UNIVERSIDAD DE HUELVA ESCUELA POLITÉCNICA SUPERIOR
ÁREA DE INGENIERÍA DE SISTEMAS ELECTRÓNICA DIGITAL
SOLUCIÓN AL EXAMEN SEGUNDO PARCIAL
31-MAY-2000 TIEMPO: Tres horas

P2. (4 puntos) Teniendo en cuenta la interfaz con el 8051, (a)diseñar la conexión de un registro de solo escritura y otro de solo lectura, ambos en la dirección 0FFF0H del mapa de memoria externa. Utilizar registros LS374, decodificador LS138 y puertas lógicas. No es necesario decodificar totalmente el bus de direcciones, la zona libre de memoria de datos externa (xdata) comienza en 8000H y termina en FFFFH.(b) Especificar razonadamente el mapa de memoria resultante tras la implementación anterior para la zona de datos externa. Señales:

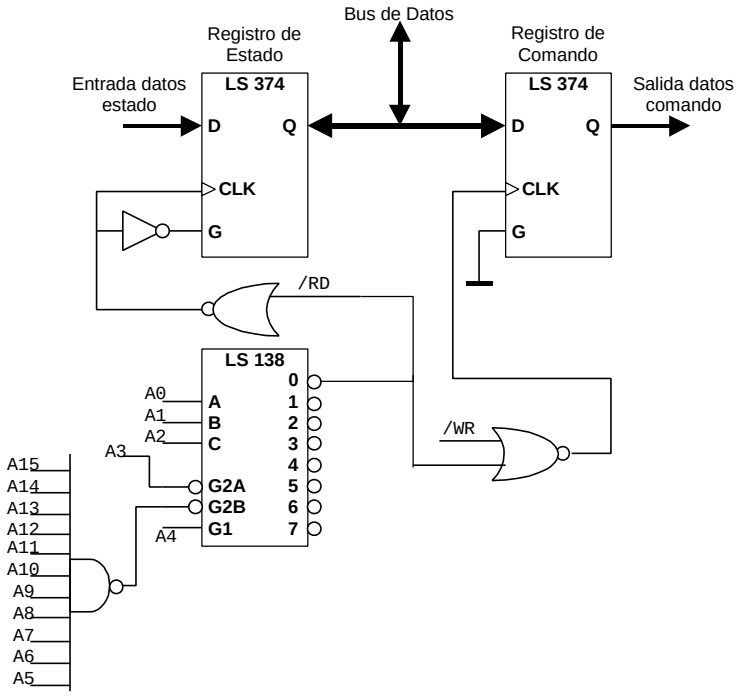
/WR:	Escritura
/RD:	Lectura
A15..A0:	Líneas 15 a 0 del bus de direcciones.
D7..D0:	Líneas del bus de datos.

SOLUCIÓN:

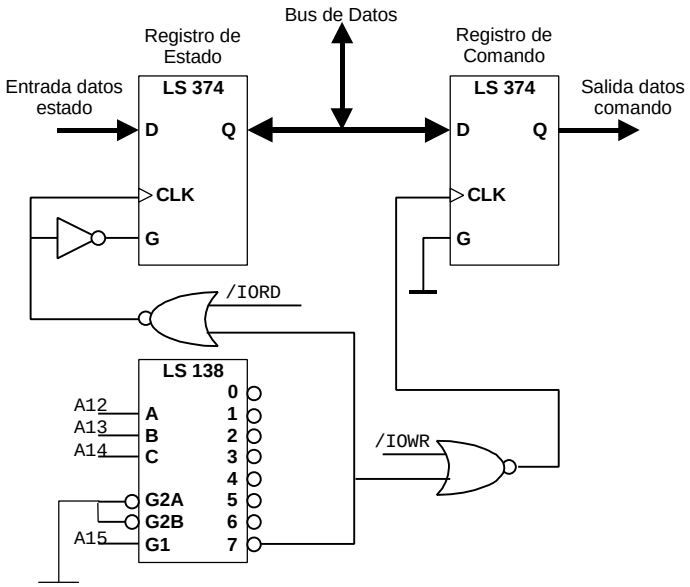
Debemos generar las señales CK y G de cada registro de ocho bits LS374. La señal CK es activa por flanco ascendente, luego los datos se guardarán es ese momento. El registro de comando será un registro de solo escritura y el de estado de solo lectura y ambos residirán en la misma posición de memoria de datos del 8051(xdata). Hay dos formas de realizar el decodificador de direcciones:

1.- Mediante decodificación completa (como se explica en los apuntes del 8051), se debe realizar un circuito combinacional que genere un nivel lógico 0 al direccionarse la palabra de memoria solicitada. El mapa de memoria solo tendrá una línea para nombrar ambos registros de solo lectura y de solo escritura.

