15. – Interrupciones

Introducción a los microprocesadores
2015

Repaso: Métodos de E/S

√ E/S controlada por programa

- El programa decide cuándo
- El programa realiza la transferencia de datos (instrucciones IN y OUT)

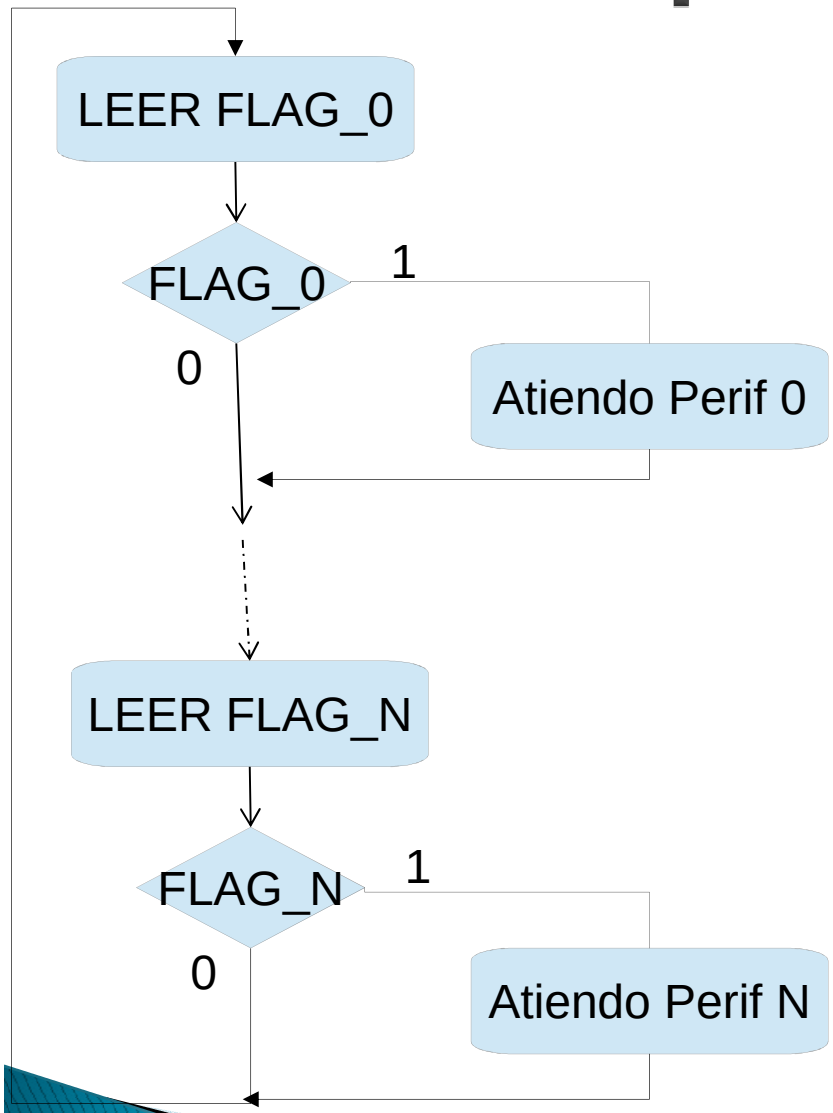
➢ **E/S controlada por Interrupciones**

- Iniciada por el hardware externo sin intervención del programa
- Realizada por la rutina de servicio a la interrupción (instr. IN y OUT)

- E/S controlada por hardware (DMA)
  - Iniciada y realizada por el hardware externo.
  - La transferencia es directa entre el dispositivo de E/S y la memoria.
  - Los datos no pasan por el microprocesador.

  (Fuera de alcance del curso)

Repaso: Polling



- El intercambio de información lo inicia el microprocesador.
- Se utiliza un FFs de estado para sincronizar la transferencia (bandera de estado)
- Esta bandera es consultada periódicamente a través de un puerto de entrada.
- El tiempo de respuesta varía dependiendo de donde se encuentre la rutina de Polling.
- Cuantos más periféricos a atender, el tiempo de respuesta aumenta.
- Si el periférico no necesita ser atendido, se insume tiempo en consultar la bandera.

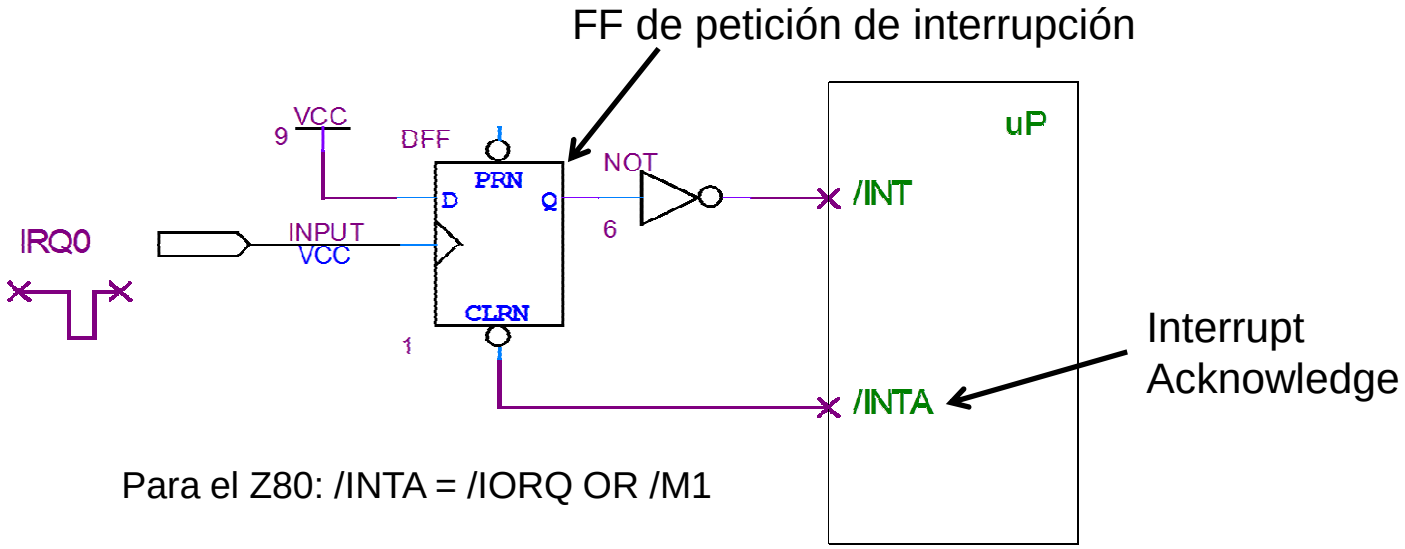
Interrupciones

- Una **interrupción** es un mecanismo por el cual se invoca a una “subrutina” en respuesta a un evento de petición iniciado por un dispositivo hardware externo.

  - Mejora el *“throughput”* del sistema (cantidad total de información útil procesada).
  - Reduce la complejidad del programa.
  - Es el dispositivo externo quien inicia la transferencia de datos al µP.
  - La “subrutina” se le llama “rutina de atención o rutina de servicio a la interrupción o ISR (Interrupt Service Rutine)”.
  - La ejecución de la subrutina queda intercalada entre 2 instrucciones consecutivas del programa en curso.
  - La petición de interrupción es asíncrona (ocurre en cualquier instante), por lo tanto, se DEBE salvar SIEMPRE el estado del µP.

Interrupciones

- Estructura simple que permite a un único dispositivo interrumpir al µP.



Para el Z80: /INTA = /IORQ OR /M1

- El µP consulta la entrada /INT al final de cada instrucción. Si esa entrada está activa, se completa la instrucción en curso, se reconoce la solicitud (/INTA =0), …… ,…….

Interrupciones

Secuencia de atención

1. Se completa la instrucción en curso
   - Se finaliza la ejecución de la instrucción en curso.
2. …

Interrupciones

Secuencia de atención

1. Se completa la instrucción en curso
2. Ciclo de reconocimiento.
   - Señal hardware de reconocimiento para avisar al periférico que va a ser atendido (/INTA = 0)
   - Se guarda la dirección de retorno en el stack para poder retornar.
   - [Se identifica el periférico que pidió interrupción para determinar cuál rutina de atención ejecutar].
   - Se salta a dirección de comienzo de la subrutina de atención.
   - [En algunos casos se borra el FF de petición (utilizando /INTA)].

Interrupciones

Secuencia de atención

1. Se completa la instrucción en curso
2. Ciclo de reconocimiento.
3. Preservar estado
   - Registros y banderas, usualmente en el stack.
   - No se puede saber para qué los estaba usando el programa principal.
   - Responsabilidad del programador

Interrupciones

Secuencia de atención

1. Se completa la instrucción en curso
2. Ciclo de reconocimiento.
3. Preservar estado
4. [Identificar dispositivo]
   - Si no se hizo por hardware en (2), el software debe identificar cuál de los dispositivos interrumpió.

Interrupciones

Secuencia de atención

1. Se completa la instrucción en curso
2. Ciclo de reconocimiento.
3. Preservar estado
4. [Identificar dispositivo]
5. Atención al dispositivo
   - Finalmente podemos atender la solicitud del dispositivo.
   - En algunos casos debemos además borrar el FF de petición (si no se hizo en (2).

