PC Assembler

USANDO **DOS**

Daniel G. A. QUADROS

Editora Campus

ÍNDICE

PREFÁCIO DA EDIÇÃO ELETRÔNICA

É com grande prazer e imenso atraso que eu disponibilizo o meu terceiro livro da coleção "**PC Assembler**" em "formato eletrônico".

Em março de 1987, embalado com a publicação dos volumes anteriores, parti para uma obra mais ambiciosa, o **PC Assembler – Usando o DOS**. Este volume se propunha a apresentar as três versões até então lançadas do MS-DOS, inclusive alguns de seus aspectos não documentados (um assunto que começava a virar febre no momento, com o lançamento dos famosos TSRs – vide o capitulo 10). Se a memória não falha, este livro foi o primeiro entregue me formato digital.

O livro teve desempenho próximo ao anterior: foram impressos 4213 livros em três edições, do final de 88 até o início de 98, quando a Editora Campus decidiu não mais reeditar a obra e me devolveu os direitos de publicação.

Esta "edição eletrônica" segue a formatação que adotei com o volumes anteriores.

Espero que seja útil para alguém.

Daniel Quadros

Maio/2017

PREFÁCIO DA EDIÇÃO ORIGINAL

Os conhecimentos contidos neste livro foram adquiridos na minha convivência desde 1983 com o sistema MS-DOS, particularmente o projeto das versões 1 e 2 do SISNE – Sistema Operacional do Nexus (na Scopus Tecnologia), e as versões 1.10 e posteriores do software Z (na Humana Informática).

Dedico, por isso, este trabalho aos meus companheiros no projeto do SISNE: Bete Vivian, Denise, Elisabeth, Francisco, Graça, Jaime, Leon, Lilian, Mary, Sebastião e Sidney. Ainda da Scopus, uma lembrança especial a Nelson Bardelli, pelo apoio e incentivo fornecidos.

Agradeço, tamb´me, o apoio da Humana Informática (particularmente o uso de equipamentos), sem o qual esta publicação não seria possível.

Ao amigo Luiz Cláudio Navarro, um agradecimento pela revisão técnica dos originais.

Finalmente, um agradecimento todo especial à minha esposa, pela paciência e apoio durante a redação e revisão dos originais.

Daniel Quadros

1 INTRODUÇÃO

1.1 OBJETIVOS DESTE LIVRO

Este livro se destina àqueles que desenvolvem programas para o PC IBM (ou compatível), particularmente aos que utilizam a linguagem assembler.

Estes programadores estão interagindo continuamente com o DOS ("Disk Operating System") - o sistema operacional do PC IBM. Detalharemos não somente os recursos fornecidos pelo DOS ao programador, mas também as estruturas de dados utilizadas pelo DOS para fornecer estes recursos.

Admite-se que o leitor já tem familiaridade com a linguagem assembler, o hardware e o "BIOS" do PC (discutidos em profundidade nas demais obras desta série). Os programadores que já tiverem experiência com o sistema CP/M-80 encontrarão algumas comparações entre os dois sistemas; esta experiência, entretanto, não é indispensável para o perfeito acompanhamento do texto.

1.2 O QUE É UM SISTEMA OPERACIONAL

Um sistema operacional é um conjunto de programas responsáveis por fornecer a infra-estrutura necessária para que programas e operadores interajam com um computador.

Nos microcomputadores, a principal função do sistema operacional é gerenciar os dispositivos de memória de massa. Estes dispositivos são normalmente discos magnéticos. Cada face destes discos é organizada em trilhas concêntricas, por sua vez organizadas em setores (Figura 1).

Figura 1: Trilhas e setores

Esta organização física do dispositivo é "escondida" pelo sistema operacional, que permite ao operador ou programador acessar conjuntos de informações contidas em um disco através de nomes definidos pelo próprio operador ou programador. Cada um destes conjuntos de informações é o que chamamos de arquivo. O sistema operacional cuida de mapear os dados dos arquivos nos setores do disco, conforme será detalhado no Capítulo 4.

1.3 HISTÓRICO DO DOS

A história do DOS começou em maio de 1979, quando uma companhia norte-americana, a Seattle Computer, finalizou o primeiro protótipo de um computador utilizando o processador 8086. Duas importantes firmas de software foram contactadas pela Seattle: a Digital Research, que estava desenvolvendo o sistema CP/M-86 e a Microsoft, que estava desenvolvendo para o 8086 uma versão do seu já famoso interpretador BASIC.

No começo de junho a Seattle já dispunha de uma versão do BASIC da Microsoft, que gerenciava diretamente o disquete (o "Stand-Alone Disk BASIC"), entretanto a conclusão do sistema operacional da Digital Research parecia cada vez mais longíqua.

Em abril de 1980, a Seattle resolveu desenvolver internamente um sistema operacional. Com o investimento de dois homens-mês, em agosto começou a ser distribuído o QDOS 0.10 (o nome provém de "Quick and Dirty" - "Rápido e Sujo").

No final de 1980 a Microsoft fechou um acordo de distribuição do sistema, vindo a adquirir todos os direitos em julho de 1981, quando o nome MSDOS foi adotado.

Neste meio tempo, o projeto do PC IBM já estava (secretamente) a todo vapor. A IBM repetiu o caminho da Seattle e procurou a Microsoft e a Digital Research (que finalmente estava liberando o CP/M-86). O que ocorreu de errado nas conversações entre a IBM e a Digital Research é desconhecido até hoje. Oficialmente, a IBM lançou o PC com três opções de sistema operacional: o UCSD Pascal, o CP/M-86 e o MSDOS; entretanto, apenas o último estava disponível para pronta entrega e tinha um preço bastante inferior aos outros. Mais ainda, a versão do MSDOS para o PC recebeu o sugestivo nome PC-DOS. Estava clara a preferência da IBM.

O lançamento de novas versões do PC-DOS pela IBM esteve sempre ligado ao lançamento de novos hardwares:

- a versão original, 1.0, foi lançada em agosto de 1981;
- em maio de 1982 foi lançada a versão 1.1, para suportar disquetes dupla face;
- em março de 1983 foi lançada a versão 2.00. Esta foi a que mais diferenças teve de sua anterior, tendo sido resultado de uma tentativa da Microsoft de aproximar o sistema MSDOS do sistema XENIX (que é uma variante do sistema UNIX), também comercializado por ela;
- em outubro de 1983 a IBM lançou a versão 2.10 do DOS. Sua principal finalidade foi acertar alguns parâmetros de disco para as unidades "slim" (meia altura) usadas no PCjr e no PC Portátil (dois modelos já descontinuados);
- em agosto de 1984, foi lançada a versão mais desapontante do DOS, a 3.00. Introduzida simultaneamente com o PC-AT, que tinha grandes diferenças em relação aos modelos anteriores, não apresentou nenhuma grande alteraçao externa. Sua principal razão de ser foram as unidades de disquete de 1.2 Mbytes e de disco rígido de 20 Mbytes utilizadas no PC-AT. Esta versão foi também a única a ter sua vida útil previamente estipulada. Aparentemente, a IBM não conseguiu concluí-la no prazo desejado, sendo obrigada a lançá-la em duas etapas para não retardar o lançamento do PC-AT;
- em março de 1985 foi liberada a versão 3.10. Sua principal característica é o suporte à rede local IBM ("IBM PC Network") e

- finalmente, em dezembro de 1985, foi lançada a versão 3.20, junto com o PC "Conversível" ("PC Convertible"), que utiliza unidades de disquete de 3 1/2 polegadas.

Obs.: Nos referiremos com frequência às versões pelo seu primeiro número (versão 1, versão 2 ou versão 3); nestes casos estamos nos referindo a características comuns a todas as versões com este primeiro número (versão 1 se refere às versões 1.0 e 1.1, versão 2 às versões 2.00 e 2.10, etc.)

De forma geral, todas as versões são compatíveis entre si, com as mais recentes contendo algumas funções além daquelas das anteriores. A decisão de atualização de versão consiste basicamente da ponderação de três fatores:

- as versões mais recentes ocupam mais memória (ver tabela abaixo);
- as versões mais recentes têm novas facilidades; e
- as versões mais recentes corrigem problemas de versões anteriores (porém podem introduzir novos problemas).

Arquivo	DOS 1.1	DOS 2.0	DOS 2.1	DOS 3.0	DOS 3.1	DOS 3.2
IBMBIO	2.048	4.608	4.736	8.964	9.564	16.369
IBMDOS	6.390	17.152	17.024	27.920	27.760	28.477
COMMAND	4.959	17.664	17.792	22.042	23.210	23.791

As versões 1.0 e 1.1 não são mais utilizadas; entretanto, as versões 2.0, 2.1, 3.0, 3.1 e 3.2 ainda coexistem no mercado. Isto se deve ao fato da versões mais recentes não oferecerem vantagens suficientes para convencer os usuários a migrar para elas. O programador não pode, portanto, exigir o uso da versão mais recente, sob pena de deixar de atingir uma grande parcela do mercado. Em termos de Brasil, a situação é agravada pelo atraso entre o lançamento de uma versão do DOS nos EUA e a liberação, pelos fabricantes nacionais, de uma versão compatível. Até o final de 1986 não existia nenhum sistema nacional compatível com o DOS 3.00 (ou posterior); a versão mais difundida é certamente a 2.10.

1.4 PC-DOS X MS-DOS

O MS-DOS é um sistema operacional para microcomputadores baseados nos microprocessadores da família INTEL 8086 (8086, 8088, 80188, 80186, 80286, 80386 etc.), e não somente aos PC compatíveis. A Microsoft comercializa um sistema "genérico" para os fabricantes de microcomputadores, que se incumbem de adaptá-lo ao seu hardware específico.

No caso da IBM, em particular, o sistema é razoavelmente alterado, sendo o produto final denominado PC-DOS. As alterações ocorrem principalmente nos utilitários e no acesso aos dispositivos.

O sistema MS-DOS genérico tem atualmente pouca importância, pois:

- existem firmas especializadas na comercialização de versões do MS-DOS já adaptadas ao hardware do PC com alta compatibilidade com o PC-DOS tanto a nível de operação como a nível interno; e
- o sistema PC-DOS funciona (sem alterações) nos PCs compatíveis (desde que eles sejam razoavelmente compatíveis...) e pode ser comprado diretamente da IBM.

Neste livro vamos tratar do PC-DOS; a maioria das informações continuará valendo para os sistemas MS-DOS "genéricos", porém quanto mais "interna" for uma informação maior a probabilidade de existir alguma diferença.

1.5 UMA RÁPIDA REVISÃO DE CONCEITOS

O microprocessador 8088 manipula dados de 8 bits (bytes), 16 bits (words) e em algumas ocasiões especiais 32 bits (double wordss). Os double words normalmente armazenam endereços, que no 8088 são constituídos de duas partes: um segmento e um offset, ambos de 16 bits. O endereço "físico" de uma posição é obtido multiplicando-se o segmento por 16 e somando-se o offset, de forma a se obter um endereço de 20 bits. Os words são armazenados na memória com o byte mais significativo no endereço maior; os double words são armazenados com o word mais significativo no endereço maior. Por exemplo, o endereço 1234H:5678H seria armazenado a partir do endereço 2000H da seguinte forma:

endereço	conteúdo
2000h	78h
2001h	56h
2002h	34h
2003h	12h

Obs.: Neste livro adotamos a convenção de indicar números hexadecimais através do sufixo "h", como nesse exemplo apresentado.

É também comum o uso do termo parágrafos na especificação de endereços e tamanhos de áreas de memória, como sinônimo de 16 bytes.

Os bits de um dado são numerados a partir de 0, que é o bit menos significativo. Desta forma o bit mais significativo de um byte é o bit 7, e o mais significativo de um word é o 15.

O 8088 possui 13 registradores de 16 bits: DS, ES, CS, SS, PC, SP, BP, SI, DI, DX, CX, BX e AX; os quatro últimos podem ser manipulados como pares de registradores de 8 bits (DH, DL, CH, CL, BH, BL, AH e AL). Os quatro primeiros são os registradores de segmento, que contêm os segmentos de dados (DS e ES), código (CS) e pilha (SS).

Com frequência vamos nos referir a interupções de software e hardware. As primeiras são desvios causados pela execução da instrução INT e as segundas são desvios causados pela sinalização de um dispositivo externo. O 8088 dispõe de 256 interrupções, armazenando os endereços das rotinas que as tratam em double words a partir do início da memória (endereço 0). O endereço da rotina que trata a interrupção "i" está armazenado no double word de endereço "4*i".

Para maiores detalhes sobre o 8088 sugere-se a leitura do primeiro volume do "PC Assembler".

2 ESTRUTURA DO DOS

Neste capítulo é apresentada uma visão global do DOS, destacando-se as suas várias partes e interligações, principalmente por ocasião da iniciação do sistema.

2.1 O CARREGADOR DO SISTEMA - BOOT

Este é o módulo inicial do sistema, sendo sua função verificar a presença do DOS no disquete e, em caso afirmativo, carregar para a memória o arquivo IBMBIO.COM. É gravado no primeiro setor dos disquetes pelo utilitário FORMAT e lido para memória pelo BIOS no final da sua iniciação (no caso de discos rígidos o esquema é um pouco mais complicado e será descrito em detalhes no Capítulo 4).

O formato geral do BOOT é apresentado a seguir; notar a presença de uma série de parâmetros que descrevem o disco e que serão vistos com mais detalhes no Capítulo 4.

