Interrupciones

INT: ejecuta la rutina de servicio a la interrupción indicada por el número. INT número

IRET: retorno de la rutina de servicio.

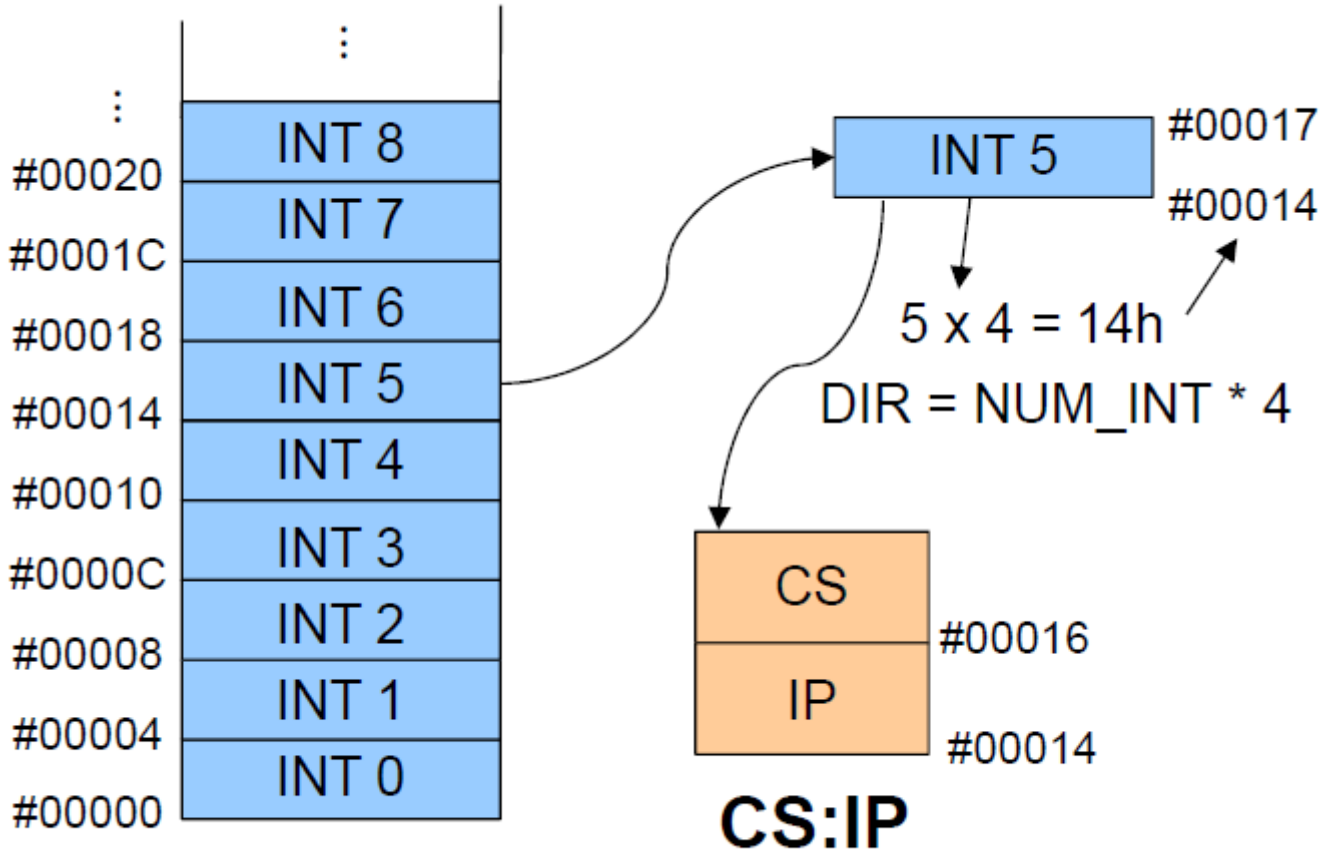
Las interrupciones son llamadas a rutinas del sistema (normalmente servicios del BIOS o del SO).

Estas rutinas están “residentes” en memoria.