Interrupciones

Secuencia de atención

1. Se completa la instrucción en curso
2. Ciclo de reconocimiento.
3. Preservar estado
4. [Identificar dispositivo]
5. Atención al dispositivo
6. Restaurar estado
   - Se debe dejar registros y banderas como los tenía el programa principal cuando fue interrumpido.
   - En general además es necesario habilitar interrupciones (EI).
   - Es responsabilidad del programador.

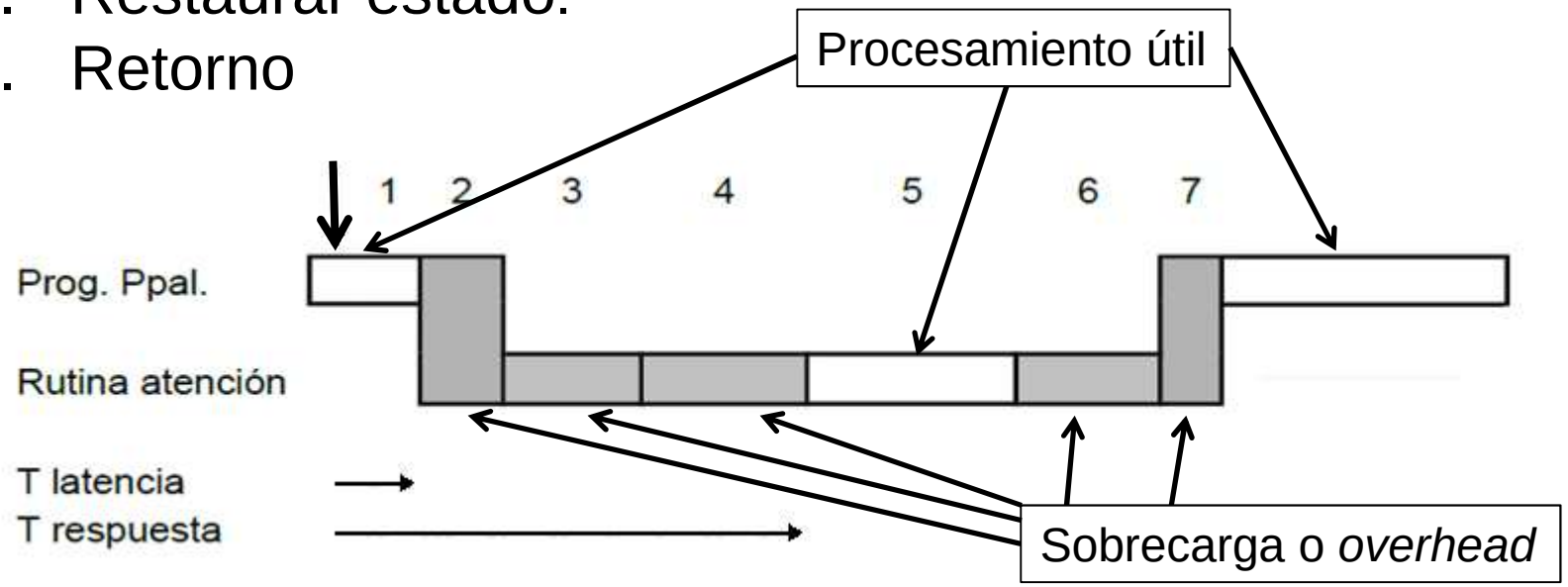
Interrupciones

Secuencia de atención

1. Se completa la instrucción en curso
2. Ciclo de reconocimiento.
3. Preservar estado
4. [Identificar dispositivo]
5. Atención al dispositivo
6. Restaurar estado.
7. Retorno
   - Para el Z80, utilizando la instrucción RET (o RETI o RETN)

Secuencia de atención

1. Se completa la instrucción en curso
2. Ciclo de reconocimiento.
3. Preservar estado
4. [Identificar dispositivo]
5. Atención al dispositivo
6. Restaurar estado.
7. Retorno



Secuencia de atención

- Tiempo de latencia:
  - Desde solicitud hasta que procesador la reconoce.
  - Tiempo variable que depende del instante de la solicitud y la instrucción en curso. ¿Cuál sería el peor caso?
- Tiempo de respuesta:
  - Desde solicitud hasta que es efectivamente atendido.
  - También variable debido al Tiempo de Latencia.
- Deadline
  - Instante a partir del cual si la petición no fue aún atendida se degrada el funcionamiento del sistema.
  - Ejemplo:
    - Reloj implementado con interrupción periódica.
    - Si no se atiende antes de la siguiente solicitud se pierde un período y el reloj atrasa.

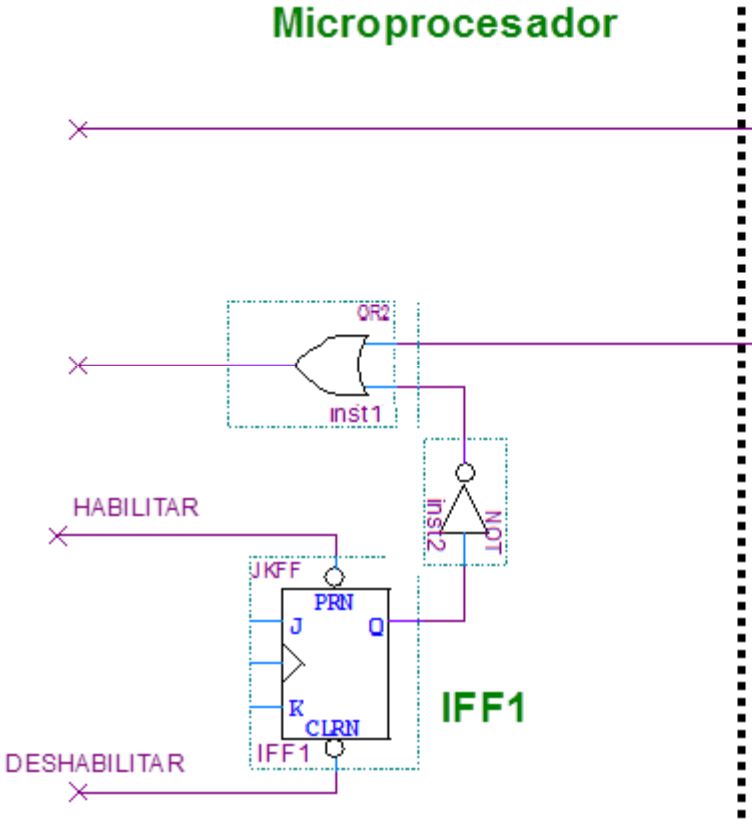
Interrupciones enmascarables

A veces es necesario no atender interrupciones:

- Para inicializaciones varias que utilizan las interrupciones:
  - Stack
  - Hardware externo
  - Microprocesador (registro Interrupciones, tabla interrupciones)
  - Variables (que son utilizadas por las interrupciones)
- Para proteger una sección de programa y evitar que sea interrumpida
  - Código que se debe ejecutar lo más rápido posible.
  - Código que debe demorar un tiempo determinado.
  - Código que accede a datos que también son manipulados por la interrupción.
    - Ej: Incrementar contador de 32 bits en ppal. cuyo valor es utilizado en la ISR.

Interrupciones enmascarables

Para enmascarar la solicitud de interrupción se utiliza un FF



Entrada para interrupciones NO enmascarables

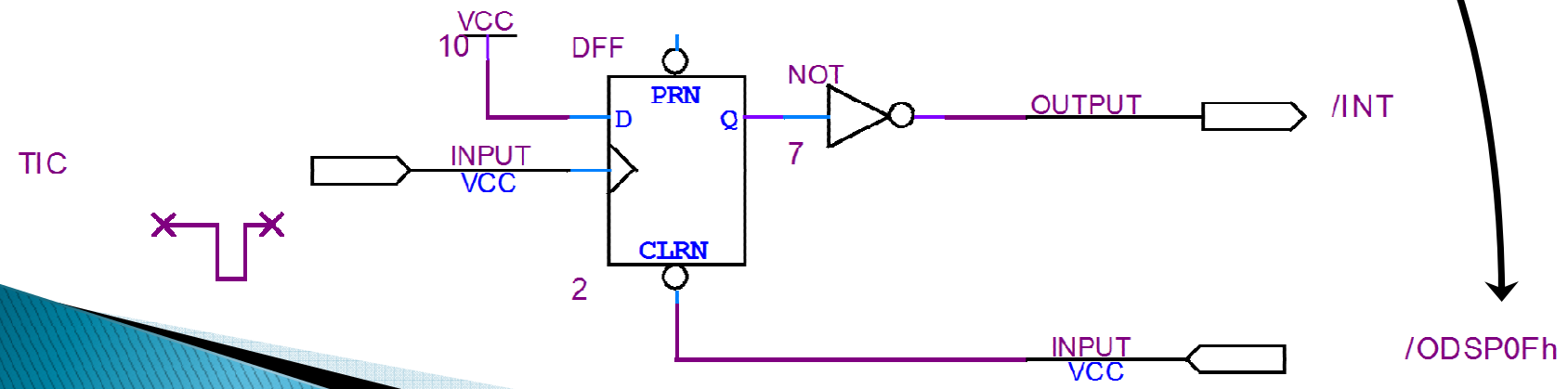
Entrada para interrupciones enmascarables