Deslocamento	Tamanho	Conteúdo
000h	3 bytes	desvio (JMP) para início do código
003h	8 bytes	identificaçao ("IBM x.xx")
00Bh	1 word	número de bytes por setor (normalmente 0200h)
00Dh	1 byte	número de setores por cluster
00Eh	1 word	número de setores reservados (normalmente 1 - o que contem o BOOT)
010h	1 byte	número de cópias da FAT (normalmente 2)
011h	1 word	número de entradas no diretório raiz
013h	1 word	número total de setores no disco
015h	1 byte	tipo de disco ("media descriptor byte")
016h	1 word	número de setores ocupados por uma cópia da FAT
018h	1 word	número de setores por trilha
01Ah	1 word	número de faces por trilha
01Ch	1 word	número de setores "escondidos" (setores físicos anteriores ao primeiro lógico)
01Eh	480 bytes	código e variáveis do BOOT
1FEh	1 word	identificação (055AAh)

Existem diferenças entre os "BOOT"s das diversas versões do DOS; de maneira geral uma versão de DOS só é carregada corretamente pelo seu respectivo BOOT. O utilitário SYS permite copiar os arquivos que contêm o DOS de um disquete para outro; entretanto, ele só atualiza o BOOT se o disco destino contiver um BOOT mais antigo que o correspon-dente à versão do SYS. Exemplificando: um disco formatado com o FORMAT do DOS 3.00 tem gravado o BOOT desta versão; se enviamos este disco para uma pessoa que usa a versão 2.10 do DOS e ela usar o SYS para colocar o sistema, o disco ficará com o BOOT da 3.00 e os arquivos de sistema da 2.10. Este disco não carregará corretamente o DOS !

A opção /B do FORMAT permite gerar discos nos quais pode ser (até o momento) gravada qualquer versão do DOS. Isto é feito gravando o BOOT da versão 1.00.

2.2 INTERFACE COM DISPOSITIVOS - IBMBIO.COM

Este módulo contém as rotinas básicas de interface com os dispositivos (console, disco, impressora etc.). É gravado como um arquivo invisível pela opção /S do utilitário FORMAT ou pelo utilitário SYS.

As rotinas de tratamento de dispositivos são discutidas no Capítulo 3.

2.3 NÚCLEO - IBMDOS.COM

É o módulo que contém as rotinas principais do DOS, sendo independente do hardware. Também é gravado como um arquivo invisível pela opção /S do utilitário FORMAT ou pelo utilitário SYS.

A interface do DOS com programas é feita por este módulo e está detalhada principalmente nos Capítulos 5 e 7.

2.4 INTERPRETADOR DE COMANDOS - COMMAND.COM

Este módulo é armazenado no arquivo COMMAND.COM, podendo ser substituído pelo programador experiente. É o responsável pela interface do operador com o DOS, fornecendo recursos para:

- executar programas
- executar arquivos de comandos
- redirecionar a entrada e saida padrões
- executar as funções básicas de manipulação de disco, através dos utilitários internos.

O COMMAND.COM é discutido com mais detalhes no Capítulo 9.

2.5 CONFIGURAÇÃO DO SISTEMA - CONFIG.SYS

O arquivo CONFIG.SYS é um arquivo texto que permite configurar diversos parâmetros do DOS, sendo lido na iniciação. Não é tratado pela versão 1 do DOS.

Dentro dos diversos parâmetros, merecem destaque:

- BUFFERS

  permite especificar o número máximo de setores de disco que o DOS poderá manter na memória (ver Capítulo 3).

- DEVICE

  permite substituir ou acrescentar rotinas de tratamento de dispositivos às presentes no IBMBIO.COM (ver Capítulo 3)

- FILES

  especifica o número máximo de arquivos que podem ser abertos simultaneamente através das funções compatíveis XENIX do DOS (ver Capítulo 4)

- FCBS

especifica o número máximo de arquivos que podem ser abertos simultaneamente através das funções compatíveis CP/M do DOS (ver Capítulo 4)

2.6 INICIAÇÃO DO DOS

A iniciação do DOS ocorre quando o sistema é ligado ou reiniciado via CTRL ALT DEL, e segue os sequintes passos:

a) O BIOS tenta ler o BOOT do primeiro disquete. Se não conseguir ele tenta ler o "Master Boot" do primeiro disco rígido; o "Master Boot" por sua vez irá carregar o BOOT do primeiro setor da partição DOS do disco.
b) O BOOT verifica a presença dos arquivos IBMBIO.COM e IBMDOS.COM, que devem ser os dois primeiros do diretório raiz. Em seguida, o arquivo IBMBIO.COM é lido na memória, assumindo-se que ele ocupa setores contiguos.
c) A iniciação do IBMBIO carrega o IBMDOS.COM, inicia os tratadores padrão de dispositivos e chama a iniciação do DOS, que prepara suas estruturas para uma configuração padrão. Utilizando o DOS, o arquivo CONFIG.SYS é lido e tratado, com a consequente atualização das estruturas do DOS. O interpretador de comandos é então carregado e disparado.
d) O interpretador de comandos inicia as suas variáveis e vetores de interrupção, protege a sua parte residente e recarrega a sua parte transiente no final da memória. A seguir, procura o arquivo AUTOEXEC.BAT. Encontrando-o, passa a executá-lo; caso contrário, apresenta o "prompt" e aguarda um comando do operador. A partir deste instante o sistema está em funcionamento normal.

2.7 MAPA DA MEMÓRIA

O mapa a seguir mostra o posicionamento dos vários módulos do DOS, ao final da sua iniciação.

```
00000H ---------------------------------
         Vetores de Interrupção
00400H ---------------------------------
         Variáveis do BIOS
00500H ---------------------------------
         "Area de Comunicação do DOS" (1)
00600H ---------------------------------
         IBMBIO.COM
       ---------------------------------
         IBMDOS.COM
       ---------------------------------
         Areas agregadas ao DOS (2)
       ---------------------------------
         COMMAND.COM - parte residente
       ---------------------------------
         Outros programas residentes
       ---------------------------------
         Área para carga de programas
       ---------------------------------
         COMMAND.COM - parte transiente
       ---------------------------------
```

(1) Esta área tem vários usos:

00500H - Controle de "hardcopy" do BIOS

00501H - Usado pelo BASIC

00504H - Usado pelo tratador de disquete, quando existe uma única unidade física, indica a quem corresponde o disquete instalado na unidade:

0 - unidade lógica A:

1 - unidade lógica B:

00510H a 00521H - Usado pelo BASIC

00522H a 00527H - Usado pelo tratador de disquetes, contém os parâmetos de disco para uso do BIOS, em substituição aos contidos na própria ROM.

00530H a 005FFH - Usado pelo utilitário MODE, que coloca aqui rotinas que interceptam os acessos às interfaces seriais e/ou impressoras, para efetuar redirecionamento e/ou tratamento de erros.

(2) Contém áreas agregadas durante a leitura do arquivo CONFIG.SYS - buffers adicionais, tratadores de dispositivos etc.

3 TRATADORES INSTALÁVEIS DE DISPOSITIVOS

Uma das grandes novidades da versão 2.00 do DOS foi a criação de uma estrutura padronizada para as rotinas de tratamento ("drivers") de dispositivos. Além disso, foi introduzida a facilidade de incorporação de novos tratadores ao DOS.

A instalação destes tratadores adicionais é feita através da inclusão de comandos "DEVICE" no arquivo CONFIG.SYS. Este comando tem a seguinte forma:

```
DEVICE = <nome de arquivo> <parâmetros>
```

Uma vez que o arquivo CONFIG.SYS é tratado na fase de iniciação do DOS, os novos tratadores só estarão disponíveis depois da carga do sistema. Isto impede, por exemplo, que o DOS seja carregado de um dispositivo que só possa ser acessado através de um tratador instalável.

3.1 DISPOSITIVOS DE BLOCO E DE CARACTER

O DOS classifica os dispositivos em dois tipos:

Dispositivos de caracter

A principal característica destes dispositivos é que o seu acesso é feito sequencialmente, caracter a caracter. São identificados por nomes de oito caracteres (que seguem as mesmas regras de nome de arquivos).

Existem três dispositivos de caracter com tratamento especial pelo DOS:

- NUL

  Este dispositivo sempre informa que não tem dados disponíveis para a leitura e aceita operações de escrita, ignorando os dados recebidos - é um "saco sem fundo". Conforme será visto adiante, o tratador deste dispositivo é o ponto de partida para o DOS localizar os demais tratadores.

- Console

  É o dispositivo usado pelo DOS para se comunicar com o operador. Normalmente é o dispositivo de nome CON.

- Relógio

  Utilizado pelo DOS para obter a data e a hora, normalmente tem o nome CLOCK$.

Os tratadores padrão do DOS suportam os seguintes dispositivos de caracter: CON (console), AUX (linha serial #1), PRN (impressora paralela #1), NUL, CLOCK$ (relógio), LPT1 (impressora paralela #1), LPT2 (impressora paralela #2), COM1 (linha serial #1) e COM2 (linha serial #2).

Dispositivos de Bloco

São dispositivos capazes de efetuar acessos aleatórios, lendo ou escrevendo blocos de dados de tamanho fixo. Normalmente estes dispositivos são discos magnéticos (disquetes ou disco rígido), com um bloco correspondendo a um setor.

Um tratador de dispositivo de bloco pode tratar vários dispositivos, denominados unidades. A cada unidade é associada pelo DOS uma letra (A, B, C etc.), atraves do qual ela é acessada.

Apesar de o tratador de dispositivos de bloco trabalhar com blocos de tamanho fixo, o DOS permite que o programador acesse estes dispositivos através de registros de tamanho variável. Para isso ele utiliza várias áreas de memória, denominadas buffers, para armazenar temporariamente setores.

Quando um programa requisita uma leitura que necessita apenas de uma parte de um setor o DOS lê o setor para um dos seus buffers e copia a parte de interesse para o programa. Da mesma forma, quando é feita uma escrita parcial em um setor o DOS irá montar o setor em um dos seus buffers para depois passá-lo ao tratador.

O gerenciamento destes buffers é uma das partes mais importantes do DOS, merecendo mais alguns comentários. Em primeiro lugar, o bom desempenho do DOS depende do bom aproveitamento dos buffers. De uma maneira geral, pode-se melhorar o desempenho mantendo-se em memória o maior tempo possível setores de leitura e/ou escrita frequente. Isso pode ser facilitado pelo aumento do número de buffers (o que é feito através do comando BUFFERS do CONFIG.SYS). Entretanto, o DOS é obrigado a verificar se um setor já está nos buffers (eventualmente alterado) antes de efetuar uma chamada direta ao tratador que o afete. Por este motivo, o aumento do número de buffers só deixa o sistema mais rápido até um certo ponto, quando ele passará a ficar mais lento quanto maior a quantidade de buffers.

Uma complicação a mais no gerenciamento dos buffers é a possibilidade da troca, pelo operador, do disco contido em uma unidade. Conforme será detalhado mais adiante, o DOS resolve este problema consultando o tratador e adotando algumas hipóteses simplificadoras.

Existe ainda a questão do tamanho de um setor e do tamanho de um buffer. Em princípio, o DOS permite o uso de unidades com setores de 32 a 64K-1 bytes. Entretanto, ele assume que um setor cabe em um único buffer e utiliza sempre um buffer para armazenar um setor (mesmo que caibam mais). O tamanho de um buffer é fixo no DOS, não sendo alterável pelo usuário, e corresponde normalmente ao maior setor utilizado pelos tratadores padrão. O tamanho do buffer é, portanto, o limitante para o tamanho máximo do setor dos dispositivos de bloco. Na prática, o tamanho do buffer tem permanecido em 512 bytes em todas as versões do DOS, sendo este também o tamanho do setor utilizado em todas as unidades de disco comercializadas pela IBM, sejam fixas ou removíveis.

Finalmente, cabe ainda discutir a famosa "barreira dos 32 Megabytes". Conforme dito, o tamanho de um setor está limitado a 512 bytes. Por outro lado, toda a interface com os tratadores de dispositivos de bloco baseia-se no acesso de blocos através do seu endereçamento por um número lógico do bloco, contido em um word. Ora, um word permite armazenar números de 0 a 0FFFFh o que significa que um dispositivo pode ter no máximo 64 K blocos de no máximo 512 bytes, num total de 32 Mbytes. O levantamento dessa barreira (que deverá acontecer algum dia) implicará o aumento do tamanho dos buffers (provavelmente com disperdício de memória no acesso às unidades atuais) ou o aumento do número máximo de setores (o que acarretará uma mudança na interface com os tratadores e uma complicação extra nos cáculos internos ao DOS que envolvam número lógico de setores).

Para concluir, resta lembrar que o DOS se preocupa também com o caso contrário, o do acesso a registros maiores que um bloco. Nestes casos, o DOS procura passar ao tratador um comando para a leitura (ou escrita) do maior número de setores consecutivos. Isto porque é normalmente possível comandar estas leituras de forma otimizada à interface, obtendo desempenho sensivelmente superior ao acesso dos blocos um a um.

3.2 CABEÇALHO DE UM TRATADOR

Cada tratador de dispositivo tem no seu começo uma estrutura de dados utilizada pelo DOS para obter informações a seu respeito.

Uma destas informações é o endereço do cabeçalho do próximo tratador, o que permite ao DOS organizar a *cadeia de tratadores*. Esta cadeia é uma lista ligada, que começa pelo cabeçalho do tratador do dispositivo NUL e inicialmente contém os tratadores padrão. A instalação de tratadores do usuário, como será detalhado adiante, faz com que os seus cabeçalhos sejam inseridos na cadeia após o cabeçalho do NUL. O formato típico da cadeia é mostrado abaixo:

```
NUL -> USR1 -> USR2 -> ... -> PAD1 -> ...
```

onde os tratadores USR são os definidos pelo usuário e os PAD são os tratadores padrão.

