• Mínimo para pasar al oral: un problema • Nombre y CI en cada hoja • Numere las hojas, indique el total en la primera	• Utilice sólo un lado de las hojas • Incluya un solo problema por hoja • **Sea prolijo**

PROBLEMA 1

Se debe diseñar un controlador para manejar el encendido y el modo de funcionamiento de un dispositivo luminoso. El dispositivo debe tener tres modos de funcionamiento: modo OFF (luz apagada), modo FIJO (luz encendida fija) y modo BLINK (luz intermitente, tiempo **T1** encendido y **T1** apagado).

El dispositivo debe cambiar de modo en forma cíclica (OFF → FIJO → BLINK → OFF → ...) cada vez que recibe un flanco de bajada en su entrada **P,** y manejar la salida **ON** que enciende (**ON** = 1) o apaga (**ON** = 0) la luz.

Luego de un Reset del sistema el dispositivo debe iniciarse en modo OFF.

La duración del intervalo T1 se debe fijar durante la inicialización del sistema a partir del valor leído del puerto **PRE[3..0]** de 4 bits como se indica más abajo. El valor de **PRE[3..0]** se mantiene estable después de un reset solo durante el tiempo suficiente para poder ser leído en la inicialización; los cambios posteriores deben ser ignorados.

Se desea incorporar la funcionalidad del controlador descripto a un **sistema T80** existente. Se utilizará un bloque Timer para medir el tiempo **T1** y los controladores de interrupciones que sean necesarios. Se trabajará por interrupciones, una asociada al flanco de bajada de **P** y otra al bloque Timer.

Cuando se ingresa al modo BLINK debe apagarse la luz inmediatamente y arrancar el Timer para comenzar a medir el primer intervalo **T1**.

Para evitar interrupciones innecesarias en modo OFF y FIJO el timer debe reprogramarse en forma adecuada.

El valor de **T1** debe fijarse en (1 / fclk) x 250 x $2^{\mathbf{PRE}}$.

Se pide:

a) Hardware del sistema. La memoria se supondrá implementada con 32KB de RAM y 32KB de ROM.

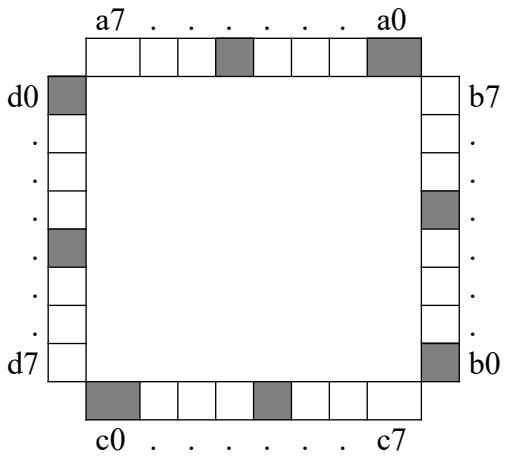
b) Rutinas de atención a interrupciones de Timer y de flanco de bajada de **P**.

c) Inicialización del sistema. Debe inicializar todo lo necesario para el funcionamiento de las funcionalidades descritas. Además debe invocar a la subrutina **init_otros** y terminar saltando a la dirección **prog_ppal**. Se debe especificar la ubicación en memoria de todo el código, tablas, variables y constantes que se utilicen mediante directivas al ensamblador.

PROBLEMA 2

La marquesina de un cartel cuadrado tiene 32 lámparas, 8 en cada uno de los lados del cartel como se indica en la figura. Cada lámpara debe comandarse con una señal independiente (1 = on, 0 = off).

Se desea diseñar un dispositivo que maneje la marquesina del cartel encendiendo las lámparas con un patrón que periódicamente se desplace creando un efecto visual de rotación en sentido horario en la marquesina. Se utiliza el patrón predeterminado de 8 bits de la figura (on-on-on-off-on-on-on-off). La rotación se realiza a razón de un desplazamiento cada 300 ms. Un pulsador **pausa** se utiliza para detener o volver a arrancar la rotación. Cuando llega un pulso a cero en **pausa** la rotación debe detenerse si estaba girando y volver a girar si estaba detenida. Para medir el tiempo se dispone de una onda cuadrada de período 50 ms en la señal **girar**.



La señal **girar** debe generar una interrupción en modo 1 que se encargará de rotar el patrón en la marquesina cuando corresponda. Esta será la única interrupción del sistema.

El programa principal debe consistir en un loop infinito que atienda a la señal pausa y tome las acciones correspondientes. En cada vuelta del loop además debe invocarse la subrutina **atiendo_otros** que se encarga de otras tareas.

Obsérvese que con el patrón indicado más arriba las lámparas se manejan de forma que siempre son a[7..0] = b[7..0] = c[7..0] = d[7..0]. Sin embargo, el hardware debe manejar de manera independiente a los 4 lados de la marquesina por medio de 4 puertos de 8 bits para permitir que una futura versión del software pueda manejar la marquesina en forma diferente.

Se pide:

a) El hardware para manejar la marquesina, incluyendo todos los puertos que sean necesarios con su correspondiente decodificación y la lógica de interrupciones. Se reservó a esos efectos el rango desde la dirección 0x80 a 0xBF del espacio de E/S. Se debe equipar al sistema con 16KB de RAM y 16KB de ROM mapeados en la mitad más baja del espacio de memoria, quedando el resto reservando para futuras ampliaciones.
b) La rutina de atención a interrupción producida por **girar**.
c) Inicialización del sistema con todo lo necesario para el buen funcionamiento de la marquesina luego del reset y el loop del programa principal. Luego de Reset la marquesina debe comenzar girando. Se supondrá implementada la subrutina **atiendo_otros** mencionada más arriba. Se debe especificar la ubicación en memoria de todo el código, tablas, variables y constantes que se utilicen mediante directivas al ensamblador.