Práctico 9 – Entrada/Salida programada e Interrupciones

Objetivo

Comprender los elementos conceptuales en el manejo de E/S mediante los mecanismos de *polling* e interrupciones.

Notas

- Al referirse a un bit por un número que indica su posición (e.g., el bit 0), se asume que el 0 es el menos significativo.
- Para manejar E/S en alto nivel se deben utilizar las funciones `dato=in(dir)` y `out(dir, dato)`.
- Para habilitar y deshabilitar interrupciones en alto nivel se pueden invocar las rutinas enable() y `disable()`, respectivamente.

Preguntas teóricas:

(a) Indique los valores visibles en los buses de direcciones, datos y control al realizar una operación de entrada salida, distinguiendo entre los casos de E/S aislada y E/S mapeada a memoria.

(b) Explique el propósito del controlador de interrupciones.

(c) Explique el mecanismo de `INT/INTA` para manejar solicitudes de interrupciones y la conexión *Daisy Chain* utilizada ocasionalmente en conjunto con este mecanismo. ¿Existe alguna prioridad entre los dispositivos de E/S al utilizar esta conexión?

(d) ¿Qué significa que una maquina sea dedicada y qué significa que no lo sea? ¿Qué consecuencias tiene el tipo de maquina (dedicada/no dedicada) sobre la solución de un determinado problema?

Ejercicio 1 ⋆ (en OpenFing)

Se considera el teclado de una terminal de caracteres controlado por un microprocesador. El dispositivo es accesible mediante dos registros de un byte `ESTADO_TECLADO` y `DATO_TECLADO`, accesibles en direcciones de E/S de solo lectura. El hardware coloca el valor 1 en el bit más significativo de `ESTADO_TECLADO` cuando hay un carácter disponible. Cuando se lee el carácter disponible en `DATO_TECLADO`, el hardware coloca el valor 0 en el bit más significativo de `ESTADO_TECLADO`.

Escribir un procedimiento que lea los caracteres y los almacene en una lista de largo variable.

Ejercicio 2 ⋆

Se desea controlar una bomba de agua mediante un microprocesador. La bomba se enciende colocando el valor 1 en el bit 2 del puerto de E/S `BOMBA`, y se apaga colocando el valor 0 en el mismo bit. Dado que el encendido no es inmediato, existe un sensor externo que coloca el valor 1 en el bit 0 del puerto `BOMBA` cuando se activa efectivamente la bomba y el valor 0 cuando la bomba está desactivada. Cuando la bomba se activa se debe activar un LED conectado al bit 4 del puerto `BOMBA`, en otro caso se debe desactivar el LED.

Escribir los procedimientos `activarBomba()` y `desactivarBomba()`, sabiendo que el puerto de E/S BOMBA es de lectura y escritura.

Ejercicio 3 ⋆ ⋆ ⋆

Se dispone de tres rutinas utilitarias `RUTINA1`, `RUTINA2` y `RUTINA3`. Un programa en ejecución es interrumpido en cualquier momento para ejecutar estas rutinas por acción del teclado. La ejecución comienza con la secuencia `<ESC><1`, `<ESC><2>` y `<ESC><3>` respectivamente.

`<ESC>` es un carácter especial, conocido. Las rutinas no terminan en ningún momento y la forma de volver al programa original, es presionando la tecla `<ESC>`. Cuando se ejecuta un programa X, presionando `<ESC><2>` se pasa a ejecutar RUTINA 2. Al presionar `<ESC>` nuevamente, se vuelve a ejecutar el programa X en la dirección donde fue interrumpido.

Escribir en un lenguaje de alto nivel un procedimiento que atienda la interrupción del teclado, lea el carácter contenido en el puerto de solo lectura de 1 byte `TECLA` y en caso de que la sucesión de caracteres recibida sea válida pase a ejecutar la rutina correspondiente. La interrupción debe retornar al programa adecuado luego de procesar una tecla.