Vejamos agora, em detalhes, a estrutura do cabeçalho:

Desloc.	Tamanho	Conteúdo
00h	1 dword	Ponteiro para o próximo cabeçalho Montado pelo DOS na instalação, contém 0FFFFH:0FFFFH no último cabeçalho da cadeia
04h	1 word	Atributos do tratador
		Bit 15 — 1 dispositivo de bloco 0 dispositivo de catacter
		Bit 14 — 1 suporta IOCTL (1) 0 IOCTL não é suportado
		Bit 13 — dispositivos de bloco: 1 formato não IBM(2) 0 formato IBM dispositivo de catactere, DOS 3.2: 1 suporta “escrita até ocupado” (3) 0 não suporta
		Bit 12 — se dispositivo de caractere: 0 não permite escrita rápida (4) 1 permite escrita rápida
		Bit 11 — 1 suporta abertura, fechamento e informação de meio removível (1) 0 não suporta estas funções
		Bits 10 a 7 — devem conter 0
		Bit 6 — se dispositivo de bloco no DOS 3.2: 1 suporta Informa/Altera Dispositivo Lógico (1) 0 não suporta estas funções
		Bits 5 e 4 — devem conter 0
		Bit 3 — 1 dispositivo de relógio 0 outro
		Bit 2 — 1 dispositivo NUL

		0 outro
	Bit 1	1 dispositivo de escrita no console
		0 outro
	Bit 0	se dispositivo de caractere:
		1 dispositivo de leitura do console
		0 outro
		se dispositivo de bloco, DOS 3.2:
		1 suporta IOCTL genérico
		0 não suporta
06h	1 word	Ponteiro para a rotina de estratégia (5)
08h	1 word	Ponteiro para a rotina de interrupção (5)
0Ah	8 bytes	Nome do dispositivo (se de caractere) Contém o nome do dispositivo, alinhado à esquerrda completado com brancos à direita. Se dispositivo de bloco, a primeira posição será preenchida pelo DOS com o número de unidades tratadas por este tratador.

(1) só no DOS 3 (estas funções serão detalhadas adiante).

(2) será visto na descrição da função "Informa BPB".

(3) "escrita até ocupado" é utilizada nos tratadores de impressora. Nela, se o dispositivo está pronto, os dados são enviados; caso contrário, um erro é retornado imediatamente.

(4) para melhora de eficiência, o DOS permite que um (e somente um) dos dispositivos de caracter disponha de uma maneira alternativa de escrita de um caracter. Esta maneira consiste na execução de INT 29h, com AL contendo o caracter a ser escrito; a rotina do tratador não deve alterar o conteúdo dos registradores. Esta facilidade é empregada pelo tratador padrão CON, que deve ser substituído para que possa ser usada em um tratador do usuário.

(5) estas rotinas são vistas no item "Requisições aos Tratadores".

3.3 INSTALAÇÃO DE TRATADORES

Conforme já dito, a instalação de um tratador é comandada pela inclusão no arquivo CONFIG.SYS de uma linha com o seguinte formato:

```
DEVICE = <nome de arquivo> <parâmetros>
```

O <nome de arquivo> especifica um arquivo que contém um ou mais tratadores. Normalmente este arquivo é uma imagem de memória (isto é, contém bytes sem nenhuma codificação adicional), que é carregada para o offset 0 de um segmento qualquer (o segmento vai depender da versão do DOS, se foram alocados buffers ou carregados outros tratadores etc.). A versão 3.00 (e posteriores) permite ainda que o arquivo contenha referências inter-segmentos a serem relocadas em função do segmento de carga; nesse caso o arquivo deve seguir o formato .EXE descrito no Capítulo 6.

No início do arquivo deve estar um cabeçalho de tratador. No caso de termos apenas um tratador no arquivo, o ponteiro para o próximo tratador deve conter 0FFFFh:0FFFFh. Caso contrário, deve conter no primeiro word o offset do segundo cabeçalho no arquivo (o segmento será preenchido pelo DOS). O segundo cabeçalho deve, por sua vez, apontar para o terceiro e assim por diante, devendo o ponteiro do último cabeçalho conter 0FFFFh:0FFFFh.

Embora o DOS 3.00 permita que um arquivo com tratadores contenha referências inter-segmentos, cabe notar que os ponteiros de encadeamento de cabeçalhos e os ponteiros para as rotinas de estratégia e interrupção são sempre offsets em relação ao segmento de carga.

No Capítulo 11 pode ser encontrado um exemplo prático da geração de um arquivo em formato "imagem de memória", contendo dois tratadores instaláveis.

Após a carga do arquivo especificado, o DOS irá iniciar os tratadores nele contidos, chamando as respectivas rotinas de iniciação e atuali-zan-do os cabeçalhos e suas estruturas internas.

Conforme já dito, o DOS insere os tratadores instaláveis após o tratador do dispositivo NUL e antes dos demais tratadores. Este posicionamento é importante devido ao fato do DOS procurar os tratadores de caracter a partir do cabeçalho do NUL. É, portanto, possível substituir os tratadores padrão de dispositivos de caracter, exceto o tratador do NUL.

3.4 REQUISIÇÕES DO DOS A UM TRATADOR

Toda vez que o DOS necessita do serviço de um tratador ele segue os seguintes passos:

1) monta uma requisição de serviço, cuja estrutura será detalhada adiante;
2) chama a rotina de estratégia do tratador (através de um "call far") com ES:BX apontando para o início da requisição. A rotina de estratégia deve simplesmente armazenar o pedido; e
3) chama a rotina de interrupção do tratador (também através de um "call far"), que deve executar o pedido.

Tanto a rotina de estratégia como a de interrupção não devem alterar o conteúdo dos registradores. Só é garantido espaço na pilha para armazenamento de todos os registradores, devendo o tratador usar uma pilha local se necessitar de mais espaço (na prática os tratadores internos do DOS utilizam a pilha do DOS apesar de empilharem mais dados além dos registradores).

Uma requisição de serviço a um tratador consiste de um cabeçalho padrão para todas as funções, seguido de dados específicos para cada uma delas. O formato do cabeçalho é definido abaixo:

Deslocamento	tamanho	conteúdo
0	1 byte	tamanho da requisição (inclui cabeçalho)
1	1 byte	número da unidade (para dispositivos de bloco)
2	1 byte	número da função a executar
3	1 word	resultado da operação
5	8 bytes	reservado ao DOS

As seguintes funções estão definidas:

0	iniciação
1	testa troca de disco (B)
2	constrói BPB (B)
3	IOCTL (leitura)
4	leitura
5	leitura sem espera (C)
6	"status" de leitura (C)
7	"flush" de leitura (C)
8	escrita
9	escrita com verificação

10	"status" de escrita (C)
11	"flush" de escrita (C)
12	IOCTL (escrita)
13	abertura (3)
14	fechamento (3)
15	informa se removível (3) (B)
19	IOCTL genérico (3.2) (B)
23	informa disp. lógico (3.2) (B)
24	altera disp. lógico (3.2) (B)

(B) somente para dispositivos de bloco

(C) somente para dispositivos de caracter

(3) somente a partir da versão 3.00

(3.2) somente a partir da versão 3.20

O resultado da operação tem o seguinte formato:

Bit	**conteúdo**
15	0 = operação bem sucedida 1 = erro, ver bits 7 - 0
14 a 10	reservados
9	"dispositivo ocupado" ver funções de "status", leitura sem espera e informação de disco removível
8	operação completada deve ser ligado pelo tratador ao final da operação
7 a 0	código do erro, se bit 15 = 1: 00h tentativa de gravação em dispositivo protegido 01h unidade inválida 02h dispositivo não pronto 03h comando desconhecido 04h erro de leitura (erro de CRC em discos) 05h tamanho da requisição incorreto 06h erro de posicionamento (seek em disco) 07h tipo de disco desconhecido 08h setor não encontrado 09h impressora sem papel 0Ah falha em escrita 0Bh falha em leitura 0Ch erro não especificado 0Dh reservado 0Eh reservado 0Fh troca ilegal de disco (normalmente os códigos 0, 1, 2, 4, 6, 7, 8 e 0Fh são usados apenas com dispositivos de bloco e os códigos 9, 0Ah e 0Bh apenas com dispositivos de caracter).

3.5 FUNÇÕES DOS TRATADORES

3.5.1 INICIAÇÃO

- Função número 0

- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho
0Dh	1 byte	número de unidades
0Eh	1 dword	endereço final da área ocupada pelo tratador
12h	1 dword	ponteiro para parâmetros na entrada, ponteiro para tabela de BPBs na saída (somente para dispositivos de bloco).
16h	1byte	letra correspondente à primeira unidade (dispositivos de bloco, 0 = A:, 1 = B: etc.)

- Descrição

  Esta função é chamada uma única vez, quando da instalação do dispositivo. O DOS fornece na requisição um ponteiro para os parâmetros contidos na linha do comando "DEVICE" do arquivo CONFIG.SYS que ocasionou a carga do tratador. Este ponteiro aponta para uma área que não deve ser alterada e contém duas partes: o nome do arquivo que contém o tratador (finalizado por um byte contendo 00h), e os parâmetros especificados após o nome (finalizado por um "*carriage return*" - 0Dh). Após estas duas informações o DOS coloca um "*line feed*" (0Ah); caso não sejam fornecidos parâmetros este código é colocado logo após o 00h que termina o nome do arquivo:

  linha no CONFIG.SYS: DEVICE = PLOTTER.SYS 3 1

  ponteiro aponta para: PLOTTER.SYS<00h>3 1<0Dh><0Ah>

  linha do CONFIG.SYS: DEVICE = FITA

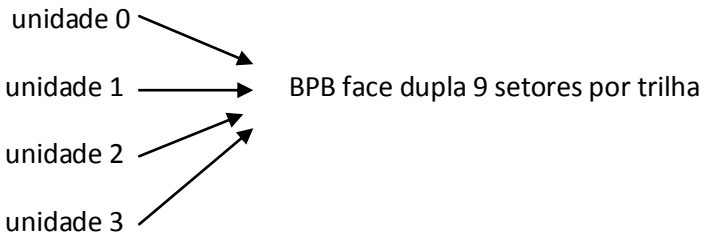
  ponteiro aponta para: FITA<00h><0Ah>

  O tratador deve efetuar as seguintes operações:

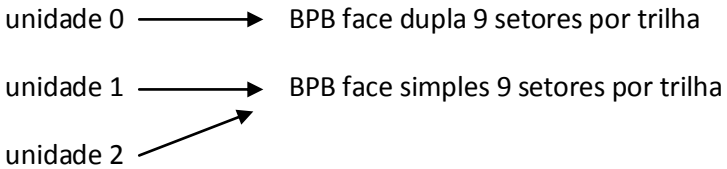
  1. Verificar a presença dos dispositivos que controla e iniciá-los;
  2. No caso de dispositivo de bloco, deve-se colocar na requisição o número de dispositivos que serão tratados. O DOS informa na requisição qual letra será associada à primeira unidade e atualiza o cabeçalho do tratador com o número de unidades informado; e
  3. No caso de dispositivo bloco, deve-se colocar na requisição o endereço de uma tabela, que contém, para cada unidade, o endereço de uma **BPB**. Os endereços são armazenados na tabela na forma de um offset em relação ao segmento que contém o cabeçalho do tratador. A BPB é uma tabela (a "BIOS PARAMETER BLOCK" - bloco de parâmetros do BIOS), que descreve as características do disco contido na unidade. No caso da unidade suportar mais de um tipo de disco, a rotina de iniciação deve colocar na tabela o endereço da BPB que descreve o de maior capacidade.

Vamos tomar como exemplo um tratador de discos 5 1/4" que saiba tratar quatro tipos de disco (face simples 8 setores por trilha, face dupla 8 setores por trilha, face simples 9 setores por trilha e face dupla 9 setores por trilha) e dois tipos de unidade (face simples e face dupla). Este tratador teria, portanto, 4 BPBs, uma para cada tipo de disco. Vamos supor ainda que o tratador suporte até quatro unidades de disco. Na iniciação o tratador descobre quantas unidades estão presentes (e qual o tipo de cada uma) e monta a tabela de BPBs de acordo:

para 4 unidades, todas dupla face:



para 3 unidades, uma face dupla e as demais face simples:



4. Colocar na requisição o endereço da última posição ocupada pelo tratador. Isto permite ao tratador tanto alocar memória ao seu final como descartar do código de iniciação (que não será mais necessário). No caso de se ter mais de um tratador em um arquivo deve-se retornar o mesmo endereço na iniciação de todos os tratadores.
5. O tratador atualiza o resultado da operação (tipicamente com 0100h - operação completada sem erros) e retorna ao DOS.

Durante a iniciação, o tratador pode chamar as funções do DOS de número 1 a 0Ch (acesso ao console) e 30h (informa versão).

3.5.2 TESTA TROCA DE DISCO

- Função número 1
- Somente para dispositivos de bloco
- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho
0Dh	1 byte	tipo de disco ("media descriptor")
0Eh	1 byte	resultado: FFh: disco foi trocado 00h: disco pode ter sido trocado 01h: disco não foi trocado
0Fh	1 dword	ponteiro para identificação anterior de volume, se trocou disco e se suporta informa-ção de disco removível (apenas para DOS 3.xx)

- Descrição

  Esta função é chamada pelo DOS no início de operações de acesso ao diretório (abertura e fechamento de arquivos, busca de nomes de arquivo ou subdiretório em diretório etc.). Sua finalidade é informar ao DOS se o conteúdo dos seus buffers é imagem fiel do disco contido na unidade.

  Na prática encontramos dois casos a serem cuidados:

  - Discos fixos ("winchesters")

    Neste caso a resposta é sempre "disco não foi trocado"

  - Discos removíveis (disquetes)

    neste caso, normalmente, o hardware não é capaz de informar com certeza a troca de disco. Unidades que tenham sensor na porta podem garantir que o disco não foi trocado, porém não podem saber se a porta foi aberta e fechada sem troca de disco. No caso das unidades normais do PC-IBM (que não tem sensor na porta) a Microsoft adotou um esquema inteligente: se não passaram mais de dois segundos desde o último acesso (e portanto a luz da unidade está acesa e o motor em movimento) é assumido que o disquete não foi trocado. Na maioria dos casos, entretanto, o tratador só pode devolver a informação que o disco pode ter sido trocado.

  Quando o tratador informa que pode ter havido uma troca (mas não tem certeza), o DOS verifica se algum de seus buffers contém dados com gravação pendente. Se não há gravação pendente o DOS opta pela segurança e admite que houve troca de disco, desconsiderando o conteúdo de seus buffers. Em caso afirmativo, o DOS admite que o disco não foi trocado, donde obtemos uma regra importante: um disco não pode ser trocado se existe nele um arquivo aberto para escrita (não é aconselhável trocá-lo também se tiver arquivo aberto para leitura, pois o DOS só detectará a troca no próximo acesso a diretório).

Concluindo que ocorreu troca de disco, o DOS descarta os buffers que contenham dados daquela unidade e chama a função "constrói BPB" do tratador (que será detalhada adiante).

