FACULTAD DE INGENIERIA INTRODUCCION A LOS MICROPROCESADORES
UNIVERSIDAD DE LA REPUBLICA DICIEMBRE 2012

• 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** • **Aprobación:** mínimo UN problema

PROBLEMA 1

Una oficina en la que se atiende público debe equiparse con un sistema LLAMADOR DE PÚBLICO para ir atendiendo a los clientes por orden de llegada. Al ingresar, cada cliente retira un número de un dispensador y luego es llamado por uno cualquiera de dos puestos de atención disponibles.

Los puestos están numerados de 01 a 02 y cada uno está equipado con un pulsador. Cuando el encargado de un puesto quiere llamar al siguiente cliente presiona el pulsador. Como resultado de esto un sistema centralizado debe desplegar en un panel el número del siguiente cliente a ser atendido (asignados cíclicamente desde 00 hasta 99) y el número del puesto desde el cual fue llamado.

Numero	Puesto
99	1

La información de número y puesto debe estar visible al menos durante aproximadamente 5 segundos. En caso de que se presione uno de los pulsadores cuando aún no ha transcurrido el tiempo, el pedido deberá guardarse en una cola y mostrarse luego que hayan transcurrido los 5 segundos.

Para implementar el llamador de público se utilizará un sistema basado en un microprocesador Z80. El programa principal será el encargado de inicializar todo luego de un reset, monitorear el estado de los pulsadores y actualiza los display o encolar el pedido según corresponda.

Para almacenar los pedidos se cuenta con la rutina *put_fifo* que espera en el acumulador el número de puesto a guardar y *get_fifo* que devuelve en el acumulador el último puesto guardado o FFh si no hay datos guardados en el fifo.

El conteo de tiempo desplegado del número, deberá realizarse utilizando una señal *tic* que generará una interrupción cada 1 segundo. Luego de transcurridas 5 interrupciones, se actualizan los displays con posibles pedidos encolados.

Se pide:

a) Hardware completo del sistema incluyendo memoria, puerto para interrupciones, puertos para display y puertos de entrada que permitan capturar los flancos de subida que producen los pulsadores al ser presionados. Se dispone de displays 7 segmentos.

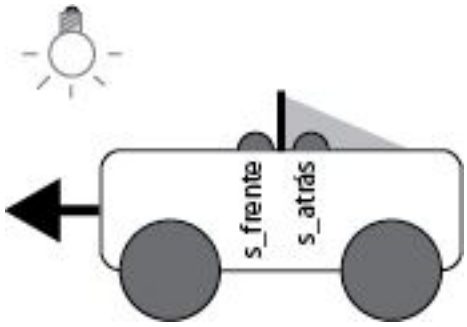
b) Reserva de memoria, inicialización luego de un reset y programa principal. Se dispone de la rutina *bin27seg* que recibe en el acumulador un número entre 0 y 99 y devuelve en B y C los dígitos para desplegar en 2 display 7 segmentos, (decenas y unidades respectivamente).

c) Rutina de atención a interrupciones.

PROBLEMA 2

Se debe controlar la posición de un carrito movido por un motor paso a paso de manera que se mueva sobre una trayectoria rectilínea siguiendo la posición de una fuente de luz, que se mueve en una trayectoria paralela por encima del carrito.

Para eso en el techo del carrito hay dos fotosensores **s_frente** y **s_atras** separados por una pequeña pantalla. Si la luz llega desde adelante, la pantalla hace sombra sobre el sensor trasero y el valor sensado por s_frente será mayor que el sensado por s_atras. El carro en este caso debe avanzar. Al revés, si la lectura del s_atras es mayor que la de s_frente entonces la luz está detrás y el carro debe retroceder. Si ambas medidas son similares la luz está sobre el carro y el motor debe permanecer detenido.



El motor se mueve de a pasos escribiendo en sus 4 entradas (A, B, C y D) la secuencia de valores mostrada en la tabla en el orden indicado (0, 1, ...3, 0, 1) para mover el carro hacia adelante, y en el orden inverso para mover el carro hacia atrás. Cada nuevo valor que se escribe hace moverse al motor 1 paso.
Para dejar detenido el motor en cualquier posición simplemente se mantiene el último valor.

Paso	Salidas (A,B,C,D)
0	1 1 0 0
1	0 1 1 0
2	0 0 1 1
3	1 0 0 1

Las salidas de los sensores son de 8 bits y los valores sensados son positivos menores que 128. Si llamamos **error** = (s_frente - s_atras), el manejo del motor se hace de acuerdo a lo siguiente:

- Si |error| < UMBRAL el motor se mantiene detenido.
- Si |error| >= UMBRAL el motor debe avanzar o retroceder según el signo del error.
- En todos los casos las salidas deben actualizarse cada T = 1/4MHz * 256 * (255 - |error|).

El tiempo **T** deberá generarse mediante interrupciones utilizando un CTC en modo 2. El reloj del sistema es de de 4MHz.

Se utilizará una variable **POSICION** de 16 bits con signo para llevar la cuenta de la posición actual del carrito, incrementando o decrementando 1 paso cada vez que se avanza o retrocede el motor. El valor 0 corresponde a la posición del carro al momento del reset del sistema.

Diseñar el sistema completo utilizando un procesador Z80.

a) Hardware completo. Los puertos a utilizar deben mapearse dentro del rango de direcciones 0x40-0x7F ya que el resto del espacio de E/S está utilizado por otros dispositivos.

b) Programa principal que monitoree los sensores y determine el período de cambio de los pasos y el movimiento a realizar (detenido, avance o retroceso). Entre lecturas consecutivas de los sensores debe invocar la subrutina atiendo_otros() para manejar otros puertos no descriptos aquí.

c) Rutina de atención a interrupción generada por el CTC que maneje el motor de acuerdo a lo determinado por el programa principal, actualice POSICION y reprograme el timer con el valor calculado por el programa principal.

d) Inicialización del sistema y directivas de reserva de memoria. Se debe invocar a la subrutina init_otros() para inicializar otras funciones y puertos. Los primeros 2 vectores de la tabla de interrupciones son utilizados por otros periféricos.