Dispositivos y Entrada/Salida

viernes, 23 de octubre de 2020 17:36

Entrada y salida de datos sin que esto sea a la memoria o a la CPU

Dispositivos -> *Drivers*

Dispositivos externos

Del simulador

- Llaves (entrada)
- Impresora (salida)
    - Buffer
- Luces (salida)
- Interrupción f10 (entrada)

Dispositivos externos y CPU

- No hay una comunicación directa entre la CPU y los dispositivos externos, la misma se lleva a cabo con:

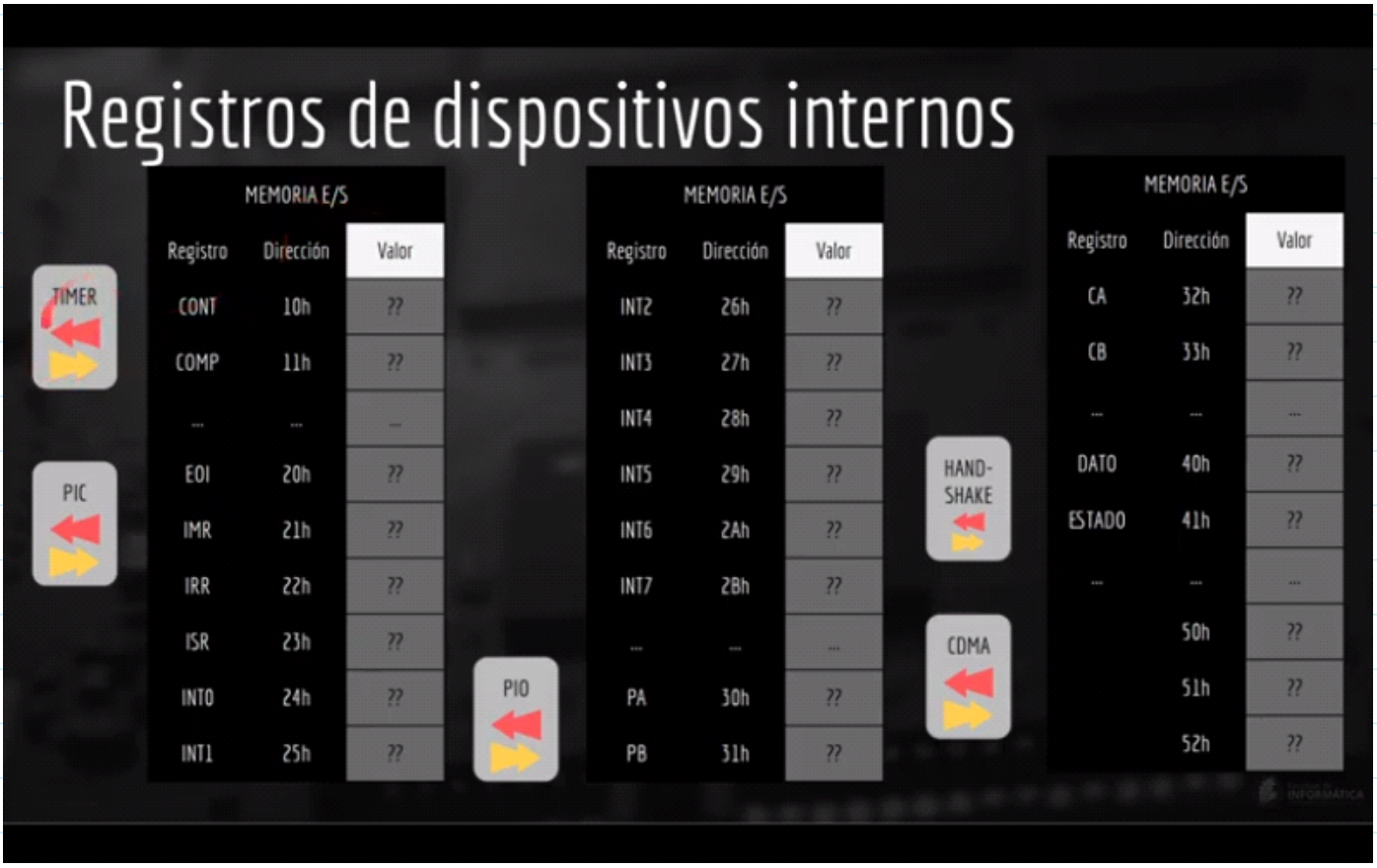
Dispositivos internos, externos y CPU

- Los **dispositivos internos** son el comunicador entre los dispositivos externos y el procesador
- Permite que el procesador sea más genérico y evoluciones de manera independiente con respecto a los dispositivos externos.

Dispositivos internos

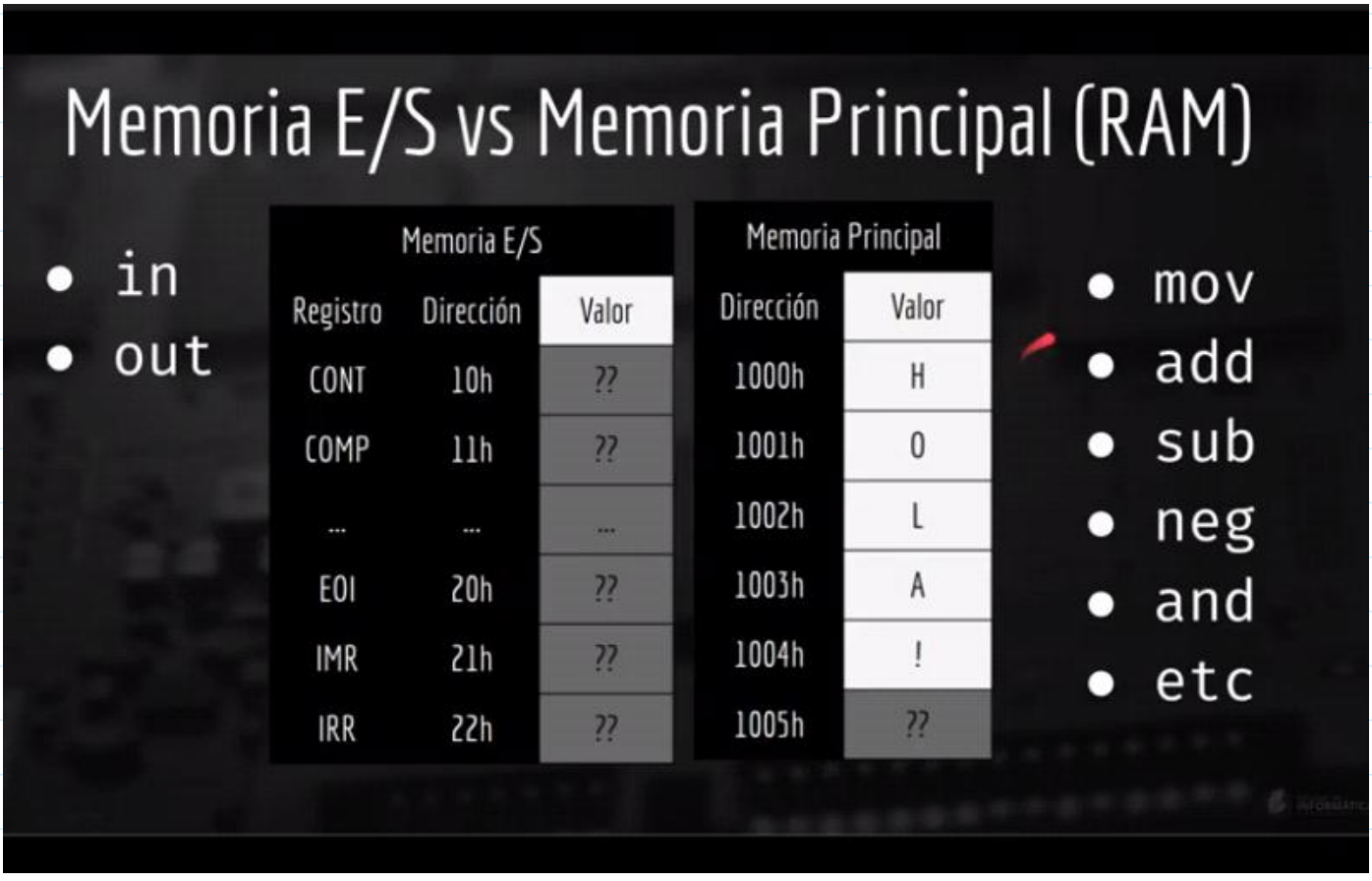
- **PIC -** Interrupciones por hardware
- **TIMER -** Llevar una cuenta del tiempo
- **PIO -** Sirve para interactuar con las luces, las llaves y la impresora (dispositivo configurable)
- **HANDSHAKE -** No tan configurable, solo sirve para la impresora
- **CDMA -** Pasar datos de un lugar al otro sin meter al procesador en el medio

