Práctico 10 – Interrupciones y entrada/salida en el microprocesador 8086.

Objetivo

El objetivo de este práctico es familiarizarse con la implementación a bajo nivel de una rutina de atención a interrupciones, así como comprender como las mismas se instalan y configuran en un sistema basado en un procesador Intel 8086.

Notas

- Se deben resolver todos los problemas en lenguaje C (alto nivel) y luego compilarlos en assembler.
- Para manejar E/S en alto nivel deben utilizar las psuedo-instrucciones del tipo `dato=in(DIR E/S)` y `out(DIR E/S, dato)`.
- Para habilitar y deshabilitar interrupciones en alto nivel se pueden invocar las rutinas `enable()` y `disable()` respectivamente.

Preguntas teóricas:

(a) ¿Dónde está ubicado en memoria absoluta el vector de interrupciones del 8086? ¿Es posible modificar esta posición?
(b) ¿Que propósito tiene el vector de interrupciones del 8086?
(c) ¿Qué implica realizar la “instalación de la rutina” en Intel 8086?
(d) ¿De qué forma se pueden recordar valores entre dos ejecuciones de la misma rutina?

Ejercicio 1 ⋆⋆

Compilar en assembler 8086 las siguientes rutinas del práctico 9:

A) Las rutinas `activarBomba()` y `desactivarBomba()` del ejercicio 2.
B) Para el ejercicio 3. si la interrupción del teclado es la 4 y la rutina de atención se ensambla a partir de la dirección de memoria física 0x41024, escriba el código assembler 8086 que instala la interrupción.

Ejercicio 2 ⋆⋆

Se considera un conjunto de mensajes que poseen la estructura **`STX canal texto ETX`** donde **`STX`** (`start of text`) y **`ETX`** (`end of text`) son caracteres especiales, **`canal`** es un byte con valor entre 0..10 y **`texto`** es una sucesión de caracteres diferentes de todos los anteriores. Se considera una cola circular de 1024 caracteres. Al recibir un

carácter, se invoca a la rutina `nuevoChar`, y el byte a leer queda en el puerto de E/S de solo lectura `MENSAJE`. La rutina `nuevoChar` debe manejar el canal, guardando en la cola los caracteres de los mensajes con canal diferente a 0.

A) Escribir en lenguaje C la rutina "`nuevoChar`".

B) Compilar en assembler 8086 sabiendo que la cola circular comienza en la dirección 2 del ES y en la dirección 0 se indica la cantidad de caracteres en la cola.

Nota: La máquina no está dedicada a la tarea.

Ejercicio 3 ★★ (en OpenFing, basado en el ejercicio 7 del práctico 9)

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 8086 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. La válvula una vez cerrada sólo podrá abrirse manualmente.
El sistema debe encender el extractor toda vez que este detecte escape de gas (la concentración de gas es peligrosa) y apagarlo cuando dicho 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.

Se dispone de un reloj externo que genera una interrupción con una frecuencia de 10 Hz la cual es atendida por la rutina `tiempo()`.

Descripción de los registros de E/S:

Para el sensor:

- El bit 0: vale 1 si la concentración de gas es peligrosa y 0 cuando no lo es.
- Del 1 al 7: reservados.
- Dirección de E/S: `SENSOR`.

Para el extractor y la válvula:

- El bit 0: puede ser seteado a 1 para enceder el extractor o a 0 para apagarlo.
- El bit 1: puede ser seteado a 0 para cerrar la válvula.
- Del 2 al 7: reservados.
- Dirección de E/S: `ACCESORIO`.

A) Escribir todas las rutinas necesarias para implementar el sistema de seguridad en un lenguaje de alto nivel.

B) Compilar en 8086.

C) Si la interrupción del reloj es la 9 y la rutina tiempo se ensambla a partir de la dirección de memoria física 0x3340A, escriba el código assembler que instala la interrupción.

