UNIVERSIDAD DE ANTIOQUIA

1803

Facultad de Ingeniería

Departamento de electrónica

Curso de informática II

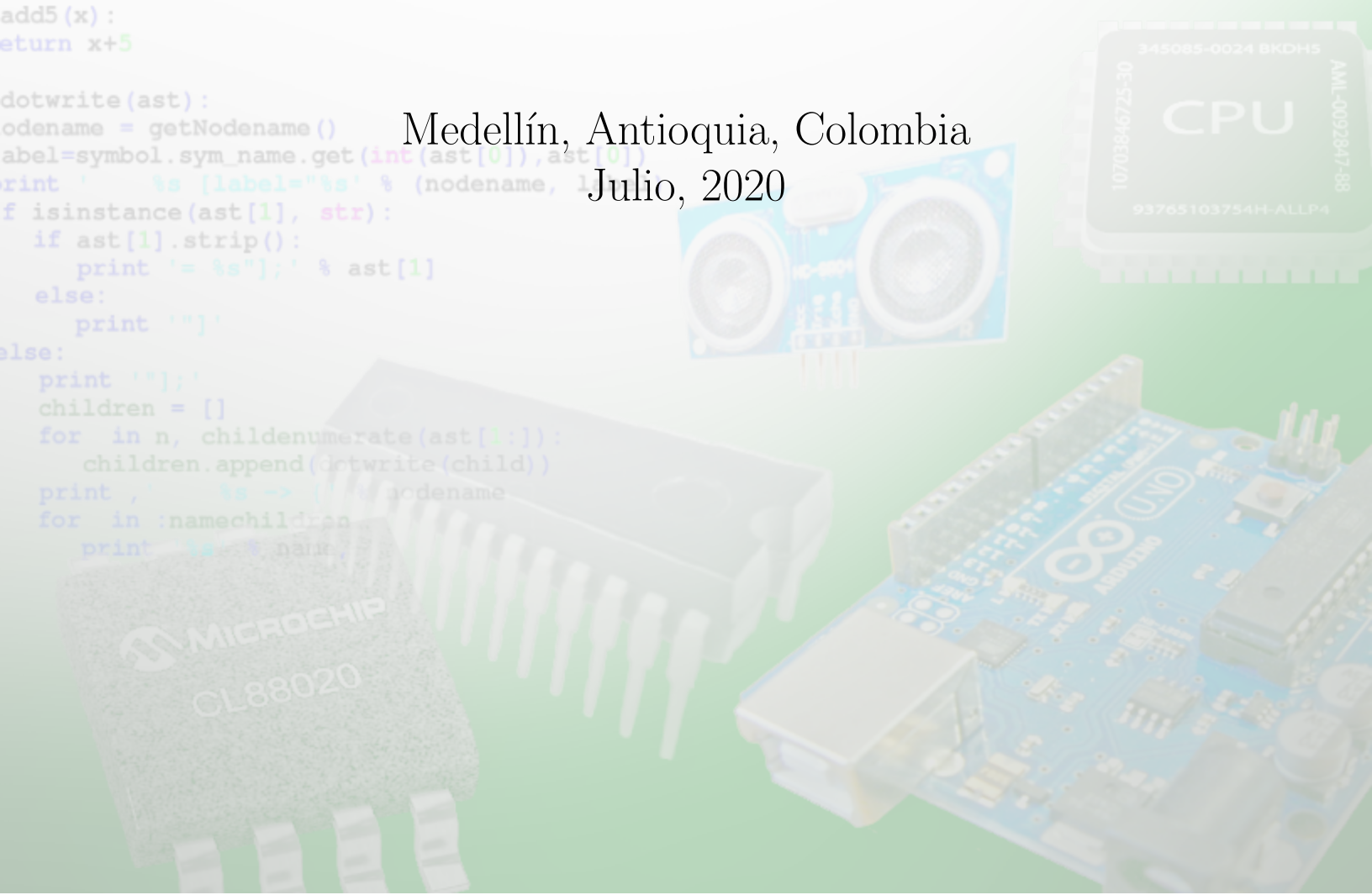
Proyecto de investigación:

Interrupciones en el campo de microprocesadores

por:

Santiago Giraldo Tabares

Medellín, Antioquia, Colombia
Julio, 2020



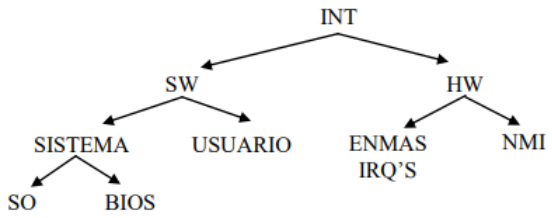
¿Qué son las interrupciones en microprocesadores?

