https://github.com/NahuelArn

Resumen

Un Collage

por VALOR (copia de una variable) copiando en un registro y despues pusheandolo
por REFERENCIA (dirección de la variable) pusheando la direccion de memoria

https://github.com/NahuelArn

Interrupciones por software

INTERRUPCIÓN	PROPÓSITO	USO
INT 0	Frena el programa	Reemplazo del **HLT** (Final del programa)
INT 6	Leer caracter	Guarda en la dirección de **BX** el carácter ingresado
INT 7	Imprime carácter	-Dirección en **BX** de donde empieza a leer -Cantidad de caracteres en **AL**

Escribir un programa que aguarde el ingreso de una clave de cuatro caracteres por teclado sin visualizarla en pantalla. En caso de coincidir con una clave predefinida (y guardada en memoria) que muestre el mensaje "Acceso permitido", caso contrario el mensaje "Acceso denegado".

```
ORG 1000H
MSJ DB "INGRESE UN NUMERO:"
FIN DB ?
CONTRA DB "AB"
CONTRA_PARTE2 DB "CZ"
CANT DB 4H

MENSAJE DB "CORRECTA"
FIN_MENSAJE DB ?
MENSAJE2 DB "INCORRECTA"
FIN_MENSAJE2 DB ?

ORG 3000H
 LEER_CONTRA:MOV BX, SP
 ADD BX,2
 BUCLE: MOV CL,[BX]
 MOV DX, BX
 MOV BX, OFFSET NUM
 INT 6
 MOV AL, 1
 INT 7

 CMP CL,NUM
 JNZ INCORRECTO

 MOV BX, DX
 INC BX
 DEC CANT
 JNZ BUCLE

 CMP CANT,0
 JZ CUMPLE

 INCORRECTO: MOV BX, OFFSET MENSAJE2
 MOV AL, OFFSET FIN_MENSAJE2 - OFFSET MENSAJE2
 INT 7
 JMP TERMINAR


 CUMPLE:MOV BX, OFFSET MENSAJE
 MOV AL, OFFSET FIN_MENSAJE - OFFSET MENSAJE
 INT 7
 JMP TERMINAR

TERMINAR: RET

ORG 1500H
NUM DB ?

ORG 2000H
MOV BX, OFFSET MSJ
MOV AL, OFFSET FIN-OFFSET MSJ
INT 7
MOV BX, CONTRA_PARTE2
PUSH BX
MOV BX, CONTRA
PUSH BX
CALL LEER_CONTRA

INT 0
END
```

Otro Ejemplo

Interrupciones por Software

INT 0, INT 6 e INT 7

Escribir un programa que lea 10 caracteres y cuando termine la lectura imprima la cadena completa en pantalla

```
ORG 1000H
    MENSAJE   DB "Ingrese 10 caracteres!"
    FIN       DB ?
    CADENA    DB ?

ORG 3000H
; Subrutina que imprime consigna en la pantalla
PRINT_MSG: MOV BX, OFFSET MENSAJE
    MOV AL, OFFSET FIN - OFFSET MENSAJE
    INT 7
    RET
```

```
ORG 2000H
    CALL PRINT_MSG ; Imprimimos mensaje
    MOV DL, 10 ; Cantidad de caracteres a leer
    MOV BX, OFFSET CADENA ; Donde vamos a insertar lo leido
    LEER: INT 6
        INC BX ; Proxima posicion en la memoria
        DEC DL
        JNZ LEER

    ; Imprimimos lo leido
    MOV BX, OFFSET CADENA
    MOV AL, 10
    INT 7
    INT 0
    END
```

"Una interrupcion es una subrutina, recibe parametros"

Interrupciones por Hardware Memoria E/S

https://github.com/NahuelArn

IN: Leer desde memoria E/S.
OUT: escribir en memoria E/S/.
Ambas instrucciones solo se pueden usar con el registro **AL.**

Memoria E/S

Lectura y escritura en E/S

La memoria de E/S es igual a la memoria común!

MEMORIA E/S

10H	
11H	
12H	
13H	
14H	
...	

