Práctica 3: Interrupciones por Hardware

Parte 1: PIC y Vector de Interrupciones

Ejercicio 1

Dirección	Registro	Función	Valor de Ejemplo
20h	**EOI**	**Fin de Interrupción**. Solo para escribir, se debe enviar el comando de fin de interrupción (valor 20h) cuando se terminó de servir o atender la interrupción actualmente atendida.	**No aplica.** El único valor válido para escribir en este registro es el 20h. `PIC EQU 20H` `REG_EOI EQU PIC` `NUM_EOI EQU REG_EOI` `MOV AL, NUM_EOI` `OUT REG_EOI, AL`
21h	**IMR**	**Registro de Máscara 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 deshabilitada o enmascarada, mientras que, si se le asigna un cero, se encontrará habilitada o desenmascarada.	**11111101B** Permite habilitar o desenmascarar la línea de interrupción INT1 (bit 1 en 0), la cual está conectada al dispositivo “Timer”. `PIC EQU 20H` `REG_IMR EQU PIC + 1` `MASK_INT1 EQU 11111101B` `MOV AL, MASK_INT1` `OUT REG_IMR, AL`
22h	**IRR**	**Registro de Petición de Interrupciones.** Almacena las interrupciones pendientes: cualquier línea de interrupción que posea una solicitud sin atender tendrá su correspondiente bit en este registro con un valor igual a uno. Cada bit de este registro representa una línea de interrupción distinta (bit 0 se asocia a la entrada INT0... bit 7 se asocia a la entrada INT7). Cuando la interrupción comience a servirse, el bit de dicha línea será nuevamente igual a cero.	**00000011B** Se recibieron y están pendientes de ser atendidas dos interrupciones en total: una en la línea INT1 y otra en la línea INT0, las cuales están conectadas a los dispositivos “Timer” y “Tecla F10” respectivamente. Ningún código Assembly es implementable aquí en forma directa porque el programador no puede modificar manualmente el valor de este registro.
23h	**ISR**	**Registro de Interrupción en Servicio.** El bit que representa la línea de interrupción que se está sirviendo o atendiendo actualmente se encontrará en uno, y los demás, en cero (bit 0 se asocia a la entrada INT0... bit 7 se asocia a la entrada INT7). Si no hay interrupciones atendiéndose, todos los bits serán iguales a cero.	**00000100B** La CPU está ejecutando el manejador de interrupción correspondiente a la línea INT2. En otras palabras, se encuentra atendiendo o sirviendo una solicitud de interrupción previamente realizada a través de esta línea por el dispositivo de hardware conectado a ella: el Handshake.

			Ningún código Assembly es implementable aquí en forma directa porque el programador no puede modificar manualmente el valor de este registro.
24h	**INT0**	**ID de Línea INT0.** Almacena el ID de la línea de interrupción INT0, conectada al dispositivo de hardware "Tecla 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 INT0, es decir, atender las solicitudes de interrupción realizadas por la Tecla F10. Si se multiplica este valor por cuatro (15*4 = 60), se obtiene la dirección de memoria donde debe encontrarse almacenada la dirección de inicio de la subrutina que gestiona la interrupción correspondiente a la Tecla F10. `PIC EQU 20H` `REG_INT0 EQU PIC + 4` `ID_INT0 EQU 15` `ORG 60` `DIR_INT0 DW RUT_INT0` `ORG 3000H` `RUT_INT0: ...` `ORG 2000H` `...` `MOV AL, ID_INT0` `OUT REG_INT0, AL` `...`
25h	**INT1**	**ID de Línea INT1.** Almacena el ID de la línea de interrupción INT1, conectada al dispositivo de hardware "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 INT1, es decir, atender las solicitudes de interrupción realizadas por el Timer. Si se multiplica este valor por cuatro (10*4 = 40), se obtiene la dirección de memoria donde debe encontrarse almacenada la dirección de inicio de la subrutina que gestiona la interrupción correspondiente al Timer. `PIC EQU 20H` `REG_INT1 EQU PIC + 5` `ID_INT1 EQU 10` `ORG 40` `DIR_INT1 DW RUT_INT1` `ORG 3000H` `RUT_INT1: ...`

			`` ORG 2000H ... MOV AL, ID_INT1 OUT REG_INT1, AL ... ```
26h	**INT2**	**ID de Línea INT2.** Almacena el ID de la línea de interrupción INT2, conectada al dispositivo de hardware "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 INT2, es decir, atender las solicitudes de interrupción realizadas por el Handshake. Si se multiplica este valor por cuatro (25*4 = 100), se obtiene la dirección de memoria donde debe encontrarse almacenada la dirección de inicio de la subrutina que gestiona la interrupción correspondiente al Handshake. ``` PIC EQU 20H REG_INT2 EQU PIC + 6 ID_INT2 EQU 25 ORG 100 DIR_INT2 DW RUT_INT2 ORG 3000H RUT_INT2: ... ORG 2000H ... MOV AL, ID_INT2 OUT REG_INT2, AL ... ```