Las posiciones de memoria donde empiezan las rutinas se guardan en una tabla en memoria. Esta tabla se encuentra al principio de la memoria en DOS: desde la dirección 0 a la 3FFh.

Existen Interrupciones Software (llamadas INT n) y Hardware (pines de procesador INTR y NMI)

Cada 4 bytes de esta tabla constituyen un vector de interrupción (offset y segmento donde comienza la rutina de servicio a esa interrupción).



Instalación de una rutina de servicio a interrupción:

```
DIR equ 4 * NUM_INT
mov ax, 0
mov es, ax
cli
mov es:[ DIR ], OFFSET rutina_servicio
mov es:[ DIR + 2 ], SEG rutina_servicio
sti
```

Fases de ejecución de una interrupción por la CPU:

1. Se apilan banderas y dirección de retorno.
2. Se ponen a 0 bit de interrupción **IF** y de traza **TF** (enmascarando interrupciones hardware y desactivando ejecución paso a paso).
3. Se lee vector de interrupción (**CS**:**IP**) con dirección de primera instrucción de la rutina de servicio.
4. Se ejecuta la rutina de servicio.
5. La rutina de servicio acaba con instrucción **IRET**.
6. Se desapilan dirección de retorno y estado.

Interrupciones BIOS

INT 1Ch: Tic del temporizador
La activa la rutina de servicio de la **INT 8** (timer).
El BIOS inicializa el vector de interrupción con una dirección que contiene la instrucción **IRET**.

Interrupciones DOS

INT 20h: Finaliza programa
Acaba ejecución de programa retornando al intérprete de comandos. Microsoft recomienda usar en su lugar **INT 21h** con **AH**=4Ch (finaliza programa, cerrando ficheros y liberando memoria).

INT 21h: *Dispatcher* del DOS
Ejecuta los distintos servicios del DOS según **AH**.

INT 27h: Finaliza programa dejando residente
Acaba ejecución de un programa .COM (*driver*) dejándolo residente en memoria.

Ejecución de programas desde el DOS

Como parte de la carga se añade una zona de 256 bytes que contiene datos relacionados con el programa (Prefijo de Segmento de Programa, PSP)

Los ficheros ejecutables pueden estar en formato .EXE o .COM, teniendo su ejecución un comportamiento ligeramente distinto.

Cuando acaba un programa, se devuelve el control al intérprete de comandos del DOS. La memoria que ocupaba se libera salvo que se deje residente.

PSP (Prefijo de Segmento de Programa)

Zona de datos de 256 bytes que encabeza los programas .EXE o .COM una vez están cargados en memoria RAM para su ejecución.

Generada por el DOS mediante el intérprete de comandos (COMMAND.COM).

Campos más destacados del PSP

Offsets 2Ch y 2Dh (2 bytes)

Número de segmento físico que contiene una copìa de las variables de entorno del DOS. Permite al programa acceder a esas variables.

Offset 80h (1 byte)
Tamaño en bytes de los parámetros del programa en línea de comandos.

Offsets 81h a FFh (127 bytes)
Códigos ASCII de los parámetros del programa en línea de comandos. Acaba con código 13 (retorno de carro). Permite al programa acceder a los parámetros indicados por línea de comandos.