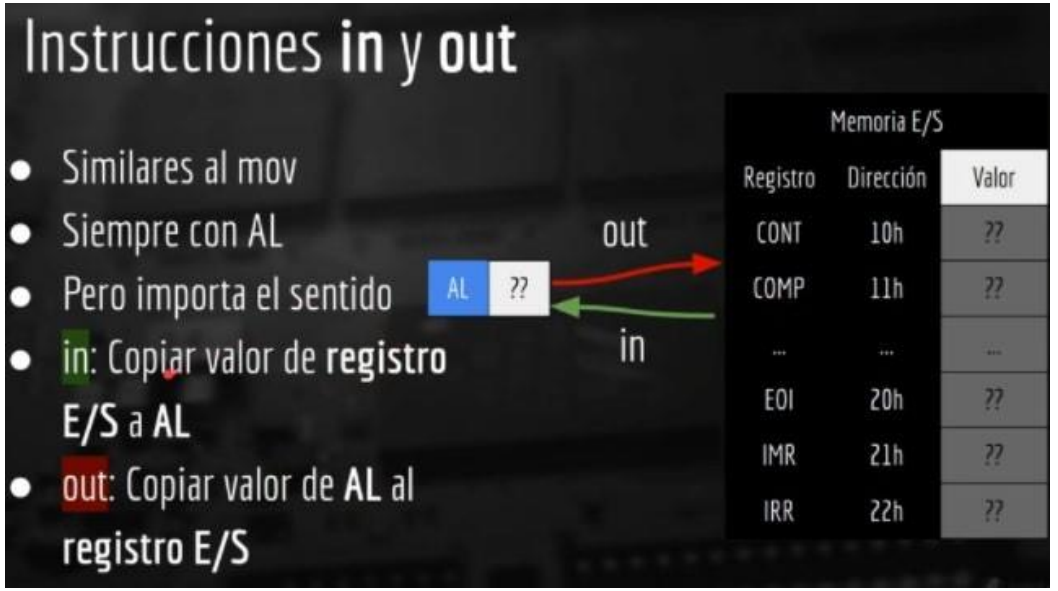
Si son iguales necesito un mecanismo que permita distinguirlas!

- Para leer desde la memoria E/S usaremos **IN**, para escribir en ella **OUT**. Ambas instrucciones **solo se pueden usar** con el registro *AL*
- Ej. lectura: leer el dato que está en la posición 40H de E/S

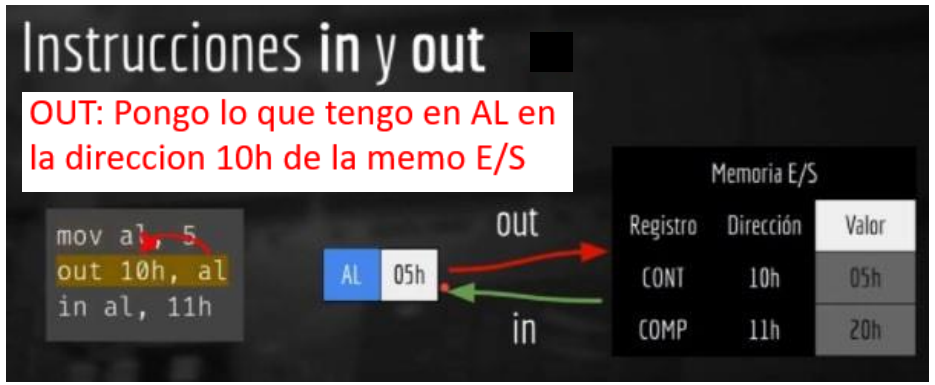
  IN AL, 40H
- Ej.escritura: poner el valor 30 en la posición 50H de E/S

  ~~*OUT 50H, 30*~~

  MOV AL, 30
  OUT 50H, AL



OUT: escribir en memoria E/S



IN: leer desde memo E/S

```
ORG 2000H
; SELECCIONAR ID 10 PARA EL F10
MOV AX, CONTAR
MOV BX, 40
MOV [BX], AX ; 40 = 2000H

; CONFIGURAR EL PIC
CLI
MOV AL, 11111110b
OUT 21H, AL ; IMR = 11111110

MOV AL, 10
OUT 24H, AL ; INT 0 = 10
STI

LOOP: JMP LOOP

INT 0


ORG 3000H
; Subrutina que atiende la interrupción de F10
CONTAR: INC DI
; AVISAR AL PIC QUE TERMINAMOS!
MOV AL, 20H
OUT 20H, AL ; EOI = 20H
IRET


END
```

https://github.com/NahuelArn

PIO (LEDS E INTERRUPTORES)

PIO

REGISTROS		
NOMBRE	**DIRECCIÓN**	**PROPÓSITO**
PA	30H	Transferencia de datos
PB	31H	Transferencia de datos
CA	32H	Configuración, bit a bit, de entrada (1), o salida (0)
CB	33H	Configuración, bit a bit, de entrada (1), o salida (0)