- En general este FF está incluido en el chip del µP y se dispone de instrucciones para setearlo (habilitar interrupciones: EI) y borrarlo (deshabilitar interrupciones: DI).
- Luego de un RESET el FF debe estar borrado (interrupciones deshabilitadas)

Ejemplo: Interrupción periódica

- Señal TIC periódica
  - TIC es la única petición de interrupción al µP
  - ¿Se puede poner TIC directamente a la entrada del µP sin utilizar el FF?.

```
rutint:
        ;preservo estado
    push af
        ;incremento variable
    ld a, (VARIABLE)
    inc a
    ld (VARIABLE), a
        ; borro FF de petición
    out (0Fh), a
        ; restauro estado y ret
    pop af
    ei
    ret
```



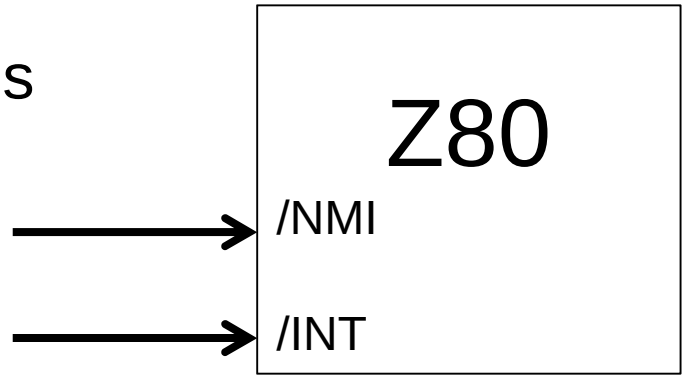
Interrupciones

- En el ejemplo anterior:
  - FF de petición se borra dentro de la subrutina
  - Por tanto, /INT activa y se sigue pidiendo interrupción al ejecutar PUSH AF
  - ¿Por qué no se vuelve a interrumpir al final de PUSH AF?

    R: **Durante el ciclo de reconocimiento se deshabilitan interrupciones.**
  - Se debe volver a habilitar las interrupciones antes de retornar
    - La instrucción EI habilita las interrupciones.

Z80: Estructura de interrupciones

- 2 entradas de interrupciones:
  - **/NMI**: Interrupciones NO enmascarables
  - **/INT**: Interrupciones enmascarables



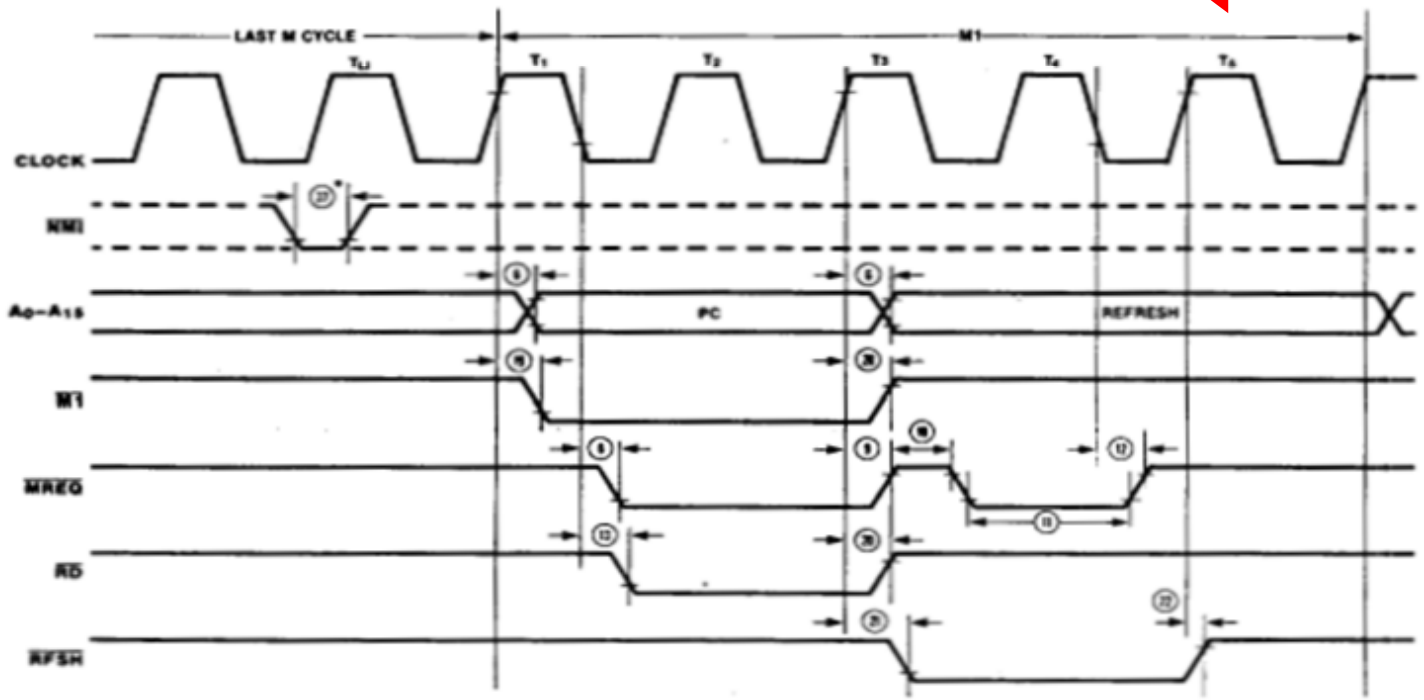
- 2 FF:
  - IFF1: Habilita y deshabilita interrupciones enmascarables.
  - IFF2: Almacena temporalmente IFF1 durante una interrupción NO enmascarable.

Z80: Estructura de interrupciones

- *NMI: Interrupciones NO enmascarables*
  - No se pueden deshabilitar.
  - Es chequeada en el último T de la instrucción en curso si ocurrió un flanco de bajada.
  - Una única interrupción
  - De mayor prioridad
  - La rutina de atención se ejecuta en la dir **0x0066**

Z80: Estructura de interrupciones

- *NMI: Interrupciones NO enmascarables*
  - Ciclo de reconocimiento
    - Es idéntico a M1 pero:
      - Descarta el OPCODE
      - Agrega un T al final (demora 5T).



Z80: Estructura de interrupciones

- *NMI: Interrupciones NO enmascarables*
  - Secuencia
    1. Ciclo de reconocimiento (5T)

       IFF2 ← IFF1

       IFF1 ← 0 (deshabilita interrupciones enmascarable)
    2. Stack ← Dirección retorno (6T)

       PC ← 0x0066
    3. --------- se ejecuta la rutina -------
    4. Retorno: RETN (6T)
       - PC ← Stack
       - IFF2 ← IFF1

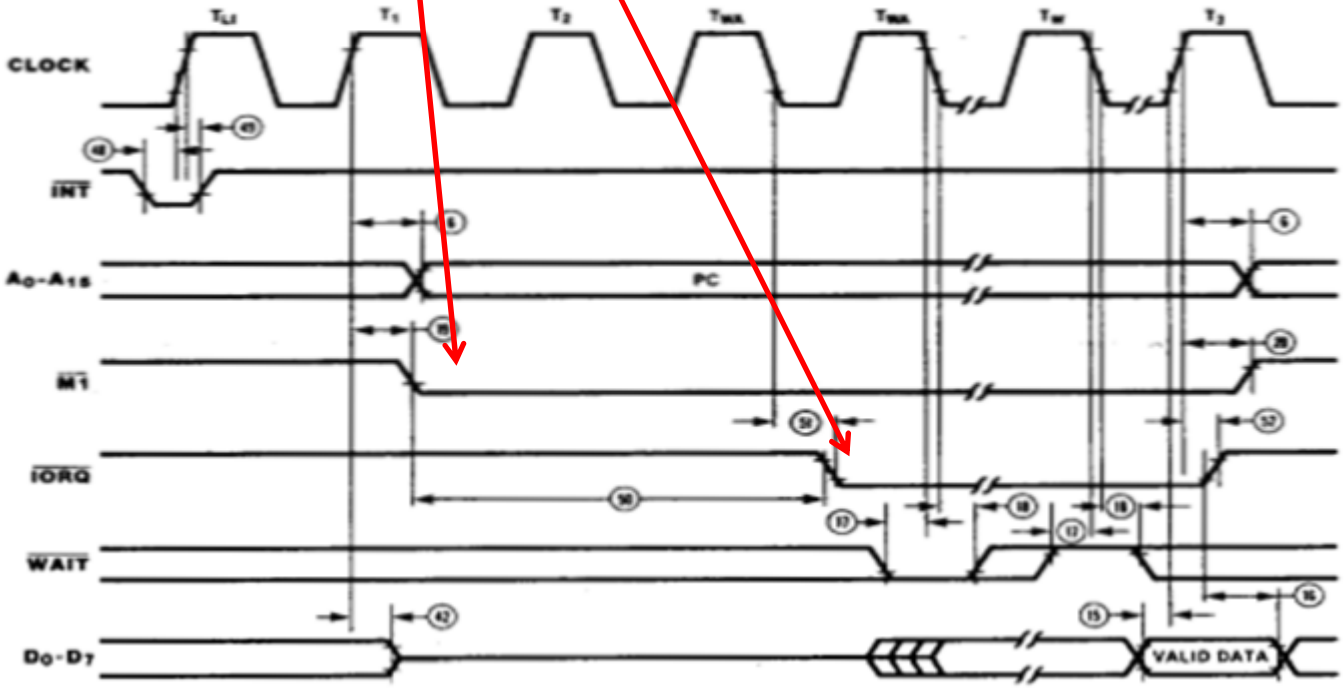
Z80: Estructura de interrupciones

- *INT: Interrupciones enmascarables*
  - Se pueden deshabilitar
    - DI deshabilita (IFF1 ← 0)
    - EI habilita (IFF1 ← 1)
  - Es chequeada el último T de la instrucción en curso si IFF1 = 1.
  - Menor prioridad que NMI
  - Existen 3 modos de interrupciones
    - Modo 0 (modo por defecto)
    - Modo 1
    - Modo 2

    (una vez diseñado el sistema, el modo de interrupciones no cambia).

