- 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 desea realizar un sistema basado en un Z80 que ejecute un programa actualizable mediante el programador que se muestra en la figura.
El sistema debe contar con el mapa de memoria indicado en el recuadro. La función de cada chip de memoria se describe a continuación:

- **ROM:** Inicialización del sistema y rutina de atención a la interrupción para atender al dispositivo programador
- **RAM0:** Memoria que almacena las variables utilizadas por el programa residente en ROM.
- **RAM1:** Memoria utilizada para almacenar temporalmente el programa que aún no ha terminado de recibirse desde el programador.
- **RAM2:** Memoria RAM no volátil donde correrá el programa cargado por el usuario.
- **RAM3:** Memoria RAM utilizada por el programa cargado por el usuario y donde se ubica el stack del sistema.

Mapa de memoria

B000h	RAM 3
8000h	RAM 2
4000h	RAM 1
2000h	RAM 0
0000h	ROM

El funcionamiento es el siguiente:
Luego de un reset el sistema deberá inicializar el puntero al stack (SP) y lo que sea necesario para que funcione el mecanismo de carga de programas de usuario que se describe más adelante. A continuación se sensará la entrada **blank**, en caso que dicha entrada esté en nivel alto se debe pasar a ejecutar la última versión del programa cargado en la **RAM2**. Si bien esta es una memoria RAM, estará alimentada con una batería que permitirá que el programa no se borre entre encendidos del sistema. Si la entrada **blank** está en nivel bajo, antes de pasar a ejecutar el programa en RAM2 se deberá escribir en el comienzo de dicha memoria el código correspondiente a la siguiente instrucción:

```
LOOP:  JR LOOP
```

Esto permite arrancar el sistema en forma segura si el contenido de la RAM2 es desconocido, pudiendo a continuación cargar un nuevo programa y pasar a ejecutarlo.

Las líneas **stb** y **prg** son utilizadas por el programador para enviar al sistema los bytes de programa e indicar que se han transferido todos los bytes, debiendo pasarse a ejecutar el programa recién enviado.
Las transacciones entre el programador y el sistema se producen cada vez que ocurre un flanco de subida en **stb** y solo en ese instante **bin** y **prg** tienen un valor válido.
Si en el flanco de **stb, prg**=1 quiere decir que hay un nuevo byte de programa en las líneas **bin**. Si en el flanco de **stb**, **prg**=0 quiere decir que se ha finalizado la transferencia y se debe sustituir el programa cargado en RAM2 por la nueva versión. *Hasta que no se termine de transmitir el nuevo programa, se deberá seguir ejecutando el anterior.*
La salida **load** debe valer “1” mientras hay una transferencia pendiente de ser completada y “0” el resto del tiempo.

Los flancos de **stb** deben producir interrupciones NO enmascarables y cada nuevo byte debe ser colocado en direcciones consecutivas a partir del comienzo de la memoria **RAM1**. Cuando se termine de cargar el código la rutina de interrupción deberá copiarlo a la **RAM2** y saltar a la dirección 0 para que se ejecute de nuevo el código de inicialización, quedando pronto para comenzar a recibir otra actualización.
Se pide:

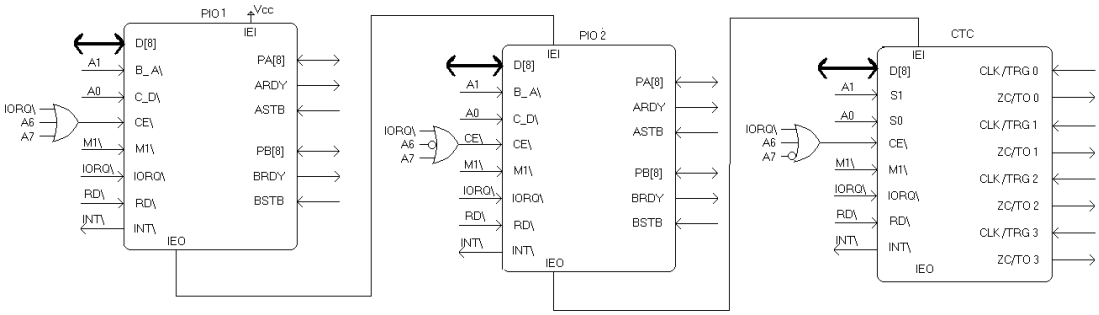
a) Hardware del sistema. Quedan libres los puertos de entrada y salida entre las direcciones 10h a 1Fh. Se dispone de chips de memoria de 8KB y 16KB.