Ejemplo

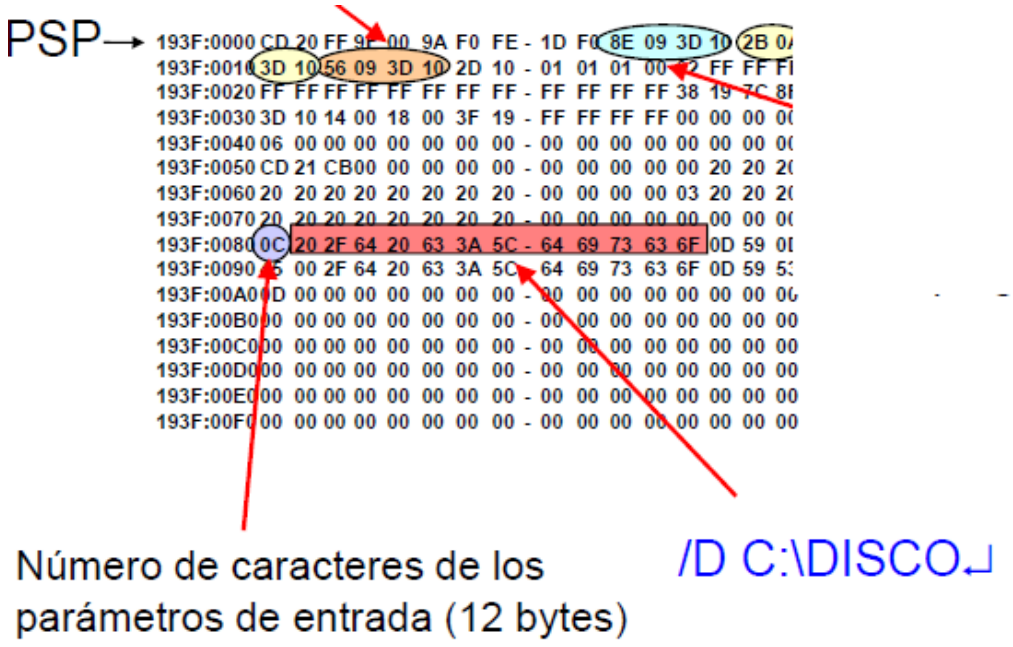
- Dadas las siguientes variables de entorno (comando **SET** de DOS):

