INSTITUTO POLITÉCNICO NACIONAL.
ESCUELA SUPERIOR DE CÓMPUTO.

SISTEMAS OPERATIVOS.

PRÁCTICA 7

Programación de dispositivos de entrada y salida.

Integrantes del equipo:

- Chavarría Vázquez Luis Enrique.
- Juárez Espinoza Ulises.
- Machorro Vences Ricardo Alberto.
- Pastrana Torres Víctor Norberto.

2CM6 Unidad 4

Índice de contenido.

Índice de figuras.

Índice de tablas

Glosario de términos

Interrupción de hardware

Las son avisos que el hardware envía al microprocesador de una computadora a través de señales físicos a los circuitos de la misma CPU. Las solicitudes de interrupción están basadas en un sistema de prioridades de modo que el procesador pueda o no ignorar determinadas peticiones. La interrupción de hardware es causada por algún dispositivo de hardware, como una solicitud para iniciar una E / S, una falla de hardware o algo similar. Las interrupciones de hardware se introdujeron como una forma de evitar perder el valioso tiempo del procesador en bucles de sondeo, esperando eventos externos. Acorde con lo encontrado en [1]

Interrupción de software

La interrupción de software se invoca mediante el uso de la instrucción INT. Este evento detiene inmediatamente la ejecución del programa y pasa la ejecución al controlador INT. El controlador INT suele ser parte del sistema operativo y determina la acción a realizar. Ocurre cuando un programa de aplicación finaliza o solicita ciertos servicios del sistema operativo. [2]

Excepciones

Hace referencia a las interrupciones causadas por la propia CPU, cuando ocurre algo no deseado, por ejemplo, una división por cero. [3]

Interrupción enmascarable

Las interrupciones de proceso que pueden retrasarse cuando se ha producido al mismo tiempo una interrupción de mucha prioridad. [3]

Interrupción no enmascarable

Las interrupciones de hardware que no se pueden retrasar y deben ser procesadas por el procesador de inmediato. [3]

Periféricos de entrada

Permite la entrada del usuario, desde el mundo exterior a la computadora. Ejemplo: teclado, mouse, etc. Acorde con lo encontrado en [4]

Periféricos de salida

Permite la salida de información, desde la computadora al mundo exterior. Ejemplo: impresora, monitor, etc. [4]

Periféricos de entrada-salida (mixtos)

Permite tanto la entrada (desde el mundo exterior al ordenador) como la salida (desde el ordenador al mundo exterior). Ejemplo: pantalla táctil, etc. [4]

Sondeo o Polling

En el sondeo, el primer dispositivo que se encuentra con el conjunto de bits de IRQ es el dispositivo al que se debe prestar servicio primero. Se llama al ISR apropiado para dar servicio al mismo. Es fácil de implementar, pero se pierde mucho tiempo interrogando el bit IRQ de todos los dispositivos. [5] [6]

Máscara de interrupción

La máscara de interrupción tiene un conjunto de bits idénticos a los del registro de interrupciones. Establecer cualquier bit (o desenmascarar) permite que la fuente de señal correspondiente genere una interrupción, lo que hace que el procesador ejecute el ISR. [5]

Vector de interrupción de hardware

El vector de interrupción es una ubicación en la memoria que programa con la dirección de su rutina de servicio de interrupción (ISR). Siempre que ocurre una interrupción sin máscara, la ejecución del programa comienza desde la dirección contenida en el vector de interrupción. [7]

Contenido (Investigación)

Interrupciones de hardware

Definición general [8]

Primero que nada, debemos preguntarnos ¿qué significa una interrupción?; para poder resolver esa pregunta primero debemos entender que es un IRQ, el cual es un acrónimo desde las palabras inglesas Interrupt Request, traducidas en castellano como solicitud de interrupción o interrupción de hardware.

Cuando un periférico, (por ejemplo, una impresora) u otro dispositivo hardware como lo pudiera ser una tarjeta de sonido, las cuales necesitan "comunicarse" con la CPU y para ello utilizan una líneas de notificación preestablecidas denominadas Líneas de Interrupción (Interrupt Request Line).

Una Unidad Central de Procesamiento, puede estar ocupada procesando billones de operaciones por segundo, lo que hace una IRQ es avisar de una nueva tarea pendiente de ser examinada. El procesador, una vez ejecutada la tarea solicitada con la IRQ, vuelve a su anterior operación. Las IRQs disponen de canales físicos dedicados en las placas base, cada uno con un nivel de prioridad y conectados a la CPU con pins.

Funcionamiento

¿Para qué sirven las IRQ? Los dispositivos hardware que necesitan ejecutarse transmiten una IRQ al procesador para llamar su atención. La tarjetas de red, de video, de sonido, un módem, los adaptadores SCSI, los dispositivos de tipo IDE/ADE, los periféricos USB, por puerto paralelo o serie, todos disponen de un canal prioritario para comunicarse con la CPU denominado "Número de IRQ". Una interrupción se genera cuando se quiere que la CPU deje de ejecutar el proceso en curso y ejecute una función específica de quien produce la interrupción. [9]

El controlador puede deshabilitar, técnicamente "enmascarar", determinadas solicitudes de interrupción, retrasando su ejecución, no obstante, hay interrupciones que no pueden ser inhibidas ("interrupciones no enmascarables").

Algunas clasificaciones de interrupción (dependiendo de la fuente). [9]

Interrupciones de software

Son las cuales se producen cuando el usuario solicita hacer algún tipo de llamadas al sistema.

Interrupciones de hardware

Hace referencia a las que son causadas cuando un dispositivo hardware requiere la atención de la CPU para que se ejecute su manejador.

Excepciones.

Hace referencia a las interrupciones causadas por la propia CPU, cuando ocurre algo no deseado, por ejemplo, una división por cero.

Clasificación de las interrupciones por hardware (Interna y externa) [10]

Podemos apreciar en la ilustración 1 podemos apreciar un diagrama con la organización de las interrupciones tanto por hardware y software.

Ilustración 1 Clasificación de las interrupciones por hardware.

