Resumen de las Prácticas 1-3

Arquitectura de Computadoras – Informática U.N.L.P.

Práctica 1: E/S con Interrupciones por Software. Subrutinas con Pasaje de Parámetros.

INT 7

VonSim utiliza la interrupción INT 7 para imprimir un string en la pantalla (siempre utiliza valores ASCII). Previo a usarla, se debe mover a BX la dirección del string, y a AL la longitud del mismo.

CÓDIGO	**SALIDA POR PANTALLA**
ORG 1000H MSJ DB "ARQUITECTURA DE COMPUTADORAS-" DB "FACULTAD DE INFORMATICA-" DB 55H DB 4EH DB 4CH DB 50H FIN DB ? ORG 2000H MOV BX, OFFSET MSJ MOV AL, OFFSET FIN - OFFSET MSJ INT 7 INT 0 END	ARQUITECTURA DE COMPUTADORAS-FACULTAD DE INFORMATICA-UNLP

INT 6

Lee un dígito por el teclado del simulador. Se almacena en BX la dirección de la etiqueta de memoria a almacenar el dígito, y luego se invoca INT 6.

CÓDIGO	**SALIDA POR PANTALLA**
ORG 1000H MSJ DB "INGRESE UN NUMERO:"	INGRESE UN NUMERO: *(DIGITO INGRESADO)*

```
FIN DB ?
ORG 1500H
NUM DB ?

ORG 2000H
MOV BX, OFFSET MSJ
MOV AL, OFFSET FIN-OFFSET MSJ
INT 7
MOV BX, OFFSET NUM
INT 6
MOV AL, 1
INT 7
MOV CL, NUM
INT 0
END
```

Pasaje de parámetros

Para invocar una subrutina, se debe introducir la palabra reservada ‘call’ y el nombre de la misma. Para retornar el dato transformado, utilizar ‘ret’.

Sin embargo, antes hay que decidir **a través de donde** pasar el dato, y si lo hacemos **por valor o por referencia**.

Por valor:

- Si se quiere enviar el dato **a través de un registro**, hay que cargar el dato a dicho registro y llamar inmediatamente a la subrutina:

```
mov ax,5
call subrutina
```

```
mov dl, 5
call subrutina
```

- Por otro lado, para enviar el dato **a través de la pila** se lo debe pushear:

```
mov bx, 5
push bx
call subrutina
pop bx
```

Por referencia:

El pasaje de parámetros a una subrutina por referencia implica enviarle no el dato sino **su dirección en la memoria**.

- **A través de un registro:**

```
mov dx, offset A
call subrutina
```

- **A través de la pila:**

```
mov cx, offset A
push cx
call subrutina
pop cx
```

Práctica 2: E/S mediante consultas de estado. Ejercicios integradores.

Dirección	Dispositivo	Nombre	Función
30h	PIO	PA	Registro de datos del puerto paralelo de comunicación A. Tiene 8 conectores programables, cada uno puede enviar o recibir un bit.
31h	PIO	PB	Registro de datos del puerto paralelo de comunicación B. Tiene 8 conectores programables, cada uno puede enviar o recibir un bit. Funciona de forma independiente a PA.
32h	PIO	CA	Registro de configuración del puerto paralelo de comunicación A (PA). Tiene 8 bits, para configurar el sentido de cada uno de los conectores correspondientes. Un valor de 0 indica que el conector es de salida, y un bit de 1 indica que es de entrada.
33h	PIO	CB	Idem CA pero para el puerto PB
40h	Handshake	Dato	Registro de datos del puerto paralelo Dato para comunicarse con la impresora. Sirve para enviar caracteres a imprimir a dicho dispositivo.
41h	Handshake	Estado	Registro de estado y configuración. Tiene 8 bits, pero solo los bits 0, 1 y 7 se utilizan para interactuar con la impresora. El formato del registro es IXXXXXSB, donde I es el bit que configura al Handshake en modo consulta de estados o interrupciones. S y B indican el valor de la señales de Strobe y Busy, respectivamente, de la impresora.

Instrucciones de memoria E/S

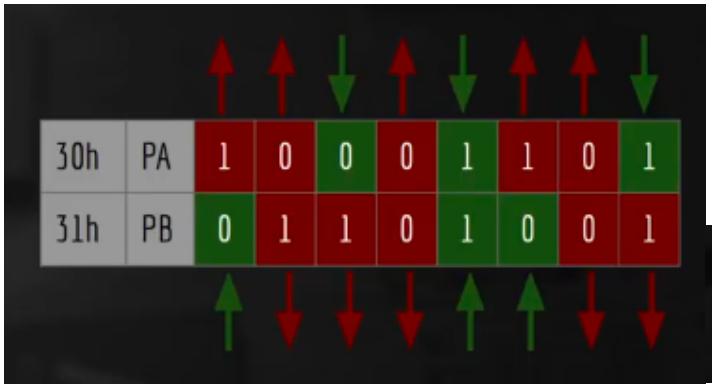
Para interactuar con la memoria de entrada/salida, se utilizan las instrucciones in y out.

