[Resumen clase 2](https://drive.google.com/file/d/1gZ4atzH_eyJJo4ZBYKbkyCwPUHj_sfFU/view?usp=sharing)

# Interrupciones

* Mecanismo mediante el cual se puede interrumpir el procesamiento normal de la CPU (ejecución secuencial de instrucciones)
* Pueden ser de origen interno o externo a la CPU

## Por qué interrumpir?

* Resultado de una instrucción (overflow, división por cero)
* Temporizador interno del procesador (permite al SO realizar ciertas funciones de manera regular)
* Operación E/S (para indicar la finalización normal de una operación)
* Fallo de hardware (error de paridad en la memoria, pérdida de energía)

## Qué hacer si interrumpen?

* En todos los casos, implica transferir el control a otro programa (gestor) que:
  + Salve el estado del procesador
  + Corrija (o responda) a la causa que ocasionó la interrupción
  + Restaure el estado original del procesador
  + Retorne a la ejecución normal del programa interrumpido
* Continúo o repito la instrucción interrumpida?

## Jerarquía de interrupciones

Si hay múltiples fuentes que pueden solicitar una interrupción se establece cuáles son las más importantes. Se consideran:

* No enmascarables: Las que NO pueden ignorarse (eventos peligrosos o de alta prioridad)
* Enmascarables: Pueden ser ignoradas (con instrucciones podemos inhibir la posible solicitud)

## Interrupciones por hardware

* Generadas por dispositivos de E/S
* Son las “verdaderas” interrupciones
* El sistema de cómputo tiene que manejar estos eventos externos “no planeados” o “asincrónicos”
* No están relacionados con el proceso en ejecución en ese momento
* Conocidas como **interrupt request**

### Traps/excepciones

Interrupciones por hardware creadas por el procesador en respuesta a ciertos eventos como:

* Condiciones excepcionales (overflow en ALU de punto flotante)
* Falla de programa (tratar de ejecutar una instrucción no definida)
* Fallas de hardware (error de paridad de memoria)
* Accesos no alineados o a zonas de memoria protegidos

## Interrupciones por software

Muchos procesadores tienen instrucciones explícitas que afectan al procesador de la misma manera que las interrupciones por hardware.

* Generalmente usadas para hacer llamadas a funciones del SO
  + Esta característica permite que las subrutinas del sistema se carguen en cualquier lugar
* No requieren conocer la dirección de la rutina en tiempo de ejecución

Hay sistemas que no permiten hacer una llamada directa a una dirección de la función del SO, por estar en una zona reservada.

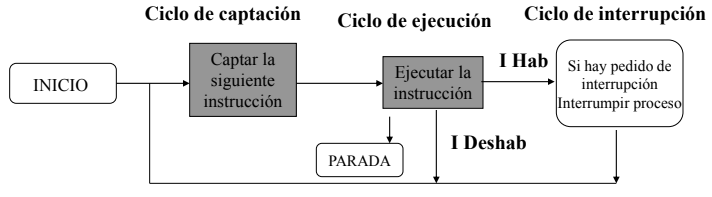
Sin las interrupciones por software:

* Debería escribir todas las funciones que necesito o,
* Al cargar un programa habría que “mirar” todas las llamadas a funciones del BIOS y SO y reemplazar en el código las direcciones de todas estas funciones invocadas

## Nuevo ciclo de instrucción

3 pasos:

* Captación
* Ejecución
* Gestión de interrupciones



### Ciclo de interrupción

* Se comprueba si se ha solicitado una interrupción (flag)
* Si no hay señal se capta la siguiente instrucción
* Si hay pedido de interrupción pendiente:
  + Se suspende la ejecución del programa en curso
  + Guarda su contexto (próxima instrucción a ejecutar y el estado del procesador)
  + Carga el PC con la dirección de comienzo de una rutina de gestión de interrupción. Se inhiben otras interrupciones
  + Finalizada la rutina de gestión, el procesador retoma la ejecución del programa del usuario en el punto de la interrupción

Cómo trabajan:

