11. – Entrada y Salida controlada por programa

Introducción a los microprocesadores
2015

E/S controlada por programa

- Para poder interactuar con el mundo exterior el sistema con microprocesador requiere de transferencia de datos:
  - hacia el sistema ➔ **ENTRADAS**
  - desde el sistema ➔ **SALIDAS**
- Se requiere circuitería adicional

E/S controlada por programa

Veremos:

- Métodos de E/S
- Puertos
- Transferencia de datos
  - Incondicional
  - Condicional (*Handshaking*)
- Polling

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)

  (La veremos más adelante)

- 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)

Puertos: Mapeo

- E/S mapeada en memoria
  - Se utiliza el mismo espacio de direcciones de memoria
  - Instrucciones de acceso a memoria (LD…..)
- E/S aislada:
  - Espacio de direcciones separado de memoria

Nuestro caso: el Z80 (y T80)

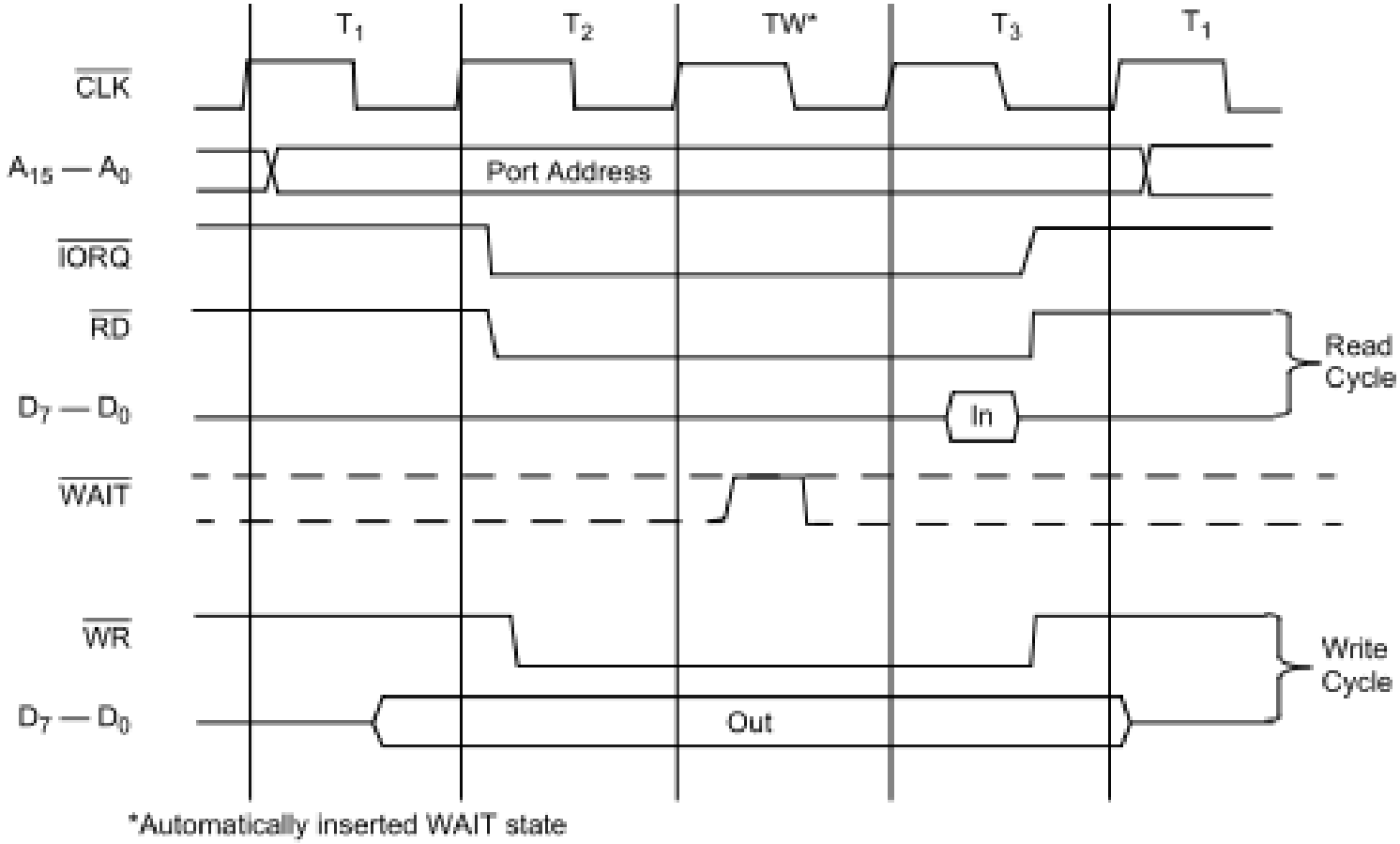
- E/S aislada: Se diferencia con /IORQ
- 256 puertos entrada
- 256 puertos salida