Z80: Estructura de interrupciones

- *INT: Interrupciones enmascarables*
  - Ciclo de reconocimiento
    - Es el mismo para los 3 modos de interrupciones.
    - IFF1 ← 0 (se deshabilitan interrupciones)
    - Señal /INTA = /M1 or /IORQ
    - Dispositivo pone en bus de datos su identificación (modos 0 y 2).



Z80: Estructura de interrupciones

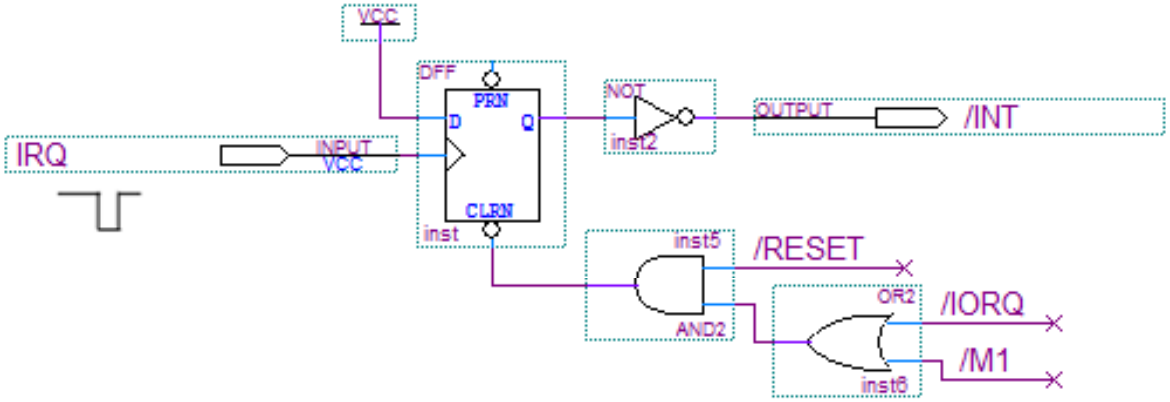
- *INT: Interrupciones enmascarables*
  - MODO 0
    - Modo por defecto luego de un Reset.
    - Incluido en Z80 por compatibilidad hacia atrás (8080)
    - Permite varios periféricos.
    - El Z80 lee el bus de datos durante el ciclo de INTA y lo interpreta como un OPCODE.
    - En general el OPCODE es una instrucción RST
    - Si el OPCODE fuera un CALL, se generan 3 ciclos INTA para completar la instrucción.

Z80: Estructura de interrupciones

- *INT: Interrupciones enmascarables*
  - MODO 1
    - Debe ejecutarse la instrucción IM1 en la inicialización.
    - La rutina de atención se ejecuta en la dir 0x0038
    - Permite solo 1 interrupción enmascarable
    - Secuencia
      1. Ciclo de reconocimiento (5T); IFF1 ← 0 (deshabilita int.)
      2. Stack ← Dirección retorno (6T); PC ← 0x0038
      3. --------- se ejecuta la rutina -------
      4. RET o RETI (PC ← Stack)) (6T)
         - Es indiferente utilizar RET o RETI

Z80: Estructura de interrupciones

- Ejemplo: 1 dispositivo que interrumpe en MODO 1



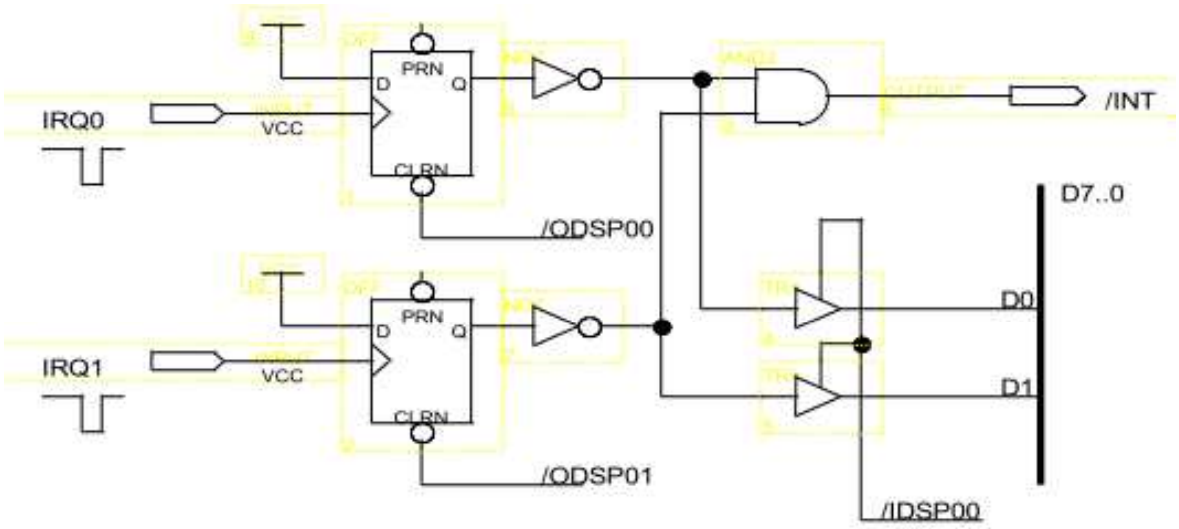

```
          org 38H
rutint1:  push …
          ……
          ……
          ……
          ……
          ……
          pop …
          ei
          ret
```

```
          org 38H
Rutint2:  ei
          push …
          ……
          ……
          ……
          ……
          ……
          pop …
          ret
```

- El FF de petición se borra:
  - Luego de un Reset
  - En el ciclo de reconocimiento con (/IORQ + /M1)
- Rutinas de atención a la interrupción:
  - rutint1: IRQ NO puede interrumpirla
  - rutint2: IRQ SI puede interrumpirla (interrupciones anidadas)

Z80: Estructura de interrupciones

- Ejemplo: 2 dispositivos que interrumpen en MODO 1



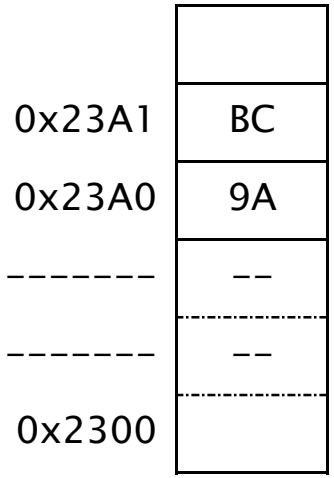

```
pendientes EQU 0
borro0     EQU 0
borro1     EQU 1

           org 38H
rutint:    push af
           in a, (pendientes)
           bit 0, a
           jr z, atiendo0
atiendo1:
           ei
           out (borro1), a
           call isr1
           pop af
           ret
atiendo0:
           out (borro0), a
           call isr0
           pop af
           ei
           ret
```

- Los FF de petición se borran por SW.
- Observar que el IRQ0 puede interrumpir a IRQ1 pero no viceversa (ei)

Z80: Estructura de interrupciones

- *INT: Interrupciones enmascarables*
  - Tabla de interrupciones
    - Ocupa 256 byte (128 direcciones)
    - Comienza en la dir dada por el registro I multiplicado por 0x0100
      - Ej I = 0x23 ➔ La tabla comienza en 0x2300
    - Durante el ciclo INTA el Z80 lee el **vector de interrupciones**.
    - Este vector es el desplazamiento en la tabla, donde se encuentra la dirección de comienzo de la rutina de atención a la interrupción.

Ejemplo:
Sea I = 0x23 y la tabla.
Si vector de interrupciones = A0 ➔ la dirección de comienzo de la rutina de atención a la interrupción es 0xBC9A



Z80: Estructura de interrupciones

- *INT: Interrupciones enmascarables*
  - MODO 2
    - Debe ejecutarse la instrucción IM2 en la inicialización.
    - Permite 128 interrupciones diferentes.
    - Utiliza la tabla de interrupciones definida anteriormente (registro I)
    - El Z80 lee el bus de datos durante el ciclo de INTA y lo interpreta como un **vector de interrupciones**.