Registro de dispositivos internos - ¿Cómo podemos interactuar con estos dispositivos?



Mediante una memoria de E/S (que no es la principal) dónde cada dispositivo tiene sus registros

MEMORIA E/S vs MEMORIA PRINCIPAL (RAM)



Memoria E/S es un espacio de memoria paralelo, distinto. Si usamos las instrucciones propias de la memoria RAM el procesador puede confundirse y no saber de qué parte de la memoria estamos

hablando. Las celdas de esta memoria ya tienen una función predefinida

- IN
- OUT

RAM que tiene memoria instrucciones

- Mov
- Add
- Sub
- Neg
- And
- etc

INSTRUCCIONES *IN* Y *OUT*

- Similares al MOV
- Siempre trabajan con AL
- Pero importa el sentido ¡!
  - Al , **in**
  - Out , **al**
- **In:** copiar valor de registro E/S a AL
- **Out**: copiar valor de AL al registro E/S

PIO - Dispositivo interno y EQUS

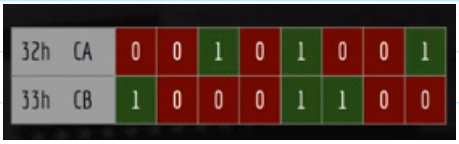
viernes, 23 de octubre de 2020 19:09

- *Programmable* Input Output device
- Dispositivo de Entrada Salida *Programable*
- Dos puertos:
  - A y B
  - **8 bits** cada uno
- Cada **bit** es *programable*
  - Entrada o salida (pero no ambas)

Cómo se programa el PIO

- Cada bit de A o B puede ser de entrada o de salida

  - **Registros** de datos , SÍ SON DATOS!
    - PA y PB
  - **Registros** de configuración, me dice que bit es de entrada y qué bit es de salida (no son datos)
    - CA y CB
    - 0 salida
    - 1 entrada



- Puedo cambiar, en un programa, la configuración el PIO de manera dinámica. Es un lujo, casi.

Direcciones * ver apunte de Priscila que lo tiene completo las direcciones, menos las de leds.

- PA -> 30h
- PB -> 31h
- CA -> 32h
- CB -> 33h

EQU

Para hacer más legibles los códigos ...

- EQU = Constante
- Legibilidad del código
- Evita errores
- No es solo para el PIO

Control de Luces con el PIO

sábado, 24 de octubre de 2020 16:50

Conexionado entre el PIO y las luces

- Configuración 0 del simulador
- Cada luz recibe un valor
  - 0 -> apagada
  - 1 -> prendida
- Cada luz se conecta a un bit de PB
  - En orden
  - Bits de salida
    - Para esto tengo que configurar CB con 0000 0000 (todos salida)
- Cambiar bits de PB prende y apaga luces
- **Pio con Luces conectadas a PB**

¿ Qué pasa si no configuro a CB con 0000 0000?

Tendré bits de entrada y no se correspondería el encendido de las luces con los datos ingresados

Ejemplo 1

- Prender las 4 luces y apagar el resto
  - Configuro CB
    - 0000 0000 (todas las luces son salida)
  - Paso 1111 0000 a PB

```
PB equ 31h
CB equ 33h
; 1 entrada
; 0 salida
Org 2000h
; configurarción de CB
Mov al , 00h ; 0000 0000
Out cb , al
; prender las luces
Mov al , 0F0h ; 1111 0000
Out pb , al
```

Ejemplo 2

- Prender todas las luces
- Luego apagarlas

```
PB equ 31h
CB equ 33h
Org 2000h
; confg. De CB
Mov al , 00h ; 0000 0000
Out cb , al

;prendo todas las luces
Mov al , 0FFh
Out pb , al

;apago todas las luces
Mov al , 0h
Out PB , al
```

Ejemplo 3

- Prender y apagar todas las luces
  - 100 veces

```
PB EQU 31 H
CB EQU 33H
Org 2000h
; configuración de cb
Mov al , 00h
Out cb , al
Mov cl , 100
Loop: mov al , 0FFh
      Out pb , al
      Mov al , 0h
      Out pb , al
```

```
Mov al , 0h
Out pb , al
Dec cl
Jnz loop
```

Recorte de pantalla realizado: 24/10/2020 17:15

Ejemplo 4

- Leer un dígito del teclado
    - Del 0 al 7
- Prender la luz del bit correspondiente

```
pb equ 31h
cb equ 33h

org 1000h
 digito db ?
 cero db "0"

org 2000h
 mov al , 00h
 out cb , al

 ; leer el digito
 mov bx , offset digito
 int 6
 mov cl , [bx]

 ;restar ascii del 0 ("3" -> 3)
 sub cl , cero

 ;rotar izq cl veces
 mov al , 1

 loop: cmp cl , 0
 jz mandar
 add al , al
 dec cl
 jmp loop

 ; prendo la luz
mandar: out pb , al

 hlt
 end
```

Me ayudo con dos variables: el carácter que leo y un cero

Lectura de llaves con el PIO

sábado, 24 de octubre de 2020 17:41

Conexionado entre el PIO y las llaves

- Configuración 0 del simulador
- Cada llave manda un valor
  - 0 -> apagada
  - 1 -> prendida
- Cada llave se conecta a un bit de PA
  - En orden
  - Bits de entrada
- Cambiar llaves cambia los bits en PA
- **Hay que configurar el PIO con llaves conectadas a PA**
  - CA: Todos 1

¿Qué pasa si no configura CA con 1111 1111?