- Son similares al MOV.
- Siempre con AL.
- Importa el sentido.
- **In:** copia valor del registro E/S a AL.

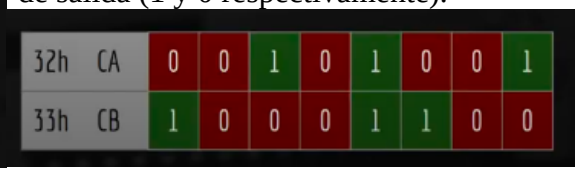
- **Out:** copia valor del registro AL al de E/S.

PIO – Programmable Input Output Device

El PIO consta de dos puertos A y B de 8 bits cada uno. Cada uno de estos bits se puede programar para que sea de entrada o de salida.



Los registros **PA** y **PB** son de datos, y se alojan en las dir. 30h y 31h respectivamente. Los registros **CA** y **CB** son de configuración, y en las direcciones contiguas determinan si cada bit va a ser de entrada o de salida (1 y 0 respectivamente).



Llaves

Puerto **PA** del PIO conectado a las 8 Llaves (dispositivo de entrada). Puerto **CA** siempre en 0FFh.

Luces

Puerto **PB** del PIO conectado a las 8 Luces (dispositivo de salida). Cada bit del PB corresponde a una luz. Puerto **CB** siempre en 0.

1 → encendida; 0 → apagada. Sin embargo, para que todo pueda funcionar, debo configurar a CB con todos bits 0 (00000000). De esta manera, CB se configura como salida y puedo enviar los bits por PB.

- **EJEMPLO 1:** si deseo que las luces se prendan así:

Debo configurar el programa así:

```
PB EQU 31H          ; PB se encuentra en 31H
CB EQU 33H          ; CB se encuentra en 33H

ORG 2000H
MOV AL, 0           ; muevo 00000000 a AL
OUT CB, AL          ; OUT a CB
```

```
MOV AL, 0F0H      ; muevo 11110000 a AL
OUT PB, AL        ; OUT a PB

HLT
END
```

- **EJEMPLO 2:** si deseo prender todas las luces y luego apagarlas, configuro así:

```
PB EQU 31H        ; PB se encuentra en 31H
CB EQU 33H        ; CB se encuentra en 33H

ORG 2000H
MOV AL, 00H       ; muevo 00000000 a AL
OUT CB, AL        ; OUT a CB

MOV AL, 0FFH      ; muevo 11111111 a AL (todas las luces prendidas)
OUT PB, AL        ; OUT a PB

MOV AL, 00h       ; muevo 00000000 a AL (todas las luces apagadas)
OUT PB, AL        ; OUT a PB

HLT
END
```

- **EJEMPLO 3:** si deseo prender y apagar las luces 100 veces, configuro así:

```
PB EQU 31H
CB EQU 33H

ORG 2000h
mov AL, 00H
out CB, AL
MOV CL, 100

LOOP:
  mov AL, 0FFH
  out PB, AL

  mov AL, 00H
  out PB, AL
```

```
DEC CL          ; decremento CL
JNZ LOOP        ; hasta que CL no haya llegado a 0, no paro

HLT
END
```

Impresora

- Puerto **PB** del PIO al puerto de datos de 8 bits de la impresora (salida)
- Puerto **PA** del PIO:
  - El bit 0 a la señal **busy** de la impresora (entrada)
  - El bit 1 a la señal **strobe** de la impresora (salida)

Pasos para imprimir una cadena con el PIO:

Subrutinas a crear:

1. **Mediante CA configuro el bit de strobe en salida y el de busy en entrada:**

```
ini_pio: mov al, 0FDh        ; strobe salida (0), busy entrada (1)
out CA, al
```

2. **Pongo todos los bits de CB en salida (00000000):**

```
mov al, 0        ; puerto de datos todo salida
out CB, al
ret
```

3. **Hago el poll para que, mientras busy esté en 1, itere:**

```
poll: in al, PA
and al,1
jnz poll
ret
```

4. **Pongo el bit de strobe en 0:**

```
strobe0: in al, PA     ; envio 0
and al, 0FDh
out PA, al
ret
```

5. **Pongo el bit de strobe en 1:**

```
strobe0: in al, PA     ; envio 1
or al, 02h
out PA, al
ret
```

6. **Subrutina imprimir_caracter:**

```
imprimir_caracter: push ax
call poll
pop ax
out PB, al
call strobe1
call strobe0
ret
```

Programa principal:

```
ORG 2000h
call ini_pio      ; configuro el PIO
call strobe0      ; pongo el bit de strobe en 0
mov bx, offset msj      ; muevo a BX el primer carácter de la cadena
mov cl, offset fin - offset msj     ; muevo a CL la longitud de la cadena
lazo: ; (SI DESEO IMPRIMIR DESDE TECLADO, ACÁ PONGO EL INT 6)
mov al, [bx]      ; mueve el primer carácter de mensaje a AL
call imprimir_caracter
inc bx
dec cl
jnz lazo
int 0
end
```