    - Secuencia
      1. Ciclo de reconocimiento (5T); IFF1 ← 0 (deshabilita int.)
      2. Stack ← Dirección retorno (6T);
      3. **PC ← lee dir. de comienzo de la rutina de la tabla de interrupciones (6T).**
      4. --------- se ejecuta la rutina -------
      5. RETI (PC ← Stack)) (6T)

         o DEBE ser RETI, no puede ser RET

Z80: Estructura de interrupciones

- Ejemplo: Inicialización en MODO 2
  - Sistema con 32kRAM y 32kROM
  - Sean 3 dispositivos cuyos vectores son el 0vo, el 2do y el 5to
  - Las rutinas de atención son rutint_0, rutint_2 y rutint_5

Solución con la tabla en RAM:

```
            ORG 0x8000
TABLA:      DS 256                ; Reservo memoria para la Tabla de Interr.

            ORG 0x0000
            LD SP, 0              ; inicializo SP en la parte sup de la RAM + 1
            LD A, Tabla / 256
            LD I, A               ; tabla de interr. al comienzo de la RAM
            LD BC, rutint_0
            LD (TABLA), BC        ; cargo rutint_0 en el lugar 0 de la tabla
            LD BC, rutint_2
            LD (TABLA+2*2), BC    ; cargo rutint_2 en el lugar 2 de la tabla
            LD BC, rutint_5
            LD (TABLA+2*5), BC    ; cargo rutint_5 en el lugar 5 de la tabla
            IM 2                  ; modo 2 de interrupciones
            …                     ; inicialización de otros dispositivos
            EI                    ; habilito interrupciones
```

Z80: Estructura de interrupciones

- Ejemplo: Inicialización en MODO 2
  - Sistema con 32kRAM y 32kROM
  - Sean 3 dispositivos cuyos vectores son el 0vo, el 2do y el 5to
  - Las rutinas de atención son rutint_0, rutint_2 y rutint_5

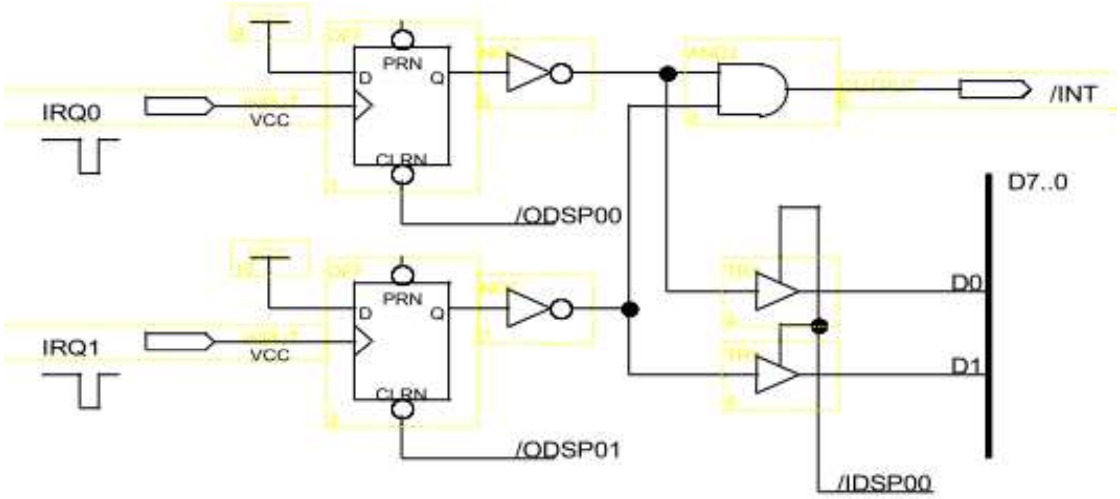
Solución con la tabla en ROM:

```
          ORG 0x0000
          LD SP, 0x0000       ; inicializo SP al fin de RAM + 1
          LD A, TABLA / 256
          LD I, A             ; tabla de interr. en dir 0x2000 en ROM
          IM 2                ; modo 2 de interrupciones
          ...                 ; inicialización de otros dispositivos
          EI                  ; habilito interrupciones
          ...
          ORG 0x2000          ; en ROM
TABLA:    DW rutint_0         ; directivas para grabar la tabla en la ROM
          DW
          DW rutint_2
          DW
          DW
          DW rutint_5
```

Interrupciones - Prioridades

- Ejemplo de prioridades:

2 dispositivos que interrumpen en MODO 1




```
pendientes EQU 0
borro0     EQU 0
borro1     EQU 1

           org 38H
rutint:    push af
           in a, (pendientes)
           bit 0, a
           jr z, atiendo0
atiendo1:
           ei
           out (borro1), a
           call isr1
           pop af
           ret
atiendo0:
           out (borro0), a
           call isr0
           pop af
           ei
           ret
```

- IRQ0 tiene mayor prioridad que IRQ1. En este caso las prioridades se manejan por software con la ubicación de “ei”.
- Esto no es posible si hay más de 2 periféricos. Se requiere +HW y +SW (aumenta complejidad, *overhead* y rápidamente se hace inmanejable).

Interrupciones – Prioridades

- Orden de prioridad para:

1) Más de una petición pendiente en ciclo INTA
   - ¿Cuál debe ser atendida?
   - En ejemplo modo 1: solución software
     - ¿Cuál flag miro primero?
   - En modo 2: **Solución Hardware**
     - ¿Quién pone su vector sobre el bus?

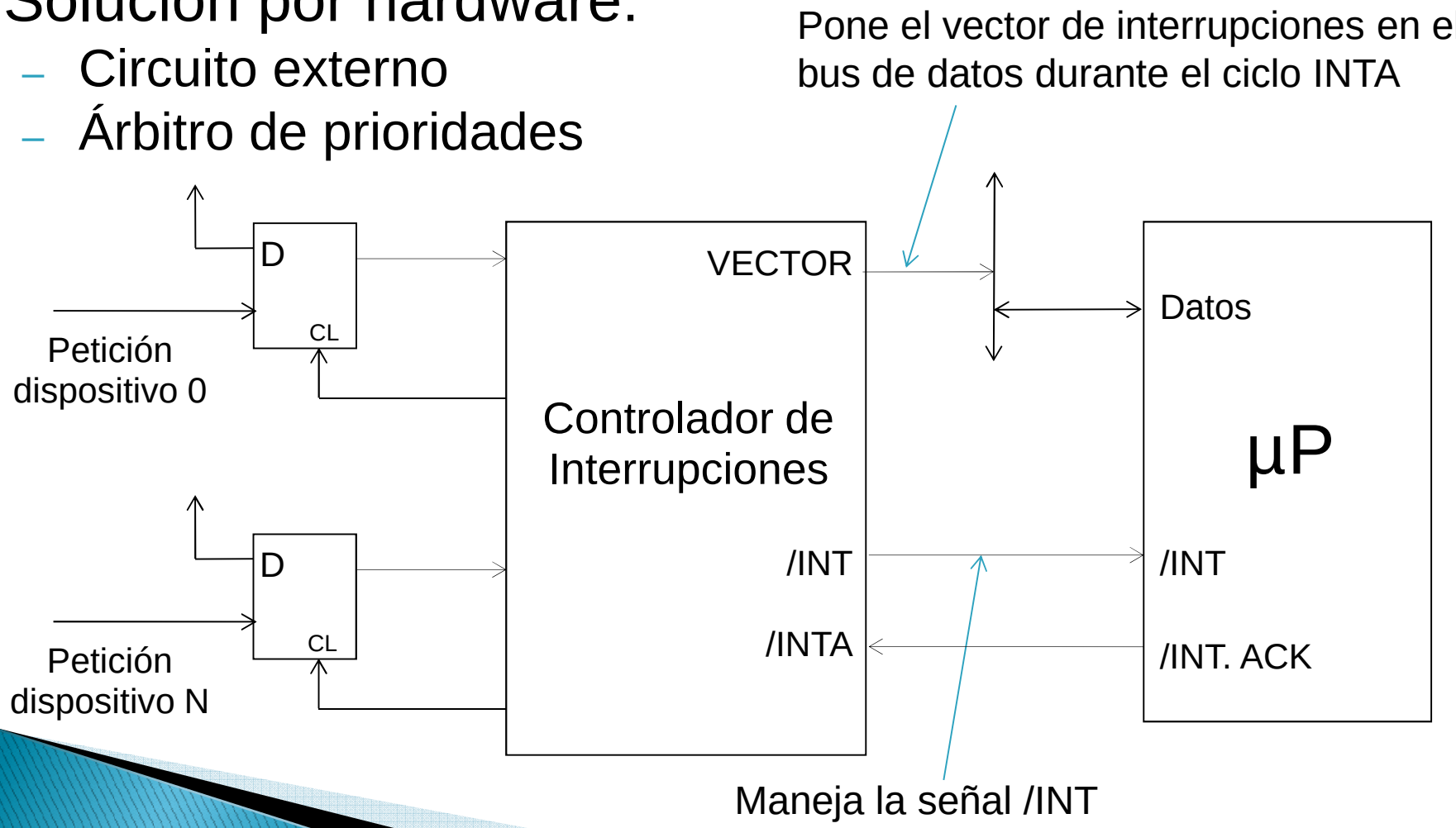
2) Periférico importante interrumpe a otro menos importante pero no a la inversa.
   - En ejemplo modo 1: jugando con ubicación de EI.
     - Imposible con más de dos periféricos.
   - En modo 2. **Solución Hardware**
     - Dejar pasar o no la solicitud del periférico hasta entrada /INT

Solución Hardware = Controlador de Interrupciones

Interrupciones – Prioridades

Controlador de Interrupciones

- Solución por hardware:
  - Circuito externo
  - Árbitro de prioridades



Interrupciones – Prioridades

Controlador de Interrupciones

- Cuando llega ciclo INTA
  - Pone en el bus de datos el vector del dispositivo de más prioridad de los pendientes de ser atendidos.