-Configuraciones de **PIO:**
PIO con luces e interruptores

30H PA: llaves.
31H PB: luces.
32H CA: llaves
33H CB luces

Escribir un programa que permite encender y apagar las luces mediante las llaves. El programa no deberá terminar nunca, y continuamente revisar el estado de las llaves, y actualizar de forma consecuente el estado de las luces. La actualización se realiza simplemente prendiendo la luz i si la llave
i correspondiente está encendida (valor 1), y apagándola en caso contrario. Por ejemplo, si solo la primera llave está encendida, entonces solo la primera luz se debe quedar encendida.

```
CA EQU 32H ;LLAVES
PA EQU 30H
CB EQU 33H ;LUCES
PB EQU 31H

ORG 1000H


ORG 2000H
;INICIALIZO LAS LUCES ENTRADA
MOV AL, 0
OUT CB, AL
;INICIALIZO LAS LLAVES SALIDA
MOV AL, 0FFH ;1111 1111
OUT CA, AL
;-----------------------------------
LOOP: IN AL, PA ;LEO EL ESTADO DE LAS LLAVES Y LO PONGO EN AL

OUT PB, AL ;CARGO EL ESTADO DE LAS LLAVES EN LAS LUCES
JMP LOOP
INT 0
END
```

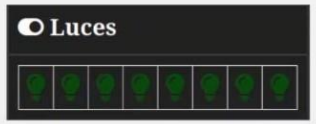
PIO

Ejemplo salida

1. Prender todas las luces. Recordar que:

- Las luces están ligadas al puerto **PB**. 1 significa encendida
- Las queremos a todas de salida!

```
MOV AL, 00000000b
OUT 33H, AL ; CB = 00000000
MOV AL, 11111111b
OUT 31H, AL ; PB = 11111111
```



2. Prender solo la primera (desde derecha)

```
MOV AL, 01H
OUT 31H, AL ; PB = 00000001
```

Si! Podemos usar hexadecimales y decimales!

-Configuraciones de **PIO:**
PIO con luces e interruptores
30H PA: llaves. 32H CA: llaves
31H PB: luces. 33H CB luces

PIO

Ejemplo entrada

Leer el estado de las llaves y prender las luces de aquellas llaves que estén en 1. Recuerden:

- Las llaves están ligadas al puerto **PA**. Las luces al **PB**.
- Queremos todos los bits de **PA** de entrada y todos los de **PB** de salida!

1. Configuramos **PA** y **PB**

```
MOV AL, 11111111b
OUT 32H, AL ; CA = 11111111
MOV AL, 00000000b
OUT 33H, AL ; CB = 00000000
```

2. Leemos **PA**
El estado de las llaves

```
IN AL, 30H
```

3. Escribimos en **PB**
y lo ponemos en las luces

```
OUT 31H, AL
```

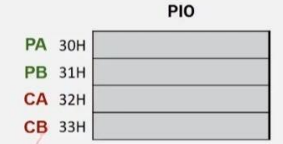
PIO (leds e interruptores)	Puerto	Nombre corto	Valor típico	
Interruptores	30h	PA	-	
Leds	31h	PB	-	
Configuración	32h	CA	11111111	Todos los bits de PA son de entrada
Configuración	33h	CB	00000000	Todos los bits de PB son de salida

PIO

Funcionamiento

Los puertos funcionan de la siguiente manera

- Cada celda (también llamado *registro*) consta de 8 bits
- Debemos **configurar** cada bit de **datos** como entrada o salida
- En los puertos de **configuración** debemos poner un 0 para que ese bit en el puerto de **datos** sea de salida, 1 para que sea de entrada



Ej.: queremos que el **PA** tenga todos los bits como entrada excepto el menos significativo

- Debemos configurar **CA**
- Todos en 1 excepto el menos significativo (11111110)

```
MOV AL, 11111110b
OUT 32H, AL ; CA = 11111110
```

PIO con Impresora https://github.com/NahuelArn

PIO CON IMPRESORA

A: ESTADO

Bit 0: busy (1 ocupada, 0 desocupada).
Bit 1: strobe (1 activado, 0 desactivado).

B: DATO

PIO (impresora)	Puerto	Nombre corto	Valor típico	
Estado	30h	PA	-	Acá se lee si está desocupada y se hace el strobe
Dato	31h	PB	-	Acá se escribe el caracter a imprimir
Configuración	32h	CA	11111101	Todos los bits de PA son de entrada MENOS el de strobe
Configuración	33h	CB	00000000	Todos los bits de PB son de salida

Escribir un programa que solicita el ingreso de cinco caracteres por teclado y los envía de a uno por vez a la impresora a través de la PIO a medida que se van ingresando. No es necesario mostrar los caracteres en la pantalla.

