Interrupciones por software y hardware

Arquitectura de computadoras

Interrupciones por software

- Una interrupción (por software):
  - ofrece un mecanismo para ejecutar una subrutina, por lo que podemos decir que es similar a esta.
  - se ejecuta de forma sincrónica, de manera similar a una subrutina.
  - se la identifica por un número (y no con una dirección como la subrutina). Ejemplo:

Interrupciones por software

- Utiliza una tabla para asociar el número de interrupción con la dirección de una subrutina:
  - A esta tabla se la denomina “vector de interrupciones”.
  - Es externa e independiente del programa.
  - Tiene 256 posiciones, por lo que a lo sumo se pueden tener 256 interrupciones.
  - Cada posición ocupa 4 bytes: 2 bytes altos siempre en 0, 2 bytes bajos contienen la dirección de la subrutina.
  - El vector se encuentra ubicado al principio de la memoria y ocupa 1024 bytes (256 posiciones * 4 bytes por posición). En principio no deberíamos ubicar ni datos, ni partes del programa de la posición 0 a la 1023 de la memoria.

Interrupciones por software

Vector de interrupciones

posición	1º byte	4 bytes para dirección de subrutina	Ult. byte
0	0	{dirección de subrutina asociada a INT 0}	3
1	4	{dirección de subrutina asociada a INT 1}	7
2	8	{dirección de subrutina asociada a INT 2}	11
	…	…	…
254	1016	{dirección de subrutina asociada a INT 254}	1019
255	1020	{dirección de subrutina asociada a INT 255}	1023

- el número de interrupción se establece al compilar el programa, queda fijo y no puede cambiarse sin volver a compilar.
- la dirección de la subrutina almacenada en una posición del vector se establece de manera independiente. Puede cambiarse en cualquier momento sin compilar el programa.

Interrupciones por software

- ¿Por qué es útil una interrupción por software (INT) si contamos con subrutinas (CALL)?
  - Al no dejar una dirección de la subrutina dentro de nuestro programa tenemos la flexibilidad de utilizar cualquier dirección.
  - Podemos usar “servicios” del sistema operativo (monitor del MSX88) que ya están implementados:
    - Imprimir una cadena
    - Leer un carácter
    - Terminar el programa
  - En distintas versiones del sistema operativo las direcciones de las subrutinas que proveen servicios pueden cambiar y nuestro programa no necesita saberlo.

Interrupciones por software

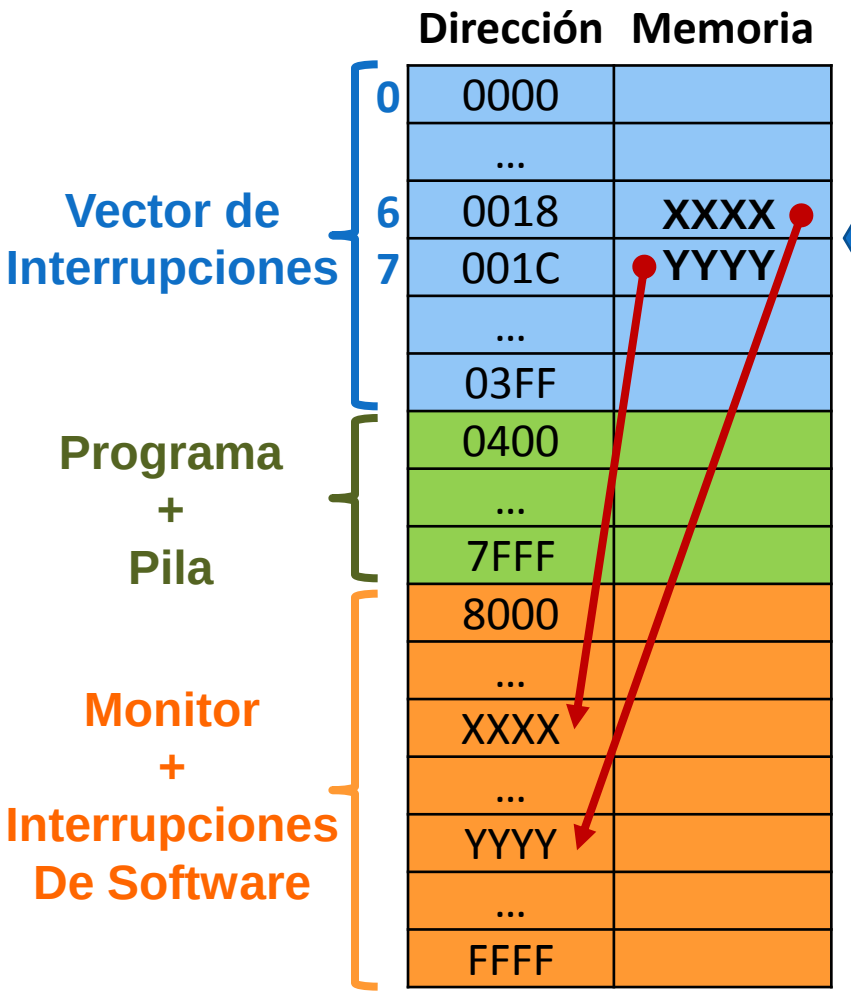
- Como funciona una interrupción(pseudo-código):
  - INT {nº de interrupción}:
    - Salva flags de estado en la pila.
    - Salva dirección de retorno en la pila.
    - Busca dirección de rutina en la posición {nº de interrupción} del vector.
    - Asigna registro IP con la dirección de la subrutina
    - (Ejecuta rutina)

Toda rutina de interrupción debe dejar los registros con los valores previos al llamado. Es decir que debe realizar PUSH de los registros que usa, al iniciar la rutina , y debe realizar POP de los mismos al finalizar

Interrupciones por software

- Para retornar de una subrutina normal (CALL) usamos la instrucción RET que desapila la dirección de retorno cuando esta finalizaba.
- Para volver de una subrutina de interrupción (INT) usamos la instrucción IRET, que además de la dirección de retorno, restaura desde la pila los flags de estado.
- IRET (pseudo-código)
  - (esta al final de la subrutina de INTERRUPCIÓN)
  - Repone los flags de estado desde la pila.
  - Asigna registro IP con la dirección de retorno desde la pila.

