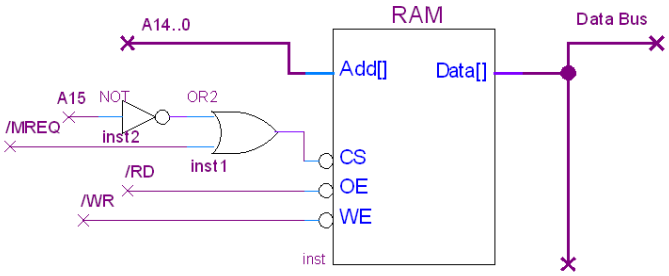
a) Utilice solo un lado de las hojas
b) Incluya un solo problema por hoja.
c) Sea prolijo.
d) Indique Nombre, CI y numere cada hoja.

Ejercicio 1 (18 pts.)

Durante un ciclo de escritura, un chip de memoria FRAM (Ferroelectric RAM) exige un tiempo de setup T_{AS} de las direcciones respecto al flanco de bajada de la última en bajar de /CS y /WE; y un tiempo de setup T_{DS} de los datos respecto al flanco de subida de la primera en subir de /CS y /WE.



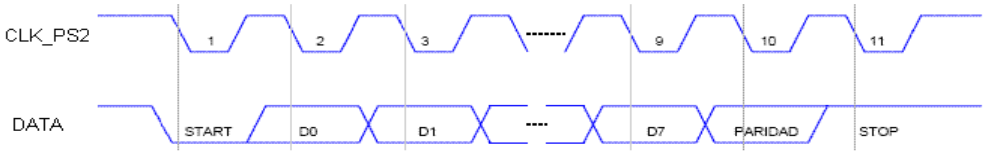
Se pide:

a) En función de la cantidad N de tiempos de espera insertados, los parámetros de tiempos del microprocesador Z80, del inversor ($T_{NOT\_min} < T_{NOT} < T_{NOT\_max}$) y de la compuerta OR ($T_{OR\_min} < T_{OR} < T_{OR\_max}$), escribir todas las condiciones necesarias para que en el ciclo de escritura en memoria se respeten los tiempos T_{AS} y T_{DS} requeridos por la memoria. Indicar para cada parámetro, cuando corresponda, si debe considerarse el valor máximo o el mínimo. Los períodos o semiperíodos de reloj T que aparezcan en las ecuaciones se deben anotar indicando a qué período corresponden dentro del ciclo (p. ej. T1, $T2_{LOW}$, etc.).

b) Determinar qué requerimientos se debe imponer a T_{AS} y T_{DS} para que el sistema funcione correctamente, si N = 0, T_{OR_min} =0, T_{OR_max} =12ns y se está utilizando un Z84C00~~06~~20 con un reloj de ciclo de trabajo 50% (T_{HIGH}=T_{LOW}) y frecuencia fclk= 6MHz~~20MHz~~.

Ejercicio 2 (26 pts.)

Se quiere equipar a un sistema Z80 tradicional con buses triestado para que sea capaz de recibir datos por una interfaz serie PS2 como la utilizada en el laboratorio, pero trabajando con interrupciones en **modo 1**:



Cada flanco de bajada de la señal **CLK_PS2** debe generar una interrupción. La rutina de servicio de interrupciones debe procesar cada bit recibido. En la interrupción correspondiente al bit de paridad se deberá:

- escribir el dato completo recibido, D[7..0] en la dirección **DATO** reservada en memoria.
- Escribir el bit de paridad recibido en el bit menos significativo de la dirección reservada **PARIDAD**, los demás bits deben completarse con 0.
- finalmente debe escribir el valor **0xFF** en la dirección reservada **HAY_DATO**.

Los datos correspondientes al bit de **START** y el de **STOP** deben descartarse.

Luego de la inicialización del sistema, pasa a ejecutarse el loop del programa principal mostrado en el siguiente recuadro:

```
loop:
  call atiendo_otros
  ld A, 0xFF
  cp A, (HAY_DATO)
  jr nz, loop
  call proceso_dato
  ld a, 0
  ld (HAY_DATO), a
  jr loop
```