EL PIO: NO PUEDE TRABAJAR CON INTERRUPCIONES!
ESO ES PARA EL HANDSHAKE

Escribir un programa que envíe datos a la impresora a través del **PIO**

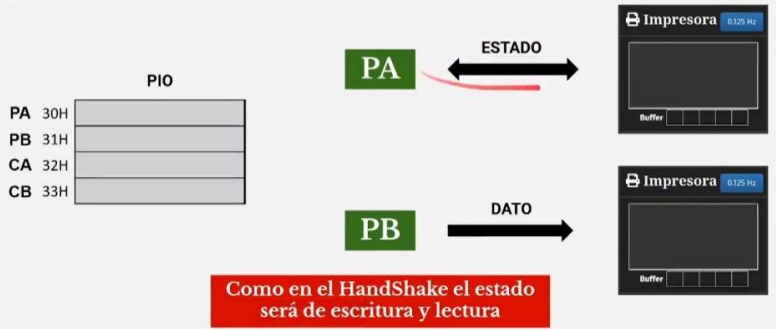
1. ¿Cómo configuramos el **PA** a partir de **CA**? Strobe en 0 (salida) y Busy en 1 (entrada)
2. ¿Cómo configuramos el **PB** a partir de **CB**? Todos de salida!
3. Consultaremos constantemente si está libre Chequear si el bit **Busy** = 0
4. Cuando la impresora esté libre mandamos el caracter a **PB** (31H)
5. Hasta que no mandemos el bit de Strobe en 1 no se va a imprimir!
6. Después de enviar el Strobe en 1, debemos volver a ponerlo en 0

Del punto 3 al 6 debemos repetirlo para cada caracter

Impresora

Configuración por PIO

Recordemos la estructura del PIO



Como en el HandShake el estado será de escritura y lectura

Registro de Estado

Veamos cuáles bits del registro **estado** son de **entrada** y cuáles de **salida**...

- **Bit 0 (busy)** - 1 si está ocupada la impresora, 0 si está libre
- **Bit 1 (strobe)** - seteando el bit en 1 le avisamos a la impresora que dejamos un caracter en **DATO** para que lo imprima

```
;CONFIG PIO
PA EQU 30H
PB EQU 31H
;PARTE DE DATOS DEL PIO
CA EQU 32H
CB EQU 33H

ORG 3000H
INI_IMPRESORA_PIO: NOP
MOV AL, 11111101B ;STROBE EN 0 (SALIDA) Y BUSY EN 1(ENTRADA)
OUT CA, AL

MOV AL, 0 ;TODOS DE SALIDA
OUT CB, AL
RET

PEDIR_CARAC: NOP        ;PARA PEDIR UN CARAC NECESITO LA DIRECCION DONDE SE VA GUARDAR EN BX Y EN AL LA CANTIDAD
PUSH BX
MOV BX, OFFSET CARACTER  ;
MOV AL, 1
INT 6
POP BX
RET

FLIP_FLOP: NOP  ;PULSO DE STROBE (INICIALIZAR EN 0) Y CUANDO RECIBE UNA LETRA QUE LO ESPERE EN 1 Y DESPUES ABAJO
PUSH AX

;STROBE EN 0
;LO DEJO ABAJO OTRA VEZ ; ESO SERIA UNA ESPECIE DE SOLO ACTIVACION POR FLANCO ASCENDENTE

IN AL, PA ; ME TRAIGO LA INFO ACTUAL
AND AL, 0FDH ;11111101B  ;FUERZO EL STROBE EN 0
OUT PA, AL


;STROBE EN 1
IN AL, PA ;PIDO EL DATO ACTUAL
OR AL, 2 ; FUERZO STROBE EN 1 ;00000010B
OUT PA, AL

;STROBE EN 0
;LO DEJO ABAJO OTRA VEZ ; ESO SERIA UNA ESPECIE DE SOLO ACTIVACION POR FLANCO ASCENDENTE

IN AL, PA ; ME TRAIGO LA INFO ACTUAL
AND AL, 0FDH ;11111101B  ;FUERZO EL STROBE EN 0
OUT PA, AL
POP AX
RET

POL: NOP   ;ENCARGADO DE VERIFICAR Q LA IMPRESORA NO ESTE OCUPADA (POLLING)
PUSH AX
IN AL, PA  ;ME TRAIGO EL ESTADO ACTUAL DE PA A AL
AND AL, 1  ; LE HAGO UN AND PARA VER SI SIGUE OCUPADO
JNZ POL    ;MIENTRAS NO ESTE LIBRE, SIGO CONSULTADO
POP AX
RET

PONER_CARACTER_A_IMPRESORA: NOP
PUSH AX
PUSH BX
MOV BX, OFFSET CARACTER
MOV AL, [BX]
OUT PB, AL
POP BX
POP AX
RET

ORG 1000H
CARACTER DB ?

ORG 2000H
CALL INI_IMPRESORA_PIO
MOV DL, 5

FOR: NOP  ;FOR DE 5 DESPUES CORTA

CALL POL

CALL PEDIR_CARAC
CALL PONER_CARACTER_A_IMPRESORA

CALL FLIP_FLOP

DEC DL
CMP DL, 0
JZ TERMIND
JMP FOR

TERMINO: INT 0
END
```

