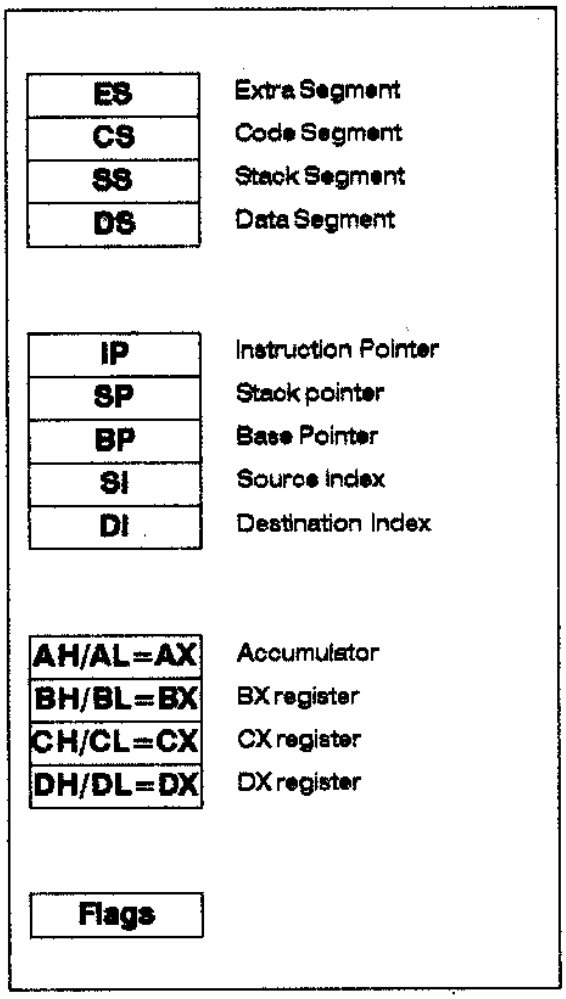
Set de instrucciones 8088/8086

Descripción de los registros internos



En total existen 14 registros de 16 bits. Cada uno de esto registros tiene una función especifica, pero se puede agruparlos por función:

-Registros para datos

-Registros de segmentos

-Registros de direccionamiento

-Registro de estados (flags)

Ya que los registros de direccionamiento no pueden acceder a un bloque de memoria mas grande que 65536 bytes (por su tamaño de 16 bits), los registros de segmento tienen la tarea de 'extender' el rango de acceso. El 8088 o 8086 tiene acceso a 1 MB en total y necesita entonces 20 bits.

El proceso de direccionamiento es entonces el siguiente:

Para acceder a cualquier byte de la memoria, cualquiera que sea la forma, se combina el contenido de dos registros, un registro de segmento y un registro direccionamiento. La combinación se realiza haciendo la suma, desplazando primero el contenido del registro de segmento 4 bits a la izquierda (multiplicación por 16).

Para direccionar la instrucción en el byte número 12345h de la memoria, podemos por ejemplo cargar el registro CS con el valor 1200h y el IP con el valor 0345h:

```
CS      12000h   (desplazado 4 bits)
IP      00345h
        --------
        12345h
```

La referencia a este lugar de la memoria se escribe como: 1200:0345. Existen por supuesto muchas otra formas de generar la misma dirección.

Al lado se ve un ejemplo de lo que hace falta para buscar un byte de instruccion. Cada tipo de movimiento de datos tiene un segmento asignado 'por default'. Por ejemplo, la busqueda de instrucciones siempre se realiza el segmento CS (Code Segment). La busqueda de datos, normalmente utiliza el registro DS (Data Segment). Operaciones con el 'stack' se realizan con el SS (Stack Segment).

Esta asignación automática en algunos casos se puede cambiar utilizando un 'prefijo' antes de la instrucción a ejecutar.

Instrucciones para transferencia de datos

```
MOV     reg8, reg8/mem8
MOV     reg16, reg16/mem16
MOV     reg8/mem8, reg8
MOV     reg16/mem16, reg16
MOV     reg8/mem8, #8
MOV     reg16/mem16, #16
MOV     reg8, #8
MOV     reg16, #16
MOV     AL, mem8
MOV     AX, mem16
MOV     mem8, AL
MOV     mem16, AX
MOV     sreg, reg16/mem16
MOV     reg16/mem16, sreg

PUSH    reg16/mem16
PUSH    reg16
PUSH    sreg

POP     reg16/mem16
POP     reg16
POP     sreg

XCHG    reg8, reg8/mem8
XCHG    reg16, reg16/mem16
XCHG    AL, reg8
XCHG    AX, reg16

IN      AL, port8
IN      AX, port16
IN      AL,[DX]
IN      AX,[DX]

OUT     port8, AL
OUT     port16, AX
OUT     [DX], AL
OUT     [DX], AX

XLAT

LEA     reg16, mem

LDS     mem32
LES     mem32

LAHF
SAHF

PUSHF
POPF
```

La mayoria de las instrucciones del 8088/8086 permite elecolón del tamaño de los datos entre bytes o words. Esta elecolón en muchos casos se realiza automáticamente, por el tamaño de los registro utilizados.

Existen algunas instrucciones que permiten además un movimiento de 32 bits.

Las instrucciones que hacen transferencia de datos están en la lista.

La tabla abajo lista las distintas formas de direccionamiento de los datos. Existen algunas excepciones a esta tabla - una es la instrucción XLAT, que siempre calcula la direcolón con la suma de AL con el registro BX.

Implícitamente, todas las operaciones con la memoria utilizan alguno de los registros de segmentos. En algunos casos, se puede forzar el uso de otro segmento.

Direccionamientos

```
Registro
            MOV     BP,SP

Inmediato
            MOV     BX,1234

Directo
            MOV     AX,addr

Indirecto registro
            MOV     CX,[BX]

Base register
            MOV     AX,[BP+3]

Indexed
            MOV     AX,[SI+3]

Base indexed
            MOV     AX,[BX+DI]

Base indexed with offset
            MOV     AX,[BP+SI+7]
```

Registro	Segmento	Prefijos válidos
IP	CS	Ninguno
SP	SS	Ninguno
BP	SS	DS, ES, CS
SI	DS	ES, SS, CS
DI	DS	ES, SS, CS
DI	ES	Ninguno(*)

(*) En instrucciones de bloques

Instrucciones para stringa.

```
LODSB     AL,DS:[SI]
LODSW     AX,DS:[SI]

STOSB     ES:[DI],AL
STOSW     ES:[DI],AX

MOVSB     byte ptr ES:[DI], byte ptr DS:[SI]
MOVSW     word ptr ES:[DI], word ptr DS:[SI]

CMPSB     byte ptr ES:[DI], byte ptr DS:[SI]
CMPSW     word ptr ES:[DI], byte ptr DS:[SI]

SCASB     AL, ES:[DI]
SCASW     AX, ES:[DI]

INSB      mem8, [DX]
INSW      mem16, [DX]

OUTSB     [DX], mem8
OUTSW     [DX], mem16
```

Prefijos de repetición

REP

REPE/REPZ
REPNE/REPNZ

Estos prefijos pueden utilizarse únicamente con instrucciones que modifican el flag Zero (CMPSB/W y SCASB/W)

A lado se encuentra la tabla de los segmentos que se utilizan por 'Default'.

En la familia 8088/8086 existen instrucciones para el procesamiento de 'stringa' (cadenas o bloques de datos). En realidad de trata aqui de otro prefijo mas que se le puede agregar a algunas instrucciones especialmente concebidas para el manejo de stringa.

Estas instrucciones pueden ser utilizadas sin este prefijo, en cuya caso ejecutan sobre una sola unidad de datos (puede ser byte o word). Todas estas instrucciones tienen en común el uso de registros especializados, y no puede ser cambiado el segmento utilizado en la operación.

En el caso de usar el prefijo REP para la repetición, el registro CX es empleado como 'contador'. La operación se repite CX veces. En el caso de los prefijos REPZ y REPNZ la operación ser repite hasta que ó CX llegue a 0 ó la operación de comparación da el resultado. (REPZ: repetir mientras que la comparación da igual)

En la tabla se muestran los operandos que utiliza cada instrucción. No es necesario incluirlos en un programa, ya que no se pueden modificar. En el caso de incluirlos, se puede escribir el código mnemónico sin la última letra que indica el tamaño del dato (por ejemplo LODS AL, [DI]). El segmento tampoco es necesario, y no se puede modificar. (Algunos assemblers no permiten el segmento en la instrucción e indican error de sintaxis).

Es interesante ver el caso de MOVSB/W. Si unos opta por la inclusión del operando y utiliza la forma abreviada del mnemónico (MOVS), es necesario de indicar de que tamaño de instrucción se trata, ya que ES:[DI] puede indicar bytes o words en la memoria. Esta indicación, para el assembler, es:

byte ptr o word ptr

Estos operadores 'califican' el lugar de memoria como byte o word.

Las instrucciones OUTSB/W son especialmente interesante para la inicialización de periféricos que necesitan una lista de parametros.

Instrucciones aritméticas

ADD	reg, reg/mem
ADD	reg/mem, reg
ADD	reg/mem, immediate
ADD	AX/AL, immediate
ADC	reg, reg/mem
ADC	reg/mem, reg
ADC	reg/mem, immediate
ADC	AX/AL, immediate
INC	reg/mem
INC	reg
AAA	
DAA	
SUB	reg, reg/mem
SUB	reg/mem, reg
SUB	reg/mem, immediate
SUB	AX/AL, immediate
SBB	reg, reg/mem
SBB	reg/mem, reg
SBB	reg/mem, immediate
SBB	AX/AL, immediate
DEC	reg/mem
DEC	reg
NEG	reg/mem
CMP	reg, reg/mem
CMP	reg/mem, reg
CMP	reg/mem, immediate
CMP	AX/AL, immediate
AAS	
DAS	
MUL	reg/mem
IMUL	reg/mem
AAM	
DIV	reg/mem
IDIV	reg/mem
AAD	
CBW	
CWD	

La mayoría de las instrucciones aritméticas son parecidas a sus equivalentes en los procesadores de 8 bits. Estas instrucciones se pueden realizar en 8 ó en 16 bits. Sin embargo, solo los registros AX, BX, CX y DX se dejan subdividir en dos bytes.

Nuevo en comparación con el 8085 es la posibilidad de ejecutar operaciones inmediatas con la memoria, sin tener que utilizar los registros. Es por ejemplo posible de realizar:

```
AND     word ptr 1234h, 40h
```

Las operaciones de incremento y decremento, también pueden realizarse sobre bytes y words.

Nuevos son la división y la multiplicación. Es posible de multiplicacar 8 bits por 8 bits, en cuyo caso siempre AL es un factor y el otro es reg/mem. El resultado se encuentra en AX (16 bits).