Ejercicio 2

Dispositivos	IMR	INT0	INT1	INT2	Dirección Vector
Tecla F10	**1111 1110**	**5**	**No importa**	**No importa**	**20**
Tecla F10	1111 1110	8	No importa	No importa	**32**
Timer	**1111 1101**	No importa	10	No importa	**40**
Handshake	1111 1011	No importa	No importa	**10**	40
Ninguno / Sin interrupciones	**1111 1111**	No importa	No importa	No importa	Ninguna
Tecla F10 y Timer	1111 1100	**10**	**20**	No importa	40 y 80
Timer y Handshake	**1111 1001**	No importa	4	8	**16 y 32**
Tecla F10, Timer y Handshake	**1111 1000**	**5**	**10**	**15**	20, 40 y 60

a)
Dispositivo = Tecla F10: conectado a la línea de interrupción INT0 del PIC.
Dirección Vector = 32: dirección de memoria número 32 correspondiente al ID del vector de interrupciones seleccionado para la Tecla F10.
IMR = 11111110B: para que se atiendan los pedidos de interrupción emitidos por la Tecla F10, debe encontrarse habilitada o desenmascarada la línea de interrupción INT0.
INT0 = 8: si la dirección correspondiente al ID del vector de interrupciones asociado a la Tecla F10 es igual a 32, entonces para obtenerse dicho ID deberá dividirse el valor de la dirección por cuatro (32 / 4 = 8). Asimismo, el ID calculado habrá de almacenarse en el registro INT0 del PIC ya que ésta es la línea conectada a la Tecla F10.
INT1 = No importa: la línea de interrupción INT1 no se encuentra conectada a la Tecla F10, sino al dispositivo Timer. Por lo tanto, el valor almacenado en el registro INT1 del PIC resulta irrelevante en este escenario.
INT2 = No importa: la línea de interrupción INT2 no se encuentra conectada a la Tecla F10, sino al dispositivo Handshake. Por lo tanto, el valor almacenado en el registro INT2 del PIC resulta irrelevante en este escenario.

b)
IMR = 11111101B: línea de interrupción INT1 desenmascarada.
Dirección Vector = 40: dirección de memoria número 40 correspondiente al ID del vector de interrupciones seleccionado para la línea de interrupción INT1.
Dispositivo = Timer: conectado a la línea de interrupción INT1 del PIC.
INT0 = No importa: la línea de interrupción INT0 se encuentra enmascarada. Por lo tanto, el valor almacenado en el registro INT0 del PIC resulta irrelevante en este escenario.
INT1 = 10: si la dirección de memoria correspondiente al ID del vector de interrupciones asociado a la línea de interrupción INT1 es igual a 40, entonces para obtenerse dicho ID deberá dividirse el valor de la dirección por cuatro (40 / 4 = 10). Asimismo, el ID calculado habrá de almacenarse en el registro INT0 del PIC ya que ésta es la línea conectada a la Tecla F10.
INT2 = No importa: la línea de interrupción INT2 se encuentra enmascarada. Por lo tanto, el valor almacenado en el registro INT2 del PIC resulta irrelevante en este escenario.

c)
Dispositivo = Handshake: conectado a la línea de interrupción INT2 del PIC.
INT2 = 10: éste es el valor del ID del vector de interrupciones seleccionado para la línea de interrupción INT2, conectada al Handshake.
IMR = 11111011B: para que se atiendan los pedidos de interrupción emitidos por el Handshake, debe encontrarse habilitada o desenmascarada la línea de interrupción INT2.
INT0 = No importa: la línea de interrupción INT0 no se encuentra conectada al Handshake, sino a la Tecla F10. Por lo tanto, el valor almacenado en el registro INT0 del PIC resulta irrelevante en este escenario.
INT1 = No importa: la línea de interrupción INT1 no se encuentra conectada al Handshake, sino al Timer. Por lo tanto, el valor almacenado en el registro INT1 del PIC resulta irrelevante en este escenario.

Dirección Vector = 40: si el ID del vector de interrupciones asociado a la línea de interrupción INT2, conectada al Handshake, es igual a 10, entonces para obtenerse la dirección de memoria correspondiente a dicho ID deberá multiplicarse el valor de este último por cuatro (10 * 4 = 40).

d)
IMR = 11111111B: todas las líneas de interrupción del PIC se encuentran enmascaradas o deshabilitadas. Por lo tanto, no se atenderá pedido de interrupción alguno enviado por ninguno de los dispositivos de hardware conectados al PIC: Tecla F10, Timer o Handshake. En consecuencia, todos los demás valores para completar en la tabla propuesta en la consigna correspondientes a este escenario resultarán irrelevantes o nulos según corresponda.
Dispositivo = Ninguno / Sin interrupciones
INT0 = No importa
INT1 = No importa
INT2 = No importa
Dirección Vector = Ninguna

Parte 2: Interrupciones con la tecla F10, Timer y Handshake

Ejercicio 3

```
EOI         EQU 20H
IMR         EQU 21H
INT0        EQU 24H     ; 1° sentencia faltante
                        ; 24H: Puerto registro INT0 del PIC
N_F10       EQU 15

            ORG 60      ; 2° sentencia faltante
                        ; 15*4 = 60: número de dirección del ID (15)
                        ; del vector de interrupciones para la línea de
                        ; interrupción INT0 (Tecla F10) del PIC
IP_F10      DW RUT_F10

            ORG 2000H
            CLI
            MOV AL, 0FEH    ; 0FEH = 11111110B
            OUT IMR, AL     ; 3° sentencia faltante
                            ; desenmascarar la línea INT0 (Tecla F10)
                            ; y enmascarar todas las demás
            MOV AL, N_F10   ; 4° sentencia faltante
            OUT INT0, AL    ; configurar y definir el ID del vector
                            ; de interrupciones para la línea INT0
                            ; (Tecla F10)
            MOV DX, 0
            STI
LAZO:       JMP LAZO

            ORG 3000H
RUT_F10:    PUSH AX
```

```
INC DX          ; 5° sentencia faltante
                ; incrementar en una unidad el número de
                ; veces que se ha presionado la Tecla F10
MOV AL, EOI
OUT EOI, AL     ; 6° sentencia faltante
                ; indicar que se ha terminado de atender
                ; la solicitud de interrupción de la
                ; Tecla F10 actualmente aquí servida
POP AX
IRET
END
```

a)

Puede comprobarse en el repertorio de instrucciones soportadas por el simulador VonSim que la instrucción CLI deshabilita todas las interrupciones enmascarables, mientras que la instrucción STI las vuelve a habilitar. Las interrupciones enmascarables son aquéllas gestionadas por el PIC y, por lo tanto, solicitadas por los dispositivos conectados a sus líneas: Tecla F10, Timer y Handshake.

