA
LSUP3:    MOV AH,WPIXEL   ;ESCRIBIR PRIMER PIXEL
          INT VIDEO       ;EJECUTE ESTO
          INC DX
          CMP DX,253
          JL  LSUP3
          ADD CX,10
          CMP CX,601
          JL INI3
          MOV DX,50       ;Y 1RA COORDENADA
INI4:     MOV CX,348      ;X 1RA COORDENADA
LSUP4:    MOV AH,WPIXEL   ;ESCRIBIR PRIMER PIXEL
          INT VIDEO       ;HACER ESTO
          INC CX
          CMP CX,353
          JL  LSUP4
          ADD DX,10
          CMP DX,451
          JL INI4
          RET
SCALES ENDP
;**********************************
CSEG      ENDS

          END START
```

ANEXO:

INSTRUCCIONES DEL MICROPROCESADOR Y FUNCIONES DEL CODEBASE

ANEXO:
INSTRUCCIONES DEL μPROCESADOR

El MICROPROCESADOR

En todos los PC, el microprocesador es el chip que ejecuta los programas. El microprocesador o unidad central de proceso: CPU (Central Process Unit) lleva a cabo una gran variedad de cálculos, comparaciones numéricas y transferencia de datos como respuesta a las peticiones de los programas almacenados en la memoria.

La CPU controla las operaciones básicas del ordenador enviando y recibiendo señales de control, direcciones de memoria y datos de un lugar a otro del ordenador a través de un grupo de sendas electrónicas llamadas bus. Localizadas a lo largo del bus, están las puertas de entrada y salida (E/S), las cuales conectan a la memoria y a los chips de apoyo al bus. Los datos pasan a través de estas puertas de E/S mientras viajan desde y hasta la CPU y otras partes del ordenador.

En los IBM PC y PS 2, la CPU siempre pertenece a la familia de microprocesadores Intel.

El Microprocesador 8088

El 8088 es el microprocesador de 16 bits que controla los ordenadores personales estándar de IBM, incluidos el PC original, el PC/XT, el PC portable y el PCjr. Prácticamente, cada bit de datos que entra o sale del ordenador pasa por la CPU para ser procesado.

Dentro del 8088, 14 registros proporcionan un área de trabajo para la transferencia de datos y proceso. Estos registros internos, que forman un área con un tamaño de 28 bytes, son capases de almacenar temporalmente datos, direcciones de memoria, instrucciones de punteros e indicadores, de estado y control. A través de estos registros, el 8088 puede acceder a 1 MB (megabyte)2, o sea, más de un millón de bytes de memoria.

EL Microprocesador 8086

El 8086 se utiliza en los modelos 25 y 30 del PS 2 (y también en muchos compatibles). El 8086 difiere del 8088 solamente en un detalle: utiliza un bus de datos de 16 bits en vez del bus de 8 bits que utiliza el 8088. Básicamente cualquier cosa que usted lea acerca del 8086 es también aplicable al 8088: para los efectos de la programación, considérelos idénticos.

EL Microprocesador 80286

El 80286 se utiliza en los PC/AT y en los PS 2, modelos 50 y 60. Aunque es totalmente compatible con el 8086, el 80286 dispone de características extra de programación que le permiten ejecutar los programas mucho más rápido que el 8086. Quizá la más importante prestación del 80286 sea su capacidad de funcionar en multitarea.

EL Microprocesador 80386

El PS/2 modelo 80 utiliza el 80386, un microprocesador más rápido y potente que el 80286. El 80386 soporta las mismas funciones básicas que el 8086 y ofrece el sistema de gestión de memoria en modo protegido que el 80286. Sin embargo, el 80386 ofrece dos importantes ventajas sobre su predecesor:

El 80386 es un microprocesador de 32 bits con 32 registros.
Puede desarrollar cálculos y direccionar memoria con 32 bits en vez de con 16.

El 80386 ofrece una gestión de memoria más flexible que la del 80286 y 8086.

EL Controlador Programable de Interrupciones

En un PC o PS/2, una de las tareas esenciales de la CPU consiste en responder a las interrupciones del hardware. Una interrupción del hardware es una señal generada por un componente del ordenador que indica que ese componente requiere la atención de la CPU. Por ejemplo, el reloj del sistema, el teclado y los controladores de disco generan todos interrupciones de hardware en distintos momentos. La CPU responde a cada interrupción llevando a cabo la actividad de hardware apropiada, como incrementando en un día el contador de tiempo o procesando una pulsación de tecla.

Controladores de Entrada/Salida

Los PC y PS/2 tienen varios subsistemas de entrada salida con circuitería de control especializada que proporciona un interfaz entre la CPU y el hardware de E/S. Por ejemplo, el teclado tiene un chip controlador propio que transforma las señales eléctricas producidas por las pulsaciones de teclas en un código de 8 bits que representa la tecla pulsada. Todas las unidades de disco disponen de circuitería independiente que controla directamente la unidad;

la CPU se comunica con el controlador a través de un interfaz coherente. Los puertos serie y paralelo también disponen de sus propios controladores de entrada/salida.

Como se Comunica el 8086

El 8086, el 80286 interactúan con la circuitería que les rodea de tres maneras: a través del acceso directo e indirecto a memoria, a través de los puertos de entrada/salida (E/S) y con las señales denominadas interrupciones.

El microprocesador utiliza la **memoria** para leer o escribir valores en las posiciones de la memoria que se identifican por una dirección numérica. A las posiciones de memoria se puede acceder de dos formas; a través del controlador de acceso directo a memoria (DMA) o a través de los registros internos del microprocesador. La unidad de disco y los puertos serie de comunicaciones pueden acceder directamente a la memoria a través del controlador DMA. Todos los demás dispositivos transfieren los datos desde y hacia la memoria a través de los registros del microprocesador.

Las **Interrupciones** constituyen la forma en la que la circuitería extra al microprocesador informa de que algo ha sucedido (como que se ha pulsado una tecla) y solicita que se emprenda alguna acción.

Los Formatos de datos en el 8086

Datos Numéricos.- El 8086 y el 80286 son ambos capaces de trabajar solamente con cuatro formatos simples de datos, todos los cuales son valores enteros. Los formatos se distribuyen en dos bloques: el byte de 8 bits y la palabra de 18 bits (2 bytes). Estas dos unidades básicas están relacionadas con la capacidad de proceso de 16 bits del 8086. El byte es la unidad fundamental; y cuando el 8086 y el 80286 direccionan memoria, los bytes son las unidades básicas direccionadas. En un byte individual, estos microprocesadores pueden trabajar con números positivos sin signo comprendidos en un rango de 0 a 255 (esto es, 2 positividades). Si el número es un valor con signo, uno de los 8 bits representa en signo, así que sólo los 7 restantes representan el valor. De esta manera, un byte individual puede representar valores comprendidos en el rango de -128 a +127.

Datos de Caracteres.- Los datos de caracteres se almacenan en el formato estándar ASCII, con el cual cada carácter ocupa 1 byte. La familia 8086 no sabe nada acerca de los caracteres ASCII y los trata como a cualquier byte, con una excepción: el conjunto de instrucciones

acomoda la suma y sustracción decimal para su ejecución en caracteres binarios codificados en decimal, BCD (Binary Coded Decimal).

Almacenamiento Inverso de palabras

A pesar de que la memoria del PC está direccionada en unidades de bytes de 8 bits, muchas operaciones introducen palabras de 16 bits. En la memoria, una palabra de 16 bits se almacena en dos bytes adyacentes de 8 bits. El byte menos significativo de la palabra se almacena en la posición de memoria más alta. Desde algunos puntos de vista, el almacenar así una palabra es hacerlo a la inversa de como sería de esperar. Debido a la apariencia inversa de este esquema de almacenamiento es denominado algunas veces como "almacenamiento inverso de palabras".

Direccionamiento Segmentado

El 8086 divide el espacio de memoria direccionable en segmentos, cada uno de los cuales contiene 64 KB de memoria. Cada segmento comienza en una dirección de párrafo: esto es, la posición de un byte que es divisible por 16. Para acceder a bytes o palabras individuales, se utiliza un desplazamiento que apunta al byte exacto de un segmento determinado. Como el desplazamiento se mide siempre en relación al comienzo del segmento, se le llama dirección relativa o desplazamiento relativo.

Juntos, un segmento y un desplazamiento forman una dirección segmentada que puede designar un byte del espacio de memoria de 1 MB del 8086. El 8086 convierte una dirección segmentada de 32 bits en una dirección física de 20 bits utilizando el valor del segmento como un valor de párrafo y añadiéndole el valor del desplazamiento. En efecto, el 8086 mueve el valor del segmento 4 bits a la izquierda y le suma el valor del desplazamiento para crear una dirección de 20 bits.

Los Registros del 8086

El 8086 se diseñó para ejecutar instrucciones y realizar operaciones aritméticas y lógicas, así como para recibir instrucciones y transferir datos a y desde la memoria. Para hacer todo esto, utiliza diversos registros de 16 bits.

Hay catorce registros, cada cual para un uso especial. Cuatro registros de uso general utilizados por los programas para mantener temporalmente los resultados intermedios y los

operandos de las operaciones aritméticas y lógicas. Cuatro registros de segmento que contienen valores de segmentos. Cinco registros de índices y punteros que contienen los desplazamientos que se utilizan con los valores de los registros de segmentos para ubicar los datos en la memoria. Y por último, un registro de indicadores que contiene nueve indicadores de 1 bit que se utilizan para grabar información del estado y controlar las operaciones del 8086.

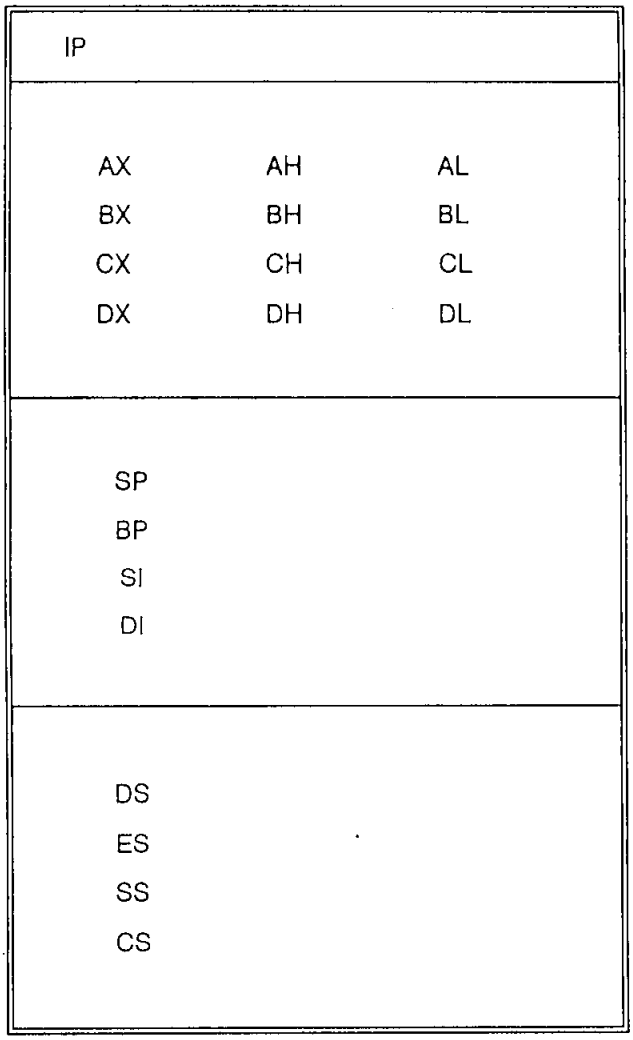
Los Registros de Datos o de Propósito General

Cuando un ordenador está procesando datos, una gran cantidad del tiempo del microprocesador se pierde en transferir los datos a y desde la memoria. El tiempo de acceso puede ser reducido, en gran medida, manteniendo los resultados y los operandos más frecuentes dentro del 8086. Hay cuatro registros de 16 bits, llamados normalmente registros de datos de uso general diseñados para éste propósito.

