# TEMA 2
# Desarrollo de Algoritmos

# Creación y consulta de Tablas

| Binario | 00H | 01H | 02H | 03H | 04H | 05H | 06H | 07H | 08H | 09H | 0AH | 0BH | 0CH | 0DH | 0EH | 0FH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASCII | 30H | 31H | 32H | 33H | 34H | 35H | 36H | 37H | 38H | 39H | 41H | 42H | 43H | 44H | 45H | 46H |

```
/************************************************
Funcion Bin_Ascii
Convierte un número hexadecimal en ASCII
************************************************/
unsigned char code
   tabla_ascii[]={0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,
                  0x38,0x39,0x41,0x42,0x43,0x44,0x45,0x46};

unsigned char Bin_Ascii(unsigned char car) {
return tabla_ascii[car];
}
```

# Transferencia de bloques de datos

## Zonas de memoria:

| Zona de memoria 8051 | Modificador | Tipo puntero | Cantidad máxima de datos | Depuración |
|---|---|---|---|---|
| Interna de Datos | data | char | 256 bytes | D: |
| Externa de Datos | xdata | int | 64 Kbytes | X: |
| Código | code | int | 64 Kbytes | C: |
| Acceso Indirecto | idata | char | 128 bytes | I: |

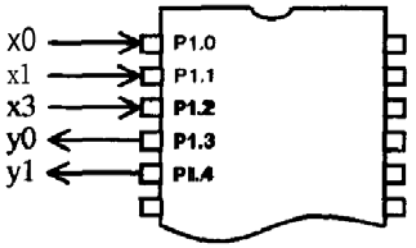
## Las transferencias de datos pueden ser:

- De memoria interna a externa (data a xdata) y viceversa.
- De memoria de código a interna (code a data).
- De memoria de código a externa (code a xdata).

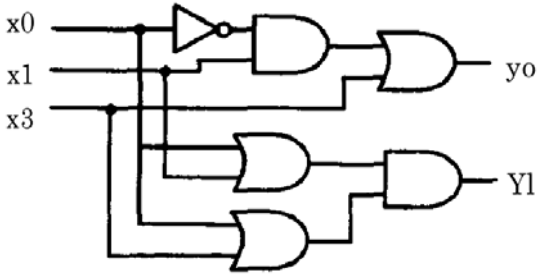
```
void trans(unsigned char data *ptr1,unsigned char code *ptr2, unsigned int
tam) {
int i;

     i=0;
     while (i<tam) {
          *(ptr1+i)=*(ptr2+i); i++;
     }
}
```

# Funciones boleanas

```
/************************************
 Funcion  F_Bol
 Entradas:    P1.0, P1.1, P1.2
 salidas:     P1.3, P1.4
 ************************************/

/* entradas */
sbit X0=P1^0;
sbit X1=P1^1;
sbit X3=P1^2;
/* salidas */
sbit Y0=P1^3;
sbit Y1=P1^4;

void F_Bol(void) {
   Y0=(~X0&X1)|X3;
   Y1=(X0|X1)&(X0|X3);
}
```




Funciones lógicas



# Retardos de Tiempo

```
 /*************************************************
      Función RETARDO
 *************************************************/
void retardo(int msec) {
int i;

while (msec--)
 for(i=0;i<100;i++);
}

/*************************************************
 Función RETARDO2
 *************************************************/

void retardo2(int msec) {
int i,j;

while (msec--)
 for(i=0;i<100;i++)
      for(j=0;j<100;j++);
}
```

# Operaciones matemáticas con aritmética entera

- El empleo de float o double sobrecarga los recursos limitados del microcontrolador.
- Se emplea int en lugar de float o double.
- Se realiza un escalado de los números y una cantidad fija de decimales.

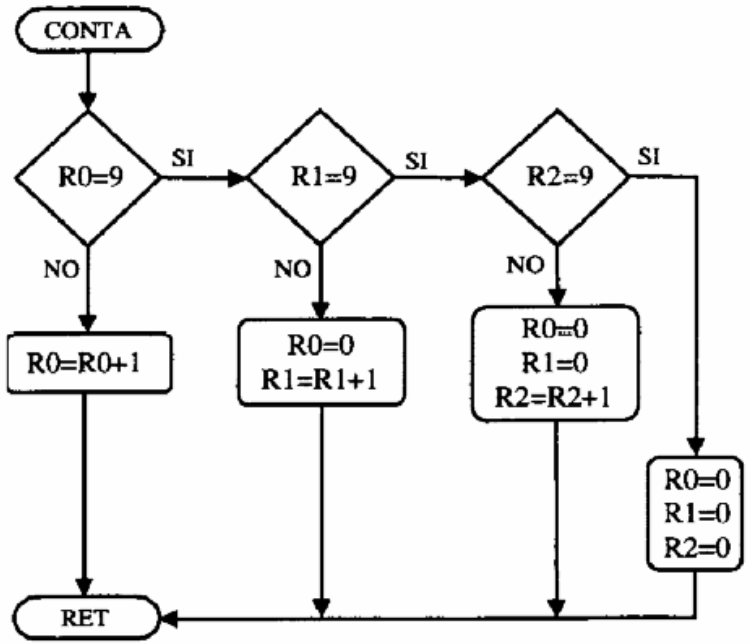
```
/*********************************************
 Subrutina raiz_cuadrada (Raiz cuadrada por 1000)
 Entrada: valor entre 0 y 1 (restringido y multiplicado por 10)
 Salida: El resultado multiplicado por 1000

*********************************************/

unsigned int code
tabla_raiz[10]={0,1000,1414,1732,2,2236,2449,2645,2828,3};

unsigned int code raiz_cuadrada(unsigned int var) {

if (var==0) return 0;
if (var<10) return tabla_raiz[var];
else return 0;
```

```
}
```

# Contadores BCD

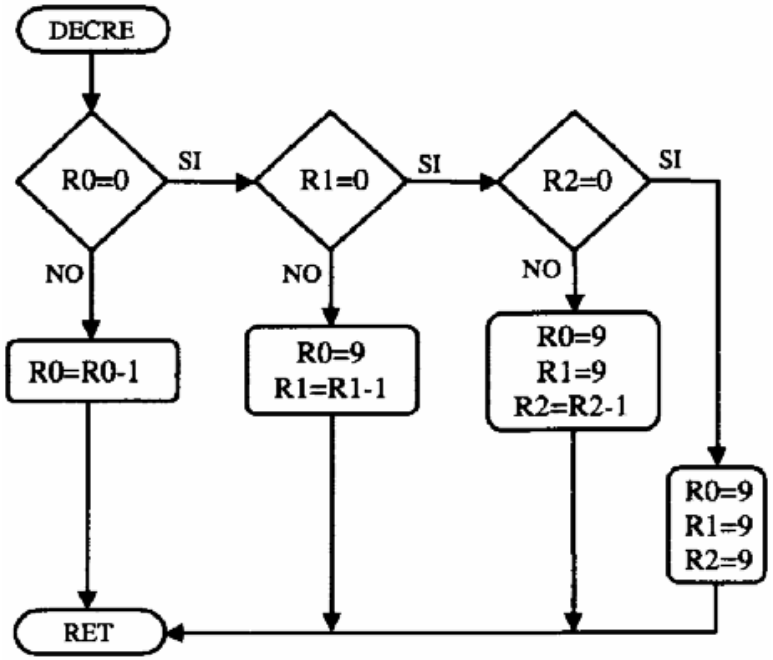
- Para mostrar información numérica en pantalla LED o LCD es necesario convertir a código BCD.
- El microprocesador solo maneja formato binario.
- Cuando tenemos pocos recursos es mejor trabajar directamente en BCD.

```
/*******************************************
  Funcion Conta
  Incrementa unidades

*******************************************/

unsigned char uni=0,dec=0,cen=0;

void Conta(void) {
if (uni==9) {
     if (dec==9) {
          if (cen==9) uni=dec=cen=0;
          else {uni=dec=0;cen++;}
     }
     else {uni=0;dec++;}
}
else uni++;

}
```



# Contadores BCD

```
/*****************************************
***
  Funcion Decre
  Decrementa unidades
*****************************************
**/

void Decre(void) {

if (uni==0) {
 if (dec==0) {
      if (cen==0) uni=dec=cen=9;
      else {uni=dec=9;cen--;}
 }
 else {uni=9;dec--;}
}
else uni--;
}
```