Si estas dos instrucciones no se encontraran incluidas en el programa durante la configuración del PIC, la CPU podría recibir un pedido de interrupción de un dispositivo conectado a una línea cuyo ID correspondiente al vector de interrupciones aún no esté configurado en su respectivo registro INT del PIC (INT0, INT1, INT2, etc.). Por ejemplo, en este caso, si la persona que ejecuta el programa presionara la tecla F10 rápidamente, la CPU podría buscar, basándose en el valor actual del registro INT0, la subrutina manejadora de las interrupciones solicitadas a través de la línea INT0, conectada a dicha tecla, en la dirección de memoria equivocada y termine ejecutando código inadecuado, erróneo y potencialmente peligroso para este escenario.

b)

El valor 0FEH o 11111110B es utilizado en el programa para configurar el registro IMR del PIC. Al asignarle un cero únicamente al bit 0 del registro y forzar los demás a uno, se habilita o desenmascara exclusivamente la línea de interrupción INT0. De este modo, solamente se atenderán durante la ejecución de este programa las solicitudes de interrupción enviadas por la Tecla F10, conectada a dicha línea.

c)

La CPU debe enviar al PIC el comando de fin de interrupción para indicarle que efectivamente terminó de atender la solicitud de interrupción actualmente servida. Este envío implica esencialmente la acción de escribir el valor 20H en el registro EOI del PIC ejecutando las siguientes instrucciones del programa:

```
MOV AL, EOI
OUT EOI, AL
```

d)

La dirección de comienzo de la subrutina que atiende las solicitudes de interrupción emitidas por la Tecla F10 a través de la línea INT0 del PIC se puede encontrar definida y almacenada en

la dirección correspondiente al ID del vector de interrupciones de la línea INT0. Al tratarse dicho ID del valor 15 (éste es el número cargado en el registro INT0 del PIC), su dirección asociada en el vector será la número 60. Se observa en el programa que allí se halla la etiqueta "RUT_F10", la cual representa simbólicamente la dirección de inicio en la memoria de instrucciones de la subrutina buscada, denominada, tal como esta etiqueta lo indica, "RUT_F10" y encargada de gestionar las interrupciones de la línea INT0: 3000H.

Ejercicio 4

b)

```
PIC         EQU 20H
REG_EOI     EQU PIC
VAL_EOI     EQU REG_EOI      ; se ha terminado de atender la solicitud
                             ; de interrupción actualmente servida
REG_IMR     EQU PIC + 1
INT0_IMR    EQU 11111110B    ; desenmascarar la línea INT0 (Tecla F10)
                             ; del PIC y enmascarar todas las demás
FIN_IMR     EQU 11111111B    ; enmascarar todas las líneas del PIC
REG_INT0    EQU PIC + 4
ID_INT0     EQU 10           ; ID del vector de interrupciones para la
                             ; línea INT0
N_REPET     EQU 5            ; cantidad de veces que se repetirá el
                             ; mensaje
FIN_REPET   EQU 0            ; se han terminado de imprimir todas las
                             ; repeticiones del mensaje

            ORG 40           ; 10*4 = 40: dirección del vector de
                             ; interrupciones para el ID de la línea
                             ; INT0
DIR_INT0    DW RUT_INT0      ; subrutina para gestionar las
                             ; interrupciones de la línea INT0

            ORG 1000H
MENSAJE     DB "Vamos las interrupciones!", 0AH
REST_REPET  DB N_REPET       ; cantidad restante de repeticiones del
                             ; mensaje

            ORG 3000H
INI_DISP:   PUSH AX
            MOV AL, INT0_IMR
            OUT REG_IMR, AL
            MOV AL, ID_INT0
            OUT REG_INT0, AL
            POP AX
            RET

            ORG 3200H
RUT_INT0:   PUSH AX
            INT 7
            DEC REST_REPET
            JNZ FIN_INT0
INHIBIR:    MOV AL, FIN_IMR
            OUT REG_IMR, AL
FIN_INT0:   MOV AL, VAL_EOI
```

```
            OUT REG_EOI, AL
            POP AX
            IRET

            ORG 2000H
            CLI
            CALL INI_DISP
            MOV BX, OFFSET MENSAJE
            MOV AL, OFFSET REST_REPET - OFFSET MENSAJE
            STI
BUCLE:      CMP REST_REPET, FIN_REPET
            JNZ BUCLE
            INT 0
            END
```