En el caso de la multiplicación de 16 x 16 bits, un factor es siempre AX, el otro es reg/mem. El resultado esta en DX:AX (DX la palabra mas significativa). La versión IMUL de la misma instrucción realiza una multiplicación de dos números enteros, con signo.

La división funciona de manera similar a la multiplicación. En el caso de 8 bits, se divide siempre AX (16l) por un valor de 8 bits (reg/mem). AL tiene el resultado de la división, mientras que AH tiene el resto. En el caso de la operación en 16 bits, DS:AX tiene el valor a dividir (32 bits), y se divide por reg/mem, quedando el resultado en AX, y el resto en DX.

Del 8086 se conoce la instrucción DAA (Decimal adjust Accumulator), que permite cálculos con valores en BCD. El 8086/88 agrega unas cuantas variaciones de esta instrucción.

DAA en el 8086 es la abreviación de Decimal Adjust for Addition. En realidad también en el 8085 el DAA funcionaba solamente correcto para una suma. El procedimiento es el siguiente:

Si los 4 bits menos significativos de AL contienen un valor mas grande que 9 o si el AF (Auxiliary Carry) esta en 1, se suman 6 a AL, y AF esta puesto en 1. Luego, si AL es mas grande que 9Fh, ó si el CY (Carry) esta en 1, se suman 60h a AL. De esta forma podemos realizar sumas de números que estan representados en formato BCD. (El AF es el flag que marca, luego de una suma, si hubo un 'carry' entre los dos nibbles sumados. Ej. 59h + 76h no produce AF, 59h + 59h sí)

DAS realiza la misma operación después de una resta.

En realidad, el formato que utilizan las otras instrucciones de corrección (AAA, AAD, AAM y AAS) es aún mas sencillo que BCD. En este caso se trabaja con números sin empaquetar, o sea con un solo dígito en el acumulador (AL).

Si sumamos dos dígitos de este tipo, (valores en 0 y 9), el resultado de una suma puede ser de 0 a 18d (12h). La instrucción AAA (ASCII adjust for Addition) hace lo siguiente:

Si el resultado de una suma de dos bytes con valores entre 0 y 9 da un resultado que en los cuatro bits menos significativos es mas grande que 9, o si el AF esta en 1, se suman 6 a AL, se incrementa AH, y se ponen AF y el CY en 1. El resultado tiene siempre los 4 bits mas significativos en 0.

AAS hace lo equivalente luego de una resta. Si el resultado da un valor mas grande que 9 (solo en los 4 bits menos significativos) o si el AF esta en 1, se restan 6 de AL, de resta 1 de AH, y AF y CY estan en 1.

AAM (ASCII adjust for Multiplication) es un poco diferente: en este caso, AL se divide por 10, y el resultado de la división va a AH. AL recibe el resto (AL mod 10).

AAD hace la operación inversa de AAM: El valor en AH se multiplica por 10 y se suma a AL. AH luego se borra.

CBW convierte bytes a words. Esto es util, porque la operación realiza con valores con signo. O sea, un byte negativo (-1 a -128) se convierte en su equivalente en formato word. CWD hace lo mismo convirtiendo de 16 bits a 32 bits, utilizando el par de registros DX:AX como 'acumulador' de 32 bits.

Instrucciones lógicas

```
NOT     reg/mem

SHL     reg/mem, 1
SHL     reg/mem, CL    (=SAL)
SHR     reg/mem, 1
SHR     reg/mem, CL
SAR     reg/mem, 1
SAR     reg/mem, CL

ROL     reg/mem, 1
ROL     reg/mem, CL
ROR     reg/mem, 1
RCL     reg/mem, 1
RCL     reg/mem, CL
RCR     reg/mem, 1
RCR     reg/mem, CL

AND     reg, reg/mem
AND     reg/mem, reg
AND     reg, immediate
AND     AX/AL, immediate

TEST    reg, reg/mem
TEST    reg, immediate
TEST    AX/AL, immediate

OR      reg, reg/mem
OR      reg/mem, reg
OR      reg, immediate
OR      AX/AL, immediate

XOR     reg, reg/mem
XOR     reg/mem, reg
XOR     reg, immediate
XOR     AX/AL, immediate
```

Nuevamente, la mayoria de las instrucciones son familiares. Todas las instrucciones de pueden ejecutar sobre datos de 8 o de 16 bits.

Particularmente util son las operaciones inversas que permite el 8086/88: realizar una función AND entre un registro y la memoria, pero dejar el resultados de la operación en la memoria, no en el registro como era la norma en los procesadores de 8 bits.

Nuevo también es la posibilidad de realizar las rotaciones y desplazamientos de los registros o de la memoria con mas de 1 bit. En este caso, el registro CL (solo 8 bits) indican la cantidad de bits a 'shiftear'.

Procesadores mas avanzados (80286+) permiten definir directamente en la instrucción el desplazamiento necesario.

TEST, es el equivalente lógico de la operación de prueba CMP en el rango matemático. En lugar de hacer una resta, TEST hace un AND entre los operandos, y deja los 'flags' puestos según el resultado. Ninguno de los operandos se modifica.

Instrucciones de transferencia de control

```
CALL    rel16
CALL    reg16/mem16
CALL    segment:offset
CALL    dword ptr mem
JMP     rel8
JMP     rel16
JMP     reg16/mem16
JMP     segment:offset
JMP     dword ptr mem
RET
RET     data16
RETF
RETF    data16

JE      rel8        (=JZ)
JL      rel8        (=JNGE)
JLE     rel8        (=JNG)
JB      rel8        (=JNAE, =JC)
JBE     rel8        (=JNA)
JP      rel8        (=JPE)
JO      rel8
JS      rel8        (ex-JM)
JNS     rel8        (ex-JP)
JNE     rel8        (=JNZ)
JNL     rel8        (=JGE)
JNLE    rel8        (=JG)
JNB     rel8        (=JAE, =JNC)
JNBE    rel8        (=JA)
JNP     rel8        (=JPO)
JNO     rel8

LOOP    rel8
LOOPE   rel8        (=LOOPZ)
LOOPNE  rel8        (=LOOPNZ)

JCXZ    rel8
```

Nuevo, al 8086, son las instrucciones que transfieren el control ('saltan') con una dirección relativa a la actual. (La familia 6800 de Motorola utiliza esta técnica).

Por ejemplo, la instrucción CALL rel16, contiene un valor de 16 bits que se suma a la dirección que tiene el IP (Instruction Pointer) después de buscar la instrucción CALL. (O sea, la dirección de la instrucción siguiente al CALL). Al tener 16 bits, permite saltos en todos el segmento que esta apuntado por el CS (offsets de -32768 a 32767).

Otra forma de la instrucción permite un salto a la dirección contenido por el reg16 o por la memoria en mem16. En este caso se trata de una dirección física, no un desplazamiento. Nuevamente, el destino se tiene que encontrar dentro del actual segmento.

Para transferir el control a una rutina en otro segmento, podemos utilizar la tercera variación de la instrucción, donde se especifica el valor completa del destino, tanto su segment (que se cargará al CS) como el offset (que va al IP). En este caso, el stack recibe dos 'words': la dirección completa de la instrucción siguiente, para que la rutina en el otro segmento tenga la posibilidad de retornar al originador del llamado.

La cuarta implementación es similar, solo que busca la dirección completa en memoria.

Ya que existen dos grandes formas de transferencia de control, intra-segment y inter-segment, necesitamos dos formas de retorno también: un RET que busca 16 bits en el stack, y un RETF que busca la dirección completa de 32 bits.

Las instrucciones condicionales en el 8086/88 son nuevamente similares a los procesadores de 8 bits, con algunas adiciones. La dirección de destino se calcula con el desplazamiento de 8 bits de la instrucción: no es posible de realizar saltos condicionales a distancias mas grandes de +127 y -128 desde la instrucción que sigue al salto.

La mayoría de los mnemónicos tiene un sinónimo (a veces dos). Esto no quiere decir que el procesador tiene distintas instrucciones; es solo una ayuda para clarificar el programa, brindado por el ensamblador.

Las instrucciones LOOP, LOOPZ, y LOOPNZ son similares a los antes vistos REP, REPZ y REPNZ. Solamente difieren en el hecho que REP repite una sola instrucción, y los 'LOOP' especifican una dirección de 'salto', permitiendo así la repetición de varias instrucciones.

CX implementa un contador de 16 bits. LOOP decrementa CX, y, si CX no es cero todavía, ejecuta el salto al destino especificado. (En algunos procesadores esta instrucción se denomina DJNZ: Decrement and Jump if Not Zero).

LOOPZ y LOOPNZ agregan una variante: igual que el LOOP, decrementan CX, y terminan la repetición cuando CX llega a 0, pero además se controla, antes de ejecutar el salto, el estado del flag 'Z'ero. LOOPZ, entonces, decrementa CX, y salta solamente si CX <> 0 y Z = 1.

Para facilitar determinación del motivo de terminación de un REPZ/NZ o LOOPZ/NZ, de agrega una instrucción condicional especializada, JCXZ.

INT data8
INT3
INTO
IRET
LOCK

Las interrupciones formas una categoria aparte en las instrucciones de transferencia de control. El 8086/88 permite hasta 256 diferentes interrupciones en el sistema. Para este fin, el procesador reserva una tabla de 1024 bytes ubicicada en la dirección 0000:0000h de la memoria. En esta tabla se encuentran 256 direcciones de 32 bits (segment y offset) de las rutinas de servicio.

El periférico puede generar, junto con la iterrupción, un 'vector' en el intervalo que lo pida el procesador, para comunicar que rutina desea ver ejecutada. La mayoria de los periféricos no tienen esta posibilidad y delegan la tarea de la 'identificación' a un integrado especializado, en la PC el 8259.

Interrupciones son llamadas a rutinas tipo inter-segment: la dirección de retorno que se guarda en el stack es de 32 bits. Además, se almacena el registro de flags en el stack automáticamente.

Para poder simular las interrupciones por software, y para aprovechar de otras ventajas que presta este sistema de llamadas a rutinas, se implemento la instrucción INT, que tiene como operando el número de la rutina.

INT3 es una versión comprimida (un solo byte) de esta instrucción que salta a rutina 3, que normalmente se utiliza para 'debugging'.

INTO es un llamado a INT 4 si el flag Overflow esta en 1.

LOCK genera una señal de hardware, y bloquea el uso del bus durante la ejecución de la instrucción siguiente. Esto es para evitar en sistema multi-procesador que ambos procesadores modifiquen simultaneamente el mismo lugar de una memoria compartida.