Na versão 3.00 foi introduzida no DOS a possibilidade do tratador ler do disco um "identificador de volume". Neste caso, quando o tratador informar que ocorreu (com certeza) uma troca e o DOS considerá-la inválida, o DOS pede a recolocação do disco anterior. No caso do tratador não suportar identificação de volume e indicar no seu cabeçalho que suporta informação de disco removível (que é o caso comum), ele deve retornar como identificação a cadeia "NO NAME", terminada por 00h.

3.5.3 CONSTROI BPB

- Função número 2
- Somente para dispositivos de bloco
- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho
0Dh	1 byte	tipo de disco ("media descriptor")
0Eh	1 dword	endereço de buffer do DOS
12h	1 dword	ponteiro para BPB

- Descrição

Esta função é chamada quando da troca de disco e deve determinar o tipo de disco contido na unidade e retornar na requisição um ponteiro para a respectiva BPB.

O DOS fornece na requisição o tipo de disco que estava anteriormente na unidade e o endereço de um buffer, cujo uso depende do bit 13 do atributo do tratador (definido no seu cabeçalho):

bit 13 = 0 (formato IBM)

> Neste caso, o primeiro setor da primeira cópia da FAT (tabela de alocação de arquivos, que será vista com detalhes no Capítulo 4) deve estar no mesmo setor para todos os tipos de disco aceitos pela unidade. Isto porque o DOS leu este setor para o buffer antesde chamar esta função, sem saber qual o tipo de disco contido na unidade. O buffer não pode ser alterado pelo tratador, porém pode ser consultado. Normalmente a sua primeira posição é utilizada para descobrir o tipo de disco.

bit 13 = 1 (formato não IBM)

> Neste caso, o conteúdo do buffer é indefinido, podendo ser usado como área de rascunho pelo tratador.

Abaixo é detalhada a estrutura da BPB ("BIOS Parameter Block" - Bloco de Parâmetros do BIOS); uma cópia da BPB está gravada no setor de "boot" do disco (compare esta estrutura com a estrutura do "boot" fornecida no Capítulo 2).

Deslocamento	Tamanho	Conteúdo
00h	1 word	número de bytes por setor
02h	1 byte	número de setores por cluster(*) tem que ser potência de 2
03h	1 word	número de setores reservados
05h	1 byte	número de cópias da FAT (*)
06h	1 word	número máximo de entradas no diretório raiz (*)
08h	1 word	número total de setores no disco
0Ah	1 byte	tipo de disco ("media descriptor ") valores definidos para formato IBM: 0FFh - 5 1/4" 8 setores/trilha 2 faces 0FEh - 5 1/4" 8 setores/trilha 1 face 0FDh - 5 1/4" 9 setores/trilha 2 faces 3 1/2" 9 setores/trilha 2 faces 0FCh - 5 1/4" 9 setores/trilha 1 face 0F9h - 5 1/4" 15 setores/trilha 2 faces 0F8h - disco fixo (winchester)
0Bh	1 word	número de setores ocupados por uma cópia da FAT

(*) o significado destes campos será visto no Capítulo 4

Além destas informações (que interessam ao DOS e não ao tratador), existem mais três parâmetros que devem ser armazenados após a BPB e são de interesse do tratador:

Deslocamento	Tamanho	Conteúdo
0Ch	1 word	número de setores por trilha
0Eh	1 word	número de faces
10h	1 word	número de setores “escondidos”

A importância destes parâmetros está no fato do DOS trabalhar sempre em termos de setores "lógicos", que devem ser convertidos pelo tratador em setores "físicos" (face, trilha e setor). Isto é feito através das seguintes operações:

1. Soma-se ao setor lógico o número de setores escondidos (setores físicos não "enxergados" pelo DOS);
2. Divide-se o número obtido pelo número de setores por cilindro (número de setores por trilha vezes o número de faces). O quociente é a trilha a ser acessada;
3. Divide-se o resto da divisão anterior pelo número de setores por trilha. O quociente é a face; o resto é o setor a ser acessado.

Finalmente, no caso de tratadores que suportem identificação de volume, esta chamada indica a troca legal de disco, devendo a identificação do novo disco ser lida e armazenada.

3.5.4 LEITURA, IOCTL (LEITURA), ESCRITA, ESCRITA COM VERIFICAÇÃO E IOCTL (ESCRITA)

- Funções números 3, 4, 8, 9, 12

- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho
0Dh	1 byte	tipo de disco ("media descriptor") (indefinido para dispositivos de caracter)
0Eh	1 dword	endereço de transferência
12h	1 word	número de bytes (dispositivo de caracter) ou setores (dispositivo de bloco)
14h	1 word	setor inicial (indefinido para dispositivos de caracter)
16h	1 dword	ponteiro para identificação do volume, em caso de erro 0Fh (troca ilícita de disco)

- Descrição

  Todas estas funções envolvem a transferência de dados. No caso das funções IOCTL a transferência é entre o DOS e o tratador; nos demais casos é entre o DOS e o dispositivo.

  O tratador deve transferir os dados especificados (que podem ser no máximo 0FFFFh bytes) e atualizar o número de bytes ou setores e o resultado da operação na requisição.

  A escrita com verificação se destina somente a dispositivos de bloco, nos quais é possível verificar se os dados escritos podem ser lidos sem erros (não implica em verificar se os dados lidos são iguais ao escritos). Quando a verificação não for possível, deve ser feita escrita normal.

  No caso de dispositivos de bloco, algumas observações merecem ser feitas:

  1. Em alguns casos o DOS pode requisitar uma escrita de 64 Kbytes com um endereço da forma XXXX:YYYY com YYYY diferente de zero. Nestes casos o tratador pode ignorar os YYYY últimos bytes;
  2. O hardware do PC impede a transferência, via DMA, de blocos que contenham uma mudança nos 4 bits mais significativos do endereço ("fronteira física de 64 Kbytes"). Esta limitação deve ser solucionada pelo tratador, normalmente copiando o setor que contém a fronteira para um buffer local ao tratador e quebrando a transferência em até três partes: os setores antes da fronteira, o setor que contém a fronteira e os setores após a fronteira:

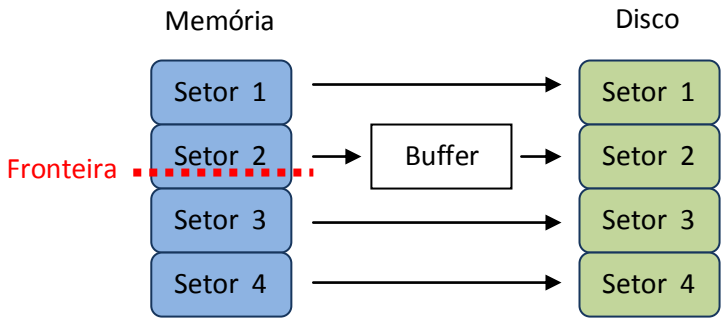


Figura 2: Tratamento de fronteira de DMA

3. Caso o tratador suporte identificação de volume, ele pode saber (através das funções de abertura e fechamento) se existe arquivo aberto na unidade. Em caso afirmativo, se ocorreu troca de disco o tratador deve devolver como resultado da operação o erro 0Fh (troca ilícita de disco) e colocar na requisição ponteiro para a identificação do disco que deve ser recolocado (o DOS se encarrega de pedir a recolocação do disco)

3.5.5 LEITURA SEM ESPERA

- Função número 5
- Somente para dispositivos de caracter
- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho
0Dh	1 byte	código lido do dispositivo

- Descrição

  Se o dispositivo tem um caracter pronto para leitura, deve-se desligar o bit de ocupado no cabeçalho da requisição e retornar o próximo caracter que será lido. Caso contrário, deve-se ligar o bit de ocupado no cabeçalho da requisição.

  Esta função permite ao DOS saber qual será o próximo caracter a ser lido de um dispositivo (se já estiver disponível) sem efetuar a sua leitura. Seu uso típico é na detecção de códigos especiais no console (tais como Control S, Control P, Control C etc.) durante outras operações que não a leitura. Caso o código informado por esta função não seja especial, ele pode ser deixado no dispositivo até que ocorra uma operação de leitura.

3.5.6 "STATUS" DE LEITURA

- Função número 6
- Somente para dispositivos de caracter
- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho

- Descrição

  Tendo o dispositivo um buffer, isto é, se ele pode ter armazenados caracteres para leitura, o bit de ocupado do resultado deve ser ligado, se este buffer estiver vazio, ou desligado, se existerem caracteres prontos para leitura.

  No caso de dispositivos sem buffer, o bit de ocupado deve ser desligado. Isto se deve ao fato de o DOS só efetuar leitura quando detectar que o dispositivo está "não ocupado".

Esta função permite que o DOS realize outras tarefas (por exemplo, testar se foi digitado Control C) enquanto aguarda um caracter estar disponível, quando o dispositivo dispõe de um buffer. É importante notar que o buffer não necessita ser uma fila com várias posições (como no teclado); pode ser um registrador fornecido pelo hardware (por exemplo, o registrador de dados de uma UART).

3.5.7 "FLUSH" DE LEITURA

- Função número 7
- Somente para dispositivos de caracter
- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho

- Descrição

Esta função causa o cancelamento (flush) de leituras pendentes, isto é, o esvaziamento do buffer de leitura do dispositivo. O DOS utiliza esta função, por exemplo, nas funções de leitura de console que exigem a digitação de uma tecla, ignorando teclas digitadas anteriormente.

3.5.8 "STATUS" DE ESCRITA

- Função número 10
- Somente para dispositivos de caracter
- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho

- Descrição

Esta função deve ligar o bit de ocupado no resultado se o dispostivo ainda está tratando um pedido de escrita anterior.

Um retorno com o bit de ocupado desligado indica que o dispositivo está livre para tratar a próxima requisição de escrita.

3.5.9 "FLUSH" DE ESCRITA

- Função número 11
- Somente para dispositivos de caracter
- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho

- Descrição

  Esta função causa o cancelamento de escritas pendentes, isto é, o esvaziamento do buffer de escrita do dispositivo.

3.5.10 ABERTURA E FECHAMENTO

- Funções número 13 e 14
- Somente chamadas no DOS 3.xx, se o bit 11 do atributo do tratador estiver ligado
- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho

- Descrição

  Estas funções permitem que o tratador saiba se existem arquivos abertos no dispositivo. Isto é feito mantendo-se um contador que é incrementado a cada chamada à função de abertura e decrementado a cada função de fechamento. O conteúdo deste contador será o número de arquivos abertos.

  No caso de dispositivos de caracter, este contador pode ser usado para executar procedimentos especiais de iniciação e terminação (por exemplo, colocar um cabeçalho no início de cada impressão e pular de folha no final).

  No caso de dispositivos de bloco, o contador indica quando um disco removível pode ser trocado legalmente.

  Importante: Os dispositivos CON, AUX e PRN estão sempre abertos.

3.5.11 INFORMA SE DISCO REMOVÍVEL

- Função número 15
- Somente chamada no DOS 3.xx, se o bit 11 do atributo do tratador estiver ligado
- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho

- Descrição

  Esta função deve retornar no bit de ocupado do resultado se a unidade referenciada no cabeçalho é de disco removível (bit desligado) ou de disco fixo (bit ligado).

  É acessível ao programador de utilitários, através dos serviços do DOS. A informação obtida permite, por exemplo, dar mensagens diferentes nos dois casos ("Coloque na unidade X: o disco que contém o arquivo" ao invés de "Arquivo não encontrado na unidade X:" etc.).

3.5.12 IOCTL GENÉRICO

- Função número 19
- Somente chamada no DOS 3.2, se o bit 0 do atributo do tratador estiver ligado
- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho
0Dh	1 byte	08h
0Eh	1 byte	função
0Fh	2 words	reservado (conteúdo de SI e DI)
13h	1 dword	endereço do bloco de parâmetros

- Descrição

  Esta função permite o acesso a uma unidade de disco trilha a trilha. Os parâmetros e funções são focalizados no Capítulo 7, na descrição da função 44h do DOS (IOCTL).

3.5.13 INFORMA UNIDADE LÓGICA E ALTERA UNIDADE LÓGICA

- Funções número 23 e 24
- Somente chamada no DOS 3.2, se o bit 6 do atributo do tratador estiver ligado
- Formato da requisição:

deslocamento	tamanho	conteúdo
00h	13 bytes	cabeçalho
0Dh	1 byte	unidade

- Descrição

  Estas funções são utilizadas quando existem duas unidades lógicas associadas a uma unidade física (por exemplo, quando temos uma única unidade de disquete, que corresponde a A: e B:). Na saída "unidade" deve indicar qual a unidade lógica presente na unidade física especificada. Veja também a descrição da função 44h do DOS (IOCTL) no Capítulo 7.

3.6 O DISPOSITIVO CLOCK$

Conforme já vimos, um dos bits do atributo de um tratador permite indicar se ele trata o dispositivo de relógio, utilizado pelo DOS para obter a data e hora atuais. Nos tratadores padrão do DOS, este dispositivo tem o nome CLOCK$. Este tratador suporta o relógio não volátil do PC-AT, porém no PC e no PC-XT utiliza um contador mantido pelo BIOS (zerado a cada iniciação) para manter a hora e variáveis internas para manter a data.

Diante disso, muitas vezes é interessante substituí-lo por um tratador instalável que suporte um relógio não volátil (opcional na maioria das placas multifunção).

O tratador do dispositivo CLOCK se assemelha a um tratador normal de caracter, exceto que nas leituras e escritas são sempre transferidos 6 bytes com o seguinte formato:

deslocamento	tamanho	conteúdo
0	1 word	número de dias desde 1/1/80
2	1 byte	minutos (0 a 59)
3	1 byte	horas (0 a 23)
4	1 byte	centésimos de segundos (0 a 99)
5	1 byte	segundos (0 a 59)

A escrita no dispositivo permite acertar o relógio calendário; a leitura fornece a data e hora atuais.