PIC : interrupciones

https://github.com/NahuelArn

REGISTROS		
NOMBRE	**DIRECCIÓN**	**PROPÓSITO**
EOI	20H	Avisa al PIC que se terminó una interrupción (Antes de volver de las subrutina de la interrupción debemos poner el valor 20H en el EOI
IMR	21H	Sirve para habilitar(0) o deshabilitar(1) alguna interrupción. Ej: xxxx1110 (INT0 habilitada).
IRR	22H	Sirve para indicar cuál dispositivo solicita (1) o no solicita (0) la interrupción.
ISR	23H	Indica cuál dispositivo está siendo atendido
INT0-INT3	24H-27H	Almacena la ID de la interrupción.

Tenemos 4 tipos de dispositivos externos

- F10 - INT 0
- TIMER - INT 1

  Emite una interrupción cuando el registro CONT y COMP son iguales.

REGISTROS		
NOMBRE	**DIRECCIÓN**	**PROPÓSITO**
CONT	10H	Se incrementa en uno por segundo
COMP	11H	Contiene el tiempo límite para interrumpir

- HANDSHAKE - INT 2

REGISTROS		
NOMBRE	**DIRECCIÓN**	**PROPÓSITO**
DATO	40H	Carácter a imprimir
ESTADO	41H	Estado y control Bit 0 (Busy) - 1 ocupada , 0 libre be) - 1 activado, 0 desactivado Bit 7 (Interrupción) - 1 activado, 0 desactivado

El HANDSHAKE puede generar interrupciones, y no hay que verificar si está ocupada la impresora, mientras que en el PIO si. El HANDSHAKE es solo para impresoras, mientras que el PIO no.

Partamos desde un ej. simple: contar las veces que se presionó la tecla F10 en DL

Vamos a realizar los siguientes pasos:

1. Escribir la subrutina que se ejecutará cuando se produzca la interrupción (que finaliza con **IRET**)
2. Elegir un ID de interrupción (cualquiera menos 0, 3, 6 ó 7)
3. Poner la dirección de la subrutina en el ***Vector de interrupciones*** (ya veremos qué es esto)
4. Configurar el **PIC**
   a. Bloquear las interrupciones con la sentencia **CLI**
   b. Poner el ID en el PIC para la interrupción que nos interesa
   c. Desenmascarar la interrupción
   d. Desbloquear las interrupciones con la sentencia **STI**

El **PIC** contiene los siguientes campos

PIC

EOI	20H	Le avisa al **PIC** que la interrupción ya fue atendida
IMR	21H	Para habilitar o deshabilitar alguna interrupción
IRR	22H	Indica cuáles dispositivos externos solicitan interrumpir
ISR	23H	Indica cuál dispositivo externo está siendo atendido
INT 0	24H	Contiene *ID* asignado al F10
INT 1	25H	Contiene *ID* asignado al Timer
INT 2	26H	Contiene *ID* asignado al Handshake
INT 3	27H	Contiene *ID* asignado al CDMA

1. Escribir un programa que permita seleccionar una letra del abecedario al azar. El código de la letra debe generarse en
un registro que incremente su valor desde el código de A hasta el de Z continuamente. La letra debe quedar seleccionada al
presionarse la tecla F10 y debe mostrarse de inmediato en la pantalla de comandos.

```
ORG 40
PUNTERO DW 3000H
ORG 3000H
CONTAR: INT 7 ;IMPRIMO EL CARACTER ACTUAL
 MOV AL, 20H  ; AVISO AL PIC QUE TERMINAMOS
 OUT 20H, AL  ;EOI = 20H
IRET          ;VUELVO DE LA INTERRUPCION

ORG 1000H
CARACTER DB "A"

ORG 2000H
              ;ACA SETEO TODO
 ;MOV AX, CONTAR ; EN AX GUARDO LA DIRECCION DE CONTAR
 ;MOV BX, 40  ; A BX LE ASIGNO 40
 ;MOV [BX], AX  ;EN ;LA DIRECCION 40H / PONGO LA DIRECCION DE CONTAR


 CLI      ;LLAMO A LA INTERRUPCION / CONFIGURAR EL PIC
 MOV AL, 11111110B   ;ELIJO Q INTERRUPCION VOY A USAR, EN ESTE CASO F10
 OUT 21H, AL         ;ACA DEFINIMOS QUE INTERRUPCION VAMOS A USAR

 MOV AL, 10  ;ID ACA DEFINIMOS EL ID POR EL CUAL SE VA A MULTIPLCIAR X 4
 OUT 24H, AL ;ACA DECIMOS QUE VAMOS A ESCRIBIR EN LA DIRECCION 24H EL ID QUE PUSIMOS EN AL
 STI

 MOV BX, OFFSET CARACTER  ;ME POSICIONO EN LA DIREC DE CARACTER

 REINICIAR: MOV BYTE PTR [BX],40H ;UNO ANTES DE A EN HEXA

 LOOP: INC BYTE PTR [BX] ;INCREMENTO EN 1, LO QUE CONTIENE [BX] OSEA QUE VOY A TENER "A"
 MOV AL,1
 ;MOV AH,0
 CMP BYTE PTR [BX],5AH ;me fijo que no llegue al final
 JZ REINICIAR ;SI LLEGUE A "Z" VUELVO A EMPEZAR DESDE A
 JMP LOOP ;SINO PASO A LA SOGUIENTE LETRA, INCREMENTANDO EN UNO EL ASSCI
INT 0
END
```

