

PIO, HANDSHAKE, DMA y periféricos

Arquitectura de computadoras

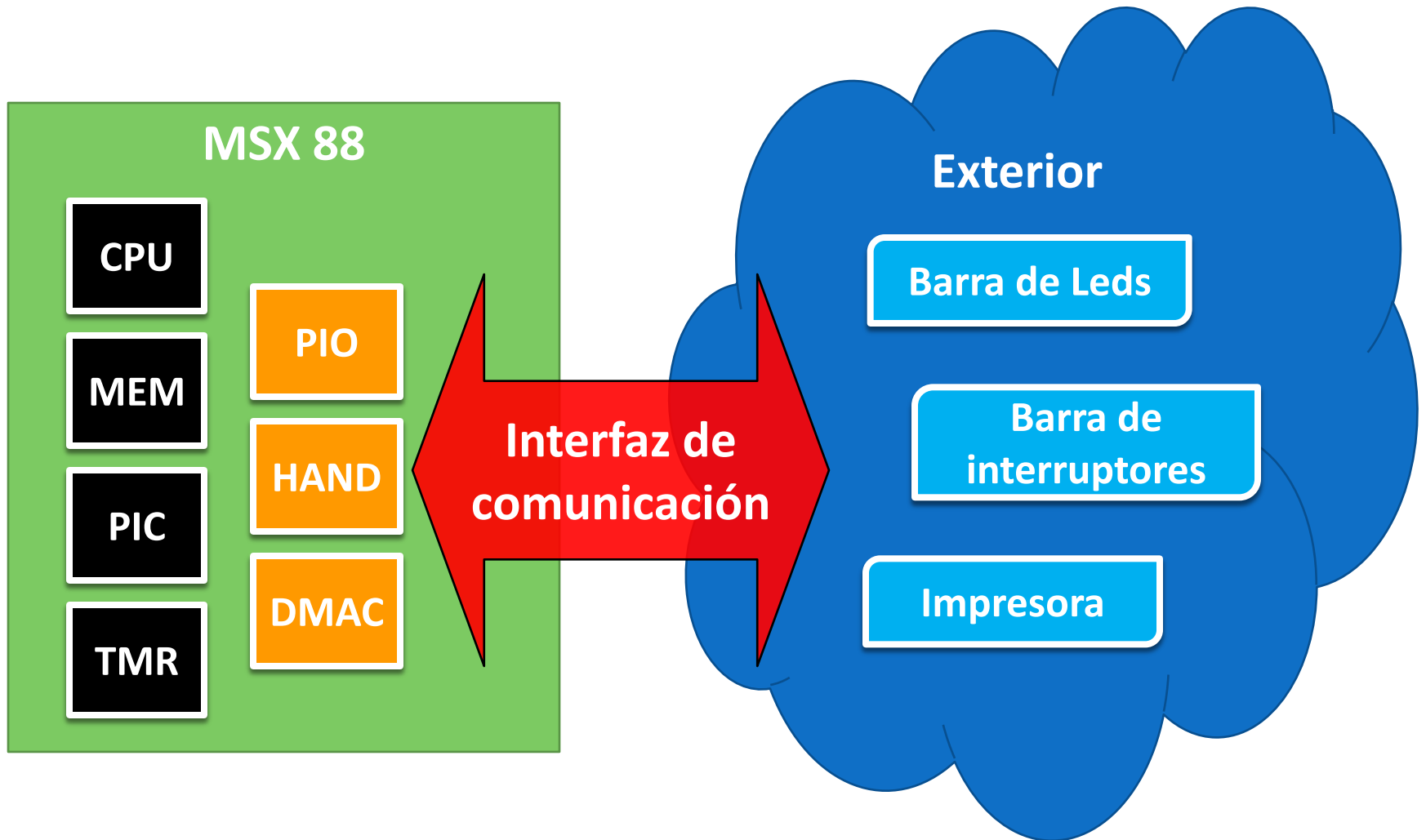
Periféricos

- Periféricos del MSX88 (Hardware /dispositivos que están “fuera” del MSX88):
 - Barra de Leds
 - Interruptores
 - Impresora
- Comunicación:
 - Necesitamos hardware /dispositivos internos (adicionales) para poder comunicarnos con hardware/dispositivos externos
 - Necesitamos entender como funciona cada dispositivo interno y cada periférico

Periféricos

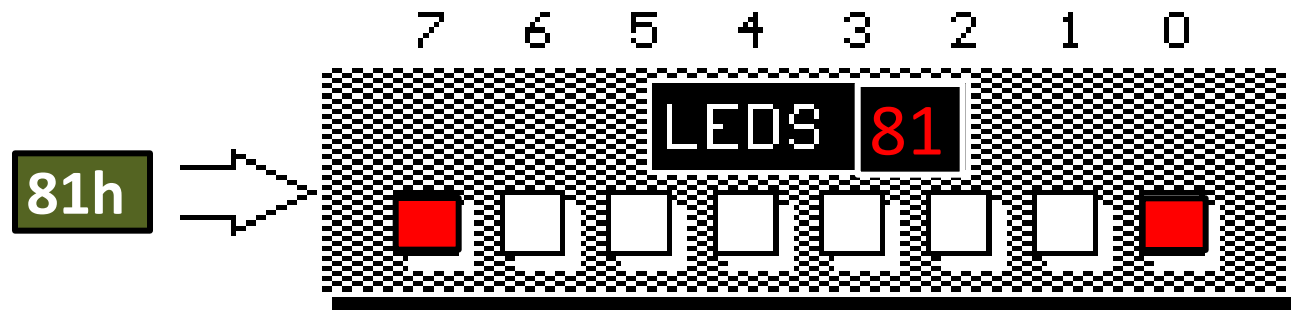
- Dispositivos que asisten al MSX88 para comunicarse con el mundo exterior:
 - PIO (Entrada/Salida Periférica): permite conectarnos con cualquier dispositivo externo.
 - Handshake: permite comunicarnos con una impresora de manera eficiente.
 - DMAC (controlador de acceso directo a memoria): permite hacer transferencias eficientes entre memoria y dispositivos

Periféricos



Periféricos – Barra de leds

- Es un grupo de 8 leds (diodos emisores de luz)
- Se conecta en el MSX88 a un puerto que permite controlar el estado de cada led
- Para encender un led se pone en 1 el bit deseado en el puerto



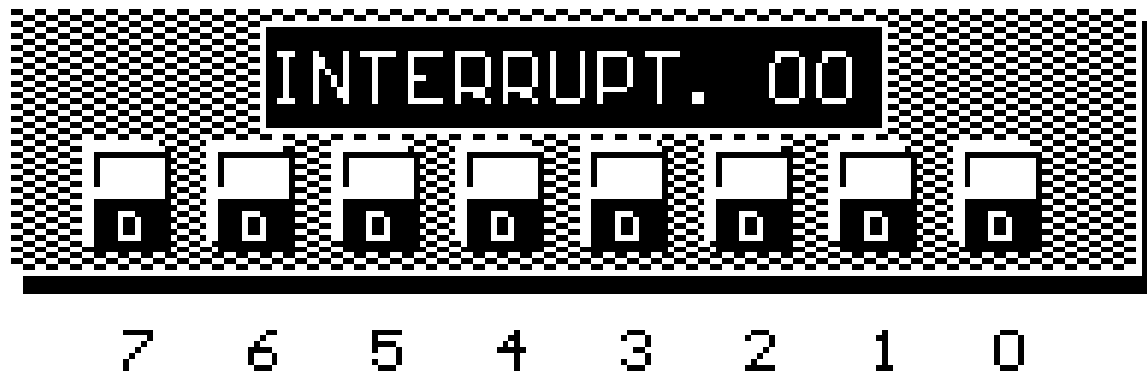
Si queremos encender el bit 7 y 0 y mantener apagados los demás hay que enviar a la barra de leds el valor 81h, 129 o 10000001b

Periféricos – Barra de leds

- La funcionalidad de los leds es didáctica
- En aplicaciones reales, con el hardware adecuado, con 1 bit es posible controlar dispositivos que tienen 2 estados (apagado/encendido):
 - Bombas de agua para riego y tanques
 - Cierre/apertura de puertas y ventanas
 - Encendido/apagado de luces
 - Activar/desactivar alarmas
 - Activar/desactivar refrigeración/calefacción
- Con mas bits o combinaciones de bits se pueden controlarse mas estados.