Esto no quiere decir que no pueda cambiar las llaves con el PA, porque las llaves no pueden tener esa funcionalidad. Las llaves mandan datos y la CPU los reciben, así que por más que yo ponga un 0 en el CA no significa que yo al subir la llave esta un 1 o 0 en PA.

Ejemplo 1

Leo el valor de las llaves y lo guardo en al

- Configurar CA
  - 1111 1111
- Leo PA

```
pa equ 30h
ca equ 32h
; 1 entrada
; 0 salida
org 2000h
; configuración de CA
mov al , 0ffh ; 1111 1111
out ca , al

; leer
in al , pa

hlt
end
```

Ejemplo 2

Tengo que adivinar una clave

- 1 solo intento
- Leer el valor de las llaves
- Coincide con la clave
  - Mostrar "correcto"

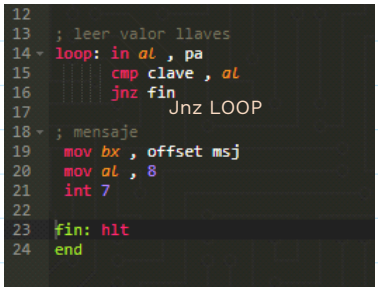
```
pa equ 30h
ca equ 32h

org 1000h
clave db 06fh ; 0110 1111
msj db "Correcto"

org 2000h
;configuración de CA
mov al , 0ffh ; 1111 1111
out ca , al
; leer valor llaves
in al , pa
cmp clave , al
jnz fin

; mensaje
mov bx , offset msj
mov al , 8
int 7

fin: hlt
end
```

Ejemplo 3

Adivinar la clave pero con infinitos intentos

- Leer el valor de las llaves hasta que coincidan

```
pa equ 30h
ca equ 32h

org 1000h
clave db 06fh ; 0110 1111
msj db "Correcto"

org 2000h
;configuración de CA
mov al , 0ffh ; 1111 1111
out ca , al

; leer valor llaves
loop: in al , pa
      cmp clave , al
      jnz fin
```



Se usa una **consulta de estado** ...

- La consulta de estado es **ineficiente**, porque si la CPU es rápida y nosotros lentos (que lo somos) y el tiempo que transcurre entre el cambio de llaves es mucho, es tiempo en el cuál el procesador está en el loop y pierde recursos.
- Es también llamada **poll.**

Impresora Centronics

sábado, 24 de octubre de 2020 19:00

Impresora y conexiones

Tiene 3 conexiones y , en total , 10 bits / 10 cablecitos.

- **Busy (ocupada):** Es una señal de la impresora a la computadora
  - Siempre hay que verificar que la impresora **no** esté ocupada.
  - Está en 0 si puedo evitar datos
  - Está en 1 si, efectivamente, **la impresora está ocupada** y no le puedo enviar información. La impresora **ignora** cualquier dato que le enviemos.
- **Strobe (señal):** Es una señal de la computadora a la impresora, de cuándo le estamos mandando un dato.
  - Para indiciar envío de datos. Es flanco ascendente
  - Se pone en 0, luego en 1
- **Datos:** 8 líneas de datos, ya que es un puerto paralelo. O sea, 8 bits que van en forma paralela. de la computadora a la impresora
  - 8 bits para codificar un carácter

Busy y Buffer de Impresión

- La CPU es más rápida que la impresora
- Para esto existe el **buffer**
  - El buffer es la memoria de la impresora
  - Guarda caracteres hasta que se impriman
  - Es una memoria limitada
  - Si se llena ...
    - Busy = 1
    - No se reciben más datos
      - Hasta que haya más lugar.

Impresora desocupada + STROBE en 1 // 0 // flanco ascendente

- Busy = 0
  - La impresora puede recibir datos
- Datos = 61h (código ASCII de la "a")
- Strobe en 1
  - No imprime , porque necesito un flanco ascendente
- Strobe en 0
  - No imprime
- Strobe en 0 y luego 1
  - Pone "a" en el buffer

- Luego imprime

¿Por qué tenemos el flanco ascendente?

Esto sirve para cuándo, por ejemplo, tengo 2 letras iguales ("Alla") e indica que el carácter ese tiene que imprimirse

Y me permite cambiar el dato.

Conexión de la impresora con el PIO o el Handshake

- Con el **PIO** es un poco más engorrosa, pero tiene la ventaja de que es un dispositivo genérico y programable
- El **Handshake** tiene más facilidad, pero no es genérico, está hecho específicamente para comunicarse con la impresora.

Impresora con Handshake y polling

sábado, 24 de octubre de 2020 21:00

Handshake

- Dispositivo interno de e/s
- Diseñado específicamente para impresora Centronics
- No sirve para comunicarse con otros dispositivos
- Manda señal de Strobe automáticamente

Conexión ente el Handshake y la impresora

- Conexionado 2 del simulador
- Registros Dato / Estado, no requiere configuración

- Se conecta mediante los registros
  - Los 8 de la impresora en el registro DATO de la dirección 40h
    - Conectados en orden
    - Conexión paralela. Viajan al mismo tiempo por distintos canales
  - El registro de ESTADO en el 41 h. Con estos dos bits controlamos la sincronización con la impresora para saber cuándo le enviamos datos
    - El bit más significado (0) es el de *busy*
    - El bit 1 es el de *strobe*
- A diferencia del **PIO** , el Handshake no requiere ninguna configuración.

¿Cómo imprimo un carácter?

- Escribir en el registro Estado
- Byte con el código ASCII del carácter

```
1   dato equ 40h
2
3   org 1000h
4   char db "a"
5
6   org 2000h
7   ;cargo el carácter
8    mov al , char
9   ; lo cargo a dato
10   out dato , al
11
12  int 0
13  end
```

- Si la impresora es muy lenta, puede ser que termine el programa antes de que yo pueda ver el carácter impreso

Pero si yo quiero usar consulta de estado ... (manera correcta)

Para fijarnos que la impresora no esté ocupada a la hora de imprimir el carácter

- Leo *busy* hasta que sea 0
  - Registro de **estado**
  - Ver el 0 (busy)
- Envío el carácter
  - Registro de **datos**
  - Escribo el byte

Consulta de estado

¿Cómo verifico un solo bit?

- Utilizando una máscara
  - 0000 0001 = 1 = 01h
- La función **AND** (para ver lo mismo que estaba antes)
  - XXXX XX**SB** **AND** 0000 0001
- Comparo el resultado
  - Jz para saltar si busy = 0
  - Jnz para saltar si busy = 1

```
; constantes
dato equ 40h
estado equ 41h

;variables
org 1000h
char db "a"

; pp
org 2000h

;este loop trae el valor del estado al registro AL
loop: in al , estado
      ; hace un and y aislamos en AL el bit de busy
      and al , 1
      ; si está en 1, vuelve al loop
      jnz loop

;salio del loop, así que mando el carácter
 mov al , char
 out dato , al

 int 0
end
```

¿Cómo imprimo una cadena de caracteres?