Una vez visto el diagrama de la ilustración 1 podemos pasar a analizar cada una de las clasificaciones a detalle.

Interrupciones por hardware (internas)

Las interrupciones internas son generadas por ciertos eventos que surgen durante la ejecución de un programa. Este tipo de interrupciones son manejadas en su totalidad por el hardware y no es posible modificarlas.

Interrupciones por hardware (externas)

Las interrupciones externas las generan los dispositivos periféricos, como pueden ser: teclado, impresoras, tarjetas de comunicaciones, etc. También son generadas por los coprocesadores, no es posible desactivar a las interrupciones externas.

- *Externa enmascarada*
  - En estas el procesador puede ignorarla, debido a que con un control de software se pueden por medio del hardware activar o desactivar.
  - Son empleadas con periféricos en general.
  - Si se considera la parte del control del software sobre el hardware, el procesador directamente puede aceptar o simplemente enmascarar las señales de interrupción.

- *Interna sin enmascarar*
  - Se le da prioridad más alta que las enmascarables.
  - Se usa principalmente en eventos que pudieran ser catastróficos para el sistema.
  - El procesador no puede evitar atenderlas.
  - No puede ser deshabilitada por el software bajo ningún termino.

En la ilustración de abajo (ilustración 2) podemos apreciar con claridad algunas de las interrupciones por hardware más comunes, del mismo modo en la tercera columna tenemos explicado a detalle el dispositivo que les corresponde. [11]

Entrada al 8259	INT 80x86	Dispositivo
IRQ 0	8	Chip temporizador del sistema (timer chip)
IRQ 1	9	Teclado
IRQ 2	0Ah	Conexión con el PIC 2(IRQ's 8–15)
IRQ 3	0Bh	Puerto serie 2 (COM2/COM4)
IRQ 4	0Ch	Puerto serie 1 (COM1/COM3)
IRQ 5	0Dh	Puerto paralelo 2 en AT, reservado en sistemas PS/2
IRQ 6	0Eh	Controladora de floppy
IRQ 7	0Fh	Puerto paralelo 1
IRQ 8/0	70h	Reloj de tiempo real
IRQ 9/1	71h	Redirección de la IRQ 2
IRQ 10/2	72h	Reservado
IRQ 11/3	73h	Reservado
IRQ 12/4	74h	Reservado en AT, **Ratón PS/2** en sistemas PS/2
IRQ 13/5	75h	Interrupción de la FPU (Floating Point Unit)
IRQ 14/6	76h	Controladora de disco duro
IRQ 15/7	77h	Reservado

Ilustración 2 Algunas de las interrupciones que podemos encontrar por medio del hardware.

Una vez que hemos analizado en profundidad toda la parte de las interrupciones por hardware procedemos a explicar los distintos tipos de interrupciones por software que podemos encontrar en nuestros sistemas.

Interrupciones por software del sistema operativo

- Interrupciones por software del sistema operativo (BIOS y DOS)

Interrupciones por software del usuario

- Simplemente son aquellas programadas por el usuario, es decir, el usuario decide cuando y donde ejecutarlas, generalmente son usadas para realizar entrada y salida.

Llegados a este punto de la investigación de nuestra práctica 7 podemos dar paso a las interrupciones del procesador y todas las peticiones que existen dentro de la misma clasificación y además entender como es que estas siguen y respetan una prioridad especifica.

Excepciones del procesador [12]

Tomando en cuenta el funcionamiento básico de nuestro procesador pueden ocurrir circunstancias excepcionales; es usual citar como ejemplo el caso de una división por cero. En estos casos, el procesador genera una excepción, que es tratada como si fuese una interrupción software, con la diferencia de que el número de interrupción asociado depende del tipo de excepción.

Ahora bien, esta es la jerarquía que se sigue acorde con su número de posición en que los iremos mencionando a continuación. Primero que nada, irán las excepciones del procesador, posteriormente tendremos la interrupciones software, luego podemos encontrar las interrupciones hardware no

enmascarables y, por último, pero no menos importante encontraremos las interrupciones hardware enmascarables.

Aclaraciones fundamentales sobre las IRQ (Interrupt request) y clasificación [13]

Antes de poder ver como es que las solicitudes de interrupción se clasifican bajo una configuración estandar, debemos analizar algunos de los aspectos básicos sobre ellos.

Cabe destacar que en el cuadro que presentaremos a continuación (tabla 1), las IRQ (Interrupt Request o solicitudes de interrupción) por prioridad, también entraremos a entender mejor el uso, aportaremos una descripción y del mismo podremos visualizar los conflictos.

Tabla 1 Descripción y orden de las Interrupt Request Line.

	Prioridad	Uso	Descripción	Conflictos
IRQ 0	**1**	**Temporizador del sistema**	Petición de interrupción reservada al cronómetro del PC que sincroniza todos los componentes. No disponible para otros periféricos u otros dispositivos.	En el manejo de los IRQ, este interrupt no debería generar conflictos, en caso contrario investigar si hay incidencia de hardware en la placa base.
IRQ 1	**2**	**Controlador del teclado**	No utilizable para otros dispositivos, IRQ exclusiva para el teclado, aunque se trate de sistemas que no lo llevan.	IRQ reservada, en caso de conflicto comprobar el hardware de la tarjeta madre o el controlador del teclado.
IRQ 2	**-**	**Interruptor de cascada para IRQs 8-15**	Conecta las interrupciones IRQ del 8 al 15. No utilizada en la mayoría de sistema, en caso de empleo cambiar los dispositivos en IRQ 9 a otras líneas de interrupción como IRQ 10 o IRQ 11.	Los típicos conflictos en IRQ 2 proceden del uso simultáneo de diferentes dispositivos en IRQ 2 e IRQ 9.
IRQ 3	**11**	**Puerto serie 2 (COM2)**	Interruptor automático para el segundo puerto serie, en algunos casos	Un problema frecuente procede de dispositivos que tratan de emplear