Estos registros se conocen como AX, BX, CX y DX. Cada uno de ellos puede ser subdividido y utilizado separadamente como registros de 8 bits de orden superior o más significativos se conocen como AH, BH, CH y DH, y los de orden inferior o menos significativos como AL, BL, CL y DL.

Los registros de datos se utilizan, casi siempre, como áreas de trabajo temporales, particularmente para operaciones aritméticas. Las sumas y las restas se pueden realizar en la memoria, pero los registros son más rápidos.

Flags	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



El registro AX (acumulador) es el registro principal para el cálculo de operaciones aritméticas.

El registro BX (base) se puede utilizar para apuntar al comienzo de una tabla de traducción en la memoria. También para contener el desplazamiento de una dirección segmentada.

El registro CX (contador) se utiliza como un contador de repeticiones para el control de bucles y traslados repetidos de datos.

El registro DX se utiliza para almacenar datos con propósitos generales.

Los Registros de Segmentos

El direccionamiento completo de una posición de memoria está compuesto por el valor, de 16 bits, del segmento, y por el valor, de 16 bits, del desplazamiento unido al segmento.

Hay cuatro registros, llamados CS, DS, ES y SS, que se emplean para identificar cuatro segmentos específicos de la memoria. Hay cinco registros de desplazamiento, los cuales trataremos brevemente, que se pueden utilizar para almacenar los desplazamientos relativos de los datos dentro de cada uno de los cuatro segmentos.

Cada registro de segmento se utiliza para un tipo específico de direcciones:

El registro CS identifica al segmento de código, el cual contiene el programa que se está ejecutando.

Los registros DS y ES identifican los segmentos de datos, donde se almacenan los datos que se utilizan en el programa.

El registro SS identifica el segmento de pila.

Los Registros de Desplazamiento

Hay cinco registros de desplazamiento que utilizan junto con los registros de segmento para contener las direcciones segmentadas. Un registro, llamado puntero de instrucción (IP), contiene el desplazamiento de la instrucción actual del segmento de código; dos registros llamados registros de pila, están íntimamente relacionados con la pila y los otros dos que quedan, llamados registros índice, se utilizan para direccionar cadenas de datos.

El puntero de instrucción, IP (Instruction Pointer), también llamado contador de programa, PC (Program Counter), contiene el desplazamiento en el segmento de código donde está ejecutando el programa actual. Se utiliza con el registro CS para seguir la pista de la siguiente instrucción que debe ser ejecutada.

Los programas no tienen acceso directo al registro IP, pero hay varias instrucciones, como JMP y CALL, que implícitamente cambian el valor de IP.

Los registros de pila, llamados el puntero a la pila, SP (Stack Pointer) y el puntero base, BP (Base Pointer), proporcionan los desplazamientos en el segmento de pila. SP proporciona la ubicación de la cima actual de la pila. Los programas rara vez cambian directamente el valor de SP. En vez de eso, cuentan con las instrucciones PUSH y POP para actualizar implícitamente SP, BP es el registro que generalmente se utiliza para acceder al segmento de pila directamente.

Los registros de índice, llamados el índice fuente, SI (Source Index) y el índice destino, DI (Destination Index), se puede utilizar para propósitos generales de direccionamiento de datos. También, todas las instrucciones de movimiento y comparación de cadenas utilizan SI y DI para direccionar las cadenas de datos.

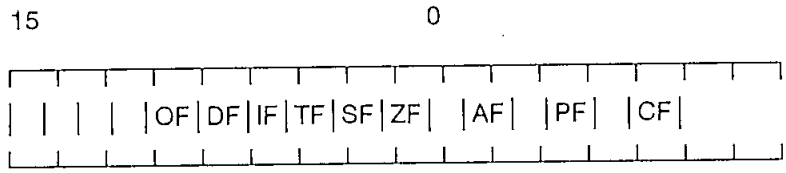
El Registro de Indicadores

El registro décimocuarto y último, llamado el registro de indicadores es, realmente una colección de bits individuales de estado y control llamados indicadores. Los indicadores se mantienen en un registro, de tal manera que pueden ser salvados o recuperados como un conjunto coordinado, o inspeccionados como datos ordinarios. Sin embargo, normalmente, se establecen y comprueban como elementos independientes, no como un conjunto.

En el registro de indicadores de 16 bits del 8086 hay nueve indicadores de 1 bit, dejando 7 bits sin uso. (El 80286 y el 80386 utilizan alguno de estos indicadores sin uso para soportar el trabajo en modo protegido). Los indicadores pueden dividirse lógicamente en dos grupos: seis indicadores de estado, los cuales registran información del estado del microprocesador (normalmente indican qué sucede con una comparación o con una operación aritmética), y tres indicadores de control, los cuales dirigen algunas de las instrucciones del 8086. Esté preparado para ver una gran diversidad de notaciones para los indicadores, incluido nombres distintos para definir si están activados o desactivados.

Banderas

Hay nueve banderas estándar en total, seis se utilizan para indicar resultados de operaciones. Las banderas de estado varían generalmente luego de una operación aritmética o lógica, para indicar propiedades del resultado de tales operaciones. Dentro de estas banderas tenemos:



ZF si el resultado es cero o comparación de igualdad.

SF indica signo.

CF "carry flag".

AF "auxiliary carry flag" (correspondiente a los cuatro primeros bits y usada para simular operaciones decimales).

OF "overflow flag".

PF "parity flag".

y las tres restantes se utilizan para controlar operaciones del procesador:

DF "direction flag" que controla la dirección de las rotaciones (movimientos a derecha o izquierda).

IF "interruption flag", que habilita o deshabilita las interrupciones externas.

TF "trap flag" que pone al procesador en modo paso a paso para depuración de programas.

En el AT existen dos banderas especiales:

NT para tareas de anidamiento.

IOPL controla el nivel de prioridad de entrada/salida.

Direcionamiento de la Memoria a través de los Registros

La memoria siempre se direcciona mediante la combinación de un valor segmeto y uno desplazamiento relativo, los valores de segmento proceden siempre de uno de los cuatro registros de segmento, por el contrario el desplazamiento relativo puede especificarse de muchas maneras diferentes (mírese la tabla de modos de direccionamiento), por cada instrucción de máquina que accede a la memoria, el 8086 calcula una dirección efectiva combinando una, dos ó tres de las siguientes:

El valor de BX o BP.

El valor SI o DI.

El valor del desplazmiento que el la parte propia de la instrucción.

Entre los posibles modos existentes para direccionar un operando se tienen:

- Inmediato: Aquel en el cual, en la instrucción se define un valor que actuará como operando.

- Registro: Cuando uno de los operandos de una instrucción es un registro.

- Directo: Cuando la dirección de memoria de un operando se especifica directamente en la instrucción que se realizará. Se debe observar que este tipo de direccionamiento es un caso especial de direccionamiento indirecto, pero en el cual el "offset" es cero.

- Indirecto: El direccionamiento indirecto define una dirección de memoria, en la cual se encuentrá el operando de una instrucción utilizando para el efecto: un registro base, un registro índice, un registro base al que se le suma el contenido de un registro índice, un registro base o índice al que se le suma un valor de "offset" y un registro base al que se le suman el valor contenido en un registro índice y un "offset".

Puede emplearse el método inmediato y directo cuando conozca el desplazamiento de una ubicación específica de memoria. Debe utilizarse otro de los métodos distintos cuando no pueda especificarse que dirección será hasta que no se ejecute el programa.

Puertos de Entrada/Salida E/S del 8086.

La Familia de microprocesadores del 8086 controla y se comunica con muchas partes del ordenador a través de los puertos de entrada y salida (E/S). Los puertos de E/S son las entradas a través de las cuales pasa la información, tanto como si viaja desde o hacia un dispositivo de E/S, como un teclado o una impresora.

Cada puerto se identifica con un número de puerto de 16 bits el cual puede variar en un rango de 00H a FFFFH (65535). La CPU identifica un puerto en particular por su número.

Como hace cuando accede a la memoria la CPU utiliza los buses de datos y direcciones como conductos para comunicarse con los puertos. Para acceder a los puertos la CPU manda primero una señal en el bus del sistema para avisar a todos los dispositivos de E/S de que la dirección que va en el bus es la de un puerto. Entonces la CPU envía la dirección del puerto, el dispositivo cuya dirección de dispositivo coincida responderá.

El número de puerto direcciona una ubicación en memoria que está asociada con un dispositivo de entrada salida, pero que no forma parte de la memoria principal. En otras palabras el número de puerto de E/S no es lo mismo que la dirección de memoria. Por ejemplo el puerto de ES 3D8H no tiene nada que ver con la dirección de memoria 003D8H. para acceder a un puerto no se utiliza la dirección de transferencia de datos como MOV y STOS. En

vez de ello se utiliza la s instrucciones INT y OUT que están reservadas para el acceso a los puertos de E/S.

Interrupciones del 8086.

Una interrupción es una indicación al microprocesador de que se necesita su inmediata atención, la familia del 8086 puede responder tanto a las interrupciones de hardware como de software.

Un dispositivo de hardware puede generar una señal de interrupción que es procesada por el Controlador Programable de Interrupciones. (PIC) y pasada al microprocesador.

En software la instrucción INT genera una interrupción. En ambos casos el microprocesador detiene el proceso y ejecuta una rutina residente en la memoria llamada **Rutina de tratamiento de las interrupciones**, después de que esta rutina de interrupción ha realizado su tarea, el microprocesador continúa con el proceso desde el punto en que se interrumpió.

El 8086 soporta 256 interrupciones diferentes, identificadas cada una por un número entre 00H y FFH (decimal 255).

Interrupciones de software.

Probablemente el tipo más familiar de interrupción es el generado por la instrucción INT.

La interrupción "software" generada intencionalmente por nuestros programas, que permite solicitar los servicios BIOS y DOS: Entre las principales tenemos.

- Interrupciones "software" del PC, para activar partes de los programas de la ROM-BIOS, como por ejemplo visualizar en pantalla.

- Interrupciones "software" del DOS.