Handshake por **consulta de estado** (polling)

https://github.com/NahuelArn

Handshake

Registros

Así como el timer tiene **COMP** y **CONT** el handshake tiene sus propios registros

Estos bits tiene diferentes significados dependiendo de si los pusimos en **entrada** o **salida**

Handshake

Ejercicio 1

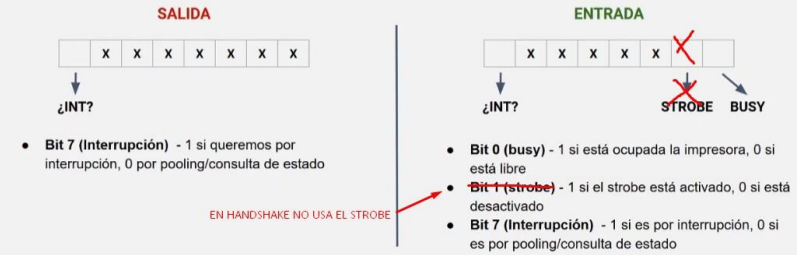
Escribir un programa que envíe datos a la impresora a través del Handshake. La comunicación se debe establecer por **consulta de estado** (polling)

1. Debemos configurar ¿En qué configuramos el bit de **INT**? En 0! No queremos interrupciones!
2. Consultaremos constantemente si está libre Chequear si el bit **Busy** = 0
3. Cuando la impresora esté libre mandamos el caracter a **DATO** (40H)

Handshake

Registros

Los bits del registro **estado** tiene diferentes significados dependiendo de si los pusimos en **entrada** o **salida**



Escribir un programa que envíe datos a la impresora a través del Handshake. La comunicación se debe establecer por **consulta de estado** (polling)

```
;Ejemplo Gena Poling
HAND_DATO    EQU 40H
HAND_ESTADO  EQU 41H

ORG 1000H
MENSAJE DB "El Handshake la rompe"
FIN           DB ?


ORG 2000H
; Configuro el Handshake para el polling
IN AL, HAND_ESTADO ; Tomo estado actual
AND AL, 07FH ; 7FH = 01111111
OUT HAND_ESTADO, AL ; Estado = 0xxxxxxx

; Recorremos el mensaje y lo enviamos caracter
; a caracter hacia la impresora
MOV BX, OFFSET MENSAJE ; Para recorrer el mensaje
POLL: IN AL, HAND_ESTADO ; Tomo el estado actual
    AND AL, 1 ; Chequeo el primer bit
    JNZ POLL ; Mientras sea 1 sigo en el loop
    MOV AL, [BX] ; Tomo el caracter
    OUT HAND_DATO, AL ; Lo envio al registro de datos
    INC BX ; Avanzo a la siguiente posicion
    CMP BX, OFFSET FIN ; Chequeo si llegue al final
    JNZ POLL
INT 0
END
```

https://github.com/NahuelArn

Handshake por **interrupcion**

CONFIGURAR INTERRUPCIÓN

Pasos para configurar una interrupción:

1. Escribir la subrutina que se ejecutará cuando se produzca la interrupción (Esta finaliza con el **IRET**)
2. Elegir un ID de interrupción (cualquiera menos 0, 3, 6 o 7)
3. Poner la dirección de la subrutina en el ***Vector de interrupciones***
4. Configurar el **PIC**
   A. Bloquear las interrupciones con la sentencia **CLI**
   B. Poner el ID en el PIC para interrupción que nos interesa
   C. Habilitar/Deshabilitar la interrupción
   D. Desbloquear las interrupciones con la sentencia **STI**