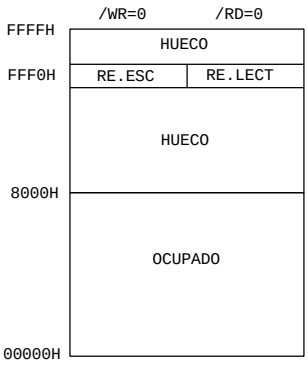
La decodificación se hará para la dirección de 16 bits FFF0H, o sea A15..A8 deben estar todos a 1 y A7..A2 todos a 0. Si empleamos puertas lógicas y el decodificador LS138, el esquema lógico quedaría de la siguiente forma:



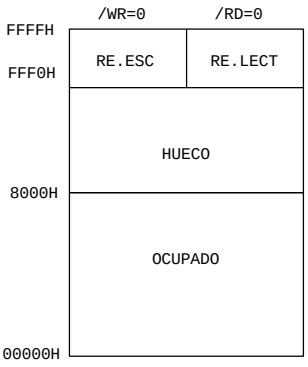
2.- Mediante decodificación incompleta (como se ha realizador en la práctica 2 de micro). Si se emplean solo los 4 bits más significativos del bus de direcciones, ambos registros se repiten para cada dirección de memoria desde la dirección F000H hasta la dirección FFFFH, no causando conflicto con la zona de memoria ocupada que se extiende desde 0H a 8000H.



Los mapas de memoria de ambas soluciones serian los siguientes:



DECODIFICACIÓN COMPLETA



DECODIFICACIÓN INCOMPLETA

P3. (3 puntos) Dado el siguiente código en lenguaje C de un sistema de control de ascensores en un edificio de 5 plantas donde XP4 es el sensor de piso, XP0 los pulsadores de cabina y XP2 los pulsadores de llamada, todos puertos externos de solo lectura.

```
#include <reg51.h>
sbit sensorpuerta, cerrada, puertaabre, puertaon, motorsube,
motoron, sobrepeso;
#define SI 1
#define NO 0
/* variables del programa */
char memoria[6]={NO, NO, NO, NO, NO, NO };
int pisodestino,pisoactual;
unsigned char sensor,cabina,llamada;
bit ocupado,bajando;
/* prototipos de varias funciones */
void retardo(int x);      /* retardo de x milisegundos */
```

```
void ascensor(bit sube); /* control del ascensor:
                                sube=0->baja, sube=1->sube*/
void para(void);          /* para el motor del ascensor */
int emergencia(int x);   /* suena x segundos */
unsigned int piso(unsigned char); /* devuelve el piso
seleccionado por el sensor o pulsadores */

void main(void) {
      ocupado=NO; pisodestino=-1;
      while(1) {
           sensor=XP4;
           while(sobrepeso) emergencia(5);
           pisoactual=piso(sensor);
           if (!ocupado) {
                cabina=XP0;
                if (cabina) {
                     pisodestino=piso(cabina); ocupado=SI;
      if(pisodestino<pisoactual) {bajando=SI;ascensor(NO);}
      if(pisodestino>pisoactual) {bajando=NO;ascensor(SI);}
                }
           for(i=5;i!=0;i--)
                if (memoria[pisoactual]) {
      if(pisodestino<pisoactual) {bajando=SI;ascensor(NO);}
      if(pisodestino>pisoactual) {bajando=NO;ascensor(SI);}
      break; }
           }
           else if(pisodestino==pisoactual) para();
           llamada=XP2;
           if (llamada) {
                memoria[piso(llamada)]=SI;
                if (!ocupado) {
                     pisodestino=piso(llamada);
                     ocupado=SI; memoria[pisodestino]=NO;
      if (pisodestino<pisoactual){bajando=SI;ascensor(NO);}
      if (pisodestino>pisoactual ){bajando=NO;ascensor(SI);}
                }
           }
           if ((memoria[pisoactual]) && bajando) {
                para();
                ascensor(NO); memoria[pisoactual]=NO;
           }
      }
}
```

a) Analizar el código fuente y explicar el funcionamiento del sistema, dar una especificación textual empleando “situaciones tipo” del sistema y/o diagramas de flujo.

b) Codificar en lenguaje C la rutina “emergencia” que devuelve un entero y toma un parámetro también entero. Su finalidad será la de activar una sirena de emergencia durante el número de segundos especificado. Esta alarma se encuentra conectada en el bit 5 del puerto 3. Si alguien pulsa el pulsador de apagar alarma, conectado al bit 4 del puerto 3, ésta dejara de sonar y la función devolverá 0, en caso contrario devolverá 1.