Periféricos – Microinterruptores

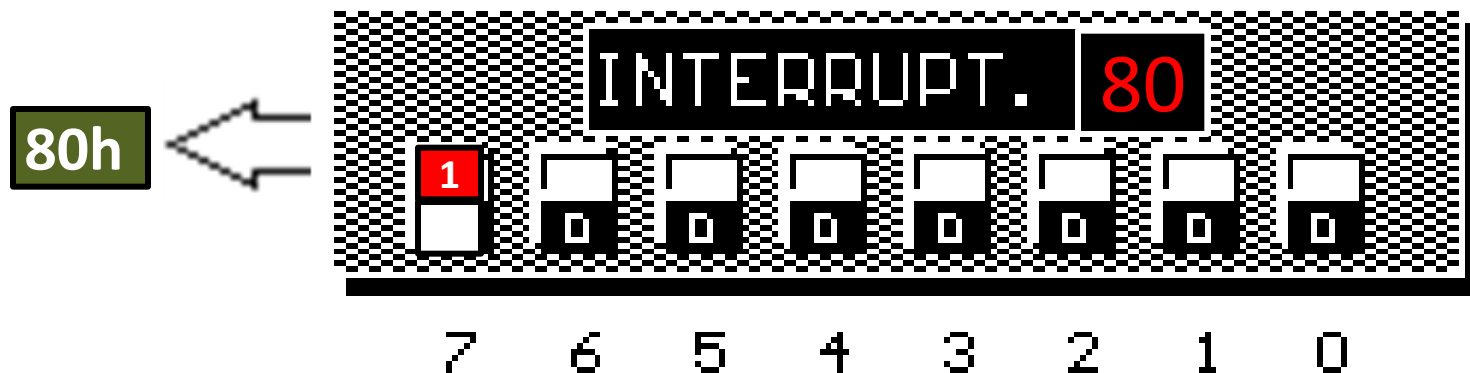
- Es una barra de 8 interruptores (botones con 2 estados)
- Se conecta en el MSX88 a un puerto que permite “senar” (saber el estado) los interruptores.
- Cuando se activa/desactiva un interruptor modifica el bit asociado en el puerto de entrada.



Periféricos – Microinterruptores

- Los interruptores son externos y para controlarlos es necesario ejecutar el comando en el simulador:
 - M{nro de bit} : invierte bit {nro de bit}
 - M{valor hexa}: establece todos los valores simultaneamente.

Ejecutamos el comando: M7



Periféricos – Microinterruptores

- La funcionalidad de los microinterruptores es didáctica
- En aplicaciones reales, con el hardware adecuado, con 1 bit es posible sensor dispositivos que tienen 2 estados (apagado/encendido):
- Aplicaciones reales:
 - Detectores de ventanas, puertas cerradas o abiertas
 - Detector de niveles bajo/alto para líquidos
 - Teclados
 - Detectores de presencia
 - Detectores de humo, gases

Periféricos – Impresora



- Características:
 - 8 líneas de entrada (**data**) para recibir un carácter ASCII
 - 1 línea de salida (**busy**) para indicar si esta ocupada o disponible para recibir un carácter e imprimirlo:
 - Línea en 1 ocupado, en 0 libre
 - 1 línea de entrada (**strobe**) para indicarle cuando hay un carácter para imprimir en data:
 - Toma carácter de **data** cuando pasa de 1 a 0
 - Tiene un cola de impresión (**buffer**) de 5 caracteres:
 - Cuando la cola está llena, la impresora activa la línea **busy** para indicar que no puede recibir caracteres.
 - Cuando esta lista para imprimir un carácter, lo saca del **buffer**, lo imprime y desactiva la línea de **busy**

Periféricos – Impresora

- Pseudo-código para enviar un carácter:

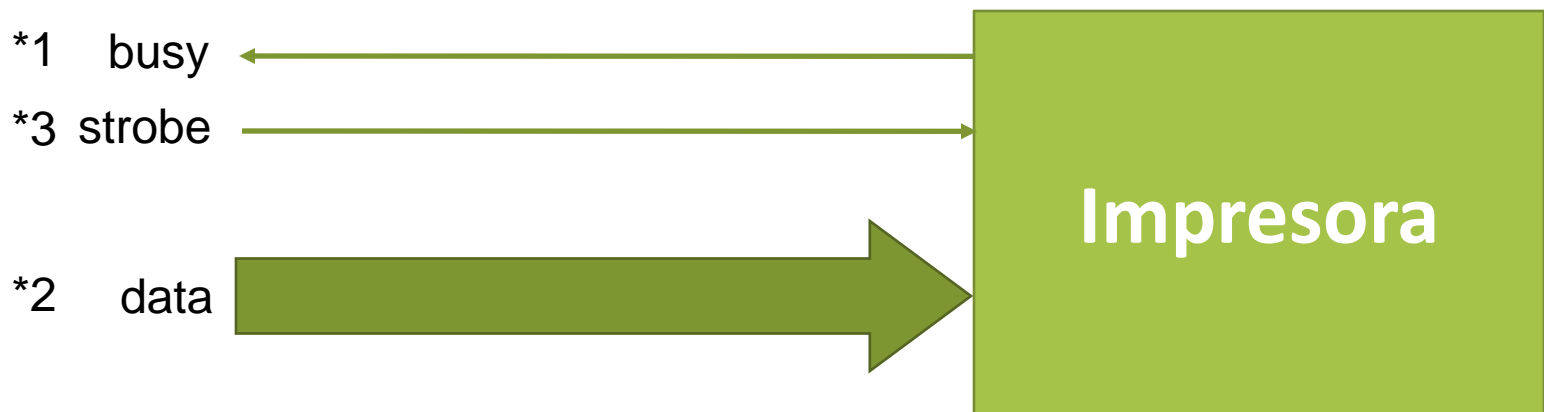
Mientras la línea **busy** este en 1 (*1):

Esperar sin hacer nada

Enviar a **data** el carácter a imprimir (*2)

Enviar un 1 a la línea **strobe** (*3)

Enviar un 0 a la línea **strobe** (*3)



Periféricos – PIO

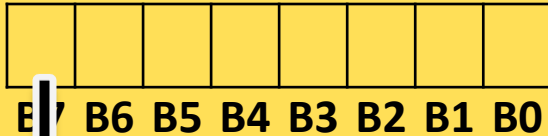
- Dispone de hasta 16 bits o líneas para comunicarse con periféricos (salen 16 “cables” al exterior)
- La dirección de cada línea es configurable. Puede usarse como entrada o como salida (no son simultáneas)
- Los 16 bits se distribuyen entre dos registros del dispositivo: PA (puerto A) y PB (puerto B)
- Tiene 2 registros adicionales donde se indica el sentido (entrada o salida) de cada uno de los bits:
 - CA (configuración de A) para cada bit de PA
 - CB (configuración de B) para cada bit de PB
 - Un bit en 0 configura como salida, un bit en 1 como entrada
 - La electrónica para que un bit sea de entrada o de salida es diferente. Por esto son necesarios los bits de configuración