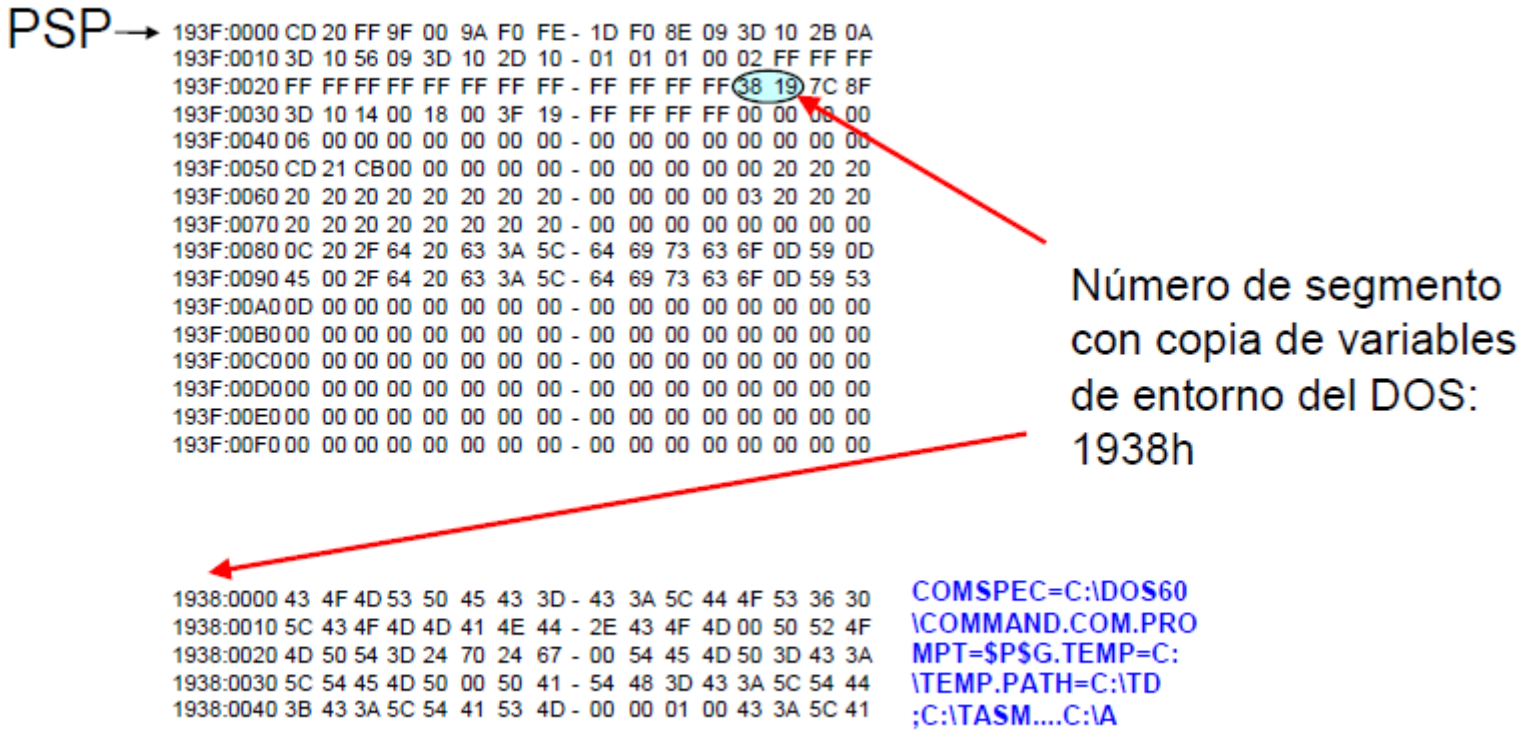
  COMSPEC=C:\DOS60\COMMAND.COM
  PROMPT=PG
  TEMP=C:\TEMP
  PATH=C:\TD;C:\TASM

- Si se ejecuta el programa PROGRAMA con los parámetros /D y C:\DISCO:

  C:\> PROGRAMA /D C:\DISCO

El PSP tendría la siguiente forma:




PSP
193F:0000 CD 20 FF 9F 00 9A F0 FE - 1D F0 8E 09 3D 10 2B 0A
193F:0010 3D 10 56 09 3D 10 2D 10 - 01 01 01 00 02 FF FF FF
193F:0020 FF FF FF FF FF FF FF FF - FF FF FF FF 38 19 7C 8F
193F:0030 3D 10 14 00 18 00 3F 19 - FF FF FF FF 00 00 00 00
193F:0040 06 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
193F:0050 CD 21 CB 00 00 00 00 00 - 00 00 00 00 00 20 20 20
193F:0060 20 20 20 20 20 20 20 20 - 00 00 00 00 03 20 20 20
193F:0070 20 20 20 20 20 20 20 20 - 00 00 00 00 00 00 00 00
193F:0080 0C 20 2F 64 20 63 3A 5C - 64 69 73 63 6F 0D 59 0D
193F:0090 45 00 2F 64 20 63 3A 5C - 64 69 73 63 6F 0D 59 53
193F:00A0 0D 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
193F:00B0 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
193F:00C0 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
193F:00D0 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
193F:00E0 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
193F:00F0 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
Número de segmento con copia de variables de entorno del DOS: 1938h
1938:0000 43 4F 4D 53 50 45 43 3D - 43 3A 5C 44 4F 53 36 30
1938:0010 5C 43 4F 4D 4D 41 4E 44 - 2E 43 4F 4D 00 50 52 4F
1938:0020 4D 50 54 3D 24 70 24 67 - 00 54 45 4D 50 3D 43 3A
1938:0030 5C 54 45 4D 50 00 50 41 - 54 48 3D 43 3A 5C 54 44
1938:0040 3B 43 3A 5C 54 41 53 4D - 00 00 01 00 43 3A 5C 41
COMSPEC=C:\DOS60
\COMMAND.COM.PRO
MPT=PG.TEMP=C:
\TEMP.PATH=C:\TD
;C:\TASM....C:\A

Tres tipos de ficheros ejecutables en DOS:
.BAT comandos del DOS (no código máquina)

.EXE
Son programas en código máquina.
Generados por un montador (*linker*) a partir de uno o varios ficheros de código objeto generados por un compilador o ensamblador.

.COM
Son programas en código máquina.
El programa ocupa un único segmento físico de 64 KB con código, datos y pila.