Espacios de Entrada y Salida separados

- Instrucciones específicas (IN y OUT)

Puertos

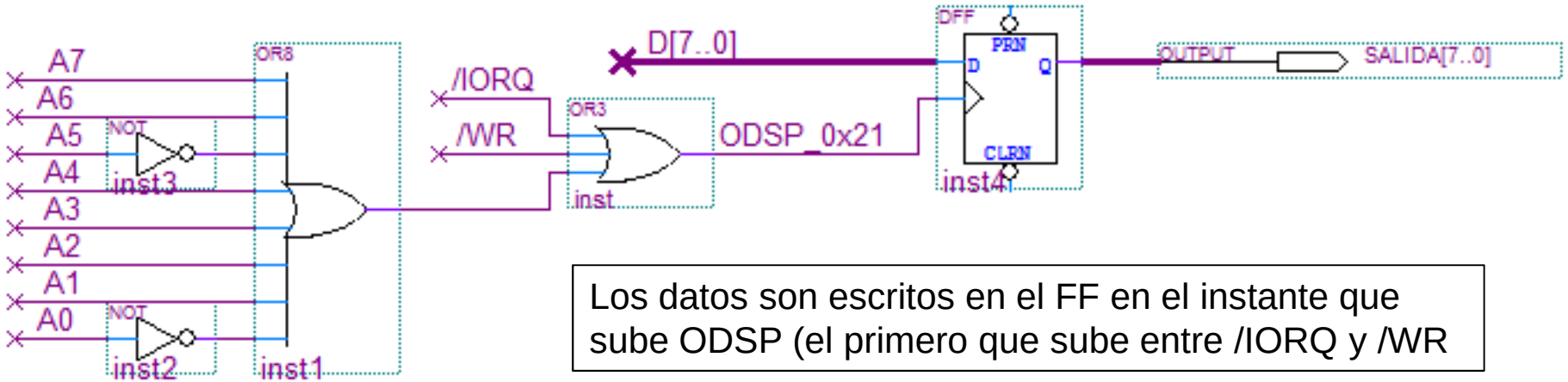
- Ciclo de Entrada / Salida



Puertos: Salida

- Puerto de salida = REGISTRO
  - Elemento de memoria (Latch o FF) que capture contenido del bus de datos en el instante en que están presentes.
  - Pulso ODSP_xx a partir de /IORQ, /WR y A[7..0]
  - ODSP: Output Device Select Pulse (activo por nivel bajo)