Periféricos – PIO

- Se conecta al MSX88 a partir de la dirección 30H:
 - PA esta en la dirección 30H
 - PB esta en la dirección 31H
 - CA esta en la dirección 32H
 - CB esta en la dirección 33H
- Podemos conectar la barra de leds e interruptores :
 - El comando del simulador “P1 C0” muestra esta configuración
 - En PA se conectan los interruptores y en PB los leds
- Podemos conectar la impresora:
 - El comando del simulador “P1 C1” muestra esta configuración
 - En el bit 0 de PA se conecta la línea **busy**
 - En el bit 1 de PA se conecta la línea **strobe**
 - En PB se conectan las líneas de datos

PIO - Configuración de 1 bit

PA – Puerto A



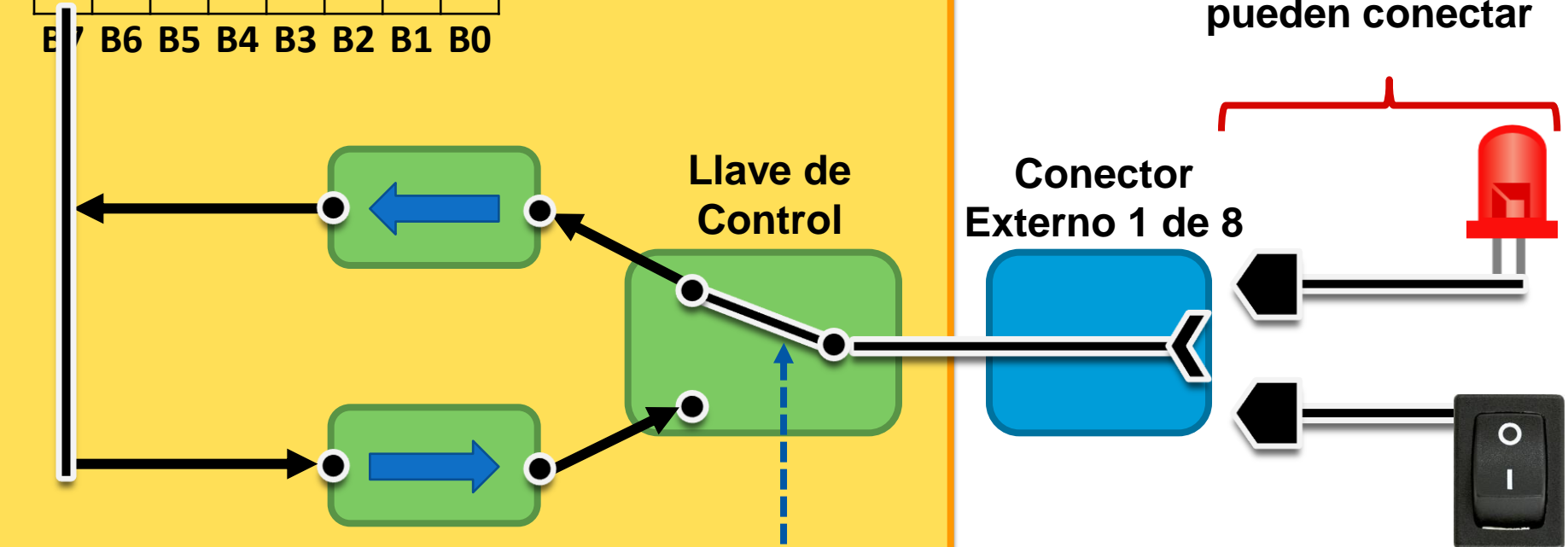
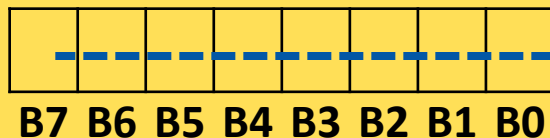
PIO

Llave de Control

Conector Externo 1 de 8

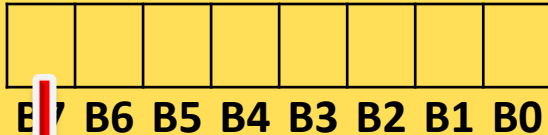
Dispositivos que se pueden conectar

CA – Config. Puerto A



PIO - Configuración de 1 bit – 1 LED

PA – Puerto A



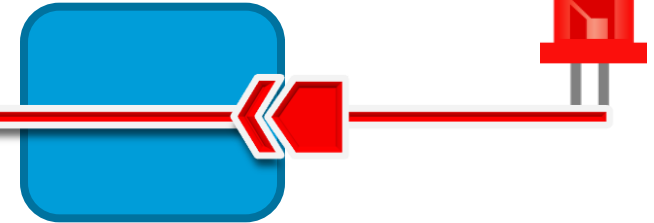
PIO

Llave de Control

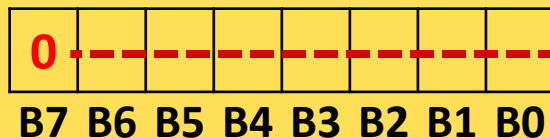
Luego de configurado CA:

- ✓ Poner 1 en bit 7 de PA enciende el LED
- ✓ Poner 0 en bit 7 de PA apaga el LED

Conector Externo 1 de 8



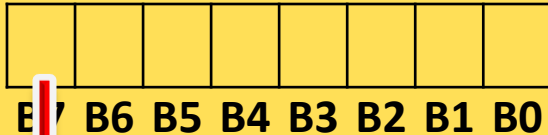
CA – Config. Puerto A



El bit 7 de CA habilita la electrónica necesaria para transmitir al exterior un 1 o 0 a través del bit 7 del PA