- Interrupciones "software" de aplicaciones.

- Interrupciones de tabla, direccionadas por la tabla de vectores, en las cuales existen algunas que tienen número de interrupción pero no se pueden utilizar nunca, ya que no hay rutina de control de interrupción para ellas.

Interrupciones "hardware"

El microprocesador responde a las interrupciones de hardware de manera muy similar que las interrupciones de software, transfiriendo el control de una rutina de tratamiento de la interrupción, la diferencia importante recae en la manera en que se produce la señal de interrupción.

Los dispositivos como el contador de tiempo del sistema, teclado y el puerto de comunicaciones serie pueden génerar interrupciones en un conjunto de líneas reservadas de solicitud de interrupción. Cuando se da una interrupción de hardware particular, el número correspondiente de la interrupción se sitúa en el bus de datos del sistema donde el microprocesador puede controlarlo.

De las 8 señales de interrupción disponibles en los computadores personales XT compatibles, solo 6 están disponibles en las ranuras para las tarjetas de extensión. Las señales se utilizan para:

IRQ0	Reloj en tiempo real
IRQ1	Teclado
IRQ2	PC-Net
IRQ3	COM2
IRQ4	COM1
IRQ5	Disco duro
IRQ6	Diskette
IRQ7	Impresora (manejo del puerto IEEE-488)

Teclado

Es un dispositivo de entrada, que contiene los siguientes tipos de teclas: estándar, que corresponden a los existentes en una máquina de escribir; numéricas, que corresponden a las disponibles en una calculadora; de función, utilizadas para que sean reconocidas por un programa y desarrollen una función específica en dicho programa; y especiales.

La ROM BIOS define la interrupción 16H (22 decimal), para el teclado. La subrutina de interrupción guarda el código de la tecla aplastada y de su liberación (el mismo código de la tecla + 80h). Para las teclas Shift y las de conmutación, se debe además mantener el registro del estado actual.

El DOS presenta varias funciones para el teclado que son ejecutadas a través de la interrupción 21H. Entre las principales se tiene, la función 0AH que define un "buffer" para el teclado donde se pueda construir la cadena. Los 3 primeros bytes, de esta área tienen propósitos específicos; el primero indica el espacio de trabajo del buffer, el segundo byte es actualizado por el DOS para indicar el número de bytes real a la entrada. El tercer byte es el comienzo de la cadena de entrada.

Cadenas de texto

Los datos de texto se forman con caracteres individuales que ocupan un byte cada uno, de acuerdo a la tabla de caracteres ASCII. Con el fin de definir cadenas de caracteres (unidad combinada de grupos de octetos colocados uno tras otro) las cuales no tienen una longitud definida, se debe determinar su longitud y donde termina la cadena.

Para esto se utilizan dos métodos: el primero graba un número de un byte al comienzo, que define la longitud de la cadena. Con este método se limita el número máximo de caracteres en 255 (éste es el método utilizado por el editor de BASIC). Otra forma es definir el final de la cadena con un carácter delimitador el cual no se considera parte de la cadena.

Los caracteres más utilizados como delimitadores son el 0 o nulo, en el cual todos los bits son 0 y el otro que se suele utilizar es el carácter 13, que entre los caracteres de control se define como retorno de carro.

Los códigos ASCII se ordenan de la siguiente forma:

```
   00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

30         ! " # $ % & ' ( ) * + , - . / 0 1

50  2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E

70  F G H I J K L M N O P Q R S T U V W X Y
```

90 Z [\ ] ^ _ ' a b c d e f g h i j k l m

110 n o p q r s t u v w x y z { | } ~ ⌂ Ç ü

130 é â ä à å ç ê ë è ï î ì Ä Å É æ Æ ô ö ò

150 û ù ¡ Ö Ü ¢ £ ¥ ₧ ƒ á í ó ú ñ Ñ ª º ¿ ⌐

170 ¬ ½ ¼ ¡ « » ▒ ▓ ▒ | ┤ ╡ ╢ ╖ ╕ ╣ ║ ╗ ╝ ╜

190 ╛ ┐ └ ┴ ┬ ├ ─ ┼ ╞ ╟ ╚ ╔ ╩ ╦ ╠ ═ ╬ ╧ ╨ ╤

210 ╥ ╙ ╘ ╒ ╓ ╫ ╪ ┘ ┌ ▉ ▄ ▌ ▐ ▀ α ß Γ π Σ σ

230 µ τ Φ Θ Ω δ ∞ φ ε ∩ ≡ ± ≥ ≤ ⌠ ⌡ ÷ ≈ ° ∙

250 · √ ⁿ ² ■

Video

Todos los PC y PS/2 tienen un subsistema de video responsable de producir la imagen que aparece an la pantalla. El corazón del subsistema de video es el circuito de propósito específico que debe ser programado para generar las señales eléctricas que controlan la imagen de video. La mayoría de los miembros de la familia PC/XT/AT necesitan que se instale un adaptador de pantalla, una placa especial de circuitos de video que se conecta en uno de los canales de expansión del ordenador. Por el contrario todos los PS/2 están ya equipados con la circuitería y por tanto, no necesitan este adaptador de pantalla.

Casi todos los subsistemas de video pueden programarse fundamentalmente en dos modos diferentes, llamados por IBM modo de texto y modo gráfico. En modo de texto solamente se puede visualizar caracteres de texto, aunque muchos de estos caracteres se pueden utilizar para producir dibujos lineales sencillos. El modo gráfico se emplea principalmente para dibujos complejos, aunque también pueden utilizarse para representar caracteres de texto con gran variedad de formas y tamaños.

Existen dos tipos de monitores: monocromáticos y de color, los cuales se diferencian además por la resolución que pueden presentar en los caracteres que se despliegan. Entre los

monitores más comunes podemos encontrar: MDA (Monochrome Display Adapter), HGC (Hercules Graphics Card), CGA (Color Graphis Adapter), EGA (Enhanced Graphics Adapter), VGA (Video Graphics Array).

La asignación de pines en los distintos tipos de monitores es:

Pin	MDA,HGC	CGA	EGA	VGA(Analógicos)
1	Gnd	Gnd	Gnd	Red(0-0.7V)
2	Gnd	Gnd	2nd Red	Green(0-0.7V)
3	N/A	Red	Red	Blue(0-0.7V)
4	N/A	Green	Green	Mon ID(2)(IN)
5	N/A	Blue	Blue	Gnd
6	Intensity	Intensity	2nd Green	Red return
7	Video	N/A	2nd Blue	Green return
8	H.Sync(+)	H.Sync(+)	H.Sync(+)	Blue return
9	V.Sync(-)	V.Sync(-)	V.Sync(-)	N/A
10			Sync return	
11			Mon ID(0)(IN)	
12			Mon ID(1)(IN)	
13			H.Sync(OUT)	
14			V.Sync(OUT)	
15			RESERVED	

ROM-BIOS

La ROM-BIOS está dividida en 3 partes:

- En la primera están el POST y las rutinas de inicialización (crear vectores de interrupción, preparación del equipo, inicializar registros, carga de parámetros, etc.), luego están las ampliaciones al BIOS que algunos equipos opcionales necesitan (y por lo tanto se requiere inicializar), las cuales ocupan los bloques C, D y E de la memoria, cuyos 2 primeros bytes son 55AA. No tienen que estar en conflicto con otra ampliación y tienen que empezar en un múltiplo de 2K. La última parte son las rutinas de arranque.

- Las otras dos partes de la ROM-BIOS son el manejo de las interrupciones del "hardware" y la ejecución de servicios.

Para manejar las necesidades del "hardware" como pulsar una tecla, manejar el reloj interno y los discos, etc. está la sección de manejo de interrupciones del "hardware".

Interrupciones de servicio de la ROM-BIOS.- Para pedir un servicio de interrupción de la ROM-BIOS, se carga su número en uno de los registros del micro-procesador (AH) y se activa la interrupción necesaria, usando la instrucción INT.

Las interrupciones son instrucciones que detienen la ejecución del programa principal y colocan el apuntador de instrucciones en una localidad de memoria específica, en la cual se inicia la subrutina de atención a la interrupción; luego de ejecutado el programa de interrupción, se retorna al programa principal, para lo cual, la subrutina de interrupción debe contener la instrucción IRET. Los principales servicios de interrupción disponibles se listan a continuación.

Interrupciones Intel y Bios

Int.	Tipo	Descripción	
0	Intel	Overflow:	Ocurre cuando al dividir se presenta sobreflujo
1	Intel	Paso a paso:	Simula la ejecución de programas paso a paso para su depuración
2	BIOS	NMI:	Aparece si se detecta error de paridad en la memoria (parity check 1 para la tarjeta del sistema y parity check 2 para adicionales)
3	Intel	Breakpoint:	Para el procesamiento, en una dirección particular
4	Intel	Overflow int:	Interrupción con posibilidad de retorno, en caso de overflow
5	BIOS	Print Screen:	Imprime el contenido de la pantalla sin salir del programa principal.
8	BIOS	Timer:	Maneja la interrupción del temporizador 8253, que proviene del canal 0. Ocurren 18,2 interrupciones por segundo. La rutina lleva el conteo desde que se energizó el computador. Llama además a INT 1Ch, la que puede contener una rutina escrita por el usuario.
9	BIOS	Tecla presionada:	Se produce cada vez que se presiona o se libera una tecla. Para manejo del teclado se utiliza INT 16h.
E	BIOS	Diskette:	Utilizada para el manejo de los diskettes.
F	DOS	INTO:	Igual que INT 4h.
10	BIOS	Video:	Permite el manejo del display. Tiene varios servicios.
11	BIOS	Equipo disponible:	Proporciona número de puertos para impresora, adaptadores de juego, puertos seriales, unidades de diskette, modo de video y

		tamaño de la RAM.

Int.	Tipo	Descripción
12	BIOS	Tamaño Memoria: Proporciona el tamaño de la memoria.
13	BIOS	I/O en Diskette:Tiene varios servicios para I/O en disco.
14	BIOS	Puerto serie: Permite la entrada/salida de datos al puerto serial.
15	BIOS	I/O Cassette: Controla operaciones de entrada/salida en cassette.
16	BIOS	I/O de teclado: Permite leer datos desde el teclado. Utiliza AX.
17	BIOS	I/O de impresora: Proporciona comunicación con la impresora. Utiliza AX y DX.
18	BIOS	ROM-BASIC: Llama a la memoria en la que se almacena el intérprete BASIC.
19	BIOS	Arranque: Lee el sector 1 de la pista 0 del disco en la unidad A, donde transfiere el control.
1A	BIOS	Reloj: Permite seleccionar o leer el contenido del reloj. CX contiene los MSB del conteo y DX los LSB.
1B	DOS	Ctrl-Break: Interrupción que se produce cuando en el teclado se presiona Ctrl-Break o Ctrl-C.
1C	BIOS	Retorno ficticio: Provoca la ejecución de una instrucción IRET.
1D	BIOS	Parámetros de video: Es una tabla de bytes y rutinas necesarias para establecer varios parámetros para gráficos.