- Ejemplo: Puerto de salida en la dirección 0x21

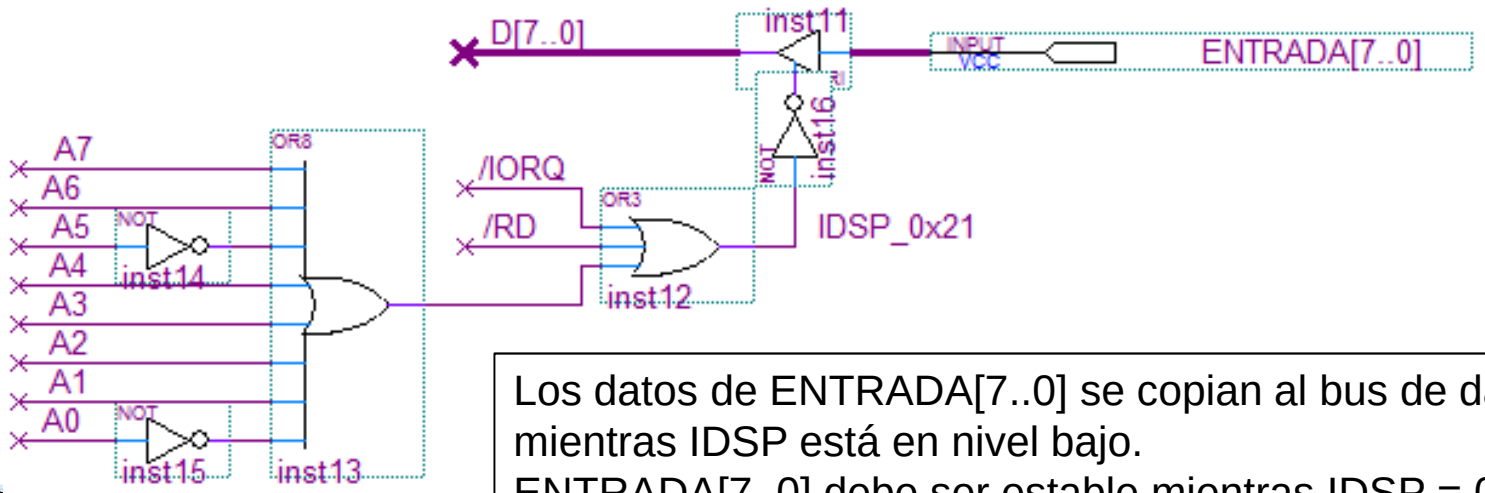


Los datos son escritos en el FF en el instante que sube ODSP (el primero que sube entre /IORQ y /WR

Puertos: Entrada sin memoria

- Puerto de entrada sin memoria:
  - Buses triestado ➔ BUFFER triestado (Z80)
  - Buses multiplexados ➔ Compuerta AND (T80) (lo veremos en breve)
  - Pulso IDSP_xx a partir de /IORQ, /RD y A[7..0]
  - IDSP: Input Device Select Pulse (activo por nivel bajo)

- Ejemplo: Puerto de entrada en la dirección 0x21 **para buses triestado**

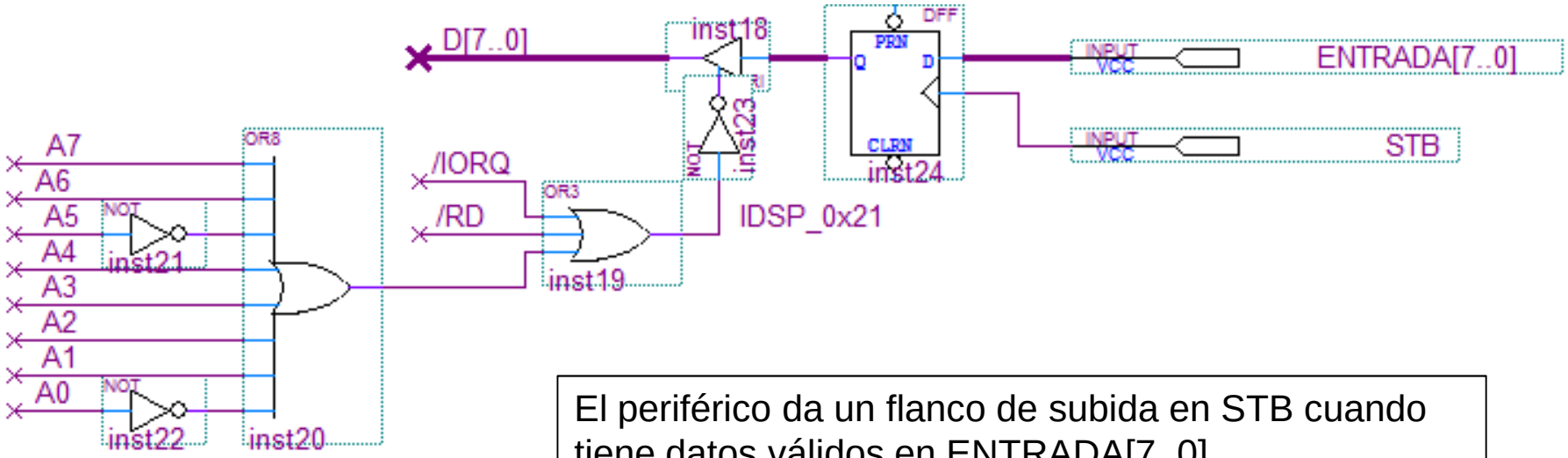


Los datos de ENTRADA[7..0] se copian al bus de datos mientras IDSP está en nivel bajo.
ENTRADA[7..0] debe ser estable mientras IDSP = 0.