# Contador BCD con ajuste decimal

```
/*******************************************************
  Funcion Conta2
  Ajuste decimal en C
*******************************************************/

unsigned char dig01=0x00,dig23=0x00;

void Conta2(void) {

/* incrementa contador */
dig01++;
/* ajuste decimal en unidades */
if ((dig01&0x0F)>9) dig01+=6;
/* ajuste decimal en decenas */
if ((dig01&0xF0)>0x90) dig01+=0x60;

/* si ha habido acarreo se incrementan
 centenas y se vuelve a ajustar */
if (dig01==0) {
    dig23++;
    if ((dig23&0x0F)>9) dig23+=6;
    if ((dig23&0xF0)>0x90) dig23+=0x60;
    }
}
```

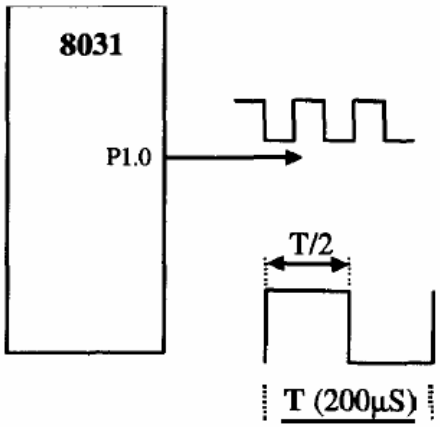
# Generación de señal cuadrada

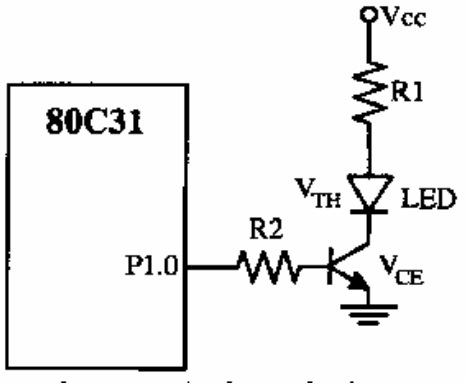
```
/* definición de salida */
sbit SALIDA=P1^0;

void retardo(unsigned char msec) {

while (msec--);
}

void main(void) {
/* Bucle infinito en función main */
while(1) {
 /* Complementa estado lógico de P1.0 */
 SALIDA=~SALIDA;
 /* Llama a la rutina de retardo */
 retardo(18);
 }
}
```

```
RSEG  ?PR?_retardo?GENERADOR
?C0001:
     MOV       R6,R7
     DEC       R7
     MOV       A,R6
     JNZ       ?C0001
     RET

RSEG  ?PR?main?GENERADOR
?C0004:
     CPL       SALIDA
     MOV       R7,#012H
     LCALL     _retardo
     SJMP      ?C0004
```