1E	DOS	Tabla del diskette
1F	DOS	Tabla de gráficos

Impresión de pantalla (INT 5h)

Copia a la impresora el contenido de la pantalla, mientras el programa principal mantiene el control de los datos que se despliegan.

Puede ser solicitado desde cualquier programa realizando una instrucción de interrupción INT 5.

Se inicia con el servicio 3 de la INT 10h para saber la posición del cursor y el 15 de la INT 10h para saber las dimensiones de la pantalla. Almacena la posición del cursor y lo mueve de arriba abajo solicitando el servicio 8 y un servicio para impresión para cada posición, luego restaura el cursor a su posición original y devuelve el control al programa que se está corriendo.

Temporizador (INT 8h)

Regresa un conteo del tiempo en las localidades 0040:006C y 0040:006E; la primera parte contiene los bits menos significativos y la segunda los más significativos del contador de tiempo de 32 bits. La señal del canal 0 del temporizador oscila generando una interrupción 18,2 veces por segundo y la rutina de tratamiento suma uno, cada vez, al contador del reloj.

Tecla presionada (INT 9h)

Detiene la ejecución del programa principal, cada vez que se presiona o se libera una tecla, almacenando la información que proviene del teclado en el buffer respectivo.

Servicios de Video (INT 10h)

Servicio	Descripción	
0	Modo	AL contiene el modo de video. Si AL=0: modo texto 40* 25 B/N. 1: modo texto 40* 25 Color. 2: modo texto 80* 25 B/N. 3: modo texto 80* 25 Color. 4: modo gráf. 320*200 Color. 5: modo gráf. 320*200 B/N. 6: modo gráf. 640*200 B/N.
1	Tipo de cursor	Bits 4-0 de CH indican la línea donde comienza el cursor. CL termina . Los restantes bits deben ponerse en 0.
2	Selección de posición cursor	(DH,DL)=(renglón,columna) donde se colocará el cursor. La esquina superior izquierda corresponde a (0,0). En BH número de página(0 para gráf)
3	Lectura de posición cursor	(DH,DL)=(renglón,columna) donde se encuentra el cursor. (CH,CL)=dimensiones del cursor.
4	Lectura posición lápiz óptico	Informa si el lápiz óptico está accionado y en que lugar está tocando la pantalla, en términos de la cuadrícula o posición del pixel gráfico
5	Página desplegada	Permite seleccionar una de las páginas de video para ser desplegada

	activa	AL=0-7 para 40*25. AL=0-3 para 80*25 (página 0,1,2 ó 3).

Servicio	Descripción	
6 7	Scroll de ventana	Define una ventana rectangular y mueve los datos de arriba abajo (6) o de abajo arriba (7). AL=número de líneas (si AL=0 ventana en blanco) (CH,CL)=coordenadas esquina superior izquierda (fila, columna). (DH,DL)= inferior derecha . BH= atributo a utilizar para las líneas en blanco.
8	Lectura de carácter	Lee el carácter en el cursor y su atributo. BH=página en exhibición. AL=código del carácter. AH=atributo.
9	Escritura de carácter con atributo	Escribe el carácter en el cursor con el atributo especificado. BH=página activa. CX=conteo de caracteres. AL=carácter a escribir. AH=atributo del carácter.
10	Escritura de carácter	Escribe un carácter con el atributo pre-especificado para esa posición Se usan los mismos registros que para el servicio anterior, menos AH.
11	Selección paleta	Selecciona la paleta de colores que se va a utilizar.
12	Escribe un punto	DX=número de la fila. CX=número de la columna. AL=color.
13	Lee un punto	DX=número de la fila. CX=número de la columna. AL=punto leído.
14	Escritura corrida	Escribe un carácter y avanza el cursor a la siguiente posición AL=carácter. BL=color fondo (modo gráfico). BH=página activa.
15	Modo actual	AL=modo de video activo.AH=número columnas en pantalla.BH=página activa

Listado de equipo (INT 11h)

Proporciona un resumen de las opciones disponibles en la unidad del sistema, nos dice cuantas unidades de disco tiene (de 0 a 4), el número de puertos paralelos y serie, si hay adaptador de juegos, pero no dice nada de los discos duros ni de los lápices ópticos. El resumen se proporciona en el registro AX.

Tamaño de memoria (INT 12h)

Proporciona, en el registro AX, la cantidad de memoria RAM instalada en el sistema en KB (bloques de 1024 bytes).

I/O en diskette (INT 13h)

Los servicios de disco son básicamente seis:

0 Inicializa la unidad de disco y su controlador.
1 estado de la unidad de disco.
2 lee sectores del disco en la memoria siempre que estén todos ellos en la misma pista.
3 escribe sectores (inverso al 2).
4 verifica los datos escritos en un disco (corresponde a VERIFY ON) comprobando errores de paridad, etc.
5 formatea una pista de un disco.

Los servicios del pórtico serial (INT 14h) son 4:

0 inicializar el puerto fijando los parámetros básicos.
1 enviar un byte al puerto.
2 leer un byte
3 información del estado, que indica cosas tales como si el dato está preparado.

I/O del teclado (INT 16h)

Cuando se presiona o libera un carácter desde el teclado, se almacena un byte de información en el "buffer" de entrada de teclado. Dicha información contiene un 1 en el bit 7 para tecla presionada y un 0 para tecla liberada y los restantes 7 bits contiene el código de barrido de la tecla presionada (existen 83 códigos de barrido, uno para cada tecla). Debido a que en ocasiones es necesario presionar dos teclas al mismo tiempo (antes de que cualquiera de ellas sea liberada), es necesario almacenar el byte correspondiente a cada tecla. El código de barrido es traducido a un carácter ASCII extendido una vez que se lee el contenido del "buffer" del teclado. El "buffer" puede contener hasta 20 bytes de información.

0 lee el siguiente carácter del "buffer" de entrada del teclado, los caracteres se presentan en su formato de 2 bytes y se almacenan en AX; AL regresa el código ASCII de la tecla presionada y AH el código de barrido.

1 la bandera ZF informa si hay preparada alguna entrada del teclado (ZF=1 indica buffer vacío), si existen entradas en el "buffer", informa sobre el primer carácter comunicando sus octetos en AX, pero el carácter se queda en el "buffer" hasta ser leído con el servicio 0.

2 comunica los bits de estado del teclado en AL.

Puerto paralelo (INT 17h)

0 envía el byte contenido en AL a la impresora.

1 inicializa la impresora.

2 comunica, en AH, el estado de la impresora.

El número del puerto paralelo con el que se está operando debe ser colocado en DX, siendo el puerto predefinido el 0.

ROM-BASIC (INT 18h) y Arranque (INT 19h)

Estas dos interrupciones permiten pasar el control a una de las dos rutinas especiales internas del BIOS, la ROM-BASIC y las rutinas de arranque, que provocan que la computadora reinicialice, desde disco, el sistema operativo. Es posible activar estas rutinas simplemente con solicitar estas interrupciones.

Archivos

En general, el usuario puede nombrar un archivo con cualquier combinación de letras y números. Para el nombre del archivo se dispone de 8 caracteres y para la extensión de 3.

Existen programas que utilizan la extensión para reconocer a sus archivos, entre las más conocidos se puede mencionar las siguientes extensiones:

.DBF (Data Base File) archivos de bases de datos;

.WKS (Work Sheet) y .WK1 archivos de hoja electrónica,
.PRN (Printer) archivos de impresión y
.PIC (Graphic) archivos de gráficos, en el programa LOTUS;
.WQ1 archivos de hoja electrónica QPRO;
.DWG (Drawing) archivos de gráficos de AUTOCAD;
.BAK y BK! (Backup) archivos de respaldo de varios procesadores de palabras;

.SYS (system) para archivos que en esencia son programas .COM pero que deben ser cargados a partir del archivo CONFIG.SYS;
.WPG (Word Perfect Graphic) gráficos de word perfect; etc.

Además de todos los servicios de la ROM-BIOS, ya que el DOS duplica dichos servicios para evitar dependencias de los programas respecto del tipo de máquina utilizada, mediante el DOS se pueden acceder a servicios de disco de alto nivel, entre los cuales se diferencia:

Los "servicios de archivo tradicionales" que están basados en el uso del Bloque de control de archivo (FCB o "file control block") utilizado para proporcionar el nombre y la identificación de los archivos con los que se trabajará. Entre otras cosas, estos servicios pueden localizar archivos (con * y ?), abrir un archivo para lectura o escritura, cerrarlo, realizar lectura o escritura secuencial de principio a fin y lectura o escritura aleatoria.

El programador debe mantener un área de datos en la memoria, conteniendo la información del archivo. El FCB contiene información particular que describe al archivo. El sistema usa dicha información para el acceso de I/O al archivo. El PSP, "Program Segment Prefix", tiene espacio para dos FCBs, uno con offset 5Ch y el otro con 6Ch. Los FCBs y FCBs extendidos pueden estar abiertos o no abiertos. Un FCB no abierto contiene especificador de drive y nombre de un archivo, en el nombre se pueden incluir caracteres comodín (el * y ?, conocidos como "wildcard"). Cuando la llamada al sistema que permite abrir un archivo (función 0Fh) llena todos los campos del FCB, se dice que el FCB se ha abierto.

Todo acceso usando el FCB se realiza a través de un buffer de datos conocido como el DTA, "disk transfer address". Cuando se da el control a un programa de usuario, el DTA se llena con offset 80h en el PSP. Este DTA tiene suficiente espacio para transferir 128 bytes. El programa puede mover el DTA a cualquier sitio, usando la función 1Ah.

Campos del FCB:

Nombre	Tamaño en bytes	Offset	Descripción
Drive number	1	00h	1 especifica drive A, 2 drive B. Cuando el FCB se usa para crear o abrir un archivo, 0 especifica el drive por omisión ("default"). Esto significa que la función 0Fh (llamada al sistema

para abrir un archivo) carga este campo con el número del drive por omisión.

Filename	8	01-08h	cadena de caracteres (ocho caracteres), llenada con espacios en blanco, si es necesario, y justificada a izquierda. En este campo se pueden poner nombres como LPT1 o PRN para proveer entrada/ salida con dispositivos.
Extension	3	09-0Bh	cadena de 3 caracteres justificada a izquierda, llenada con espacios en blanco si es necesario. Si todo el campo está lleno de espacios en blanco, no existe extensión.
Current Block	2	0Ch-0Dh	apunta al bloque (128 records) que contiene el registro corriente ("current record"). El llamado a la función del sistema para apertura de archivo (0Fh) pone este campo a cero. Los campos de "Current Block" y "Current Record", conjuntamente, definen el "record pointer".
Record Size	2	0Eh-0Fh	Este campo contiene el tamaño en bytes de un registro lógico. La función 0Fh pone este campo en 128, se debe llenar este campo antes de que el archivo sea abierto.
File size	4	10h-13h	Contiene el tamaño en bytes del archivo. La primera palabra contiene la parte menos significativa del tamaño.