3.7 O TRATADOR ANSI.SYS

Uma das grandes decepções do programador DOS é a pobreza de funções oferecidas pelo console padrão: apenas retorno do carro (CR), mudança de linha (LF), recuo do cursor (BS) e sinal sonoro (BEL). Os demais caracteres são simplesmente escritos na tela do PC. Isto obriga o programador a acessar o vídeo através do BIOS para obter efeitos simples como limpeza de tela e posicionamento de cursor.

Procurando resolver este problema, a Microsoft incluiu no disquete do DOS (a partir da versão 2.00) um tratador instalável de nome ANSI.SYS que substitui o tratador padrão de console, acrescentando vários comandos adicionais.

O ANSI.SYS oferece comandos poderosos com portabilidade - um fabricante de micros "quase compatíveis" pode criar a sua versão do ANSI.SYS que permita executar os mesmos comandos que o ANSI.SYS da IBM. O preço pago é a memória ocupada pelo tratador e, principalmente, a performance. A existência do DOS, do tratador ANSI e do BIOS entre o programa aplicativo e o vídeo é muitas vezes fatal.

Por outro lado, para aplicações que façam pouco acesso ao vídeo e principalmente para as escritas em linguaguens com difícil acesso ao BIOS, o tratador ANSI é uma boa solução.

O acesso a todos os recursos do tratador é feito através de sequências de controle escritas no tratador como caracteres normais. Todas as sequências são iniciadas pelo caracter ESC (01Bh) seguido do caracter '[' (05Bh).

Na notação aqui usada, que é a mesma do padrão ANSI ("*American National Standards Institute*" - Instituto Nacional Americano de Padrões), utilizaremos o símbolo # para indicar um parâmetro numérico composto por dígitos decimais em ASCII (por exemplo '2', '02', '102', etc). Caso seja especificado zero ou o valor seja

omitido da sequência, um valor padrão é assumido para cada função. As linhas e colunas do vídeo são numeradas a partir de 1.

Função: posiciona cursor
Sequência: ESC [# ; # H ou
ESC [# ; # f
Descrição: move o cursor para a posição especificada. O primeiro parâmetro é a linha (padrão = 1) e o segundo é a coluna (padrão = 1).

Função: sobe cursor
Sequência: ESC [# A
Descrição: sobe o cursor o número de linhas especificado (padrão = 1), sem mudar de coluna. Pára na primeira linha do vídeo.

Função: desce cursor
Sequência: ESC [# B
Descrição: desce o cursor o número de linhas especificado (padrão = 1), sem mudar de coluna. Pára na última linha do vídeo.

Função: avança cursor
Sequência: ESC [# C
Descrição: avança o cursor o número de colunas especificado (padrão = 1), sem mudar de linha.

Função: recua cursor
Sequência: ESC [# D
Descrição: recua o cursor o número de colunas especificado (padrão = 1), sem mudar de linha.

Função: informa posição do cursor
Sequência: ESC [6 n
Descrição: após a escrita desta sequência, será lido do console a resposta no formato
ESC [# ; # R
onde o primeiro parâmetro é a linha e o segundo a coluna.

Função: salva posição do cursor
Sequência: ESC [s
Descrição: salva a posição do cursor, para posterior restauração.

Função: restaura posição do cursor
Sequência: ESC [u

Descrição: move o cursor para a posição que ele ocupava quando recebeu o último comando "salva posição do cursor".

Função: limpa a tela
Sequência: ESC [2 J
Descrição: limpa toda a tela e posiciona o cursor no início (linha 1 coluna 1)

Função: limpa linha a partir do cursor
Sequência: ESC [K
Descrição: limpa a tela, do cursor (inclusive) até o fim da linha.

Função: seleciona atributos
Sequência: ESC [# ; ... ; # m
Descrição: seleciona o atributo a ser usado na escrita dos próximos caracteres. Os seguintes parâmetros são aceitos:

0 normal (branco sobre preto)
1 intenso
4 sublinhado (placa monochrome apenas)
5 piscante
7 reverso
8 invisível
30 frente preta
31 frente vermelha
32 frente verde
33 frente amarela
34 frente azul
35 frente magenta
36 frente cian
37 frente branca
40 fundo preto
41 fundo vermelho
42 fundo verde
43 fundo amarelo
44 fundo azul
45 fundo magenta
46 fundo cian
47 fundo branco

Função: seleciona modo

Sequência: ESC [= # h ou

ESC [? 7 h

Descrição: seleciona modo do vídeo:

0	40 x 25 alfanumérico preto & branco
1	40 x 25 alfanumérico colorido
2	80 x 25 alfanumérico preto & branco
3	80 x 25 alfanumérico colorido
4	320 x 200 gráfico colorido
5	320 x 200 gráfico branco & preto
6	640 x 200 gráfico branco & preto
7	liga mudança automática de linha após escrita na última coluna

Função: de-seleciona modo

Sequência: ESC [= # l ou

ESC [? 7 l

Descrição: idêntico a seleciona modo, exceto que parâmetro 7 desliga mudança automática de linha

Função: redefine teclado

Sequência: ESC [# ; # ; ... p ou

ESC [# ; " string " ; ... p

Descrição: redefine uma sequência do teclado, associando um ou mais códigos a ela. O primeiro código define a sequência que está sendo redefinida, pode ser um código ASCII de 1 a 255 ou um 0 seguido de um segundo código de 0 a 255. No segundo caso trata-se de um código "ASCII estendido" (ver Apêndice C). Os demais códigos definem uma sequência que será gerada quando o teclado gerar a primeira. Vejamos alguns exemplos:

converter 'a' em 'z':

código de 'a' em ASCII = 97 decimal

código de 'z' em ASCII = 122 decimal

sequência: ESC [9 7 ; 1 2 2 p

converter F1 em DIR B: <CR>

código de F1 = 0, 59 ("ASCII estendido")

código de CR em ASCII = 13 decimal

sequência: ESC [0 ; 5 9 ; " D I R " ; 1 3 p

3.8 O TRATADOR VDISK.SYS

O tratador VDISK.SYS é um tratador de dispositivo de bloco que simula um disco na memória do PC.

Originalmente (na versão 2 do DOS), este tratador era somente um exemplo, sendo fornecida uma listagem (com pequenos erros) no manual do DOS. A Microsoft deixava a cargo do interessado digitar o programa e assemblá-lo.

Na versão 3 a IBM passou a fornecer no disquete do DOS tanto a listagem como o arquivo final, pronto para ser carregado. Foram também acrescentadas algumas características importantes: os parâmetros do disco (tamanho do setor, do diretório e memória total ocupada) passaram a ser alteráveis na linha do comando DEVICE e a memória do PC-AT, acima do primeiro megabyte, é suportada.

A título de ilustração, vejamos como estes tratadores implementam as funções necessárias:

INICIAÇÃO

Analisa os parâmetros (DOS 3) e "formata" a memória correspondente ao disco, gravando os setores de BOOT, tabela de alocação e dire-tório.

TESTA TROCA de DISCO

CONSTROI BPB

INFORMA se REMOVIVEL

O disco simulado em memória é um disco fixo, tendo a sua BPB montada na iniciação (DOS 3) ou fixa (DOS 2).

LEITURA

Consiste em copiar os dados do endereço passado pelo DOS para a área correspondente aos setores no disco em memória. No caso de memória normal, a transferência é feita por REP MOVSB; no caso da memória de expansão do PC-AT, utiliza-se uma função do BIOS.

ESCRITA

ESCRITA COM VERIFICACAO

Consistem em copiar os dados da área correspondente aos setores no disco em memória para o endereço passado pelo DOS. No caso de memória normal, a transferência é feita por REP MOVSB; no caso da memória de expansão do PC-AT, utiliza-se uma função do BIOS.

Obs.: Na primeira versão do BIOS do PC-AT existia um erro de lógica que podia causar problemas quando da transferência de dados entre a memória normal e a expansão.

4 ARQUIVOS

4.1 O QUE É UM ARQUIVO

Um arquivo em disco é uma sequência de bytes, à qual associamos um nome. O DOS se encarrega de associar este nome a setores do disco. Conforme veremos adiante, o DOS utiliza duas estruturas para controlar os arquivos em disco: o diretório, que armazena os nomes dos arquivos, e a tabela de alocação, que controla quais setores estão alocados.

O DOS fornece funções para:

- CRIAR um arquivo

  colocando um nome no diretório

- RENOMEAR um arquivo

  alterando o nome no diretório

- SUPRIMIR um arquivo

  retirando o nome do diretório e liberando a área que ele ocupava

- ABRIR um arquivo

  localizando o arquivo a partir do nome fornecido pelo progra-ma

- LER dados de um arquivo

- ESCREVER dados em um arquivo

- etc.

Para o DOS, todos os arquivos são iguais: uma sequência de bytes. Não existe, como em outros sistemas, tipos de arquivo (binário, texto, executável, sequencial, randômico etc.). Estru-turas de arquivo mais sofisticadas são determinadas pelos progra-mas que as manipulam, não pelo DOS. Um exemplo desta filosofia são os arquivos executáveis .EXE (que serão dissecados no Capítulo 6); estes arquivos podem ser manipulados normalmente pelas funções de arquivo, porém, quando tratados pela função de carga de programa, os dados contidos no seu início são inter-pretados como um cabeçalho que contém informações sobre como deve se proceder a carga.

A flexibilidade no acesso de arquivos é obtida pela existência de facilidades para executar acessos do tipo "ler x bytes a partir do i-ésimo", isto é, o DOS permite o uso de registros variáveis (os "x bytes") e o acesso direto (no caso ao i-ésimo byte e seguintes).

O DOS mantém ainda várias informações sobre um arquivo, como data e hora da última atualização, tamanho etc. Em particular merece uma discussão mais profunda a questão do tamanho do arquivo.

Conforme dito no Capítulo 1, o DOS teve como ponto de partida a compatibilidade com as funções do CP/M-80, visando facilitar a conversão de programas já existentes. Uma característica do CP/M-80 é que o programador só acessa arquivos em blocos de 128 bytes. Desta forma, todo arquivo tem um tamanho

múltiplo de 128. Ora, normalmente as informações que queremos armazenar não têm tamanho múltiplo de 128 bytes. Isto obrigou os programadores a adotarem convenções externas ao CP/M para marcarem o fim real dos dados. A convenção mais difundida para arquivos texto consiste em tratar o caracter Control-Z (1Ah) como marca de fim de arqui-vo. Esta convenção cria alguns problemas, mesmo ao nível do CP/M. Por exemplo, o programa de cópia de arquivos, o PIP ("Peripheral Interchange Program") adota esta convenção, exceto se na linha de comando for colocada a opção [O] (de objeto). O operador novato (e o experiente) acaba esquecendo de usar a opção ao copiar arquivos não texto, e o resultado é que o arquivo pode acabar sendo copiado parcialmente.

O DOS apresenta um grande avanço, permitindo a leitura e escrita de blocos de 1 byte a 64K - 1 bytes com uma única chamada. O tamanho do arquivo é mantido a nível de byte, tornando desnecessária a existência de marcas especiais.

Entretando, vários programas trouxeram do CP/M a convenção do Control-Z, o que cria uma pequena torre de babel, existindo programas que:

- acrescentam um único Control-Z no fim dos arquivos;
- gravam arquivos sempre de tamanho múltiplo de 128, com o último registro completado com Control-Z;
- idem, porém só colocam um Control-Z como marca de final, completando o registro com "lixo";
- não consideram o Control-Z como marca de fim de arquivo, tentando tratá-lo como dado;
- exigem arquivos com tamanho múltiplo de 128, com Control-Z marcando o final dos dados no último registro;
- e assim por diante.

Uma complicação adicional é que o DOS permite acessar de forma semelhante a arquivos em disco os dispositivos de caracter (NUL, CON, PRN etc.). Como estes dispositivos não têm um tamanho definido, o DOS normalmente considera o Control-Z como fim de arquivo para eles. Por exemplo, a leitura do console (dispositivo CON) é finalizada pela digitação do Control-Z e a escrita no console (via funções de arquivo) é terminada ao se encontrar um Control-Z nos dados. Um programa que tente acessar indistintamente arquivos em disco e dispositivos pode se complicar. Por exemplo, alguns programas quando escrevem no console um registro com um control-z no meio consideram o fato do registro ser gravado parcialmente como erro de disco cheio! Se isto tudo não bastasse, o DOS 2.0 apresenta problemas quando tentamos escrever outros registros no dispositivo após o que continha Control-Z. Experimente usar o SORT do DOS 2.0 para ordenar um arquivo texto do WordStar (que usa Control-Z) e mostrar o resultado (SORT < ARQUIVO); você será obrigado a digitar Ctrl-Alt-Del.

Algumas regras simples podem tornar o seu programa mais compatível com os diversos formatos de arquivo texto:

1. na leitura, considere o primeiro Control-Z como fim de arquivo, ignorando os caracteres seguintes;
2. na escrita, forneca a opção de colocar Control-Z como marca de final de arquivo;
3. na escrita, quando o DOS indicar que escreveu um número menor do que você mandou escrever, verifique se é um arquivo em disco ou dispositivo; no primeiro caso ocorreu disco cheio.

Obs.: O que são arquivos texto ? É bastante comum classificar-se os arquivos em arquivos texto e arquivos binário. Os arquivos textos normalmente contêm apenas os caracteres ASCII

alfanuméricos (20h a 7Eh) e delimitadores de fim de linha CR ("Carriage Return" - 0Dh) e LF ("Line Feed" - 0Ah), enquanto que os binários podem conter qualquer código de 00h a FFh. A limitação dos caracteres válidos torna os arquivos textos mais transportáveis entre diversos equipamentos e sistemas.

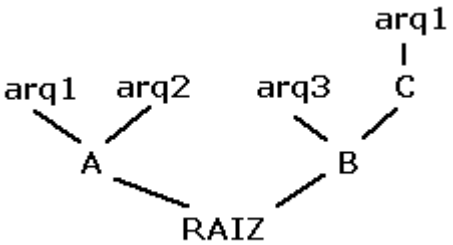
4.2 DIRETÓRIOS E SUBDIRETÓRIOS

O ponto de partida para a localização de um arquivo em disco é o diretório, que é uma tabela que contém nomes de arquivos e informações sobre eles.