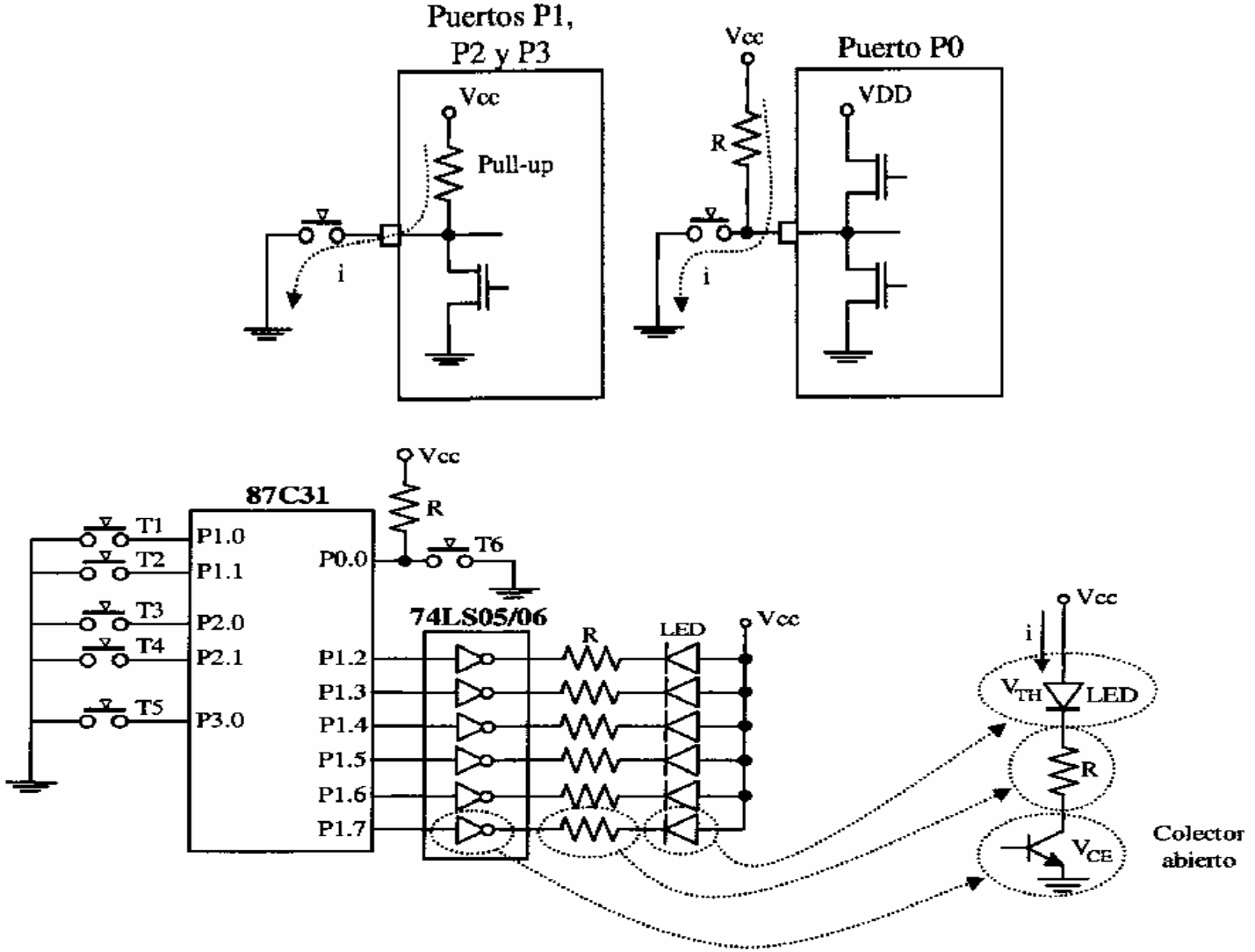
$$T_{Ret} = 5 + 5 \cdot R7 + 2 \ (CM)$$

$$t_{CPL} = 1 + 2 + 2 + t_{Ret} = 100 CM \text{ o } 100 \mu s$$



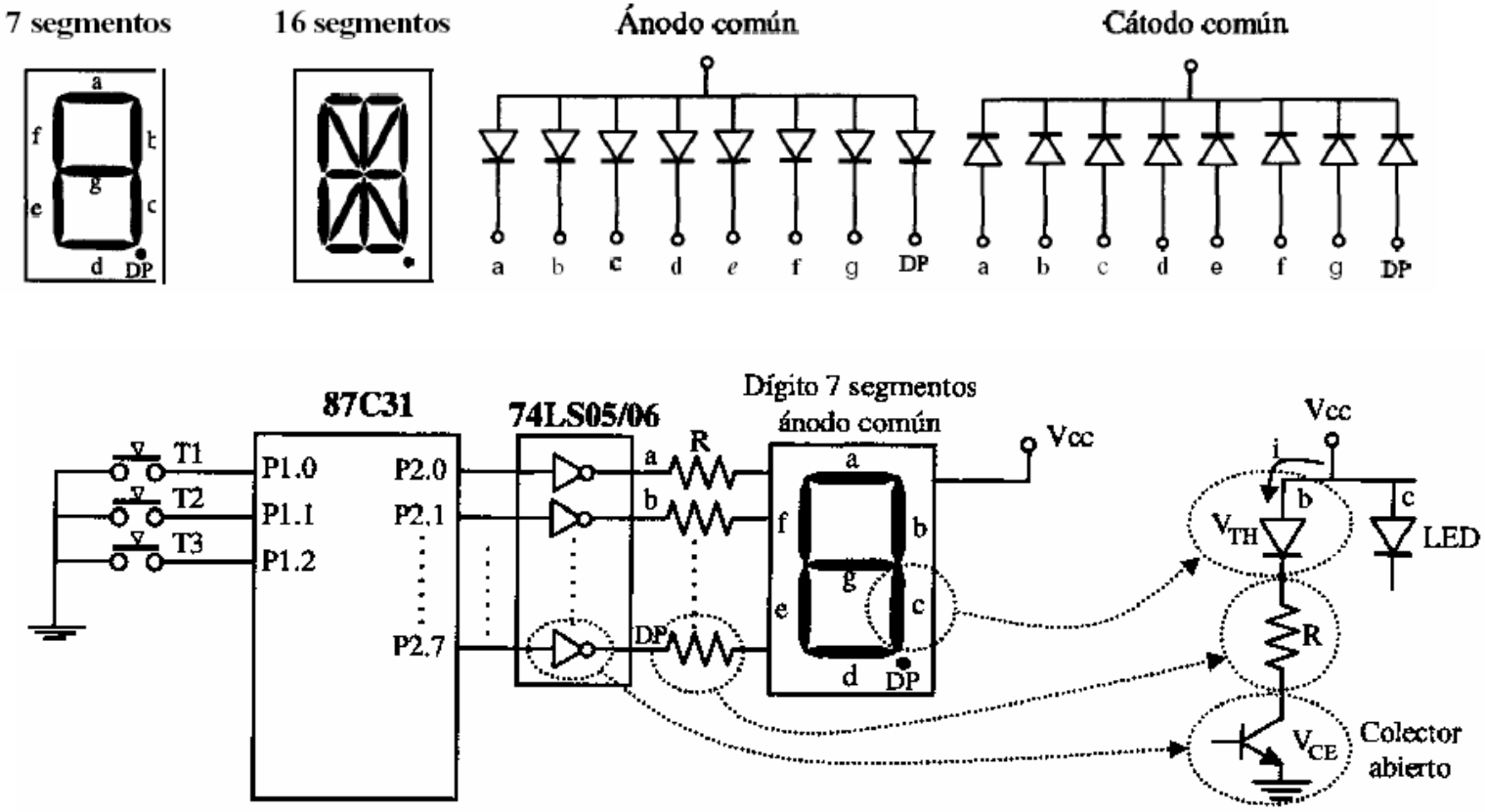



# Conexión de teclas al microcontrolador



```
#include <reg51.h>
/**************************************
 Programa para la lectura de la teclas
**************************************/

/* Definicion de teclas */
sbit Tecla_T1=P1^0;
sbit Tecla_T2=P1^1;
sbit Tecla_T3=P2^0;
sbit Tecla_T4=P2^1;
sbit Tecla_T5=P3^0;
sbit Tecla_T6=P0^0;

/* Definicion de LEDS */
sbit Led_L1=P1^2;
sbit Led_L2=P1^3;
sbit Led_L3=P1^4;
sbit Led_L4=P1^5;
sbit Led_L5=P1^6;
sbit Led_L6=P1^7;

/**************************************
 Rutina Principal (Testeo de teclas,encendido/apagado de leds)
**************************************/
void main(void) {
/* pongo a 1 entradas */
/* aunque despues del reset ya estan a uno */
Tecla_T1=Tecla_T2=Tecla_T3=Tecla_T4=Tecla_T5=1;

while (1) {
/* compruebo teclado */
     if(!Tecla_T1) P1=0x07; /*0000 0lllb P1.2 apaga resto*/
     else if(!Tecla_T2) P1=0x0B; /*0000l0llb P1.3 resto*/
     else if(!Tecla_T3) P1=0x13; /*0001 00llb P1.4 resto*/
     else if(!Tecla_T4) P1=0x23; /*0010 0011b P1.5 resto*/
     else if(!Tecla_T5) P1=0x43; /*0l00 0011b P1.6 resto*/
     else if(!Tecla_T6) P1=0x83; /*1000 0011b P1.7 resto*/
     /* Apaga solo leds conectados a P1.2-7 */
     else P1&=0x03;
     }
}
```

# Conexión digito 7 segmentos

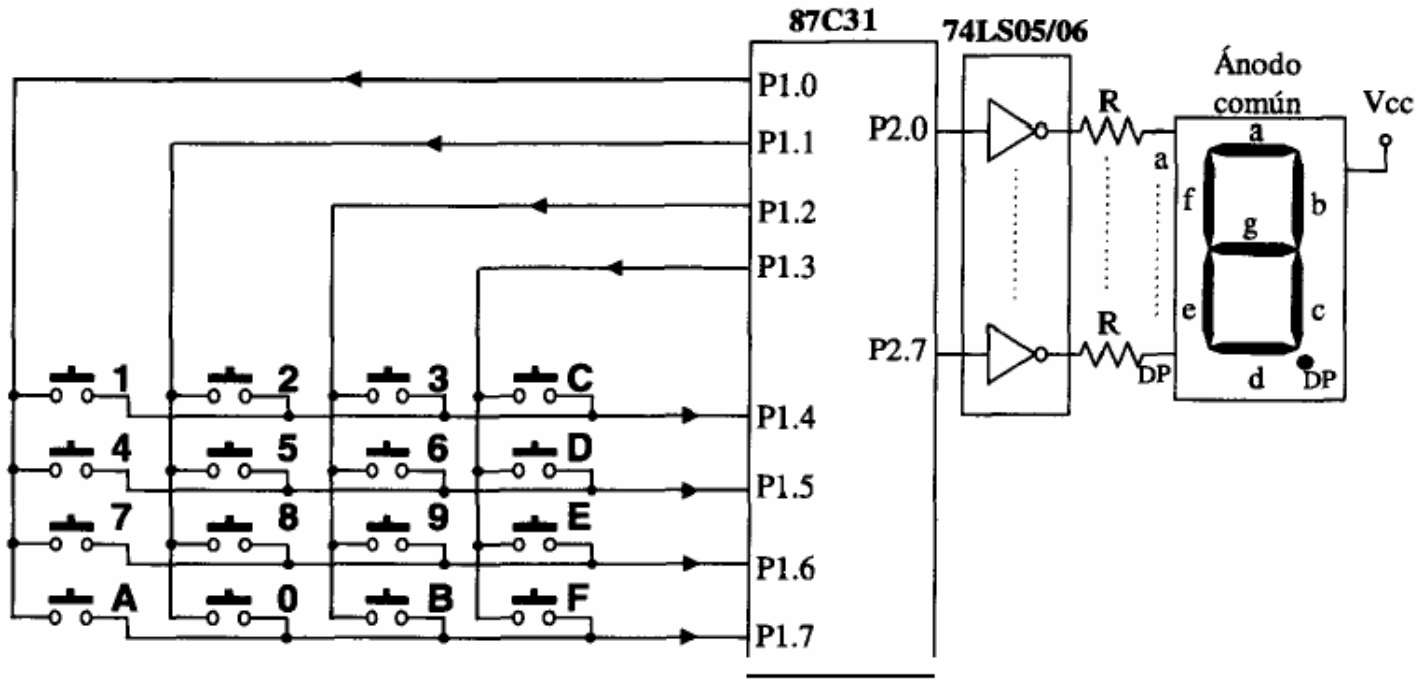


# Conexión digito 7 segmentos

```
#include <reg51.h>
/*********************************************************
 Programa para la conexión de un digito de 7-segmentos
*********************************************************/
/* definición teclas */
sbit Tecla_T1=P1^0;
sbit Tecla_T2=P1^1;
sbit Tecla_T3=P1^2;

#define DISPLAY P2

/*********************************************************
 Rutina principal
*********************************************************/
void main(void) {

DISPLAY=0; /* Apaga todos los leds del dígito */
while (1) {
 /* Comprueba si se ha pulsado T1 */
 if (!Tecla_T1) DISPLAY=0x77; /* 0111 0111b Pone letra A */
 /* Comprueba si se ha pulsado T1 */
 else if (!Tecla_T2) DISPLAY=0x7C; /* 0111 1100b Pone letra B */
 /* Comprueba si se ha pulsado T1 */
 else if (!Tecla_T3) DISPLAY=0x79; /* 0111 1001b Pone letra C */
 else DISPLAY=0;
 }
}
```

# Conexión teclado matricial

- Se realiza un barrido de teclas empezando por la columna de la izquierda.
- Detecta tecla pulsada y decide una sola tecla en función de su prioridad.
- Devuelve la primera tecla detectada en **ASCII mayúsculas**.
- Visualiza el valor de la tecla convirtiendo el carácter ASCII detectado en código de 7 segmentos.