Ejercicio 5

a)

```
TIMER       EQU 10H
REG_CONT    EQU TIMER       ; 10H → Puerto registro CONT del Timer
INI_CONT    EQU 0
REG_COMP    EQU TIMER + 1   ; 11H → Puerto registro COMP del Timer
VAL_COMP    EQU 2           ; pedir interrupción cada dos segundos
PIC         EQU 20H
REG_EOI     EQU PIC         ; 20H → Puerto registro EOI
VAL_EOI     EQU REG_EOI
REG_IMR     EQU PIC + 1     ; 21H → Puerto registro IMR
INT1_IMR    EQU 11111101B   ; desenmascarar línea INT1 (Timer)
FIN_IMR     EQU 11111111B
REG_INT1    EQU PIC + 5     ; 25H → Puerto registro INT1
ID_INT1     EQU 10

            ORG 40
DIR_INT1    DW RUT_INT1

            ORG 1000H
MENSAJE     DB "Vamos las interrupciones!", 0AH
FIN_MSJ     DB ?

            ORG 3000H
INI_DISP:   PUSH AX
            MOV AL, INT1_IMR
            OUT REG_IMR, AL
            MOV AL, ID_INT1
            OUT REG_INT1, AL
            MOV AL, VAL_COMP
            OUT REG_COMP, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
            POP AX
            RET

            ORG 3200H
RUT_INT1:   PUSH AX
            INT 7
```

```
FIN_INT1:  MOV AL, VAL_EOI
           OUT REG_EOI, AL
           MOV AL, INI_CONT
           OUT REG_CONT, AL
           POP AX
           IRET

           ORG 2000H
           CLI
           CALL INI_DISP
           MOV BX, OFFSET MENSAJE
           MOV AL, OFFSET FIN_MSJ - OFFSET MENSAJE
           STI
BUCLE:     JMP BUCLE
           INT 0
           END
```

b)

```
TIMER      EQU 10H
REG_CONT   EQU TIMER       ; 10H → Puerto registro CONT del Timer
INI_CONT   EQU 0
REG_COMP   EQU TIMER + 1   ; 11H → Puerto registro COMP del Timer
VAL_COMP   EQU 2           ; pedir interrupción cada dos segundos
PIC        EQU 20H
REG_EOI    EQU PIC         ; 20H → Puerto registro EOI
VAL_EOI    EQU REG_EOI
REG_IMR    EQU PIC + 1     ; 21H → Puerto registro IMR
INT1_IMR   EQU 11111101B   ; desenmascarar línea INT1 (Timer)
FIN_IMR    EQU 11111111B
REG_INT1   EQU PIC + 5     ; 25H → Puerto registro INT1
ID_INT1    EQU 10
T_INI      EQU 0           ; tiempo inicial del programa
T_FIN      EQU 10          ; tiempo de ejecución total del programa

           ORG 40
DIR_INT1   DW RUT_INT1

           ORG 1000H
MENSAJE    DB "Vamos las interrupciones!", 0AH
FIN_MSJ    DB ?

           ORG 3000H
INI_DISP:  PUSH AX
           MOV AL, INT1_IMR
           OUT REG_IMR, AL
           MOV AL, ID_INT1
           OUT REG_INT1, AL
           MOV AL, VAL_COMP
           OUT REG_COMP, AL
           MOV AL, INI_CONT
           OUT REG_CONT, AL
           POP AX
           RET
```

```
            ORG 3200H
RUT_INT1:   PUSH AX
            INT 7
            ADD DL, VAL_COMP
            CMP DL, T_FIN
            JNZ FIN_INT1
INHIBIR:    MOV AL, FIN_IMR
            OUT REG_IMR, AL
FIN_INT1:   MOV AL, VAL_EOI
            OUT REG_EOI, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
            POP AX
            IRET

            ORG 2000H
            CLI
            CALL INI_DISP
            MOV BX, OFFSET MENSAJE
            MOV AL, OFFSET FIN_MSJ - OFFSET MENSAJE
            MOV DL, T_INI
            STI
BUCLE:      CMP DL, T_FIN
            JNZ BUCLE
            INT 0
            END
```

c)

```
TIMER       EQU 10H
REG_CONT    EQU TIMER       ; 10H → Puerto registro CONT del Timer
INI_CONT    EQU 0
REG_COMP    EQU TIMER + 1   ; 11H → Puerto registro COMP del Timer
VAL_COMP    EQU 10          ; pedir interrupción cada diez segundos
PIC         EQU 20H
REG_EOI     EQU PIC         ; 20H → Puerto registro EOI
VAL_EOI     EQU REG_EOI
REG_IMR     EQU PIC + 1     ; 21H → Puerto registro IMR
INT1_IMR    EQU 11111101B   ; desenmascarar línea INT1 (Timer)
FIN_IMR     EQU 11111111B
REG_INT1    EQU PIC + 5     ; 25H → Puerto registro INT1
ID_INT1     EQU 10
T_INI       EQU 0           ; tiempo inicial del programa
T_FIN       EQU 10          ; tiempo de ejecución total del programa

            ORG 40
DIR_INT1    DW RUT_INT1

            ORG 1000H
MENSAJE     DB "Vamos las interrupciones!", 0AH
FIN_MSJ     DB ?

            ORG 3000H
INI_DISP:   PUSH AX
            MOV AL, INT1_IMR
```

```
            OUT REG_IMR, AL
            MOV AL, ID_INT1
            OUT REG_INT1, AL
            MOV AL, VAL_COMP
            OUT REG_COMP, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
            POP AX
            RET

            ORG 3200H
RUT_INT1:   PUSH AX
            INT 7
            MOV DL, T_FIN
INHIBIR:    MOV AL, FIN_IMR
            OUT REG_IMR, AL
FIN_INT1:   MOV AL, VAL_EOI
            OUT REG_EOI, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
            POP AX
            IRET

            ORG 2000H
            CLI
            CALL INI_DISP
            MOV BX, OFFSET MENSAJE
            MOV AL, OFFSET FIN_MSJ - OFFSET MENSAJE
            MOV DL, T_INI
            STI
BUCLE:      CMP DL, T_FIN
            JNZ BUCLE
            INT 0
            END
```