PIO - Configuración de 1 bit – 1 Botón

PA – Puerto A

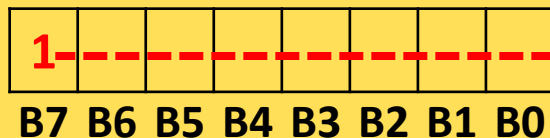


PIO

Llave de Control

Conector Externo 1 de 8

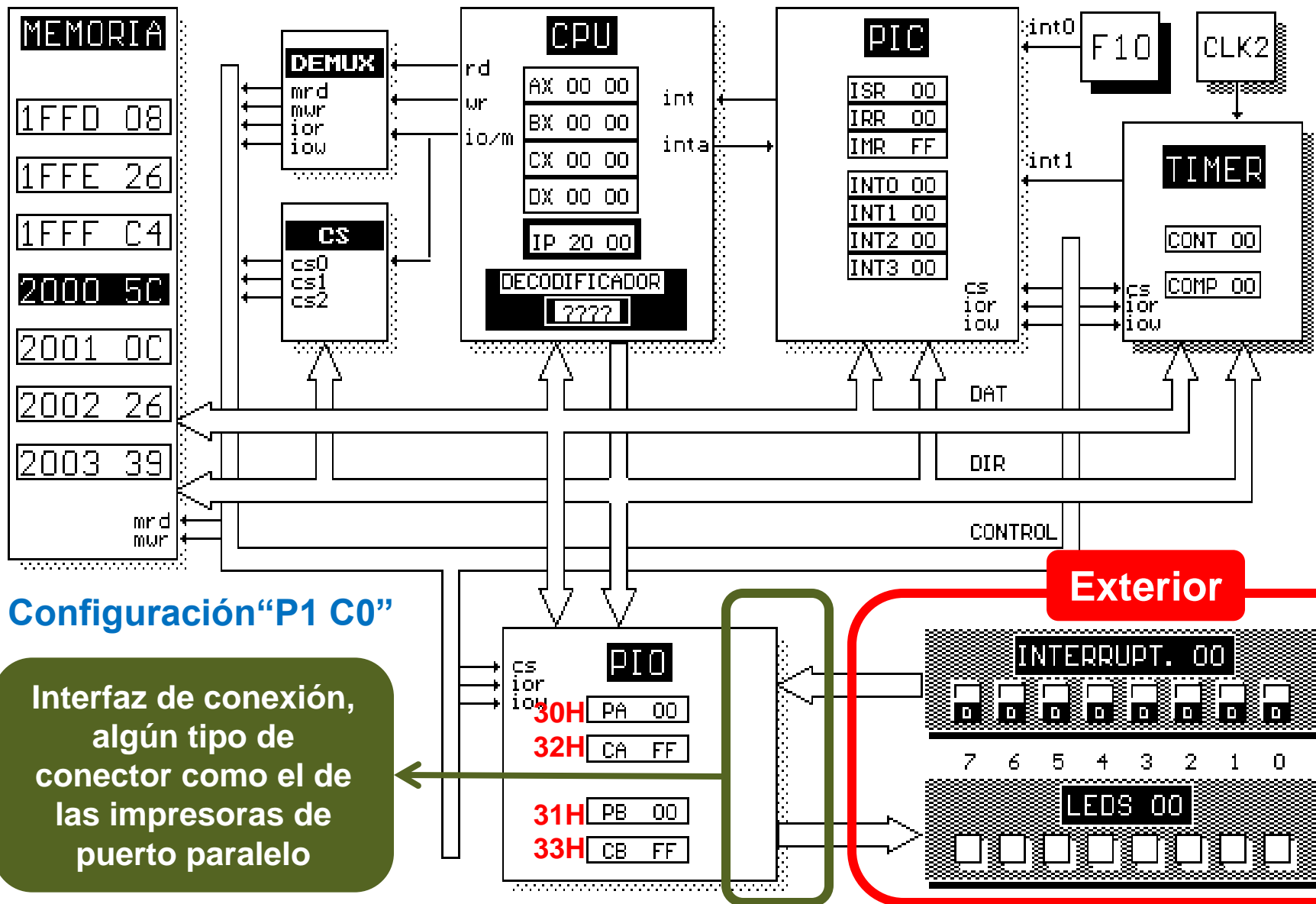
CA – Config. Puerto A



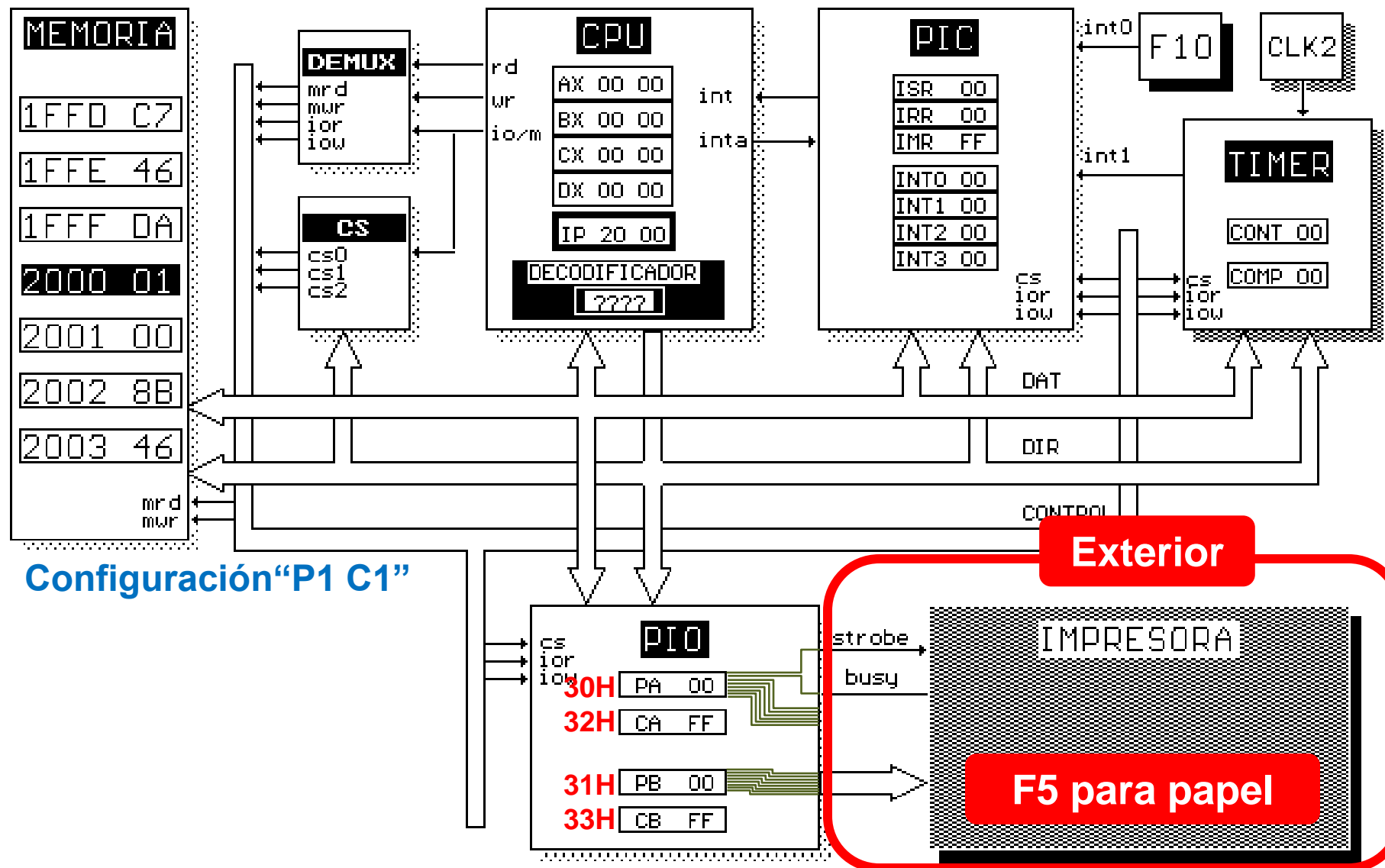
- Luego de configurado CA:
- ✓ Leer un 1 en bit 7 indica interruptor presionado
 - ✓ Leer un 0 en bit 7 indica interruptor sin presionar

El bit 7 de CA habilita la electrónica necesaria para leer desde exterior un 1 o 0 a través del bit 7 del PA

Periféricos – PIO + Leds + Interruptores



Periféricos – PIO + Impresora



Ejercicio 1 – controlar leds con interruptores

```
PA EQU 30H
PB EQU 31H
CA EQU 32H
CB EQU 33H
```

```
ORG 2000H
    MOV AL, 0FFH ; PA entradas (Microconmutadores)
    OUT CA, AL
    MOV AL, 0    ; PB salidas (Luces)
    OUT CB, AL
POLL: IN AL, PA
    OUT PB, AL
    JMP POLL
END
```

Ejercicio 4 – Imprimir texto con impresora y PIO

```
PIO EQU 30H      ;PIO =PA  PIO+1=PB
                  ;PIO+2=CA  PIO+3=CB
```

```
1. ORG 1000H
2. MSJ D B "ARQUITECTURA DE "
3. DB "COMPUTADORAS"
4. FIN DB ?
5.
6. ORG 2000H
7. MOV AL, 0FDH ; INIC. PIO IMPRESORA
8. OUT PIO+2, AL ; CONF. STROBE Y BUSY
9. MOV AL, 0
10. OUT PIO+3, AL ; CONF. DATOS
11. IN AL, PIO ; LEE STROBE Y BUSY
12. AND AL, 0FDH ; 253 o 11111101b
13. OUT PIO, AL ; FIN INICIALIZACION
```

```
14. MOV BX, OFFSET MSJ
15. MOV CL, OFFSET FIN-OFFSET MSJ
16. POLL: IN AL, PIO ; LEE STROBE Y BUSY
17. AND AL, 1 ; SOLO DEJA BUSY
18. JNZ POLL
19. MOV AL, [BX] ; RECUPERA CARACTER
20. OUT PIO+1, AL ; ENVIA A DATA (PB)
21. IN AL, PIO ; PULSO 'STROBE'
22. OR AL, 02H ; PONE 1 en bit 2
23. OUT PIO, AL
24. IN AL, PIO
25. AND AL, 0FDH ; PONE 0 en bit 2
26. OUT PIO, AL ; FIN PULSO
27. INC BX
28. DEC CL
29. JNZ POLL
30. INT 0
31. END
```

¿Qué es poll o polling?