Son, como su nombre indica, el cese de las tareas que se estén realizando en el procesador (rutina), con el fin de atender otra tarea tangente a su funcionamiento principal (subrutinas). Son solicitudes/señales que recibe el procesador para atender tareas específicas. Estas solicitudes pueden provenir de periféricos externos y de manera inesperada, o de un algoritmo y de forma pre-programada. Podemos hacer una analogía con una situación cotidiana: En un aula, un alumno levanta la mano interrumpiendo el transcurso de la clase, con el fin de que el maestro resuelva una duda puntual. En este caso, la rutina sería la clase, la interrupción sería el levantamiento de la mano y la subrutina constaría de resolver la duda. Una vez resuelta la inquietud, el maestro continúa con la clase. (Para profundizar a nivel técnico, ver [1])

¿Se puede hablar de la historia de las interrupciones?

Se puede hablar de la historia de las interrupciones en tanto que se hable de la historia de los microprocesadores. La idea de las interrupciones podemos verla aplicada hasta en el primer microprocesador, el 4004 de Intel (desarrollado en 1971) (estudiar el 4004 en [3]. Recorrido historico en [6]). Este realizaba un sondeo de todos los dispositivos esperando que alguno tuviese la configuración de bits que indicara que tenía una tarea para ser atendida. A esta técnica se la llama polling, y consta básicamente en preguntarle a todos los puertos/dispositivos, constantemente, si necesitaban atención para una tarea, y en caso de haber solicitudes de interrupción, atenderlas una a una (Profundizar en [4]). Llevado a la analogía anterior, sería cómo si el profesor estuviera cada 2 minutos, preguntando a todos sus estudiantes, si tienen alguna duda puntual. Esto era algo ineficiente y con el tiempo, el polling sería desplazado progresivamente por las interrupciones que conocemos hoy. Denominadas con este mismo nombre, o en inglés, ïnterruption request", cambian el modelo de sondeo, por el de "aviso", donde el procesador solo debe estar habilitado para recibir señales de interrupción, o como analizábamos antes, a que el estudiante levante la mano. Esto aumenta el costo a nivel Hardware pero es más rápido. Este cambio de mecanismo fue progresivo y es difícil datar un punto histórico exacto, sin embargo podríamos mencionar procesadores como el Intel 8080 e Intel 8085 [2], que fueron de los primeros en implementar el chip controlador de interrupciones (PIC) Intel 8259 [7], quien proveía al procesador con 15 canales de interrupción mediante los cuales el resto del hardware podía enviar sus solicitudes (datasheet 8080 [5] y sobre los PIC en [9]).

Tipos de interrupciones:



Clasificaciones. Referencia a la imágen: [10]
Existen de forma general, y según su proveniencia, dos tipos de interrupciones. Las interrupciones por hardware y las interrupciones por software.

Por Software:

Son solicitudes enviadas por un programa en ejecución, por lo tanto, su origen radica en el mismo procesador. Dado lo anterior, se les denomina "internas", pues un algoritmo en ejecución llega a una orden propia que lo obliga a pausar su curso y a llamar al S.O para que se gestione la realización de otra tarea externa. Se dice que son programables dado a que están explícitas en el código fuente del algoritmo en ejecución.

En estas, cabe aclarar que tanto el lenguaje del programa como el hardware utilizado IMPORTAN y afectan la implementación, dado que cada lenguaje utiliza sus propias palabras reservadas y funciones para enviar la petición de interrupción a

un determinado canal y dadas ciertas condiciones o ciertos parámetros para la funciones. Además, el hardware debe estar adecuado con los suficientes canales de gestión de interrupciones, y debe consonar con las expectativas del lenguaje a utilizar. También afecta la cercanía del lenguaje utilizado con el lenguaje de máquina. En varios tipos de interrupciones es primordial la velocidad de atención, por lo tanto los lenguajes de bajo nivel ofrecen una ligera ventaja en ese aspecto. También, cuanto más se avanza en la miniaturización de los microprocesadores, también se avanza tanto en la velocidad de emisión de las peticiónes como en la de la ejecución de las subrutinas. También hay microprocesadores que tienen excepciones propias dadas la arquitectura de su ISA como su arquitectura física, así que en conclusión, el hardware y software utilizados marcan la diferencia.

Por Hardware:

Son provenientes de periféricos externos, como por ejemplo, dispositivos de entrada y salida. Dado lo anterior, se les denomina "externas", y debido a su naturaleza, no son eventos programados, ergo, pueden ocurrir en cualquier momento. Consideremos el trabajo de construcción de un edificio: Pueden ocurrir fallos inesperados, pero por ejemplo: las escaleras del piso 5 no están programadas para derrumbarse a las 2 semanas de comenzada la construcción, no es propio de su naturaleza y dependerá de factores externos.