HANDSHAKE

Impresora

- Puerto Dato del HANDSHAKE al puerto de datos de 8 bits de la impresora (salida) en la dirección **40H**.
- Puerto Estado del HANDSHAKE en la dirección **41H** con **OUT ESTADO 7FH**:
  - El bit 0 a la señal busy de la impresora (entrada)
  - El bit 1 a la señal strobe de la impresora (salida)

Pasos para imprimir una cadena con el HANDSHAKE:

Es más simple que con el PIO. Al no necesitar del STROBE, las únicas subrutinas a crear son **poll** e **imprimir_cadena_handshake**. Esta última es análoga al **imprimir_cadena** del PIO, solo que reemplazaremos lo que le incumbe al STROBE por **out DATO, AL**.

Subrutinas a crear:

1. **POLL:**

```
in AL, ESTADO
AND AL, 1
JNZ POLL
RET
```

2. **IMPRIMIR_CARACTER_HANDSHAKE:**

```
push AX
CALL POLL
pop AX
OUT DATO, AL
RET
```

Programa principal:

```
ORG 2000H          ; Prog principal
in al, ESTADO
and al, 7Fh        ; modo consulta de estado (01111111)
out ESTADO, al
mov bx, offset msj
mov cl, offset fin - offset msj
LAZO: mov al, [bx]
call imprimir_caracter_hand
inc bx
dec cl
jnz lazo
int 0
end
```

Práctica 3: Interrupciones por Hardware.

Dirección	Registro	Función	Valor de Ejemplo
20h	EOI	**Fin de Interrupción.** Solo para escribir, se debe mandar el comando de fin de interrupción cuando se terminó de servir la interrupción. (Valor 20h)	No aplica
21h	IMR	**Registro de Mascara de Interrupciones.** Permite enmascarar selectivamente las interrupciones que va a recibir el PIC. Cada bit de este registro representa una línea de interrupción (bit 0 se asocia a la entrada INT0 ... bit 7 se asocia a la entrada INT7). Si el valor es puesto en 1, esa interrupción estará inhibida.	1111 1101 Esta habilitada la interrupción INT1 (bit 1 en 0). Esa interrupción está conectada al Timer.
22h	IRR	**Registro de Petición de Interrupciones.** Almacena las interrupciones pendientes. Cada bit de este registro representa una línea de interrupción (bit 0 se asocia a la entrada INT0 ... bit 7 se asocia a la entrada INT7). Cuando la interrupción se satisface, el bit de dicha línea se pone a cero.	0000 0011 Se recibió y están pendientes de ser atendidas las interrupciones INT0 e INT1. Esa interrupción está conectada a la tecla F-10 y al Timer.

23h	ISR	**Registro de Interrupción en Servicio.** El bit que representa la línea de interrupción que se está atendiendo estará en 1 interrupción (bit 0 se asocia a la entrada INT0 ... bit 7 se asocia a la entrada INT7), el resto en 0. Si no hay interrupciones atendiéndose todos los bits valen 0.	0000 0100 La CPU está ejecutando el manejador de interrupción INT2. Esa interrupción está conectada al *HandShake*.
24h	INT0	**ID de Línea INT0.** Almacena el ID de la interrupción asociada al dispositivo *F10* para buscar en el vector de interrupciones la dirección de comienzo de la subrutina que lo atiende.	15 Es el número de vector que se le enviará a la CPU para manejar la interrupción. Si multiplico por 4 (15×4=60) nos da la dirección de memoria donde debe estar la dirección de rutina que maneja la interrupción correspondiente a *F10*.
25h	INT1	**ID de Línea INT1.** Almacena el ID de la interrupción asociada al dispositivo *Timer* para buscar en el vector de interrupciones la dirección de comienzo de la subrutina que lo atiende.	10 Es el número de vector que se le enviará a la CPU para manejar la interrupción. Si multiplico por 4 (10×4=40) nos da la dirección de memoria donde debe estar la dirección de rutina que maneja la interrupción correspondiente al *Timer*.
26h	INT2	**ID de Línea INT2.** Almacena el ID de la interrupción asociada al dispositivo *Handshake* para buscar en el vector de interrupciones la dirección de comienzo de la subrutina que lo atiende.	25 Es el número de vector que se le enviará a la CPU para manejar la interrupción. Si multiplico por 4 (25×4=100) nos da la dirección de memoria donde debe estar la dirección de rutina que maneja la interrupción correspondiente al *Handshake*.

- Los registros internos del PIC se sitúan a partir de la dirección 20H.
- Son accedidos con operaciones lectura y escritura en el espacio de E/S (IN y OUT).
- Interrupciones hardware asignadas:
  - INT0 – tecla F10
  - INT1 – Timer

- INT2 – Handshake
- INT3 – DMA
- INT4 a INT7 no usadas