Continuación tabla 1

			también *default* para el serial 4 (COM4).	COM2 y COM4 simultáneamente.
IRQ 4	**12**	**Puerto serie 1 (COM1)**	Interruptor por defecto para el serial primero (COM1) y para COM3. Es habitual el uso de un ratón con salida serial en IRQ 4 en aquellos PC que no emplean el conector PS/2 para mouse.	Fuentes de conflictos recurrentes son un modem que trata de utilizar COM3/IRQ 4 y dispositivos que emplean a la vez COM1 y COM3 en IRQ 4.
IRQ 5	**13**	**Puerto sonido / paralelo 2 (LPT2), COM3, COM4**	En ausencia de un segundo puerto para impresora LPT2, IRQ 5 es asignado primariamente a la tarjeta de sonido o como una alternativa para los puertos seriales COM.	Si se utiliza el 2º puerto paralelo, impresora u otro aparato, se atribuirá *ipso facto* el interrupt request 5. Mejor asignar un IRQ alejado del número 5 a dispositivos (e.g. de red) que acepten interrupts con números elevados.
IRQ 6	**14**	**Controlador de Floppy Disk (disquete)**	Interruptor reservado al controlador de la disquetera (lector disquete).	Los conflictos en IRQ 6 son improbables y pueden derivar del intento de asignar un periférico.
IRQ 7	**15**	**Puerto paralelo 1 (LPT1), COM3, COM4**	Normalmente destinado al uso de impresoras, en caso contrario, y salvo la posibilidad de conflictos, puede ser empleado para todo terminal que use puertos paralelos.	Los conflictos en IRQ 7 son pocos frecuentes. En caso de utilizar dos puertos paralelos, asegúrate de haber asignado el segundo al IRQ 5 o a otro interrupt request disponible.
IRQ 8	**3**	**RTC - Reloj en tiempo real (en CMOS)**	No utilizable por otros dispositivos, IRQ 8 maneja los eventos que necesitan ser parámetros al tiempo real.	Un eventual conflicto en este interrupt puede ser síntoma de un problema de hardware en la tarjeta madre.
IRQ 9	**4**	**Libre**	Disponible para diferentes adaptadores,	Puede generar conflicto con IRQ 2 ya que IRQ 9

Continuación tabla 1

			normalmente empleado para tarjetas de red. En la mayoría de computadoras puede emplearse libremente al no tener asignación de *default*.	aprovecha IRQ 2 para interactuar con la CPU. Es también la razón por la que IRQ 9 posee una prioridad más alta.
IRQ 10	**5**	**Libre**	Disponible para adaptadores y periféricos genéricos, e.g. tarjetas de red, de sonido, adaptador SCSI y PCI, canal IDE secundario.	Los conflictos en IRQ 10 son improbables. Puede precisar de cambio de asignación en BIOS setup si se utiliza una PC card.
IRQ 11	**6**	**Libre**	Disponible para adaptadores y periféricos genéricos.	Eventuales conflictos podrían proceder de tarjetas PCI video.
IRQ 12	**7**	**Ratón con conector PS/2, tarjeta de red, PCI video, IDE terciario**	En caso de no usar un ratón PS/2, puede emplearse IRQ 12 para un adaptador de red.	Si la entrada PS/2-ratón está habilitada en BIOS, y se está usando el ratón vía PS/2, evitar utilizar IRQ 12 para otros dispositivos. Conflictos ocasionales por asignación vía BIOS de PCI video.
IRQ 13	**8**	**Coprocesador matemático (FPU o NPU)**	Interruptor dedicado y no utilizable para ningún otro dispositivo.	No genera conflictos, salvo en caso de eventual incidencia hardware en la placa madre, en el microprocesador o en el coprocesador matemático.
IRQ 14	**9**	**Canal IDE primario**	IRQ 14 puede emplearse para otros dispositivos, por ejemplo, un adaptador SCSI, en aquellas maquinas que no usan dispositivos IDE siempre y que se haya	Muy poco frecuentes, en el caso de utilizar solo dispositivos SCSI y designar interrupt 14, comprobar que todos los controladores IDE estén desactivados.

Continuación tabla 1

			deshabilitado el canal IDE en la BIOS.
IRQ 15	**10**	**Canal IDE Secundario / Libre**	Reservado al controlador IDE secundario, puede utilizarse para tarjetas de red o SCSI previa deshabilitación en el BIOS setup. Un conflicto en IRQ 15 es comúnmente debido al uso de dispositivos no IDE /ATA sin haber desactivado en BIOS.

Cabe destacar que todos los datos que hemos visto estan estrictamente apegados a una configuración de tipo estandar o por default.

Diferencias entre interrupciones por software y hardware [10]

Interrupción de hardware

La interrupción de hardware es causada por algún dispositivo de hardware, como una solicitud para iniciar una E / S, una falla de hardware o algo similar. Las interrupciones de hardware se introdujeron como una forma de evitar perder el valioso tiempo del procesador en bucles de sondeo, esperando eventos externos.

Por ejemplo, cuando se completa una operación de E / S, como leer algunos datos en la computadora desde una unidad de cinta.

Interrupción de software

La interrupción de software se invoca mediante el uso de la instrucción INT. Este evento detiene inmediatamente la ejecución del programa y pasa la ejecución al controlador INT. El controlador INT suele ser parte del sistema operativo y determina la acción a realizar. Ocurre cuando un programa de aplicación finaliza o solicita ciertos servicios del sistema operativo.

A continuación, podemos ver una tabla ejemplo para mostrar diferencias en concreto entre ambos tipos de interrupciones. Esto se puede apreciar de forma directa en la tabla 2. [14]

Tabla 2 Diferencias entre las interrupciones.

