Arquitectura de computadoras

- 2020

Explicación Práctica 2

Interrupciones

- Mecanismo mediante el cual se puede interrumpir el procesamiento normal de la CPU (ejecución secuencial de instrucciones de un programa).

- Según su origen se clasifican en:
  - **Internas** → Interrupciones por Software
  - **Externas** → Interrupciones por Hardware

- Según su importancia se clasifican en:
  - **No enmascarables**: no pueden ignorarse → Indican eventos peligrosos o de alta prioridad
  - **Enmascarables**: pueden ser ignoradas

Interrupciones

- ¿Por qué utilizar interrupciones?
  - Resultado de una ejecución de una instrucción (ej: overflow, división por cero, etc)
  - Temporizador interno del procesador (ej: permite al SO realizar tareas de forma regular)
  - Operación de E/S (ej: finalización de impresión)
  - Por un fallo de hardware

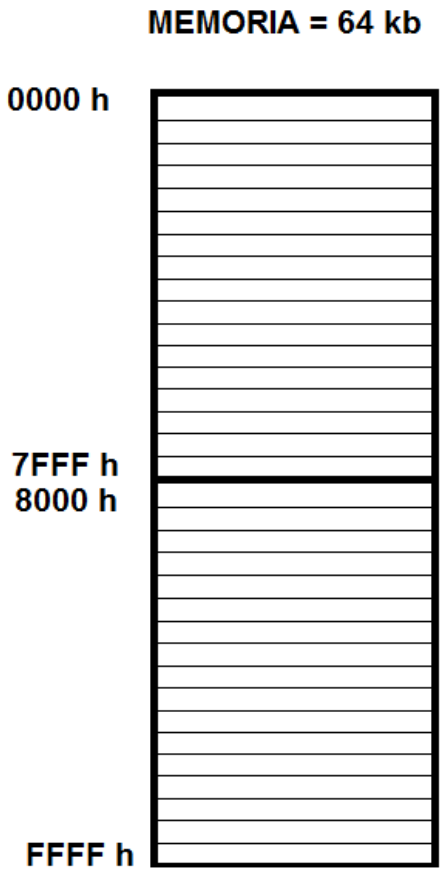
Interrupciones

- Cuando un dispositivo quiere interrumpir debe envíar la siguiente información:
  - Una **señal** de pedido de interrupción
  - Un **identificador** (para indicar el tipo de interrupción)
- Se usa un “Controlador de Interrupciones Programable” (**PIC**)
  - Permite configurar qué interrupciones estan habilitadas y dónde se encuentra la rutina que atiende cada interrupción

Interrupciones

- Una interrupción puede suceder en cualquier momento, por lo que antes de ir a la rutina de atención de la misma, se guarda automaticamente en la pila:
  - Las banderas → **Flags**
  - La dirección de retorno → **IP**