Todo disco utilizado pelo DOS possui uma região reservada para o diretório base, mais conhecido como diretório raiz. A partir da versão 2, o DOS passou a permitir que, além de informações sobre arquivos, um diretório pudesse conter:

- uma identificação de volume ("volume label")
- informações sobre outros diretórios (os subdiretórios)

Um subdiretório, por sua vez, pode conter informações sobre arquivos e mais subdiretórios, assim sucessivamente. Esta estrutura se assemelha a uma árvore, com o diretório base sendo a raiz e os arquivos as folhas:



diretório	subdiretórios	arquivos
RAIZ	A B	-
A	-	arq1 arq2
B	-	arq3
C	-	arq1

São as seguintes as diferenças entre o diretório raiz e os subdiretórios:

RAIZ

- posição e tamanho definidos na formatação
- pode conter identificação de volume

SUBDIRETÓRIO

- alocado dinamicamente; tamanho limitado pela área livre em disco
- contém duas entradas especiais:

  "." - aponta para o próprio subdiretório

  ".." - aponta para o diretório anterior (pai)

4.3 NOMES DE ARQUIVOS E SUBDIRETÓRIOS

Para se selecionar um arquivo é necessário especificar:

- a unidade de disco em que ele se encontra
- o diretório em que ele está descrito
- o seu nome dentro do diretório

A unidade de disco é especificada por uma letra seguida de ":" ("A:", "B:" etc.). Caso a unidade não seja especificada, o DOS assume a unidade chamada padrão. A unidade padrão pode ser selecionada pelo operador ou por um programa.

O diretório é descrito completamente através de um caminho ("*path*") que parte do diretório raiz e termina no diretório em que o arquivo está descrito. O DOS mantém, para cada unidade de disco, uma informação de diretório atual ("*current directory*"), semelhante à informação de unidade padrão.

Os nomes de arquivos e subdiretórios são compostos de duas partes: um nome principal ("*basename*"), com 1 a 8 caracteres, e uma extensão ou sufixo ("*extension*"), que pode estar ausente ou ter até três caracteres. Estas duas partes são separadas por um ".". Os caracteres válidos para nomes são:

os números '0' a '9' (códigos ASCII 30h a 39h)

as letras 'A' a 'Z' (códigos ASCII 41h a 5Ah) e

'a' a 'z' (códigos ASCII 61h a 7Ah)

os símbolos '!' (21h) '#' (23h) '$' (24h) '%' (25h)

'&' (26h) '(' (28h) ')' (29h) '-' (2Dh)

'_' (5Fh) '`' (60h) '{' (7Bh) '}' (7Dh)

'~' (7Eh)

As letras minúsculas são convertidas pelo DOS em maiúsculas; a partir da versão 3, o DOS passou a suportar os caracteres acima de 80h em nomes, procurando converter as letras minúsculas acentuadas nas suas correspondentes maiúsculas. Através da passagem de parâmetros inválidos ou da escrita direta em disco (sem usar as funções de arquivo) é possível criar nomes que contenham outros caracteres. Estes arquivos poderão ou não ser acessados de forma confiável pelas funções do DOS e utilitários - não é muito aconselhável realizar experimentos em discos com arquivos importantes.

Vejamos alguns nomes normais:

ARQUIVO

ARQUIVO.MEU

1

1.1

ETC.TAL

Um ponto a frisar é que os nomes de subdiretórios e arquivos seguem as mesmas regras; em particular ambos podem ter extensão (o uso de nomes de subdiretório com extensão não é comum).

Vejamos agora como se especifica um caminho. O diretório raiz é representado por uma barra reversa "\", e os nomes dos subdire-tórios são também separados por "\"s. Um caminho que comece por "\" parte do diretório raiz; caso contrário parte do diretório atual. Podemos usar os símbolos "." e ".." para indicar o diretório atual e anterior em um caminho. Retomando a árvore de diretório, e considerando que o diretório atual seja "\B", temos:

Especificação	caminho completo
arq3	\B\arq3
\A\arq1	\A\arq1
..\A\arq1	\A\arq1
C\arq1	\B\C\arq1

e assim por diante.

O uso de subdiretórios permite organizar melhor um disco (principalmente um de grande capacidade). Os nomes podem, entretanto, ficar um pouco grandes:

unidade 2 caracteres (u:)

caminho segundo o manual do DOS, até 64 caracteres

nome 8 (principal) + 3 (extensão) + 1 (".")

TOTAL: 78 caracteres !

Nas chamadas ao DOS (mas não em todos os utilitários) é permitido o uso de '/' ao invés de '\', conforme o usado no sistema UNIX.

Os caracteres '?' e '*' são aceitos em nomes de arquivos por várias funções do DOS, com os seguintes significados:

'?' indica que qualquer caracter pode ocupar a sua posição

'*' indica que qualquer caracter pode ocupar a sua posição e as seguintes até o final do nome (principal ou extensão) em que ele estiver

Desta forma obtemos uma especificação múltipla de arquivo, isto é, uma especificação que se refere simultaneamente a mais de um arquivo. Vamos denominar estas especificações de ambíguas. Alguns exemplos:

XIS.* referencia todos os arquivos de nome principal XIS

XIS.? referencia os arquivos de nome principal XIS e que não têm extensão ou têm extensão com um caracter

* referencia todos os arquivos que não têm extensão (obs.: o utilitário DIR do DOS transforma esta especificação em *.*; isto é uma característica do utilitário e não do DOS)

. referencia todos os arquivos da unidade

Uma facilidade ausente no DOS é o tratamento de nomes ambíguos no caminho; apenas o último nome pode conter os caracteres '*' e '?'.

4.4 CONTROLE DE ALOCAÇÃO: CLUSTERS E FATS

Uma das principais funções do DOS é controlar a alocação dos discos, ou seja, quais setores estão livres e quais estão associados a arquivos.

A unidade básica de alocação do DOS é o cluster, que é um múltiplo de setor (obrigatoriamente potência de dois: 1, 2, 4, 8 etc, setores). A anotação do uso dos clusters é feita na tabela de alocação de arquivos, a FAT ("*File Allocation Table*"). Esta tabela, da qual existem normalmente duas cópias, contém uma entrada para cada cluster do disco.

O ponto de partida para um acesso a um arquivo é um diretório ou subdiretório. Nele o DOS encontra (entre outras informações) o tamanho (em bytes) do arquivo e o número do primeiro cluster alocado a ele. Conforme veremos adiante, estas informações são extraídas quando da abertura do arquivo e atualizadas somente no seu fechamento. Desta forma, o diretório não é acessado nas operações de leitura e escrita.

As entradas na FAT constituem uma lista ligada: a entrada correspondente ao cluster inicial (apontado pelo diretório) contêm o número do segundo cluster, cuja entrada na FAT "aponta" para o terceiro cluster e assim por diante. Valores especiais na FAT indicam o último cluster de um arquivo, clusters livres e clusters que contém setores defeituosos. A Figura 3 resume o significado do diretório e da FAT.

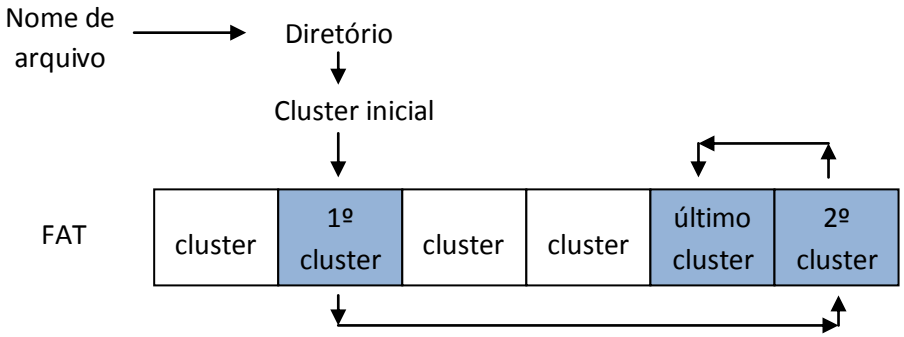


Figura 3: Diretório e FAT

A inclusão de novos dados em um arquivo é feita através da alocação de novos clusters ao final da cadeia; a supressão de um arquivo é feita marcando o arquivo no diretório como suprimido (permitindo o uso para novos arquivos da entrada correspondente) e marcando na FAT como livres os clusters que ele ocupava.

Antes de prosseguirmos, cabe aqui um parênteses quanto à recuperação de arquivos suprimidos acidentalmente. A operação de supressão deixa o diretório praticamente intacto (como veremos com mais detalhes, apenas o primeiro caracter do nome é apagado), porém apaga toda a cadeia de clusters mantida na FAT. Desta forma, é imediata a localização do primeiro cluster, porém complicada a localização dos demais.

Potencialmente, todo cluster marcado como livre após a supressão pode ter pertencido ao arquivo. A tendência normal do DOS é a alocação de clusters consecutivos, porém se dois arquivos estiverem sendo escritos alternadamente ou se um disco continha vários arquivos pequenos que foram suprimidos (deixando várias regiões livres isoladas) um arquivo pode estar espalhado por todo o disco. No caso de um

arquivo binário, cujos dados não possam ser facilmente reconhecidos, a determinação de quais clusters (e em que ordem!) pertenciam ao arquivo é um trabalho capaz de levar qualquer um à insanidade.

Obs.: Esta situação é diferente da do sistema CP/M-80, no qual a informação de alocação é armazenada no próprio diretório e permanece inalterada após a supressão; neste caso, a recuperação do arquivo é trivial.

O número de setores por cluster depende do tipo de disco. A escolha deste parâmetro é um compromisso entre duas considerações:

- quanto menor o cluster, menor o espaço desperdiçado em fragmentação, isto é, espaço não utilizado no final do arquivo (nos arquivos grandes a tendência é o desperdício de meio cluster em média; existem porém os arquivos pequenos - arquivos "batch", pequenos programas etc, - que podem desperdiçar bem mais);
- por outro lado, um cluster grande diminui o tamanho da FAT, tornando mais provável a presença dos seus setores nos buffers de disco. Durante a leitura e escrita em arquivos, o DOS acessa frequentemente a FAT, seja para localizar o próximo cluster ou para achar um cluster livre e acrescentá-lo ao final da cadeia. A presença dos setores da FAT na memória é fundamental para um bom desempenho do DOS. Na versão 1 o tamanho das FATs dos disquetes era de 1 setor e existia um buffer para cada unidade destinado especificamente para o armazenamento da FAT.

A tendência da IBM tem sido no sentido de clusters menores e FAT maiores para as unidades mais recentes de disco. Esta política exige que um maior número de buffers seja configurado (o que consome mais memória, obrigando a compra de configurações maiores e mais caras).

Vejamos agora como é estruturada a FAT. Nas versões 1 e 2 do DOS, cada entrada ocupava 12 bits, ou seja, cada duas entradas consecutivas ocupava 3 bytes, conforme indicado abaixo:

entrada par	p2 p1 p0
entrada ímpar	i2 i1 i0
armazenamento na FAT	p1|p0 i0|p2 i2|i1
	(p2, p1, p0, i2, i1 e i0 são grupos de 4 bits)

A versão 3 passou a utilizar 16 bits por entrada, quando o disco contém mais de 4086 clusters de dados. Neste caso, o armazenamento é mais direto:

Conteúdo da entrada	x3 x2 x1 x0
Armazenamento	x0 x1 x2 x3

O DOS adota a convenção de que o número do primeiro cluster de dados é 2, o que libera duas entradas na FAT (referentes ao que seriam os clusters "0" e "1") para conter outras informações. O primeiro byte da FAT contém o tipo de disco (o "media descriptor byte"); os bytes restantes das duas primeiras entradas são preenchidos com 0FFh.

Os valores usados para os formatos padrão são:

FFh disquete 5 1/4, face dupla, 8 setores por trilha

FEh disquete 5 1/4, face simples, 8 setores por trilha

FDh disquete 5 1/4, face dupla, 9 setores por trilha

disquete 3 1/2, face dupla, 9 setores por trilha

FCh disquete 5 1/4, face dupla, 15 setores por trilha

F8h disco fixo ("winchester")

A obtenção do conteúdo da entrada i da FAT é obtida da seguinte forma:

- FAT com entradas de 12 bits

  1) Calcule o endereço do primeiro byte da entrada, em relação ao início da FAT: some i com i deslocado um bit para a direita (ou seja, multiplique i por 1.5 e despreze a parte fracionária). Vamos chamar de "X" o valor do word que começa neste endereço.
  2) Se i for par, o conteúdo desejado é obtido pelo "and" de X com 0fffh; se i for ímpar, o conteúdo desejado é obtido deslocando X 4 bits para a direita.

- FAT com entradas de 16 bits

  O conteúdo é o word de endereço inicial, em relação ao início da FAT, 2 vezes i.

Supondo que BUF_FAT seja uma área de memória que contenha toda a FAT, as rotinas a seguir retornam em AX o conteúdo da entrada na FAT referente ao cluster cujo número está em BX:

```
; 1. caso - entradas de 16 bits

        SHL     BX,1
        MOV     AX,BUF_FAT [BX]
        RET

; 2. caso - entradas de 12 bits

        MOV     SI,BX
        SHR     SI,1
        MOV     AX,BUF_FAT [BX+SI]
        JC      IMPAR
        AND     AX,0FFFFH
        RET
IMPAR:
        MOV     CL,4
        SHR     AX,CL
        RET
```

Uma entrada da FAT pode conter os seguintes valores:

(0)000indica	cluster livre
(F)FF0 a (F)FF6	indica cluster reservado
(F)FF7	indica cluster com setor defeituoso
(F)FF8	indica último cluster de um arquivo
(F)FF9 a (F)FFF	são marcas alternativas de fim de arquivo (não usadas normalmente)
outros valores	número do cluster seguinte no arquivo

(X) indica que X só aparece nas FATs de 16 bits

O utilitário CHKDSK realiza a consistência do conteúdo da FAT, detectando os seguintes erros:

- Clusters marcados como alocados, porém não associados a arquivos ("*lost clusters found*")

  Isto ocorre quando um arquivo não é fechado corretamente após a escrita; o DOS pode atualizar a FAT durante a escrita porém só atualiza o diretório quando o arquivo é fechado. Desta forma os clusters alocados estão ligados entre si, porém não estão ligados ao diretório.