Número de la diferencia.	Interrupción por hardware.	Interrupción por software.
1	La interrupción de proceso es una interrupción generada por un dispositivo o hardware externo.	La interrupción de software es la interrupción generada

Continuación tabla 2

		por cualquier sistema interno de la computadora.
2	No incrementa el contador del programa.	Incrementa el contador del programa.
3	La interrupción de hardware se puede invocar con algún dispositivo externo, como una solicitud para iniciar una E / S o la aparición de una falla de hardware.	La interrupción de software se puede invocar con la ayuda de la instrucción INT.
4	Tiene la prioridad más baja que las interrupciones de software}	Tiene la máxima prioridad entre todas las interrupciones.
5	La interrupción de hardware se activa mediante hardware externo y se considera una de las formas de comunicarse con los periféricos externos, el hardware.	La interrupción del software es activada por software y se considera una de las formas de comunicarse con el kernel o de activar llamadas al sistema, especialmente durante el manejo de errores o excepciones.
6	Es un evento asincrónico.	Es un evento sincrónico.
7	Las interrupciones de hardware se pueden clasificar en dos tipos: 1. Interrupción enmascarable. 2. Interrupción no enmascarable.	Las interrupciones de software se pueden clasificar en dos tipos: 1. Interrupciones normales. 2. Excepción.
8	Las depresiones de pulsaciones de teclas y los movimientos del mouse son ejemplos de interrupciones de hardware.	Todas las llamadas al sistema son ejemplos de interrupciones de software.

Sistema de interrupciones básico [13]

Descripción general del sistema básico

En esta imagen (ilustración 3) es una descripción general de un sistema de interrupciones (sesgado ligeramente a micros PIC) que es cierta para cualquier módulo de interrupciones y es útil para comprender cómo controlar y usar las interrupciones.

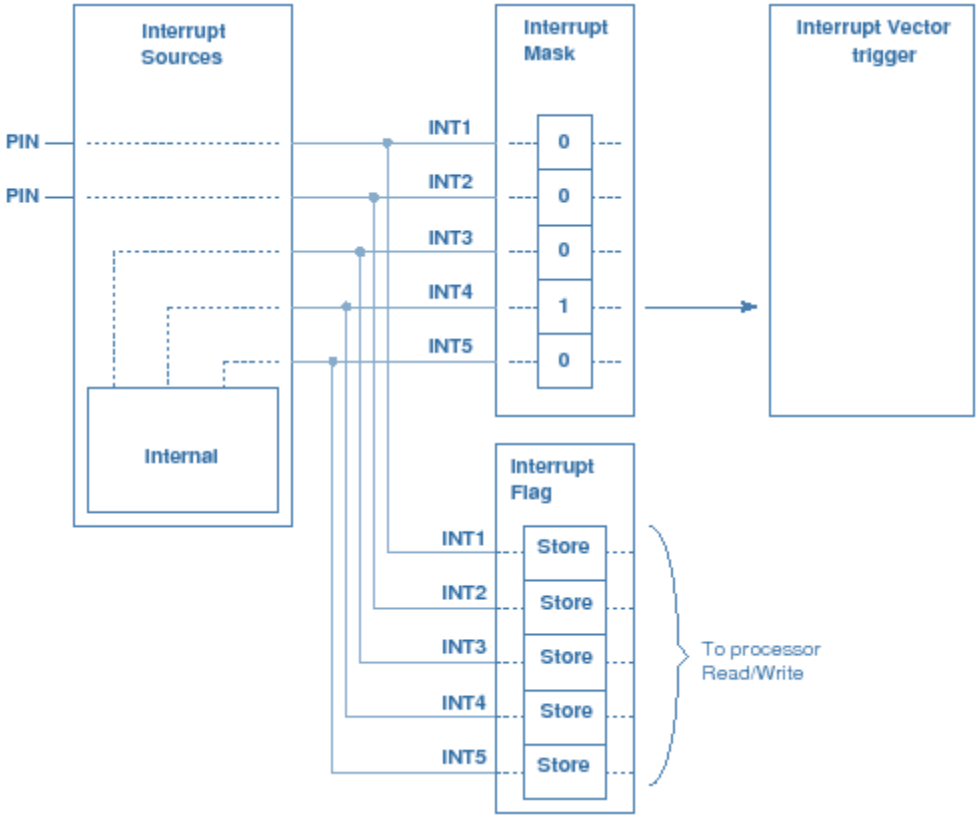

Ilustración 3 Esquema simple de un sistema de interrupciones.

Ahora procederemos a explicar parte por parte y con más detalle nuestro sistema de interrupciones de la ilustración 3 para entenderlo mejor.

Aclaraciones respecto a las entradas del sistema

Estas son señales que inician una interrupción. Pueden ser pines externos (donde un flanco ascendente o descendente de una entrada activa la interrupción) o interrupciones periféricas internas, por ejemplo, ADC completado, datos de recepción en serie recibidos, desbordamiento del temporizador, etc.

Explicación sobre las interrupciones (banderas)

Cada fuente de alarma de proceso tiene un indicador de interrupción asociado; siempre que se activa la alarma, se activa el indicador de interrupción correspondiente. Estos indicadores generalmente se almacenan como bits dentro de un registro de interrupciones.

El procesador puede leer y escribir en el registro de interrupciones, leer de él para averiguar qué interrupciones ocurrieron y escribir en él para borrar los indicadores de interrupción.

Máscara de interrupción

La máscara de interrupción tiene un conjunto de bits idénticos a los del registro de interrupciones. Establecer cualquier bit (o desenmascarar) permite que la fuente de señal correspondiente genere una interrupción, lo que hace que el procesador ejecute el ISR.

Vector de interrupción de hardware

El vector de interrupción es una ubicación en la memoria que programa con la dirección de su rutina de servicio de interrupción (ISR). Siempre que ocurre una interrupción sin máscara, la ejecución del programa comienza desde la dirección contenida en el vector de interrupción.

Prioridad de conexión en cadena [6]

Esta forma de decidir la prioridad de interrupción consiste en la conexión en serie de todos los dispositivos que genera una señal de interrupción. El dispositivo con la prioridad más alta se coloca en la primera posición seguido por los dispositivos de menor prioridad y el dispositivo que tiene la prioridad más baja entre todos se coloca en el último de la cadena.