Control del procesador

```
CLC
SEC
CMC

CLD
SED

CLI
STI

HLT

WAIT
ESC
```

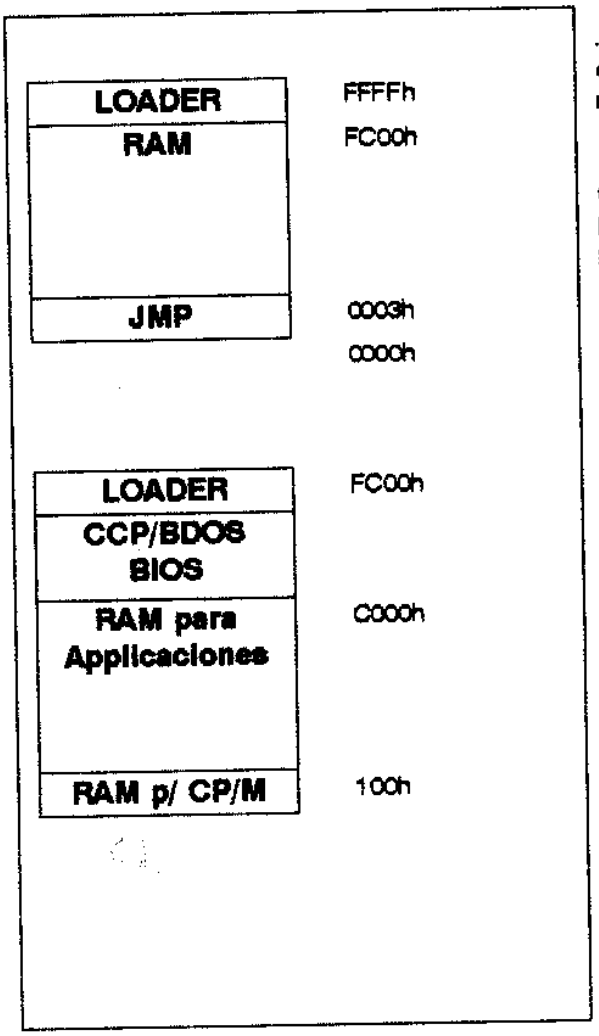
Las instrucciones CLC, SEC y CMC respectivamente borran, ponen en 1, y complementan el CY

CLD indica incremento para las siguientes operaciones de strings (STOS, LODS etc.) SED indica decremento.

CLI deshabilita las interrupciones, STI habilita las insterrupciones.

HLT suspende procesamiento del procesador hasta la aparición de una interrupción externa habilitada, o el reset.

WAIT y ESC son instrucciones que permiten el uso de co-procesadores. ESC con su operando pasa el comando al co-procesador. WAIT espera la terminación del comando. La ejecución asínorona de las instrucciones del co-procesador permite un aumento de la velocidad.



Historicamente, MsDOS heredó el formato .COM de su antecesor CP/M. En realidad MsDOS es más que similar a CP/M, aunque en versiones más recientes recibió muchas funciones nuevas.

Es interesante entonces de ver primero el sistema que utiliza CP/M para la carga de sus programas. CP/M (de Digital Research) fue escrito para procesadores de la familia 8080. Todos los utilitarios originales fueron originalmente diseñados para el set de instrucciones de este procesador. Recién después han aparecido versiones de programas para el Z80.

El proceso de carga del sistema operativo en la memoria en CP/M es el siguiente:

-Al resetear, el 8080 busca su primera instrucción en la dirección 0000h de la memoria. En esta dirección se enouentra un JMP a lo que se llama el 'BOOTSTRAP LOADER', normalmente un programa que reside en una EPROM en la parte superior de la memoria.

-Este programa es el encargado de leer las primeras pistas del disco y transferir el contenido a la memoria justo debajo de la EPROM (también en la parte alta de la memoria).

-El programa cargado es el sistema operativo CP/M que consta de 3 partes:

-BIOS: Basic I/O System, que contiene las rutinas que son especiales para cada sistema diferente. (En el caso de CP/M, es necesario de modificar el sistema operativo en el disco si quiere arrancar con ese disco en otro sistema)

-CCP: Console Command Processor, el programa encargado de hacer la interface entre el operador y el nucleo del sistema operativo. Esta parte interpreta los comandos del usuario y los traduce en comandos para

-BDOS: Basic Disk Operating System. Aqui están las verdaderas rutinas del sistema operativo: rutinas que manejan el directorio, que abren los ficheros en disco, que lean del teclado, escriben a la pantalla etc. Estas rutinas funcionan en 'alto nivel', quiere decir que no se ocupan de saber como funciona la pantalla, o el disco. Las tareas de bajo nivel son para el BIOS.

El CCP entiende una mínima cantidad de instrucciones básica, como DIR, ERA(se), TYPE. Cuando recibe un comando que no conoce, busca en el directorio para un fichero que tenga ese nombre, con la extensión .COM. De encontrarlo, lo carga en el area RAM para aplicaciones (desde 100h) y salta a la dirección 100h, donde el programa tiene que arrancar.

Todas las aplicaciones o programas en CP/M se cargan desde la dirección 100h en la memoria. El SP (Stack Pointer) esta inicializado en un pequeño lugar que prevé CP/M para eso. Si el programa desea utilizar mas espacio, tiene que ocuparse de modificar el SP.

Para que los programas de aplicaciónes no tengan que ocuparse de las características particulares de cada máquina, CP/M brinda la posibilidad de utilizar sus servicios para los accesos a los periféricos. Como CP/M puede cargarse en diferentes lugares, según el hardware de la máquina, se implementó un mecanismo de llamados que es independiente de la ubicación del sistema operativo. (De hecho, en CP/M 3.00, ni siquiera debe estar en los mismos 64 kB que el programa de aplicación, como lo demuestra la C-128).

El siguiente programa para CP/M imprime 'A' en la impresora:

```
MVI     C, 5
MVI     E,41h
CALL    5
```

CP/M, al arrancar, carga en la direcoión 0005h un JMP a su punto de 'servicio' para llamadas externas. Aplicaciones que desean utilizar al BDOS, cargan el registro C con el número de la función a ejecutarse, y prepara los registros según las necesidades de cada funoión.

Esta separación entre las funciones del programa y del sistema operativo hizo de CP/M el primer sistema que permitia de desarrollar programas comerciales que funcionaban en máquinas de distintos fabricantes, y fue la razón de su gran éxito.

Además, los documentos del sistema operativo CP/M describen, paso por paso, como implementar un BIOS para una nueva computadora

La aparición de la PC

IBM cambió el mundo con la fabricación de la PC. En su momento, no elegió un sistema operativo, y Digital Research adaptó su CP/M para la PC, modificando muy poco en el funcionamiento general del sistema. La diferencia principal era de delegar las funciones del BIOS a la EPROM, para independizar aún más el sistema operativo en disco del hardware de cada máquina.

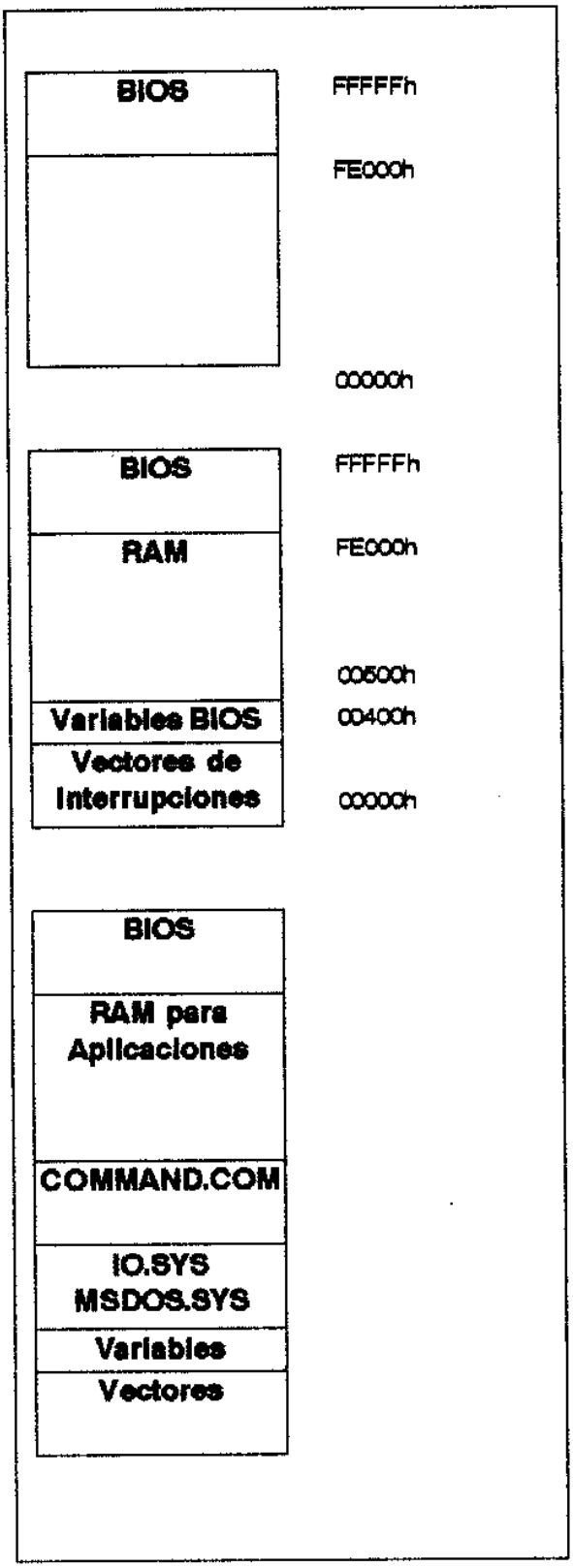
Paralelamente, MicroSoft desarolló el PC-DOS, y luego una versión para computadoras 'compatibles', el MS-DOS.

Las primeras versiones de ambos sistemas operativos estaban pensados en PCs con solo 64kB o 128 kB de memoria, y discos flexibles únicamente. Recién la versión 2.xx permitió ficheros y directorios compatibles con discos grandes.

Mientras que MsDOS era (y sigue siendo) básicamente un sistema operativo para una tarea (no multi-tasking), CP/M86 permite tener hast 4 programas simultaneamente en la memoria. La razón de la elección de MsDOS fue probablemente un interés comercial de IBM en MicroSoft..

El sistema de carga del sistema operativo MsDOS es muy preoido a lo que paso en el caso de CP/M.

La dirección de arranque en los procesadores de la familia 8086 se encuentra en la direccion 0FFFF0h de la memoria, no en 0 como pasaba con el 8080.



En las PC, que tienen el BIOS en la ROM, antes empezar con la carga del programa de disco, se realiza un 'test' de todo el hardware necesario para el correcto funcionamiento. Además, las PC están equipados de un sistema de control de errores en la memoria, y este sistema exige una lectura inicial para preparar los bits de 'parity'. Un segundo 'barrido' controla si todos los lugares de memoria fueron correctamente inicializado.

Se realiza un control sobre las mismas ROM, para controlar si el contenido está sin alterar.