Ejercicio 4 ⋆ ⋆ ⋆

Tiro al blanco es un sencillo juego controlado por un procesador 8086. Consiste en 8 lámparas que se iluminan en secuencia a intervalos de 0.1 segundos. Cada lámpara tiene asociado un botón y el objetivo del jugador es presionar el botón mientras la lámpara correspondiente está encendida. Si esto se logra, la lámpara queda encendida. El juego termina al lograr dejar encendidas las 8 lámparas o si pasan 900 segundos.

La lámpara enésima (0..7) se enciende poniendo en 1 el bit enésimo del puerto de E/S `0x11` (de lectura / escritura).
Se dispone de un timer que genera una interrupción cada 50 ms y que es atendido por la rutina `timer()`.

Al presionar un interruptor se genera una interrupción que es atendida por la rutina `interruptor()` y en el byte del puerto de E/S `0x10` (de sólo lectura) se escribe un dígito entre 0 y 7 correspondiente al último interruptor presionado.
Para iniciar el juego se debe presionar cualquier botón. Una vez terminado, queda pronto para volver a iniciar el juego en cualquier momento.

Al finalizar deberá guardarse en la dirección de memoria absoluta 0x340080 el tiempo empleado por el participante en completar su objetivo. Si éste no se logra en un tiempo menor a 900 segundos el juego debe finalizar escribiendo 0xB0B0 en dicha dirección.

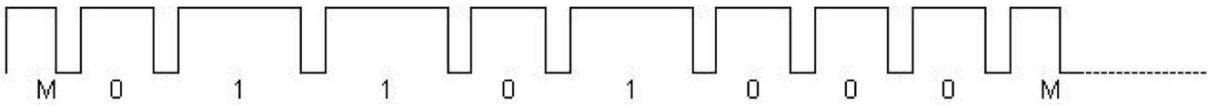
A) Implementar en un lenguaje de alto nivel el juego descripto.

B) Compilarlo en assembler 8086.

C) Las interrupciones de `timer` y `interruptor` son la 13 y 14 respectivamente. Asumiendo que la primera está ensamblada a partir de la dirección 0x55900 y la segunda a partir de la dirección 0x5600C, complete el vector de interrupciones de forma tal que el CS guardado tenga el mismo valor para ambas posiciones.

Ejercicio 5 ★★★

Se desea construir un dispositivo que decodifique una transmisión digital binaria codificada en ancho de pulso (*Pulse Width Modulation*, PWM). La técnica PWM consiste en modificar la duración de un pulso (tiempo en que la señal está en 1). En este caso concreto un 0 se codifica como un pulso que dura 30 ms (en estado 1), mientras que un 1 se codifica con un ancho de pulso de 50 ms (en estado 1). Un pulso de ancho menor que 30 ms se considera una "marca" que indica que finaliza un byte y comienza otro (la marca no es un bit y eventualmente puede aparecer antes de completarse 8 bits en cuyo caso se rellena con 0 a la izquierda o luego de más de 8 bits en cuyo caso se toman los 8 últimos recibidos).



El dispositivo recibe el tren de pulsos a través de una entrada binaria que puede ser leída en el bit menos significativo del byte de E/S en la dirección `PULSO`. La aparición de un flanco ascendente invoca a la rutina de atención `flanco()`. Se dispone además de un timer de frecuencia 1 KHz que genera una interrupción que invoca a la rutina de atención `timer()`.
El sistema debe decodificar los bits recibidos serialmente, codificados en el ancho de los pulsos, rearmar y escribir cada byte recibido en el puerto de E/S (solo escritura) en la dirección `SALIDA`.

Se pide:

(a) Implementar en un lenguaje de alto nivel todas las rutinas necesarias para que funcione el decodificador según los requerimientos, teniendo en cuenta que la máquina es dedicada.

(b) Escribir el código 8086 que instala las rutinas *flanco() y timer()*, sabiendo que las mismas son las número 15 y 16 respectivamente, y que se deben instalar ambas en el segmento 0x5500.