En el sistema de conexión en cadena, todos los dispositivos están conectados en serie. La solicitud de línea de interrupción es común a todos los dispositivos. Si algún dispositivo tiene una señal de interrupción en estado de nivel bajo, la línea de interrupción pasa al estado de nivel bajo y habilita la entrada de interrupción en la CPU. Cuando no hay interrupción, la línea de interrupción permanece en estado de nivel alto.

La CPU responde a la interrupción habilitando la línea de reconocimiento de interrupciones. Esta señal es recibida por el dispositivo 1 en su entrada PI. La señal de reconocimiento pasa al siguiente dispositivo a través de la salida PO solo si el dispositivo 1 no solicita una interrupción.

La siguiente figura (ilustración 4) muestra el diagrama de bloques para el sistema de prioridad de conexión en cadena.

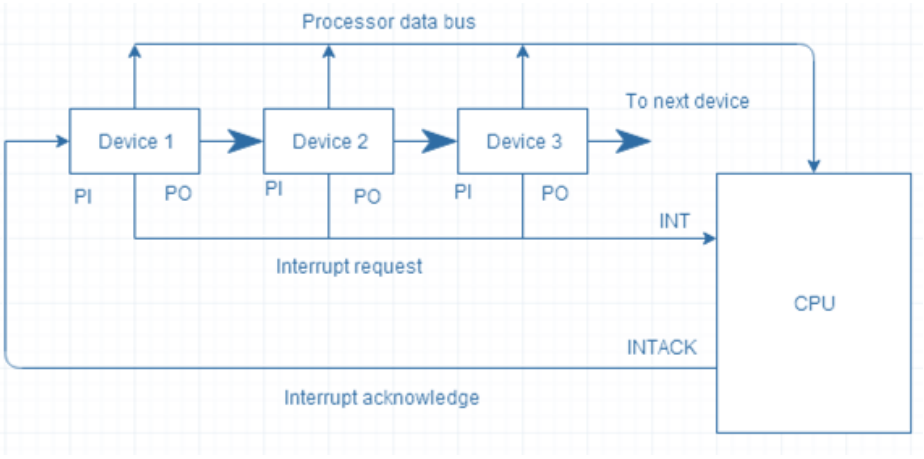

Ilustración 4 Prioridad de conexión en cadena para las interrupciones.

Códigos y ventanas de ejecución

Programa71.c

Realizar un programa donde se aplique el hardware “mouse” de la computadora.

Código completo y explicación en general

Código completo

En la siguiente ilustración (ilustración 5) podemos apreciar el código de nuestra práctica y de manera posterior podemos encontrar la explicación general del código que hemos presentado en esta práctica número 7.

```
#include <stdlib.h>
void main(){
    int x,y,direccion;
     x=0;
     y=0;
    direccion=1;
    int vueltas=0;
    while (1){
        Sleep(50);
        if(direccion==1){
            x++;
            y++;
        }else{
            if(direccion==-1){
                x--;
                y--;
            }
        }

        if(x==100){
           direccion=-1;
            printf("\r X = %i , Y = %i         ", x , y);
            Sleep(500);
        }else{
            if(x==0){
            vueltas++;
               direccion=1;
                printf("\r X = %i , Y = %i          ", x , y);
                Sleep(500);
            }else{
               printf("\r X = %i , Y = %i          ", x , y);
            }
        }
         SetCursorPos(x,y);
         if(vueltas==2){
           break;
         }
    }
}
```

Ilustración 5 Código del Programa71.c

Explicación general del código mostrado

Ahora procedemos a mostrar nuestra explicación general del código que hemos presentado en la imagen superior.

Primero se importan cuatro librerías para poder trabajar de manera correcta el programa en Windows. La primeria librería <stdio.h> es para poder hacer la salida de información. La segunda librería <windows.h> para poder trabajar con la API de Windows. La tercera <winuser.h> para trabajar con los dispositivos de Windows (esta vez el ratón). La última librería <stdlib.h> es la estándar para programas de C.

En la función principal se declara las variables “x” (coordenadas en el eje x), “y” (coordenadas en el eje y) y dirección (dirección de movimiento del puntero del ratón, adelante si es 1 y hacia atrás si es -1). Luego se declara “y” con el valor de cero y también x, paro poner dirección con el valor de uno. La última variable se declarar un variable de vueltas con el número cero para ver cuantas veces se hace el ciclo de ir hacia adelante y atrás en diagonal.

La siguiente parte del código en un ciclo infinito con while donde primero se duerme al programa unos 50 milisegundos ,luego si el valor de la variable dirección es uno se aumenta por uno las variables de ”x” y “y”. Si la variable tuviera el valor de -1 se le decrementa por uno a ambas variables (x,y).

Luego se evalúa si “x” tiene el valor máximo de coordenada, cien que de ser así cambia el valor de dirección a menos uno se imprime el valor de las variables de “x,” y la “y”. Aparte para ver más bien la impresión se duerme el programa 500 milisegundos. Si no se cumple la condición de “x” igual a cien se evalúa con un if si “x” es igual a cero, que en caso de ser así se incrementa con valor uno la variable de vueltas , se asigna el valor de la variable de dirección igual a uno, se imprime el valor de “x” y de la variable “y”. Luego para imprimir mejor la impresión se duerme el programa de 500 milisegundos. Si no se cumple el if se imprime solamente la variable de “x” y la variable “y”.

Como último paso se posiciona el puntero del ratón en las posiciones de las variables “x” y “y”, aparte de que se evalúa si la variable de vueltas es igual a dos, que en caso de ser cumplida sale del ciclo while y termina el programa

Ejecución (Imagen y explicación)

Este programa sencillo tiene la funcionalidad de moverse desde la esquina superior izquierda de la pantalla y moverse en una diagonal hasta llegar a las coordenadas (x=100, y=100) y volver por la misma trayectoria a las esquina superior izquierda. Esta acción la haría dos veces antes de acabar y cuando el puntero del ratón se mueve muestra las coordenadas del ratón mientras se mueve.