Interrupciones por software

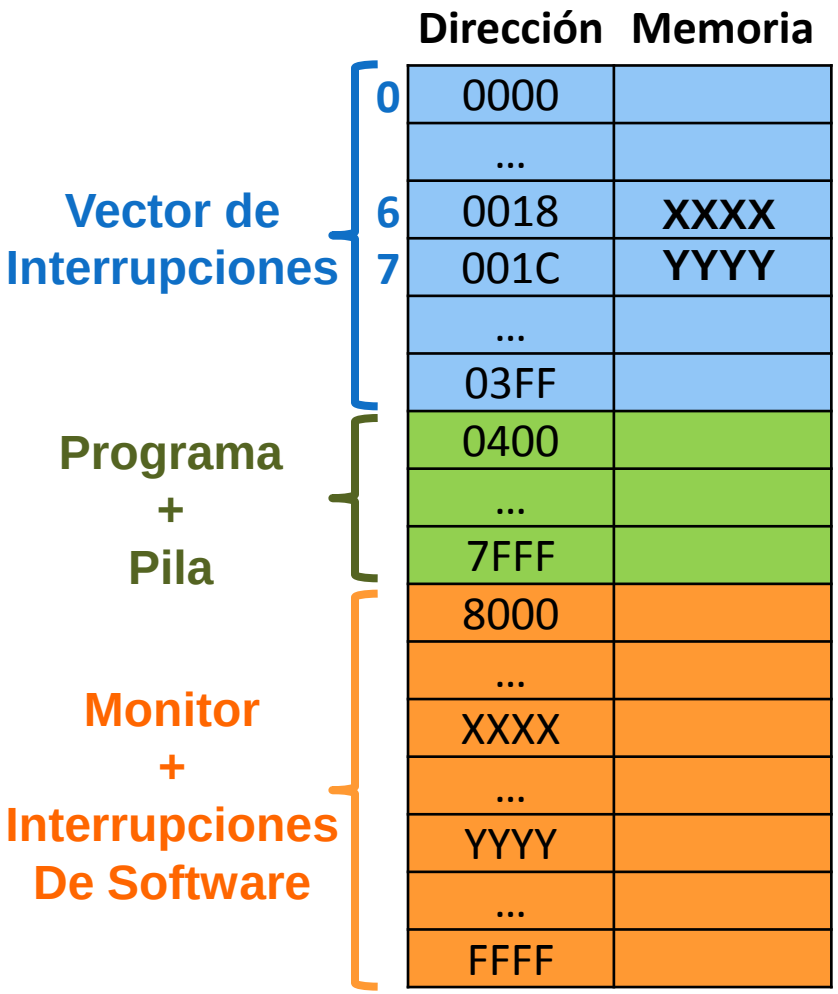


Cuando se carga el monitor en memoria se inicializan las ***posiciones*** *del vector* de interrupciones con las ***direcciones*** *de las subrutinas* correspondientes

Código en el espacio de memoria del monitor

```
ORG XXXXh
LeerCaracter:
...
IRET
...
ORG YYYYh
ImprimirCadena:
...
IRET
```

Interrupciones por software



Para invocar desde nuestro programa a la rutina *LeerCaracter* usamos la instrucción **INT 6** y para *ImprimirCadena* usamos **INT 7**

Código en el espacio de memoria del monitor

```
ORG XXXXh
LeerCaracter:
...
IRET
...
ORG YYYYh
ImprimirCadena:
...
IRET
```

Interrupciones por software

Interrupciones por software del MSX88:

- Finalizar programa (HALT):
  - INT 0
- Depurar programas (no lo utilizamos):
  - INT 3
- Leer 1 Carácter (sin eco):
  - BX = dirección (OFFSET) almacenamiento
  - INT 6
- Escribir String:
  - BX = dirección (OFFSET) almacenamiento
  - AL = cant. Car. a escribir
  - INT 7

Interrupciones por software

Ejercicio 1:

```
1.  ORG 1000H
2.  MSJ DB "ARQUITECTURA DE COMPUTADORAS-"
3.  DB "FACULTAD DE INFORMATICA-"
4.  DB 55H   U
5.  DB 4EH   N
6.  DB 4CH   L
7.  DB 50H   P
8.  FIN DB ?
9.  ORG 2000H
10. MOV BX, OFFSET MSJ
11. MOV AL, OFFSET FIN-OFFSET MSJ
12. INT 7
13. INT 0
14. END
```

¿Son números, letras o algo más?
¿De que depende?

Luego de la ejecución de esta instrucción en la consola del MSX88 aparece el texto: "ARQUITECTURA DE COMPUTADORAS-FACULTAD DE INFORMATICA-UNLP"

Termina la ejecución del programa, igual que HALT

Interrupciones por software

Ejercicio 4: Lectura de datos desde el teclado.

Escribir un programa que solicite el ingreso de un número (de un dígito) por teclado e inmediatamente lo muestre en la pantalla de comandos, haciendo uso de las interrupciones por software INT 6 e INT 7.

Interrupciones por software

Ejercicio 4:

```
1.  ORG 1000H
2.  MSJ DB "INGRESE UN NUMERO:"
3.  FIN DB ?
4.  ORG 1500H
5.  NUM DB ?
6.  ORG 2000H
7.  MOV BX, OFFSET MSJ
8.  MOV AL, OFFSET FIN-OFFSET MSJ
9.  INT 7
10. MOV BX, OFFSET NUM
11. INT 6
12. MOV AL, 1
13. INT 7
14. MOV CL, NUM
15. INT 0
16. END
```

Imprime en la consola: “INGRESE UN NUMERO:”