- La forma **INCORRECTA** es **SIN CONSULTAR EL ESTADO !!!!!111**

```
; constantes
dato equ 40h
estado equ 41h

;variables
org 1000h
cadena db "Milanesas"

; pp
org 2000h

;cargo la cadena
 mov cl , 9
 mov bx , offset cadena

;este loop trae el valor del estado al registro AL
; es la misma consulta de estado que antes
loop: in al , estado
      ; hace un and y aislamos en AL el bit de busy
      and al , 1
      ; si está en 1, vuelve al loop
      jnz loop ; mini - loop

      ;voy cargando la cadena
      mov al , [bx]
      out dato , al
      inc bx
      dec cl
      jnz loop
 int 0
end
```

- Se termina de imprimir cuándo los caracteres se terminan
- Se utiliza loop porque ambas etiquetas sirven para eso

Impresora con PIO y polling - 1 char

sábado, 24 de octubre de 2020 21:31

PIO * TIT

- Dispositivo interno de E/S
- Programable / configurable
    - Sirve para comunicarse con varios dispositivos
- No conoce las impresoras Centronics, así qué:
    - Señal de Strobe " a mano"

Conexión entre el PIO y la impresra *subt

- Conexionado 1 del simulador
- Utiliza los registros PA y PB, por lo cual requiere configuración
    - **Registro PA** = 30h
        - Registro de *Strobe y Busy*
    - **Registro CA** = 32h
        - Los primeros 6 bits no nos importan que valor tengan
        - Nos importan que los últimos 2 bits tengan el valor adecuado para funcionar
            - El bit número uno (correspondiente al *strobe*) tendrá el valor 0 (salida)
            - El bit número cero (correspondiente al *busy*) tendrá el valor 1 (entrada)
            - XXXX XX**01**
    - **Registro PB** = 31h
        - Registro de DATOS
    - **Registro CB** = 33h
        - Lo ponemos todo como salida
            - 0000 0000

¿Cómo imprimo un carácter? * subit

- Configuro PA y PB mediante CA y CB
- Escribo en PB el carácter
- **Mando señal de Strobe**
    - Recordar que es 0 , luego 1 . Por el flanco ascendente

```
PSEUDO CODIGO
ca = XXXX XX01
cb = 0000 0000

pb = "a"
Strobe (PA) = 0
Strobe (PA) = 1
```

```
; constantes
pa equ 30h
pb equ 31h
ca equ 32h
cb equ 33h

;variables
org 1000h
char db "a"


;pp
org 2000h
; CONFIGURACIONES
; configuro CA (strobe y busy)
mov al , 01h
out ca , al
; configuro CB (datos)
mov al , 00h ; todos de salida
out cb , al

; envio el carácter
mov al , char
out pb , al
; pongo el strobe en 0
in al , pa
and al , 0fdh ; 1111 1101  máscara para dejar todos iguales a los bits
; menos el bit en strobe que lo ponemos en 0
out pa , al

; pongo el strobe en 1
in al , pa
or al , 02h ; 000 0010
; alrevés que lo de arriba, lo pongo en 0
out pa , al

int 0
end

```

Subrutina para configurar CA y CB

Utilizamos subrutinas para la configuración de CA y CB ya que el programa se hace muy complejo sino.

```
; constantes
ca equ 32h
cb equ 33h

org 3000h
 ; configuro CA
conf_pio: mov al , 01h
          out ca , al
          ; configuro CB
          mov al , 00h
          out cb , al
          ret
```

Subrutina para mandar señal de *strobe*

```
; constantes
pa equ 30h
pb equ 31h

org 3100h
strobe0: in al , pa
         and al , 0fdh ; 1111 1101
         out pa , al
         ret

org 3200h
strobe1: in al , pa
         or al , 02h ; 0000 0010
```

Programa principal me quedaría algo como esto... *énfasis*

```
org 2000h

call conf_pio

mov al , char
out pb , al
```

```
org 3200h
strobe1: in al , pa
         or al , 02h ; 0000 0010
         out pa , al
         ret
```

```
mov al , char
out pb , al

call strobe0
call stobe1
```

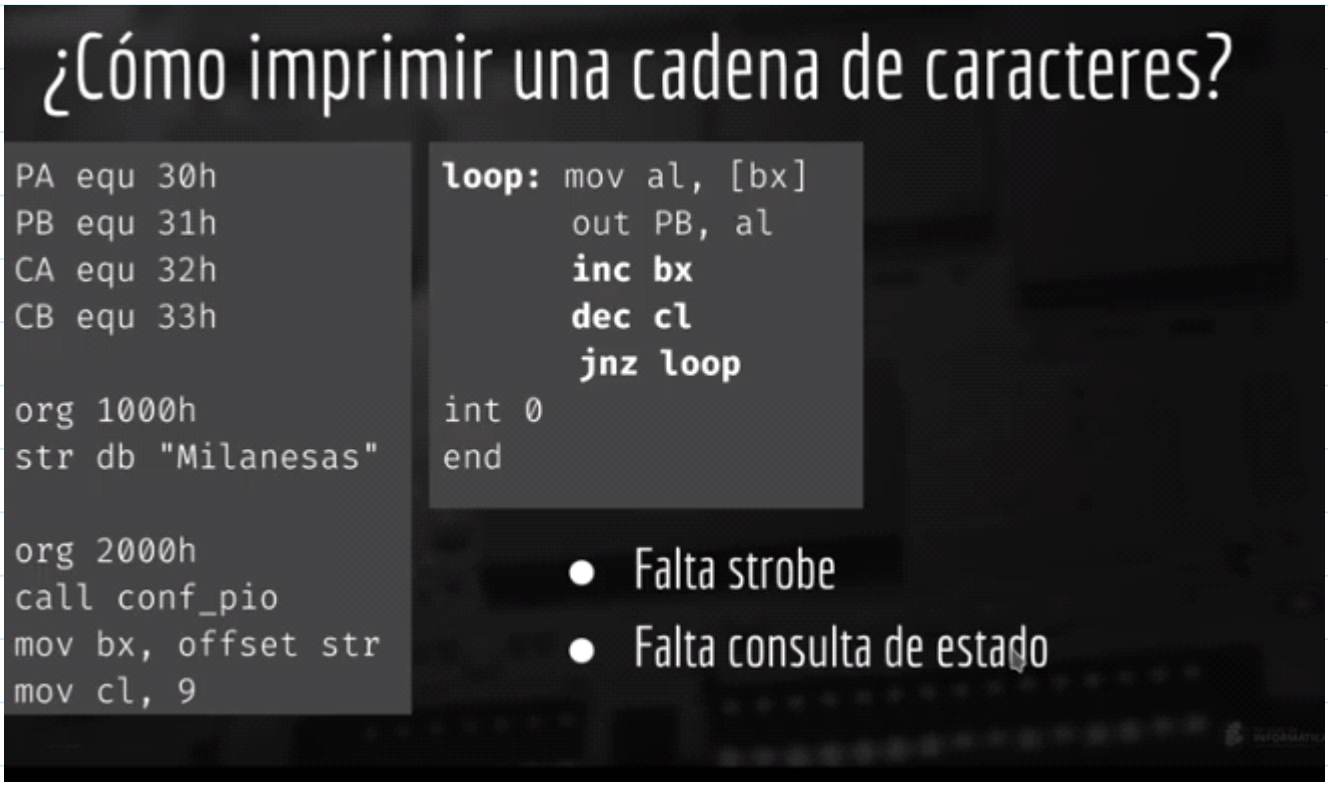
Es más cómodo hacer dos subrutinas para el strobe