En el siguiente link anexamos un video para que usted pueda ver la ejecución de nuestra práctica número 7, https://youtu.be/3viN1kta_cM

En la imagen (ilustración 6) de abajo podemos ver una pequeña captura de la compilación del programa

```
C:\Users\Ricardo Alberto\Documents\Quinto Smeestre\SistemasOperativos>gcc Programa71.c -o Programa71

C:\Users\Ricardo Alberto\Documents\Quinto Smeestre\SistemasOperativos>Programa71
 X = 0 , Y = 0
C:\Users\Ricardo Alberto\Documents\Quinto Smeestre\SistemasOperativos>gcc Programa71.c -o Programa71

C:\Users\Ricardo Alberto\Documents\Quinto Smeestre\SistemasOperativos>Programa71
 X = 0 , Y = 0
C:\Users\Ricardo Alberto\Documents\Quinto Smeestre\SistemasOperativos>gcc Programa71.c -o Programa71

C:\Users\Ricardo Alberto\Documents\Quinto Smeestre\SistemasOperativos>Programa71
 X = 45 , Y = 45
```

Ilustración 6 Ejecución del programa.

Ahora procedemos con la parte de la ejecución de nuestro programa, el cual nos ayuda muchísimo para poder mover el cursos en las coordenadas que hemos definido de forma previa. Podemos apreciar de forma precisa el desplazamiento en la ilustración 7 y la ilustración 8.

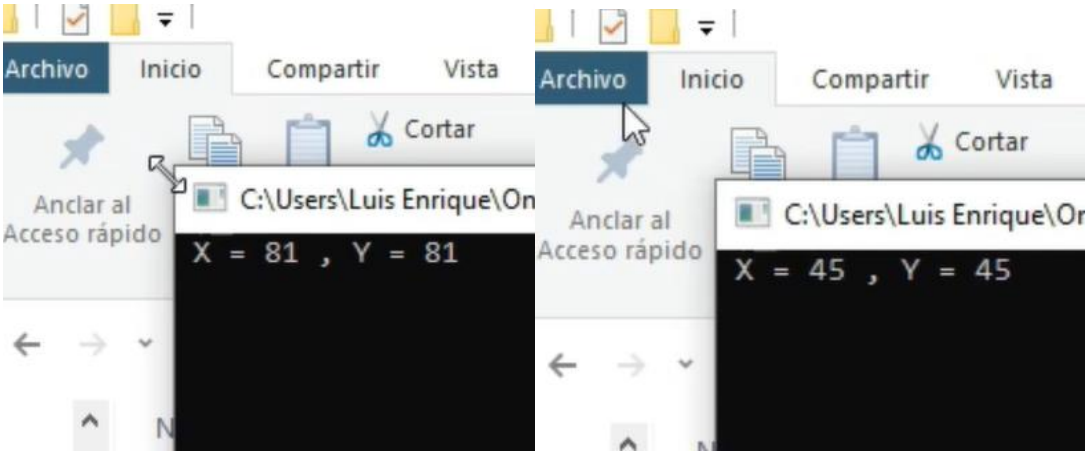

Ilustración 7 Coordenada del cursor en x= 81 y eje y = 81

Ilustración 8 Coordenada del cursor en x =45 y eje y = 45

Conclusiones

Chavarría Vázquez Luis Enrique

Para dar comienzo con mis conclusiones, primero que nada quiero destacar que todo proceso de investigación que hemos realizado durante el desarrollo de la práctica desde mi punto de vista ha sido sumamente satisfactorio y desde luego me atrevo a decir que gratificante, digo esto porque a decir verdad me puedo dar cuenta de la complejidad que tiene este tema referente a los periféricos de entrada y salida, ya no solo por la enorme cantidad de fabricantes que puede haber, sino que del mismo modo todo el desarrollo tecnológico, implementaciones y el diseño mismo de los sistemas operativos para poder lidiar con toda la concurrencia de dispositivos que pudieran haber conectados a una computadora es sin duda algo que solemos dar por hecho que debe estar presente en nuestras computadoras o dispositivos inteligentes, pero que en realidad detrás de todo ello la cantidad de conocimiento y la amalgama de creatividad en ingeniería que hay es ciertamente un poco abrumador al principio, pero conforme fuimos a ahondando en más y más fuentes de información, dichas dudas que iban surgiendo se solucionarán y también muchas ideas que ya tenía sobre los periféricos que interactúan con la computadora pudieran ser clarificadas.

Ahora más que nunca, valoro todo el trabajo que hay detrás de dispositivos tan simples como un ratón, un teclado de computadora o inclusive el mismo panel táctil que dispositivos como mi computadora portátil tienen incluidos de fábrica; pero por otra parte también creo humildemente que en términos generales me siento en mejores condiciones para poder experimentar con implementaciones en el hardware y desarrollar algunas ideas que tengo para poder ofrecer algunas soluciones. Siendo sincero, cuando yo en mis tiempos libres desarrollo algunas aplicaciones, siempre había sido para mí un problema poder desarrollar interacciones con el hardware, pero ahora que me he dado el tiempo de poder investigar más a fondo sobre cómo funcionan los dispositivos de entrada y salida en el sistema operativo me siento con los ánimos reanimados para poder probar de primera mano junto con nuestros conocimientos en implementaciones de circuitos, algunas aplicaciones para poder entender a mayor profundidad que innovaciones se pueden traer la mesa en términos de hardware y no solamente software, lo cual al final del día pienso que ayuda a salir de la zona de confort.