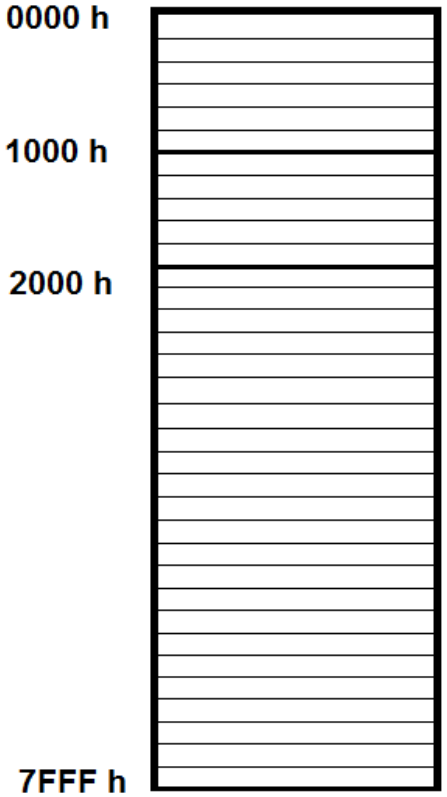
Interrupciones



→ **Memoria direccionable** (32 kb)
Datos y programas

→ **Memoria no direccionable** (32 kb)
Sistema Operativo

Interrupciones



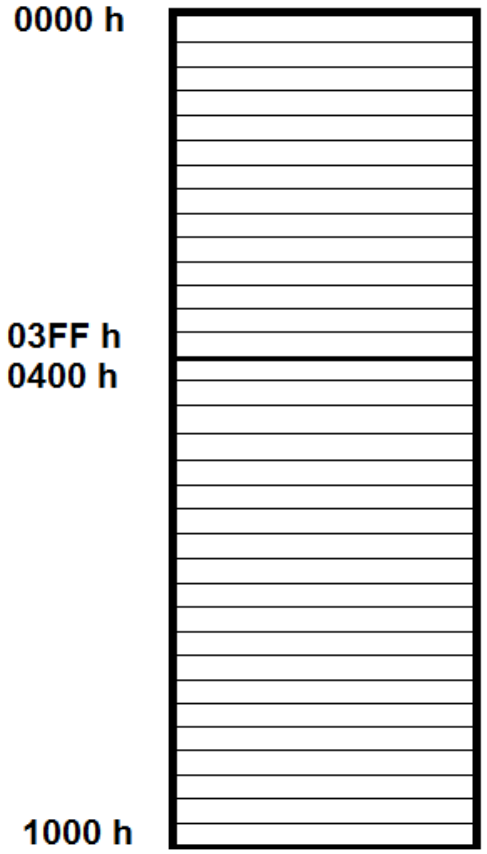
→ ¿?

→ Memoria de datos

→ Memoria de instrucciones

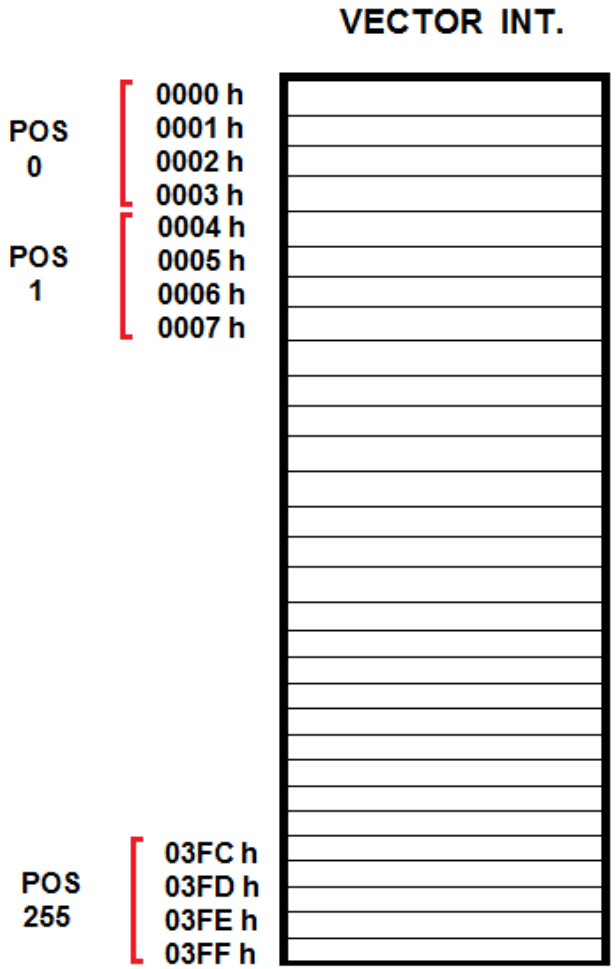
→ (Pila)

Interrupciones



→ **Vector de interrupciones** (0..255)

Interrupciones



→ Cada lugar del vector tiene **32 bits**, que permiten guardar la **dirección de la rutina** que atiende a la interrupción

Interrupciones

- **Interrupciones por software**: utilizan posiciones del vector preasignadas. Por eso dichas posiciones no pueden ser utilizadas para definir otras rutinas de interrupción
  - INT 0 → **HLT**
  - INT 3 → **DEBUG** (punto de parada para depuración)
  - INT 6 → **LEER** UN CARACTER DESDE TECLADO
    - **BX** debe tener la dirección donde se almacena el carácter
  - INT 7 → **ESCRIBIR** EN PANTALLA UNA CADENA DE CARACTERES
    - **BX** debe tener la dirección de inicio de la cadena de caracteres
    - **AL** debe tener la cantidad de caracteres a escribir