Ejercicio 6

a)

```
TIMER       EQU 10H          ; dispositivo Timer
REG_CONT    EQU TIMER
INI_CONT    EQU 0
REG_COMP    EQU TIMER + 1
VAL_COMP    EQU 1
PIC         EQU 20H          ; dispositivo PIC
REG_EOI     EQU PIC
VAL_EOI     EQU REG_EOI
REG_IMR     EQU PIC + 1
INT0_IMR    EQU 11111110B
INT1_IMR    EQU 11111101B
FIN_IMR     EQU 11111111B
REG_INT0    EQU PIC + 4
ID_INT0     EQU 10
REG_INT1    EQU PIC + 5
ID_INT1     EQU ID_INT0 + 1
VAL_FIN     EQU '0'          ; valor final del contador regresivo
```

```
            ORG 40
DIR_INT0    DW RUT_INT0

            ORG 44
DIR_INT1    DW RUT_INT1

            ORG 1000H
MENSAJE     DB "Por favor, ingrese el valor inicial del contador "
            DB "regresivo:", 0AH
VAL_ACT     DB ?, 0AH  ; valor actual del contador regresivo (no se
                       ; valida el carácter ingresado por el usuario)
FIN_ACT     DB ?

            ORG 3000H
INI_DISP:   PUSH AX
            MOV AL, ID_INT0
            OUT REG_INT0, AL
            MOV AL, ID_INT1
            OUT REG_INT1, AL
            MOV AL, INT0_IMR
            OUT REG_IMR, AL
            POP AX
            RET

            ORG 3200H
USUARIO:    PUSH BX
            PUSH AX
            INT 7
            MOV BX, CX
            INT 6
            POP AX
            POP BX
            RET

            ORG 3400H
RUT_INT0:   PUSH AX
HAB_INT1:   MOV AL, INT1_IMR
            OUT REG_IMR, AL
FIN_INT0:   MOV AL, VAL_EOI
            OUT REG_EOI, AL
            MOV AL, VAL_COMP
            OUT REG_COMP, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
            POP AX
            IRET

            ORG 3600H
RUT_INT1:   PUSH AX
            INT 7
            DEC BYTE PTR [BX]
            CMP BYTE PTR [BX], VAL_FIN
            JNS FIN_INT1
INHIBIR:    MOV AL, FIN_IMR
            OUT REG_IMR, AL
```

```
FIN_INT1:  MOV AL, VAL_EOI
           OUT REG_EOI, AL
           MOV AL, INI_CONT
           OUT REG_CONT, AL
           POP AX
           IRET

           ORG 2000H
           MOV BX, OFFSET MENSAJE
           MOV AL, OFFSET VAL_ACT - OFFSET MENSAJE
           MOV CX, OFFSET VAL_ACT
           CALL USUARIO
           CLI
           CALL INI_DISP
           MOV BX, CX
           MOV AL, OFFSET FIN_ACT - OFFSET VAL_ACT
           STI
BUCLE:     CMP BYTE PTR [BX], VAL_FIN
           JNS BUCLE
           INT 0
           END
```

b)

```
TIMER      EQU 10H          ; dispositivo Timer
REG_CONT   EQU TIMER
INI_CONT   EQU 0
REG_COMP   EQU TIMER + 1
VAL_COMP   EQU 1
PIC        EQU 20H          ; dispositivo PIC
REG_EOI    EQU PIC
VAL_EOI    EQU REG_EOI
REG_IMR    EQU PIC + 1
INT0_IMR   EQU 11111110B
INT01_IMR  EQU 11111100B
REG_INT0   EQU PIC + 4
ID_INT0    EQU 10
REG_INT1   EQU PIC + 5
ID_INT1    EQU ID_INT0 + 1
VAL_INI    EQU '9'          ; valor inicial del contador regresivo
VAL_FIN    EQU '0'          ; valor final del contador regresivo
EN_EJEC    EQU 1            ; contador en ejecución
EN_PAUSA   EQU 0            ; contador en pausa

           ORG 40
DIR_INT0   DW RUT_INT0

           ORG 44
DIR_INT1   DW RUT_INT1

           ORG 1000H
VAL_ACT    DB VAL_INI, 0AH ; valor actual del contador regresivo
FIN_ACT    DB ?

           ORG 3000H
INI_DISP:  PUSH AX
```

```
            MOV AL, ID_INT0
            OUT REG_INT0, AL
            MOV AL, ID_INT1
            OUT REG_INT1, AL
            MOV AL, INT01_IMR
            OUT REG_IMR, AL
            MOV AL, VAL_COMP
            OUT REG_COMP, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
            POP AX
            RET

            ORG 3200H
RUT_INT0:   PUSH AX
            CMP DL, EN_EJEC
            JZ PAUSAR
REANUDAR:   MOV DL, EN_EJEC
            MOV AL, INT01_IMR
            OUT REG_IMR, AL
            MOV AL, VAL_EOI
            OUT REG_EOI, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
            JMP FIN_INT0
PAUSAR:     MOV DL, EN_PAUSA
            MOV AL, INT0_IMR
            OUT REG_IMR, AL
            MOV AL, VAL_EOI
            OUT REG_EOI, AL
FIN_INT0:   POP AX
            IRET

            ORG 3400H
RUT_INT1:   PUSH AX
            INT 7
            DEC BYTE PTR [BX]
            CMP BYTE PTR [BX], VAL_FIN
            JNS CONTINUAR
REINICIAR:  MOV BYTE PTR [BX], VAL_INI
            MOV DL, EN_PAUSA
            MOV AL, INT0_IMR
            OUT REG_IMR, AL
            MOV AL, VAL_EOI
            OUT REG_EOI, AL
            JMP FIN_INT1
CONTINUAR:  MOV AL, VAL_EOI
            OUT REG_EOI, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
FIN_INT1:   POP AX
            IRET

            ORG 2000H
            CLI
```

```
            CALL INI_DISP
            MOV BX, OFFSET VAL_ACT
            MOV AL, OFFSET FIN_ACT - OFFSET VAL_ACT
            MOV DL, EN_EJEC
            STI
BUCLE:      JMP BUCLE
            INT 0
            END
```

Ejercicio 7

b)

Si, como se propone en el primer caso, la velocidad de la CPU (parámetro "velocidad de simulación" del simulador del VonSim) es superior a aquélla correspondiente a la impresora (parámetro "velocidad de impresión" del simulador del VonSim), entonces sería recomendable configurar a esta última en modo interrupción. De esta manera, la CPU dispondrá de un intervalo considerable para realizar múltiples tareas útiles rápidamente en lugar de permanecer ociosa.