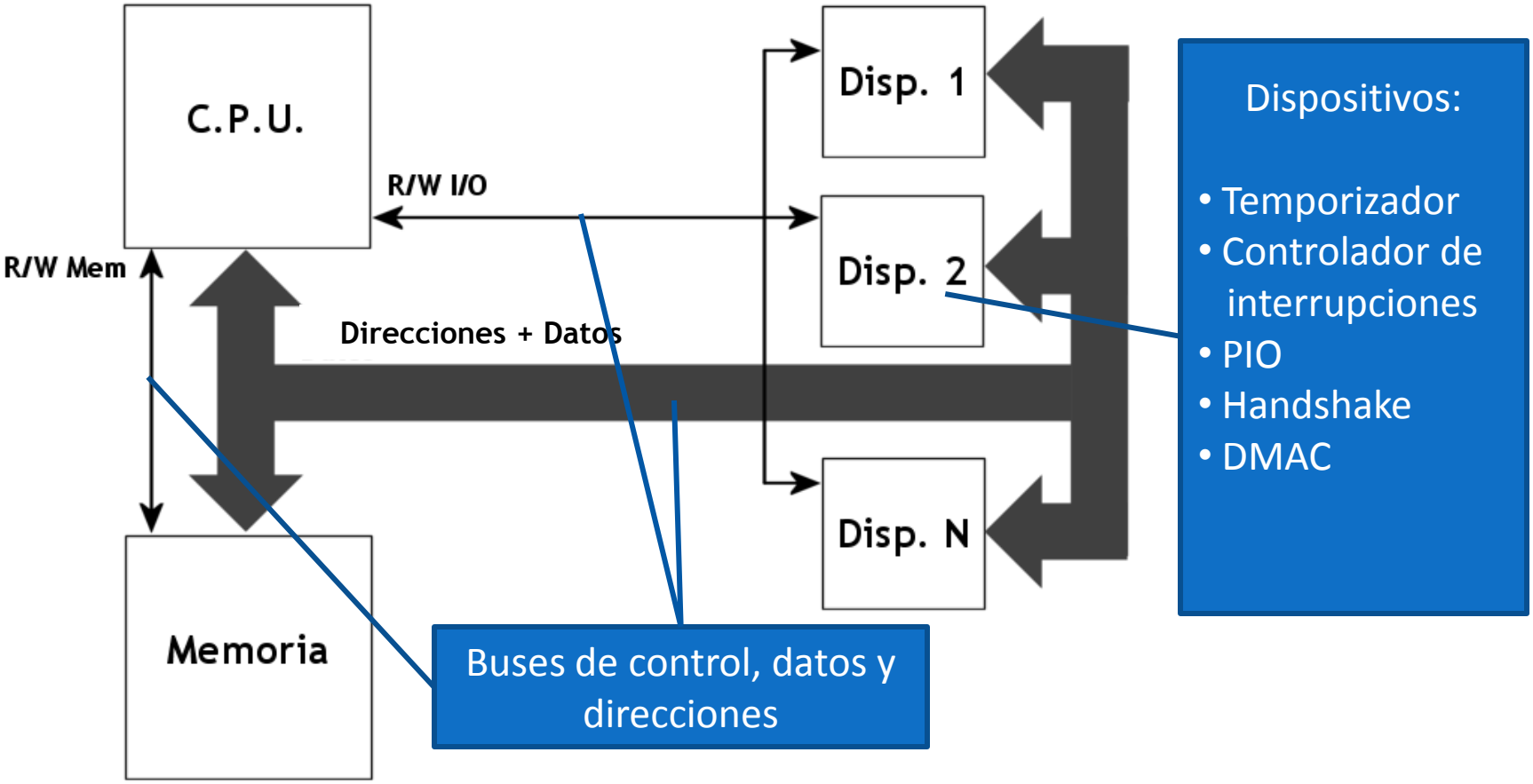
Espera el ingreso de un carácter desde la consola

Imprime en la consola el carácter recién ingresado

Comprobación visual de que se ingresó el carácter

Entrada / Salida MSX88

Esquema de arquitectura del MSX88:



Notar que las señales de control para acceder a memoria y a dispositivos son distintas

Entrada / Salida MSX88 - Dispositivos

- Denominamos dispositivo al hardware adicional a la CPU (similar a esta) con una función bien específica.
- Generalmente son mucho mas lentos que la CPU.
- Tienen registros propios (similar a la CPU) que utilizan para realizar las funciones para las que están diseñados.
- Los registros están “cableados” a los buses del sistema, y se leen/escriben de manera similar a la que se lee/escribe la memoria.

Entrada / Salida MSX88 - Dispositivos

- Escribir en estos registros nos permite:
  - Configurar el dispositivo
  - Enviar datos
- Leer estos registros nos permite:
  - Saber el estado en que se encuentra el dispositivo
  - Recibir datos
- Tipos de Dispositivos del MSX88:
  - PIC: Controlador de Interrupciones Programable
  - Temporizador o Timer: para medir tiempo
  - PIO o Handshake: comunicarnos con equipos externos (ej: impresora)
  - DMAC: Transferencia eficiente entre memoria y dispositivos