Ejercicios

1) Implementar un programa que muestre en la pantalla un mensaje previamente almacenado en memoria de datos.

```
        ORG 1000H
MSJ DB "ARQUITECTURA DE COMPUTADORAS"
        DB "FACULTAD DE INFORMATICA"
        DB 55H
        DB 4EH
        DB 4CH
        DB 50H
FIN  DB ?

        ORG 2000H
        MOV BX, OFFSET MSJ
        MOV AL, OFFSET FIN - OFFSET MSJ
        INT 7
        INT 0
        END
```

Ejercicios

2) Escribir un programa que solicite el ingreso de un número (de un dígito) por teclado e inmediatamente lo muestre en la pantalla.

```
        ORG 1000H
MSJ     DB "INGRESE UN NUMERO:"
FIN     DB ?

        ORG 1500H
NUM     DB   ?

        ORG 2000H
        MOV BX, OFFSET MSJ
        MOV AL, OFFSET FIN - OFFSET MSJ
        INT 7
        MOV BX, OFFSET NUM
        INT 6
        MOV AL, 1
        INT 7
        MOV CL, NUM
        INT 0
        END
```

Ejercicios

- Los ejercicios 1 a 9 de la práctica utilizan sólo **interrupciones por software**

- A partir del ejercicio 10 en adelante se incorporan las diferentes **interrupciones por hardware**

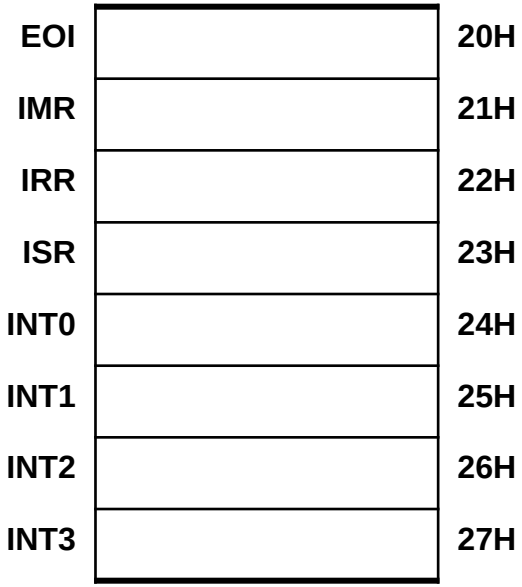
Interrupciones

- **Interrupciones por hardware**
  - Conocidas como “interrupt request” (pedido de interrupción)
  - Son generadas por dispositivos de E/S
  - No están relacionadas con el proceso en ejecución en ese momento
  - El sistema de cómputo debe ser capaz de manejar estos eventos externos “no planeados” ó “asincrónicos”

    → **Por eso es que ante un pedido de interrupción se guarda los flags y el IP de retorno**

Interrupciones

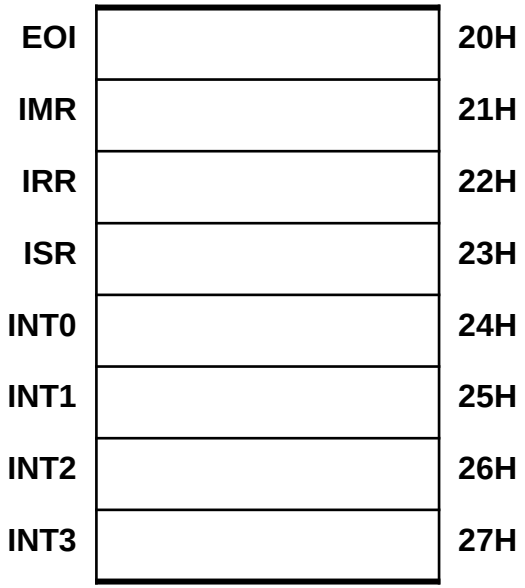
- **Controlador de Interrupciones Programable (PIC)**



- **EOI:** fin de int.
- **IMR:** máscara de int.
- **IRR:** peticiones de int.
- **ISR:** int. en servicio
- **INT0…INT3:** posición del vector en donde se encuentra la dir. de la rutina que atiende a cada int.
- **INT4…INT7:** existen pero no se utilizan.