Denominamos así a una operación que continuamente espera que cambie una condición en un dispositivo

¿Cuál es el problema y cómo lo evitamos?

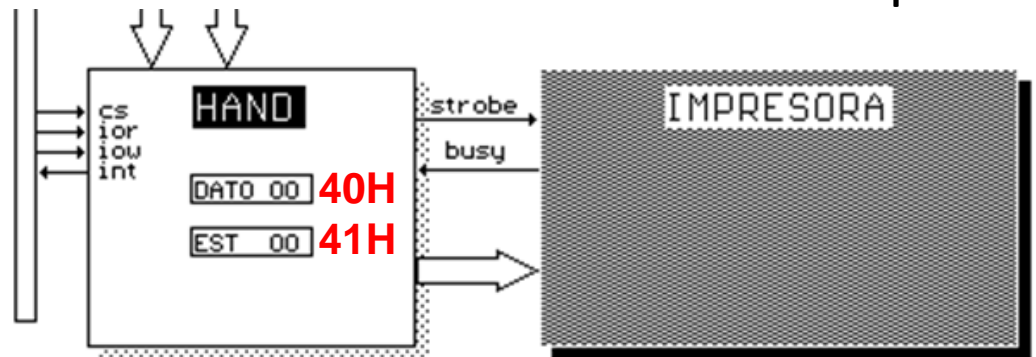
El problema es la ineficiencia. Puede evitarse cuando el dispositivo genera una interrupción cuando la condición que esperamos cambia

Periféricos – HANDSHAKE

- Dispositivo especializado para comunicarse con impresoras
- Ventajas:
 - Genera el pulso en la línea de **strobe** de forma automática
 - Puede generar interrupciones:
 - Puede generar una interrupción cuando la impresora puede recibir un carácter. Con este mecanismo podemos escribir un carácter sin necesidad de **esperar** o perder tiempo consultando que este lista (evita **polling**)
- Desventajas:
 - Solo permite la comunicación con la impresora, mientras que la PIO puede comunicarse con cualquier dispositivo
- Esta conectado a partir de la dirección 40H
- Tiene solo 2 registros
- Conectado a PIC a través de la línea **Int2**

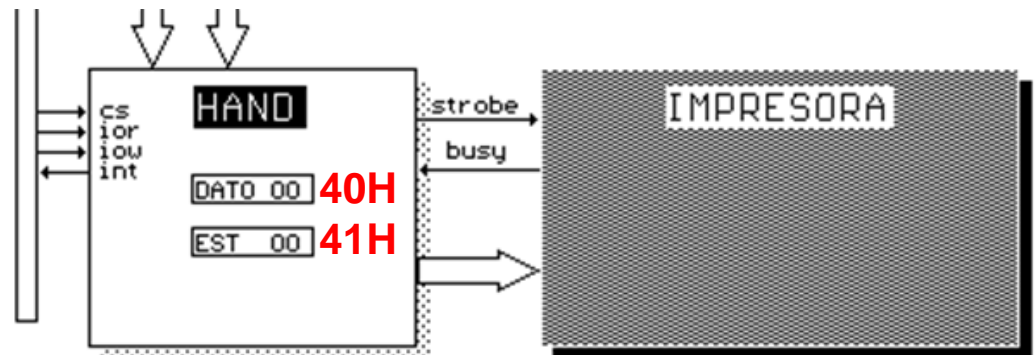
Periféricos – HANDSHAKE

- Registro de datos (data):
 - Ubicado en la dirección 40H:
 - 8 bits para almacenar un código ASCII
 - Cada vez que se escribe un valor, el hadshake espera que la impresora este libre (línea busy=0) y luego genera el pulso strobe
 - Si estamos usando interrupciones es seguro escribir el carácter para que lo envíe a la impresora
 - Si NO estamos usando interrupciones hay que verificar que el bit busy este en 0. La escritura de un nuevo carácter hace que se pierda el anterior