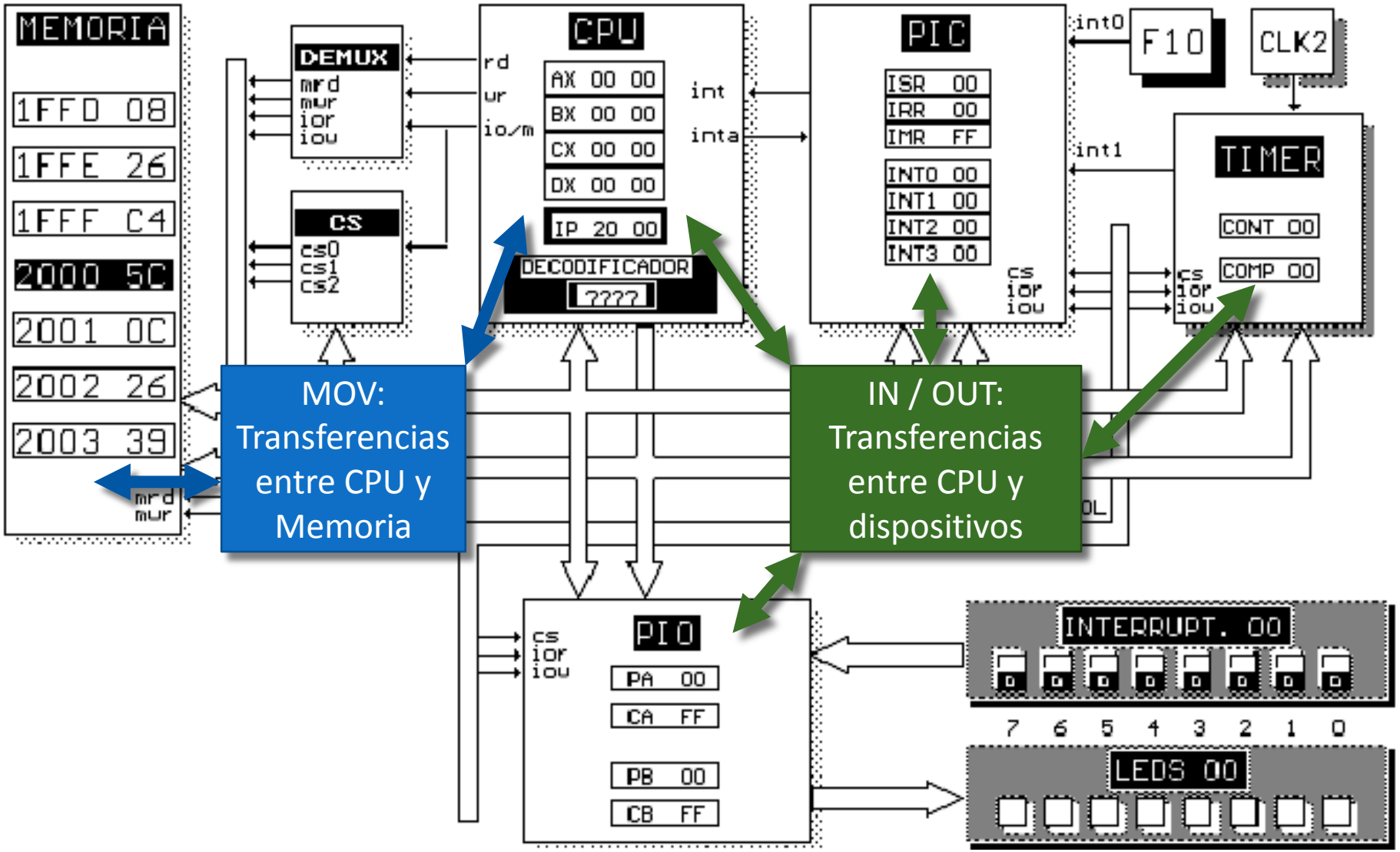
Entrada / Salida MSX88 - Instrucciones

- Como para acceder a memoria y a dispositivos se utilizan señales de control diferentes, también se utilizan instrucciones diferentes

- Para salida (CPU ➔ Dispositivo):
  - OUT DX, AL
  - OUT {número de puerto o dirección de E/S}, AL
- Para entrada (Dispositivo ➔ CPU):
  - IN AL, DX
  - IN AL, {número de puerto o dirección de E/S}

Solo puede utilizarse AL (8bits) y DX
o
AL y un valor inmediato con la dirección del puerto