La primera instrucción ejecutable está en la dirección 256 (100h) respecto al origen del segmento. Se debe usar la directiva **ORG** 256 antes de la primera instrucción de ensamblador.
Se crean a partir de un .EXE con el comando **EXE2BIN** o directamente con la opción **/t** del montador (TLINK).

Ejecución de programas **.EXE**:

CS y **SS** inicializados por el DOS. **DS** y **ES** apuntan al PSP.
IP inicializado con dirección indicada en directiva **END**.
SP inicializado con valor más alto del segmento de pila.
Al acabar el programa se devuelve el control al sistema operativo (intérprete de comandos) y se libera la zona de memoria donde se cargó el programa.

Ejecución de programas **.COM**:

CS, **DS**, **ES** y **SS** apuntan al PSP.
IP se inicializa a 256 (posición siguiente al PSP).
SP se inicializa con 0FFFEh.
Al acabar el programa se devuelve el control al sistema operativo (intérprete de comandos) y se libera la zona de memoria donde se cargó el programa.

Programas residentes (*Terminate & Stay Resident, TSR)*

Programas .COM o .EXE que terminan su ejecución dejando sin liberar parte de la memoria que ocupan.
Su posición en memoria suele almacenarse en forma de vector de interrupción.
Pueden ser llamados desde otros programas en ejecución o desde rutinas de servicio de interrupción.

Programas residentes .COM (*instalación*)

Finalizan con **INT** 27h.
DX debe contener el *offset* de la posición siguiente a la última que se quiere dejar residente.

Constan de dos partes:

La información (código, variables, …) que queda residente.
El código que instala la información.

Programas residentes .COM (*desinstalación*)

Ha de ejecutarse un programa o rutina (desinstalador) que libere la memoria que se dejó residente.
Se libera un segmento físico de memoria mediante **INT**
21h con **AH**=49h y **ES**=número de segmento.

Se deben liberar dos segmentos físicos:

Segmento de código del programa residente (suele guardarse en algún vector de interrupción).
Segmento de variables de entorno (offset 2Ch del PSP).

Antes de liberar un programa es conveniente comprobar que está realmente instalado:

Vector de interrupción distinto de cero
Primeros bytes de la rutina de servicio son los del programa que se desea desinstalar (firma digital del programa).

```
codigo SEGMENT
        ASSUME cs : codigo

        ORG 256

inicio: jmp instalador

; Variables globales
tabla   DB  "abcdf "
flag    DW 0

; Rutina de servicio a la interrupción
rsi PROC FAR
        ; Salva registros modificados
        push ...
        ; Instrucciones de la rutina
        ...

        ; Recupera registros modificados
        pop ...
        iret
rsi ENDP
...
```

```
...
instalador PROC
        mov ax, 0
        mov es, ax
        mov ax, OFFSET rsi
        mov bx, cs
        cli
        mov es:[ 40h*4 ], ax
        mov es:[ 40h*4+2 ], bx
        sti
        mov dx, OFFSET instalador
        int 27h ;  Acaba y deja residente
                ;  PSP, variables y rutina rsi.
instalador ENDP

codigo ENDS
END inicio
```

```
desinstalar_40h PROC          ; Desinstala RSI de INT 40h
      push ax bx cx ds es

      mov cx, 0
      mov ds, cx                    ;  Segmento de vectores interrupción
      mov es, ds:[ 40h*4+2 ]        ;  Lee segmento de RSI
      mov bx, es:[ 2Ch ]            ;  Lee segmento de entorno del PSP de RSI

      mov ah, 49h
      int 21h         ;  Libera segmento de RSI (es)
      mov es, bx
      int 21h         ;  Libera segmento de variables de entorno de RSI

      ;  Pone a cero vector de interrupción 40h
      cli
      mov ds:[ 40h*4 ], cx          ;  cx = 0
      mov ds:[ 40h*4+2 ], cx
      sti

      pop es ds cx bx ax
      ret
desinstalar_40h ENDP
```

RTC Real Time Clock

- IBM incluyó en el PC-AT el chip RTC Real Time Clock (MC146818 de Motorola) alimentado por batería.
- Tecnología CMOS de bajo consumo (idónea para baterías).
- Tiene memoria RAM estática de 64 bytes para la configuración del sistema:
- Puede generar interrupciones periódicas, alarmas y señales hardware.