Fecha	2	14h-15h	Fecha de la última escritura contiene la fecha del cambio más reciente. Se mapean como se ilustra:

OFFSET 15h

Y Y Y Y Y Y Y M

15 9 8

OFFSET 14h

M M M D D D D D

5 4 0

El año se representa como el numero del año a partir de 1980.

Hora	2	16h-17h	Hora de la última escritura contiene la hora del cambio más reciente. Los segundos aparecen en incrementos de dos segundos, se mapea como se ilustra:

OFFSET 17h

H H H H H M M M

15 11 10

M M M S S S S S

5 4 0

Reserved	8	18h-1Fh	El DOS se reserva el uso de estos campos.
Current record	1	20h	Apunta a uno de los 128 registros del bloque corriente. Se debe llenar el campo de "current record" antes de hacer la lectura o escritura secuencial al archivo, ya que la llamada al sistema para abrir archivo no inicializa este campo. Los campos de "current record" y "current block" conjuntamente definen el "record pointer".
Relative record	4	21h-24h	Cuenta desde el inicio del archivo, comenzando desde cero, y apunta al registro seleccionado actualmente. Se debe llenar el campo "relative record" antes de hacer una lectura o escritura secuencial al archivo, ya que la llamada al sistema para apertura de archivo no inicializa este campo. Un tamaño de registro menor que 64 bytes usa las dos palabras de este campo, un tamaño de más de 64 bytes usa solo los primeros 3 bytes.

En el FCB, con offset de 5Ch desde el PSP (Program Segment Prefix), el último byte del "relative record field" está en el primer byte del "unformatted parameter area", que comienza en "offset" 80h. Este es el DTA por omisión.

FCB extendido:

Añadiendo un prefijo de 7 bytes al FCB standard se produce un FCB extendido. El FCB extendido puede crear o buscar el directorio por archivos con atributos especiales. La lista de los 7 bytes del prefijo se adjunta:

Nombre	Tamaño	Valor	Offset (decimal)
Flag Byte	1	FFh	-7
Reserved	5	-	-6
Attribute Byte	1	-	-1
read only		01h	
hidden file		02h	
system file		04h	
volume ID		08h	
directory		10h	
archive		20h	

Para operar archivos existe una alternativa al uso del FCB; es el uso del "handle" que es un número de 2 bytes que identifica en forma unívoca a cada archivo que se está utilizando por un programa, con lo cual la información de control del archivo se mantiene a salvo.

Un "handle" es un valor binario de 16 bits usado por el DOS para tener acceso a un archivo o dispositivo. El nombre del archivo o dispositivo es especificado (como cadena de caracteres ASCII terminada por un byte de ceros y que puede contener "drive", "path" y nombre) al DOS durante la apertura, y el DOS regresa el "handle", que se utilizará para los siguientes accesos.

Este método no requiere un DTA. El usuario especifica la ubicación de los datos que serán transferidos de o hacia el archivo, para los requerimientos de funciones que utilizan el "handle".

DOS pre-especifica 5 "handles" que están abiertos cuando se da el control a un programa de usuario, que son:

Nombre	Handle	Dispositivo por omisión
Standard input	0000h	CON
Standard output	0001h	CON
Standard error	0002h	CON
Standard auxiliary	0003h	AUX

Standard printer 0004h PRN

Los dispositivos PRN (LST y LPT1), COM1 (AUX), COM2, LPT2, y LPT3 deben ser pre-definidos para procesar una cadena de control de 2 bytes llamada "device configuration word" (DCW). Esta palabra es pasada por, o leída de, un "handler" de dispositivo, usando la petición de función 44h.

La DCW tiene el siguiente formato:

1 2 0 3 3 0 4 4 0 5 5 5 6 0 7 7

código definición

0 Reservada.

1 tipo de puerto (0=paralelo, 1=serial).

2 Número de bits por carácter (0=7, 1=8).

3 Número de puerto (00=1, 01=2, 10=3, 11=4).

4 Tipo de verificación de ocupado (00=ninguna, 01=SCF, 10=DSR, 11=Xon/Xoff).

5 "Baud rate" (000=110, 001=150, 010=300, 011=600, 100=1200, 101=2400, 110=4800, 111=9600).

6 Número de bits de parada (0=1, 1=2).

7 Tipo de chequeo de paridad (00=ninguna, 01=par, 10=ninguna, 11=impar).

INSTRUCCIONES DEL 8086

Instrucciones para movimiento de datos

Instrucción	Operandos posibles	Banderas O	D	I	T	S	Z	A	P	C	Operación
MOV	mem/reg1, mem/reg2										[mem/reg1] ← [mem/reg2] (No permite [mem] — [mem])
	mem/reg, dato										[mem/reg] — dato
	reg, dato										[reg] — dato
	ac, mem										[ac] ← [mem]
	mem, ac										[mem] ← [ac]
	seg_reg, mem/reg										[seg_reg] ← [mem/reg]
	mem/reg, seg_reg										[mem/reg] ← [seg_reg]
XCHG	mem/reg1, mem/reg2										[mem/reg1] ⟷ [mem/reg2] (No permite [mem] ⟷ [mem])
	reg										[AX] ⟷ [reg]
XLAT											[AL] ← [[AL] + [BX]]
LDS	reg, mem										[reg] ← [mem], [DS] ← [mem + 2]
LEA	reg, mem										[reg] ← mem (offset de la dirección)
LES	reg, mem										[reg] ← [mem], [ES] ← [mem + 2]
PUSH	mem, reg										[SP] ← [SP] - 2, [[SP]] ← [mem/reg]
	reg										[SP] ← [SP] - 2, [[SP]] ← [reg]
	seg_reg										[SP] ← [SP] - 2, [[SP]] ← [seg_reg]
PUSHF											[SP] ← [SP] - 2, [[SP]] ← [FLAGS]
POP	mem, reg										[mem/reg] ← [[SP]], [SP] ← [SP] + 2
	reg										[reg] ← [[SP]], [SP] ← [SP] + 2
	seg_reg										[seg_reg] ← [[SP]], [SP] ← [SP] + 2
POPF		x	x	x	x	x	x	x	x	x	[FLAGS] ← [[SP]], [SP] ← [SP] + 2
LAHF											[AH] ← Con las banderas SZxAxPxC (x = indeterminado)
SAHF						x	x	x	x	x	Las banderas SZxAxPxC ← [AH]

Instrucciones aritméticas

Instrucción	Operandos posibles	Banderas O D I T S Z A P C	Operación
ADC	mem/reg1, mem/reg2 mem/reg, dato ac, dato	x x x x x x x x x	[mem/reg1] ← [mem/reg1] + [mem/reg2] + [C] [mem,reg] ← [mem/reg] + dato + [C] [ac] ← [ac] + dato + [C]
ADD	mem/reg1, mem/reg2 mem/reg, dato ac, dato	x x x x x x x x x	[mem/reg1] ← [mem/reg1] + [mem/reg2] [mem,reg] ← [mem/reg] + dato [ac] ← [ac] + dato
INC	mem/reg reg	x x x x x x x	[mem/reg] ← [mem/reg] + 1 [reg] ← [reg] + 1
AAA		? ? ? x ? x	Ajuste ASCII de AL para la adición
DAA		? x x x x x	Ajuste decimal de AL para la adición
SUB	mem/reg1, mem/reg2 mem/reg, dato ac, dato	x x x x x x x x x	[mem/reg1] ← [mem/reg1] - [mem/reg2] [mem,reg] ← [mem/reg] - dato [ac] ← [ac] - dato
SBB	mem/reg1, mem/reg2 mem/reg, dato ac, dato	x x x x x x x x x	[mem/reg1] ← [mem/reg1] - [mem/reg2] - [C] [mem,reg] ← [mem/reg] - dato - [C] [ac] ← [ac] - dato - [C]
DEC	mem/reg reg	x x x x x x x	[mem/reg] ← [mem/reg] - 1 [reg] ← [reg] - 1
AAS		? ? ? x ? x	Ajuste ASCII de AL para la substracción
DAS		? x x x x x	Ajuste decimal de AL para la substracción
NEG	mem/reg	x x x x x x x	[reg] ← -[reg] + 1
MUL	mem/reg (8 bits) mem/reg (16 bits)	x ? ? ? ? ? x	[AX] ← [AL] * [mem/reg] (sin signo) [DX][AX] ← [AX] * [mem/reg] (sin signo)
IMUL	mem/reg (8 bits) mem/reg (16 bits)	x ? ? ? ? ? x	[AX] ← [AL] * [mem/reg] (con signo) [DX][AX] ← [AX] * [mem/reg] (con signo)
AAM		? x x ? x ?	Ajuste ASCII para la multiplicación. AH y AL usados
DIV	mem/reg (8 bits) mem/reg (16 bits)	? ? ? ? ? ? ?	([AH]←residuo, [AL]←cociente) de [AX]/[mem,reg] ([DX]←residuo, [AX]←cociente) de [DX][AX]/[mem,reg] (sin signo)
IDIV	mem/reg (8 bits) mem/reg (16 bits)	? ? ? ? ? ? ?	([AH]←residuo, [AL]←cociente) de [AX]/[mem,reg] ([DX]←residuo, [AX]←cociente) de [DX][AX]/[mem,reg] (con signo)
CBW			[AH] ← [AL7]
CWD			[DX] ← [AX15]
AAD		? x x ? x ?	Ajuste ASCII para la división

Instrucciones de comparación y lógicas

Instrucción	Operandos posibles	Banderas O	D	I	T	S	Z	A	P	C	Operación
CMP	mem/reg1, mem/reg2 mem/reg, dato ac, dato	x				x	x	x	x	x	[mem/reg1] - [mem/reg2] [mem/reg] - dato [ac] - dato
AND	mem/reg1, mem/reg2 mem/reg, dato ac, dato	x				x	x	?	x	x	[mem/reg1] ← [mem/reg1] AND [mem/reg2] [mem/reg] ← [mem/reg] AND dato [ac] ← [ac] AND dato
NOT	mem/reg										[mem/reg] ← -[mem/reg]
OR	mem/reg1, mem/reg2 mem/reg, dato ac, dato	x				x	x	?	x	x	[mem/reg1] ← [mem/reg1] OR [mem/reg2] [mem/reg] ← [mem/reg] OR dato [ac] ← [ac] OR dato
TEST	mem/reg1, mem/reg2 mem/reg, dato ac, dato	x				x	x	?	x	x	[mem/reg1] AND [mem/reg2] [mem/reg] AND dato [ac] AND dato
XOR	mem/reg1, mem/reg2 mem/reg, dato ac, dato	x				x	x	?	x	x	[mem/reg1] ← [mem/reg1] XOR [mem/reg2] [mem/reg] ← [mem/reg] XOR dato [ac] ← [ac] XOR dato

Instrucciones para manejo de cadenas

Instrucción	Operandos posibles	Banderas O	D	I	T	S	Z	A	P	C	Operación
LODS											[ac] ← [[SI]], [SI] ← [SI] ± δ (± depende de DF) (δ = 1 para 8 bits o δ = 2 para 16 bits)
MOVS											[[DI]] ← [[SI]], [SI] ← [SI] ± δ (misma observación)
STOS											[[DI]] ← [ac], [DI] ← [DI] ± δ (misma observación)
CMPS		x				x	x	x	x	x	[[SI]] - [[DI]], [SI] ← [SI] ± δ, [DI] ← [DI] ± δ(")
