# E/S mediante Interrupciones

*Rafael Ignacio Zurita* *Rodolfo del Castillo*
*rafa@fi.uncoma.edu.ar* *rdc@fi.uncoma.edu.ar*

**Resumen**

En los sistemas que soportan E/S mediante interrupciones los dispositivos notifican un cambio de estado al CPU automaticamente. Las diferentes arquitecturas difieren en la implementación de este mecanismo, pero historicamente, todas comparten algunos conceptos comunes de funcionamiento y configuracion. En este articulo se encuentran explicado estos conceptos y el principio basico de su funcionamiento. Finalmente, se presentan las implementaciones de las interrupciones en arquitecturas modernas.

*This is a sample document to showcase page-based formatting. It contains a chapter from a Wikibook called Sensory Systems. None of the content has been changed in this article, but some content has been removed.*

Note that in this template, you may have a multi-paragraph abstract. However, that it is not necessarily a good practice. Try to keep your abstract in one paragraph, and remember that the optimal length for an abstract is 200-300 words.

## 2 Vision general

Una interrupción de E/S es una ***señal asincrónica*** recibida por el procesador de una computadora, para indicar que debe «interrumpir» el curso de ejecución actual y pasar a ejecutar código específico para tratar esta situación. Las interrupciones son originadas por los dispositivos periféricos, para indicarle a la CPU que el estado del dispositivo cambió (por ejemplo, cuando una tecla fue presionada, o cuando la transferencia DMA desde el disco a memoria finalizó).

El programa siendo ejecutado hasta el momento de la interrupción es suspendido temporalmente, para pasar a ejecutar una ***subrutina de servicio de atención*** de la interrupción, la cual, por lo general, pertenece al sistema operativo, al BIOS, o es una parte especial del programa en ejecución en sistemas embebidos. Una vez finalizada dicha subrutina, se reanuda la ejecución del programa.

## 2 Operación básica de funcionamiento de la E/S mediante interrupciones

Para realizar una operación de E/S mediante interrupciones el programa en ejecución solicita(*) al dispositivo realizar una E/S. Esta acción se realizará mediante instrucciones ejecutadas por la CPU, para modificar adecuadamente los registros del control y estado del dispositivo. El controlador del dispositivo detecta el cambio del contenido de sus registros y determina cómo debe proceder.

Por ejemplo, si el periférico es un dispositivo de salida, la operación es comunmente una tranferencia y la CPU escribe tambien un dato (el dato que desea enviar el programa que inicia la operación) en el registro de datos del dispositivo (además de la modificación a los registros de estado y control). Una vez que se completó la transferencia del dato, el controlador del dispositivo modifica sus registros de estado para asentar la situacion, y dispara una interrupción para notificar a la CPU que la operación de transferencia finalizó. Es tarea de la CPU (hw) y de la subrutina de servicio de atención de la interrupción detectar si la transferencia fue satisfactoria (leyendo los registros de estado del dispositivo) y noti-

ficar al programa de la situación en caso de que fuses necesario. Un esquema de acciones en el tiempo desde el punto de vista del HW puede verse en la figura 1.

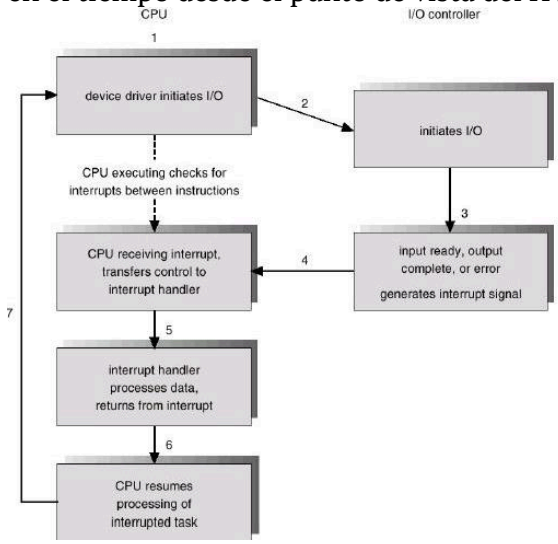


**Figure 1:** Operación básica de E/S mediante interrupciones. Vista desde el punto de vista HW.

Las interrupciones suceden asincrónicamente, y son generadas por los dispositivos periféricos para solicitar atención de la CPU. Por ejemplo. cuando un disco rigido transfiere datos a memoria via DMA y completa una lectura, el DMA solicita atención de la CPU. Lo mismo sucede cuando se presiona una tecla o se mueve el ratón, ambos periféricos cambian de estado, interrumpen, y solician atención de la CPU para comunicar la actividad. (*)(definir SEÑAL: pueden ser varias lineas conectadas a la CPU, como en el motorola 68000)

El mecanismo de interrupciones fue la solución a la E/S programada (polling) que permitió al procesador desentenderse de esta problemática, y delegar en el dispositivo periférico la responsabilidad de comunicarse con él cuando lo necesitara.