- Tamanho no diretório não concorda com o número de clusters alocados ("*Allocation error, size adjusted*")

  Pode ocorrer pelo mesmo motivo acima.

- Número de cluster inválido em uma cadeia ("*First cluster number is invalid*", "*Has invalid cluster*")

  Pode ocorrer se um disco for trocado em momento impróprio, ocasionando a gravação da FAT ou Diretório no disco errado.

- Cluster alocado simultaneamente a dois arquivos ("*File is cross-linked*")

  Idem

A alocação de clusters consecutivos a um arquivo permite o seu acesso de forma mais rápida. Já a alocação fragmentada (isto é, clusters não consecutivos) obriga várias chamadas ao tratador de disco, a movimentação da cabeça de leitura etc.

Por estes movivos, o DOS procura alocar clusters consecutivos. Nas versões 1 e 2 o DOS procurava clusters livres sempre a partir do início da FAT; a partir da versão 3 os clusters livres passaram a ser procurados a partir do último alocado.

4.5 ESTRUTURA DE DIRETÓRIOS E SUBDIRETÓRIOS

A anotação de uma entrada no diretório raiz ou subdiretório ocupa 32 bytes, tendo o seguinte formato:

desloc.	tamanho	conteúdo
0	11 bytes	nome do arquivo/subdiretório/rótulo primeiro byte indica o estado da entrada: 00h entrada nunca usada; o DOS assume que todas as entradas seguintes também estão livres, encerrando buscas no diretório ao encontrar a primeira com esta marca [2] 05h indica que o primeiro caracter do nome é E5h [3] E5h indica entrada que já foi usada e posteriormente liberada demais valores correspondem ao primeiro caracter do nome
11	1 byte	tipo (atributo) da entrada: bit 0 = 1 indica entrada protegida ("read only"); o DOS não permite supressão ou escrita. [2] bit 1 = 1 entrada "escondida", não é acessível pelas buscas comuns no diretório. bit 2 = 1 arquivo/diretório de sistema, não é acessível pelas buscas comuns de diretório. bit 3 = 1 entrada descreve rótulo do disco ("*volume label*"). Somente uma entrada pode ter este tipo e tem que estar no diretório raiz. [2] bit 4 = 1 entrada descreve subdiretório. [2] bit 5 = 1 entrada foi alterada. Este bit é usado por utilitários de "back-up" para detectar quais arquivos foram alterados desde a sua última execução. Os bits 3 e 4 são mutuamente exclusivos (não podem ser 1 simultaneamente). Os bits 1 e 2 não têm significado para rótulo, porém podem ser usados para subdiretórios. Os bits 0, 1, 2 e 5 podem ser ativados/desativados através de função do DOS.
12	10 bytes	reservados (atualmente sem uso)
22	2 bytes	hora da criação ou última atualização da entrada: bits 0 a 4 segundos divididos por 2 bits 5 a 10 minutos bits 11 a 15 hora
24	2 bytes	data da criação ou última atualização da entrada: bits 0 a 4 dia bits 5 a 8 mês bits 9 a 15 ano (0 a 119, corresponden-do a 1980 a 2099)
26	2 bytes	número do primeiro cluster do arquivo
28	4 bytes	tamanho do arquivo em bytes,; se entrada não corresponde a arquivo, contém zero

[2] e [3] indicam, respectivamente, itens que foram introduzidos nas versões 2 e 3.

O diretório raiz tem, como já vimos, tamanho e localização fixos. Já os subdiretórios são alocados de forma idêntica a arquivos, ocupando portanto clusters de dados.

A principal diferença entre uma entrada que descreve um arquivo e uma que descreve um subdiretório (afora o atributo) é que a segunda tem o campo de tamanho sempre igual a zero. Isto impede que subdiretórios sejam acessados pelas funções normais de arquivos, ao contrário do que ocorre no sistema UNIX e derivados. Através do DOS, o programador acessa um diretório como uma "caixa preta", podendo criar, suprimir, alterar e procurar entradas sempre através de nomes (que na maioria dos casos podem ser ambíguos).

Atualmente todos os discos utilizados no PC tem setores de 512 bytes, que é múltiplo de 32. Na eventualidade (pouco provável) de se utilizar um setor de tamanho não múltiplo de 32 o DOS colocará sempre um número inteiro de entradas por setor, desperdiçando os bytes restantes.

O diretório raiz é iniciado na formatação, sendo o primeiro byte de cada entrada preenchido com E5h na versão 1 e 00h nas demais. As outras posições das entradas são normalmente preenchidas com F6h (valor que é gravado nos setores pelo operação de formatação física).

Os subdiretórios são alocados, e portanto iniciados, conforme necessário. O DOS aloca um cluster de cada vez, preenchendo-o com zeros.

Uma diferença importante entre um subdiretório e o diretório raiz é a presença de duas entradas especiais no início do subdiretório. A primeira possui o nome '.' e aponta (através do cluster inicial) para o próprio diretório. A segunda tem o nome '..' e aponta para o diretório "pai", isto é, aquele no qual o subdiretório foi criado. Caso o diretório pai seja o raiz, o cluster inicial de '..' é marcado como zero (lembre-se, os clusters 0 e 1 não existem).

4.6 TABELA DE PARTIÇÃO DE DISCO RÍGIDO

Os micros compatíveis como PC-IBM permitem que um disco rígido ("winchester") seja dividido em partições. Cada partição consiste em um conjunto de trilhas consecutivas destinado a um sistema operacional. Idealmente, cada sistema operacional deve tratar a sua partição como sendo uma unidade de disco inteira, "escondendo" as demais partições.

O controle das partições é mantido no carregador principal ("master boot"), que reside no primeiro setor do disco e é carregado pelo BIOS. Dentro deste setor existe a tabela de partições e o programa carregador propriamente dito. A tabela de partições tem o seguinte formato:

desloc.	tamanho	conteúdo
1BEh	16 bytes	descrição da partição 1
1CEh	16 bytes	descrição da partição 2
1DEh	16 bytes	descrição da partição 3
1EEh	16 bytes	descrição da partição 4

onde cada descrição contém:

desloc.	tamanho	conteúdo
0	1 byte	indicador de partição ativa contém 80h na partição que será ativada pelo carregador principal e 00h nas demais
1	1 byte	face onde está o setor inicial da partição
2	1 word	setor e trilha iniciais da partição. O byte mais significativo contém os bits menos significativos do número da trilha e o byte menos significativo contém nos bits 0 a 5 o número do setor e nos bits 6 e 7 os bits mais significativos da trilha. Este formato é o utilizado nas chamadas ao BIOS.
4	1 byte	indicador de sistema operacional 1 corresponde a DOS, FAT com entradas de 12 bits; 4 a DOS, FAT com entradas de 16 bits.
5	1 byte	face onde está o setor final da partição
6	1 wod	setor e trilha finais da partição. O byte mais significativo contém os bits menos significativos do número da trilha e o byte menos significativo

		contém nos bits 0 a 5 o número do setor e nos bits 6 e 7 os bits mais significativos da trilha. Este formato é o utilizado nas chamadas ao BIOS.
8	1 dword	número de setores que antecedem a partição (é o número relativo do setor inicial)
12	1 dword	número de setores na partição

O programa contido no carregador principal procura a partição ativa, carrega o primeiro setor dela para a memória (em 0:7C00h) e passa o controle à sua primeira posição. O primeiro setor de cada partição deve, portanto, conter o carregador do sistema operacional nela contido.

Todas as partições começam no primeiro setor da primeira face de um cilindro, exceto a primeira partição, que começa no segundo setor (o primeiro contém o carregador principal).

O tratador padrão de discos do DOS consulta o BIOS para saber da presença de unidades de disco rígido e, em caso afirmativo, lê a tabela de partições para determinar o setor onde está o carregador do DOS (que contém a BPB). Os setores anteriores à partição do DOS são anotados na BPB como setores "escondidos".

Obs.: O suporte a disco rígido foi introduzido na versão 2 do PC-DOS, juntamente com o PC-XT. Existiam diversos esquemas para uso de winchesters com a versão 1, porém sem suporte oficial da IBM ou da MicroSoft.

4.7 RESUMO DAS ESTRUTURAS DE DISCO

Juntando o que foi visto nos itens anteriores com o que foi apresentado no Capíulo 3, podemos agora ter uma visão geral da organização de discos no DOS.

O primeiro setor do disco (da partição DOS, no caso de disco rígido) contém o carregador do sistema. A partir do segundo setor temos as duas cópias da FAT e o diretório raiz. Os setores seguintes correspondem a clusters de dados (ver figura 4).

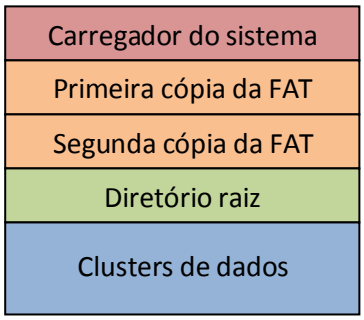


Figura 4: Localização das Estruturas de Disco

A tabela a seguir mostra os principais parâmetros dos tipos de disco suportados pelo DOS padrão:

Tamanho	5 ¼	5 ¼	5 ¼	5 ¼	5 ¼	3 ½
setores/trilha	8	8	9	9	15	9
# de trilhas	40	40	40	40	80	80
# de faces 1	2	1	2	2	2	
tipo de disco	FEh	FFh	FCh	FDh	F9h	FDh
setores/FAT	1	1	2	2	7	4
setores/diretório	4	7	4	7	14	7
# de entradas	64	112	64	112	224	112
setores/cluster	1	2	1	2	1	2
# de clusters	313	315	351	354	2371	712
Kbytes de dados	156,5	315	175,5	354	1185,5	712

No caso de discos rígidos a conta fica um pouco mais complicada. Apesar do carregador do DOS conter uma cópia da BPB, o DOS calcula os parâmetros a partir do tamanho da partição, utilizando para isto de 4 faixas:

Setores na partição	setores por cluster	entradas no diretório	entradas na FAT
de 64 a 511	1	64	12 bits
de 512 a 2047	2	112	12 bits
de 2048 a 8191	4	256	12 bits
de 8192 a 32679	8	512	12 bits
acima de 32679	16	1024	12 bits

A tabela acima se refere à versão 2 do DOS. Na versão 3 a última faixa foi alterada para:

Setores na partição	setores por cluster	entradas no diretório	entradas na FAT
acima de 32679	4	512	16 bits

A versão 3 verifica no carregador se a identificação de sistema é IBM 2 e , em caso afirmativo, utiliza a BPB no carregador ao invés dos valores calculados, permitindo assim o acesso a qualquer disco formatado na versão 2.

4.8 ACESSANDO ARQUIVOS

O programador que deseja acessar arquivos através do DOS dispõe de duas maneiras, uma compatível com o CP/M (as "tradicionais", na nomenclatura MicroSoft), presente em todas as versões, e outra, semelhante ao sistema UNIX, presente a partir da versão 2.

Bloco de controle de arquivo (FCB) e Área de transferência de disco (DTA)

O acesso a arquivos de forma compatível com o CP/M utiliza uma estrutura de dados, o bloco de controle de arquivo - o FCB ("*File Control Block*"), residente no programa do usuário. O formato de uma FCB é:

Deslocamento	tamanho	conteúdo
0	1 byte	unidade (1 = A:, 2 = B: etc.) na abertura pode-se colocar 0, que o sistema substitui pela unidade padrão
1	8 bytes	nome principal alinhado à esquerda e completado com brancos à direita
9	3 bytes	extensão do nome alinhado à esquerda e completado com brancos à direita
12	1 word	número do bloco atual em relação ao início do arquivo cada bloco contém 128 registros

14	1 word	tamanho do registro (em bytes) iniciado com 128 na abertura, pode ser alterado pelo usuário
16	1 dword	tamanho do arquivo em bytes
20	1 word	data de criação ou última atualização no mesmo formato do diretório
22	10 bytes	reservado ao DOS (ver adiante)
32	1 byte	registro no bloco atual número do registro (0 a 127) no bloco
33	4 bytes	registro relativo número do registro em relação ao início do arquivo. Este campo só é usado nas funções de leitura/escrita randômica e por blocos. Se o tamanho do registro for maior ou igual a 64 bytes, somente os três primeiros bytes são utilizados.

O conteúdo da região reservada ao DOS varia conforme a versão do sistema. As tabelas a seguir mostram, respectivamente, os conteúdos nas versões 2 e 3:

Deslocamento	tamanho	conteúdo (DOS 2)
22	1 word	hora da criação ou última atualização do arquivo, no mesmo formato do diretório
24	1 byte	informação sobre o dispositivo, no formato usado na função IOCTL do DOS
25	1 dword	endereço do cabeçalho do tratador
29	3 bytes	uso desconhecido

Deslocamento	tamanho	conteúdo (DOS 3)
22	1 word	hora da criação ou última atualização do arquivo, no mesmo formato do diretório
24	1 word	informação sobre o dispositivo, no formato usado na função IOCTL do DOS
26	1 dword	endereço do cabeçalho do tratador
30	2 bytes	uso desconhecido

Os diversos campos da FCB retratam a evolução dos sistemas CP/M e DOS:

1. Na versão 1 do CP/M-80 não existia o conceito de acesso randômico a um registro, sendo o acesso sequencial controlado pelos campos bloco e registro atual. O campo registro relativo não existia.
2. Na versão 2 do CP/M-80 surgiu o acesso randômico, com a introdução do campo registro relativo. Em ambas as versões o tamanho do registro era fixo em 128 bytes, com o registro relativo ocupando três bytes na FCB
3. No DOS o tamanho do registro passou a ser selecionável pelo programador, entre 1 e 64K-1 bytes.