Notas:

- Para obtener la dirección de una rutina se dispone de la función

```
rutina* radr (string nombre_rutina)
```

- Para pasar el control a una dirección determinada se usa la función

```
void jmp(rutina* dirección)
```

- Para obtener la dirección del programa interrumpido se usa la función

```
rutina* padr()
```

Ejercicio 4 ★★

Un procesador posee tres dispositivos conectados a una misma línea de interrupción. El procesador atiende por nivel y los dispositivos interrumpen también por nivel. Existen tres direcciones de E/S: `ESTADO_1`, `ESTADO_2` y `ESTADO_3`. Cada una de estas entradas está asociada a un dispositivo y el hardware coloca el valor 1 en su bit menos significativo cuando el dispositivo correspondiente solicita una interrupción. Para atender al dispositivo n se debe invocar la función `atencionDispositivoN()`.

Escribir una rutina de interrupción en alto nivel capaz de atender a los tres dispositivos, en las siguientes situaciones:

A) Se considera un esquema de prioridades fijas (pri(disp 1) > pri(disp 2) > pri (disp 3)).

B) Se considera esquema de prioridades variables. Existe la función pri(dispositivo) que devuelve el valor de la prioridad del dispositivo.

Ejercicio 5 ★★★

Se desea implementar un concentrador de comunicaciones para recibir datos a través de cuatro líneas de entrada y enviarlos por una única línea de salida. Cada línea de entrada tiene un controlador asociado que se maneja por medio de dos registros de sólo lectura, uno de control (`EST_CONT_n`) y otro de datos (`DATO _n`), de un byte cada uno, ubicados en las siguientes direcciones:

Controlador	Puerto	Dirección	Puerto	Dirección
0	`EST_CONT_0`	0x10	`DATO_0`	0x11
1	`EST_CONT_1`	0x12	`DATO_1`	0x13
2	`EST_CONT_2`	0x14	`DATO_2`	0x15
3	`EST_CONT_3`	0x16	`DATO_3`	0x17

El bit 0 de `EST_CONT_n` en 1 indica que hay un dato disponible en `DATO_n`. El bit se apaga cuando es leído el registro `DATO_n`. Los bits 1 al bit 7 no se usan. El registro `DATO_n` contiene el dato recibido por la línea n. Cuando algún controlador de entrada tiene un dato disponible, genera una interrupción que activa la rutina `intEntrada()`.

La línea de salida tiene su propio controlador con un registro de datos `DATO_S`, de un byte, ubicado en la dirección de E/S 0x20. Toda vez que se escribe un byte en `DATO_S` el controlador transmite el byte por la línea de salida y al terminar la transmisión genera una interrupción que activa la rutina `intSalida()`.

Se debe cumplir que al ingresar un dato (1 byte) por la línea n (0..3) se transmite a la salida el número n (1 byte) y luego el dato ingresado (1 byte).

Escribir en alto nivel las rutinas necesarias para implementar el concentrador.

Observaciones:

El procesador está dedicado exclusivamente a la ejecución de las rutinas que implementan el concentrador.

El concentrador deberá aceptar los bytes de entrada y guardarlos en un buffer en caso de que el canal de salida esté siendo utilizado, hasta un máximo de `MAX_BUFFER` bytes. En caso de que el buffer esté lleno se descartarán los caracteres recibidos.

Ejercicio 6 ⋆

Sea un sistema que controla las luces de iluminación de los pasillos de un edificio. El sistema cuenta con un botón de encendido en cada piso y se desea que una vez transcurrido 45 segundos desde que fue presionado el botón de algún piso, éstas se apaguen.

Se sabe que:

- Cuando se aprieta un botón en cualquier piso, se invoca una rutina de interrupción llamada `boton()`.
- El encendido de todas las luces se realiza colocando el valor en 1 el bit menos significativo del byte de la dirección 0x20 y las luces se apagan colocando el valor 0 en el mismo bit.
- Se cuenta con un timer que genera una interrupción cada un segundo, la cual invoca a la rutina de interrupción `tiempo()`.

Escribir en lenguaje de alto nivel las rutinas necesarias para implementar el controlador de luces.

Ejercicio 7 ⋆ ⋆ ⋆ (en OpenFing)

Se desea controlar el escape de gas en una fábrica de recarga de garrafas. Por este motivo se instala un sistema de seguridad controlado por un procesador utilizado en forma dedicada. El sistema está formado por un sensor de gas, un extractor y una válvula de corte.