Interrupciones

- **Controlador de Interrupciones Programable (PIC)**



- **EOI** → escribir 20h finalizar
- **IMR**: enmascara con ‘1’
- **IRR**: indica pedido con ‘1’
- **ISR**: indica en servicio con ‘1’
- **INT0**: F10
- **INT1**: TIMER
- **INT2**: HANDSHAKE
- **INT3**: DMA

Lineas asignadas por defecto a estos periféricos

Interrupciones

- **Controlador de Interrupciones Programable (PIC)**
  - Dado que se trata de memoria de E/S, las direcciones del PIC son accedidas con operaciones de lectura y escritura en el espacio de E/S → **IN** y **OUT**
  - Mientras se esta configurando los registros del PIC, no se deben atender las interrupciones que pudiesen ocurrir. Para esto se cuenta con dos instrucciones específicas:
    - **CLI** → CLEAR INTERRUPS
    - **STI** → SET INTERRUPS

Ejercicios

Interrupción por hardware: tecla F10

10) Escribir un programa que, mientras ejecuta un lazo infinito, cuente el nro de veces que se presiona la tecla F10 y acumule este valor en DX.

```
PIC     EQU 20H
EOI     EQU 20H
N_F10       EQU 10

            ORG 40
IP_F10      DW RUT_F10

            ORG 2000H
            CLI
            MOV AL, 0FEH
            OUT PIC+1, AL ; IMR
            MOV AL, N_F10
            OUT PIC+4, AL ; INT0
            MOV DX, 0
            STI
LAZO:  JMP LAZO
```

```
            ORG 3000H
RUT_F10:    PUSH AX
            INC DX
            MOV AL, EOI
            OUT EOI, AL  ; EOI
            POP AX
            IRET

            END
```

Timer

- Posee dos registros de 1 byte c/u:
  - **CONT (10h):** registro contador → ***se incrementa su valor cada un segundo***
  - **COMP (11h):** registro de comparación
- Cuando estos registros coinciden en su valor, se provoca una señal de interrupción de TIMER

Ejercicios

Interrupción por hardware: TIMER

12) Implementar un reloj segundero que muestre en pantalla los segundos transcurridos (00-59 seg) desde el inicio de la ejecución.

```
TIMER   EQU   10H
PIC     EQU   20H
EOI     EQU   20H
N_CLK   EQU   10

        ORG 40
IP_CLK  DW    RUT_CLK

        ORG 1000H
SEG     DB    30H
        DB    30H
FIN     DB     ?
```

```
        ORG 2000H
        CLI
        MOV AL, 0FDH
        OUT PIC+1, AL       ; PIC: IMR
        MOV AL, N_CLK
        OUT PIC+5, AL       ; PIC: INT1
        MOV AL, 1
        OUT TIMER+1, AL     ; TIMER: reg COMP
        MOV AL, 0
        OUT TIMER, AL       ; TIMER: reg CONT
        MOV BX, OFFSET SEG
        MOV AL, OFFSET FIN-OFFSET SEG
        STI
LAZO:   JMP LAZO
```

Ejercicios

Interrupción por hardware: TIMER

12) Implementar un reloj segundero que muestre en pantalla los segundos transcurridos (00-59 seg) desde el inicio de la ejecución.

```
TIMER   EQU   10H
PIC     EQU   20H
EOI     EQU   20H
N_CLK   EQU   10

        ORG 40
IP_CLK  DW    RUT_CLK

        ORG 1000H
SEG     DB    30H
        DB    30H
FIN     DB     ?
```

```
        ORG 3000H
RUT_CLK: PUSH AX
        INC SEG+1
        CMP SEG+1, 3AH
        JNZ RESET
        MOV SEG+1, 30H
        INC SEG
        CMP SEG, 36H
        JNZ RESET
        MOV SEG, 30H
RESET:  INT 7
        MOV AL, 0
        OUT TIMER, AL
        MOV AL, EOI
        OUT PIC, AL
        POP AX
        IRET
```