Con lo que respecta al desarrollo del problema, la verdad es que el programa pareció un poco intimidante pero así como lo mencioné anteriormente, poco a poco mientras investigábamos fue más sencillo dar con nuestra solución para poder manipular el cursor del ratón de la computadora; otro punto del que en esta práctica me di cuenta es que en windows por su gran cuota de mercado cuenta con una tremenda cantidad de controladores para el hardware, motivo por el cual también se estuvo trabajando con este sistema operativo al cual ya estamos bastante habituados, a decir verdad es una alegría poder trabajar en windows por su facilidad, pero el lado negativo de esto es que en Linux y todas sus distribuciones en comparación al sistema operativo de Microsoft es un poco menor robusto en este apartado, lo cual desde la forma en que yo lo veo pone en desventaja a los sistemas operativos de la comunidad y da la pauta a que no por elección, a sino por necesidad muchos usuarios se tengan que ver obligados a usar el sistema operativo de Microsoft, inclusive yo puedo decir que viví este problema en carne propia ya que en mi computadora no aprovecha al máximo su tarjeta gráfica dentro del entorno de Linux y en el sistema operativo de Microsoft el aprovechamiento superior debido a que se tiene los controladores de Nvidia mucho más trabajados en el ya mencionado sistema operativo.
Podemos notar que, a veces la documentación suele ser un poco escasa el para el trabajo con ciertos dispositivos, por tanto al inicio de la práctica fue desafiante encontrar la información adecuada para poder poner manos a la obra, para nuestra suerte del trabajo con la biblioteca de windows.h y

winuser.h, volvió fácil el trabajo con dispositivos en este sistema operativo, a pesar de todo estos sigo teniendo un tremendo respeto por todo lo que tiene que ver con programar controladores para dispositivos ya que en general consideró que simplemente impensable la cantidad de variantes que pueden haber acorde a la necesidad que el mercado exija ya grado de especialización del dispositivo, lo cual desde mi punto de vista exige un grado de inversión a nivel de conocimientos técnicos y capital humano bastante considerable.

Juárez Espinoza Ulises

Esta práctica de dispositivos de entrada y salida, que no son más que los medios por los cuales nosotros como usuarios no comunicamos con los sistemas de procesamiento de información o llamado de manera general software, específicamente en el caso de la práctica, los dispositivos de entrada y salida de las computadoras, me hizo reflexionar sobre todo aquello que desconocemos, a diario hacemos uso de estos dispositivos, pero no sabemos porque funcionan y no le damos el valor que tienen.

Son de suma importancia para gozar de las funcionalidades que nos brinda el software de la computadora, más ahora que debido a la contingencia debemos mantenernos en casa y realizar nuestras actividades desde casa, como estudiante a diario hago uso de estos dispositivos llámese cámara web, micrófono, mouse e incluso ahora escribiendo esto gracias al teclado, se aprecia entonces que estos dispositivos están presentes en la mayoría de los dispositivos electrónicos si no es que en todos, en lo que compete a los sistemas operativos cada uno maneja sus dispositivos de entrada y salida de manera diferente.

Para esta práctica se nos pedía que un programa en lenguaje c interactuara con el mouse de la computadora, en un principio todo el equipo pensó que se tenía que hacer en Linux, ya hemos venido trabajando con este sistema sin embargo aún no nos terminamos de adaptar del todo, ya que hemos sido usuarios de Windows toda la vida, cuando nos enteramos que podía ser en Windows nos alegramos aun sin saber que sería prácticamente lo mismo, pero por familiaridad y porque pensamos que existe más información sobre el tema en el sistema operativo Windows que en Linux nos alegramos un poco.

Al ser sinceros nos equivocamos existe poca documentación sobre como trabajar con los dispositivos de entrada salida en ambos sistemas operativos, a mi parecer lo realmente complicado de esta práctica fue documentarnos, sin embargo, leyendo un poco sobre las bibliotecas windows.h que nos permite trabajar con la API del sistema operativo del mismo nombre y winuser.h que nos permite trabajar con los dispositivos de Windows la práctica se hizo de manera sencilla, escasas 60 líneas.

Sin embargo, esto no quiere decir que se fácil para los sistemas operativos trabajar con ellos, ni que los códigos que permiten esta interacción también lo sean, ya que nuestro programa solo mueve en diagonal y en una pequeña área el cursor del mouse y aun asi fue difícil entender la lógica y programar esta funcionalidad aun utilizando un lenguaje de programación casi estándar como c, imaginemos lo que se necesitara para mover el curso por toda la pantalla y de manera libre aún con lenguaje c. En general son las conclusiones que me dejo la práctica, no porque algo funciona significa que sea por si solo y mucho menos que es sencillo

Machorro Vences Ricardo Alberto

En esta práctica se vio lo que son los dispositivos de entrada/salida viendo como es que estos interactúan con el software. Para esto se vio que la forma en la que se interactúa con estos dispositivos varía mucho dependiendo de los sistemas operativos en los que operan ya que como estos difieren mucho en la forma en que tratan los procesos los dispositivos se ven afectados por esto.

Esta diferencia las note mucho por el hecho de que al principio de esta práctica pensé que esta iba a ser en Linux, pero no fue en Windows donde vi que las bibliotecas para poder usar dispositivos físicos tanto en Windows como en Linux no tienen una documentación tan clara para ser usada en cualquier código C y C++ de manera rápida, por lo que para poder usar tan sea una estructura, un método o variable se tienen que investigar muchos más elementos antes de ser empleada y para el caso de Linux se complica mucho por las descargas que se tienen que hacer para usar un programa.

Esta falta de documentación sencilla hiso que se me complicara el realizar esta práctica ya que como había mucha libertad en que tenía que hacer el programa varias ideas que tenía con el programa en cuestión estaban relacionadas con hacer clic, pero no entendía de manera sencilla como se manejaba este al grado de que investigando incluso por tres horas no encontraba algo que me diera una pista de como integrar al programa esa función.

Por suerte en la plataforma de Classroom había videos para poder emplear las librerías de Windows para poder tener una idea de como se puede usar en este caso el hardware del ratón y asi dar una idea de que posibles códigos se podían hacer.

En resumen, esta práctica me dejo ver que a nivel de programación muy bajo es muy difícil trabajar con el hardware en general por los conceptos que se manejan además de que no hay mucha documentación para principiantes que solo necesiten una simple función u elemento en su programa. Esto hace que mi perspectiva se expanda por el hecho de que si en un programa que se supone que es de alto nivel como C esto se complica, debe de ser en extremo difícil en lenguajes que si están diseñados para el hardware como ensamblador.