Entrada / Salida MSX88 - Dispositivos



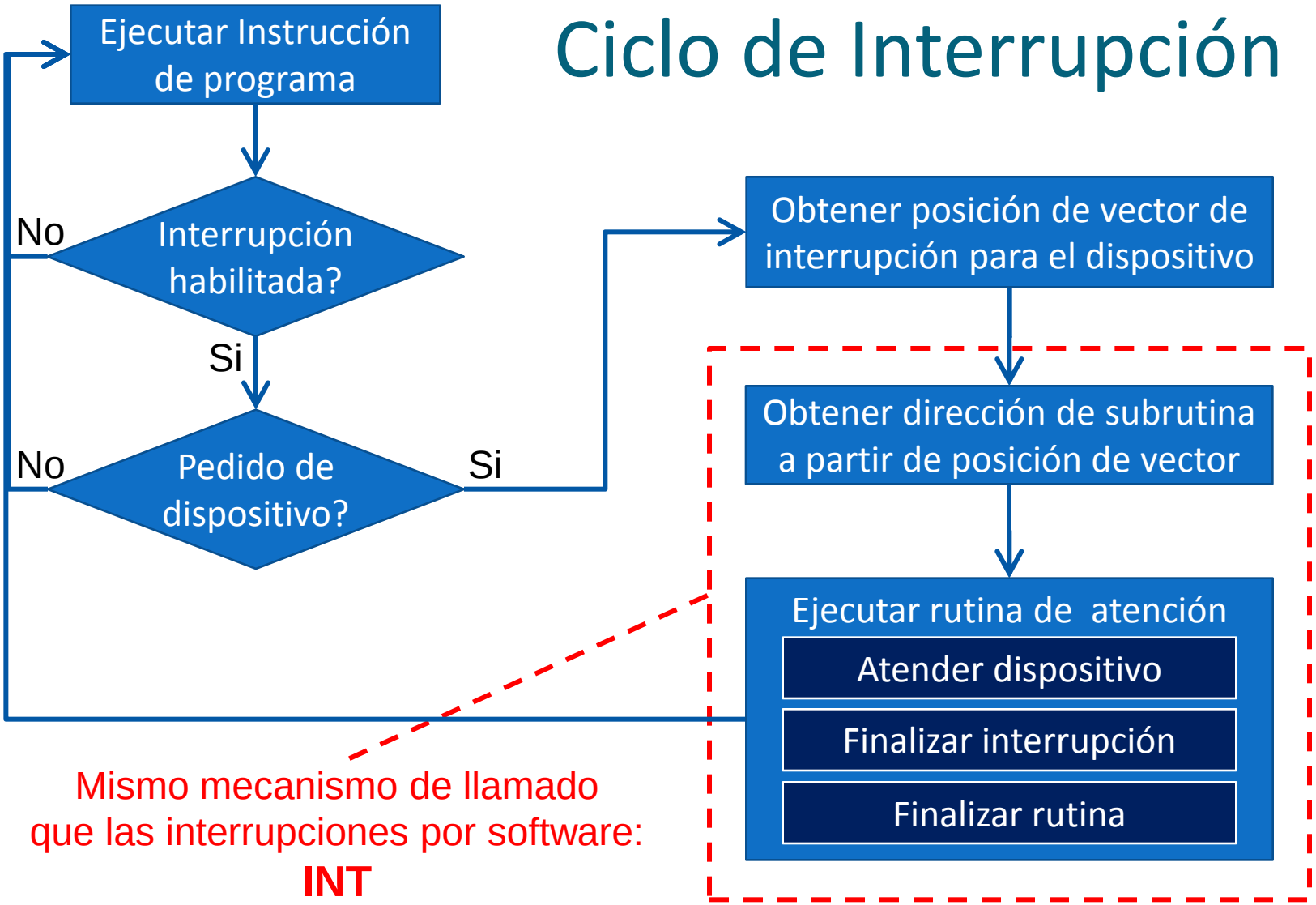
Interrupciones por hardware

- Es un pedido que realiza un dispositivo (hardware) a la CPU para resolver un evento asociado a este.
- Una interrupción por hardware suspende (interrumpe) el flujo normal de la ejecución de un programa para ejecutar código especial para resolver el requerimiento del dispositivo.
- Para que un dispositivo genere interrupciones, el programador debe inicializarlo adecuadamente.
- Cuando un programa finaliza se debe restablecer el dispositivo para que no continúen las interrupciones
- Ventaja: evita el “polling” de los dispositivos (polling es consulta iterativa hasta que un dispositivo este listo)

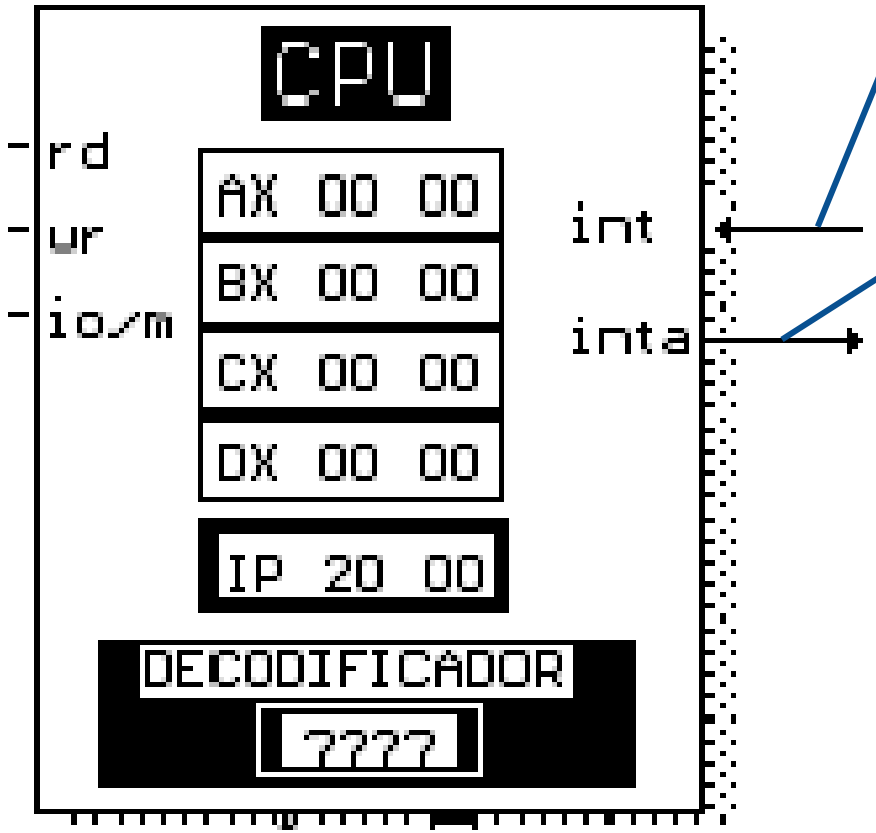
Interrupciones por hardware

- Ciclo de interrupción:
  - Completa ejecución de instrucción del programa en curso
  - La CPU verifica que hay interrupción de un dispositivo y no esta enmascarada (interrupciones habilitadas)
  - Obtiene la posición del vector de interrupción asociada al dispositivo
  - Obtiene la dirección de la subrutina de atención de interrupción de la posición del vector
  - Ejecuta subrutina:
    - Se atiende el requerimiento del dispositivo
    - Se indica que el dispositivo fue atendido
    - Finaliza subrutina
  - Comienza ejecución de una nueva instrucción del programa

Interrupciones por hardware



Interrupciones por hardware



Cuando un dispositivo necesita interrupción activa la señal “int”

Cuando la CPU esta lista para procesar la interrupción activa la señal “inta” para notificar al dispositivo que esta lista para la interrupción

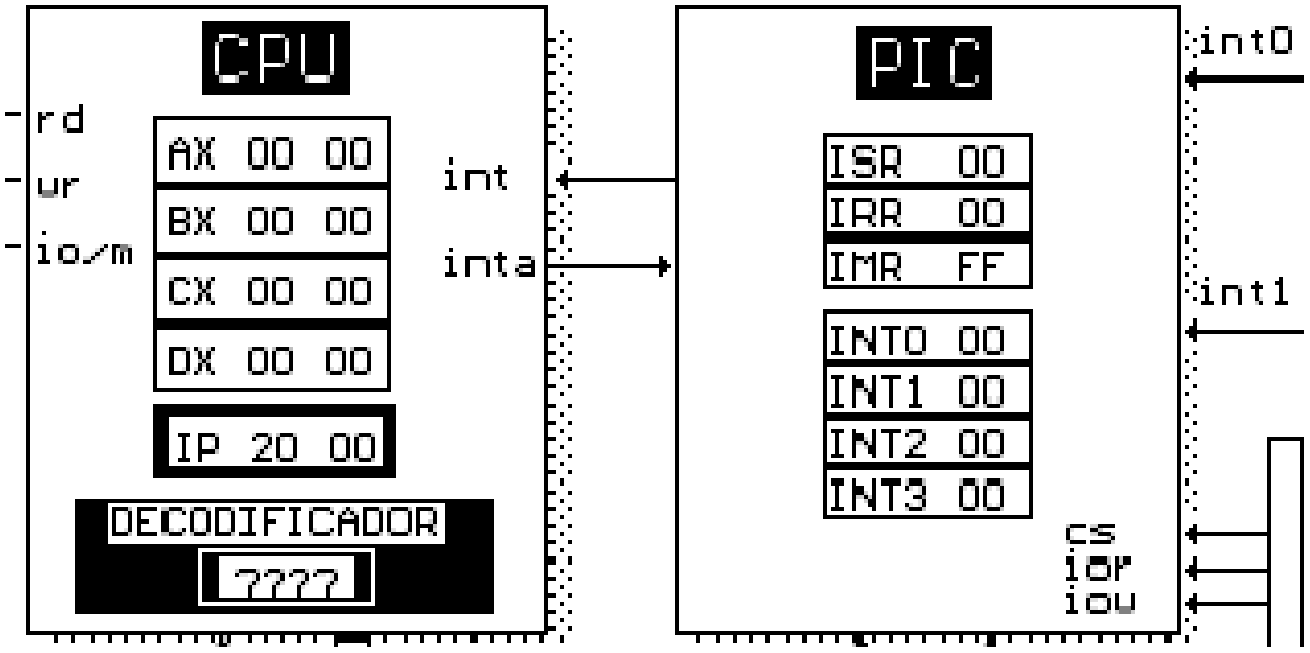
Una interrupción se ejecuta entre instrucciones. Se verifica la condición de interrupción luego de la ejecución de cada instrucción.