Puertos: Entrada con memoria

- Puerto de entrada con memoria:
  - Agrega al caso anterior un elemento de memoria (latch o FF) el cual es manejado por el dispositivo externo.
  - De esta forma se aseguran los datos estables en ENTRADA[7..0]

- Ejemplo: Puerto de entrada con memoria en la dirección 0x21 **para buses triestado**



El periférico da un flanco de subida en STB cuando tiene datos válidos en ENTRADA[7..0]

Transferencia de datos

Transferencia incondicional

- Se presupone que el dispositivo está listo
- Cuando el programa quiere hacer una transferencia, la hace sin más

- Ejemplos salidas:
  - Indicadores luminosos
  - Configuración a un periférico

- Ejemplos entradas:
  - Llaves (SW en placa lab)
  - Status de un periférico

Transferencia de datos

Transferencia condicional (HANDSHAKING)

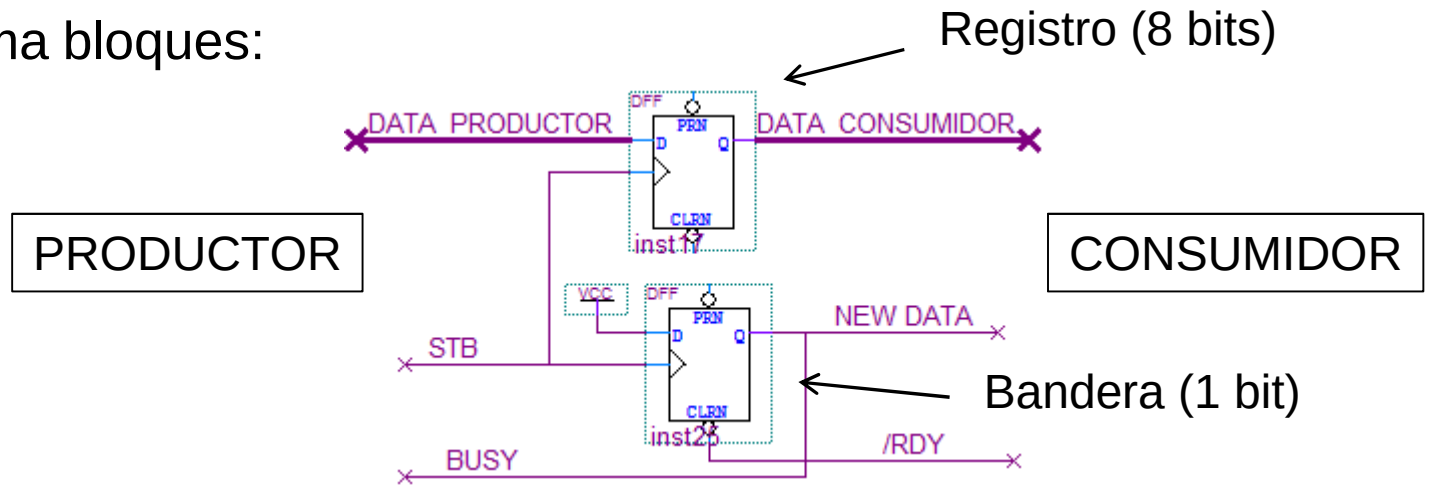
- Consulta previa al dispositivo para ver si está pronto.

- Sincronización productor-consumidor
  - Productor genera datos a su velocidad
  - Consumidor los procesa de a una palabra por vez, le lleva un tiempo cada palabra.
  - ¿Cómo sincronizarlos?