Excepciones:

Existe otro tipo de interrupciones que podríamos calificar como internas. Hablamos de las excepciones. Las excepciones son interrupciones que se originan de las instrucciones anormales del lenguaje del algoritmo. En otras palabras, son aquellas que detienen y abortan el programa en ejecución por una instrucción mal construída, como las divisiones entre cero, violación de privilegios, error de los canales, instrucciones ilegales, etc. Son sincrónicas, pues se presentan en el momento exacto en el que el procesador llega a determinada instrucción y procede a dar un tratamiento que generalmente termina en la notificación al usuario/sistema y con el aborto de la ejecución, reanudando la rutina previa (ver [11]).

Por Timer:

Así mismo, podríamos mencionar un subtipo de las interrupciones por software, bastante específico y de uso común en microcontroladores. Hablamos de los timers. Estos permiten que las instrucciones de un algoritmo se ejecuten en un tiempo planificado y no a la vertiginosa velocidad de nuestro procesador. Permite generar pausas y su utilización puede varíar (o de plano, existir) dependiendo del lenguaje de programación utilizado y la compatibilidad de nuestro hardware. Estas nos permiten, por ejemplo, controlar el tiempo de ejecución de un algoritmo mientras se ejecuta otra tarea en paralelo de forma constante. A nivel practico, podríamos encender y apagar un LED cada n segundos y tener otro algoritmo ejecutandose normalmente en segundo plano (un Buzzer, por ejemplo). Hay funciones como las de la librería TimerOne.h en arduino, las cuales permiten manipular estos timers para interrumpir, por periodos de tiempo controlados, una labor.

Clasificación según la prioridad de su solicitud:

Las anteriores no son las únicas formas de categorizar las interrupciones. Tambien podemos clasificar las solicitudes de interrupción (no tanto LAS interrupciones como tal) según si son o no ignorables, proviniendo de un periferico Hardware.

Enmascarables:

Cuando el procesador recibe una petición de interrupción, este está en capacidad de apoyarse del software para decidir si ignorar o no esta petición, continuando con el curso normal de ejecución. En el registro de estatus (FLAG REGISTER) del procesador, o más específicamente, del controlador de interrupciones que este tiene, deben estar habilitadas las interrupciones cambiando el estado binario de un bit que hace de bandera o switch para cada

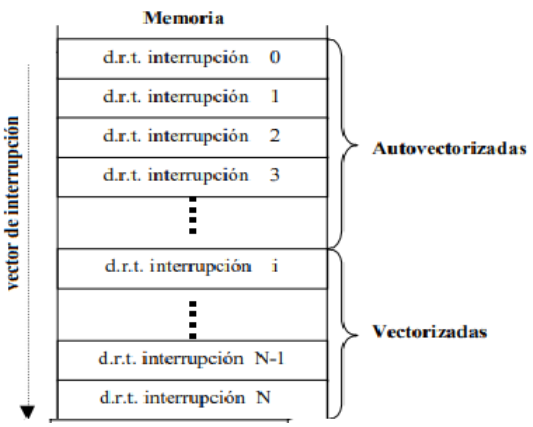
canal de interrupciones. El estado de dicho bit puede ser manipulado por el usuario con instrucciónes de software. Se pueden inhibir todos los canales de interrupción o también se pueden inhibir canales de manera individual. Estas permiten que el controlador de interrupciones las pueda filtrar y administrar según la prioridad que requieran sus periféricos emisores, haciendo eficiente la gestión de los limitados canales de interrupción y permitiendo que la capacidad multi-tarea de los microprocesadores no se desborde provocando daños, paradas o ineficiencia.

No enmascarables:

Este tipo de solicitudes de interrupción son aquellas que no pueden ser interrumpidas y el procesador si o si debe efectuar la subrutina correspondiente para tratarla. Estas son especialmente útiles cuando el tiempo de respuesta a una tarea externa debe ser inmediato, de primerísima prioridad y en situaciones críticas que requieran tratamiento no solo obligatorio, sino rápido.

*Referencia sobre las anteriores clasificaciones en [10]

Clasificación según método de asignación de subrutina:



Referencia a la imágen: [8]

Por último, también podemos llegar a toparnos con términos que aluden a la forma en la que se conectán las peticiónes recibidas con las subrutinas o tratamientos que deben llevarse a cabo. En la memoria del computador, se encuentra almacenado un vector de interrupciones que almacena en cada una de sus posiciones, las direcciones de memoria de cada una de las subrutinas que debe ejecutar el procesador ante cada solicitud específica. a continuación se explica en que forma pueden llegar las peticiones de interrupción:

No vectorizadas:

Estas, también llamadas 'Autovectorizadas', son aquellas a las que el periferico emisor de la petición no les ha asignado un vector de interrupción, por lo tanto debe ser el procesador quién, a través de software, asigne un canal para que dicha solicitud pueda ser procesada. El procesador también debe inspeccionar el periferico emisor para determinar la prioridad de la petición.