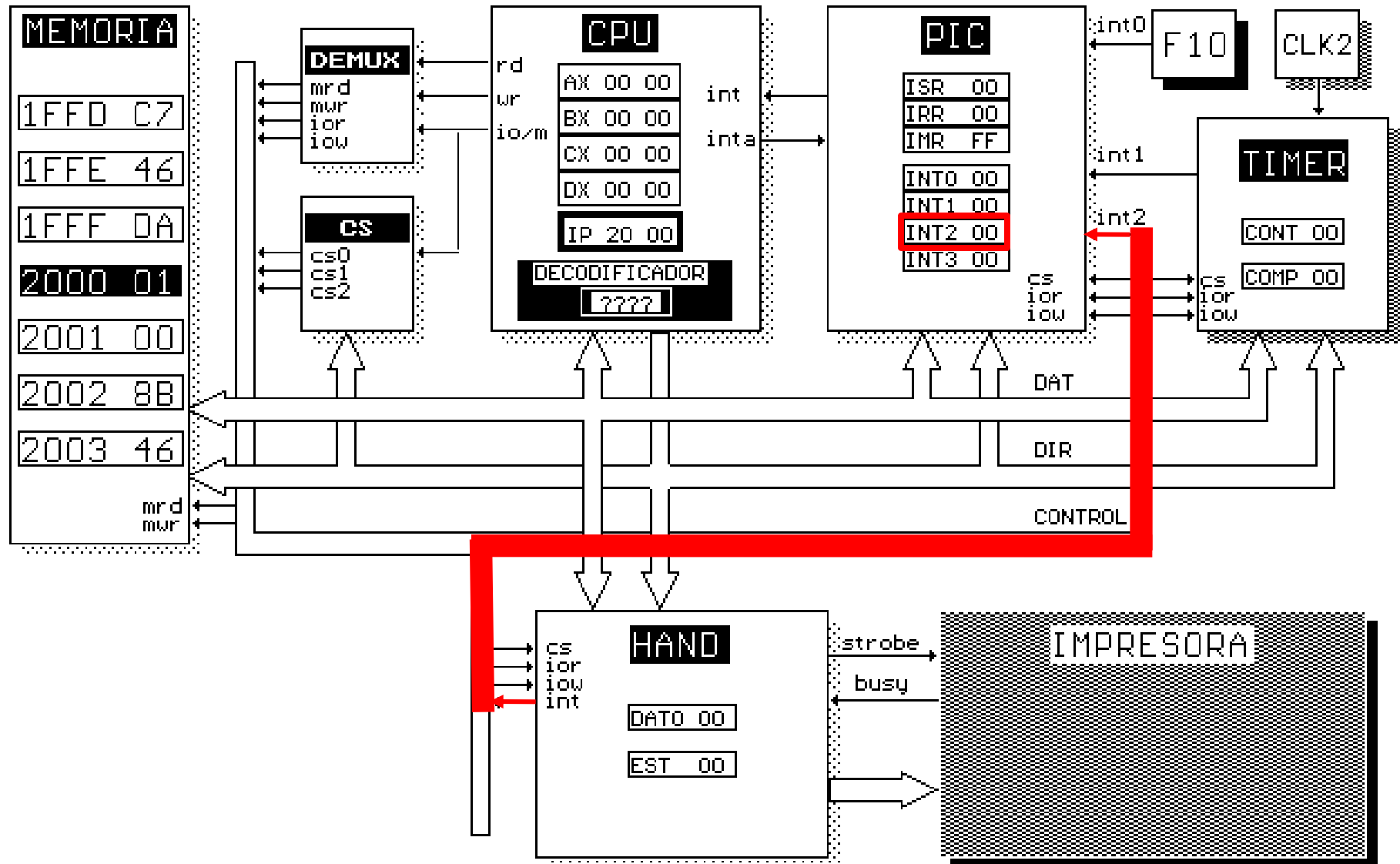


Periféricos – HANDSHAKE

- Registro de estado y control(status):
 - Ubicado en la dirección 41H
 - Funciones de los bits:
 - Bit 0: conectado a línea **busy** de la impresora (solo útil para polling)
 - Bit 1: conectado a línea **strobe** de la impresora
 - Bit 2-6: sin uso
 - Bit 7: Control de interrupción:
 - Valor en 1: habilita interrupciones
 - Valor en 0: deshabilita interrupciones
- El comando “P1 C2” muestra esta configuración del Handshake con la impresora



Periféricos – Handshake + Impresora



Ej 7 - Imprimir con Handshake sin interrupciones

HAND EQU 40H

ORG 1000H

MSJ DB "FACULTAD DE "

DB "INFORMATICA"

FIN DB ?

ORG 2000H

```
1.      IN  AL, HAND+1          ; LEE STATUS DE HAND
2.      AND AL, 7FH             ; 7FH = 127 = 01111111B PONE A 0 BIT 7
3.      OUT HAND+1, AL         ; ESCRIBE STATUS DE HAND
4.      MOV BX, OFFSET MSJ      ; PUNTERO A TEXTO
5.      MOV CL, OFFSET FIN-OFFSET MSJ
6.POLL: IN AL, HAND+1          ; LEE STATUS DE HAND
7.      AND AL, 1               ; DEJA VALOR DE BIT 0
8.      JNZ POLL
9.      MOV AL, [BX]            ; RECUPERA CARACTER
10.     OUT HAND, AL            ; ESCRIBE STATUS DE HAND
11.     INC BX
12.     DEC CL                  ; FALTA 1 MENOS
13.     JNZ POLL
14.     INT 0
15.     END
```

Status	7	6	5	4	3	2	1	0
41H	int	X	X	X	X	X	str	busy

Ej 8 - Imprimir con Handshake con interrupciones

```
PIC EQU 20H
HAND EQU 40H
N_HND EQU 10
```

Status	7	6	5	4	3	2	1	0
41H	int	X	X	X	X	X	str	busy

```
ORG 40 ORG 1000H
IP_HND DW RUT_HND
```

```
ORG 1000H
MSJ DB "UNIVERSIDAD "
DB "NACIONAL DE LA PLATA"
FIN DB ?
```

```
ORG 3000H ; RUTINA DE INTERRUPCION
```

```
1. RUT_HND: PUSH AX
2.     MOV AL, [BX]
3.     OUT HAND, AL ; CARÁCTER A DATOS
4.     INC BX      ; PROXIMO
5.     DEC CL      ; DESCUENTA
6.     MOV AL, 20H ; AVISA A PIC
7.     OUT PIC, AL
8.     POP AX
9.     IRET
```

```
ORG 2000H
```

```
1.     MOV BX, OFFSET MSJ
2.     MOV CL, OFFSET FIN-OFFSET MSJ
3.     CLI          ; BLOQUEA INTERRUPCIONES
4.     MOV AL, 0FBH ; FBH = 251 = 11111011B
5.     OUT PIC+1, AL ; ESCRIBE IMR
6.     MOV AL, N_HND ; POS. 10 DE VECT INT
7.     OUT PIC+6, AL ; REGISTRO INT2 DE PIC
8.     MOV AL, 80H   ; 80H = 128 = 10000000B
9.     OUT HAND+1, AL ; REG ESTADO (CON INT)
10.    STI          ; HABILITA INTERRUPCIONES
11. LAZO: CMP CL, 0 ; CUANDO TERMINA?
12.    JNZ LAZO
13.    IN AL, HAND+1 ; LEE ESTADO HAND
14.    AND AL, 7FH   ; 7FH = 127 = 01111111B
15.    OUT HAND+1, AL ; REG ESTADO (SIN INT)
16.    INT 0
      END
```

DMA - ¿Por qué?

Supongamos que queremos transferir 200 bytes de una dirección de memoria a otra. Tenemos:

- AX= dirección origen
- DX=dirección destino
- CL=200

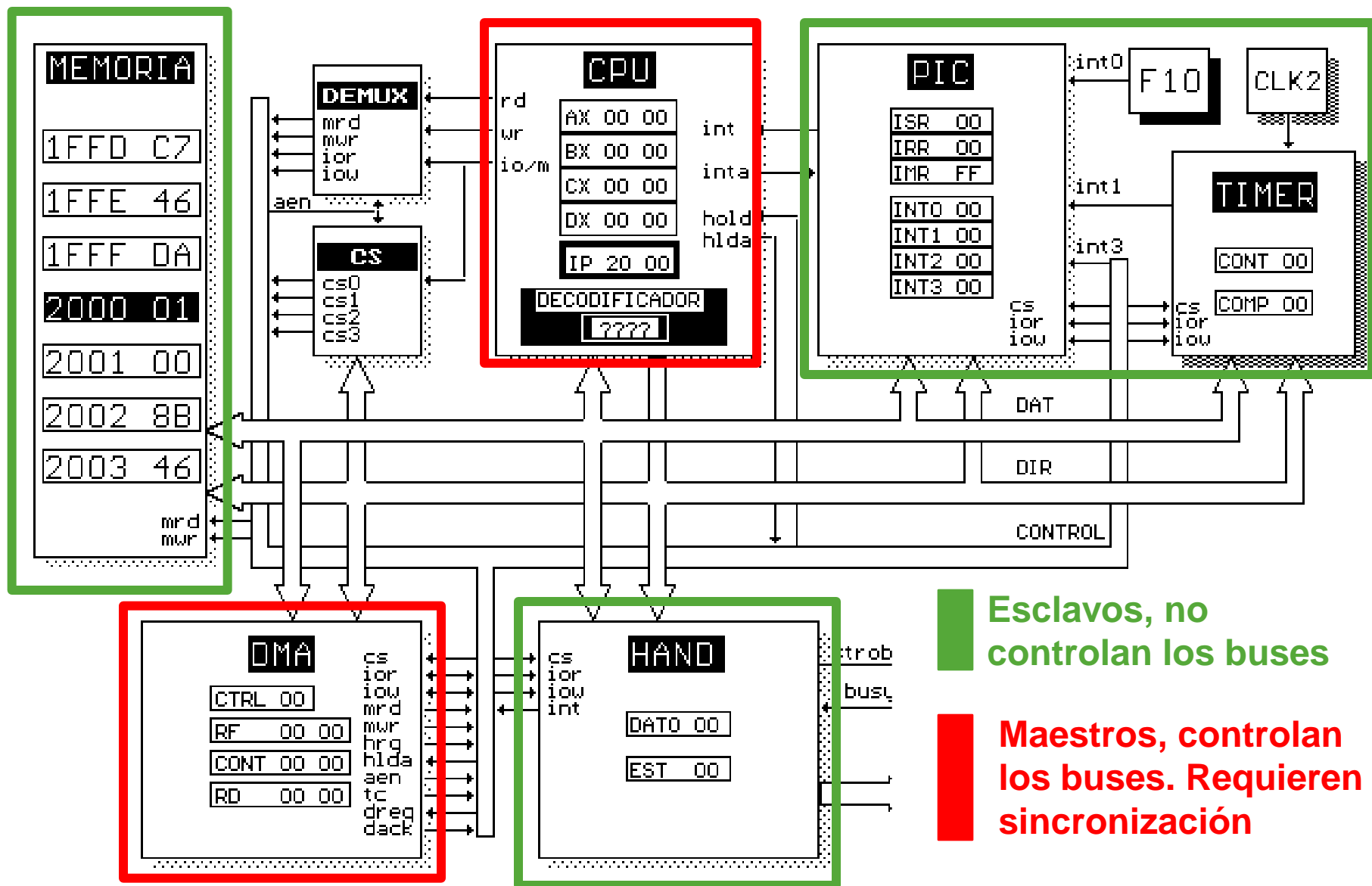
			Accesos
Otro_byte:	MOV BX, AX	;Dirección actual de origen	2
	MOV CH, [BX]	;Recupera byte	2+1
	INC AX	;Apunta a próximo byte	2
	MOV BX, DX	;Dirección actual de destino	2
	MOV [BX], CH	;transfiere 1 byte a destino	2+1
	INC DX	;Apunta a próximo byte	2
	DEC CL	;Decrementa bytes faltantes	2
	JNZ otro_byte		2