- Ej. productor puede ser un teclado
  - Si leo dos veces seguidas una letra “U”, cómo distingo el caso tecla apretada todo el tiempo del caso se presionó dos veces la misma tecla?

HANDSHAKING

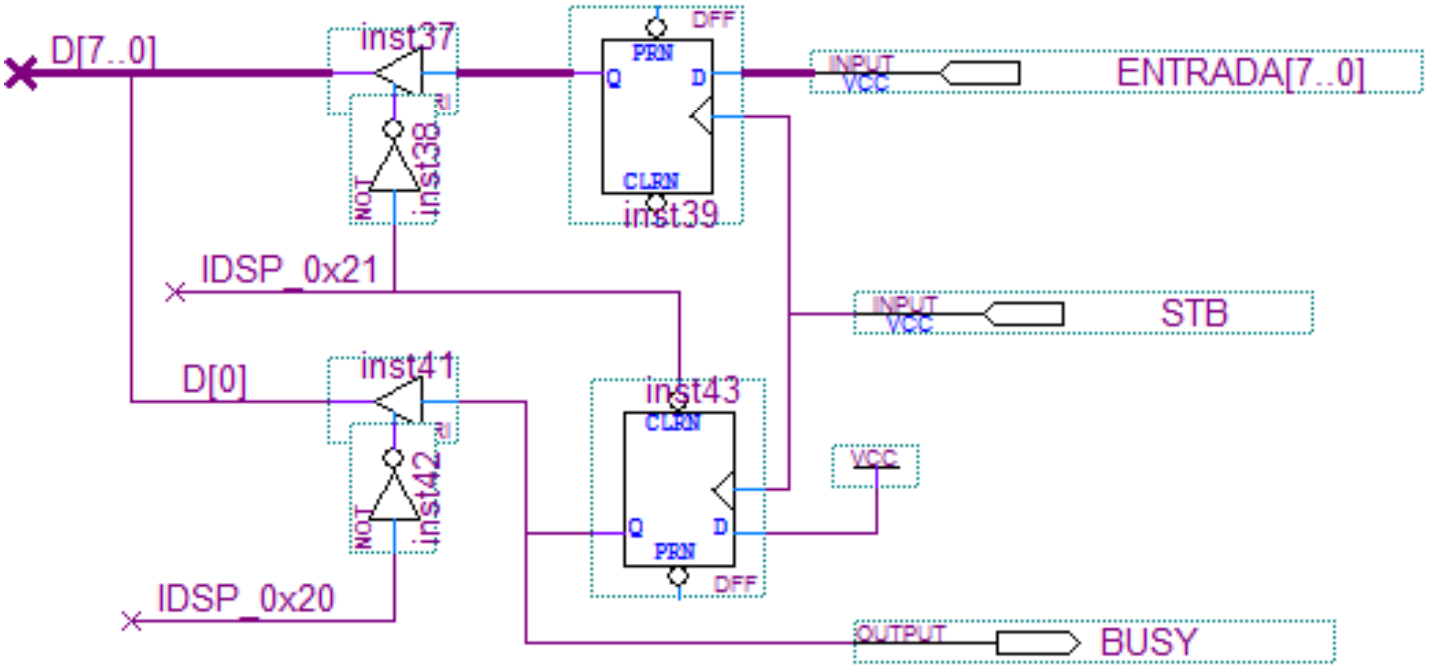
- Diagrama bloques:



- Productor
  - Consulta la bandera para saber si dato anterior fue consumido
  - Prende bandera cuando escribe nuevo dato
- Consumidor
  - Consulta la bandera para saber si hay un nuevo dato
  - Borra la bandera cuando lee el dato