Luego, el test, comprueba la memoria de la pantalla, los periféricos, como el teclado, los discos, controladores de interrupciones, DMA, etc.

El BIOS también inicializa los vectores de interrupción en la parte baja de la memoria, para permitir a programas de aplicación de hacer uso de sus rutinas.

Una vez comprobado el buen funcionamiento de todo el hardware, el BIOS controla si en la memoria alta reside alguna extensión para el BIOS, como por ejemplo una extensión para el manejo de un disco rígido, que en las PCs y XTs no está prevista en el BIOS original. En el caso de detectar la presencia de una ROM extra, pasa el control a este programa (y luego a las otros, de detectarse mas ROMs).

Finalmente, se carga, el BIOS intenta de leer el primer sector de la primera pista del drive A:. Este sector contiene un pequeño programa llamado el 'BOOTSTRAP' que se carga en una dirección variable en la memoria. Este programita se encarga de las características especiales que puede tener este disco, y busca en el directorio dos ficheros, IO.SYS y MSDOS.SYS, en este orden. De estar presentes, los transfiere a la memoria, y transfiere control a ellos (En algunos sistemas, el IO.SYS es el encargado de cargar luego el MSDOS.SYS).

El BIOS residente se encargó de algunos aspectos de la configuración de hardware del sistema. MsDOS da al usuario de configurar algunos parametros luego de cargarse en la memoria, en particular parametros que modifican el comportamiento del sistema operativo. Esta descripción se encuentra en un fichero llamado 'CONFIG.SYS'. Cargado el MSDOS.SYS, este intenta de leer el CONFIG.SYS, y ejecuta los comandos. De encontrar indicaciones especiales, MSDOS busca otro fichero mas, el COMMAND.COM. Y, finalmente, el COMMAND.COM muestra 'A>' en la pantalla..

La similitud con CP/M es evidente. La diferencia más grande es la 'fijación' del BIOS en la ROM de la computadora. IO.SYS con MSDOS.SYS son juntos el BDOS de CP/M, y el COMMAND.COM es el CCP... Los comandos son también muy parecidos, DIR, TYPE, DEL (antes ERA)...

Las PC permiten uso directo de las funciones del BIOS a traves de interrupciones, aunque esta técnica tiene sus peligros, ya que no todos los fabricantes no implementan las funciones exactamente iguales (en la práctica es necesario de utilizar algunas de las funciones del BIOS en forma directa, ya que los de MsDOS son a veces demasiados 'inteligentes').

La téonica aprobada para utilizar las funciones del sistema es a traves de una interrupción especial, 21h. Al llamar esta interrupción, el registro AH contiene el número de la función a ejecutarse. Los otros registros tiene que prepararse según las necesidades de cada función. Similar a CP/M? Más que similar - ya que las primeras 30 o 40 funciones tienen las mismas funciones como en CP/M... En total hay una lista de alrededor de 100 funciones disponibles, con tareas tan diversas como leer el teclado, abrir y escribir ficheros en disco, poner en hora el reloj del sistema.

Carga de programas de aplicación

Como se carga ahora un programa en una PC? Las direcciones bajas están ocupadas por vectores de interrupciones, variables para el BIOS, y el sistema operativo.

El sistema mas sencillo es el caso de programas del tipo .COM. MsDOS, al recibir el comando de carga de un programa de este tipo, busca, inmediatamente arriba del lugar donde reside en memoria, si existe suficiente lugar para la carga de la aplicación. De ser asi, prepara primero 256 bytes con valores, simulando los primeros 256 bytes de CP/M (con el mismo desplazamiento y todo!) y desde ahi carga el programa, que no puede ocupar más de 64 kB, incluido datos. Antes de dar control al programa, MsDOS carga los segmentos con la dirección del bloque de memoria. De esta forma, para la aplicación, todo parece a un ambiente de CP/M...

Rápidamente la limitación de los 64 kB se hizo sentir y la necesidad para un sistema de carga mas flexible resultó en los ficheros del tipo .EXE. El proceso en este caso en mucho más complejo, ya que al ser tan grande, el programa contiene necesariamente instrucciones para modificar los segmentos. Como al cargar el programa en la memoria, estas instrucciones contienen valores que tienen que ser modificados según la ubicación en la memoria, los ficheros .EXE contienen una tabla con todos los lugares en el programa donde hay que hacer modificaciones (Fixups).

Además, el fichero EXE contiene información sobre el largo del programa, número de modificaciones a hacer, espacio de datos que requiere el programa, ubicación y tamaño del segmento para el SP, valor inicial del SP, valor del IP, CS al arrancar...

Este prefijo contiene tipicamente 512 bytes. (siempre múltiplos de 512)

Un programa .COM puede retornar al sistema operativo luego de su ejecución a través de un RET, la forma preferencial para un .EXE es por una llamada a INT 21h, función 4Ch.

Vectores de interrupción

Existen cinco tipos de interrupciones en una PC:

-Interrupciones determinados por el procesador, como por ejemplo la que ocurre al dividir por cero,

-Interrupciones instalados por el BIOS, que sirven a eventos controlados por el controlador 8259, por ejemplo el vector 9, correspondiente a la IRQ1 (en el bus de la PC), generado por el teclado.

-Interrupciones instalados por el BIOS, que cumplen funciones de control sobre los periféricos, por ejemplo vector 10h, que permite a programas de modificar el sistema de pantalla.

-Interrupciones instalados por el sistema operativo, que hacen de interface entre los programas de aplicación y el sistema operativo, por ejemplo vector 21h que es el enlace entre la aplicación y MsDOS para la gran mayoría de las operaciones con ficheros.

-Interrupciones instalados por las aplicaciones, que pueden ser de gran variedad. Un ejemplo es el vector 33h que típicamente es encargado de manejar el mouse.

Interrupciones del procesador

INT 0
INT 1
INT 2
INT 3
INT 4
INT 5
INT 6
INT 7

Estas interrupciones están definidos por por el propio procesador y no pueden cambiarse, aún modificando el 'hardware' del sistema.

INT 0 ocurre cuando el procesador realiza una división por cero. Atención: no es lo mismo cuando por ejemplo en BASIC se divide por cero: en BASIC, o en la mayoría de los casos, la división se realiza por una rutina de división. En este caso no se produce una 'falla' de hardware, sino una detección por software. INT 0 ocurre cuando se divide por cero usando las instrucciones DIV y IDIV del procesador.

INT 1 ocurre cuando el procesador esta en modo 'single step'.

INT 2 contiene la dirección de la rutina que se ejecutará al recibir un NMI (Non-maskable interrupt). Nota: en la PC el NMI en realidad no es incondicional: a través de un circuito externo, es posible de deshabilitar la interrupción. Esta esta asignado principalmente a la detección de errores de paridad en la memoria.

INT 3 es una interrupción especial que puede ser generada por la instrucción INT3. Esta instrucción de un solo byte se utiliza típicamente en programas de diagnósticos, como por ejemplo el DEBUG del mismo MsDOS. Cuando el programa quiere ejecutar una sola instrucción de un programa, reemplaza la instrucción siguiente por un INT3, y modifica el contenido del vector

instrucción siguiente por un INT3, y modifica el contenido del vector correspondiente por la dirección de su propia rutina de servicio.

INT 4 es la interrupcin llamada cuando ocurre un 'overflow' en una operación matemática, utilizando la instrucción condicional INTO (interrupt if overflow).

INT 5 no esta definida en el 8088/8086. En las PC con estos procesadores, esta interrupción esta designada a la función del BIOS para imprimir la pantalla (PrintScreen). En las AT (con 80286 y mayor), INT 5 ocurre cuando aparece un error de BOUNDS, cuando el procesador detecta una infracción de accesso a la memoria protegida. En las AT, para brindar compatibilidad con XT, esta interrupción esta detectada por el BIOS como PrintScreen.

INT 6, también en 80x86, es llamada cuando el procesador detecta una instrucción inválida (no existente).

INT 7, indica (80x86), que el coprocesador no está disponible.

Interrupciones de hardware de PC

Computadoras de la familia PC/XT tienen un solo controlador de interrupciones del tipo 8259. En las AT se agregó otro controlador mas, puesto en 'serie' con el primero.

INT 8

INT 9

INT 0Ah

INT 0Bh

INT 0Ch

INT 0Dh

INT 0Eh

INT 0Fh

INT 8 es la rutina del reloj del sistema. Un timer (8253) esta programado para dividir el reloj del procesador por un número (65536) obteniendo así una interrupción periodica (cada 18.2065 Hz). Ya que este número nada tiene que ver con segundos, la rutina que convierte estos 'ticks' a la hora real es bastante comleja. Contando 18 pulsos por segundo significaria un error de 1%. Por eso la rutina hace correcciones cada minuto (1092 pulsos), cada hora (65543 pulsos) y cada dia (1573040 pulsos). Aún así, el cristal usado para el reloj del procesador normalmente no es de tan buena calidad como para garantizar un padrón confiable. En las AT, otro integrado está encargado de mantener la hora exacta (MC146818), con un cristal de precisión, y con alimentación por bateria. Al arrancar la computadora, el BIOS lee la hora en este chip y lo transfiere a la rutina de INT 8, que desde ahi mantiene el reloj. Por eso, aún con ATs mucho mas rápidos, tiene que tener internamente una forma de generar el clock de 14.318 MHz de la XT. Muchos programas hacen uso de los 'ticks' de 18 Hz.

INT 9. Cada cambio en el teclado, genera una interrupción INT 9, tanto al apretar, como al soltar de la tecla. El teclado manda a la PC un código 'scan' (barrido) que indica físicamente cual tecla cambió de posición. Es la tarea de la rutina de servicio de combinar la información recibida en el valor ASCII correcto (por ejemplo, combinar Shift+letra, o memorizar NumLock).

INT 0Ah en XT se utiliza a veces para un controlador de disco rígido. En las AT, esta interrupción es el punto de conexión para el segundo controlador.

INT 0Bh. Esta linea (IRQ 3) está reservada para interrupciones del canal de comunicaciones RS-232, COM 2. De estar habilitado (normalmente el BIOS No lo utiliza), recibe las llamadas de servicion de un integrado 8250 (versiones mas modernas usan versiones mejoradas en CMOS). En ATs, esta linea puede estar compartida para el COM4. El programa de servicio tiene que determinar, cual de los dos fue el causante de la interrupción, observando los registros internos.

INT 0Ch. Similar a INT 0Bh, pero para COM1 (y COM 3 en las ATs).

INT 0Dh. Rutina de servicio para la interface paralela (impresora) LPT2.

INT 0Eh. Controlador de discos flexibles.

INT 0Fh. Similar a 0Dh, para LPT1.