## 3 Funcionamiento

Todos los dispositivos que deseen comunicarse con el procesador por medio de interrupciones deben tener una línea asignada, capaz de notificar al CPU cuando le requiere para realizar una operación. Esta línea se denomina comunmente IRQ o INT (ACLARAR QUE OSCURECE).

## 3 Secuencia del funcionamiento de un sistema con interrupciones (procesamiento de una interrupción)

- 0. Habilitar las interrupciones globales y comenzar a ejecutar un programa. Cuando un periférico necesita atención:
- 1. Terminar la ejecución de la instrucción máquina en curso.
- 2. Deshabilitar las interrupciones
- 3. Resguardar el estado del procesador y el valor del contador de programa, de manera que la CPU, al terminar la ejecución de la subrutina de atención, pueda seguir ejecutando el programa a partir de la última instrucción máquina ejecutada en 1.
- 4. Ejecutar un salto (mediante una instrucción de bifurcación o actualización del PC) a la dirección donde está almacenada la subrutina de servicio de interrupción (Interrupt Service Routine, o abreviado ISR) y ejecutar esa subrutina que tiene como objetivo atender al dispositivo que generó la interrupción.
- 5. Una vez que la subrutina de servicio de la interrupción termina, restaurar el estado del procesador que se había resguardado en el paso 2. Continuar la ejecución del pro-

grama principal iniciado en 0.

## 4 Configuracion

Generalmente existen, en cada arquitectura que soporta interrupciones, mecanismos de configuracion o programacion de las interrupciones. Las cuales permiten: Habilitar y deshabilitar de manera global las interrupciones Habilitar y deshabilitar las interrupciones de periféricos específicos (enmascarar)

Asignar prioridades a los diferentes periféricos, lo cual permite, ante interrupciones en simultaneo, decidir en qué orden se deben atender las interrupciones (dispositivos) que necesitan atención. Tambien existe, casi siempre, interrupciones "no enascarables", que no pueden ser deshabilitadas debido a que son "indispensables" para el funcionamiento del computador. Estas siempre interrupen al procesador.

Asignar a cada periférico una rutina de atención ante una interrupcion.

## 5 Mecanismo de indentificación y ejecución de la rutina de atención

Generalmente la CPU realiza la siguiente secuencia de acciones ante una interrupción que necesite atención:

- 1. La CPU detecta la activación de su linea de interrupcion
- 2. Finaliza la instrucción en curso
- 3. Resguarda el estado del procesador
- 4. Reconoce la interrupcion. Comunmente, activa una señal que permite al periférico o controlador de interrupciones priorizar y detectar que el procesador está a punto de atender la interrupcion.
- 5. La CPU Identifica al periférico que necesita atención: de manera general, suele ser un ciclo de lectura desde la CPU, ante lo cual, el periferico o controlador de interrupciones se identifica con un valor que coloca en el bus de datos, comunmente llamado numero de vector.
- 6. La CPU, al leer el numero de vector, realiza una segunda lectura desde una tabla (llamada tabla de vectores), utilizando el numero de vector como identificador de la "entrada"de la tabla a leer. La tabla se encuentra en memoria y la dirección inicial (base) de la table es conocida por la CPU.
- 7. La dirección base + el numero de vector le permiten al hardware de la CPU realizar esta segunda lectura de la entrada especifica en la tabla de vectores, que corresponde al periférico que interrumpió.
- 8. De la tabla de vectores la CPU obtiene el vector, el cual, dependiendo de la arquitectura, es una dirección (de bifurcación) o una instrucción (la arquitectura define uno u otro, no ambos):
    - Si es una dirección para la arquitectura en particular, entonces el hw de la CPU modifica el PC con esta dirección y ejecuta la siguiente instrucción. De esta manera comienza la ejecución de la rutina de servicio a partir del PC ya modificado.
    - Si es una instrucción, entonces generalmente la instrucción es un salto, lo cual permite que cada vector (cada entrada en la tabla de vectores) contenga una instrucción salto a una rutina de servicio diferente.