SCAS		x				x	x	x	x	x	[ac] - [[DI]]
REP	prefijo a instruc										usa CX como contador, repite la instrucción hasta que CX=0
REPE REPZ											sale del lazo si CX=0 o si se pone la bandera ZF luego de cualquier ejecución de la instrucción
REPNE REPNZ											sale del lazo si CX=0 o si se quita la bandera ZF luego de cualquier ejecución de la instrucción

Instrucciones que controlan el contador de programa

Instrucción	Operandos posibles	Banderas O D I T S Z A P C	Operación
CALL	addr		[SP]←[SP]-2,[[SP]]←[PC], [SP]←[SP]-2,[[SP]]←[CS], [PC]←offset de dirección, [CS]←segmento de dirección
	disp16		[SP]←[SP]-2, [[SP]]←[PC], [PC]←[PC]+disp16
	mem(SEG+PC)		[SP]←[SP]-2,[[SP]]←[PC], [SP]←[SP]-2,[[SP]]←[CS], [PC]←[mem], [CS]←[mem + 2]
	mem/reg		[SP]←[SP]-2, [[SP]]←[PC], [PC]←[mem/reg]
RET			[PC]←[[SP]], [SP]←[SP]+2
			[PC]←[[SP]], [SP]←[SP]+2, [CS]←[[SP]], [SP]←[SP]+2
	disp16		
	disp16		[PC]←[[SP]], [SP]←[SP]+2+disp16
			[PC]←[[SP]], [SP]←[SP]+2, [CS]←[[SP]], [SP]←[SP]+2+disp16
JMP	addr		[PC]←offset de dirección, [CS]←segmento de dirección
	disp		[PC] ← [PC] + disp
	disp16		[PC] ← [PC] + disp16
	mem(SEG+PC)		[PC] ← [mem], [CS] ← [mem + 2]
	mem/reg		[PC] ← [mem/reg]

Instrucciones de salto condicional

Instrucción	Operandos posibles	Banderas O D I T S Z A P C	Operación
JNBE(JA)	disp		Si ([C] OR [Z])=0, entonces [PC] ← [PC] + disp
JNB JNC JAE	disp		Si ([C] = 0, entonces [PC] ← [PC] + disp
JB JC JNAE	disp		Si ([C] = 1, entonces [PC] ← [PC] + disp
JBE JNA	disp		Si ([C] OR [Z])=1, entonces [PC] ← [PC] + disp
JE JZ			Si [Z]) = 1, entonces [PC] ← [PC] + disp
JG JNLE			Si([Z]=0 AND ([S]=[O]))=1, entonces [PC]←[PC]+disp
JGE JNL			Si [S] = [O], entonces [PC] ← [PC] + disp
JL JNGE			Si [S] <> [O], entonces [PC] ← [PC] + disp
JLE JNG			Si ([S]=[O] AND [Z]=0)=1,entonces [PC] ← [PC] + disp
JNE JNZ			Si [Z]=0, entonces [PC] ← [PC] + disp
JNO			Si [O]=0, entonces [PC] ← [PC] + disp
JNP JPO			Si [P]=0, entonces [PC] ← [PC] + disp
JNS			Si [S]=0, entonces [PC] ← [PC] + disp
JO			Si [O]=1, entonces [PC] ← [PC] + disp
JP JPE			Si [P]=1, entonces [PC] ← [PC] + disp
JS			Si [S]=1, entonces [PC] ← [PC] + disp
JCXZ			Si [CX]=0, entonces [PC] ← [PC] + disp

Instrucciones de lazo

Instrucción	Operandos posibles	Banderas O D I T S Z A P C	Operación
LOOP	disp		[CX]←[CX]-1, Si [CX]=0, entonces [PC] ← [PC] + disp
LOOPZ LOOPE	disp		[CX]←[CX]-1,Si[CX]<>0 y [Z]=1,entonces[PC]←[PC]+disp
LOOPNZ LOOPNE	disp		[CX]←[CX]-1,Si[CX]<>0 y [Z]=0,entonces[PC]←[PC]+disp
JCXZ	disp		Si [CX]=0, entonces [PC] ← [PC] + disp

Instrucciones de control al procesador

Instrucción	Operandos posibles	Banderas O D I T S Z A P C	Operación
CLC		0	[C] ← 0
CMC		x	[C] ← -[C]
CLD		0	[D] ← 0
CLI		0	[I] ← 0
STC		1	[C] ← 1
STD		1	[D] ← 1
STI		1	[I] ← 1
NOP			Ninguna operación
ESC	mem		bus de direcciones/datos ← [mem]
LOCK			cerrar el bus
WAIT			espera por señal externa para reiniciar operación
HLT			halt

Instrucciones de entrada/salida

Instrucción	Operandos posibles	Banderas O D I T S Z A P C	Operación
IN	ac,DX		[ac] ← [puerto (DX)]
	ac,puerto		[ac] ← [puerto]
OUT	ac,DX		[puerto (DX)] ← [ac]
	ac,puerto		[puerto] ← [ac]

Instrucciones de interrupción

Instrucción	Operandos posibles	Banderas O D I T S Z A P C	Operación
INT		0 0	[SP]←[SP]-2, [[SP]]←[FLAGS], [I]←0, [T]←0, [SP]←[SP]-2, [[SP]]←[CS], [SP]←[SP]-2,[[SP]]←[PC] [CS]←[vector(segmento)], [PC]←[vector(offset)]
INTO		0 0	Si [O]=1, [SP]←[SP]-2, [[SP]]←[FLAGS], [I]←0, [T]←0, [SP]←[SP]-2, [[SP]]←[CS], [SP]←[SP]-2,[[SP]]←[PC] [CS]←[00012h], [PC]←[00010h]
IRET		x x x x x x x x x	[PC]←[[SP]], [SP]←[SP]+2, [CS]←[[SP]], [SP]←[SP]+2, [FLAGS]←[[SP]], [SP]←[SP]+2

Instrucciones de rotación

Instrucción	Operandos posibles	Banderas O D I T S Z A P C	Operación
RCL	mem/reg, cuenta	x x	Rota mem/reg a la izquierda pasando por CF. cuenta puede ser 1 o el contenido de CL.
RCR	mem/reg, cuenta	x x	Rota mem/reg a la derecha pasando por CF. cuenta puede ser 1 o el contenido de CL.
ROL	mem/reg, cuenta	x x	Rota mem/reg a la izquierda, el bit más alto se copia en CF. cuenta puede ser 1 o contenido de CL
ROR	mem/reg, cuenta	x x	Rota mem/reg a la derecha, el bit más bajo se copia en CF. cuenta puede ser 1 o el contenido de CL
SAL SHL	mem/reg, cuenta	x x x x ? x x	Rota mem/reg a izquierda pasando por CF. Se pone 0 en el bit menos significativo. cuenta =1 o [CL]
SAR	mem/reg, cuenta	x x x x ? x x	Manteniendo el bit de signo, rota a derecha alimentando CF con el bit más bajo. cuenta =1 o [CL]
SHR	mem/reg, cuenta	x x x x ? x x	Rota mem/reg a derecha, pone 0 al bit más significativo, el bit más bajo se copia en CF.cuenta=1 o [CL]

RUTINAS DE SERVICIO DEL DOS

Servicio **Descripción**

00h Terminar programa. Llama a la INT 20h; AH=00, CS=dirección del segmento del PSP. No retorna información.

01h Leer entrada estándar con eco. Espera hasta que se coloque un carácter desde el dispositivo de entrada estándar, el cual es colocado en la salida estándar. Si el carácter es Ctrl-Break, se ejecuta la INT 23h; AH=01. Retorna en AL el carácter ingresado.

02h Mostrar carácter. Envía un carácter al dispositivo estándar de salida. Si se recibe un Ctrl-Break desde el dispositivo de entrada estándar, se ejecuta INT 23h; AH=02, DL=carácter a ser desplegado. No retorna información.

03h Entrada auxiliar. Espera hasta que se coloque un carácter desde el dispositivo auxiliar estándar; AH=03. Retorna en AL el carácter ingresado.

04h Salida auxiliar. Envía un carácter al dispositivo auxiliar de salida. AH=04, DL=carácter a ser desplegado. No retorna información.

05h Imprimir carácter. Envía un carácter a la impresora estándar. Si se recibe un Ctrl-Break desde el dispositivo de entrada estándar, se ejecuta INT 23h; AH=05, DL=carácter a ser impreso. No retorna información.

06h Entrada/salida directa a consola. Permite realizar operaciones de entrada/salida a los dispositivos estándar. El valor colocado en DL determina el tipo de operación a realizarse. AH=06, Si DL=0FFh, se lee un carácter del dispositivo estándar de entrada (no se revisa si se ha presionado Ctrl-Break), si DL contiene otro valor, el contenido de DL es tratado como un carácter que se coloca en el dispositivo estándar de salida. Información que se retorna: Si la bandera de carry está colocada, no existe un carácter disponible, si la bandera de carry no está colocada, existe un carácter de retorno desde el dispositivo de entrada estándar, el cual se encuentra disponible en AL.

07h Entrada directa por consola. Espera por un carácter disponible desde el dispositivo estándar de entrada. No se coloca el carácter en la salida estándar ni se verifica la

presencia de Ctrl-Break. AH=07h. En AL se retorna el carácter que ingresó desde el dispositivo de entrada.

08h Leer entrada estándar. Espera por un carácter disponible desde el dispositivo estándar de entrada. No se coloca el carácter en la salida estándar pero sí se verifica la presencia de Ctrl-Break. AH=08h. En AL se retorna el carácter que ingresó desde el dispositivo de entrada.

09h Mostrar cadena. Envía a la salida estándar una cadena de caracteres terminada con el signo de dólar ($); éste signo no aparece en la salida. AH=09h, DS:DX=dirección en la que se ha almacenado la cadena. No se retorna información.

0Ah Entrada de teclado con "buffer". Espera por un carácter disponible en el dispositivo de entrada, lo coloca en el dispositivo de salida y en el buffer, a partir del tercer byte, se repite el procedimiento hasta recibir la tecla Enter. Si se llena el buffer al máximo de caracteres menos uno, se ignoran los caracteres adicionales y se envía el ASCII 7 (para que suene el parlante) al dispositivo de salida, hasta recibir la tecla Enter. Se verifica la presencia de Ctrl-Break, caso en el cual se realiza una llamada a INT 23h. AH=0Ah, DS:DX=dirección del buffer de entrada. En el primer byte del buffer de entrada se debe definir el máximo número de caracteres, incluyendo el Enter que se espera al final de la cadena; y en el segundo byte se actualiza automáticamente el número de caracteres escritos, sin incluir el Enter. No se retorna información.

0Bh Verificar estado de la entrada estándar. Verifica si existen caracteres disponibles en la entrada estándar. En caso afirmativo, AL retorna FFh, caso contrario, AL retorna 0. Si se recibe un Ctrl-Break desde el dispositivo de entrada estándar, se ejecuta INT 23h. AH=0Bh. Se retorna información en AL.