Si queremos que la CPU ignore los pedidos de interrupción podemos “enmascarar” el bit de interrupción del registro de flags.

CLI: deshabilita interrupciones (flag I = 0)
STI: habilita interrupciones (flag I = 1)

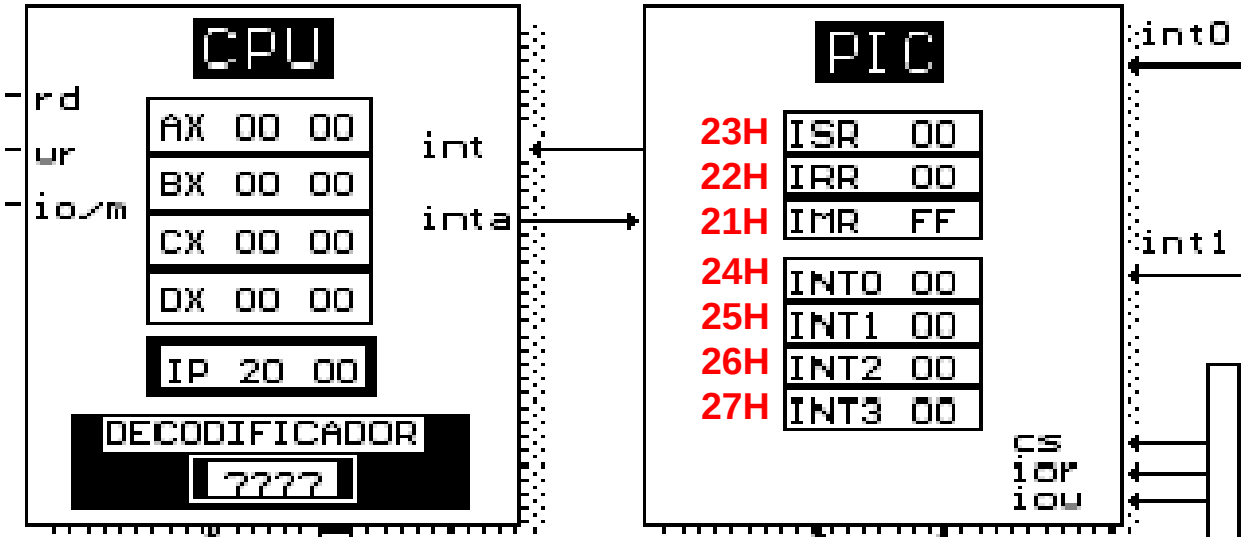
Interrupciones por hardware

- Con el esquema anterior solo podemos atender un dispositivo o fuente de interrupción.
- Necesitamos un dispositivo que nos permita administrar interrupciones para varios dispositivos.



Interrupciones por hardware - PIC

- El dispositivo PIC (Programmable Interrupt Controller) permite controlar hasta 8 fuentes (dispositivos) de interrupción.
- Esta “cableado” o conectado en la dirección de entrada/salida 20H (dirección del 1er registro).
- Tiene 12 registros que permiten configurar y consultar los dispositivos de interrupción.



PIC - Registros

- EOI (End of Interrupt, 20H):
  - solo se usa el bit 5 de este registro. Se debe poner a 1 para indicar que finalizo la atención del dispositivo que pidió interrupción

- IMR (Interrupt Mask Register, 21H):
  - cada bit se asocia a una fuente de interrupción diferente:
    - 0 ➔ habilita
    - 1 ➔ deshabilita
  - Si un dispositivo pide interrupción y el bit asociado esta en 1, no se propaga la interrupción a la CPU (no confundir con el flag I, este es a nivel PIC)

PIC - Registros

- IRR (Interrupt Request Register, 22H):
  - cada bit se asocia a una fuente de interrupción diferente (bit 0 a dispositivo 0, bit 1 a dispositivo 1, etc.)
  - Indica que un dispositivo necesita atención (puede haber varios pedidos simultaneos)