Por el contrario, si la velocidad de la impresora es superior a aquélla correspondiente a la CPU, tal como se plantea en el segundo escenario, entonces sería recomendable configurar dicha impresora en modo consulta de estado. La razón para establecer esta afirmación yace en que la complejidad añadida al programa por las instrucciones y declaraciones requeridas para gestionar las solicitudes de interrupción enviadas por la impresora no se justifica si el tiempo de espera y ocio de la CPU es relativamente reducido.

c)

```
HAND        EQU 40H          ; dispositivo Handshake
REG_DATO    EQU HAND
REG_EST     EQU HAND + 1
INT_EST     EQU 10000000B    ; bit 7 (I = Interrupción) en uno:
                             ; impresora en modo interrupción
POLL_EST    EQU 01111111B    ; bit 7 en cero: impresora en modo consulta
                             ; de estado (sin interrupciones)
PIC         EQU 20H          ; dispositivo PIC
REG_EOI     EQU PIC
VAL_EOI     EQU REG_EOI
REG_IMR     EQU PIC + 1
INT2_IMR    EQU 11111011B
FIN_IMR     EQU 11111111B
REG_INT2    EQU PIC + 6
ID_INT2     EQU 10
LONG_CAD    EQU 10           ; longitud de la cadena a imprimir
FIN_CAD     EQU 0            ; se han impreso todos los caracteres

            ORG 40
DIR_INT2    DW RUT_INT2

            ORG 1000H
INGRESO     DB "Por favor, ingrese una cadena de diez caracteres:", 0AH
CADENA      DB ?

            ORG 3000H
```

```
USUARIO:    PUSH AX
            PUSH BX
            PUSH DX
            INT 7
            MOV BX, CX
LECTURA:    INT 6
            INC BX
            DEC DL
            JNZ LECTURA
            POP DX
            POP BX
            POP AX
            RET

            ORG 3200H
INI_DISP:   PUSH AX
            MOV AL, ID_INT2
            OUT REG_INT2, AL
            MOV AL, INT2_IMR
            OUT REG_IMR, AL
            IN AL, REG_EST
            OR AL, INT_EST
            OUT REG_EST, AL
            POP AX
            RET

            ORG 3400H
RUT_INT2:   PUSH AX
            MOV AL, [BX]
            OUT REG_DATO, AL
            INC BX
            DEC DL
            JNZ FIN_INT2
INHIBIR:    IN AL, REG_EST
            AND AL, POLL_EST
            OUT REG_EST, AL
            MOV AL, FIN_IMR
            OUT REG_IMR, AL
FIN_INT2:   MOV AL, VAL_EOI
            OUT REG_EOI, AL
            POP AX
            IRET

            ORG 2000H
            MOV BX, OFFSET INGRESO
            MOV AL, OFFSET CADENA - OFFSET INGRESO
            MOV CX, OFFSET CADENA
            MOV DL, LONG_CAD
            CALL USUARIO
            CLI
            CALL INI_DISP
            MOV BX, CX
            STI
BUCLE:      CMP DL, FIN_CAD
            JNZ BUCLE
```

```
            INT 0
            END
```

d)

```
HAND        EQU 40H          ; dispositivo Handshake
REG_DATO    EQU HAND
REG_EST     EQU HAND + 1
INT_EST     EQU 10000000B    ; bit 7 (I = Interrupción) en uno:
                             ; impresora en modo interrupción
POLL_EST    EQU 01111111B    ; bit 7 en cero: impresora en modo consulta
                             ; de estado (sin interrupciones)
PIC         EQU 20H          ; dispositivo PIC
REG_EOI     EQU PIC
VAL_EOI     EQU REG_EOI
REG_IMR     EQU PIC + 1
INT02_IMR   EQU 11111010B
FIN_IMR     EQU 11111111B
REG_INT0    EQU PIC + 4
ID_INT0     EQU 10
REG_INT2    EQU PIC + 6
ID_INT2     EQU 11
IMP_CURSO   EQU 1            ; impresión actualmente en curso
IMP_FIN     EQU 0            ; impresión finalizada: se canceló
                             ; prematuramente o se imprimieron todos los
                             ; caracteres

            ORG 40
DIR_INT0    DW RUT_INT0

            ORG 44
DIR_INT2    DW RUT_INT2

            ORG 1000H
CADENA      DB "UNIVERSIDAD NACIONAL DE LA PLATA"
FIN_CAD     DB ?

            ORG 3000H
INI_DISP:   PUSH AX
            MOV AL, ID_INT0
            OUT REG_INT0, AL
            MOV AL, ID_INT2
            OUT REG_INT2, AL
            MOV AL, INT02_IMR
            OUT REG_IMR, AL
            IN AL, REG_EST
            OR AL, INT_EST
            OUT REG_EST, AL
            POP AX
            RET

            ORG 3200H
INHIBIR:    PUSH AX
            MOV DL, IMP_FIN
            IN AL, REG_EST
            AND AL, POLL_EST
```

```
            OUT REG_EST, AL
            MOV AL, FIN_IMR
            OUT REG_IMR, AL
            POP AX
            RET

            ORG 3400H
RUT_INT0:   PUSH AX
            CALL INHIBIR
FIN_INT0:   MOV AL, VAL_EOI
            OUT REG_EOI, AL
            POP AX
            IRET

            ORG 3600H
RUT_INT2:   PUSH AX
            MOV AL, [BX]
            OUT REG_DATO, AL
            INC BX
            DEC CL
            JNZ FIN_INT2
            CALL INHIBIR
FIN_INT2:   MOV AL, VAL_EOI
            OUT REG_EOI, AL
            POP AX
            IRET

            ORG 2000H
            CLI
            CALL INI_DISP
            MOV BX, OFFSET CADENA
            MOV CL, OFFSET FIN_CAD - OFFSET CADENA
            MOV DL, IMP_CURSO
            STI
BUCLE:      CMP DL, IMP_FIN
            JNZ BUCLE
            INT 0
            END
```