Lectura del RTC:

- Se escribe (OUT) en el puerto 70h la dirección de la posición que se desea leer.
- Se realiza una lectura (IN) del puerto 71h.

Escritura del RTC:

- Se escribe (OUT) en el puerto 70h la dirección de la posición en la que se desea escribir.
- Se escribe (OUT) en el puerto 71h el valor que se desea escribir.

RTC Real Time Clock

- **El RTC no genera peticiones de interrupción por defecto. Es necesario programarlo para ello.**

Registro A (enviar un valor 0Ah al puerto 70h)

- Leer o escribir sobre el puerto 71h el valor dado por:

UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0

Update_In_Progress (sólo lectura): Cuando está a 0 indica que se puede leer/escribir en los puertos del reloj sin que interfiera con actualizaciones internas.

DV2...DV0	Frecuencia del oscilador
000	4.193404 MHz
001	1.048576 Mhz
010	32.768 kHz

- Cálculo de RS a partir de la frecuencia deseada de interrupciones periódicas del reloj:

$$RS = 1 + \log_2 \frac{32768\ (Hz)}{Frecuencia\ (Hz)}$$

- Ejemplo: si se desea una frecuencia de 512 Hz
  - RS = 7 = 0111b

- **Se usa el registro B para determinar qué evento va a producir la interrupción (alarma, interrupción periódica o cambio de hora). Genera la interrupción 70h y está conectado al IRQ 0 del esclavo.**

Registro B (enviar un valor 0Bh al puerto 70h)

- Leer o escribir sobre el puerto 71h el valor dado por:

SET	PIE	AIE	UIE	SQWE	DM	12/24	DSE

PIE: Se habilitan las interrupciones periódicas.

Ejemplo programacion RTC Habilitacion + frecuencia

```
confRTC PROC FAR
        push ax
        mov al, 0Ah

                ; FIJAR LA FRECUENCIA
        out 70h, al             ; Accede a registro 0Ah
        mov al, 00101110b       ; DV=010b, RS=1110b (14 == 4 Hz)
        out 71h, al             ; Escribe registro 0Ah

                ; ACTIVAR INTERRUPCIONES
        mov al, 0Bh
        out 70h, al             ; Accede a registro 0Bh
        in al, 71h              ; Lee registro 0Bh
        mov ah, al
        or ah, 01000000b        ; Activa PIE
        mov al, 0Bh
        out 70h, al             ; Accede a registro 0Bh
        mov al, ah
        out 71h, al             ; Escribe registro 0Bh

        pop ax
        ret
confRTC ENDP
```

RSI para RTC

- La rutina de atención a la interrupción ha de comprobar si el evento que ha generado la interrupción es el deseado leyendo el registro C.

  Registro C (enviar un valor 0Ch al puerto 70h)
  - Leer (sólo lectura) sobre el puerto 71h el valor dado por:

IRQF	PF	AF	UF	----	----	----	----

  - Cuando están a 1 determinan el tipo de suceso que ha provocado la interrupción.
    - IRQF: Petición de interrupción
    - PF: Interrupción periódica
    - AF: Alarma
    - UF: Actualización de la hora/fecha

- Al final de la rutina se debe mandar el EOI correspondiente a los PICs (8259) esclavo y maestro.

```
mov al, 20h
out 20h, al
out A0h, al
```

Ejemplo RSI para RTC

```
RTC_rsi PROC FAR
                sti
                push ax
                mov al, 0Ch
                out 70h, al      ; Accede a registro 0Ch de RTC
                in al, 71h       ; Lee registro 0Ch de RTC
                …..

          final:                 ; Envía EOIs (RTC)
                mov al, 20h
                out 20h, al      ; Master PIC
                out 0A0h, al     ; Slave PIC
                pop ax
                iret
RTC_rsi ENDP
```