Carácter + consulte de estado ...

```
; constantes
pa equ 30h
pb equ 31h
ca equ 32h
cb equ 33h
; variables
org 1000h
char db "a"

; subrutinas
org 3300h
poll: in al , pa
      and al , 1
      jnz poll
      ret

; pp
org 2000h
call conf_pio

;consulta de estado BUSY = 0
call poll

; envio el carácter
 mov al , char
 out pb , al

 call strobe0
 call strobe1

int 0
end
```

Impresoras con PIO y polling - cadena

sábado, 24 de octubre de 2020 23:55



Cómo imprimir una cadena de caracteres + consulta de estado + strobe

```
; constantes
pa equ 30h
pb equ 31h
ca equ 32h
cb equ 33h

; variables
org 1000h
cadena db "Milanesas"


---------
; subrutinas
org 3300h
poll: in al , pa
      and al , 1
      jnz poll
      ret

; pp
org 2000h
call conf_pio

mov bx , offset cadena
mov cl , 9
;consulta de estado BUSY = 0
call strobe0 ; inicizlización de strobe en 0
loop: call poll
      mov al , [bx]
      out pb , al
      call strobe1 ; acá invierto y pongo primero en 1
      ; genero flano ascendente y vuelvo el strobe a 0
      call strobe0 ; luego en  0
      ; así al ejecutar de nuevo el lazo, estoy volviendo a generar
      ; el flanco
      inc bx
      dec cl
      jnz loop

int 0
end
```

Impresora con Handshake e interrupciones

domingo, 25 de octubre de 2020 1:30

¿ Qué era el Handshake?

- Era un dispositivo interno (de entrada / salida) que está como intermediario con la CPU.
- Diseñado específicamente para impresoras Centronics
- No sirve para comunicarse con otros dispositivos
- Manda señal de *strobe* automáticamente.

Hasta ahora

Vimos dos maneras de imprimir

- Con el Handhskae haciendo polling
- Con el PIO haciendo polling

Pero ahora vamos a ver una última, la definitiva (?)

- **Con el Handshake e interrupciones**

Esto significa algo terrorífico ...

Conexión entre el Handshake , la impresora y el PIC

- Bit de interrupción , el primerito primerito. Está **registro estado** , es el bit más significativo
  - 1 . Interrupciones activadas

- Sólo se agrega la dirección de la línea 2

  ¿Cómo configuro el bit de interrupción?

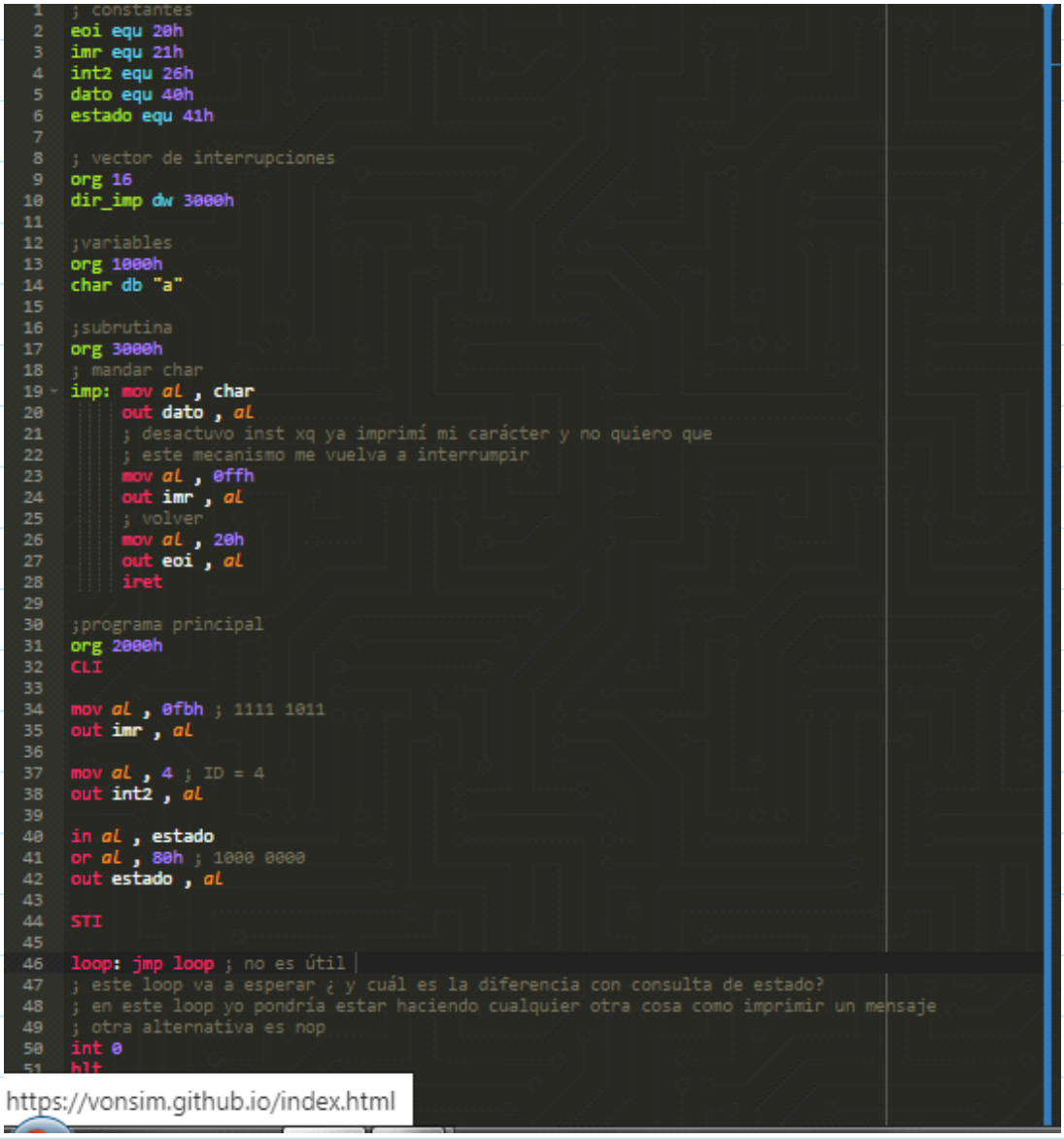
  - Mediante un **or** entre:
    - **X**xxx xx**SB**
    - **1**xxx 0000
      - **1**xxx xx**SB**