Parte 3: Ejercicios de repaso para el parcial

Ejercicio 4

```
PIC         EQU 20H          ; dispositivo PIC
REG_EOI     EQU PIC
VAL_EOI     EQU REG_EOI
REG_IMR     EQU PIC + 1
INT0_IMR    EQU 11111110B
FIN_IMR     EQU 11111111B
REG_INT0    EQU PIC + 4
ID_INT0     EQU 10
PIO         EQU 30H          ; dispositivo PIO (impresora)
REG_PA      EQU PIO
REG_PB      EQU PIO + 1
```

```
REG_CA      EQU PIO + 2
REG_CB      EQU PIO + 3
VAL_CA      EQU 00000001B
VAL_CB      EQU 00000000B
ES_BUSY     EQU 00000001B
STROBE_0    EQU 11111101B
STROBE_1    EQU 00000010B
NO_F10      EQU 0           ; aún no se ha presionado la tecla F10
SI_F10      EQU 1           ; la tecla F10 ha sido presionada
LONG_CAD    EQU 10          ; longitud de la cadena a imprimir
FIN_CAD     EQU 0           ; se han impreso todos los caracteres

            ORG 40
DIR_INT0    DW RUT_INT0

            ORG 1000H
INGRESO     DB "Por favor, presione la tecla F10 para comenzar:", 0AH
CADENA      DB ?

            ORG 3000H
INI_PIO:    PUSH AX
            MOV AL, VAL_CB
            OUT REG_CB, AL
            MOV AL, VAL_CA
            OUT REG_CA, AL
            IN AL, REG_PA
            AND AL, STROBE_0
            OUT REG_PA, AL
            POP AX
            RET

            ORG 3200H
INI_PIC:    PUSH AX
            MOV AL, ID_INT0
            OUT REG_INT0, AL
            MOV AL, INT0_IMR
            OUT REG_IMR, AL
            POP AX
            RET

            ORG 3400H
RUT_INT0:   PUSH AX
            MOV DL, SI_F10
INHIBIR:    MOV AL, FIN_IMR
            OUT REG_IMR, AL
FIN_INT0:   MOV AL, VAL_EOI
            OUT REG_EOI, AL
            POP AX
            IRET

            ORG 3600H
LEER_CAD:   PUSH BX
            PUSH DX
LEER_CAR:   INT 6
            INC BX
```

```
            DEC DL
            JNZ LEER_CAR
            POP DX
            POP BX
            RET

            ORG 3800H
POLL:       PUSH AX
BUCLE_P:    IN AL, REG_PA
            AND AL, ES_BUSY
            JNZ BUCLE_P
            POP AX
            RET

            ORG 4000H
IMP_CAR:    PUSH AX
            MOV AL, [BX]
            OUT REG_PB, AL
            IN AL, REG_PA
            OR AL, STROBE_1
            OUT REG_PA, AL
            IN AL, REG_PA
            AND AL, STROBE_0
            OUT REG_PA, AL
            POP AX
            RET

            ORG 4200H
IMP_CAD:    PUSH AX
            MOV BX, SP
            ADD BX, 4
            MOV DX, [BX]
            ADD BX, 2
            MOV BX, [BX]
BUCLE_IMP:  CALL POLL
            CALL IMP_CAR
            INC BX
            DEC DL
            JNZ BUCLE_IMP
            POP AX
            RET

            ORG 2000H
            MOV BX, OFFSET INGRESO
            MOV AL, OFFSET CADENA - OFFSET INGRESO
            INT 7
            CLI
            CALL INI_PIC
            MOV DL, NO_F10
            STI
BUCLE:      CMP DL, SI_F10
            JNZ BUCLE
            MOV BX, OFFSET CADENA
            MOV DL, LONG_CAD
            CALL LEER_CAD
```

```
            CALL INI_PIO
            PUSH BX
            PUSH DX
            CALL IMP_CAD
            POP DX
            POP BX
            INT 0
            END
```