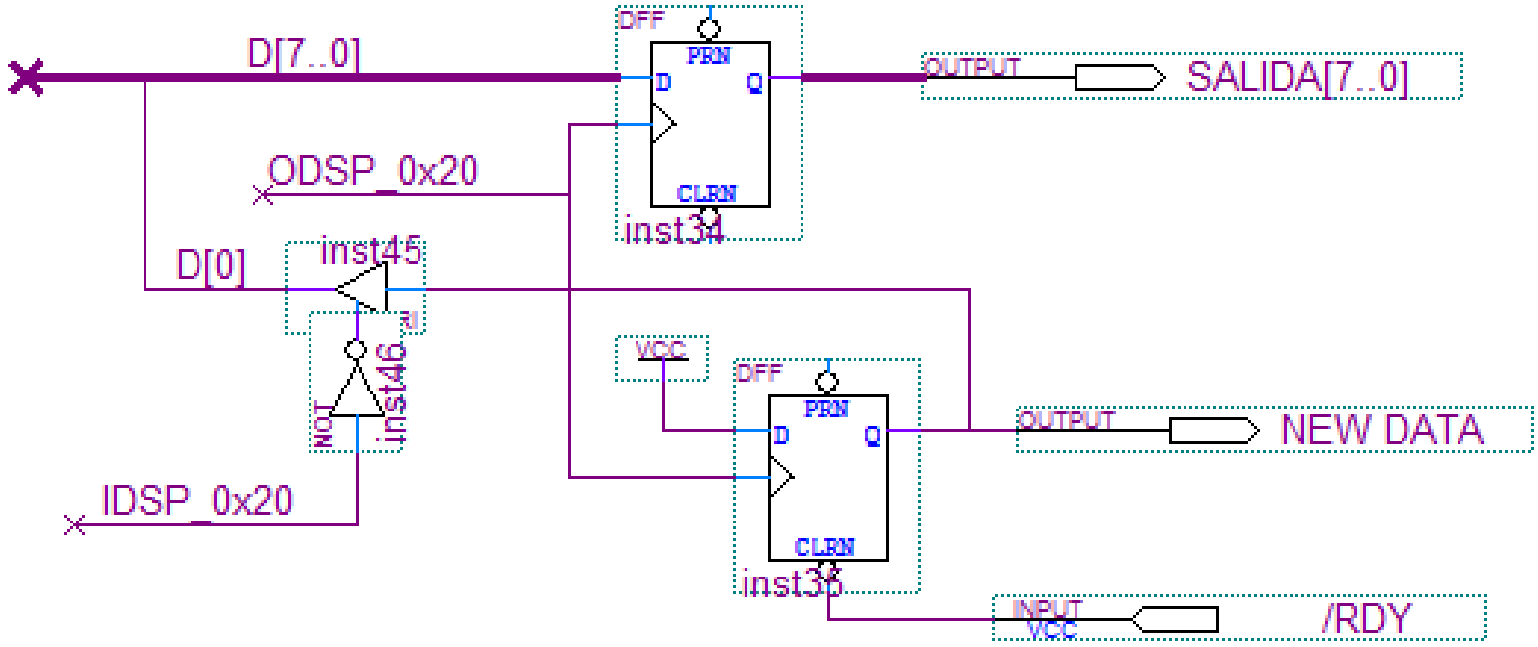
HANDSHAKING

- Puerto de entrada con handshaking
  - Puerto de entrada para datos
  - Puerto de entrada para bandera
  - Borrado de bandera con el pulso IDSP del puerto de datos