Pastrana Torres Victor Norberto

Para el desarrollo de esta práctica llevamos a cabo la implementación de la simulación de un clic, el primer vistazo de la practica fue algo agradable porque no lucia como una práctica difícil si la comparamos con la anterior, en este caso la dificultad a la que nos enfrentamos fue a la de encontrar la documentación de la biblioteca Windows.h, esta biblioteca es un archivo cabecera específico de Windows para la programación en lenguaje C y C++ que contiene las declaraciones de todas las funciones de la biblioteca Windows API con las cuales podemos realizar múltiples cosas sin la necesidad de desarrollar más cosas. De esta forma nos damos cuenta la importancia de la documentación de los proyectos, muchas ocasiones cada que nos toca desarrollar un proyecto, odiamos la parte de la documentación porque para nosotros es más cómodo el pasarnos el día frente a la computadora desarrollando, diseñando, implementando algo, pero el hecho de tener que explicar cómo funciona no lo es del todo.

Esta unidad de dispositivos que entrada y salida me pareció muy interesante porque antes no me había preguntado cómo funcionaba, como usuario promedio lo único que hacemos es conectar un cable oprimir unos cuantos botones o dar un par de clics, pero ya que nos adentramos un poco en este tema nos damos cuenta que es algo complejo por que tan solo el hecho de desarrollar un driver para un dispositivo conlleva consigo un gran costo de implementación, el diseño, las pruebas, el desarrollo e incluso la corrección de errores toman un gran tiempo de estudio. Además, el controlador no es el único componente que establece la comunicación entre la computadora y el dispositivo, también se requiere de una pieza física que conocemos como controladora, el desarrollo de esta pieza implica aún más tiempo de estudio y desarrollo.

Los dispositivos son una parte muy importante de las computadoras, con ellos se permite que el alcance de las computadoras aumente, al inicio el dispositivo más común era la pantalla o el teclado, actualmente esto ha avanzado y podemos conectar cosas como cámaras fotográficas, dispositivos móviles o incluso impresoras 3D, considero que este último dispositivo es uno de los avances más sorprendentes porque antes el pensar que a través de una computadora podríamos obtener cosas desde 0, únicamente estaba reservado para películas de ciencia ficción pero ahora es una realidad y día a día vemos que el alcance de este dispositivo puede aumentar, incluso hay estudios que intentar logar la impresión de tejido óseo con los cuales el nivel de las prótesis aumentara a un grado inimaginable. Podemos definir a la computadora como una máquina capaz de procesar información, en primer instancia esta definición podría quedarse un poco corta porque el alcance las computadoras actualmente va más allá, hoy en día las computadoras están en todas partes y con su ayuda se logran muchas cosas, desde los ordenadores que son empleados por los estudiantes para sus clases en línea, hasta computadoras que ejecutan sistemas en tiempo real como el piloto automático de tesla, o las computadoras responsables de trazar trayectorias para cada avión tomando en consideración factores como el clima, más vuelos planeados, el destino, etc. Las computadoras hoy en día son necesarias para muchas cosas.

Bibliografía

[1] study tonight, «study tonight,» study tonight, [En línea]. Available: https://www.studytonight.com/computer-architecture/input-output-organisation. [Último acceso: 12 Diciembre 2020].

[2] geeks for geeks, «geeks for geeks,» geeks for geeks, 2020. [En línea]. Available: https://www.geeksforgeeks.org/difference-between-hardware-interrupt-and-software-interrupt/. [Último acceso: 11 Diciembre 2020].

[3] best-microcontroller-projects, «best-microcontroller-projects,» best-microcontroller-projects, [En línea]. Available: https://www.best-microcontroller-projects.com/hardware-interrupt.html. [Último acceso: 9 Diciembre 2020].

[4] Universidad de valencia, «Universidad de valencia,» Universidad de valencia, [En línea]. Available: https://www.uv.es/varnau/AEC_01.pdf. [Último acceso: Diciembre 2020].

[5] StackExchange, «StackExchange,» StackExchange, 2011. [En línea]. Available: https://unix.stackexchange.com/questions/17998/what-are-software-and-hardware-interrupts-and-how-are-they-processed. [Último acceso: 11 Diciembre 2020].

[6] fing, «fing,» fing, [En línea]. Available: https://www.fing.edu.uy/tecnoinf/mvd/cursos/arqcomp/material/teo/arq-teo08.pdf. [Último acceso: 7 Diciembre 2020].

[7] Quora, «Quora,» 16 Agosto 2016. [En línea]. Available: https://www.quora.com/How-does-a-hardware-interrupt-work-Does-it-use-its-own-bus-or-the-systems-bus-Is-it-the-same-for-software-interrupts. [Último acceso: 6 Diciembre 2020].

[8] researchgate, «researchgate,» researchgate, [En línea]. Available: https://www.researchgate.net/publication/327053482_Application_Software_For_Learning_CPU_Process_of_Interrupt_and_IO_Operation. [Último acceso: Diciembre 2020].

[9] MIT EDU, «MIT EDU,» MIT EDU, [En línea]. Available: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-004-computation-structures-spring-2017/c18/c18s1/. [Último acceso: 12 Diciembre 2020].

[10] Y. Cetina. [En línea]. Available: https://www.slideshare.net/yesyduc10/interrupciones-115449020. [Último acceso: Diciembre 2020].

[11] S. Hal, «Spideri Hal,» Spideri Hal, [En línea]. Available: https://es.slideshare.net/SpiderHal/interrupciones-de-hardware. [Último acceso: Diciembre 2020].

[12] B. Schlich, «Model Checking of Software for Microcontrollers.,» RWTH Aachen University, Aachen, 2008.

[13] G. N. T. S. B. W. C. Herberich, «Proving correctness of an efficient abstraction for interrupt handling.,» ENTCS, Elsevier, Amsterdam , 2008.

[14] arqdecomputadorasatzin, «arqdecomputadorasatzin,» arqdecomputadorasatzin, [En línea]. Available: http://arqdecomputadorasatzin.blogspot.com/2018/03/interrupciones-de-hardware.html. [Último acceso: 15 Diciembre 2020].

Anexos

Video de la práctica número 7. https://youtu.be/3viN1kta_cM