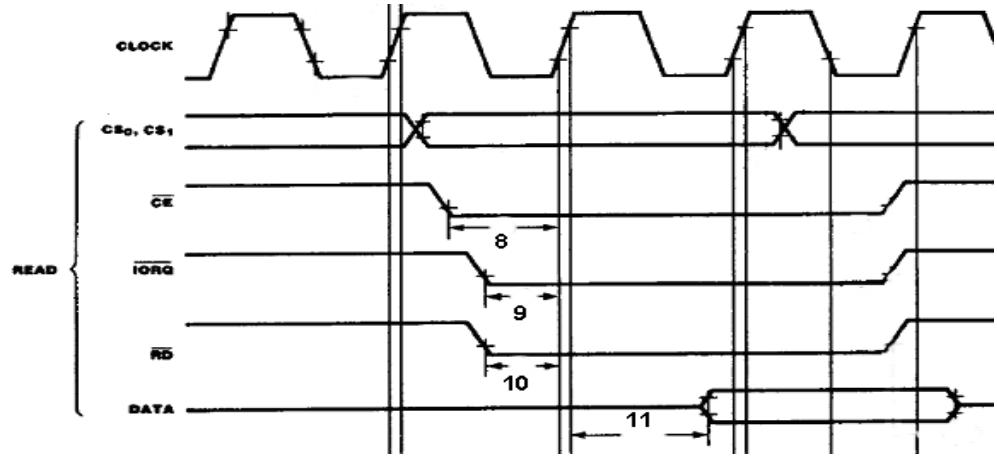
b) Software completo del sistema: inicialización y rutina de atención a la interrupción.

Problema 2

Parte a)

Se dispone de una placa de propósito general con un procesador Z80, 32Kb de ROM, 32Kb de RAM, dos PIO y un CTC, decodificados como se muestra en la figura.





8	TsCE(C)	/CE to Clock Rise Setup Time	ns mín.
9	TsIO(C)	/IORQ Fall to Clock Rise Setup Time	ns mín.
10	TsRD(C)	/RD Fall to Clock Rise Setup Time	ns mín.
11	TdC(DO)	Clock Rise to Data Out Delay	ns máx.

La figura reproduce parcialmente el ciclo de lectura del CTC especificado en su hoja de datos.
Se pide:

Plantear las inecuaciones que deben cumplirse para el buen funcionamiento de la conexión entre el CTC y el Z80 que involucren a los parámetros de tiempo del CTC que se indican en la figura. Plantearlas en función de los retardos de compuertas y de los parámetros del Z80 y del CTC, indicando el peor caso para cada parámetro. Cuando aparezcan ciclos T indicar a cual período (T1, T2, TWA, etc.) corresponden.

Parte b)

La placa de la parte (a) se utiliza para controlar un sistema electromecánico.
Durante el diseño se detectaron problemas causados por rebotes en la señal **s_cruda** de un bit. Por ese motivo se decide generar una nueva señal **s_limpia** a partir de la señal original de la siguiente forma:

- Se muestrea la entrada **s_cruda** cada 1ms.
- En cada instante de muestreo el valor leído se escribe en la salida **s_limpia** solamente si coincide con las dos lecturas anteriores. En caso contrario no se modifica la salida s_limpia.

Para implementar esta parte del sistema se dispone del puerto B de PIO1 y del canal 0 del CTC. Este último se utilizará para generar una interrupción periódica que se supondrá que es la única interrupción.

Se pide:

i. *Rutina de atención a la interrupción que realice la función descrita y su ubicación en memoria. Indicar claramente a qué bits del puerto B se conectan las señales* **s_cruda** y **s_limpia**
ii. *Reserva de memoria para las variables utilizadas.*
iii. *Suponer cargado al comienzo de la ROM el código del recuadro. Escribir la subrutina INI_LIMPIAR que haga las inicializaciones necesarias (periféricos, variables, interrupciones, etc.) para el buen funcionamiento de la eliminación de rebotes (nota: fck = 4 MHz, 1ms = 4000 x T, 4000 = 16 x 250).*

```
ORG 0000H

LD SP, 0
CALL INI_LIMPIAR
CALL INI_OTROS
JP PROG_PPAL
```