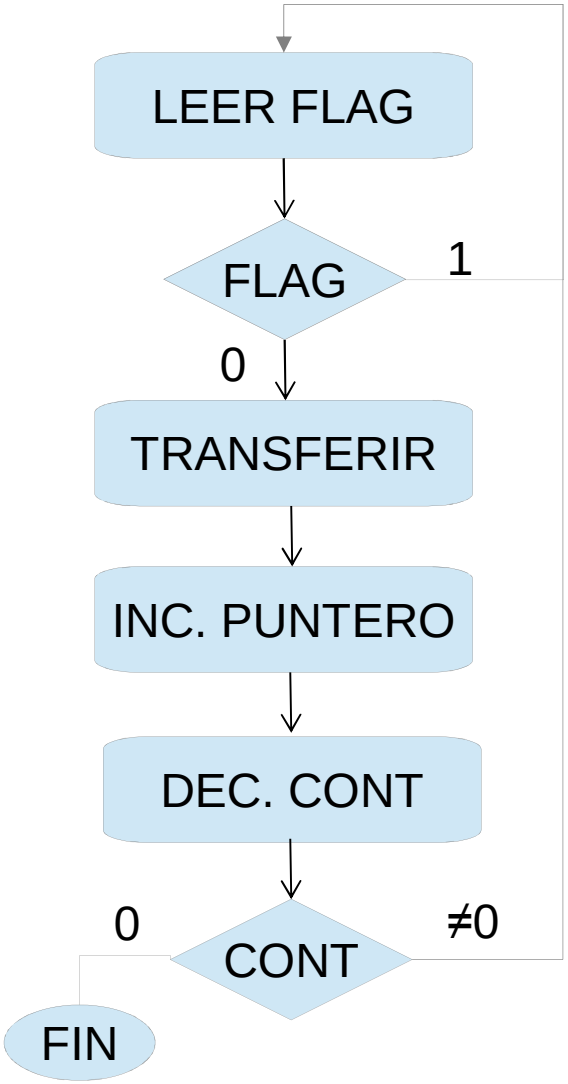
HANDSHAKING

- Puerto de salida con handshaking
  - Puerto de salida para datos
  - Puerto de entrada para bandera
  - Prendo la bandera con pulso ODSP del puerto de datos



HANDSHAKING – Ejemplo

- Subrutina
  - Transfiere N bytes a puerto salida en dirección 0x20
  - Handshake: Flag en bit menos significativo de puerto 0x20 de entrada
  - Parámetros:
    - B: cant. bytes N
    - HL: direcc. memoria datos.



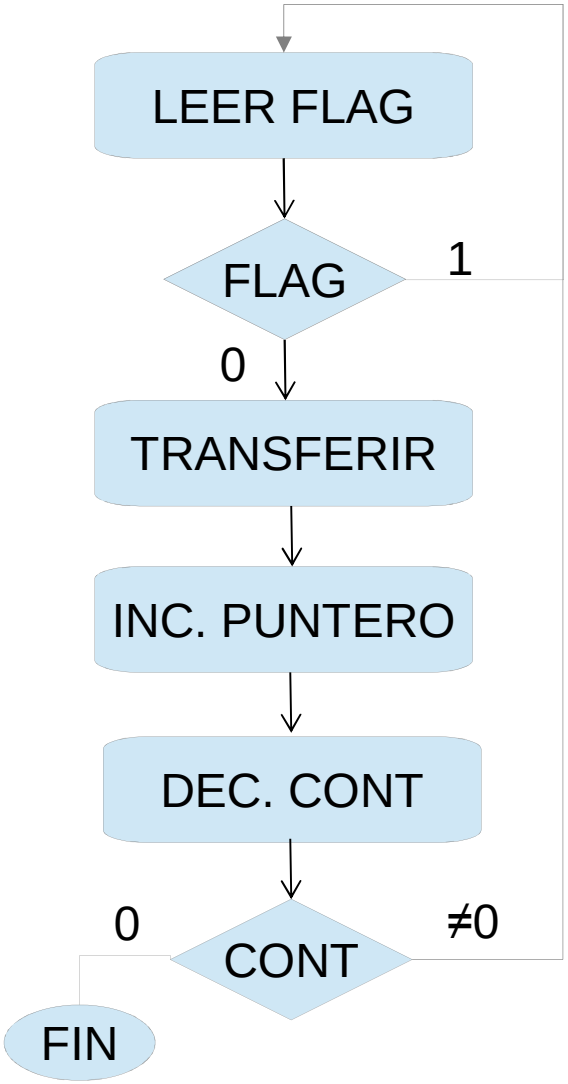
- Seudocódigo:

```
Para cont=N hasta 1 hacer
  Mientras flag=1 hacer nada
  Transferir
  Incrementar puntero a memoria
Fin_para
```

HANDSHAKING – Ejemplo

```
; SUBRUTINA escribo_bytes
; escribe n-bytes de memoria a un puerto
; ASUME:       B: n-bytes
;              HL: dir base en memoria
; DESTRUYE: A, B, HL
; DEVUELVE: nada

escribo_bytes:
      in A, (20H)
      bit 0, A
      jr nz, escribo_bytes
      ld a, (hl)
      out (20H), a
      inc hl
      djnz escribo_bytes
      ret
```