- Cuando llega una solicitud
  - La pasa al procesador (activando INT) sólo si el dispositivo es de más prioridad que el dispositivo que está siendo atendido.
  - Caso contrario espera a que terminen todos los de mayor prioridad

- Chip 8259 de Intel
  - Utilizado en computadores personales

Interrupciones – Prioridades

Estrategias de Prioridades:

Para el manejo de prioridades pueden utilizarse diferentes estrategias. Algunas de ellas:

- **Jerarquía fija de prioridades** (“Fully Nasted”)
  - Las prioridades se fijan a priori y no se modifican.
  - Un periférico puede interrumpir a otro de menor prioridad, pero no viceversa.
- **Rotación automática**
  - El último periférico en ser atendido pasa a ser el de menor prioridad.

Interrupciones – Prioridades

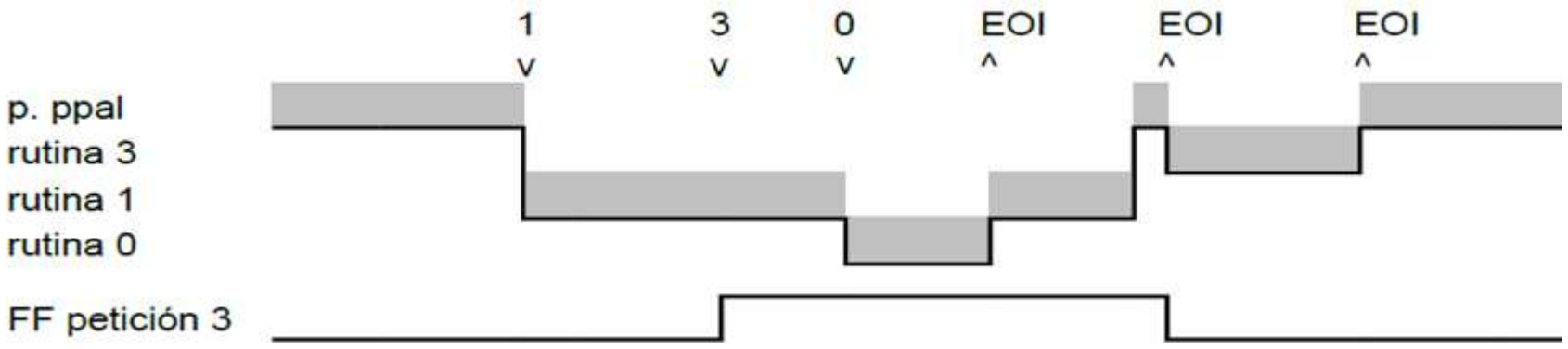
Jerarquía fija de prioridades (“Fully Nasted”)

- *Política*
  - *Siempre se está atendiendo al de mayor prioridad*
- Asignación de prioridades a los dispositivos
  - Problema complejo
  - Varias características a considerar:
    - Requerimientos en tiempo respuesta.
    - Qué tan catastrófico es no atenderlo a tiempo.
    - Periodicidad
    - Duración de rutina de atención a interrupción
  - Hay un único orden de prioridades.
  - Se debe garantizar que se respetan todos los *deadlines*

Interrupciones – Prioridades

Jerarquía fija de prioridades (“Fully Nasted”)

- Ejemplo: Periféricos p0, p1, p2, p3
  - Prioridades fijas (p0 mayor prioridad)
  - Inicialmente ejecutando programa principal
  - Interrumpe p1 → pasa a ejecutar rutina 1
  - Pide interrupción p3
    - Menor prioridad que p1, debe esperar
  - Pide atención p0
    - Mayor prioridad → pasa a ejecutar rutina 0
  - P3 recién es atendido después de terminar p0 y p1



EOI: End Of Interrupt

Interrupciones – Prioridades

Jerarquía fija de prioridades (“Fully Nasted”)

- Controlador necesita saber quién está ejecutando en cada momento.
- Sabe cuándo comienza cada rutina
- Necesita mecanismo para saber cuándo termina.
- Solución 8259:
  - Comando “Fin de Interrupción” (EOI) dado por el procesador escribiendo en un puerto.
- Solución Zilog
  - En modo 2 se retorna con instrucción RETI
    - Ídem RET
    - Opcode diferente (dos bytes ED 4D)
    - Controlador observa bus hasta ver pasar el opcode de RETI

Interrupciones – Prioridades

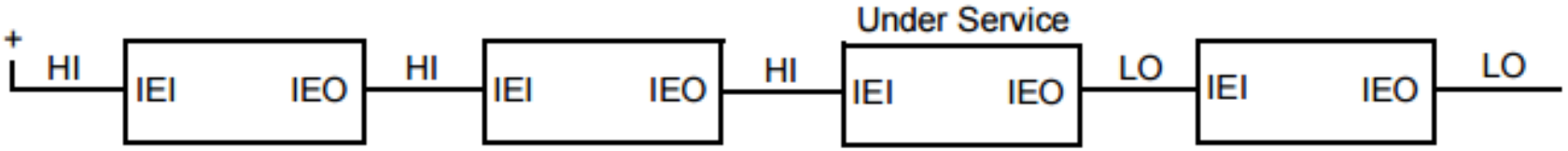
Prioridad rotativa

- Más “justo” para dispositivos de similar importancia
- Cada vez que se atiende a un dispositivo, éste pasa al último lugar en el orden de prioridad.
- Permite poner cota superior a tiempo de respuesta (lo que no es posible en el esquema *“Fully Nested”).*

Interrupciones – Prioridades

“Daisy chain”

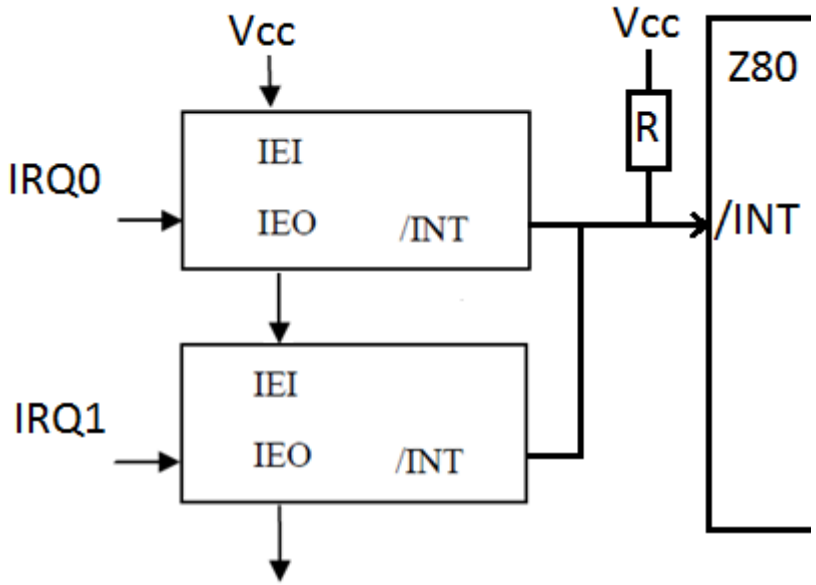
- Mecanismo distribuido de arbitración de prioridades utilizado por Zilog.
- Es un esquema prioridad fija (*Fully nested*).
- Cada bloque tiene las entradas/salidas:
  - Entrada IEI (Interrupt Enable Input).
  - Salida IEO (Interrupt Enable Output)
  - Salida /INT hacia el Z80.
- Los bloques se conectan en cadena:



Interrupciones – Prioridades

“Daisy chain”

- Entrada IEI (Interrupt Enable Input).
  - En nivel bajo, indica que algún periférico de mayor prioridad no finalizó su solicitud.
  - La entrada IEI del bloque de mayor prioridad se conecta a Vcc
- Salida IEO (Interrupt Enable Output)
  - Conectada a la IEI del siguiente bloque.
  - IEO = 0 (inactiva) si tiene un solicitud sin finalizar o IEI = 0.
  - En el caso particular en que tiene una solicitud pendiente que no fue reconocida (INTA) y detecta un RETI en el bus de datos pone IEO = 1 hasta finalizar el RETI.
- Salida /INT
  - /INT = 0 si hay petición pendiente y IEI = 1
  - Salida en colector abierto. Se conecta mediante pull up a /INT del Z80.



IRQ0 mayor prioridad que IRQ1

Interrupciones – Prioridades

“Daisy chain”

- ¿Cómo detecta un dispositivo un ciclo INTA?
- R: Tiene como entradas /M1 e /IORQ provenientes del Z80 (/M1 + /IORQ = /INTA)

- ¿Cómo sabe un dispositivo que debe poner SU vector de interr. en el bus de datos?
- R: Detecta un ciclo INTA, tiene una petición pendiente e IEI = 1.

- ¿Como sabe un dispositivo cuando finaliza su ISR (Interrupt Service Rutine)?
- R: Observa el bus de datos detectando el OPCODE de la instrucción RETI. Si durante la ejecución del RETI, IEI = 1, lo reconoce como un EOI (End Of Interrupt).