SOLUCION:

a) El funcionamiento es similar al comentado en clase, excepto que se añade un sensor de sobrepeso que dispara una alarma y además el bucle for que recorre la lista de peticiones guardadas no emplea el índice i, con lo cual no se atienden peticiones guardadas cuando no está ocupado el ascensor. Debería haber sido:

```
else for (i=5;i!=0;i--) if (memoria[i]) {
      if(pisodestino<pisoactual) {bajando=SI;ascensor(NO);}
      if(pisodestino>pisoactual) {bajando=NO;ascensor(SI);}
     break; }
```

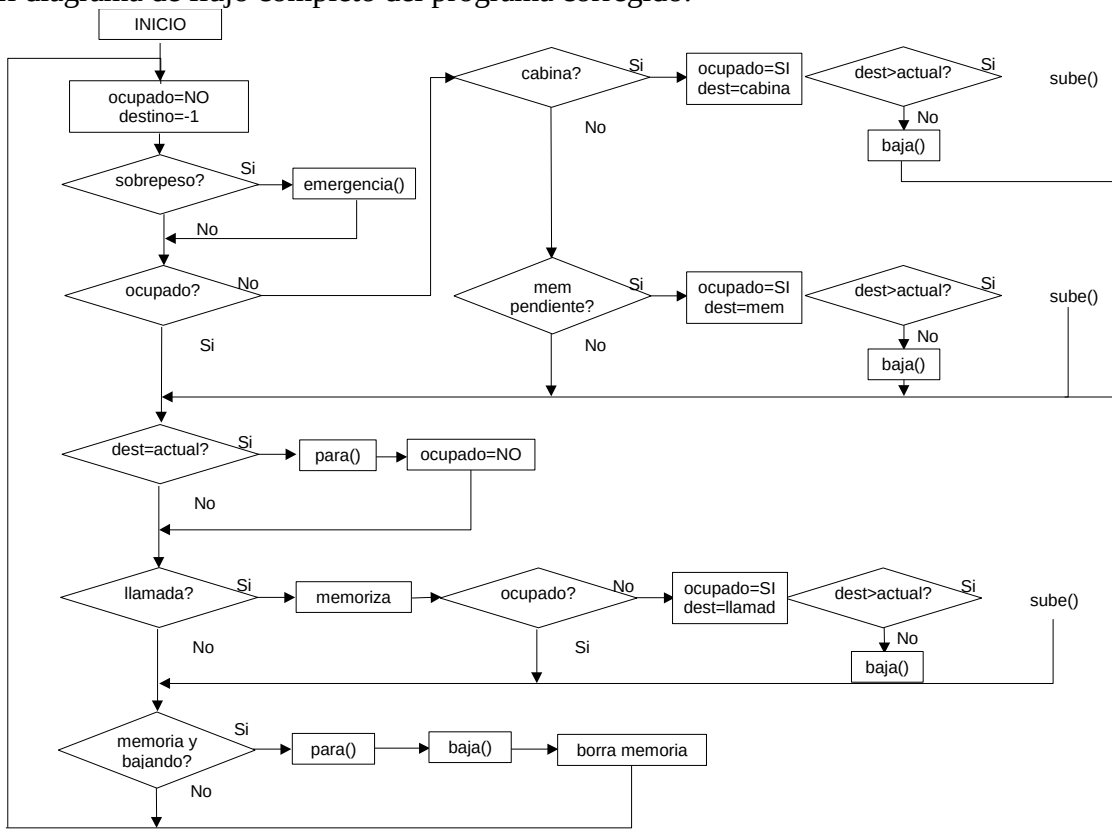
Tal como está el código, si estuviera ocupado y se pulsara la llamada al mismo tiempo que estuviera pasando por el piso en el que se hace la llamada se ejecutaría el cuerpo del if. En ese caso el ascensor seguiría bajando y no habría malfuncionamiento.

Sin embargo, en el caso en que no estuviera ocupado (solo al principio del programa, pues una vez comenzado el programa no hay ninguna línea que ponga la variable ocupado a NO) el ascensor puede quedar activado si coincide el piso actual con el de llamada. Esto sucede para un piso destino = -1, con lo cual el ascensor no pararía al llegar a la planta baja y se produciría malfuncionamiento.

Una vez que se haya producido alguna llamada el ascensor queda permanentemente en estado de ocupado, pues no se restablece el estado de no ocupado. Esto debería hacerse una vez llegado al piso destino, en:

```
if (pisodestino==pisoactual) {para();ocupado=NO;}
```

El diagrama de flujo completo del programa corregido:



b) La función realizada en lenguaje C toma un parámetro (número de segundos de retardo) y devuelve 0 si se paro la alarma con el pulsador o bien 1 si se llego a cumplir el tiempo completo. Empleamos un bucle for para atender al pulsador cada 10 mseg. El bucle externo while cuenta el número de segundos solicitado. La sentencia return termina la ejecución de la función y retorna inmediatamente al programa principal. La codificación de la subrutina es la siguiente:

```
/* definición de los puertos de E/S */
sbit pulsador=P3^4;
sbit alarma=P3^5;

int emergencia(int x) {
      int i;

      /* primero hace sonar la alarma */
      alarma=SI;
      while (x--) {
           /* retarda 1 seg, atiende pulsador cada 10mseg */
           for (i=0;i<100;i++) {
                    retardo(10);
                    if (pulsador) {alarma=NO;return(0);}
           }
      }
      return(1);
}
```