0Ch Vaciar "buffer" y leer entrada estándar. Vacía el buffer de entrada estándar. Si AL tiene 01, 07, 08 o 0Ah, se ejecuta el correspondiente servicio de interrupción, Si AL contiene otro valor, se vacía el buffer y retorna cero en AL. Si se recibe un Ctrl-Break desde el dispositivo de entrada estándar, se ejecuta INT 23h. AH=0Ch; AL=servicio requerido. Se retorna 0 en AL si no se requirió uno de los servicios permitidos.

0Dh Reset de disco. Asegura que los buffers internos se refieran a los discos ubicados en los drives. Escribe todos los buffers que han sido modificados, y los marca como

vacíos. Esta función no actualiza los directorios, por tanto se debe cerrar todos los archivos cuya extensión ha variado, antes de llamar a esta función. AH=0Dh. No se retorna información.

0Eh Selección de disco. Selecciona el drive por defecto. AH=0Eh, DL=número del drive (0=A), si DL es mayor que el número de drives existentes, el último es seleccionado. En AL se retorna el número de drives lógicos existentes.

0Fh Buscar el archivo requerido y abrir el respectivo FCB.

10h Actualizar la información del directorio para un archivo que ha sido modificado y cerrarlo.

11h Buscar en el directorio el primer archivo que cumpla con la especificación de nombre requerida (pueden usarse los caracteres * y ?).

12h Buscar en el directorio el siguiente archivo que cumpla con la especificación de nombre requerida (se usa luego de haber llamado al servicio 11h.

13h Busca en el directorio todos los archivos que cumplan con la especificación de nombre y los borra.

14h Lectura secuencial de un archivo. Se lee el registro definido por el DTA y se incrementa los valores de bloque y registro definidos.

15h Escritura secuencial de un archivo. Similiar al anterior pero para operaciones de escritura.

16h Busca en el directorio por un área vacía y crea un archivo. En caso de existir en el directorio un archivo con el mismo nombre, se borra dicho archivo y se crea un archivo vacío con el mismo nombre.

17h Busca en el directorio un archivo que cumpla con la especificación de nombre requerida y le cambia de nombre por aquel contenido en la dirección con offset 11h en el FCB.

19h Retorna en AL el número correspondiente al disco actualmente seleccionado.

1Ah Define como Disk Transfer Address a la direccionada por DS:DX.

1Bh Retorna un apuntador al byte de identificación del FAT, que permite determinar el tipo de disco.

1Ch Obtener descriptor de tipo de disco para un drive específico. Similar al anterior, pero se debe definir en DL el drive al que nos estamos refiriendo.

21h Lectura aleatoria de un archivo.

22h Escritura aleatoria de un archivo.

23h Busca en el directorio un archivo que cumpla con la especificación de nombre determinada y retorna el tamaño del archivo.

24h Se define, en el FCB, el registro relativo del archivo direccionado.

25h DS:DX contienen una dirección de 4 bytes que se almacenará en la tabla de vectores de interrupción para la interrupción especificada en AL.

26h Crea el segmento de programa definido en DX. Se prefiere utilizar el servicio 4Bh.

27h Transfiere al DTA de un FCB abierto un número de registros (definido en CX) de un archivo, luego de hber calculado la cantidad de datos a ser leídos.

28h Escritura de bloque aleatoria. Similar a la anterior pero para escritura.

29h DS:SI apuntan a una cadena de caracteres en la cual está el nombre de un archivo. Si el nombre del archivo es encontrado, se crea un FCB en la dirección apuntada por ES:DI.

2Ah Retorna la fecha actual del sistema operativo como números binarios, en CX retorna el año, en DH el mes, en DL el día y en AL el día de la semana (0=Sunday).

2Bh CX y DX contienen una fecha (igual que para el servicio 2Ah) que se carga como actual para el sistema operativo.

2Ch Retorna la hora actual, en CH la hora, en CL los minutos, en DH los segundos y en DL las centésimas.

2Dh Definir la hora contenida en CX y DX (igual que para el servicio 2Ch) para el sistema operativo.

2Eh Cambiar la bandera de verificación después de la escritura. Si AL=1 se realiza la verificación, si AL=0 se escribe sin verificar.

2Fh Leer Disk Transfer Address en ES:BX.

30h Leer la versión del DOS en AX.

31h AL contiene un código de salida del programa, el cual es enviado cuando se termina el proceso, pero no se borra de la memoria, con el fin de mantener un proceso residente en memoria.

33h Permite ampliar el control de Ctrl-Break (Ctrl-C) para que se realice en cada llamada al sistema.

35h Lee en ES:BX el vector de interrupción asociado con la interrupción definida en AL.

36h Leer el espacio libre en el disco definido en DL. Se retorna BX=clusters disponibles, DX=clusters por drive, CX=bytes por sector, AX=sectores por cluster.

38h Retornar en 32 bytes de memoria direccionada por DS:DX la información dependiente del país (cuyo código se encuentra en AL). Si DX=-1, se utiliza para cambiar el país seleccionado.

39h Crea el subdirectorio contenido en una cadena terminada con ceros, apuntada por DS:DX.

3Ah Borra el subdirectorio especificado como en 39h.

3Bh Cambiar el directorio actual a aquel especificado como en 39h.

3Ch Crear archivo usando handle.

3Dh Abrir un archivo o dispositivo con handle.

3Eh Cerrar el handle de un archivo o dispositivo.

3Fh Leer de archivo o dispositivo.

40h Escribir en un archivo o dispositivo.

41h Borrar un archivo.

42h Mover un apuntador de archivo.

43h Cambiar atributos. Los atributos pueden ser asignados a un archivo en su creación, y pueden ser asignados o cambiados llamando a la función 43h. Los atributos que son afectados por la función 43h son:

Hex	Definición
01h	Read-only
02h	Hidden
04h	System
20h	Archivo

44h Define o lee la información de dispositivos asociados con handles abiertos. Puede además enviar o recibir cadenas de control a un dispositivo o a un handle de dispositivo.

45h Duplicar un handle de archivo.

46h Obligar un duplicado de handle.

47h Regresar texto del directorio corriente.

48h Reservar un espacio de memoria disponible, con extensión especificada en BX.

49h Liberar memoria reservada.

4Ah Ampliar o reducir un bloque de memoria reservada.

4Bh Cargar y ejecutar un programa.

4Ch Terminar un proceso.

4Dh Sacar el código de retorno de un subproceso.

4Eh Encontrar archivo que cumple con la especificación y atributos.

4Fh Paso por un directorio buscando archivo.

54h Retornar estado de la bandera de verificación después de escritura.

56h Mover un archivo a otro directorio.

57h Leer o cambiar fecha y hora de un archivo.

ANEXO:

FUNCIONES PRINCIPALES DE CODE BASE

A continuación se presentan las principales funciones de CodeBase utilizadas en el programa de administración de datos, y archivos.

#include "d4all.h"

Esta línea especifica que prototipo de función debe ser incluido para el encabezado del archivo.

extern unsigned_stklen = 15.000;

La línea especifica la medida del stack para el compilador TURBOC.

C4CODE code_base;

La línea declara en memoria para la estructura de datos "C4CODE".

D4DATA *base;

La línea de arriba declara el puntero para una estructura "D4DATA" este puntero retorna por "d4opend()" la llamada es grabada en esta variable.

F4FIELD

Define la estructura de registros, con sus respectivos campos.

F4FIELD *

Este es un puntero a la estructura F4FIELD, el cual permite localizar el dato cuando la estructura ha sido abierta.

T4TAG

Define la estructura para archivos índices.

F4FIELD_INFO

Permite definir cada uno de los campos de la estructura de datos.

F4TAG_INFO

Permite definir cada uno de los campos de la estructura de archivos índices.

cb.safety = 0

Pone una seguridad para no generar un código de error, en el momento que retorna, especialmente en las funciones d4create, y d4rename.

d4append d4append(D4DATA*)

Añade un record al archivo de datos.

d4close d4close(D4DATA*)

Cierra un archivo de datos.

d4close_all d4close_all(D4DATA*)

Cierra todos los archivos de datos.

d4create D4DATA *d4create(C4CODE *, char *name, F4FIELD_INFO*, T4TAG_INFO)

Crea el archivo de datos y posiblemente el índice del archivo.

d4field F4FIELD *d4field(D4DATA*, char *field_name)

Retorna el puntero al campo.

d4field_j F4FIELD *d4field_j(D4DATA*, char *field_name)

Retorna el puntero del campo j.

d4index D4INDEX *d4index(D4DATA *, char *index_name)

retorna el puntero a un archivo índice.

d4init d4init(C4CODE)

Inicializa el Codebase

d4opend D4DATA *d4opend(C4CODE*, char *file_name)

Abre el archivo de datos.

d4pack d4pack(D4DATA*)

Empaqueta el archivo de datos, para remover borrando archivos.

d4reindex d4reindex(D4DATA *)

Reindexa todos los archivos abiertos de un archivo de datos.

d4seek d4seek(D4DATA *, char *seek_info)

Busca un string de caracteres.

d4top d4top(D4DATA *)

Posiciona en la parte superior del archivo de datos.

m4assing m4assing(F4FIELD *, char *prt)

Asigna un string a una entrada memo.

m4assing_n m4assing_n(F4FIELD *, char unsigned)

Asigna un número de bites de memoria a una entrada memo.

m4str *m4str(F4FIELD *)

SISTEMA DE ADQUISICION DE LARINGOGRAMAS

MANUAL DEL USUARIO

Edición de paciente

código de paciente.....: 001

fecha.....: 05/07/93

observación.....: Paciente de la E.P.N

(G)RAFICAR (N)UEVO (B)ORRAR (S)ALIR

Pulsando la tecla ENTER, el programa despliega la información pertinente del paciente.

Desde el menú que se muestra en la parte inferior de la pantalla se puede:

(G)RAFICAR.- Permite visualizar el laringograma correspondiente a la sesión de trabajo actual y que ha sido anteriormente grabado.

Dentro de la pantalla de graficación existen opciones que permiten ampliar la escala de frecuencia de la señal original:

1.3.- Nuevo Paciente.