Impresora Handshake, nos facilita el pulso de Strobe

Conceptual
Si nosotros escribimos es registro de configuracion
Si lo leemos es un registro de estado

Dispositivos de E/S

¿Cuáles son los diferentes dispositivos y dónde se conectan?

handshake (impresora)	Puerto	
Dato	40h	
Estado	41h	0xxxxxxx => por consulta de estado 1xxxxxxx => por interrupciones xxxxxxx0 => libre xxxxxxx1 => ocupada

Si se utiliza por consulta de estado (configurar estado con el bit más significativo en 0) entonces se parece a la PIO (loopear hasta que el bit menos significativo -busy- valga 0)

Escribir un programa que envíe datos a la impresora a través del Handshake. La comunicación se debe establecer por **interrupción**

1. Debemos configurar ¿En qué configuramos el bit de **INT**? En 1! ~~No~~ queremos interrupciones!
2. Ya no consultaremos constantemente si está libre Nos interrumpirá cuando esté libre!
3. Cuando la impresora nos interrumpa mandamos el caracter a **DATO** (40H)

```
;HAND-SHAKE
HAND_DATO EQU 40H
HAND_ESTADO EQU 41H
;PIC
EOI EQU 20H
IMR EQU 21H
HANDSHAKE EQU 26H

ORG 1000H
STRING DB "00000" ;0,0,0,0,0
SALTO DB 0AH
FLAG DB 0

;STRING DB ?;
ORG 40
PUNTERO DW 3000H
ORG 3000H

CMP FLAG, 1
JZ INVERSO
;LE MANDO EL CARACTER / NORMAL
MOV AL, [BX]
OUT HAND_DATO, AL
INC BX
;DEC DL
CMP BX, OFFSET SALTO
JNZ CASI_FIN
INC FLAG
JMP CASI_FIN

;LE MANDO EL CARACTER / INVERSO

INVERSO: NOP
MOV AL, [BX]
OUT HAND_DATO, AL
DEC BX
DEC DH          ;INVERSO
JNZ CASI_FIN

;AVISO QUE YA TERMINE
MOV AL, EOI
OUT EOI, AL

;DESACTIVO EL HANDSHAKE PARA QUE NO ME JODA
IN AL, HAND_ESTADO
AND AL, 07FH ;01111111
OUT HAND_ESTADO, AL ;ESTADO = 0XXXXXXX

IRET

;AVISO QUE YA CASI TERMINE
CASI_FIN: MOV AL, EOI
OUT EOI, AL
IRET


ORG 2000H

;PEDIR CARACTERES
MOV DL, 5
MOV BX, OFFSET STRING
LOPARDO: CMP DL, 0
JZ FIN_PEDIDO
INT 6
INC BX
DEC DL
JMP LOPARDO
FIN_PEDIDO: NOP

CLI
MOV AL, 0FBH ;11111011B AVISO AL IMR QUE TIPO DE INTERRUPCION ES, EN ESTE CASO ES EL HANDSHAKE
OUT IMR, AL

MOV AL, 10          ;ID
OUT HANDSHAKE, AL   ;1004-40

;CONFIGURAR EL HAND-SHAKE PARA INTERRUPCION
IN AL, HAND_ESTADO
OR AL, 80H ;10000000
OUT HAND_ESTADO, AL ;1XXXXXXX

MOV DL, 6
MOV DH, 6
MOV BX, OFFSET STRING
STI

;ESTA ES LA VENTAJA DEL HANDSHAKE POR INTERRUPCION NO TIENE QUE ESTAR PENDIENTE
;SI LA IMPRESORA ESTA LIBRE PUEDE HACER OTRAS COSAS EN LA PC
LAZO: NOP
 NOP ; Esto es el Counter
 NOP ; Esto es Youtube
 NOP ; Esto es el Chrome
JMP LAZO

INT 0
END
```

IMPRIMIR CON HANDSHAKE CON INTERRUPCIONES

```
CONSTANTES:
EOI. 20H | IMR 21H | INT1 26H
DATO 40H | ESTADO 41H

ORG 1000H
MSJ DB "INTERRUPCION"
FIN DB ?

ORG 2000H
CLI
; CONF IMR PARA USAR HAND
MOV AL, 1111 1011B
OUT IMR, AL

; CONF ID DE INTERRUPCION
MOV AL, 20 (ID DE INT, ID x 4 = 80
OUT INT2, AL

; CONF ESTADO
IN AL, ESTADO
OR AL, 1000 0000B ; O 80H
MOV BX, OFFSET MSJ
MOV CL, OFFSET FIN- OFFSET MSJ

STI
LAZO: CMP CL. 0
        JNZ LAZO
        INT 0
        END

ORG 80
IP_INT2 DW 3000H

ORG 3000H
RUT_INT2 : MOV AL, [BX]
           OUT DATO, AL
           INC BX
           DEC CL
          JNZ VOLVER

 SI CL= O TERMINAMOS LA
EJECUCION Y DESACTIVAMOS
TODAS LAS INTERRUPCIONES
           MOV AL, 111 1111B
           OUT IMR, AL

VOLVER: MOV AL, EOI
           OUT EOI, AL
IRET
```