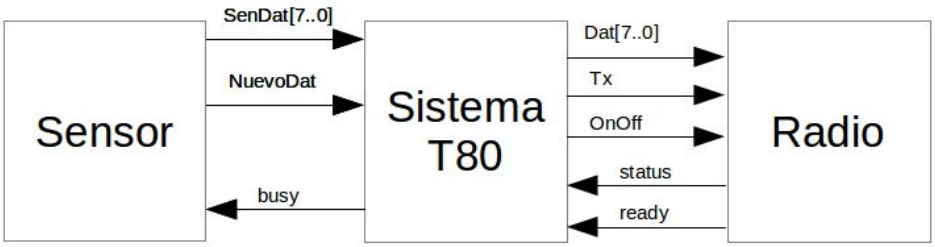
Se pide:

a) Diseñar el hardware de puertos e interrupción asociados a la interfaz PS2. Se puede utilizar para ello el rango de direcciones **0x40** a **0x7F** de E/S.

b) La rutina de atención a la interrupción en **modo 1**.

c) La inicialización del sistema y las directivas de reserva de memoria para las variables utilizadas. El sistema tiene **32KB** de **ROM** y **32KB** de **RAM**. Se deberá incluir la invocación a una subrutina **init_otros** y terminar con un salto al loop del programa principal.

Ejercicio 3 (56 pts.)



Se desea enviar datos generados por un sensor a través de una radio. Para ello se deberá diseñar el siguiente sistema.

El sensor de la figura genera datos de 8 bits a una tasa variable. Cuando tiene un dato nuevo espera a que su entrada **Busy** esté en nivel bajo, coloca el dato en su salida **SenDat** y genera un flanco de subida en su salida **NuevoDat** indicando que el dato en su salida es válido. La señal **Busy** debe subir a 1 cuando se detecta un flanco de subida en la señal **NuevoDat** y debe bajar a 0 cuando el sistema lee el nuevo dato.

La entrada **OnOff** enciende a la radio cuando está en 1 y la apaga cuando está en 0. Si está encendida, cuando se genera un flanco de subida en su entrada **Tx**, esta envía el dato que se encuentra en su entrada **Dat**. Luego, al finalizar la transmisión genera un pulso a 1 de corta duración en su salida **ready** e indica en su salida **status** si la transmisión se realizó con éxito (**status**=1) o si ha fallado (**status**=0).

Los datos generados por el sensor deben ser almacenados temporalmente en una cola implementada por software. Para ello se cuenta con dos subrutinas, **PutCola** que recibe en el acumulador el dato a agregar en la cola y **GetCola** que si la cola no está vacía extrae un dato de la cola y lo devuelve en el acumulador con el flag Z apagado, y en caso contrario devuelve el flag Z activado. **PutCola** y **GetCola** se suponen implementadas.

La transmisión de datos se debe realizar periódicamente a intervalos **Tesp** de 5 minutos. Cuando finaliza el tiempo **Tesp**, si hay datos en la cola se debe prender la radio y comenzar a enviar datos hasta que la cola se encuentre vacía. En caso contrario no se realizará niguna acción hasta el próximo fin de intervalo **Tesp.** Cada dato se transmitirá manejando las entradas **Dat** y **Tx** de la radio ya descriptas. Cuando llega el pulso de **ready** se debe leer el estado de la transmisión en la salida **status** de la radio. Si la transmisión fue exitosa se debe transmitir el siguiente dato de la cola, de lo contrario se debe realizar una retransmisión del último dato enviado. Una vez que se vacía la cola se debe apagar la radio y esperar a que finalice el tiempo **Tesp** para comenzar con una nueva transmisión. Por último, si al culminar el tiempo **Tesp**, la cola se encuentra vacía, se deberá contar un nuevo tiempo **Tesp** antes de comenzar la transmisión.

Se debe diseñar un sistema con microprocesador T80 y buses multiplexados que implemente el funcionamiento descripto. Se trabajará por interrupciones en modo 2 para implementar la transmisión de datos mientras que el programa principal será el encargado de recibir los datos del sensor y agregarlos en la cola.

Se cuenta con un bloque Timer para medir el tiempo Tesp y se debe generar interrupciones con un flanco de subida de la señal **ready** y al finalizar la cuenta del Timer. La frecuencia del reloj es 32768Hz y notar que 2seg = 2^16/32768Hz.

Se pide:

a) Todo el HW necesario para el sistema, utilizando 32K de ROM y 32 K de RAM.

b) Todo el software (inicialización del sistema, programa principal, directivas para reserva de memoria y rutinas de atención a la interrupción).

Nota: se puede suponer que las tasas de generación y transmisión de datos son tales que la cola nunca se llena y que la transmisión de los datos, incluyendo retransmisiones, siempre termina antes de **Tesp**.