# Teclado matricial

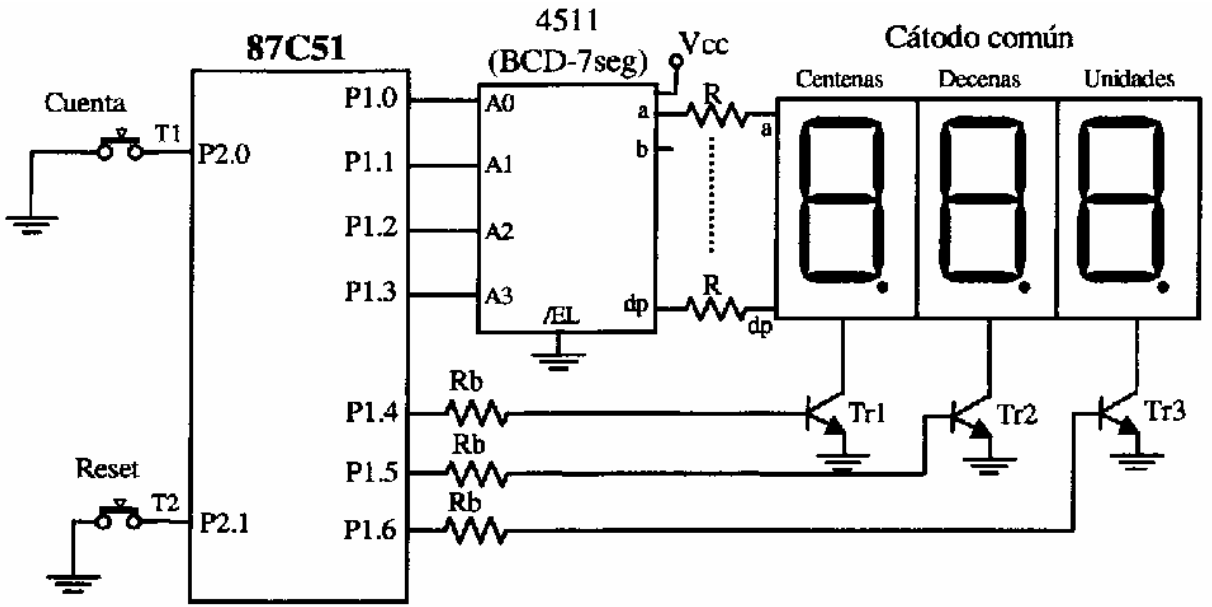
```
#include <reg51.h>

bit Det_tecla(void) {
/* Lee puerto P1(filas) y fuerza 4 bits bajos 1, si no pulsa devuelve cero. */
return ~(P1&0x0F);
}

/* convierte tecla pulsada en numero de tecla con prioridad*/
unsigned char code cod_pri[]={0,1,2,1,3,1,2,1,4,1,2,3,1,2,3,1};
/* convierte numero de tecla en caracter a generar */
unsigned char code tabla_teclado[]={'1','4','7','A','2','5','8','0',
                   '3','6','9','B','C','D','E','F'};

unsigned char lee_teclado(void) {
P1=0xFE; /* 1111 1110b   Habilita primera col. */
if (Det_tecla())     return tabla_teclado[cod_pri[(P1&0xF0)>>4]];
P1=0xFD; /* 1111 1101b   Habilita segunda col. */
if (Det_tecla())     return tabla_teclado[cod_pri[(P1&0xF0)>>4]+4];
P1=0xFB; /* 1111 1011b   Habilita tercera col. */
if (Det_tecla())     return tabla_teclado[cod_pri[(P1&0xF0)>>4]+8];
P1=0xF7; /* 1111 0111b   Habilita cuarta col. */
if (Det_tecla())     return tabla_teclado[cod_pri[(P1&0xF0)>>4]+12];
return 0;
}
```

# Visualización en 7 segmentos

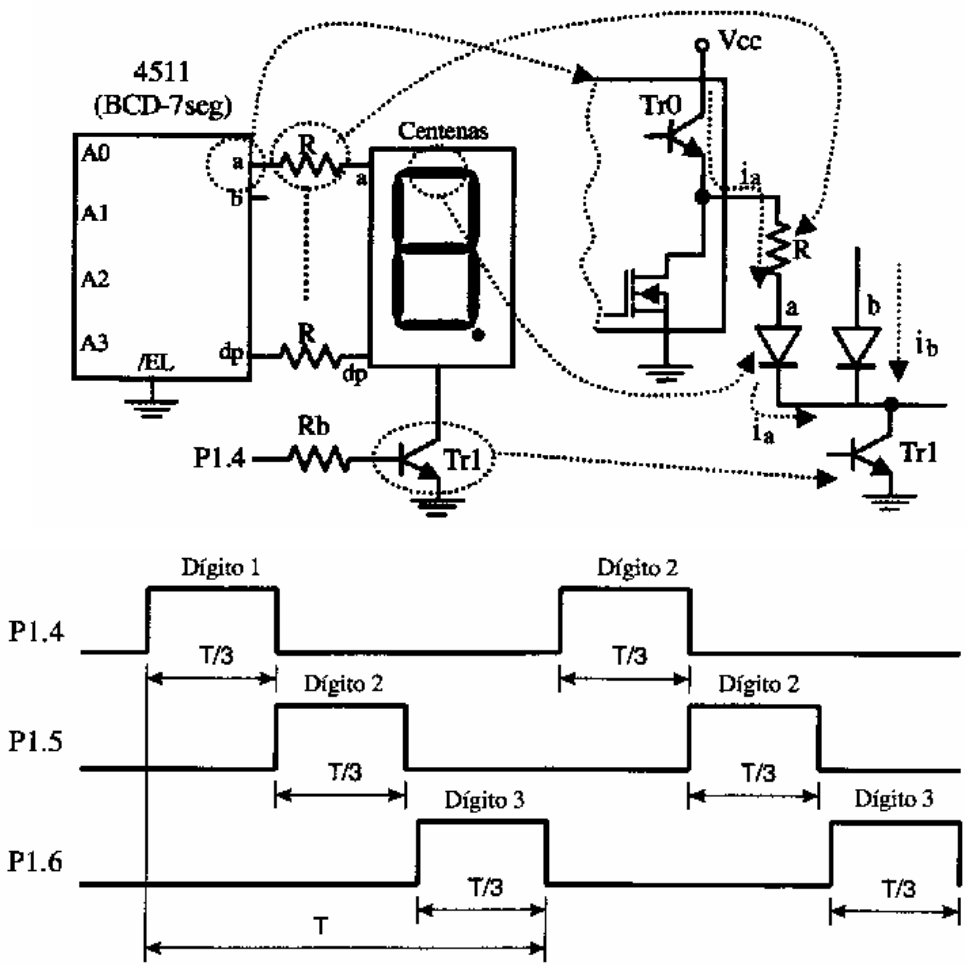
```
/*************************************************
 Visualización de un carácter en 7 segmentos
*************************************************/
unsigned char code ascii_7seg[16]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,
                                   0x7D,0x07,0x7F,0x6F,0x77,0x7C,0x39,
                                    0x5E,0x79,0x71};
void Display(unsigned char cod_ascii) {
if (cod_ascii<0x30) return; /* si intentamos visualizar codigos sale */
if (cod_ascii<0x39) { /* si intentamos visualizar números resta 0x30 */
  P2=ascii_7seg[cod_ascii-0x30]; /* restamos 0x30=Ascii('0') */
  return;
}
if (cod_ascii<0x47) { /* si intentamos visualizar letras mayúsculas(A-Z) resta 0x36
*/
  P2=ascii_7seg[cod_ascii-0x37]; /* restamos 0x37=Ascii('A') */
  return;
} /* Todo para obtener un índice de 0 a 15 para el array ascii_7seg */
}

void main(void) {
unsigned char tecla;
P2=0; /* Apaga todos los leds del digito */
while (1) {
  if (tecla=lee_teclado())
       Display(tecla);
   } }
```