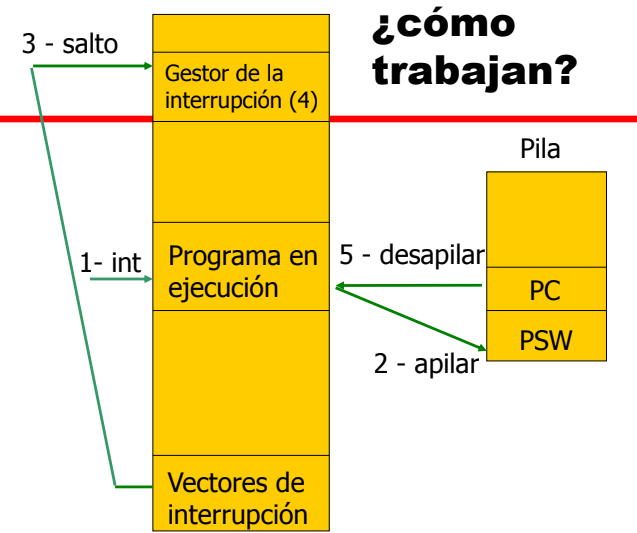
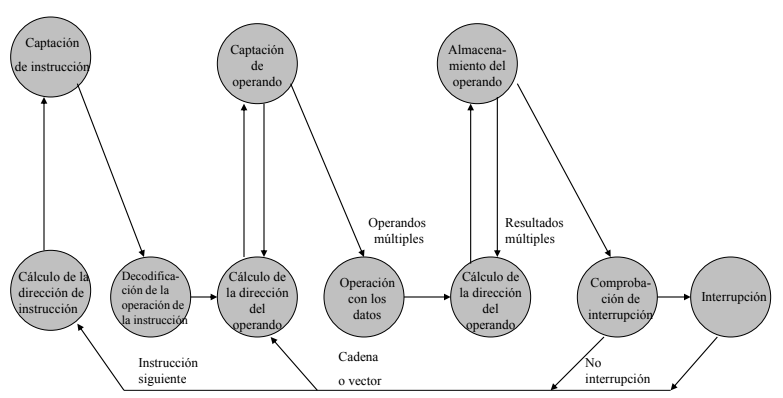


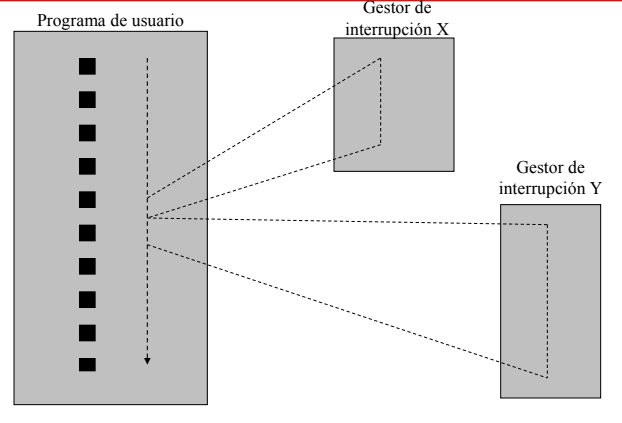
Diagrama de estados de un ciclo de instrucción con interrupciones:



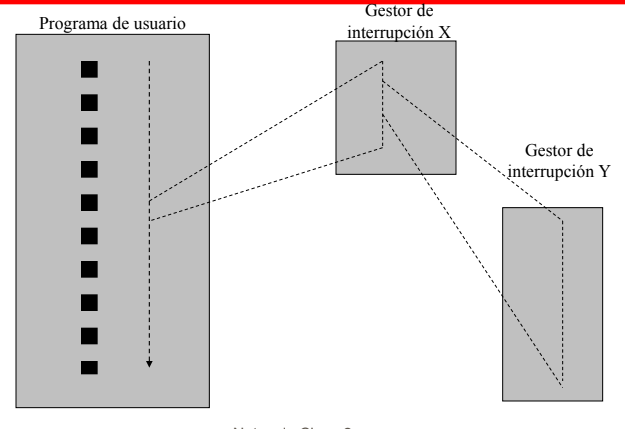
## Interrupciones múltiples

* Interrupciones inhabilitadas
  + El procesador puede y debe ignorar la señal de petición de interrupción si se produce una interrupción en ese momento
  + Si se hubiera generado una interrupción se mantiene pendiente y se examinará luego, una vez que se hayan habilitado nuevamente
  + Ocurre una interrupción, se inhabilitan, se gestiona la misma y luego se habilitan otra vez
  + Por lo tanto las interrupciones se manejan en un orden secuencial estricto
* Definir prioridades
  + Una interrupción de prioridad más alta puede interrumpir a un gestor de interrupción de prioridad menor
  + Cuando se ha gestionado la interrupción de prioridad más alta, el procesador vuelve a las interrupciones previas (de menor prioridad)
  + Terminadas todas las rutinas de gestión de interrupciones se retoma el programa del usuario

## Procesamiento de interrupciones secuenciales



## Procesamiento de interrupciones anidadas (priorización)



## Reconocimiento de interrupciones

Interrupciones multinivel

* Cada dispositivo que puede provocar interrupción tiene una entrada física de interrupción conectada a la CPU
* Es sencillo pero caro