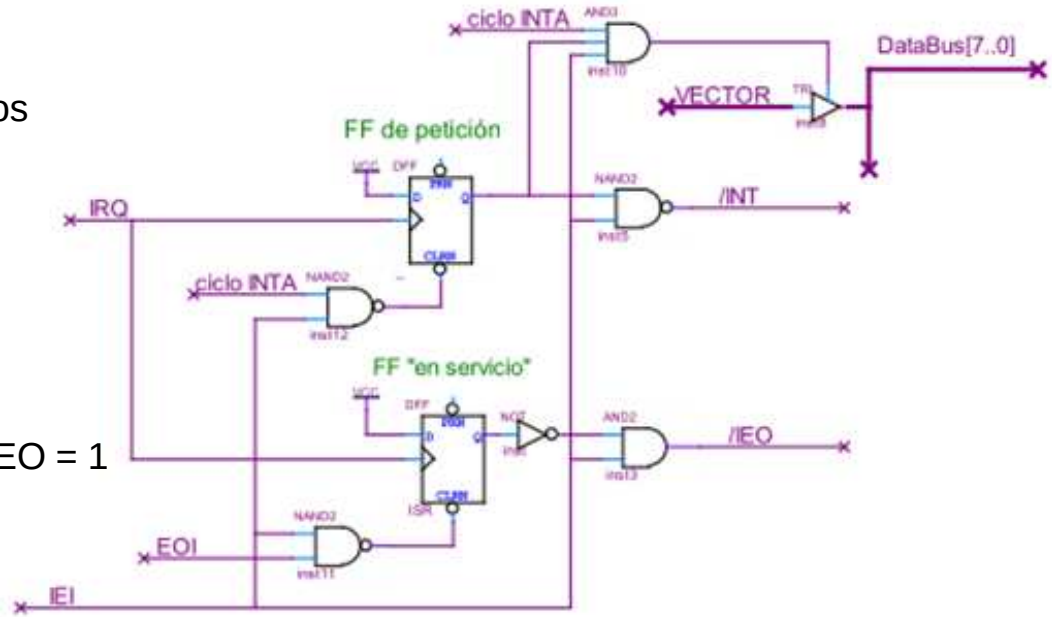
Interrupciones – Prioridades

Secuencia “Daisy chain”

1. Llega una solicitud de interrupción.
   - Se memoriza en el FF de petición
   - Se baja IEO
   - Si IEI = 1 ➔ /INT = 0
2. Detecta ciclo INTA
   - Solicitud pendiente
   - IEI = 1

   }
   - VI en bus datos
   - Borra petición
   - IEO = 0
3. Atención a ISR
   - Se mantiene IEO = 0
4. Fin de Interrupción
   - Se detecta RETI en bus de datos
   - IEI = 1

   } - IEO = 1

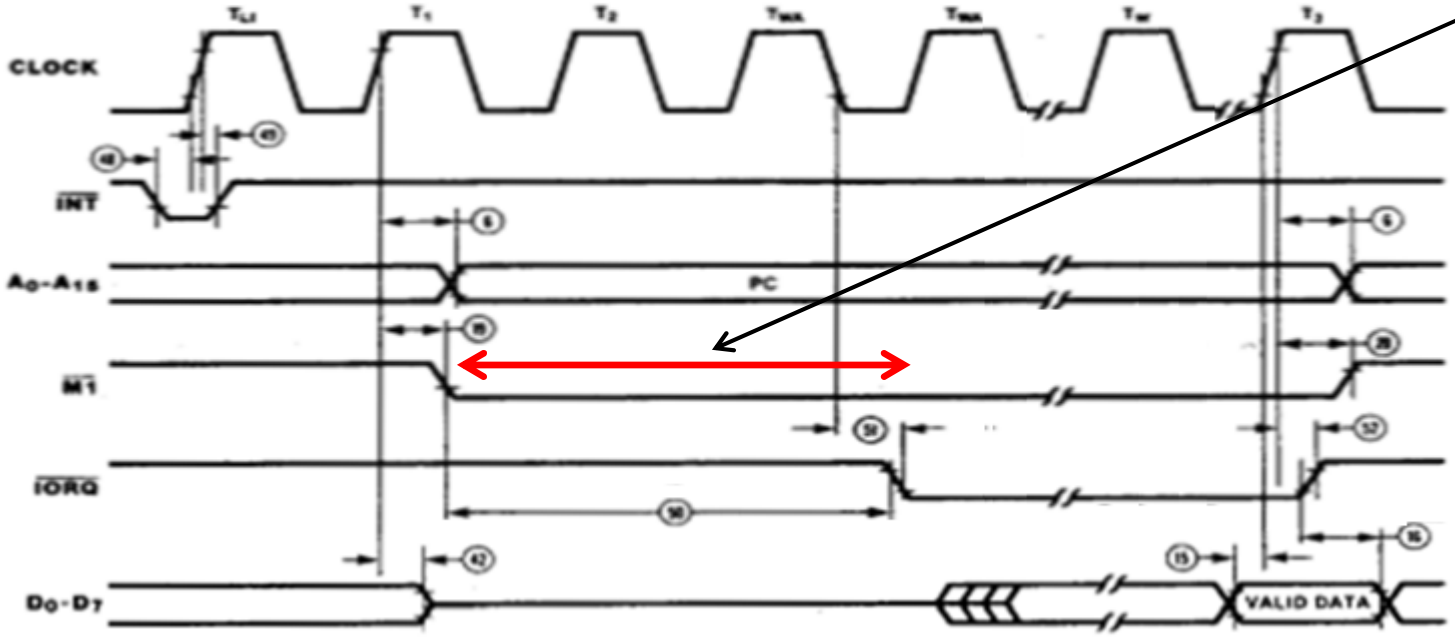


Este circuito es solo a los efectos de explicación pues no contempla todos los detalles del protocolo Dasy Chain.

Interrupciones – Prioridades

Propagación IEI – IEO en “Daisy chain”

- En modo 2 de interrupciones el Z80 permite hasta 128 dispositivos diferentes.
- Existe un retardo de propagación de IEI e IEO a lo largo de la cadena.
- Para permitir que finalice la propagación, ningún dispositivo puede cambiar su estado de requerimiento de interrupción (/INT e IEO) cuando /M1 está activa.
- El tiempo entre la bajada de /M1 y de /IORQ se utiliza para resolver la propagación.

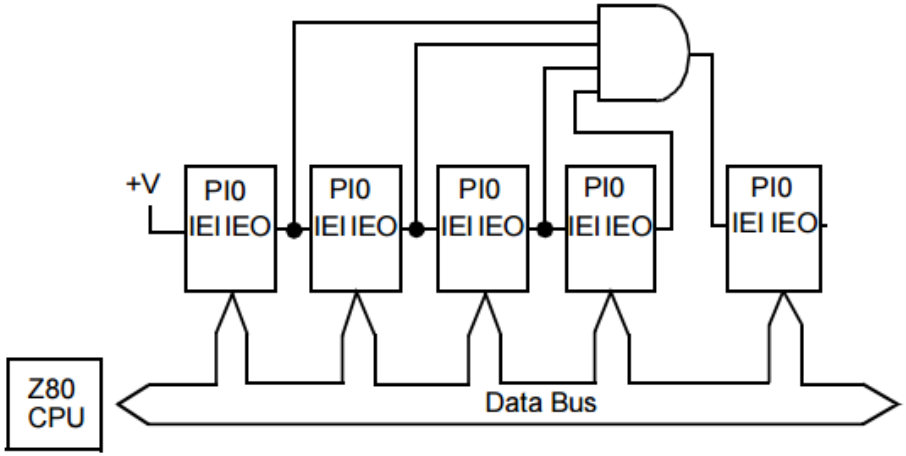


Interrupciones – Prioridades

Propagación IEI – IEO en “Daisy chain”

- ¿Qué sucede si el tiempo entre la bajada de /M1 e /IORQ no es suficiente?

  R1: Existen soluciones por Hardware para minimizar el tiempo de propagación



  R2: Si esto no es suficiente es posible extender este tiempo con Hardware adicional (esto escapa a este curso).

- Lo IMPORTANTE es saber que existen muchos detalle que requieren un estudio detallado para llegar a una solución.

Interrupciones – Prioridades

Ejercicio 8.1

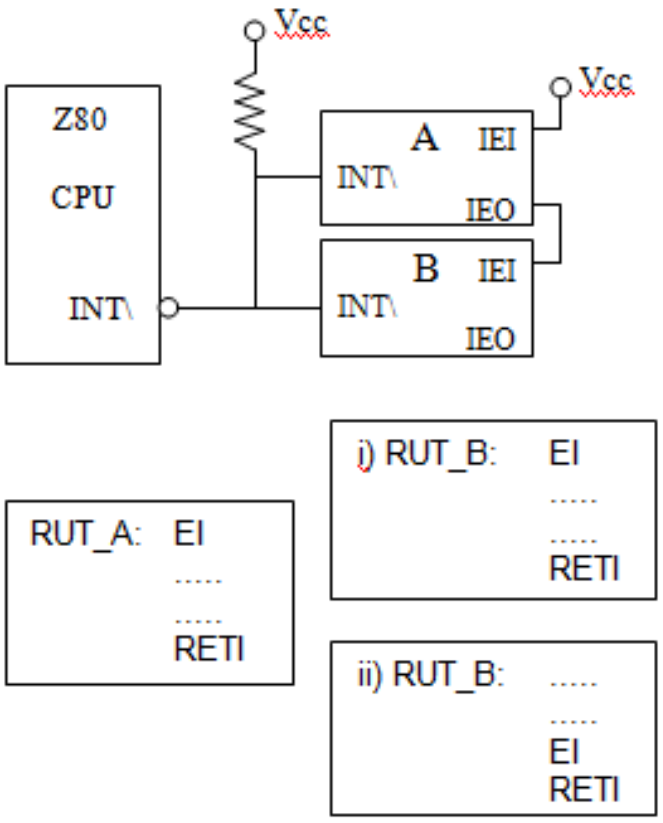
Indicar en un diagrama de tiempos cual tramo de programa (prog. principal, RUT_A o RUT_B) se está ejecutando en cada instante para los dos casos de la rutina RUT_B i) y ii).