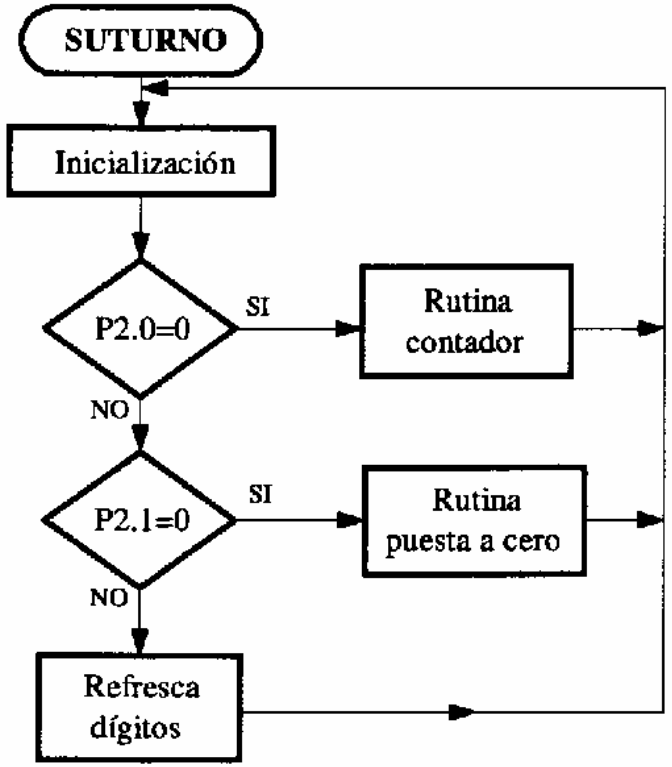
# Multiplexado LED

- Aplicación de Su Turno.
- Cada segmento se enciende por separado.
- Se van cambiando rápidamente para dar la sensación que están encendidos continuamente.
- Se emplean transistores para multiplexar cada segmento.
- Se emplea un decodificador BCD a 7 segmentos, por lo que el código que entrega el micro ahora es BCD.



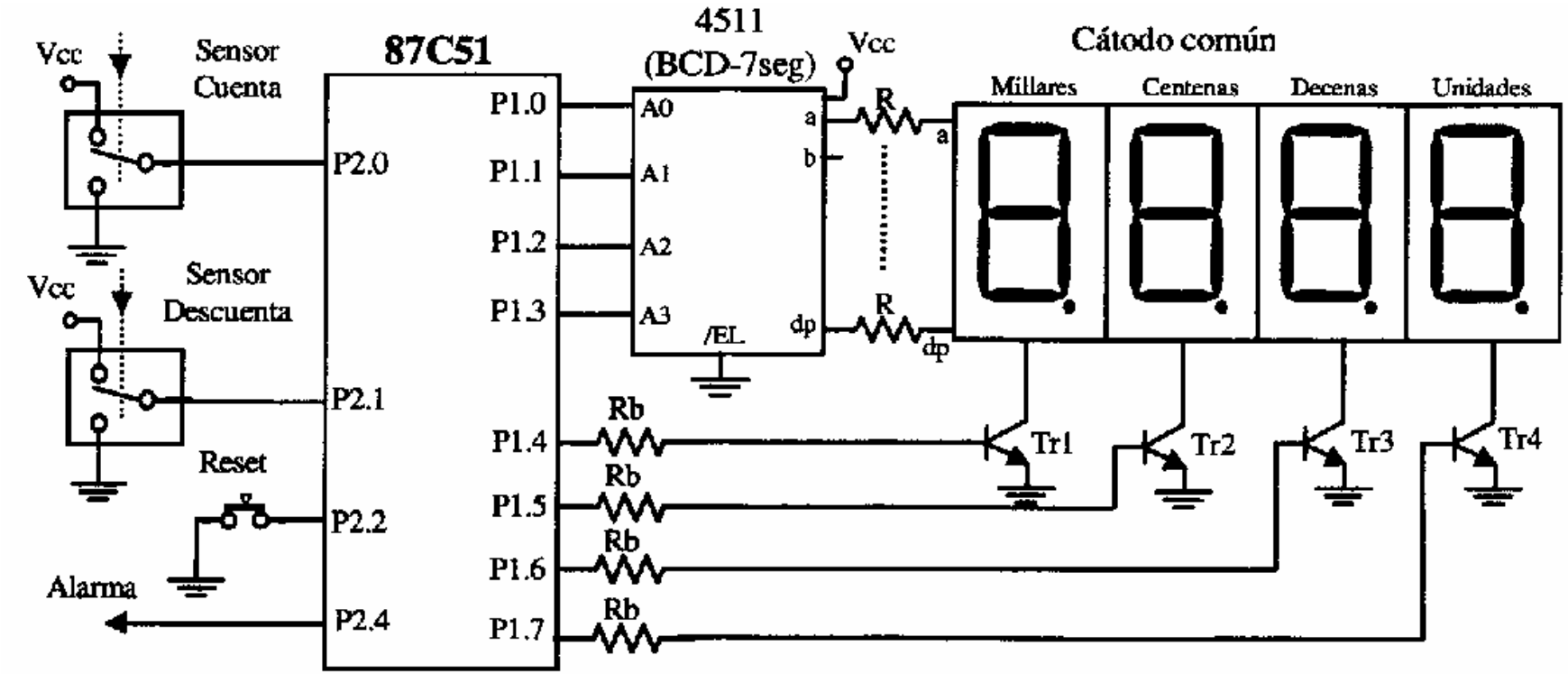
# Multiplexado LED



```
#include <reg51.h>
/* prototipo funcion delay */
void delay(unsigned int);

/***********************************************
  Programa para la gestión del turno de espera
 ***********************************************/
/* pulsadores */
sbit CUENTA=P2^0;
sbit RESET=P2^1;
/* salidas transistores */
sbit TRAN1=P1^4;
sbit TRAN2=P1^5;
sbit TRAN3=P1^6;

unsigned char uni,dec,cen; /* variables de cuenta */

/***********************************************
 Funcion CONTA: incrementa contador BCD
***********************************************/
void Conta(void) {
if (uni==9) {
      if (dec==9) {
            if (cen==1) uni=dec=cen=0;
            else {uni=dec=0;cen++;}
      }
      else {uni=0;dec++;}
}
else uni++;
}
/***********************************************
 Función Refresca: actualiza la pantalla numérica
***********************************************/
void Refresca(void) {
P1|=cen; /* coloco centenas */
TRAN1=1; /* enciendo centenas */
delay(10); /*espero un tiempo */
TRAN1=0; /* apago centenas */
P1|=dec; /* coloco decenas */
TRAN2=1; /* enciendo decenas */
delay(10); /*espero un tiempo */
TRAN2=0; /* apago decenas */
P1|=uni; /* coloco unidades */
TRAN3=1; /* enciendo unidades */
delay(10); /*espero un tiempo */
TRAN3=0; /* apago unidades */
}

/***********************************************
 Función Principal
***********************************************/
void main(void) {
 unsigned char i;

 P1=0x00; /* Pone a cero entrada del 4511 y en corte Tr1, Tr2 y Tr3 */
 uni=dec=cen=0;
 while (1) {
       if (!CUENTA) Conta();
       if (!RESET) uni=dec=cen=0;
       for (i=0;i<20;i++) Refresca(); /* muestra valor en pantalla */
       /* espera un poco para evitar rebotes de pulsadores */
      }
}
```