Línea de interrupción única

* Una sola entrada física de pedido de interrupción a la que están conectados todos los dispositivos
* Se debe “preguntar” a cada dispositivo si ha producido el pedido de interrupción (técnica polling/encuesta)

Interrupciones vectorizadas

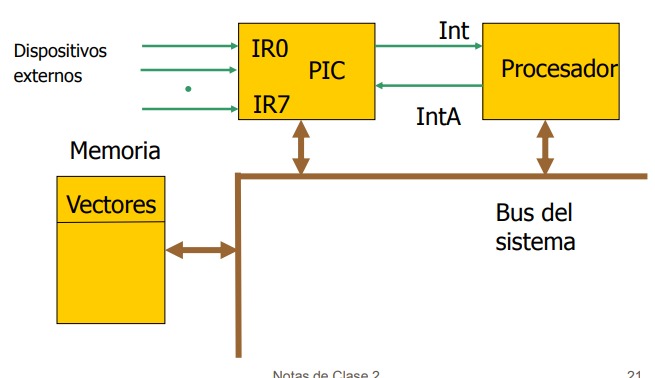
* El dispositivo que quiere interrumpir, además de la señal de pedido de interrupción, debe colocar en el bus de datos un identificador (vector). Lo coloca
  + Directamente el periférico o,
  + El controlador de interrupciones (que se ocupa de todo)

## Escenario de trabajo

* Si el procesador tiene una única entrada de pedido de interrupciones
* Si tenemos varios productores de interrupciones

Lo solucionamos con el **PIC** (Dispositivo Controlador Programable de Interrupciones)

### Conexionado



## Interrupciones del MSX88

* Hardware
  + Línea INT (Con respuesta de reconocimiento INTA)
  + Línea NMI (Procesos de atención son por salto indirecto)
* Software
  + Instrucción INT xx (Para retorno desde el gestor debe usarse la instrucción IRET)
* Proceso de atención vectorizado

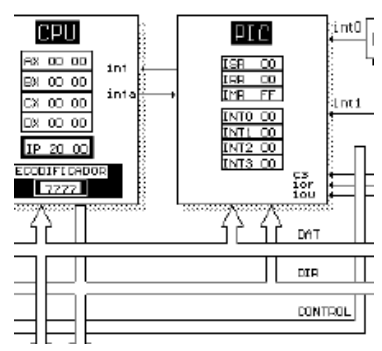
### Tabla de vectores de interrupción

* Es el nexo entre tipo de interrupción (0..255) y el procedimiento designado para atenderla
* Cada entrada es una doble palabra (4 bytes)
  + Dirección del procedimiento que brinda el servicio
  + Ej: 0000yyyy, donde yyyy es la dirección lógica/física
* Vectores preasignados
  + Tipo 0 - Finaliza ejecución del programa
  + Tipo 3 - punto de parada para depuración/seguimiento
  + Tipo 6 - lectura de entrada std. Requiere el uso de BX
  + Tipo 7 - escritura de salida std. Requiere BX y AL

### Controlador de interrupciones

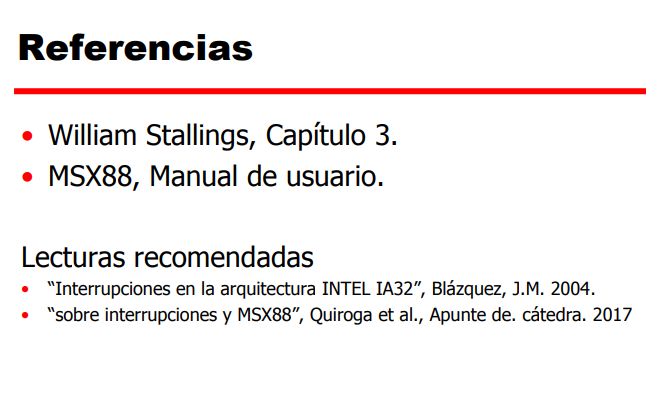
Registros internos del PIC

* EOI: Para comandos (para fin de int escribir 20h)
* IMR: Mascara de int (enmascara con 1)
* IRR: Petición de int (indica con bit en 1)
* ISR: Int de servicio (indica con bit en 1)
* INT0...INT7 (cada una con su vector)



### Conexionado y direccionamiento

* Los registros internos del PIC se situan a partir de la dirección 20H
* Son accedidos con operaciones lectura y escritura en el espacio de E/S (IN y OUT)
* Interrupciones hardware asignadas
  + INT0 - Tecla F10
  + INT1 - Timer
  + INT2 - Handshake
  + INT3 - DMA
  + INT4 a INT7 no usadas

[Anexo clase 2](https://drive.google.com/file/d/1TGiqBRUKew81guApCOUvKab_Gv1MaqLi/view?usp=sharing)