Las INT ocurren en:

t = 0	Interrumpe A.
t = 1ms	Interrumpe B.
t = 8ms	Interrumpe B.
t = 10ms	Interrumpe A.

Nota:

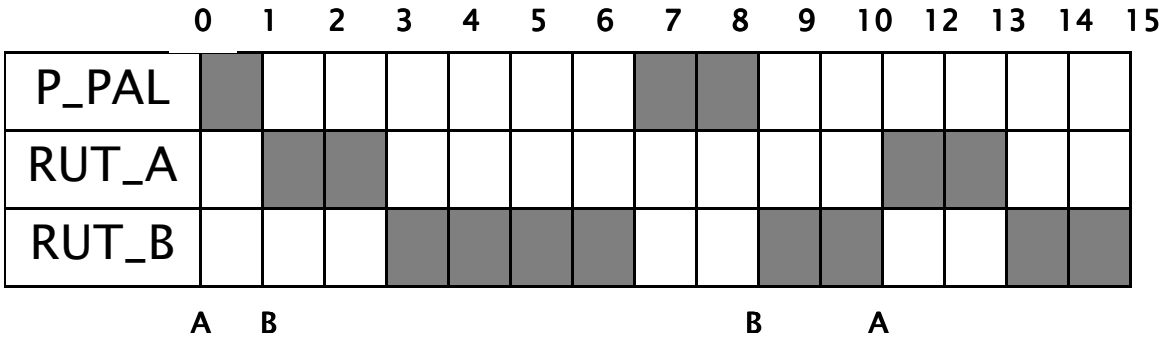
- Duración de la ejecución de RUT_A: 2ms
- Duración de la ejecución de RUT_B: 4ms
- Suponga duración del ciclo de reconocimiento a INT despreciable.

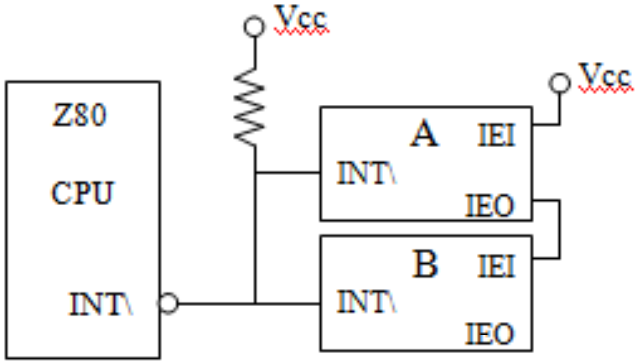


Interrupciones – Prioridades

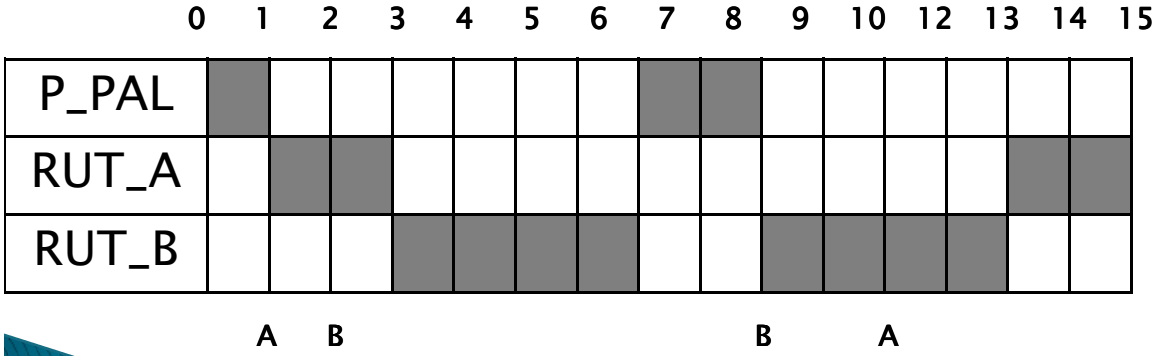
Ejercicio 8.1 (solución)

Caso i)





Caso ii)



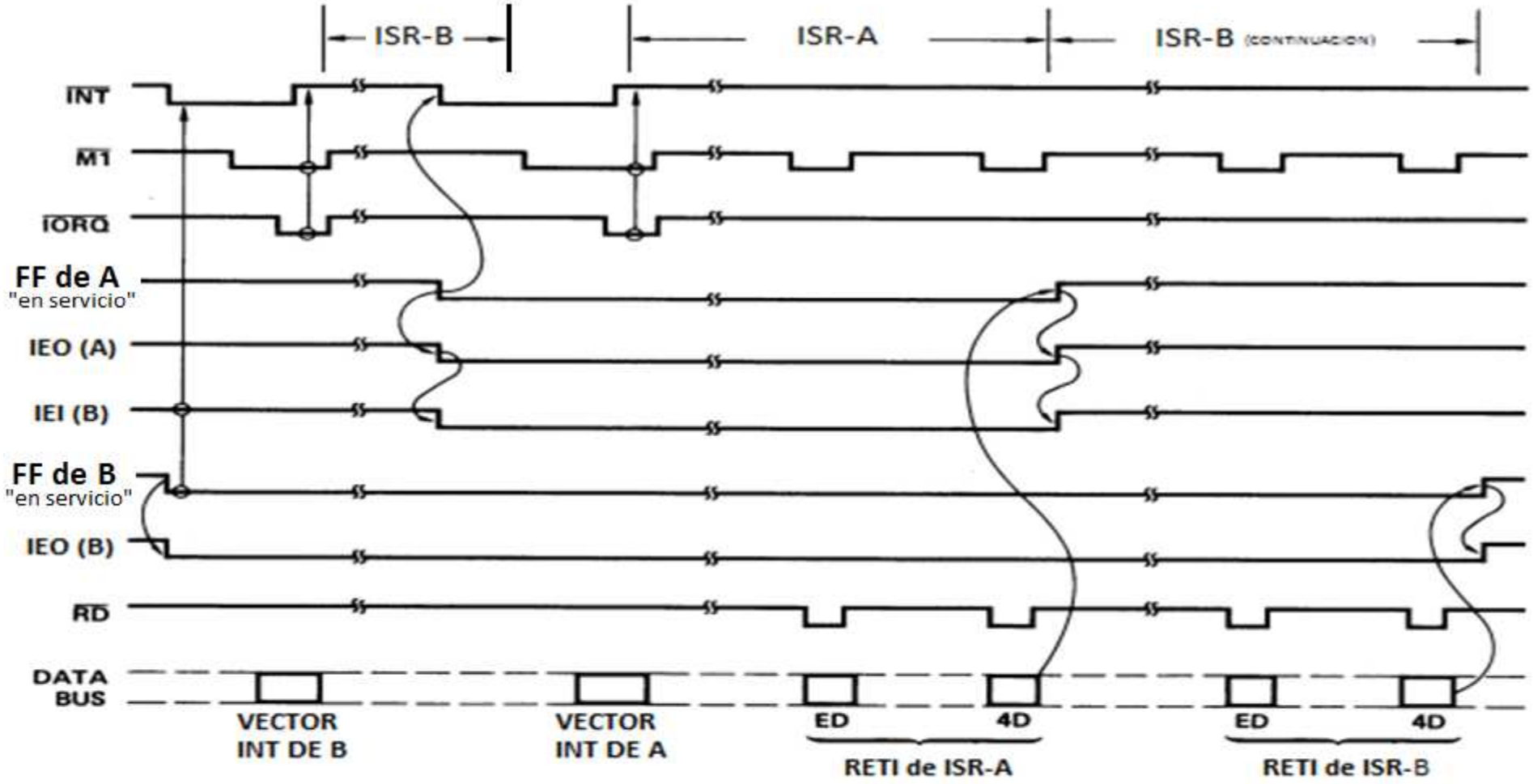

```
RUT_A:  EI
        .....
        .....
        RETI
```

```
i) RUT_B:  EI
           .....
           .....
           RETI
```

```
ii) RUT_B:  .....
            .....
            EI
            RETI
```

Interrupciones – Prioridades

Ejercicio 8.1: Diagrama de tiempos Caso i) (a partir de los 8ms)



Interrupciones – Prioridades

Ejercicio 8.1: Diagrama de tiempos Caso ii) (a partir de los 8ms)

- **A** tiene un requerimiento pendiente y no fue atendido; al detectar un RETI (que no va a ser para **A**) habilita IEO para que sea reconocido por **B** de menor prioridad.