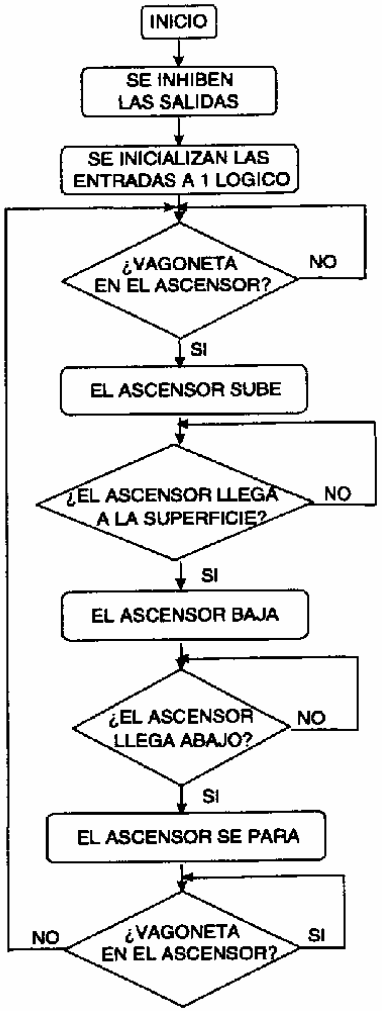
# Contador de Piezas

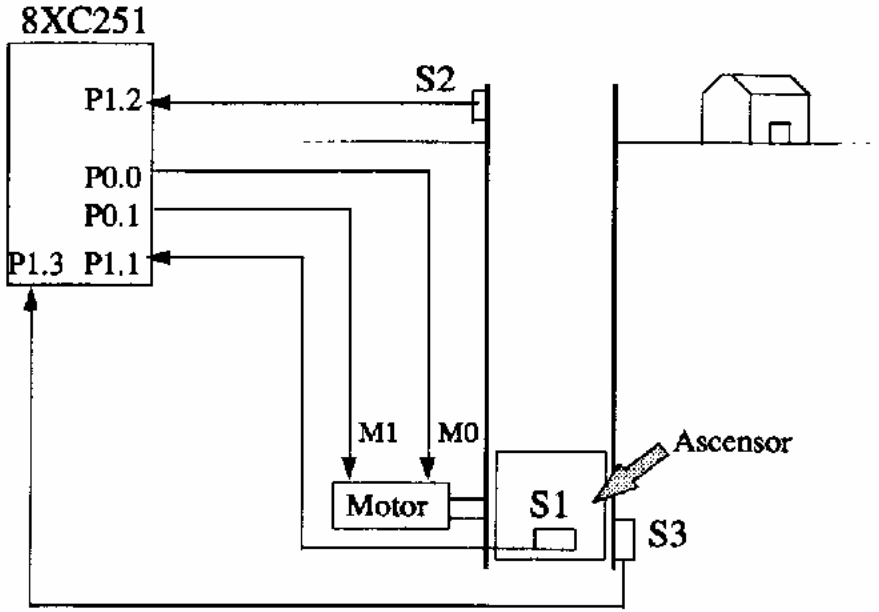


```
#include <reg51.h>
/*************************************************************
**
 Programa para el contador de piezas
*************************************************************
*/
#define OFF 0
#define ON 1
/* pulsadores */
sbit CUENTA=P2^0;
sbit DESCUENTA=P2^1;
sbit RESET=P2^2;
/* salidas transistores */
sbit TRAN1=P1^4;
sbit TRAN2=P1^5;
sbit TRAN3=P1^6;
sbit TRAN4=P1^7;
sbit ALARMA=P2^4;

/* prototipo de rutina retardo */
void delay(unsigned int);

unsigned char uni,dec,cen,mil; /* variables de cuenta */
/***********************************************
 Funcion Cuenta: incrementa contador BCD y señala alarma
***********************************************/
void Cuenta(void) {
if (uni==9) {
     if (dec==9) {
          if (cen==9)
               if (mil==9) ALARMA=ON; /* señala alarma y no
modifica cuenta */
               else {uni=dec=cen=0;mil++;}
          else {uni=dec=0;cen++;}
     }
     else {uni=0;dec++;}
}
else uni++;
}
/***********************************************
 Funcion Descuenta: decrementa contador BCD
***********************************************/
void Descuenta(void) {

if (uni==0) {
     if (dec==0) {
          if (cen==0)
               if(mil==0) ALARMA=ON;
               else {uni=dec=cen=9;mil--;}
          else {uni=dec=9;cen--;}
     }
     else {uni=9;dec--;}
}
```

```
/************************************************
 Funcion Refresca: actualiza la pantalla numerica
************************************************/

void Refresca(void) {
P1|=mil; /* coloco millares */
TRAN1=1; /* enciendo millares */
delay(10); /*espero un tiempo */
TRAN1=0; /* apago millares */
P1|=cen; /* coloco centenas */
TRAN2=1; /* enciendo centenas */
delay(10); /*espero un tiempo */
TRAN2=0; /* apago centenas */
P1|=dec; /* coloco decenas */
TRAN3=1; /* enciendo decenas */
delay(10); /*espero un tiempo */
TRAN3=0; /* apago decenas */
P1|=uni; /* coloco unidades */
TRAN4=1; /* enciendo unidades */
delay(10); /*espero un tiempo */
TRAN4=0; /* apago unidades */
}

/************************************************************
 Función Principal
************************************************************/
void main(void) {
unsigned int i;

P1=0x00;  /* Pone a cero 4511.Corte Tr1,2,3 y 4 */
uni=dec=cen=mil=0;  /* Pone a cero los registros de digito
*/
ALARMA=OFF;              /* Pone P2.4 a cero, alarma
desactivada */

while (1) {
     if (!CUENTA)   Cuenta(); /* incrementa */
     if (!DESCUENTA) Descuenta();  /* decrementa */
     if (!RESET)    {
               uni=dec=cen=mil=0; /* pone a cero */
               ALARMA=OFF;    /* desactiva alarma */
               }
     for(i=0;i<10;i++) Refresca();
 /* Repite para evitar rebote */
     }
}
```

# Control de un Ascensor



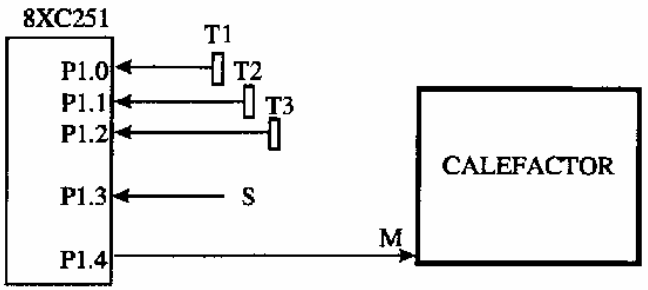



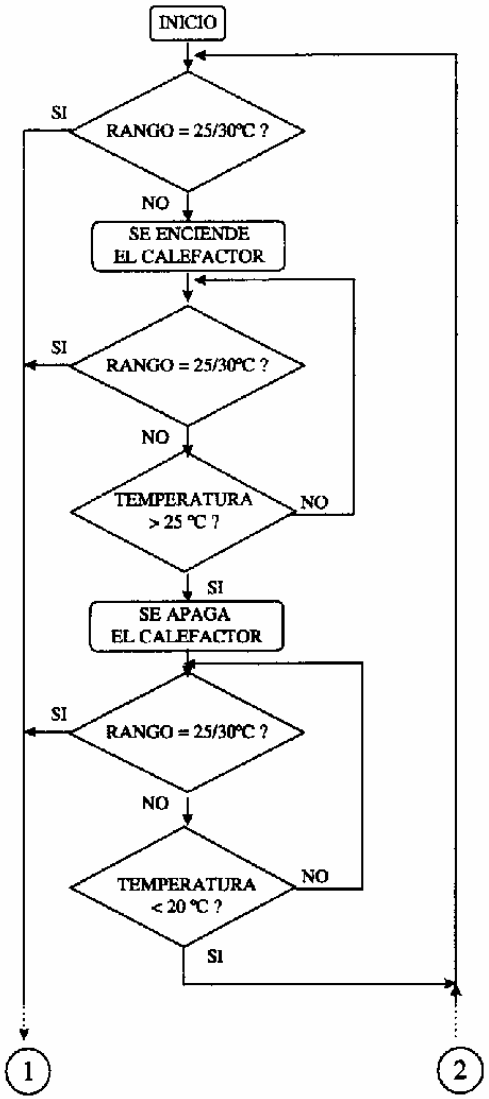

| M1 | M0 | Funcionamiento del motor |
|---|---|---|
| 0 | 0 | El motor está parado |
| 0 | 1 | El motor gira a la derecha y el ascensor sube |
| 1 | 0 | El motor gira a la izquierda y el ascensor baja |
| 1 | 1 | El motor está parado |