¿Cuántos accesos a memoria serían necesarios para transferir 1 byte? → 2 18

DMA - Características

- Es un dispositivo que permite transferir de manera eficiente bloques de datos a memoria y a dispositivos
- Realiza accesos a memoria y a otros dispositivos como la CPU, es “una CPU especializada en transferencia”
- Comparte los buses con la CPU, por lo que se necesitan señales especiales para que sincronicen los accesos. Los pasos serían:
 - DMAC pide a la CPU acceso a los buses
 - La CPU acepta y se desconecta de los buses (no ejecuta mas instrucciones)
 - El DMAC toma el control de los buses y realiza la transferencia
 - El DMAC indica a la CPU que termino la transferencia
 - La CPU toma el control (reinicia la ejecución de instrucciones)

DMA - Características



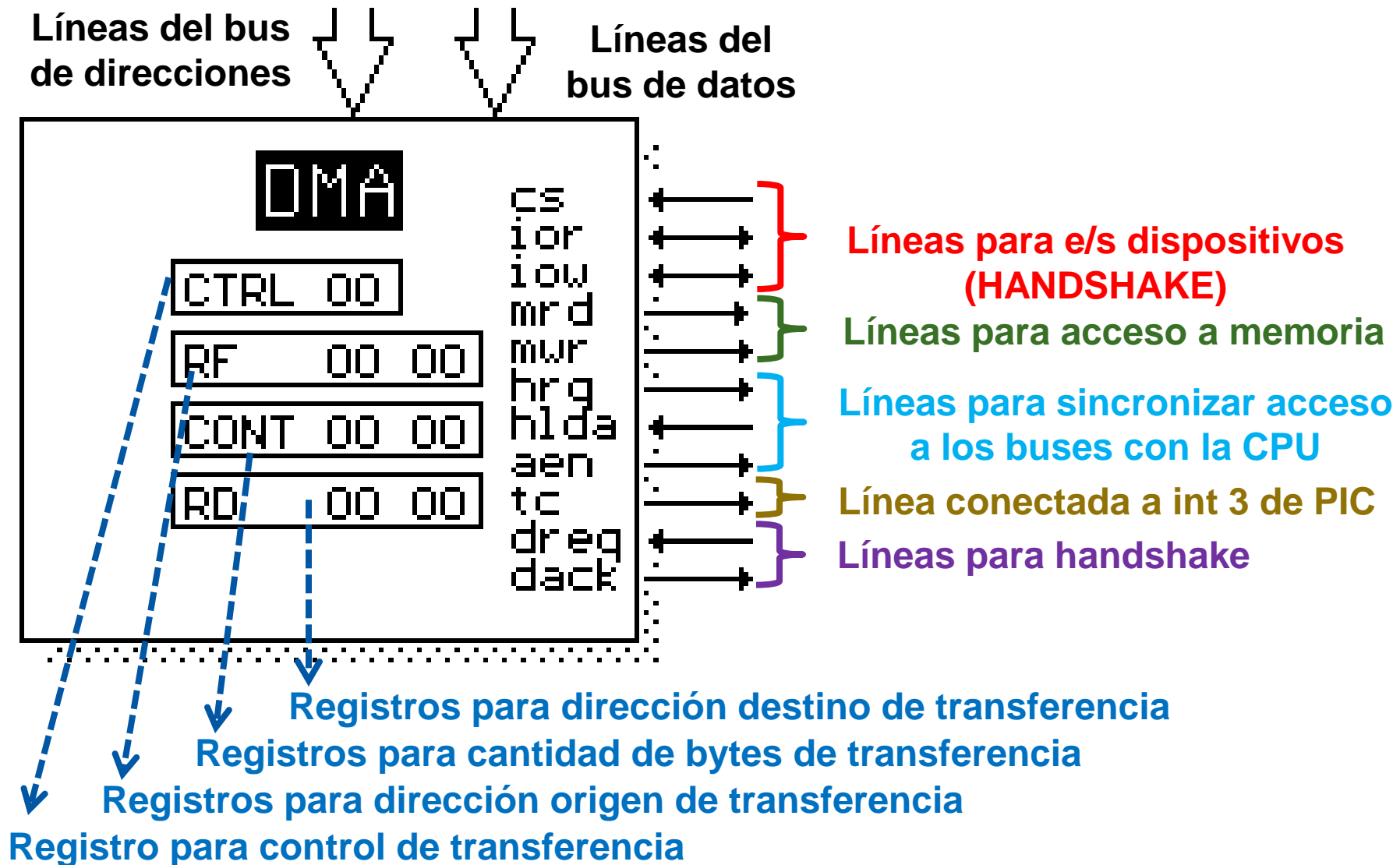
DMA - Características

- Tipos de transferencias:
 - Memoria → Memoria
 - Memoria → Periférico
 - Periférico → Memoria
- Modos de Transferencias:
 - Por bloque o ráfagas (burst): acuerda con la CPU para generar una transferencia completa, sin detenerse
 - Por demanda o robo de ciclos: transfiere el bloque de a un byte por vez. Se alterna con la CPU entre byte y byte (La CPU ejecuta instrucción, DMA transfiere byte, CPU ejecuta instrucción, DMA transfiere byte, etc.
- Al transferir el último byte genera una interrupción para que el programador detenga o re programe otra transferencia

DMA – Características MSX88

- Para las transferencias Memoria-Memoria requiere:
 - Una dirección origen
 - Una dirección destino
 - Una cantidad de bytes a transferir
 - Modo en el que se va a transferir (**Bloque** o Demanda)
- Para las transferencia Memoria-Periférico solo admite el Handshake (impresora) porque esta conectado (cableado) físicamente. Requiere:
 - Una dirección origen
 - Una cantidad de bytes a transferir
 - Modo en el que se va a transferir (Bloque o **Demanda**)
- Transferencias Periférico-Memoria no admite. ¿Por qué?
 - Porque el handshake no recibe datos de la impresora





DMA - Conexión y registros



DMA - Conexión

Sin DMA



-  Pedido de datos por interrupción
-  Envío de datos para impresión
-  Rol activo en la transferencia
-  Fin de transferencia por interrupción