Vectorizadas:

Son aquellas que sí cuentan con un vector de interrupción. Una vez se reconoce la interrupción al dispositivo externo, este asigna el vector a la solicitud, con lo que llegará a una determinada posición en memoria para que el procesador pueda proceder luego con la subrutina correspondiente.

Referencia a vectorización en [8]

Ejmplo de implementación:

A continuación, se adjunta el link a una simulación en Tinkercad elaborada para ejemplificar una interrupción externa sobre una placa Arduino UNO R3, con la respectiva explicación del código fuente: `https://www.tinkercad.com/things/g2QTusrjTsS`.

Referencias

[1] Ruz Ortíz, J.J. *Tema 9: interrupciones*, Dpto. de Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid.
http://www.fdi.ucm.es/profesor/jjruz/WEB2/Temas/Curso05_06/EC9.pdf

[2] Quintero Reyes, A.G. *interrupciones*, [Archivo de video], recuperado de https://www.youtube.com/watch?v=bIXYtXL587M. Fuente verificada en https://repository.unad.edu.co/handle/10596/5356

[3] Shima, M. *The 4004 CPU of My Youth: Developing the world's first microprocessor.*, IEEE SOLID-STATE CIRCUITS MAGAZINE, (Winter, 2009).
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4776533

[4] Chishti, Z. *Microprocessor System Design*Electrical and Computer Engineering Dept, Maseeh College of Engineering and Computer Science, Portland State University
http://web.cecs.pdx.edu/~zeshan/ece585_lec3.pdf

[5] Intel *8080A/8080A-1/8080A-2. 8 bit N-channel microprocessor [official datasheet]*
http://www.datasheetcatalog.com/datasheets_pdf/8/0/8/0/8080.shtml

[6] Morse, S.P. Ravenel, B.W. Mazor, S. Pohlman, W.B. *Intel Microprocessors: 8008 to 8086*, IEEE Computer, Vol 13, No. 10, pages 42-60, October 1980 pp. 73-74.
http://stevemorse.org/8086history/8086history.doc

[7] Intel *8259A PROGRAMMABLE INTERRUPT CONTROLLER (8259A/8259A-2) [Official datasheet]*
https://pdos.csail.mit.edu/6.828/2005/readings/hardware/8259A.pdf

[8] Ruz Ortiz, J.J. *Tema 3. Entrada/salida programada e interrupciones*, Dpto. de Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid.
http://www.fdi.ucm.es/profesor/jjruz/EC-IS/Temas/Tema%203-%20Entrada_salida%20programada%20e%20interrupciones.pdf

[9] Wainer, G.A. *The Programmable Interrupt Controller*, SYSC-3006* - Computer Organization, Department of Systems and Computer Engineering, Carleton University (Fall, 2011).
http://www.sce.carleton.ca/courses/sysc-3006/f11/Part18-PIC.pdf

[10] Beltrán Martínez, B. *Ensamblador. Interrupciones*,Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, México.
http://bbeltran.cs.buap.mx/Interrupciones.pdf

[11] *PRÁCTICAS DE MICROPROCESADORES PRÁCTICA V. "MANEJO DE EXCEPCIONES". Implementación de un reloj*,DPTO. ELECTRÓNICA, AUTOMÁTICA E INFORMÁTICA INDUSTRIAL, Universidad Politécnica de Madrid, España.
http://www.ieef.upm.es/webantigua/spain/Asignaturas/Microprocesadores2002/practicas/up-p5.pdf

[12] Santamaría, E. *Microprocesador 68000: Hardware y Software*, Universidad Pontificia de Comillas(pp. 22 - 25).
https://books.google.com.co/books?id=Us64bLlWh54C&printsec=frontcover#v=onepage&q&f=false

[13] Valdés Peres, F.E. Pallás Areny, R. *Microcontroladores Fundamentos y Aplicaciones con PIC*, Universidad de Oriente, Cuba. Universidad Politécnica de Cataluña, España, 2007.(pp. 217 - 218).
https://books.google.com.co/books?id=ODenKGOHMRkC&printsec=frontcover#v=onepage&q&f=false

[14] Burgess, P. *Adafruit NeoPixel Überguide* [Guía oficial de uso de las librerías de Adafruit Neopixel][referencia del uso de la implementación en Arduino]
https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library-use