Edición de paciente

código de paciente.....: 001

fecha.....: 05/07/93

observación.....: Paciente de la E.P.N

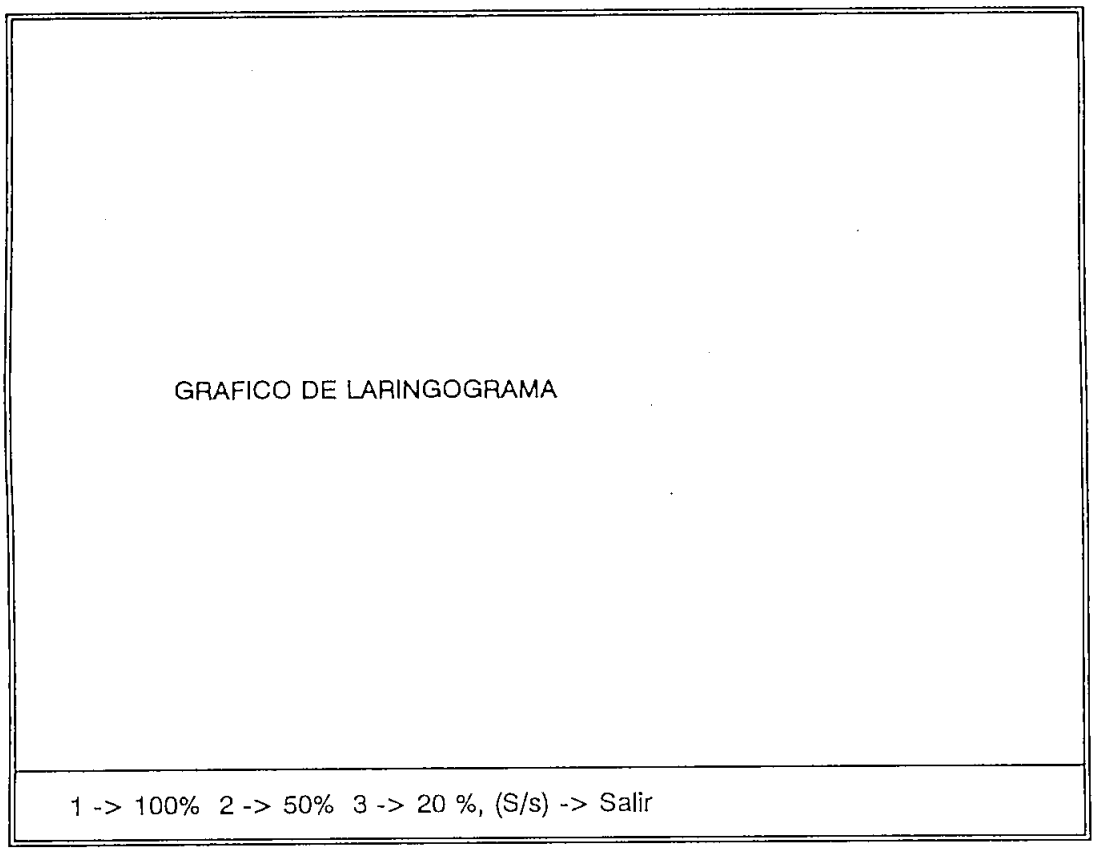
(G)RAFICAR (N)UEVO (B)ORRAR (S)ALIR

Pulsando la tecla ENTER, el programa despliega la información pertinente del paciente.

Desde el menú que se muestra en la parte inferior de la pantalla se puede:

(G)RAFICAR.- Permite visualizar el laringograma correspondiente a la sesión de trabajo actual y que ha sido anteriormente grabado.

Dentro de la pantalla de graficación existen opciones que permiten ampliar la escala de frecuencia de la señal original:



Al salir de la pantalla de graficación, el programa va al siguiente archivo del paciente, y así sucesivamente hasta llegar al final de los archivos grabados.

(N)UEVO.- Permite añadir un nuevo laringograma al archivo del paciente. Si se escoge esta opción, aparece una pantalla que permite llenar nuevos datos referentes al nuevo laringograma que se desea almacenar.

Ingreso de datos del nuevo laringograma

código.....: 001

fecha.....:

observaciones.....:

Archivo laringograma.....:

(*) ERROR: no existe archivo de datos <?>

Cuando se solicita "Archivo laringograma :", se debe ingresar el nombre del laringograma que se desea añadir en ese momento, y que debió ser previamente grabado en el disco, o diskette con la opción (2) del menú principal (2.- Adquisición de datos). Si el archivo seleccionado no existe se presenta un mensaje de error (*).

Después de grabado, el sistema va al siguiente archivo del mismo paciente; y si es el último retorna, al menú anterior.

(B)ORRAR.- Permite marcar el archivo que se visualiza en ese momento, para borrarlo posteriormente, lo cual se realiza con la opción (3) del menú principal: "mantenimiento de datos ".

(S)ALIR.- Permite abandonar esta pantalla de edición para retornar al menú anterior.

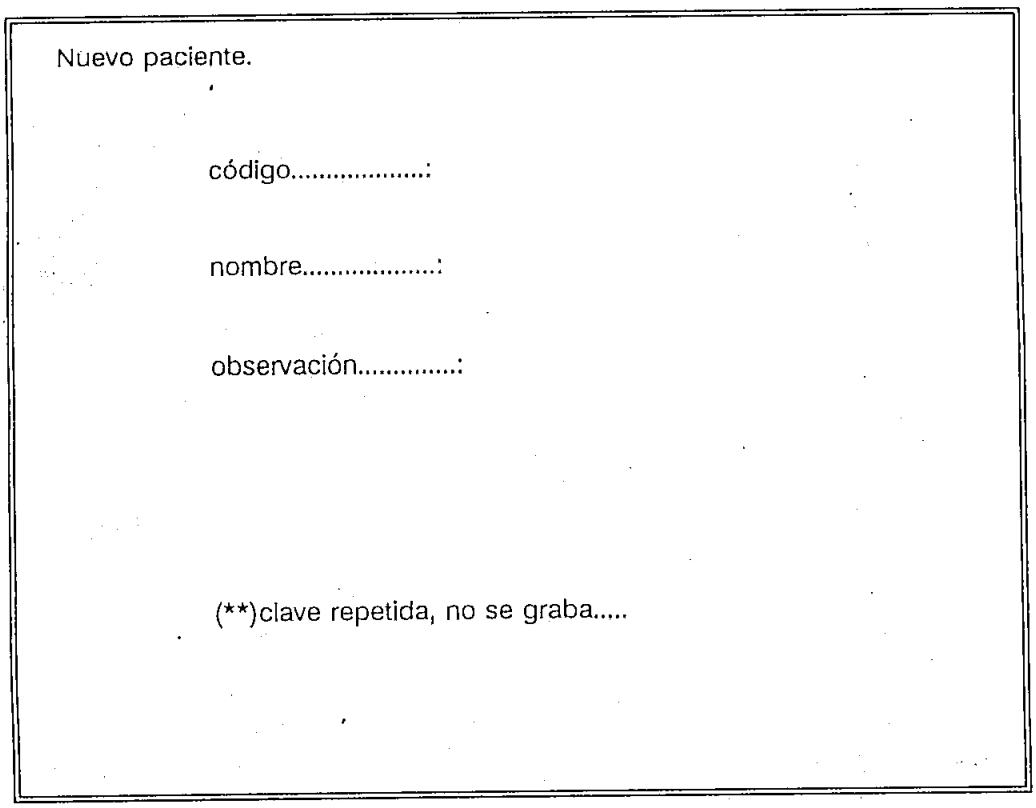
1.3.- Nuevo Paciente.

Esta opción permite el ingreso de un nuevo paciente dentro de la base de datos. Se debe asignar un código a cada paciente. Si este código que se dio al nuevo paciente, se halla repetido, en la parte inferior de la pantalla sale la siguiente advertencia:

(**) Clave repetida, no se graba....

Y la información ingresada no se acepta. Es recomendable antes de hacer uso de esta opción, visualizar el listado de los pacientes para verificar los códigos ya utilizados.

El formato de nuevo paciente es el siguiente:



1.4.- Borrar paciente.

Esta opción permite borrar un paciente del archivo de datos, realiza una función similar a la del borrado en edición de paciente.

2.- ADQUISICION DE DATOS.

Cuando el usuario selecciona esta opción llama al programa principal de adquisición de datos (laringogramas).

Inicialmente el programa pregunta el nombre del archivo con el que se desea grabar el laringograma.

Se debe ingresar el nombre y presionar ENTER para continuar. Inmediatamente se despliega en la pantalla una cuadrícula sobre la cual se comienza a desplegar la señal captada desde la laringe.

En la parte superior de la pantalla se encuentra el nombre del archivo que, eventualmente, será asignado al laringograma al grabarse. El menú principal, y las opciones de ayuda que permiten guiar al paciente para que escoja lo que desee hacer durante la captación de señales tiene las siguientes opciones:

(M)EMORIZAR, (G)RABAR, (R)EDIBUJAR, (B)ORRAR, (S)ALIR.

Seleccionar pulsando la tecla correspondiente a la primera letra de la función a realizar.

(M)EMORIZAR.-

Congela en la pantalla del computador la imagen que aparece en la pantalla del computador.

(G)RABAR.- Permite almacenar en un diskette o dispositivo similar la imagen que se presenta en la pantalla.

(R)EDIBUJAR.-

Inicia nuevamente la captación de señales de la laringe, sin borrar la pantalla o imagen anterior.

(B)ORRAR.- Limpia la pantalla de adquisición de datos completamente, redibujando las escalas y reinicia el proceso de adquisición de datos.

(S)ALIR.- Permite abandonar el proceso de adquisición de datos retornando al menú principal del programa.

A continuación de describe brevemente el proceso a seguirse durante la captación de los laringogramas cuando se ha ingresado al programa de adquisición de datos.

Para captar una señal en forma correcta es necesario colocar el estetoscópio pegado a la laringe, en una posición que permita captar vibraciones en el momento del habla. Preferible a un extremo de la así denominada "manzana de Adán".

En la pantalla del computador podemos observar las señales producidas desde la laringe, con la opción (M)EMORIZAR, podemos congelar la imagen en la pantalla, si esta es la correcta, y deseamos grabarla en el diskette u otro dispositivo similar. Se presiona la tecla (G)RABAR. luego de que se almacenó el archivo, el sistema retorna nuevamente a captar otra señal. Con la opción (B)ORRAR, se puede limpiar la pantalla, y redibujar las escalas, e iniciar el proceso de captación de señal.

Si el laringograma congelado con la opción (M)EMORIZAR, no se desea grabarlo, simplemente con (R)EDIBUJAR, se capta nuevamente la señal o con (B)ORRAR se limpia completamente la pantalla.

Es necesario anotar, que la opción (G)RABAR, permite almacenar cuantas veces se desee un archivo, sobreponiendo el nuevo archivo seleccionado al grabado anteriormente, quedando asignado el último archivo grabado, al nombre dado al inicio del programa. Por la razón indicada anteriormente, para cada nuevo laringograma (con diferente nombre) que se desee cargar se debe ingresar nuevamente al programa de adquisición de datos.

3.- MANTENIMIENTO DE DATOS.

Esta opción es utilizada únicamente si durante la sección de trabajo el usuario decidió borrar algún archivo, marcándola como tal.

Para finalizar totalmente la sesión de trabajo pulsamos la tecla " 0 ", saliendo completamente del programa.

Esta opción es utilizada únicamente si durante la sección de trabajo el usuario decidió borrar algún archivo, marcándola como tal.

Para finalizar totalmente la sesión de trabajo pulsamos la tecla " 0 ", saliendo completamente del programa.