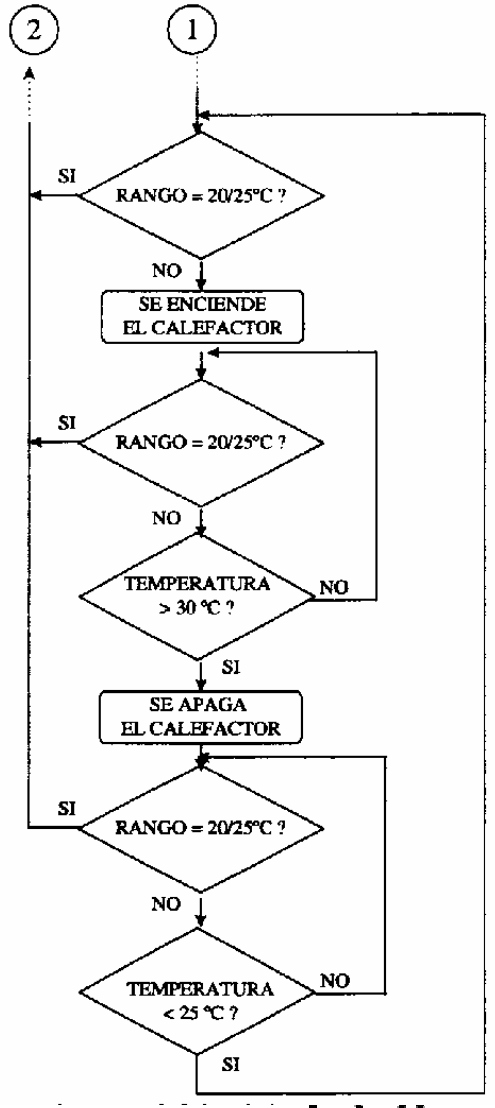
# Control de un Ascensor

```
#include <reg51.h>
/*******************************************
 Control del ascensor
*******************************************/
/* entradas sensores */
sbit VAGONETA=P1^1;
sbit SUPERFICIE=P1^2;
sbit SOTANO=P1^3;
/* defines de motores */
#define PARA 0x03
#define SUBE 0x01
#define BAJA 0x02

void main(void) {
P0|=PARA; /* Inicialización de puertos, motor detenido */
P1=0xFF; /* P1 entrada */
while (1) {
    while (!VAGONETA); /* espera vagoneta */
    P0|=SUBE; /* si hay vagoneta sube */
    while (!SUPERFICIE); /* espera llegar arriba */
    P0|=BAJA; /* para ascensor */
    while (!SOTANO); /* espera llegar abajo */
    P0|=PARA;
    While (VAGONETA); /* espera salida vagoneta */
    }
}
```

# Control de un calefactor



```
#include <reg51.h>
/*****************************************
 Control del calefactor
*****************************************/
#define OFF 0
#define ON 1
/* entradas */
sbit SEL=P1^3;
sbit ST1=P1^0;
sbit ST2=P1^1;
sbit ST3=P1^2;
/*salidas */
sbit CALEFACTOR=P1^4;


/* Funcion que devuelve temperatura aproximada */
unsigned char Temperatura(void) {
     if (ST3) return 31;
     if (ST2) return 26;
     if (ST1) return 21;
     return 19;
}


/* estructura para guardar max y min */
struct control {
     unsigned char min;
     unsigned char max;
     };
/* tabla de temperaturas */
struct control tabla_temp[2]={{20,25},{25,30}};


void main(void) {
unsigned char temp;

CALEFACTOR=ON;
while(1) {
  temp=Temperatura();
  if (temp>tabla_temp[SEL].max)    CALEFACTOR=OFF;
  else if (temp<tabla_temp[SEL].min) CALEFACTOR=ON;
  }
}
```

# Control de una cinta elevadora

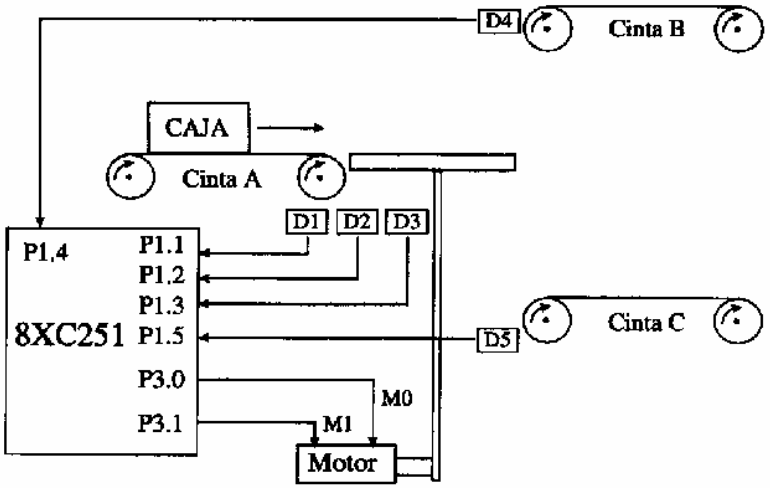
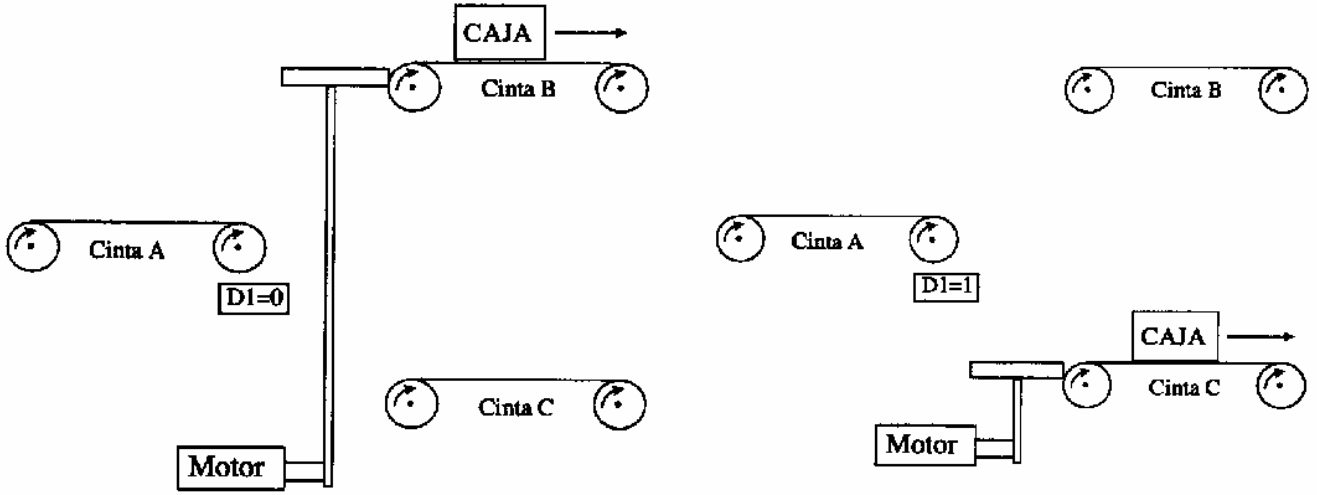
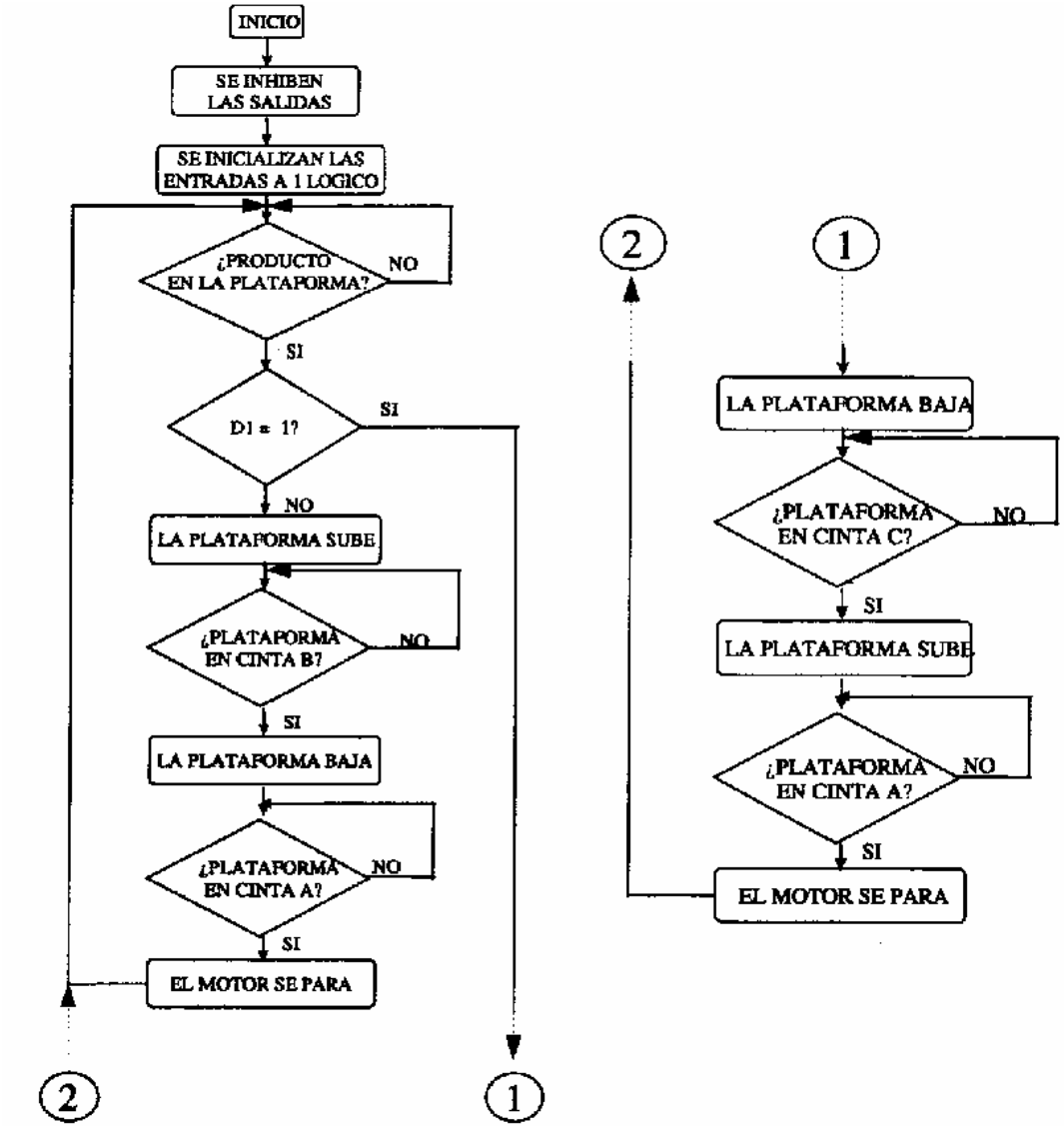