  PIC

  - Imr
  - Int2
  - Vector de interrupciones

Ejemplo 1

- Desde una *subrutina de atención* imprimo un carácter.
- Cambia completamente la forma de trabajar
  - Ahora no se imprime desde el programa principal, sino desde la subrutina
  - Busy = 0 -> handshake -> handshake le avisa al pic -> pedido de interrupción



```
; constantes
eoi equ 20h
imr equ 21h
int2 equ 26h
dato equ 40h
estado equ 41h

; vector de interrupciones
org 16
dir_imp dw 3000h

;variables
org 1000h
char db "a"

;subrutina
org 3000h
; mandar char
imp: mov al , char
     out dato , al
     ; desactuvo inst xq ya imprimí mi carácter y no quiero que
     ; este mecanismo me vuelva a interrumpir
     mov al , 0ffh
     out imr , al
     ; volver
     mov al , 20h
     out eoi , al
     iret

;programa principal
org 2000h
CLI

mov al , 0fbh ; 1111 1011
out imr , al

mov al , 4 ; ID = 4
out int2 , al

in al , estado
or al , 80h ; 1000 0000
out estado , al

STI

loop: jmp loop ; no es útil
; este loop va a esperar ¿ y cuál es la diferencia con consulta de estado?
; en este loop yo pondría estar haciendo cualquier otra cosa como imprimir un mensaje
; otra alternativa es nop
int 0
hlt
```

https://vonsim.github.io/index.html

Hlt

End

Y ahora una cadena!!1

```
; constantes
eoi equ 20h
imr equ 21h
int2 equ 26h
dato equ 40h
estado equ 41h
; vector de interrupciones
org 16
dir_imp dw 3000h
;variables
org 1000h
cadena db "Milanesas"
;subrutina
org 3000h
; mandar char
imp: mov al , [bx]
     out dato , al
     inc bx
     dec cl
     jnz volver
     ; desactuvo inst xq ya imprimí mi carácter y no quiero que
     ; este mecanismo me vuelva a interrumpir
     mov al , 0ffh
     out imr , al
volver: mov al , 20h
        out eoi , al
        iret
;programa principal
org 2000h
CLI
;confg del pic
mov al , 0fbh ; 1111 1011
out imr , al
mov al , 4 ; ID = 4
out int2 , al
;;;;;;
in al , estado
or al , 80h ; 1000 0000
out estado , al
;;;;;;
mov bx , offset cadena
mov cl , 9 ; contador de letras

STI
loop: cmp cl , 0
      jnz loop
int 0
hlt
end
```

Ejecución más eficiente

Interrupción VS Consulta de estado (polling)

domingo, 25 de octubre de 2020 15:43

¿Cómo sé cuál de las dos me conviene?

- Tengo que ver qué hace mi programa
- Velocidad de la CPU y la velocidad de los dispositivos.

Ventaja de las interrupciones

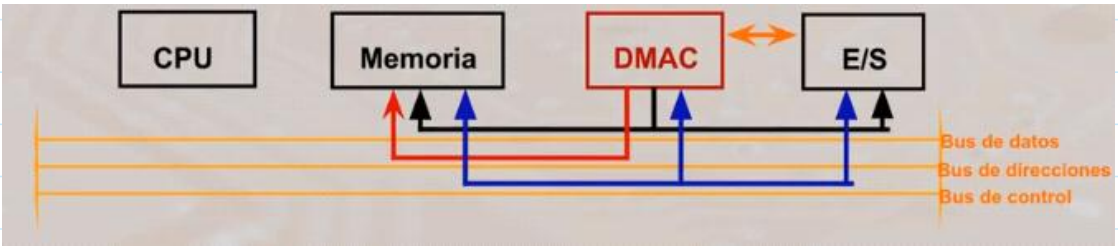
- Al mismo tiempo nuestro programa puede hacer dos cosas.
- Tengo que preservar los registros
- No pasa mucho tiempo con la impresora libre
- Si la impresora es rápida (misma velocidad que el CPU) no guarda en el buffer, por lo cual tener interrupciones no está ayudando ,por lo cual el programa no va a poder hacer nada útil, porque la impresora va a estar interrumpiendo todo el tiempo.
    - Es fuente de ineficiencia, porque al interrumpir suceden varias cosas (se guarda la dirección de retorno, los flags , se restaura, se manda a eoi)

Acceso Directo a Memoria (DMA)

domingo, 25 de octubre de 2020 16:48

Transferencia de datos utilizando el acceso directo a la memoria.

El **controlador de DMA** es un dispositivo capaz de controlar una transferencia de datos entre un periférico y memoria sin intervención de la CPU, quitándole trabajo a la CPU. El CDMA **solo sirve para copiar datos.**



Etapas de una transferencia DMA

1. Inicialización de la transferencia.
    a. Configurar el CDMA. Actúa el procesador
2. Realización de la transferencia
    a. No hace falta hacer nada. La CPU no actua, solo actúa el DMA
3. Finalización de la transferencia
    a. Atender la interrupción.

Controlador de DMA

El **Controlador de DMA (*DMAC*)** debe actuar como maestro del bus durante la transferencia DMA y debe ser capaz de:

- **Solicitar** el uso del bus mediante las señales y la lógica de arbitraje necesarias
- **Especificar la dirección** de memoria sobre la que se realiza la transferencia
- **Generar las señales** de control del bus
- **Tipo** de operación (lectura / escritura)
- Señales de **sincronización** de la transferencia.

Modos de transferencia

Si el DMAC sólo toma el control del bus durante los intervalos de tiempo en los que la CPU no hace uso del mismo, el **rendimiento del sistema no sufrirá degradación alguna.** Se distinguen dos tipos de transferencias:

- Por **ráfagas** (*bloques*)
- Por **robo de ciclo** (*demanda*)

DMA modo ráfaga (por bloque)

Cuándo la CPU concede el bus, el DMAC no lo libera hasta haber finalizado la transferencia de todo el bloque de datos completo

- **Ventaja:** La transferencia se realiza de forma rápida.
- **Desventaja:** Durante el tiempo que dura la transferencia la CPU no puede utilizar el bus con memoria, lo que puede degradar el rendimiento del sistema (a menos que haga cosas que no requiere el bloque de memoria)

Problemas con DMA

Se puede degradar el rendimiento de la CPU si el DMAC hace uso intensivo del bus

Si el bus está ocupado en una transferencia DMA, la CPU ***no puede acceder a memoria*** para leer instrucciones y/o datos

¿Solución? :

DMA modo robo de ciclo (por demanda)

Cuándo la CPU concede el bus al DMAC, se realiza la transferencia de un byte y después el DMAC libera el bus.

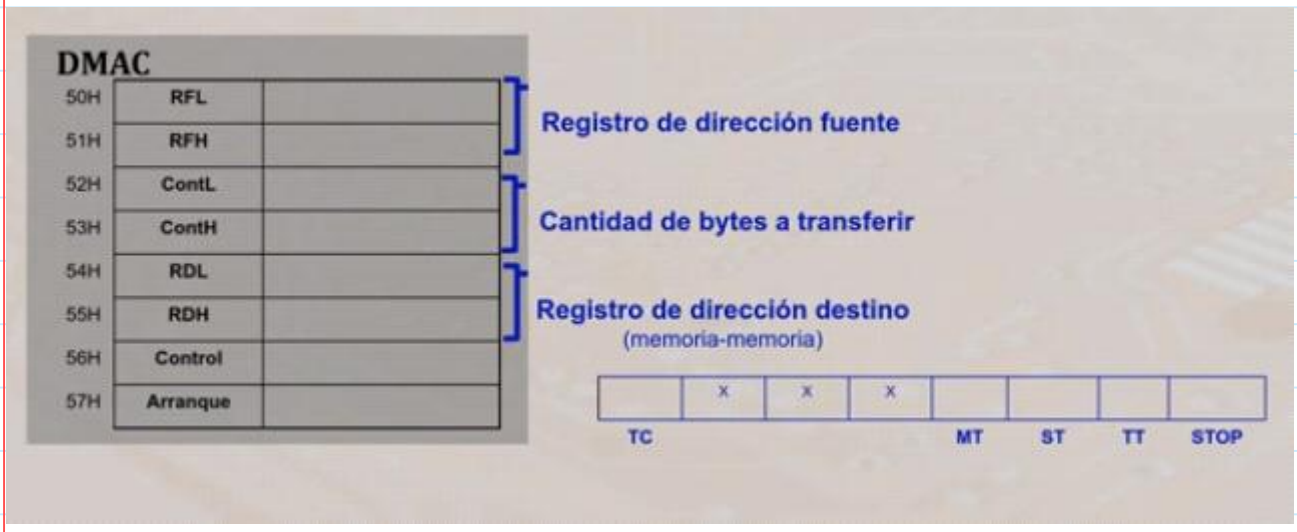
El DMAC solicita el control del bus tantas veces como sea necesario hasta finalizar la transferencia del bloque completo