INT 70h. En ATs, que están equipados con el integrados para reloj de tiempo real MC136818, esta rutina recibe una llamada, si está programada la alarma en este integrado. Esto permite de interrumpir cualquier programa, y ejecutar un procedimiento especial.

INT 71h. Esta interrupción emula la INT 0Ah de la PC/XT para compatibilidad. Las llamadas a IRQ 2 están conectadas aquí.

INT 72h. Está destinado a servir COM 3. Sin embargo en la gran mayoría de los comunicación, no está prevista esta posibilidad (y tampoco en las interfases RS-232), y se utiliza la misma linea (IRQ4) que COM1.

INT 73h. Similar a 72h, para COM 4. Misma observación.

INT 74h. No utilizada.

INT 75h. Interrupción del coprocesador al terminar su operación.

INT 76h. Controlador de discos rígidos en ATs.

INT 77h. No utilizada.

INT 70h

INT 71h

INT 72h

INT 73h

INT 74h

INT 75h

INT 76h

INT 77h

Servicios del BIOS

INT 5, como ya mencionados antes, es la dirección de la rutina que se llama al apretar la combinación de teclas Shift+PrintScreen. Normalmente, en modo texto, para esta rutina se utiliza el código que está en el BIOS. Sin embargo esta rutina tiene limitaciones: No puede imprimir caracteres extendidos, ni tampoco gráficos. Existen una serie de rutinas que reemplazan la función del BIOS y sí prestan estos servicios.

INT 5

INT **10h** es el interrupt que permite el acceso a las rutinas de pantalla. La variedad de opciones que brinda este interrupt es tan grande que hace falta 'subfunciones'. Estas funciones se eligan a través de un número en el registro AH. En los otros registros, se pasan parametros según la subfunción.

INT 10h, **función 0**: seleccionar el modo de operación de la pantalla. Permite seleccionar modo 40x25 caracteres texto, o 640x200 puntos gráfico.

AL=0: 40x25 texto, monocromático.

AL=1: 40x25 texto, color.

AL=2: 80x25 texto, monocromático.

AL=3: 80x25 texto, color.

AL=4: 300x200 puntos gráfico, 4 colores. (CGA)

AL=5: 320x200 puntos gráfico, 2 colores (CGA, monocromático).

AL=6: 640x200 puntos gráfico, 2 colores (CGA, monocromático).

AL=7: 80x25 texto, monocromático (Hercules, texto y 720x348 gráfico)

AL=8: algunas EGA: 132x25 texto.

AL=9...0Ch: reservados.

AL=0Dh: 320x200 gráfico, 16 colores, EGA

AL=0Eh: 640x200 gráfico, monocromático (EGA)

AL=0Fh: 640x350 gráfico, monocromáatico (EGA).

AL=10h: 640x350 gráfico, 16 colores (EGA).

Nota: Adaptadores gráficos de mas resolución (VGA, SVGA) reemplazan la INT 10h, por su propia versión, agregando mas funciones representativas de su capacidades.

INT 10h - **función 1**: programación del tamaño del cursor. Registro CH contiene la linea de arranque, CL la linea final.

INT 10h - **función 2**: posicionar el cursor. BH tiene la página, DH la linea, DL la columna (Atención: primera linea/columna tienen número 0!)

INT 10h - **función 3**: interrogar la posición del cursor. Utiliza los mismos registros que la función 2.

INT 10h - **función 4**: interrogar la posición del lapiz óptico. Devuelve: AH=0 si el lapiz no está activo. AH=1: activo y entonces: DH indica la linea, DL la columna en modo texto. En modo gráfico: CH es la linea, BX columna (modo CGA).

INT 10h - **función 5**: selecciona la página activa en modo texto (AL indica el número de la página).

INT 10h - función 6: desplazar parte de la pantalla hacia arriba. AL contiene la distancia a desplazar en número de líneas. CX las coordenadas del extremo superior izquierda, DX inferior derecha del rectángulo, y BH el atributo a usar.

INT10h - función 7: Idem a función 6, pero hacia abajo.

INT10h - función 8: leer un caracter con atributo en la posición actual del cursor. Al entrar, BH tiene que contener la página. A la salida, AL tendrá el caracter, AH el atributo.

INT10h - función 9: escribir un caracter con atributo en la pantalla, sin cambiar la posición del cursor. AL contiene el caracter, AH el atributo, y CX la cantidad de de veces a repetir la escritura.

INT10h - función 0Ah: escribir un caracter sin atributo a la pantalla, sin modificar la posición actual del cursor. AL contiene el caracter, CX el factor de repetición.

INT10h - función 0Bh: seleccionar el conjunto de colores (palette) para el modo gráfico. BH: número del conjunto (0..127), BL: el color (CGA 320x200)

INT10h - función 0Ch: poner un punto en la pantalla (gráfica). DX tiene la linea, CX la columna, en AL está el color (bits 0..6), y bit 7 de AL indica si la operación es XOR o directa (en EGA: BH contiene la página).

INT10h - función 0Dh: leer el color de un punto (gráfico). Las coordenadas se comunican como en la función 0Ch. El resultado estará en AL.

INT10h - función 0Eh: escribir un caracter en modo teletipo: escribe el caracter y avanza el cursor. AL tiene el caracter, BL el color.

INT10h - función 0Fh: interrogar el modo de video actual. AL recibe el código del modo actual, AH el número de caracteres por linea, BH el número de la página actual.

INT10h - función 10h: selección del conjunto de colores para el controlador de video VGA. Esta función tiene varias 'sub-funciones' seleccionados por el contenido del registro AL.

INT10h - función 11h: Programación del generador de caracteres de la placa EGA. También tiene varias subfunciones.

INT10h - función 12h: Configuración del controlador EGA.

INT10h - función 13h: Escribir en la pantalla EGA.

Las últimas funciones (10..13h) son muy específicas para las ATs y varían entre máquinas y controladores.

INT 11h

El INT11h permite a programas de aplicación de averiguar la configuración de hardware que está implementada. En las XT, el resultado de esta función era la lectura de las llaves en la plaqueta del procesador. En AT, la configuración se realiza mediante un programa del BIOS, se almacena esta información en una RAM dentro del integrado del reloj, y esta función devuelve un valor simulado, compatible con las XT

AX recibe la información de la configuración:

AX bit 0: 1 indica si hay discos flexibles presentes

AX bit 1: no utilizado

AX bit 2+3: cantidad de memoria en la placa principal

AX bit 4+5: modo de video de arranque: 0: inválido, 1: CGA 40x25, 2: CGA 80x25, 3: monocromático, 80x25.

AX bit 6+7: número de discos, si AXb0 = 1

AX bit 8: no utilizado

AX bit 9-11: número de puertos series

AX bit 12: 1 indica controlador de juegos presente

AX bit 13: no utilizado

AX bit 14+15: número de puertos paralelos

INT 12h

INT 12h comunica el tamaño de la memoria de trabajo en kB (via el registro AX).

INT 13h

INT 13h se encarga de las rutinas de servicio para los discos. Este interrupt trabaja nuevamente con funciones comunicadas a traves del registro AH.

INT 13h

INT 13h - función 0: reiniciar el sistema de discos

INT13h - función 1: leer el estado del sistema de discos. El resultado aparecerá en el registro AL: 0: sin errores, 1: comando inválido para el controlador, 2: marca de direccionamiento no encontrado, 3: protección contra escritura, 4: sector no encontrado, 8: error de DMA, 9: error de segmento en DMA, 16: error de CRC, 32: error en el controlador (765), 64: error de busqueda de pista, 128: error por exceso de tiempo.

INT 13h - función 2: leer sectores, Datos a proveer: DL: número del drive (0=A), DH: cabezal, DL: pista, CL: sector inicial, AL: número de sectores a leer, ES:BX: dirección de la memoria donde almacenar el resultado.

INT13h - función 3: escribir, parámetros como en función 2.

INT13h - función 4: verificar, parámetros como en función 2.

INT13h - función 5: formatear sectores, parámetros como en función 2, salvo que ES:BX apunta a la tabla de formateo; AL y CL ignorados.

Existen varias otras funciones de esta interrupción. En algunos casos se puede utilizar para leer/escribir el disco rígido, indicando el número del drive sumando 128 ($80=C). Parece haber variaciones entre distintas implementaciones.

INT 14h

INT 14h se encarga de las comunicaciones por RS-232. En su forma mas básica, permiten enviar y recibir caracteres sin utilización de interrupciones. Esto severamente limita la velocidad máxima alcanzable. Recién con sistema operativos mas actualizados, es posible de cargar un módulo residente que se encarga de flexibilizar las comunicaciones.

INT 14h - función 0: Inicializar el puerto de comunicaciones. Se llama con AH = 0, DX: el número del puerto de comunicación (0=COM1, etc.) y AL indicando velocidad, paridad, número de bits de datos y de terminación:

bits 7, 6, 5	bits 4, 3	bit 2	bits 1, 0
Velocidad (Bauds)	Paridad	Bits de terminación	Bits de datos
000: 110 Baud	x0: sin paridad	0: 1 bit	10: 7 bits
001: 150 Baud	01: paridad impar	1: 2 bits	11: 8 bits
010: 300 Baud	11: paridad par		
011: 600 Baud			
100: 1200 Baud			
101: 2400 Baud			
110: 4800 Baud			
111: 9600 Baud			

INT 14h - función 1: Enviar un caracter por la linea RS-232. AL contiene el caracter por enviarse, DX el número del puerto.

INT 14h - función 2: Recibir un caracter de la interfase serie. Llamar con DX = número del puerto. AL contiene el caracter recibido si bit 7 de AH = 0.

INT 14h - función 3: Interrogar el estado de la interfase. DX: número del puerto a interrogar. AH recibe el estado:

bit 7: exceso de tiempo

bit 6: registro de desplazamiento de transmisión vacio

bit 5: registro intermedio de transmisión vacio

bit 4: BREAK detectado

bit 3: error de formato

bit 2: error de paridad

bit 1: error: perdida de caracteres recibidos

bit 0: datos recibidos.

Además, AL recibe el estado de las lineas de control para el modem:

bit 7: señal detectado en linea de recepción (**CD**)

bit 6: indicador de llamado (**RI**)

bit 5: Data set ready (**DSR**)

bit 4: Clear to Send (**CTS**)

bit 3: Cambio en CD detectado

bit 2: Flanco descendente del indicador de llamado (RI)

bit 1: Cambio detectado en DSR

bit 0: Cambio detectado en CTS

INT 15h

INT 15 fue originalmente el manejo del casete de audio conectado a la PC. Actualmente cumple una variedad de funciones, que puede diferir entre diferentes máquinas. En ATs, funciones **80h..86h** están utilizados para realizar tareas como interrogar joystick, esperar un evento, etc. **87h..89h** son funciones para acceder a la memoria EMS. Varias funciones mas de esta INT no están documentados. Es conveniente tener mucho cuidado en el uso!