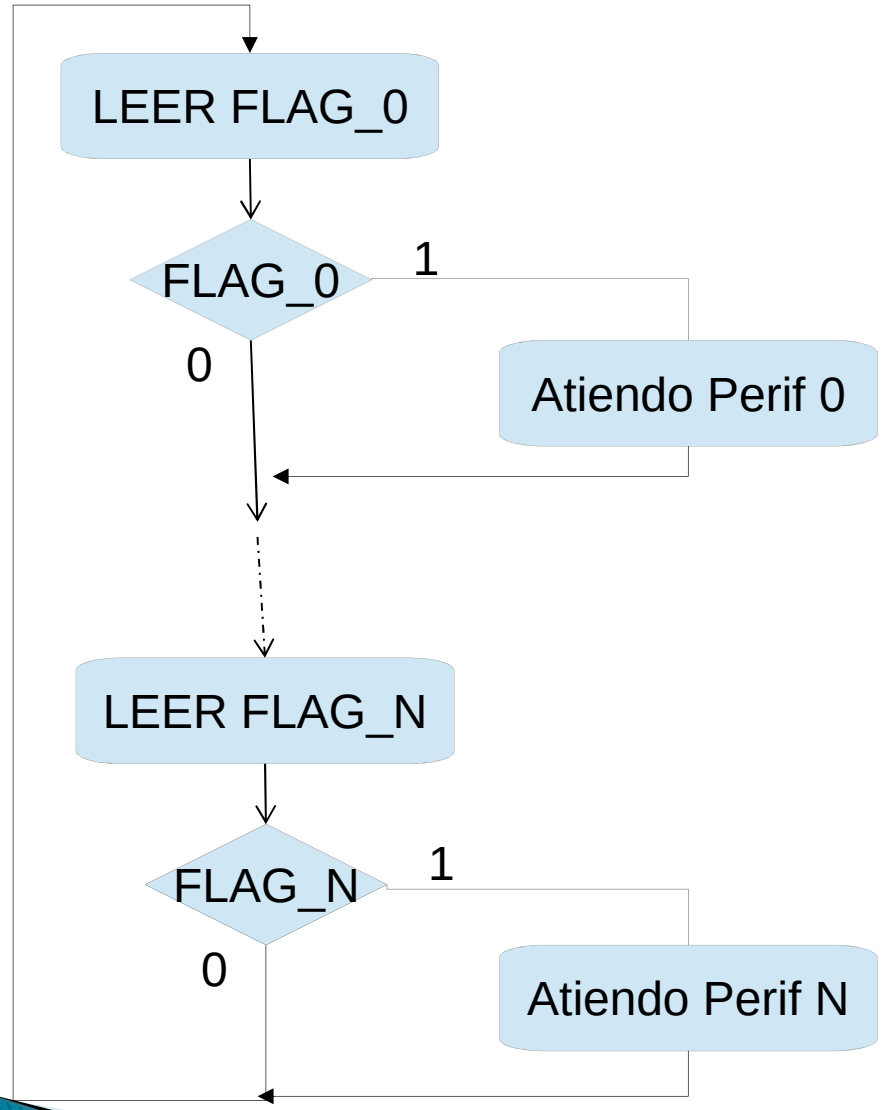
HANDSHAKING – Timeout

- ¿Qué pasa si el periférico no consume el dato?
  - Queda trancado en loop infinito
- Solución: límite de tiempo (timeout)
  - Implementación
    - Limitar iteraciones esperando flag.
    - Temporizadores hardware

POLLING

- ¿Cómo atender a varios dispositivos?
  - Ciclo consultando por turno bandera de estado de cada uno.
  - Si es necesario se llama a subrutina que lo atienda y al retornar se sigue el ciclo.
  - A este proceso se le llama POLLING.

POLLING



Tiempo de respuesta:

Es el tiempo entre que se activa la bandera y realizo la lectura o escritura correspondiente

Observción:

- El tiempo de respuesta varía dependiendo de donde se encuentre la rutina de Polling.
- Cuantos más periféricos, el tiempo de respuesta aumenta.
- Overhead

Mejora:

- Dar prioridades a determinados periféricos.