Requiere más sincronización entre el DMA y la CPU

- **Ventajas:** No se degrada el rendimiento del sistema
- **Desventajas:** La transferencia tarda más tiempo en llevarse a cabo.

Configuración y uso del DMA

domingo, 25 de octubre de 2020 17:40



El DMA es un dispositivo interno que tiene varios registros para poder configurarlo

- **REGISTRO DE DIRECCIÓN FUENTE**, tiene dos partes ya que guarda una dirección
  - Parte baja
  - Parte alta
- **REGISTRO DE DIRECCIÓN DE DESTINO**, es dónde se van a copiar los datos. Al igual que el registro de dirección fuente posee dos partes. Es memoria-memoria
  - Parte baja
  - Parte alta
- **CANTIDAD DE BYTES A TRANSFERIR**
  - Parte baja
  - Parte alta
- **REGISTRO DE CONTROL** (56h)

  Nos permite empezar, terminar y configurar la transferencia.

  - **Stop:** En el bit más significativo permite que el usuario, apretando alguna tecla (programada previamente) pueda detener la transferencia.
    - 0 -> transferencia habilitada
    - 1 -> detener la transferencia

- **TT:** Es el tipo de transferencia, nos dice para dónde vamos a estar copiando.
    - 0 -> periférico / memoria o v.v (y hace uso de ST , ya que no sé de dónde hacía dónde, solo sé el tipo)
    - 1 -> memoria / memoria
- **ST:** Sentido de la trasferencia, qué cuándo TT = 0
    - 0 -> periférico -> memoria
    - 1 -> memoria -> periférico
- **MT:** Modo de transferencia, me va a decir cuál de las dos transferencias voy a usar.
    - 0 -> por todo de ciclo
    - 1 -> por ráfaga
- **X:**
- **X:**
- **X:**
- **TC:** Terminal Count. Es de solo lectura y me va a decir en qué estado se encuentra el pasaje de datos. No le vamos a dar uso <3 ya que vemos cómo va la transferencia por interrupciones, pero podríamos usarlo para polling, pero en ese caso ¿para qué usamos el DMAC? En ese caso usamos el procesador para la transferencia de datos.
    - 0 -> transferencia en curso
    - 1 -> transferencia finalizada

- **REGISTRO DE ARRANQUE (contrario al stop)**

  Le dice que empiece la transferencia, al revés que el de STOP, que me dice que pare.

  Si yo le pongo XXXX X**111 inicio la transferencia**

DMAC Y PIC

Cuándo termina la transferencia el DMAC nos tiene que avisar, así que para eso

conectamos el DMAC a la línea 3 del PIC, así que configuramos la línea INT3 (27h) para este dispositivo.

DMA. Transferencia memoria - memoria

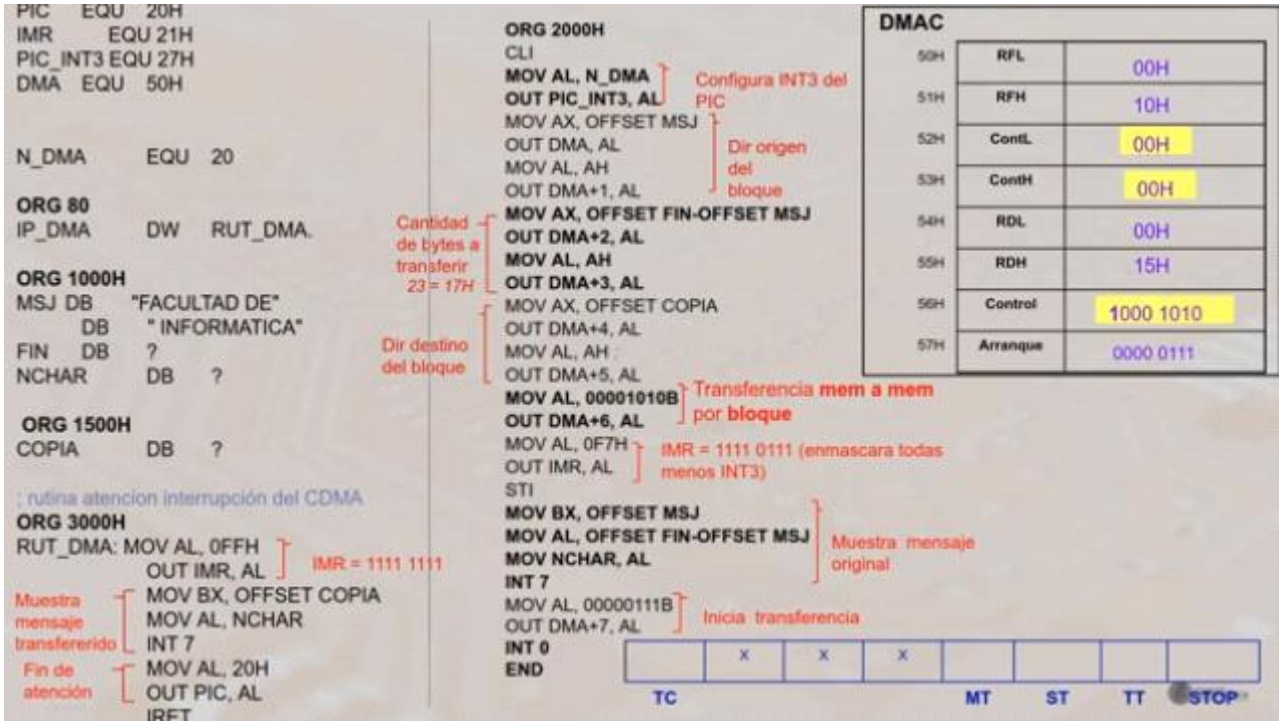
domingo, 25 de octubre de 2020 18:18

DMA. Transferencia de datos **memoria - memoria**

Escribir un programa que copie una cadena de caracteres almacenada a partir de la dirección 1000h. Los caracteres se copian a la dirección 1500h, 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 **PI C3**