Ejercicio 5

```
TIMER       EQU 10H          ; dispositivo Timer
REG_CONT    EQU TIMER
INI_CONT    EQU 0
REG_COMP    EQU TIMER + 1
VAL_COMP    EQU 12
PIC         EQU 20H          ; dispositivo PIC
REG_EOI     EQU PIC
VAL_EOI     EQU REG_EOI
REG_IMR     EQU PIC + 1
INT1_IMR    EQU 11111101B
FIN_IMR     EQU 11111111B
REG_INT1    EQU PIC + 5
ID_INT1     EQU 10
PIO         EQU 30H          ; dispositivo PIO (llaves y luces)
REG_PA      EQU PIO
REG_PB      EQU PIO + 1
REG_CA      EQU PIO + 2
REG_CB      EQU PIO + 3
VAL_CA      EQU 11111111B
VAL_CB      EQU 00000000B
LUZ_7_ON    EQU 10000000B
LUCES_ON    EQU 11111111B
LUCES_OFF   EQU 00000000B
ESPERAR     EQU 0            ; esperar hasta que se apaguen las luces
FINALIZAR   EQU 1            ; finalizar el programa: ya se apagaron las
                             ; luces o no debió hacerse ninguna acción

            ORG 40
DIR_INT1    DW RUT_INT1

            ORG 1000H
LLAVES      DB ?             ; estado actual de las llaves

            ORG 3000H
INI_PIO:    PUSH AX
            MOV AL, VAL_CA
            OUT REG_CA, AL
            MOV AL, VAL_CB
            OUT REG_CB, AL
            MOV AL, LUCES_OFF
            OUT REG_PB, AL
            IN AL, REG_PA
            MOV [BX], AL
            POP AX
            RET
```

```
            ORG 3200H
LUCES_12:   PUSH AX
            MOV DL, FINALIZAR
            MOV BX, SP
            ADD BX, 4
            MOV BX, [BX]
            MOV AL, [BX]
            AND AL, LUZ_7_ON
            JZ FIN_L_12
            CALL INI_PIC
            MOV DL, ESPERAR
            MOV AL, LUCES_ON
            OUT REG_PB, AL
FIN_L_12:   POP AX
            RET

            ORG 3400H
INI_PIC:    PUSH AX
            MOV AL, ID_INT1
            OUT REG_INT1, AL
            MOV AL, INT1_IMR
            OUT REG_IMR, AL
            MOV AL, VAL_COMP
            OUT REG_COMP, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
            POP AX
            RET

            ORG 3600H
RUT_INT1:   PUSH AX
            MOV DL, FINALIZAR
            MOV AL, LUCES_OFF
            OUT REG_PB, AL
            MOV AL, FIN_IMR
            OUT REG_IMR, AL
            MOV AL, VAL_EOI
            OUT REG_EOI, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
FIN_INT1:   POP AX
            IRET

            ORG 2000H
            MOV BX, OFFSET LLAVES
            CALL INI_PIO
            PUSH BX
            CLI
            CALL LUCES_12
            POP BX
            STI
BUCLE:      CMP DL, FINALIZAR
            JNZ BUCLE
            INT 0
            END
```

Ejercicio 7

```
TIMER       EQU 10H          ; dispositivo Timer
REG_CONT    EQU TIMER
INI_CONT    EQU 0
REG_COMP    EQU TIMER + 1
VAL_COMP    EQU 5
PIC         EQU 20H          ; dispositivo PIC
REG_EOI     EQU PIC
VAL_EOI     EQU REG_EOI
REG_IMR     EQU PIC + 1
INT01_IMR   EQU 11111100B
FIN_IMR     EQU 11111111B
REG_INT0    EQU PIC + 4
ID_INT0     EQU 10
REG_INT1    EQU PIC + 5
ID_INT1     EQU ID_INT0 + 1
PIO         EQU 30H          ; dispositivo PIO (nuevo dispositivo)
REG_PB      EQU PIO + 1
REG_CB      EQU PIO + 3
VAL_CB      EQU 00000000B
FIN_ENVIO   EQU 0            ; código ASCII para indicar el fin del
                             ; envío
ESPERAR     EQU 0            ; esperar mientras se envía la cadena
FINALIZAR   EQU 1            ; finalizar el programa: ya se envió la
                             ; cadena o el envío fue cancelado

            ORG 40
DIR_INT0    DW RUT_INT0

            ORG 44
DIR_INT1    DW RUT_INT1

            ORG 1000H
CADENA      DB "Hola!", FIN_ENVIO
FIN_CAD     DB ?

            ORG 3000H
INI_PIO:    PUSH AX
            MOV AL, VAL_CB
            OUT REG_CB, AL
            POP AX
            RET

            ORG 3200H
INI_PIC:    PUSH AX
            MOV AL, ID_INT0
            OUT REG_INT0, AL
            MOV AL, ID_INT1
            OUT REG_INT1, AL
            MOV AL, INT01_IMR
            OUT REG_IMR, AL
            MOV AL, VAL_COMP
            OUT REG_COMP, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
```

```
            POP AX
            RET

            ORG 3400H
RUT_INT0:   PUSH AX
CANCELADO:  MOV DL, FINALIZAR
            MOV AL, FIN_IMR
            OUT REG_IMR, AL
FIN_INT0:   MOV AL, VAL_EOI
            OUT REG_EOI, AL
            POP AX
            IRET

            ORG 3600H
RUT_INT1:   PUSH AX
            MOV AL, [BX]
            OUT REG_PB, AL
            INC BX
            DEC CL
            JNZ FIN_INT1
COMPLETO:   MOV DL, FINALIZAR
            MOV AL, FIN_IMR
            OUT REG_IMR, AL
FIN_INT1:   MOV AL, VAL_EOI
            OUT REG_EOI, AL
            MOV AL, INI_CONT
            OUT REG_CONT, AL
            POP AX
            IRET

            ORG 2000H
            CALL INI_PIO
            CLI
            CALL INI_PIC
            MOV BX, OFFSET CADENA
            MOV CL, OFFSET FIN_CAD - OFFSET CADENA
            MOV DL, ESPERAR
            STI
BUCLE:      CMP DL, FINALIZAR
            JNZ BUCLE
            INT 0
            END
```