INT 16h

INT 16 provee la interfase con el teclado.

INT 16h - función 0: Esperar y recibir un caracter del teclado. Devuelve el caracter en ASCII en AL, y también el código de barrido en AH.

INT 16h - función 1: Interrogar el estado del teclado. Devuelve el flag Z=1 si no hay tecla activada. Z=0 indica un caracter esperando, y en este caso AL=ASCII, y AH=código. Esta función NO descarga el caracter del buffer de teclado.

INT 16h - función 2: Interrogar el estado de las teclas especiales. AL retorna el estado:

bit 7: Estado del modo de inserción

bit 6: Mayúsculas

bit 5: Modo número

bit 4: Bloqueo de pantalla (Scroll Lock)

bit 3: Estado actual de la tecla de inserción

bit 2: Estado de la tecla de control

bit 1: Estado de tecla de Shift (izquierda)

bit 0: Estado de tecla de Shift (derecha)

INT 17h

INT 17h está encargado de los servicios de la impresora. La ventaja de está INT es la completa transparencia: no interpreta caracteres especiales como lo hacen las rutinas del sistema operativo (especialmente 1Ah, Ctrl_Z).

INT 17h - función 0: Enviar un caracter a la impresora. AL contiene el caracter a enviar, DX el número del puerto (0=LPT1). AH recibe el estado del puerto:

bit 7: Impresora no ocupada (-BUSY)

bit 6: Confirmación (ACK)

bit 5: Sin papel (Paper Out)

bit 4: Impresora seleccionada (SELECT)

bit 3: Error de E/S (I/O Error)

bit 2: no utilizado

bit 1: no utilizado

bit 0: Exceso de tiempo (Time out)

INT 17h - función 1: Reinicializar puerto de impresora. Se llama con DX=número del puerto. Retorna el estado de la interfase (como arriba). Es improbable que sea necesario de llamar esta función, ya que el BIOS y sistema operativo se encargan de prepararlo.

INT 17h - función 2: Interrogar el estado de la interfase. Devuelve el estado del puerto de impresora indicado en (DX), según la tabla arriba.

INT 18h

INT **18h**, en las primeras PC, arrancaba el BASIC en ROM. Este BASIC solo esta(ba) presente en PC originales de IBM.

INT 19h

INT 19h: 'BOOTSTRAP'. Carga el sector 1 de la pista 0 en la dirección 0000:7C00h de la RAM, y reinicia la carga del sistema operativo.

INT 1Ah

INT 1Ah: Reloj del sistema: funciones **0** y **1** leen y escriben respectivamente el reloj (mantenido por Timer 0). Registros: CX:DX contienen un contador de 32 bits, incrementado 18.3 veces por segundo. Si AL>0, cambió el dia desde la última lectura.

Funciones 2 y 3: leer y escribir el reloj de tiempo real (con bateria) en las AT. Funciones **4** y **5** leen y escriben la fecha, función **6** programa la hora de la alarma, y **7** reinicia la alarma.

INT 1Bh

INT 1Bh: Vector de Ctrl-Break. Al apretar la combinación Ctrl-BRK, se ejecutará la rutina cuya dirección se encuentra en este vector.

INT 1Ch

INT 1Ch: Interrupción de Timer. La rutina cuya dirección se encuentra aquí, se llamará 18.3 veces por segundo.

INT 1Dh

INT 1Eh

INT 1Fh

INT 1Dh, **1Eh**, y **1Fh** contienen las direcciones de los parámetros de pantalla, parámetros de disco (en uso) y la tabla de caracteres gráficos, respectivamente.

INT 20h

INT 20h es una de varias formas que existen para terminar un programa y devolver el control al sistema operativo que lo llamó. Es el equivalente de la función 00h de CP/M, y fue la manera más utilizada en las versiones antiguas de MsDOS. Sin embargo, desde la versión 2 en adelante es preferible utilizar la función 31h ó 4Ch de la interrupción 21h.

INT 21h

INT 21h es seguramente la más utilizada, y más compleja de todas las interrupciones que están a nuestra disposición. Inicialmente esta insterrupción simulaba las funciones del BDOS de CP/M. Luego se agregaron un cantidad importante de nuevas posibilidades.

En lugar de enumerar las funciones, las agruparemos aqui por categoría, para facilitar su uso.

Terminación de programas

INT 21h/00h

INT 21h/31h

INT 21h/4Ch

INT 21h-función 0h es el equivalente de la interrupción 20h. Usando esta función un programa comunica al sistema operativo que terminó su operación y que ya no necesita la memoria que ocupa. Además se ocupa de varias tareas 'administrativas', necesarias para preparar el sistema para el programa siguiente.

Llamar con AH = 0 y CS = la dirección del PSP (Program Segment Prefix). Similar a las interrupciones anteriores, AH tiene que tener siempre el número de la función. En lo que sigue lo tomamos como norma.

Int 21h/31h informa al sistema operativo, que el programa terminó su ejecución, pero que quiere seguir residente en la memoria. MsDOS no libera en este caso la memoria. Esta función permite de escribir programas 'residentes' que, por ejemplo, se instalan para servir a interrupciones por hardware. Otra posibilidad es la instalación de utilitarios activados por combinaciones de teclas especiales.

Al llamar la función, AL puede tener un código de terminación, pero DX tiene que tener el tamaño del bloque de memoria que queda residente, en parafos (bloques de 16 bytes). Ya que normalmente los programas que se instalan residentes constan de dos partes (una parte instaladora, y la parte residente), eso nos da la posibilidad de reservar solamente lugar para la parte residente, y devolver lo demas.

INT 21h/4Ch es la forma preferida para terminar programas. Además de dar la posibilidad de volver al sistema operativo en sí, da la posibilidad de volver a un programa 'pariente' que originó la ejecución. Esta función es la única que no cuenta con el contenido de los registros de segmentos en el momento de la terminación, lo que es particularmente importante en el caso de programas .EXE que modifican normalmente estos registros.

AL puede comunicar un código de terminación al programa pariente o al sistema operativo. Este código puede ser probado en BATCH files usando la variable ERRORLEVEL

Entrada de caracteres

INT 21 h/01 h es la entrada de caracteres del teclado con echo a la pantalla. De estar habilitado, esta rutina detecta la activación de la combinación Ctrl-C o Ctrl-Break y ejecuta un interrupt 23h en este caso. Para leer teclas expandidas (por ejemplo, movimiento de cursor), hay que llamar esta función una segunda vez cuando devuelve 0 la primera.

INT 21 h/01 h

INT 21 h/03h

INT 21 h/06h

Esta función espera hasta la activación de una tecla, y devuelve el resultado en AL.

INT 21 h/07h

INT 21 h/03h espera y lee un caracter del canal denominado 'AUX'. Este canal normalmente esta asignado al COM1 (puede ser redireccionado mediante el comando MODE de MsDOS). Esta función normalmente no utiliza interrupciones por hardware, y por eso puede perder datos si recibe información demasiado rápido.

INT 21 h/08h

INT 21 h/0Ah

INT 21 h/06h (denominada 'Direct Console I/O'). Esta función da posibilidad de leer caracteres del teclado o escribir en la pantalla. No espera si no hay un caracter disponible. No hace interpretación de la información, i.e. no reacciona ante un Ctrl-C o Ctrl-Break.

INT 21 h/0Bh

INT 21 h/0Ch

DL=0FFh indica requerimiento de entrada. Devuelve Z=1 cuando no hay caracter disponible, o Z=0 y AL=caracter leido.

DL=otro valor (0...0FEh) transmite este caracter a la pantalla. (Atención: no puede ser utilizado como salida universal, ya que 0FFh tiene significado especial)

INT 21 h/07h es la entrada del teclado sin echo y sin filtrado (interpretación). El caracter leido está en AL.

INT 21 h/08h es similar a 07h, pero con la detección de Ctrl-C y Ctrl-Break.

INT21 h/10h permite la entrada de caracteres, utilizando el editor de linea que esta incluido en MsDOS para corregir las lineas de comando. El programa que llama esta función tiene que prever un 'buffer' para el almacenamiento de la linea. Además, este buffer contiene, en el primer byte el largo máximo de la linea. En el segundo byte, MsDOS devuelve el largo actual de la linea tipeada. Recién en el tercer byte esta la información en sí.

Al llamar la función, DS:DX tienen segmento y offset del buffer.

Normalmente están disponibles los comandos de edición de MsDOS como F1, F2, F3, Rubout etc. Si tenemos instalado algun utilitario como DOSEDIT o CED, estos reemplazan esta interrupción, y automáticamente se hacen disponibles a nuestro programa.

INT 21 h/0Bh controla la presente de una tecla apretada. Devuelve AL = 0 si no hay datos, =0FFh si hay.

INT 21 h/0Ch borra el buffer de teclado y luego llama a una de las otras rutinas de entrada (AL comunica el número de la función a llamar).

Salida de caracteres

INT 21 h/02h transmite el caracter en DL a la consola. Si DL contiene un backspace (08h), el cursor retrocede una posición, y se mantiene ahí.

INT 21 h/02h

Si el usuario apretó Ctrl-C o Ctrl-Break, se detecta esta situación después de la salida a pantalla y llama a INT23h.

INT 21 h/04h

INT 21 h/04h es la salida al canal 'AUX', normalmente asignado a COM1. DL tiene el caracter a transmitir.

INT 21 h/05h

INT 21 h/05h transmite el caracter en DL a la salida de la impresora (PRN o LPT1). Si la impresora no esta disponible (no 'on line' o falta papel), esta rutina espera hasta que pueda imprimir.

INT 21 h/06h

INT 21 h/06h permite salida directa a la pantalla (ver entrada de caracteres).

INT 21 h/09h

INT 21 h/09h envia una cadena de caracteres, terminados por el caracter ASCII '$' a la consola. (DS:DX apuntan al principio de la linea)

Control del disco

Estas funciones permite de obtener información de y el control sobre el sistema de discos de la computadora.

INT 21 h/0Dh

INT 21 h/0Dh reinicializa el controlador de discos. También termina de escribir la información de eventuales ficheros abiertos en disco. Sin embargo, no pone al día el directorio, asi que puede ser inaccesible la información escrita (genera error cuando se corre el utilitario del DOS CHKDSK).

INT 21 h/0Eh

INT 21 h/19h

En MsDOS las operaciones a disco sin referencia explicita en el nombre del fichero, van a un drive denominado 'default drive'. Con **INT 21 h/0Eh** se puede modificar esta selección. Al llamar la rutina DL tiene que tener el número del drive (0=A, 1=B etc.)

INT 21 h/1Bh

Para obtener el número del drive actualmente designado como 'default', podemos utilizar **INT 21 h/19h**. AL obtendrá la designación.

INT 21 h/1Bh nos provee de la estructura del disco actualmente en uso.