IMPRIMIR CON HANDSHAKE MEDIANTE CONSULTA DE ESTADO

```
CONSTANTES:
DATO EQU  40H
ESTADO EQU 41H

ORG 1000H
MSJ DB "ESTAS OCUPADO??"
FIN DB ?

ORG 2000H
MOV BX, OFFSET MSJ
MOV CL, OFFSET FIN- OFFSET MSJ

POLL: IN AL, ESTADO
      AND AL, 1
      JNZ POLL

; SI AVANZO ES PORQUE ESTA DESOCUPADA
; MANDO CARACTER

      MOV AL, [BX]
      OUT DATO, AL
      INC BX
      DEC CL
      JNZ POLL
      INT 0
      END
```

https://github.com/NahuelArn

IMPRIMIR MEDIANTE PIO

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

ORG 1000H
MSJ DB "IMPRIMO X PIO"
FIN DB ?

ORG 2000H

; 0 STROBE
; 1 BUSY
MOV AL, 11111101B
OUT CA, AL

; CB TODOS SALIDA SIEMPRE
MOV AL, 0
OUT CB, AL

;FUERZO STROBE A 0
IN AL, PA
AND AL, 11111101B
OUT PA, AL

; MUEVO DIR DE PRIMER CAR
MOV BX, OFFSET MSJ
MOV CL, OFFSET FIN - OFFSET
MSJ


POLL: IN AL, PA
      AND AL, 1
      JNZ POLL

MANDO CAR A IMPRIMIR
      MOV AL, [BX]
      OUT PB, AL

      ; FUERZO STROBE A 1
      IN AL, PA
      OR AL, 00000010B
      OUT PA, AL
      ; FUERZO STROBE A 0
      IN AL, PA
      AND AL,11111101B
      OUT PA, AL

      INC BX
      DEC CL
      JNZ POLL
      INT 0
      END
```

https://github.com/NahuelArn

BUENA DATA

RI:"se utiliza cuando hay operaciones en las que se involucra la memoria principal, para no modificar el registro q apunta a los operandos, como direccionamiento indirecto o directo por memoria"

Genaro: el registro RI se utiliza para almacenar el valor actual de BX cuando hacés direccionamiento indirecto. Es decir, contiene la dirección a la que se tiene que ir a buscar el valor en memoria. Por ejemplo, si tenememos el siguiente código: MOV BX, 1006H MOV AL, [BX] Al momento de ejecutarse la segunda línea, en el registro RI estará contenido el valor 1006H, que es la dirección en memoria a donde apunta BX

EOI: se usa para comandos, para decir que es el fin de una interrupcion escribir 20H

INT0-INT7: INT 0 funciona igual que el HLT indica la finalización del programa y INT 7 se usa para imprimir en pantalla, en AL le das cuantos caracteres va a imprimir y en bx la dirección desde donde va a imprimir

IMR: Nos permite definir que interrupciones vamos a atender y cuales ignorar..

1 deshabilitada / 0 HABILITADA

IRR: te marca en 1 todas las interrupciones que pretendeN ser atendidas

Registro de petición de interrupción. Almacena las interrupciones demandadas hasta el momento. Indica cuales dispositivos externos solicitan interrumpir Así, al activarse una entrada de interrupción el bit correspondiente se pone a 1, tornándose a 0 cuando ésta pasa a ser atendida (bits 0...7 se asocian a las entradas INT0...INT7, respectivamente).

ISR: Registro de Interrupción en Servicio. Indica cual es la interrupción que está siendo atendida, SOLO TE MARCA 1 NO PUEDE ESTAR ATENDIENDO MAS DE 1 A LA VEZ mediante la puesta en 1 del bit asociado a esa entrada de interrupción (bit 0 se asocia a la entrada INT0 ... bit 7 se asocia a la entrada INT7).

IP: DONDE ESTA PARADO EL COMPILADOR APUNTA A DONDE VA TENER LA PROXIMA INTRUCCION

IR: registro de instrucción, contiene la instrucción que hay que ejecutar.

Las intrucciones CLI Y STI HABILITAN Y DESHABILITAN EL PODER RECIBIR INTERRUPCIONES, asi no se rompe o explota todo mientras ya estamos con una interrupcion