pic equ 20h

imr equ 21h

int3 equ 27h

dma equ 50h

n_dma equ 20

```
org 80
ip_dma dw rut_dma

org 1000h
msj db "h o l a"
    db "m a r c o s"
fin db ?
nchar db ?

org 1500h
copia db ?

;rutina de atención de interrución del CDMA
org 3000h
rut_dma: mov al , 0ffh
        out imr , al
        mov bx , offset copia
        mov al , nchar
        int 7
        mov al , 20h
        out pic , al
        iret
org 2000h
cli

; configuramos int 3 del pic
mov al , n_dma
out int3 , al
```

```
; dirección de origen del bloque (fuente) SIEMPRE LO HACEMOS para copiar algo de
2b
mov ax , offset msj ; dirección del mensaje
out dma, al ; primero pasamos la parte baja al RFL
mov al , ah ; pasamos lo de ah a al, porque siempre usamos al
out dma+1 , al ; después pasamos la parte alta a RFH (dma+1)

; lo mismo , pero para la cantidad de bytes a transferir
mov ax , offset fin - offset msj  ; vemos cuántos carácteres
out dma+2 , al ; lo pasamos a ContL , dma+2 es contL
mov al , ah ; lo mismo que arriba
out dma+3 , al ; lo pasamos a ContH , dma+3 es contH

; la dirección destino arranca en 1500, que es dónde vamos a copiar
; para eso usamos copia que arranca eb 1500h
mov ax , offset copia ; pasamos la dirección de copia
out dma+4 , al ; pasa la parte baja a RDL (destino bajo)
mov al , ah  ; pasa a al para hacer el out
out dma+5 , al ; pasa la parte alta a RDH (destino alto)

; c o n f i g u r a m o s  a l  D M A
; transferencia memoria a momoria y por bloque,
; mt = 1    st = 0    tt = 1  stop = 0 para no deshabilitar la transferencia
; tc es de lectura, asi´que me chupa un huevo
mov al , 00001010b
out dma+6 , al ; le mandamos la configuración al control (dma+6)

; ahora enmascaro todas las interrupciones, menos la INT3
mov al , 0f7h ; 1111 0111
out imr , al
```

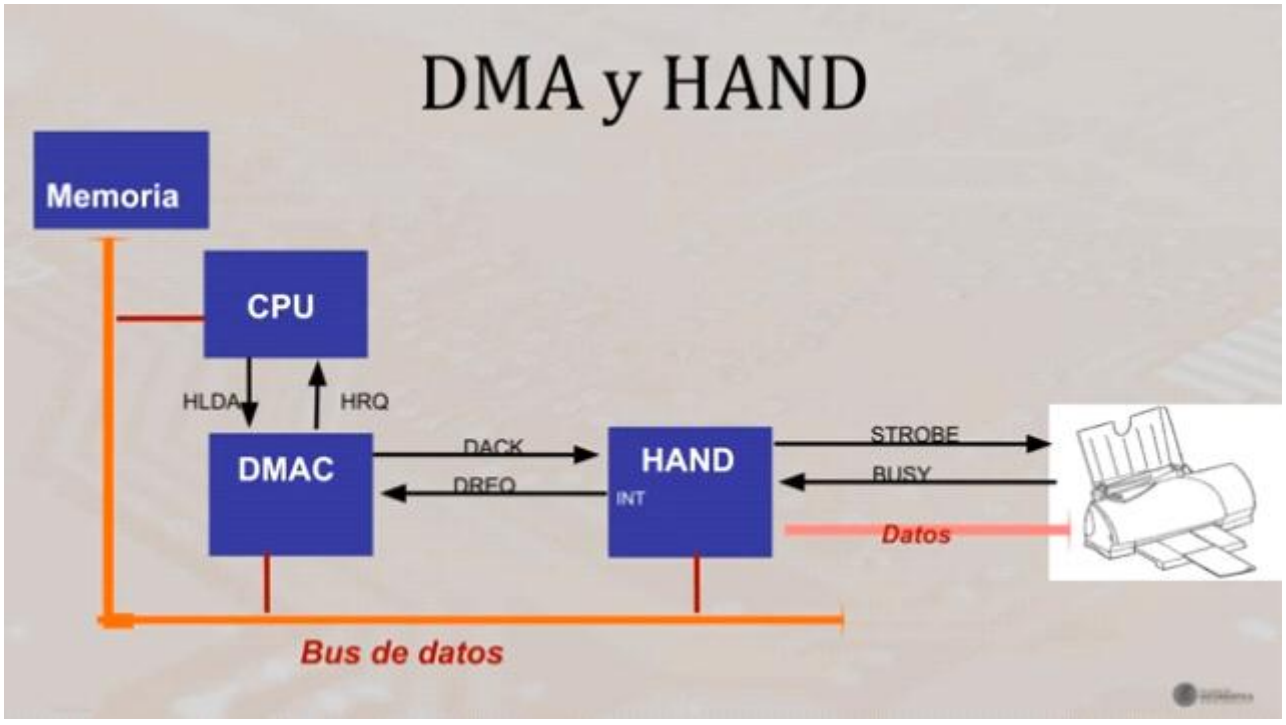
```
sti

;muestro el mensaje original
mov bx , offset msj
mov al , offset fin-offset msj
mov nchar , al
int 7

;inicio la transferencia , los últimos 3 bits en 1 hace que arranque la transferencia
mov al , 00000111b
out dma+7 , al ; arrrranca
int 0
end
```

DMA. Transferencia periférico - memoria

domingo, 25 de octubre de 2020 18:45

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*)




```
pic equ 20h

imr equ 21h

int3 equ 27h

hand equ 40h

dma equ 50h


n_dma equ 20


org 80
```

```
ip_dma dw rut_dma

org 1000h
msj db "aaa aa AAA"
fin db ?
flag db 0

; rrutina de atención del CDMA
org 3000h
; deshabilito el hand
rut_dma: mov al , 0
        out hand+1 , al
        ; indico el fin del lazo
        mov flag , 1
        ; deshabilito las interrupciones del imr
        mov al , 0ffh
        out imr , al
        ; fin de la atención
        mov al , 20h
        out pic , al
        iret

org 2000h
cli

;configuro int3 del PIC
mov al , n_dma
out int3 , al
; dirección origen del bloque
mov ax , offset msj
```

```
out dma, al
mov al , ah
out dma+1 , al
; cantidad de bytes a transferir
mov ax , offset fin - offset msj
out dma+2 , al
mov al , ah
out dma+3 , al
; no podemos dirección a destino, proque vamos a transferir a disp.
; está "cableado" en handshake al dma

; configuración del dma
; mt = 0 st = 1  tt = 0  stop = 0
; st 1 xq de memoria a periférico
; robo de ciclo, mem / periférico , de memoria a periférico
mov al , 00000100b
out dma+6 , al  ; control
; IMR = 1111 0111
mov al , 0f7h
out imr , al
;inicio la transferencia
mov al , 07h
out dma+7 , al
; configuramos el hand por interrupción
mov al , 80h
out hand+1 , al
sti
; cuando se pone en 1 se hace la transferencia
lazo: cmp flag , 1
      jnz lazo
```

int 0

end

DMA vs Interrupciones vs Polling

- Las operaciones de E/S mediante interrupciones/polling
    + Más rápidas en cuánto la inicialización
    + No requieren DMAC
    - Necesitan la intervención directa de la CPU
    - La velocidad de transferencia es limitada
- DMAC
    + Eficiente en CPU y E/S
        Sobretodo al transferir MUCHOS datos.