Con DMA

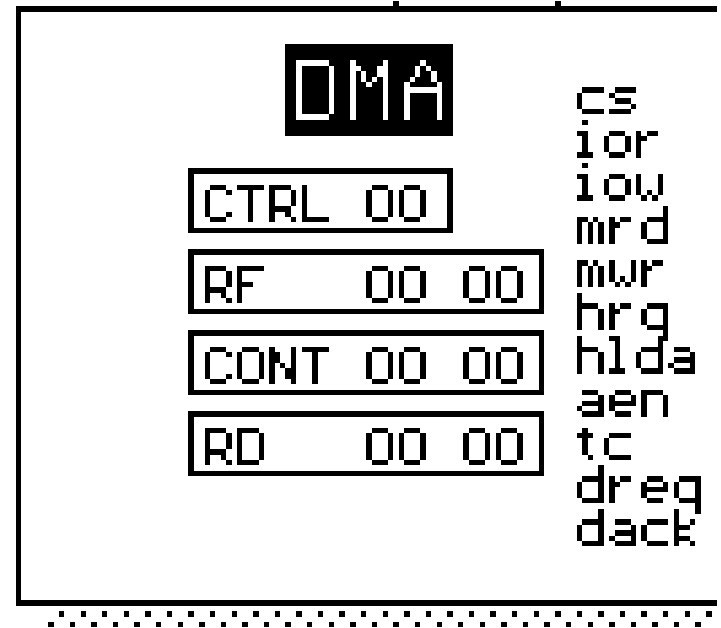


DMA – Conexión y registros

- Conectado a PIC a través de la línea **Int3**
- El comando del simulador “P1 C3” muestra esta configuración DMA + Handshake + impresora
- Ubicado en la dirección 50H
- Tiene 8 Registros

DMA - Registros

- Dirección fuente (origen), 16 bits (8+8):
 - RFL (50H)
 - RFH (51H)
- Cantidad a transferir, 16 bits (8+8):
 - CONTL (52H)
 - CONTH (53H)
- Dirección destino, 16 bits (8+8):
 - RDL (54H)
 - RDH (55H)
- Control, establece como se hace la transferencia, 8 bits
 - CONTROL (56H)
- Inicio de transferencia, 8 bits:
 - Arranque (57H): solo se necesita leer o escribir el registro con cualquier valor



DMA – Registro de control

7	6	5	4	3	2	1	0
TC	X	X	X	MT	ST	TT	Stop

- Stop (detenido):
 - Bit=1 → Transferencia detenida
 - Bit=0 → Transferencia en curso
 - escribir un 1 detiene la transferencia, escribir 0 no tiene efecto. Para reiniciar la transferencia es necesario leer o escribir el registro de arranque
- TT(Tipo de Transferencia):
 - Bit=0 → Periférico-Memoria o Memoria-Periférico
 - Bit=1 → Memoria-Memoria
- ST (Sentido de transferencia, restringido a Tipo cuando es 0):
 - Bit=0 → Periférico-Memoria
 - Bit=1 → Memoria-Periférico
- MT (Modo de transferencia):
 - Bit=0 → Bajo Demanda
 - Bit=1 → Por Bloque
- Bits 4 a 6 no se usan
- TC(Transferencia finalizada, solo lectura, genera interrupción si esta habilitada):
 - Bit=0 → Transferencia no finalizada
 - Bit=1 → Transferencia finalizada

Ejercicio 10

DMA. Transferencia de datos memoria-memoria.

Escribir un programa que copie una cadena de caracteres almacenada a partir de la dirección 1000H en otra parte de la memoria, utilizando el DMAC en modo de transferencia por bloque (ráfaga).

La cadena original se debe mostrar en la pantalla de comandos antes de la transferencia. Una vez finalizada, se debe visualizar en la pantalla la cadena copiada para verificar el resultado de la operación.

Ejecutar el programa en la configuración P1 C3.

Ejercicio 10

```
PIC EQU 20H
DMA EQU 50H
N_DMA EQU 20
```

ORG 80

```
IP_DMA DW RUT_DMA.
```

ORG 1000H

```
MSJ DB "FACULTAD DE"
      DB " INFORMATICA"
FIN DB ?
NCHAR DB ?
```

ORG 1500H

```
COPIA DB ?
```

; rutina atencion interrupción del CDMA

ORG 3000H

```
RUT_DMA: MOV AL, 0FFH
          OUT PIC+1, AL
          MOV BX, OFFSET COPIA
          MOV AL, NCHAR
          INT 7
          MOV AL, 20H
          OUT PIC, AL
          IRET
```

Muestra
mensaje
transferido

Cantidad
de bytes a
transferir

Dir destino
del bloque

ORG 2000H

```
CLI
MOV AL, N_DMA
OUT PIC+7, AL
MOV AX, OFFSET MSJ
OUT DMA, AL
MOV AL, AH
OUT DMA+1, AL
MOV AX, OFFSET FIN-OFFSET MSJ
OUT DMA+2, AL
MOV AL, AH
OUT DMA+3, AL
MOV AX, OFFSET COPIA
OUT DMA+4, AL
MOV AL, AH ;
OUT DMA+5, AL
MOV AL, 0AH
OUT DMA+6, AL
MOV AL, 0F7H
OUT PIC+1, AL
STI
MOV BX, OFFSET MSJ
MOV AL, OFFSET FIN-OFFSET MSJ
MOV NCHAR, AL
INT 7
MOV AL, 7H
OUT DMA+7, AL
INT 0
END
```

Configura INT3 del PIC

Dir origen
del bloque

Transferencia mem a
mem por bloque/ráfaga

IMR = 1111 0111

Muestra
mensaje
original

Inicia transferencia

Inmediatamente luego de ejecutar
la instrucción OUT, la CPU accede
al pedido del DMA

DMAC

50H	RFL	00H
51H	RFH	10H
52H	ContL	00H
53H	ContH	00H
54H	RDL	00H
55H	RDH	15H
56H	Control	1000 1010
57H	Arranque	0000 0111

MT=por ráfaga

00001010

TT=mem/mem

Ejercicio 11

DMA. Transferencia de datos memoria-periférico.

Escribir un programa que transfiera datos desde la memoria hacia la impresora sin intervención de la CPU, utilizando el DMAC en modo de transferencia bajo demanda (*robo de ciclo*).

Ejercicio 11

```
PIC      EQU    20H
HAND     EQU    40H
DMA      EQU    50H
N_DMA    EQU    20
```

ORG 80

```
IP_DMA   DW     RUT_DMA
```

ORG 1000H

```
MSJ      DB     " INFORMATICA"
FIN      DB     ?
FLAG     DB     0
```

; rutina atención interrupción del CDMA

ORG 3000H

```
RUT_DMA: MOV AL, 0
          OUT HAND+1, AL } Deshabilita  
                      } interrupción del HAND  
          MOV FLAG, 1 Indica fin de lazo  
          MOV AL, 0FFH } IMR = 1111 1111  
          OUT PIC+1, AL  
          MOV AL, 20H  
          OUT PIC, AL  
          IRET
```

ORG 2000H

```
CLI
MOV AL, N_DMA } Configura INT3 del PIC
OUT PIC+7, AL
MOV AX, OFFSET MSJ } Dir origen  
OUT DMA, AL         } del bloque  
MOV AL, AH
OUT DMA+1, AL
MOV AX, OFFSET FIN-OFFSET MSJ } Contidad de  
OUT DMA+2, AL             } bytes a  
MOV AL, AH                 } transferir  
OUT DMA+3, AL
MOV AL, 4 } CTRL = 0000 0100
OUT DMA+6, AL
MOV AL, 0F7H } IMR = 1111 0111
OUT PIC+1, AL
OUT DMA+7, AL Inicia transferencia
MOV AL, 80H } HAND por interrupción
OUT HAND+1, AL
STI
```

```
LAZO: CMP FLAG, 1
      JNZ LAZO
      INT 0
      END
```