  Por lo tanto, el vector es :

- una dirección de memoria de la primera instrucción de la rutina de atención; o
- una instrucción de salto que la CPU ejecuta para bifurcar a la rutina de atención de la interrupción.

Dar un ejemplo real con una linea de interrupcion para la CPU, un controlador de interrupciones, y detalles de lo anterior. Ejemplo del 8259 u otro.

Interrupciones al momento de la ejecución de la subrutina de atencion

Clasificar las arquitecturas reales en vectorizadas o no vectorizadas. Autovectorizadas.

## 6 Related Work

Preparing good-looking publications is not easy. It requires understanding of style and typography. The purpose of the templates provided by the USENIX organization is to lift the burden of caring about typography from the authors. However, the authors still remain, and will always remain, responsible for the style.

### 6.1 Word and LaTeX templates

The USENIX website includes a template for Microsoft Word, as well as LaTeX templates. Many of the settings in the CSS style sheet of this template have been copied from the LaTeX templates.

**Figure 2:** This figure is showed for illustrational purposes only; floppy disks are not required to use this template.

### 6.2 Style manuals

Besides typography, style is the second element of preparing easy-to-read publications. There are tens of good style manuals available. To mention just a couple, The Elements of Style by Strunk and White Strunk, W. Jr., and White, E.B. The Elements of Style, 4th Ed, Allyn and Bacon, August, 1999, ISBN 020530902X is a classic, and has remained a bestseller since its introduction in 1930's. From the more contemporary ones, Writing for Computer Science by Justin Zobel Zobel, J. Writing for Computer Science, Springer-Verlag, December 1997, ISBN 9813083220 seems appropriate.

## 7 Implementation

In this section we cover the features included in this template. Our goal has been that the authors do not need to make modifications to the template; instead, they should be able to concentrate on the content and style. With this in mind, this template includes a number of features. On the other hand, we have also tried to keep this document simple and easy to maintain.

This template is written in HTML, with CSS to provide styling, and a small JavaScript to help format references.

### 7.1 HTML5

This template uses HTML5 elements to aid in representing the document structure. The `section` element is used to split the text into sections, and the `header` element holds the headlines. The `figure` element is used to include figures and their corrensponding captions live inside the `figcaption` element. The `cite` element holds all references.

A small microformat, based on a convention of class names, is used to encode the name and affiliation of the authors.

### 7.2 CSS

A CSS style sheet describes how to format the HTML document into a PDF file. CSS is a declarative language which attaches property values to HTML elements and documents. Many aspects of CSS is used to achieve the presentation of USENIX papers, including:

**Figure 3:** This figure floats to the top of the page, spanning both columns.

- multi-column layout
- footnotes
- page and column floats
- multi-level counters

Some commonly used features are absent from the above list: page numbers and running headers should not be specifed by USENIX authors, these are added by those who compile the Proceedings.

### 7.3 JavaScript

This template uses JavaScript to process references. References are added at the point where they appear, and a script is later used to move the references to the end of the paper, leaving behind a numeric marker.

### 7.4 PDF

(This section has been added by Håkon Wium Lie)

In order to convert the document to PDF, a formatter is needed. Common browsers support HTML and CSS, but they do not support all the CSS functionality for page-based formatting. For example, browsers do not support footnotes or page floats. This paper has been formatted with Prince,[a] a purpose-built program for converting HTML and XML documents into PDF by way of CSS. Prince is a commercial product, but can be downloaded and used for free for non-commercial purposes.

In order for Prince to process the script included in this template, a command line option must be specified:

```
$ prince --javascript example.html
```

## 8 Tables

The table below lists recipients of the USENIX Lifetime Achievement Award in the 1900s. Notice how notes inside the table are moved to the end of the table.

| Year | Recipient |
|---|---|
| 1999 | X Window SystemGiven to the Community at Large |
| 1998 | Tim Berners-Lee |
| 1997 | Brian W. Kernighan |
| 1996 | The Software Tools Project |
| 1995 | The Creation of USENET Given to Jim Ellis and Tom Truscott |
| 1994 | Networking Technologies |
| 1993 | Berkeley UNIX |

## 9 Conclusions

Each good paper concludes the most significant findings in the end.

---

[a] www.princexml.com

## Acknowledgments

A polite author always includes acknowledgments. Thank everyone, especially those who funded the work.

## Availability

Please include a section at the end of your paper providing availability information. If the system you describe is available to others, and if more information (reports, etc.) may be obtained, indicate terms and contact information.

## References