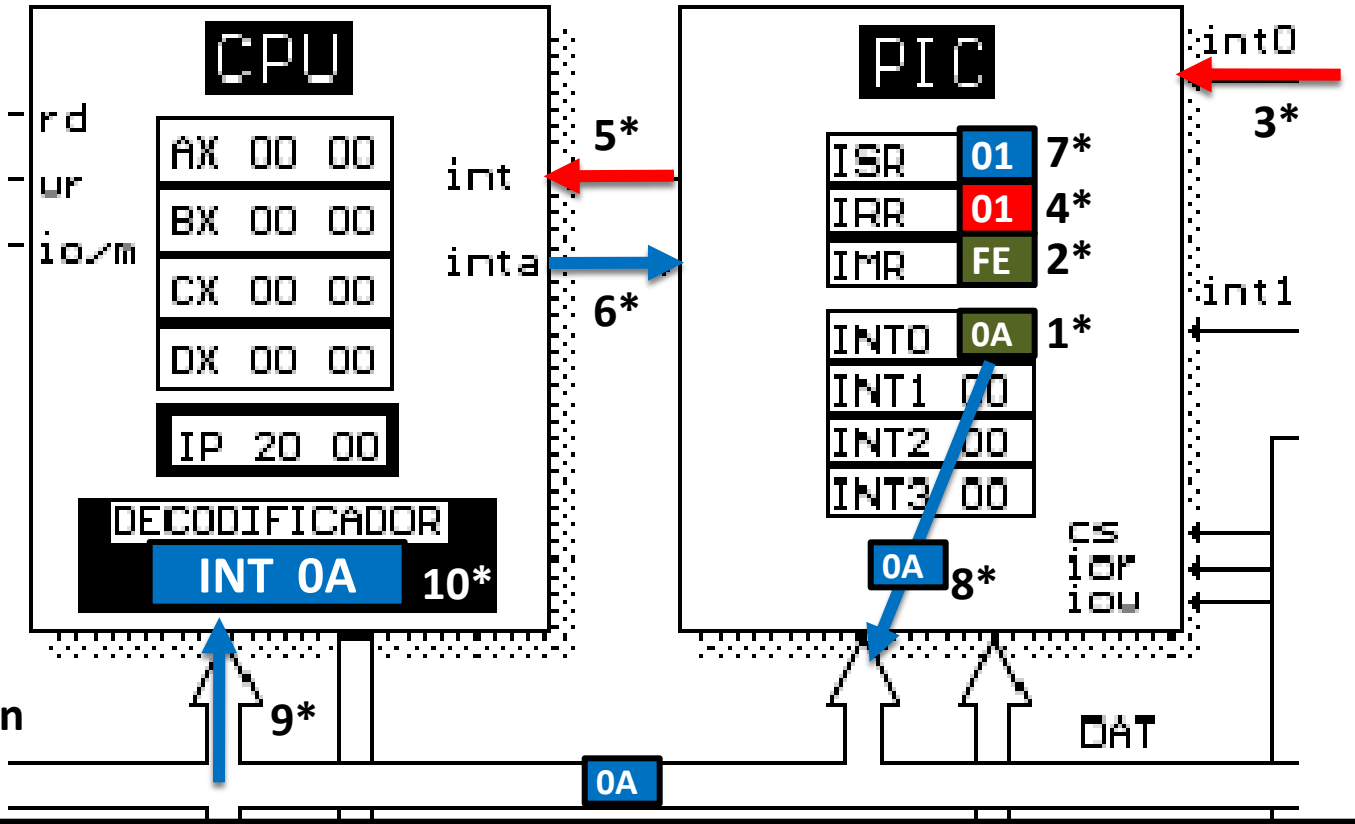
- ISR (Interrupt Service Register, 23H):
  - cada bit se asocia a una fuente de interrupción diferente
  - Indica que la CPU esta atendiendo un pedido de interrupción (solo un dispositivo a la vez):
    - Se pone el bit en 1 cuando la CPU acepta el pedido
    - Se pone el bit en 0 cuando la CPU indica al PIC que termino la interrupción (hay que escribir un 1 en el bit 5 del registro EOI)

PIC - Registros

- INT0 , 24H:
  - este registro contiene la posición del vector interrupción donde se encuentra la subrutina de atención para el dispositivo asociado a la entrada 0 del PIC.
  - Se utiliza el mismo mecanismo que la interrupción por software con la diferencia que el número de interrupción se saca de este registro
- INT1, 25H: idem pero para dispositivo 1
- INT2, 26H: idem pero para dispositivo 2

  ... ...
- INT6, 2AH: idem pero para dispositivo 6
- INT7, 2BH: idem pero para dispositivo 7

Ciclo de interrupción de dispositivo 0



* Orden de ejecución

En la posición 0A (10) del vector de interrupción se copia la dirección de la subrutina para atender al dispositivo 0

Luego se carga la posición del vector en INT0 (1*) y habilitar el dispositivo 0 en IMR (2*)

Instrucción INT 0A recupera la posición 10 del vector y llama a la subrutina de atención

MOV AL, 20H y OUT 20H,AL al final de la subrutina termina la atención

Finalmente los registros ISR e IRR vuelven al estado anterior de la interrupción

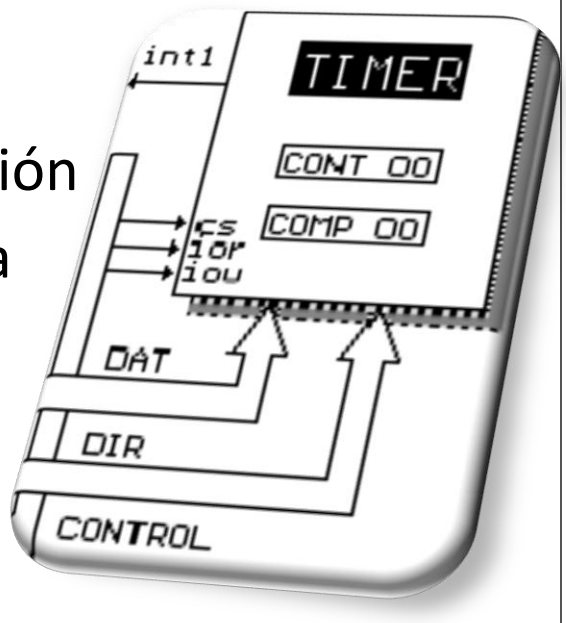
Dispositivos del MSX88

- Tecla F10:

  - Genera una interrupción al presionar la tecla F10
  - Simula un evento de un dispositivo a través de esta tecla
  - No requiere programación del dispositivo, solo la programación del PIC
  - Asociada a la entrada int0 del PIC

Dispositivos del MSX88

- TIMER: temporizador
  - Interrumpe a la CPU transcurridos una cantidad de segundos desde su programación
  - Conectado al puerto 10H de entrada/salida
  - Asociado a la entrada int1 del PIC
  - Tiene 2 registros:
    - CONT (contador ascendente): 10H
    - COMP (comparador): 11H



  - Cuando el valor de CONT coincide con el valor de COMP genera interrupción.
  - En cada interrupción es necesario restablecer el valor del registro CONT al valor original de programación para mantener la frecuencia de conteo

Acerca de INT

- Hay que poner atención al contexto en el que utilizamos la palabra INT:
  - “int” es la señal que entra a la CPU para indicar que hay una interrupción
  - “INT” es la instrucción que ejecuta una interrupción.