El sensor controla la concentración de gas en el ambiente. Es posible conocer si la concentración de gas es peligrosa o no consultando el registro de E/S del sensor.

El extractor y la válvula pueden controlarse manipulando el registro de lectura/escritura asociado a ellos. Una vez cerrada, la válvula sólo podrá abrirse manualmente.

El sistema debe encender el extractor toda vez que se detecte escape de gas (la concentración de gas es peligrosa) y apagarlo cuando el escape culmina. Si el escape persiste por más de 30 segundos se debe cerrar la válvula y apagar el extractor cuando el escape haya finalizado.

El sensor está ubicado en la dirección de E/S `SENSOR`. EI bit 0 tiene el valor 1 si la concentración de gas es peligrosa y el valor 0 cuando no lo es. Los bits del 1 al 7 son reservados.

El extractor y la válvula se controlan mediante la dirección de E/S `ACCESORIO`. El bit 0 se setea a 1 para encender el extractor y a 0 para apagarlo. El bit 1 se setea a 0 para cerrar la válvula. Los bits del 2 al 7 son reservados.

Escribir todas las rutinas necesarias para implementar el sistema de seguridad en un lenguaje de alto nivel. Se dispone de un reloj externo que genera una interrupción cada 10 Hz que es atendida por la rutina `tiempo()`.

Ejercicio 8 ⋆ ⋆ ⋆

Se desea diseñar un dispositivo para controlar una caja de seguridad con combinación para ser instalada en una cadena de hoteles. La caja de seguridad cuenta con un teclado numérico con 10 teclas para los dígitos del 0 al 9, la tecla `ENTER` y un display de 8 dígitos de 7 segmentos.

El funcionamiento del sistema debe ser el siguiente:

- Cuando la caja está abierta el huésped ingresará una clave de entre 4 y 8 dígitos en el teclado y cerrará la puerta de la caja.
- Cuando el huésped presiona la tecla `ENTER` la caja se cerrará mediante barras de metal que se extienden desde la puerta. Si el código ingresado tiene menos de 4 dígitos no se hará nada.
- Luego de cerrada la caja, si se ingresa la misma secuencia usada para cerrarla y luego se presiona la tecla `ENTER` la caja se abrirá retrayendo las barras.
- Si la clave es ingresada en forma equivocada, al presionar la tecla `ENTER` se mostrará una serie de 8 guiones en el display de 7 segmentos.
- Si la clave se ingresa mal tres veces seguidas la caja quedará bloqueada por dos horas como medida de seguridad. Cualquier combinación de teclas que se ingrese provocará un error durante el lapso de dos horas.

El display y la memoria de los dígitos presionados se vaciarán automáticamente si no se presiona ninguna tecla por 5 segundos. Si se presionan más de 8 dígitos sólo se considerarán los últimos 8 ingresados antes de presionar `ENTER`.

Para manejar el teclado se dispone de una interrupción `tecla()` que se invoca cada vez que se presiona una tecla. La tecla presionada puede leerse en el registro de E/S (solo lectura) `TECLADO`, de un byte. Las teclas de dígitos se leerán codificadas en binario en los 4 bits más significativos y la tecla `ENTER` tiene el código 0xD0.

Para activar el motor que maneja las barras se dispone del registro de E/S (solo escritura) `BARRAS`, de un byte. Las barras se extienden escribiendo el valor 1 en el bit más significativo y se retraen escribiendo el valor 1 en el bit menos significativo.

Para manejar el display se dispone de un registro de E/S (solo escritura) `DISPLAY`, de 4 bytes donde se debe escribir el número a mostrar en BCD (los dígitos más significativos del número a mostrar coinciden con los más significativos del registro). El guión a mostrar en caso de error se codifica como OxFF y para no mostrar nada se debe escribir 0xEE. Se dispone además de un timer que interrumpe con una frecuencia de 2 Hz invocando a la rutina `tiempo()`.

Implementar en un lenguaje de alto nivel el programa principal y las rutinas necesarias para que funcione el sistema de control para el dispositivo descripto.