*Fig. 5.25 Diagrama del sistema de control*



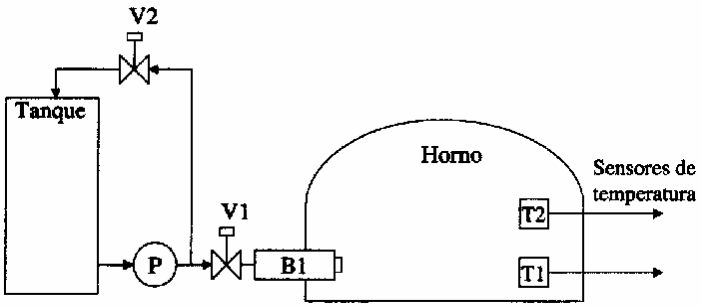
# Control de una cinta elevadora

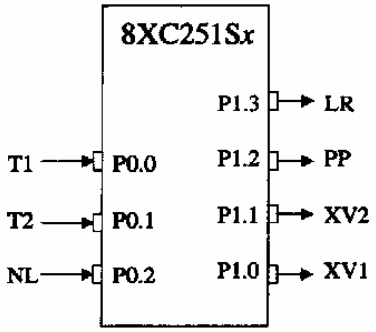


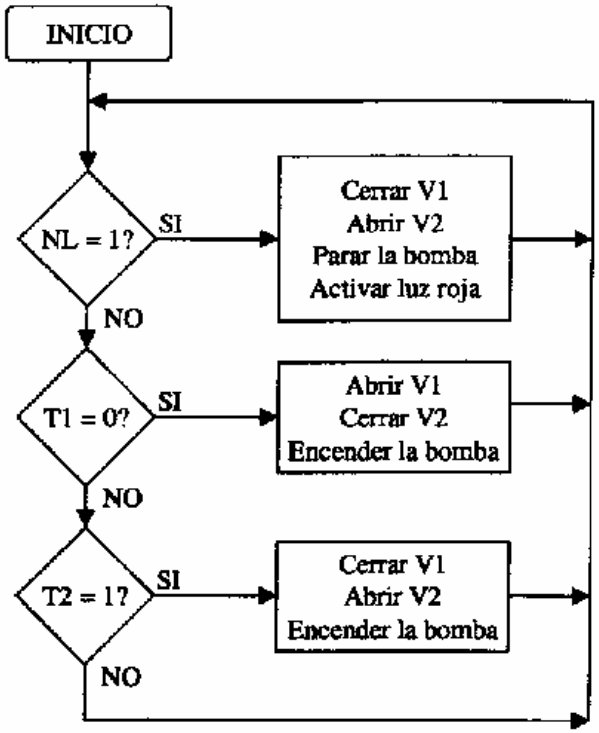
# Control de una cinta elevadora

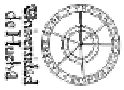
```
#include <reg51.h>
/* entradas */
sbit SD1=P1^1; sbit SD2=P1^2; sbit SD3=P1^3; sbit SD4=P1^4; sbit SD5=P1^5;
/* control del motor */
#define MOTOR P3
#define PARA 0x00
#define SUBE 0x01
#define BAJA 0x02
/*********************************************
 Control de la plataforma elevadora
*********************************************/
void main(void) {
/* inicialización de las entradas y salidas */
   MOTOR=PARA; P1=0xFF; /* todas entradas */
while (1) {
   while (!SD3); /* Se espera a que llegue producto a plataforma */
   if (SD1) { /* llevar a cinta C */
        MOTOR=BAJA;
        while (!SD4); /* Se espera plataforma llegue abajo */
        MOTOR=SUBE;
   } else { /* llevar a cinta B */
        MOTOR=SUBE;
        while (!SD5); /* Espera plataforma llegue arriba */
        MOTOR=BAJA;
   }
   while(!SD2); /* Se espera plataforma llegue cinta A */
   } }
```

# Control de temperatura de un horno





```
#include <reg51.h>

/* entradas */
sbit ST1=P0^0;
sbit ST2=P0^1;
sbit SNL=P0^2;
/* salidas */
sbit ALARMA=P1^3;
sbit BOMBA=P1^2;
sbit XV2=P1^1;
sbit XV1=P1^0;

/* defines */
#define ON 1
#define ABRE 1
#define OFF 0
#define CIERRA 0

/*******************************
 Control del horno
*******************************/
void main(void) {
P1=0x00; /* salidas a cero */
P0=0xFF; /* entradas a uno */
while (1) {
  if (SNL) { /* si disminuye nivel deposito */
     BOMBA=OFF;     /* para la bomba */
     XV1=CIERRA;    /* cierra electrovalvula 1 */
     XV2=ABRE;      /* abre electrovalvula 2 */
     ALARMA=ON;     /* enciende led fuera servicio*/
     }
  else if (ST1) { /* si baja de 275 grados */
     XV1=ABRE;      /* abre electrovalvula 1 */
     BOMBA=ON;      /* enciende bomba */
     XV2=CIERRA;    /* cierra electrovalvula 2 */
     ALARMA=OFF;    /* apaga led fuera servicio */
     }
  else if (ST2) { /* si sube de 300 grados */
     XV1=CIERRA;    /* cierra electrovalvula 1 */
     BOMBA=ON; /* enciende bomba */
     XV2=ABRE; /* abre electrovavula 2 */
     ALARMA=OFF;    /* apaga led fuera servicio */
     }
  }
}
```