Al volver de la rutina, AL tiene la cantidad de sectores/cluster, CX tiene el tamaño de un sector (en bytes) y DX la cantidad de clusters en el disco. DS:DX apunta a un byte que indica el tipo de disco en uso (llamado 'media byte').

0FFh: Doble lado/ 8 sectores por pista/ floppy

0FEh: Simple lado/ 8 sectores por pista/ floppy

0FDh: Doble lado/ 9 sectores por pista/ floppy

0FCh: Simple lado/ 9 sectores por pista/ floppy

0F9h: Doble lado/ 15 sectores por pista/ floppy

0F8h: Disco rígido/ todos los tamaños

Este 'media byte' se encuentra en todos los discos como primer byte del FAT (file allocation table)

La función **INT 21h/1Ch** es similar a la anterior, excepto que tiene la posibilidad de especificar el drive para el cual se requiere la información (en DL, donde 0= default drive, 1= A, 2= B etc.).

INT 21h/1Ch

Normalmente, el sistema operativo no controla lo escrito en el disco (salvo durante el formateo). **INT 21h/2Eh** permite aumentar la confiabilidad, obligando a MsDOS de realizar una lectura de control luego de cada escritura. (AL=0 deshabilita, AL=1 habilita el control).

INT 21h/2Eh

INT 21h/36h

INT 21h/36h nos provee de una lista de datos que permite de calcular el espacio libre que queda en el disco designado por DL (0= default). (AL=sectores/cluster, BX=número de clusters disponibles, CX: bytes por sector, DX: clusters por disco). Si el disco requerido no existe, AX es = 0FFFFh.

INT 21h/54h permite de obtener el flag de verificación (ver INT21/2Eh)

INT 21h/54h

INT 21h/58h accede al método que utiliza MsDOS para la asignación del espacio libre en la memoria (allocation strategy). Si AL=0, la función nos comunica el método actualmente en uso. Si AL=1, entonces BX indica el método que queremos implementar. Actualmente son válidos:

INT 21h/58h

BX=0: 'First fit' utiliza el primer espacio disponible

BX=1: 'Best fit' busca el espacio mas pequeño, pero que todavia puede cumplir con el requerimiento.

BX=2: 'Last fit' asigna el último bloque disponible.

Operaciones con ficheros

Por herencia de CP/M, las primeras funciones para acceder a ficheros parecen poco elegantes. Existen funciones mas recientes que son mas fáciles de usar.

INT 21h/0Fh

INT 21h/0Fh es la función que 'abre' un fichero en el disco para lectura o escritura. El fichero tiene que existir. Para crear un fichero se utiliza la función 16h. DS:DX tiene apuntar a un bloque de memoria que contiene el FCB (File Control Block). El file control block (37 bytes) tiene que estar parcialmente preparado con el nombre del fichero. MsDOS se encarga de inicializar otros campos.

INT 21h/10h

INT 21h/10h cierra el fichero. Si quedan datos en la memoria que todavia no fueron escritos en disco, esta operación se hace. Luego, los datos actualizados son entrados en el directorio.

El **INT 21h/11h** nos permite realizar una búsqueda en el directorio del disco. DS:DX tiene que apuntar a un FCB que contiene una especificación del fichero que queremos encontrar. Este nombre puede contener el caracter '?'. '?' compara igual con cualquier caracter. De haber varios ficheros que cumplen con la comparación, la función devolverá la primera ocurrencia. Luego podemos llamar a **INT 21h/12h** para encontrar la próxima entrada que cumple. Si la operación (tanto en 11h como en 12h) fue exitosa, AL vuelve 0, sino AL es 0FFh.

INT 21h/11h

INT 21h/12h

INT 21h/13h

INT 21h/16h

INT 21h/17h

INT 21h/13h nos permite borrar ficheros en el disco. Nuevamente DS:DX tiene que apuntar un FCB, y el nombre del fichero puede contener '?'. AL será 0 si la operación fue exitosa, o 0FFh si el fichero no se encontró.

INT 21h/16h crea un fichero nuevo en el directorio - si es posible. DS:DX indica el FCB a usar. '?' no esta permitido. AL vuelve 0 si la operación fue correcta, o 0FFh si el directorio estaba lleno.

INT 21h/17h permite de cambiar el nombre de un fichero en el directorio. Al entrar DS:DX tiene que apuntar a un FCB 'especial' que contiene dos nombres. Esta permitido de utilizar '?' en el nombre original, en cuyo caso todos los ficheros serán renombrados que comparan correctamente. También esta permitido de utilizar '?' en el nombre nuevo. En los lugares donde se encuentra el '?', se mantendrán las mismas letras que el nombre original.

En CP/M no habia un control exacto posible sobre el tamaño del fichero, utilizando únicamente los datos del directorio. El fin de un fichero de texto estaba indicado por un Ctrl-Z al final del texto. En el caso de ficheros binarios, que pueden tener Ctrl-Z como dato, el tamaño se determina por los datos del directorio, pero siempre en multiplos de 128 bytes.

INT 21h/23h

INT 21h/3Ch

INT 21h/3Dh

INT 21h/3Eh

INT 21h/23h data de esta época, y devuelve el tamaño de un fichero en 'records' de 128 bytes. (DS:DX apunta un FCB). Versiones mas nuevas del DOS utilizan otra función (42h) para determinar el tamaño con un byte de precisión.

INT 21h/3Ch es la versión avanzada para crear un fichero nuevo en el disco. DS:DX tiene que apuntar un lugar en la memoria que contiene la especificación del fichero, en forma de cadena de caracteres, terminados por un cero. (ASCIIZ string). En contraste a la función antigua, esta permite especificar directorios en el nombre del fichero. Además CX pueden tener un número indicando el tipo de fichero (0= normal, 1=lectura solo, 2= invisible (hidden), 3=sistema).

Esta función pertenece a la nueva familia de funciones que acostumbran de indicar el éxito o no de la operación mediante el CY. CY=0 indica ejecución correcta, CY=1 error.

En este caso, CY=0 indica que el fichero fue creado, y AX devuelve el 'file handle', un número de 16 bits que nos permite de indicar el fichero para operaciones subsiguientes.

INT 21h/3Dh funciona similar a la anterior, pero exige que el fichero ya existe (open=abrir un fichero). Aquí tenemos que indicar el tipo de acceso que deseamos realizar sobre el fichero. (AL= 0: lectura, AL= 1 escritura, AL= 2 lectura/escritura). Desde la versión 3.xx en adelante existe la posibilidad de definir aún mas modos de apertura, para permitir acceso múltiple y otros modos.

INT 21h/3Eh cierra nuevemente el fichero, utilizando el 'handle' obtenido de 3Dh o 3Ch.

INT 21h/41h es la versión actualizada para borrar ficheros. DS:DX nuevamente apunta un 'string' con la especificación del fichero a borrar (o ficheros). Ahora también puede contener '*', comparando igual con cualquier combinación de letras que sigue.

INT 21h/43h permite de manipular los atributos en el directorio. Si AL= 0 al llamar, esta función devuelve los atributos del fichero indicado. Si AL=1, CX tiene que tener los atributos nuevos. (bit 0= lectura solo, 1 = invisible, 2=sistema, 5=archivado).

INT 21h/46h recibe en BX el handle de un fichero actualmente en uso, y devuelve un nuevo handle, apuntando al mismo fichero, en la misma posición. No es muy clara la finalidad, pero un uso util de esta función es el siguiente: Se pide la duplicación del handle, y luego, con la función 3Eh, cierra el nuevo fichero. De esta forma, MsDOS pone al dia el fichero y la entrada en el directorio, sin tener que cerrar el fichero original.

INT 21h/46h hace apuntar a dos 'handles' al mismo fichero en la misma posición. Al entrar, BX tiene el primer handle, CX el segundo. Si CX apuntaba un fichero abierto, este se cierra primero. La función permite redireccionar datos a otro fichero.

INT 21h/4Eh y /4Fh realizan la busqueda en el directorio para la primera ocurencia y ocurencias subsiguientes, respectivamente.

INT 21h/56h permite renombrar ficheros. DS:CX apunta el nombre ASCIIZ actual del fichero, y ES:DI el nombre nuevo.

INT 21h/57h nos da acceso a la fecha y hora que estan grabados juntos con el nombre de cada fichero en el directorio. Si AL= 0, la función interroga el directorio, si AL= 1, el directorio será modificado con los datos que recibe la función en CX/DX. En ambos casos BX tiene que tener el handle del fichero.

CX: hora, bits 0 a 4: segundos/2, bits 5 a 10: minutos, bits 11 a 15: hora.

DX: fecha, bits 0 a 4: día, bits 5 a 8: mes, bits 9 a 15: año. El año es un número binario de 0 a 127, contando desde el año 1980. (0=1980, 127=2107, año que MsDOS seguro será obsoleto...)

INT 21h/41h

INT 21h/43h

INT 21h/45h

INT 21h/46h

INT 21h/4Eh

INT 21h/4Fh

INT 21h/56h

INT 21h/57h

Operaciones de escritura/lectura ficheros

Nuevamente tenemos aquí dos tipos de rutinas: las que se manejan con FCB y las mas recientes que se manejan con el file handle.

INT 21h/14h

INT 21h/15h

INT 21h/14h y **/15h** son las operaciones de lectura y escritura secuencial en el disco. DS:DX indica el FCB. El fichero tiene que estar abierto antes de intentar la operación.

INT 21h/21h

INT 21h/22h

INT 21h/24h

INT 21h/21h y **/22h** permite lectura y escritura con acceso aleatorio al fichero. Manejando bytes especiales en el FCB, se puede cambiar el lugar de la operación en el fichero, y el tamaño del bloque a transferir. **INT 21h/24h** permite de especificar el número del record que queremos leer en el fichero.

INT 21h/27h

INT 21h/28h

Funciones **INT 21h/27h** y **/28h** son similares a 21h y 22h, salvo que permiten lectura y/o escritura de múltiples bloques en una sola operación. CX indica el número de bloques o records a leer.

La nueva generación de rutinas generaliza el concepto de 'ficheros' e incluye periféricos en la misma familia. Todos los periféricos en la PC recibe un nombre simbólico (ej. COM1, LPT2 etc.)

INT 21h/3Fh

Como mencionado anteriormente, esta funciones utilizan el file handle para indicar el fichero. (en BX).

INT 21h/40h

INT 21h/3Fh y **/40h** son las operaciones de lectura y escritura respectivamente. CX tiene la cantidad de bytes a transferir, y DS:DX apunta al lugar de memoria que recibirá (en lectura) o contiene los datos a escribir.