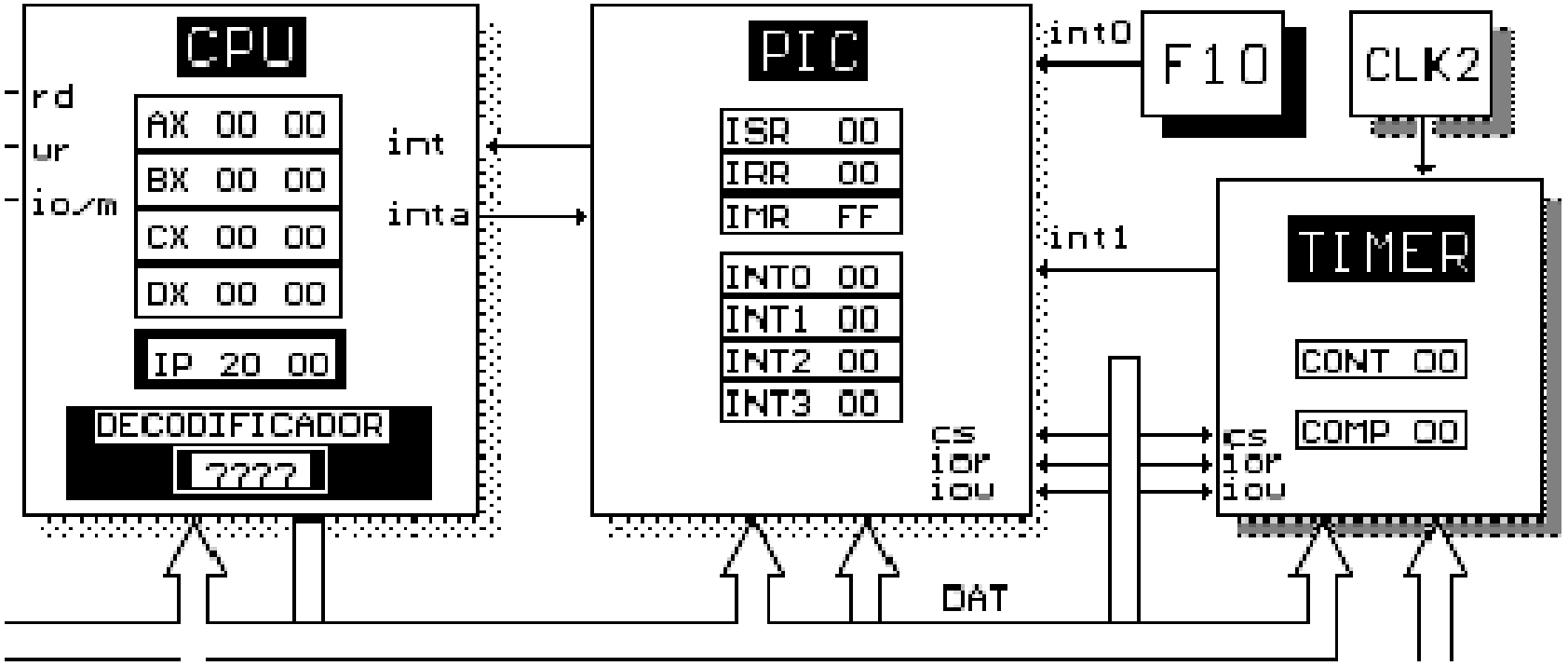
    Ej: INT 7
  - “Int0” es la señal de entrada asociada al PIC para el dispositivo 0 (Tecla F10)
  - “INT0” es el nombre que le damos al registro del PIC asociado al dispositivo 0 (dirección 24H)
  - “INT 0” es la interrupción por software que termina un programa
  - “INT0” podría ser una constante de un programa.

    Ej: INT0 EQU 24H

Ejercicio 12

Interrupción por hardware: TIMER.

Implementar a través de un programa un reloj segundero que muestre en pantalla los segundos transcurridos (00-59 seg) desde el inicio de la ejecución.



Pasos para programar TIMER:

1. Deshabilitar interrupciones de CPU
2. Programar PIC:
    1. Guardar en INT1 del PIC la posición del vector que tiene la subrutina
    2. Habilitar Interrupción solo para TIMER (bit 1 del registro IMR del PIC)
3. Programar TIMER:
    1. Inicializar CONT en 0
    2. Inicializar COMP en 1
4. Habilitar interrupciones de la CPU

Pregunta: ¿tendríamos el mismo resultado si programamos CONT con 5 y COMP con 6?

```
1.  TIMER EQU 10H
2.  PIC EQU 20H
3.  EOI EQU 20H
4.  N_CLK EQU 10
5.  ORG 40
6.  IP_CLK DW RUT_CLK

7.  ORG 1000H
8.  SEG DB 30H     ; ASCII del "0"
9.  DB 30H
10. FIN DB ?

11. ORG 3000H
12. RUT_CLK: PUSH AX ;Rutina interrup.
13.       INC SEG+1
14.       CMP SEG+1, 3AH
15.       JNZ RESET
16.       MOV SEG+1, 30H ; "0"
17.       INC SEG
18.       CMP SEG, 36H    ; "6"
19.       JNZ RESET
20.       MOV SEG, 30H    ; "0"
21. RESET: INT 7   ;Imprime tiempo
22.       MOV AL, 0
23.       OUT TIMER, AL;Reinicia CONT
24.       MOV AL, EOI
25.       OUT PIC, AL;PIC:Fin interrup
26.       POP AX
27.       IRET ;CPU Fin interrup.
```

```
28. ORG 2000H
29.    CLI
30.    MOV AL, 0FDH  ;Máscara para Disp
31.    OUT PIC+1, AL ;PIC: registro IMR
32.    MOV AL, N_CLK
33.    OUT PIC+5, AL ;PIC: reg. INT1
34.    MOV AL, 1
35.    OUT TIMER+1, AL ;TIMER: reg. COMP
36.    MOV AL, 0
37.    OUT TIMER, AL  ; TIMER: reg. CONT
38.    MOV BX, OFFSET SEG
39.    MOV AL, OFFSET FIN-OFFSET SEG
40.    STI
41. LAZO: JMP LAZO
42. END
```