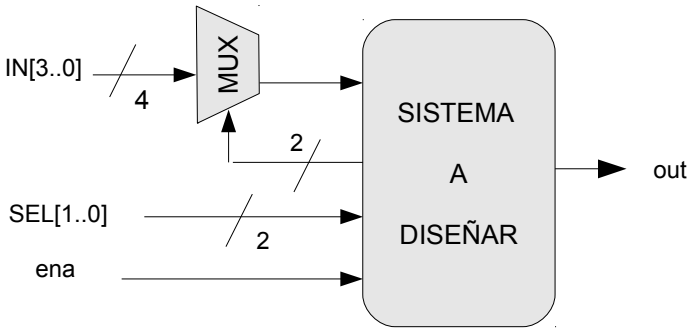
- Nombre y CI en cada hoja - Numere las hojas - Indique el total de hojas en la primera - Utilice solo un lado de las hojas	- Incluya un solo problema por hoja - **Sea prolijo** - **Aprobación:** mínimo UN problema

PROBLEMA 1



Se quiere diseñar el sistema de la figura con un microprocesador Z80.

Mientras está habilitado (**ena** = 1) el sistema debe copiar a la salida **out** los pulsos provenientes de la entrada **IN[3..0]** seleccionada por la entrada **SEL[1..0]**.

Para seleccionar la entrada IN[] adecuada se utlizará un multiplexor 4 a 1 controlado por un puerto de salida.

Nunca se deben copiar pulsos incompletos, o dicho de otra forma el pulso en la salida debe tener la misma duración que en la entrada seleccionada. Por lo tanto:

- Si cuando **ena** sube la entrada seleccionada vale uno se ignora ese pulso.
- Si cuando **ena** baja la salida vale uno, se completa el pulso.
- Si cuando cambia **SEL** la salida estaba en uno se completa el pulso antes de cambiar la entrada seleccionada.
- Si al cambiar la entrada seleccionada del multiplexor la nueva entrada vale uno, se ignora ese pulso.

El manejo de la salida **out** se hará por interrupciones trabajando en modo 1, que deben generarse tanto con el flanco de subida como con el flanco de bajada de la señal en la salida del multiplexor.

El programa principal es un loop infinito como se indica en el recuadro.

La subrutina **manejo_mux** deberá verificar si desde la vez anterior que dicha subrutina fue invocada **SEL** cambió, y en caso afirmativo manejar el multiplexor teniendo la precaución de no violar las condiciones listadas arriba.

```
loop:
  call atiendo_otros
  call manejo_mux
  jp loop
```

Se pide:

a) Hardware, incluyendo 32K de ROM, 32K de RAM y todos los puertos necesarios. Se puede disponer del rango 40h-7Fh del espacio de E/S, dejando el resto libre para otras funcionalidades.

b) Subrutina de atención a interrupciones.

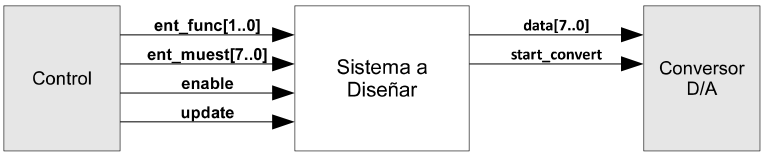
c) Subrutina **manejo_mux**

d) Inicialización, definición de constantes y directivas de reserva de memoria. En la inicialización se debe invocar a la subrutina inicializo_otros que se supondrá implementada.

Nota: se puede suponer que los cambios en **ena** y **SEL[]** son lentos comparados con la duración de los pulsos en las señales **IN[]**, y que la duración de estos pulsos es mucho mayor que el período de reloj del sistema.

PROBLEMA 2

Se desea agregar a un sistema existente la función de generar señales de distinto tipo y de frecuencia variable, mediante muestras de 8 bits de las señales que se quieren generar. Para lograr esto se almacenan las muestras de dichas señales en tablas en ROM, las cuales son leídas y convertidas a una tensión analógica por un conversor D/A.



Las señales que se tienen almacenadas en tablas en ROM son: cuadrada, sinusoidal, triangular y diente de sierra. Las direcciones de comienzo de las tablas son: **tlb_cuad , tlb_sin**, **tbl_triang** y **tbl_sierra**, y cada una de ellas contiene 256 muestras, formando un período completo de cada señal.

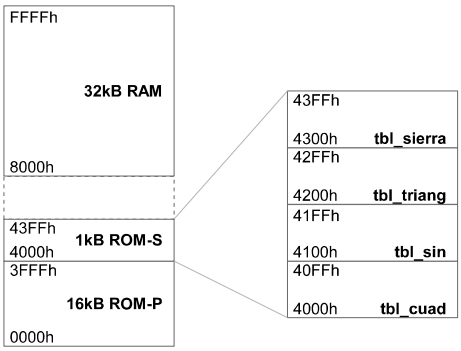
Para generar la señal seleccionada, se debe leer desde la tabla correspondiente una muestra cada X ms, y enviarla al conversor D/A. El conversor recibe las muestras en el puerto **data** de 8 bits y comienza a convertir con un flanco de subida en **start_convert**. Luego del flanco no es necesario mantener el valor en el puerto **data**. Cuando se completa un período se recomienza desde el inicio de la tabla.

La señal que se debe generar y el tiempo X entre muestras (en múltiplos de 1ms) están disponibles en los puertos **ent_func** y **ent_muest** de 2 y 8 bits respectivamente. Las posibles señales a generar se codifican en **ent_func** como: 00- cuadrada, 01- sinusoidal, 10- triangular, 11- diente de sierra. El sistema cuenta además con una entrada **enable** que indica si se debe generar o no la señal en la salida **data**.

Cada vez que el sistema recibe un flanco en la entrada **update,** debe leer ent_func, ent_muest y enable. Si **enable** está en 0 la salida **data** debe valer 0 hasta el próximo flanco de update, mientras que si **enable** vale 1 se debe generar señal de acuerdo a la tabla seleccionada por ent_func. Todos los cambios (habilitación/deshabilitación, señal y frecuencia) se hacen inmediatamente luego de recibir la orden de update, no se espera el envío de todas las muestras de la señal anterior.

Para enviar las muestras cada X ms se deberán utilizar interrupciones generadas por un CTC trabajando en modo 2.

Además de la función de generar señales, el sistema realiza otras funciones que utilizan interrupciones y puertos de E/S. Debido a esto están ocupados por otros dispositivos los cuatro primeros lugares de la tabla de interrupciones y los primeros 64 lugares del espacio de entrada salida.



El sistema a implementar debe tener el mapa de memoria de la figura.

Se pide:

a) Hardware completo del sistema.
b) Inicialización del sistema y programa principal que monitorea la señal **update**. Se dispone de las subrutinas **inicializo_otros** y **atiendo_otros** que deben invocarse respectivamente en la inicialización y en el loop del programa principal para atender funcionalidades no descriptas aquí.
c) Rutina de atención a las interrupciones que envía muestras al conversor A/D.

Notas:

- Los posibles valores en **ent_muest** están entre 1 y 255.
- El sistema funciona con un reloj de 4MHz, notar que 1ms = (1 / 4MHz)*16*250.