INT 21h/42h

INT 21h/42h modifica el puntero del fichero. Cada fichero abierto tiene en su bloque de control (invisible para el usuario) un puntero que mantiene la posición donde se realizó la última operación. Al entrar, CX:DX tiene la nueva posición que será transferida al puntero (32 bits). Además, es posible de especificar en AL el modo: 0: desplazamiento desde el principio del fichero, 1: desplazamiento relativo a la posición actual, 2: Desplazamiento desde el final del fichero).

INT 21h/5Ch

En sistemas de multitarea o en redes, la **INT 21h/5Ch** permite el control de acceso a ciertas areas de un fichero. AL=0 indica impedir el acceso, =1: permitir acceso. BX, como siempre, el handle. CX:DX indica el inicio de la zona, SI:DI el final de la zona de protección.

Operaciones con directorios

Las operaciones con subdirectorios son mas recientes, ya que en CP/M no estaban disponibles.

INT 21h/39h

INT 21h/39h y **/3Ah** permiten crear y borrar subdirectorios, respectivamente. DS:DX apunta un string ASCIIZ, indicando el camino (path) del nuevo directorio a crear o el directorio a borrar.

INT 21h/3Ah

INT 21h/3Bh

La función 3Ah no permite remover el subdirectorio si este es el actual, o si todavia contiene ficheros.

INT 21h/47h

INT 21h/3Bh selecciona el directorio activo. DS:DX indica el ASCIIZ string con la especificación del directorio.

INT 21h/47h obtiene del sistema operativo la especificación completa del directorio actual. Aqui DL tiene que tener la designación del drive (0= default, 1 = A). DS:SI apunta un espacio en la memoria que recibirá la especificación completa del directorio actual en ese drive. (desde la raiz)

Area de transferencia de datos

Las operaciones con FCB (lectura y escritura) utilizan como destino o fuente de la transferencia un area de memoria llamado Disk Transfer Area (DTA). Esta area normalmente se asigna 'por default' al arrancar cualquier programa. (En el caso de programas .COM, en el area desde 0080h hasta 00FFh). El tamaño del espacio previsto es 128 bytes (herencia de CP/M). Para permitir lectura y escritura de bloques de información mas grandes, es posible de reubicar este bloque a traves de la función **INT 21h/1Ah**. (DS:DX apunta al 'buffer')

INT 21h/1Ah

INT 21h/2Fh

INT 21h/2Fh permite interrogar al sistema operativo por la direcolón del actual buffer.

Hora y fecha del sistema

CP/M no tenia previsto la administración de un sistema con hora o fecha hasta la versión 3.xx (C-128 por ejemplo). Estas funciones son relativamente nuevas.

INT 21h/2Ah

INT 21h/2Ah obtiene la fecha actual del sistema mientras que **/2Bh** la programa. En ambos casos los registros utilizados son:

INT 21h/2Bh

CX: año (válido entre 1980 y 2100)

DH: mes (1...12)

DL: dia (1...31)

Ademas, AL obtiene el día de la semana (0=domingo).

INT 21h/2Ch y **/2Dh** controlan en forma similar al reloj interno. (/2Ch interroga, /2Dh setea).

INT 21h/2Ch

CH: hora (0..23)

INT 21h/2Dh

CL: minutos (0...59)

DH: segundos (0...59)

DL: centecimas de segundos (0...99)

En la mayoria de las computadoras, el reloj interno no tiene resolución para determinar centecimas de segundos. (normalmente resuuelve hasta 1/16.2 segundos).

Alocación dinámica de la memoria

Las máquinas con 8080 o Z-80 y CP/M con sus 64kB máximo de memoria no permitian múltiples ficheros en memoria simultaneamente. En las PC, si es posible de tener varios programas y/o bloques de datos al mismo tiempo activos. Una aplicación puede solicitar al sistema operativo mas memoria de lo que le fue asignado automáticamente al arrancar.

INT 21h/48h solicita a MsDOS la asignación de un bloque de memoria. BX tiene el tamaño del bloque requerido, y al volver con éxito, AL tendrá el segmento del principio del bloque.

INT 21h/48h

Una vez terminado el proceso, hay que 'devolver' la memoria utilizada al sistema, a traves de **INT 21h/49h**. Al llamar la función, ES tiene que tener el segmento del bloque. No hace falta indicar el tamaño del bloque, lo administra automáticamente MsDOS. No es posible de 'liberar' parcialmente un bloque de esta forma.

INT 21h/49h

INT 21h/4Ah

INT 21h/4Ah modifica el tamaño de un bloque previamente obtenido a traves de /48h. BX tiene el nuevo tamaño, ES apunta el bloque.

INT 21h/58h

INT 21h/58h es la misma función ya descrita bajo 'Operaciones con ficheros', y determina la estrategia usada en la asignación de los bloques.

Funciones para redes

Estas funciones son especificas para el uso con redes, y la descripción es demasiado complejo para describirlo aqui. Básicamente, permiten de definir nuevos perifericos al sistema (**/44h**) con todo la descripción del comportamiento de cada uno. **/5Eh** obtiene el nombre del sistema en la red.

/5Fh da acceso a la lista de redireccionamiento de la red.

INT 21h/44h

INT 21h/5Eh

INT 21h/5Fh

Varios

Para evitar la necesidad de acceder directamente a la memoria física de la computadora (y eventualmente incurrir en conflictos con otros programas en un ambiente de multitasking) las funciones **INT 21h/25h** y **/35h** dan la posibilidad de modificar y obtener los vectores. AL tiene el número del vector a cambiar y DS:DX el vector en sí.

INT 21h/25h

INT 21h/35h

INT 21h/26h copia el contenido del actual Program Segment Prefix (PSP) en un lugar asignado, y lo prepara para el uso por otro programa. (DX: segmento donde se escribirá el nuevo PSP).

INT 21h/26h

INT 21h/29h prepara un nombre de fichero para su uso en un FCB. A traves de varios bits de AL se controla el comportamiento de esta rutina. DS:SI apunta un string de caracteres que tiene el nombre del fichero, ES:DI apunta el FCB que recibirá el nombre preparado. Este proceso de 'interpretar' un nombre y formatearlo se llama 'parsing'.

INT 21h/29h

INT 21h/30h obtiene en AL:AH la versión del sistema operativo (ej. AL=3, AH=0Ah indica versión 3.10)

INT 21h/30h

INT 21h/33h controla el comportamiento del sistema operativo ante un Ctrl-Break. (AL=0 interroga, AL=1 pone un estado nuevo - DL=0 deshabilita, DH=1 habilita control)

INT 21h/33h

INT 21h/38h interroga/selecciona las características específicas a cada país. La cantidad de parámetros es demasiado grande para describir aqui.

INT 21h/38h

INT 21h/44h: ver redes.

INT 21h/44h

INT 21h/4Bh: arranca otro programa como 'hijo' o descendiente. Al terminar este, devuelve el control al programa actual.

INT 21h/4Bh

INT 21h/4Dh: interrogar por el código de terminación de un programa descendiente.

INT 21h/4Dh

INT 21h/59h solicita a MsDOS mas información sobre el último error que ocurrió con un interrupt 21h. Esta función devuelve también una sugerencia para la acción a tomar para su corrección.

INT 21h/59h

INT 21h/62h: dirección del PSP del actual programa.

INT 21h/62h

IINT 21h/63h es una función que solo existió en la versión 2.25 de MsDOS. No esta mas definida en la versión 3.xx

INT 21h/63h

Funciones 18h, 1Dh, 1Eh, 1Fh, 20h, 32h, 34h, 37h, 50h, 51h, 52h, 53h, 55h, 5Dh, 60h, y 61h son funciones reservadas (no documentadas).

INT 22h

El vector de INT 22h contiene la dirección de la rutina del sistema operativo que se ejecuta cuando se termina un programa a traves de un INT 20h, o INT 21h funciones 0, 31h o 4Ch. No se aconseja de utilizar esta dirección directamente.

INT 23h

INT 23h es la rutina que recibe control cuando el sistema operativo detecta la combinación de teclas Ctrl-C (o Ctrl-Break). Un programa de aplicación puede modificar esta dirección para implementar una 'salida de emergencia' al programa (por ejemplo para evitar bloqueo por impresora, u otro canal de entrada/salida). Ademas, el sistema operativo se encarga de poner los registros en el mismo valor que tenian antes de llamar a la interrupt que generó el INT 23h (por ejemplo el INT 21h, función 1). De esta forma, el programa que atiende el INT 23h puede determinar donde estaba la falla.

El usuario tiene varias opciones para recuperar del INT 23h:

-Puede poner un 'flag' local indicando que paso algo, o realizar alguna acción, y luego ejecutar un RETI (Return from interrupt). Todos los registros tiene que esta preservados. MsDOS en este caso reinicia la función que causó el INT 23, desde el principio. Se terminará como normalmente.

- Tomar alguna acción y luego ejecutar un RETF (Return FAR). MsDOS retomará control. El Carry le indicará que camino seguir: si el CY = 1 se abortará la aplicación totalmente. Si CY = 0, seguirá como antes.

- Mantener el control dentro de la aplicación. En este caso nunca ejecutar un RETF o RETI. Esta operación está prevista y no generará un error por parte del sistema operativo.

Se pueden llamar a otras funciones de MsDOS desde adentro de la rutina de manejo del INT 23h.

24h

INT ~~24h~~/25h

INT 24h es la rutina que recibe control cuando ocurre un error crítico en el sistema, por ejemplo errores de lectura/escritura en el disco, o errores en la impresora (falta de papel). En el caso de errores en disco, MsDOS intentará 3 veces antes de llamar a INT 24h.

La rutina que atiende al INT 24h, tiene que preservar SS, SP, DS, ES, BX, CX y DX. Solo INT 21h funciones 01h a 0Ch pueden ser utilizados dentro de la rutina.

Al volver con un RETI, el contenido de AL indicará que acción a tomar:

-AL=0, ignorar el error

-AL=1, reintentar

-AL=2, terminar el programa a traves de INT 32h

-AL=3, abortar la función en ejecución

INT 25h

INT 25h y 26h permiten lectura y escritura en sectores absolutos en el disco. En realidad forman un enlace directo entre la aplicación y el BIOS.

INT 26h

INT 27h

INT 27h es otra forma para terminar un programa e instalarlo como residente en la memoria. Esta forma no es aconsejable, y es preferible utilizar INT 21h/31h.

INT 28h, 29h, 2Ah, 2Bh, 2Ch, 2Dh, 2Eh son reservados por MicroSoft.

INT 2Fh

INT 27h dan acceso al 'print queue'. Exigencia para poder utilizar esta interrupción, es la presencia del 'print spooler' en la memoria. La forma mas sencilla es incluir el comando 'PRINT /D=LPT1' en el AUTOEXEC.BAT.

Luego, INT 27h, da la posibilidad de entrar ficheros en la cola de espera para la impresora, desde adentro de un programa. También permite de abortar todos los ficheros en la cola, o remover selectivamente ficheros.