Com o DOS foi também introduzido o conceito de "FCB estendida", que é uma FCB normal precedida de um cabeçalho com 7 bytes:

Deslocamento	tamanho	conteúdo
-7	1 byte	contém 0FFH para indicar FCB estendida
-6	5 bytes	reservados, preencher com zero
-1	1 byte	atributo do arquivo

Uma FCB normal permite acessar apenas os arquivos normais; a FCB estendida permite acessar arquivos invisíveis e/ou de sistema e rótulos de disco. Na prática as FCBs estendidas são pouco usadas e "saíram de moda" com a versão 2 do DOS.

A principal limitação da FCB é que com ela só podemos acessar o diretório corrente de uma unidade. A montagem de uma FCB a partir de um nome fornecido pelo operador é trabalhosa, porém o DOS possui uma função para isto e, inclusive, prepara FCBs para os dois primeiros parâmetros colocados na linha de comando. Uma vantagem do uso da FCB é que o limite do número de arquivos abertos através delas (desde que não estejamos compartilhando arquivos sob DOS 3) é dado pelo número de FCBs no programa (que a 36 bytes por FCB é quase infinito).

As funções compatíveis com o CP/M realizam todas as operações de entrada e saída sobre uma área de memória denominada Área de Transferência de Dados - DTA ("*Data Transfer Area*"). O programador deve, portanto, informar ao DOS o endereço da DTA antesde realizar uma leitura/escrita.

***Handles* e cadeias ASCIIZ**

O conceito de "*handles*" provém do sistema operacional UNIX. A idéia básica é o programa, ao abrir um arquivo, receber do sistema uma "chave" (o *handle*) que é usada no acesso ao arquivo. Desta forma as estruturas de controle ficam no sistema, sendo "invisíveis" ao usuário.

Na versão 2 do DOS, juntamente com o conceito de handles, foi introduzido o de "cadeias ASCIIZ" ("*ASCIIZ strings*"). Uma cadeia ASCIIZ é uma cadeia de caracteres (normalmente um nome de diretório ou arquivo) cujo fim é indicado por um byte contendo zero. Esta convenção para cadeias é a mesma da utilizada na linguagem "C".

No DOS, um handle é um número de 16 bits. Nas versões 2 e 3 o handle é, normalmente, um índice para uma tabela de 20 bytes contida em uma área denominada "PSP" (Prefixo de Segmento de Programa, que é discutido no Capítulo 6). Portanto, os valores normais para um handle são de 0 a 19. A tabela na PSP contém, por sua vez, um índice para outra tabela, que contém FCBs estendidas e algumas informações adicionais, interna ao DOS. Desta forma, todo acesso a arquivo é realizado (de forma direta ou indireta) através de FCBs.

O tamanho da tabela de FCBs, interna ao DOS, é determinado pelo comando "FILES" do arquivo CONFIG.SYS. Caso esta linha não esteja presente, o DOS utiliza uma tabela de oito entradas. Uma vez que a tabela na PSP contém 20 posições, um programa não pode, de forma simples, acessar mais de 20 arquivos simultaneamente.

Dois ou mais handles podem apontar para uma mesma entrada na tabela de FCBs, através de uma operação chamada "duplicação" de handle (DUP). Por exemplo, quando um programa ativa um programa "filho", com PSP própria, todos os handles abertos do "pai" são "herdados" pelo "filho". A Figura 5 dá uma ideia do que podemos ter em um determinado momento.

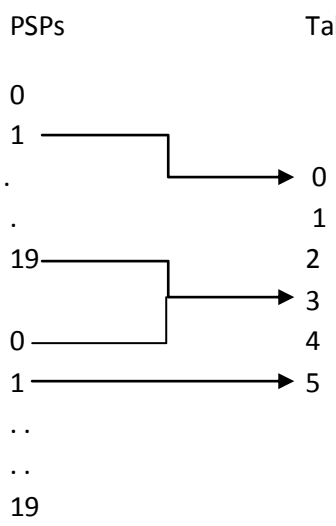


Figura 5: Handles e a tabela de FCBs

Na iniciação são abertos cinco handles especiais, que são "herdados" por todos os programas:

0 - entrada padrão

1 - saída padrão

2 - saída padrão para mensagens de erros

3 - dispositvo auxiliar padrão

4 - saída padrão em impressora

Normalmente os handles 0, 1 e 2 correspondem ao dispositivo CON, o handle 3 ao dispositivo AUX e o handle 4 ao dispositivo PRN. É possível, utilizando os caracteres ">" e "<" na linha de comando, redirecionar a saída e entrada padrão, associando-os a outros dispositivos ou a arquivos em disco.

Existe no DOS um conjunto de funções, presentes em todas as versões, destinado ao acesso ao console. Todas estas funções acessam o console através dos handles 0 e 1, suportando o seu redirecionamento. Entretanto, o DOS possui algumas limitações:

1. Ao se atingir o fim de um arquivo associado à entrada padrão, a chamada a uma função de leitura do console ficará "presa", não ocorrendo retorno ao console padrão nem a apresentação de mensagem de erro. Deve-se, portanto, ao redirecionar a entrada para um arquivo, tomar o cuidado de colocar nele todas as respostas esperadas pelo programa a ser executado
2. No caso de acabar o espaço livre no disco onde reside um arquivo associado à saída padrão, o DOS também não efetua o retorno ao console padrão ou apresentação de mensagem de erro.

As funções de leitura/escrita através de handle recebem explicitamente como parâmetros o número de bytes a serem transferidos e o endereço de transferência.

As versões mais recentes do DOS enfatizam o uso de handles, simultaneamente ao desaconselhamento do uso de FCBs. As vantagens no uso de handles são:

- funções de entrada/saída mais simples
- acesso simplificado a subdiretórios, sendo aceitos diretamente nomes completos
- acesso simplificado a arquivos invisíveis e de sistema

Por outro lado, temos as seguintes desvantagens:

- limite de 20 handles abertos simultaneamente por programa
- necessidade do uso do comando FILES para poder ter mais de oito handles diferentes abertos simultaneamente no sistema (o COMMAND tem sempre cinco handles abertos, três dos quais normalmente ocupando uma única entrada na tabela de FCBs)
- necessidade de manuseio de cadeias ASCIIZ mesmo nos programas mais simples, visto o DOS não preparar convenientemente os dois primeiros parâmetros da linha de comando
- necessidade de chamada ao DOS para movimentar o ponteiro de leitura/escrita, obter o tamanho do arquivo etc.

4.9 ACESSO A DISPOSITIVOS

Uma das características importantes do DOS é a possibilidade de acessar dispositivos (de caractere) como se fossem arquivos. Para isto basta fornecer como nome do arquivo o nome do dispositivo (unidade, diretório e extensão do nome são ignorados).

O nome do dispositivo nãodeve ser seguido de ':' (usar "CON" ao invés de "CON:" etc.). O ':' é aceito pelos utilitários (não pelo núcleo do DOS) por compatibilidade com o CP/M.

Existem duas diferenças entre o acesso a dispositivos e a arquivos:

1. Não existe acesso randômico a dispositivos; os dados são sempre lidos/escritos sequencialmente, independente da função do DOS utilizada.
2. Os arquivos são sempre acessados em modo binário; os dispositivos são normalmente acessados em modo ASCII. O modo de acesso a um dispositivo pode ser alterado através da função 44h do DOS (IOCTL).

Modo Binário

Neste modo o DOS não efetua tratamento sobre os caracteres sendo lidos/escritos:

- leitura

  Lê exatamente o número de bytes especificado, a menos que ocorra fim de arquivo. O caracter 1Ah (Control Z) não é tratado como marca de fim de arquivo. Na entrada padrão não testa os caracteres de controle Control S, Control P e Control C, nem fornece facilidades de edição.

- escrita

  Escreve exatamente o número de bytes especificado, sem qualquer alteração. Durante a escrita na saída padrão não testa a digitação dos caracteres Control S, Control P ou Control C.

Modo ASCII

Neste modo o DOS efetua vários tratamentos sobre os caracteres que estiverem sendo lidos/escritos:

- leitura

    1. Se existirem caracteres no buffer do dispositivo, obtém dele os caracteres para leitura. O número de caracteres lidos é o mínimo entre o pedido e o disponível no buffer; se sobrarem caracteres no buffer eles serão usados na próxima leitura.
    2. Se o buffer estiver vazio, lê uma linha, terminada por CR, para o buffer. Na leitura da entrada padrão permite a edição da linha que está sendo digitada.
    3. Retorna ao programa os caracteres lidos, seguidos de CR LF se o número de caracteres especificado é suficiente para incluí-los. Desta forma, os dados lidos se assemelham a um registro de arquivo texto.
    4. O caracter 1Ah (Control Z) é tratado como marca de fim de arqui-vo; e retornado sem CR LF no final.
    5. Testa na entrada padrão os caracteres de controle Control S, Control P e Control C.
    6. Na leitura da entrada padrão efetua eco na saída padrão; o caracter 09h (tab) é ecoado expandido em espaços.

- escrita

    1. Encerra a escrita ao encontrar o caracter 1Ah (Control Z).
    2. Expande o caracter 09h (TAB) em espaços. Demais caracteres de controle são convertidos em sequências '^' letra (por exemplo, 05h é escrito como ^E).
    3. Durante a escrita na saída padrão, testa a digitação dos carac-teres Control S, Control P ou Control C.

Os caracteres Control S, Control P e Control C controlam, respectivamente, a suspensão da escrita, auto-impressão e interrupção de programa (ver item 7.3)

A expansão de caracteres TAB é feita supondo tabulação fixa de oito em oito colunas.

5 INTERRUPÇÕES DE SOFTWARE DO DOS

A principal interface entre o DOS e os programas são as interrupções de software. Os vetores de interrupção 20H a 3FH estão reservados ao DOS, estando em uso atualmente:

20H - Terminação de programas

21H - Chamada de função do DOS

22H - Endereço de terminação de programa

23H - Interrupção de programa ("Ctrl Break Exit")

24H - Tratamento de erro crítico

25H - Leitura absoluta em disco

26H - Escrita absoluta em disco

27H - Termina programa, mantendo-o residente

28H - Aguardando caracter (*)

29H - Escrita rápida em dispositivo (*)

2AH - ??? (somente DOS 3)

2BH - ??? (somente DOS 3)

2EH - Chamada ao COMMAND (*)

2FH - Multiplexação (somente DOS 3)

30H - ??? (somente DOS 3)

31H - ??? (somente DOS 3)

Obs.: Os vetores indicados com (*) não estão documentados nos manuais do DOS; os marcados com ??? têm uso não esclarecido.

Os vetores 20H, 21H, 22H e 27H serão discutidos no Capítulo 7; o vetor 29H já foi visto no Capítulo 3.

5.1 INTERRUPÇÃO DE PROGRAMA

O DOS permite a interrupção da execução de um programa através da digitação de Ctrl Break ou Ctrl C no console. Os pedidos de interrupção são tratados quando da chamada a determinadas funções do DOS; não é possível interromper (sem recarga do DOS) programas que não estejam chamando o DOS (por exemplo, um programa em "loop" infinito).

O DOS trata sempre pedidos de interrupção de programa em determinadas funções de caracter, como será detalhado no Capítulo 7. Através de uma função do DOS ou do comando BREAK pode-se requisitar ao DOS o tratamento também nas demais funções.

Vejamos inicialmente como o DOS detecta um pedido de interrupção de programa. Existem duas maneiras:

1. Digitação de Ctrl Break

   O BIOS executa, no momento da interrupção do teclado, uma chamada à rotina apontada pelo vetor 1BH, que está nos tratadores padrão. Esta rotina simplesmente aciona um indicador. Quando o DOS chamar o tratador de console para verificar se existe caracter disponível, o indicador acionado causará a informação de que foi digitado Ctrl C. Desta forma, o Ctrl Break "passa na frente" de outros códigos eventualmente na fila do teclado.

2. Digitação de Ctrl C

   Neste caso, o BIOS coloca o código normalmente na fila de teclado. Quando o DOS pergunta ao tratador padrão de console se existe caractere disponível ele consulta o BIOS que só sabe informar o primeiro da fila. Portanto, o Ctrl C pode ficar "preso" atrás de outros códigos pendentes na fila do teclado.

Independente da forma de detecção do pedido, o seu tratamento é sempre o mesmo:

1. Se estava sendo executada a função 9 (escrita de cadeia) ou 10 (leitura do console com edição) do DOS, é escrito ^C carriage- return line-feed no console.
3. Os registradores são restaurados com o conteúdo que tinham quando da chamada do DOS, e é executado um INT 23H.
4. Se o controle retornar do INT 23H, a chamada ao DOS é reexecutada.

Não há restrições no que a rotina de tratamento de INT 23H pode fazer (inclusive chamar ao DOS), desde que os registradores sejam mantidos inalterados no caso de retorno ao DOS via IRET.

Na versão 1 o tratamento de INT 23H é normalmente realizado pelo COMMAND; nas versões 2 e 3 o tratamento é feito pelo próprio DOS. Em todos os casos a ação consiste em terminar o programa que estava sendo executado.

5.2 TRATAMENTO DE ERRO CRÍTICO

Quando ocorre um erro em acesso a dispositivo, o DOS executa um INT 24H com os registradores contendo os seguintes valores:

SS:SP apontam para a pilha do usuário

BP:SI apontam para cabeçalho do tratador do dispositivo [(2)]

DI contém no byte menos significativo o código do erro devolvido pelo tratador:

- 00h - tentativa de escrita em disco protegido
- 01h - unidade inválida
- 02h - unidade sem disco
- 03h - comando inválido
- 04h - erro nos dados ("CRC error")
- 05h - tamanho incorreto da requisição
- 06h - erro de posicionamento ("seek")
- 07h - tipo de disco desconhecido

08h - setor não encontrado

09h - impressora sem papel [2]

0Ah - falha em escrita [2]

0Bh - falha em leitura [2]

0Ch - outros erros

AH bit 7 = 0 - erro de leitura ou escrita em dispositivo de bloco

AL unidade lógica (0 = A:)

AH bit 0 0 leitura

1 escrita

bits 2 e 1 00 erro na área do DOS

01 erro na FAT

10 erro em diretório

11 erro na área de dados

bit 3 1 se "falha" aceito [3]

bit 4 1 se "retenta" aceito [3]

bit 5 1 se "ignora" aceito [3]

bit 7 = 1 - outros erros

se dispositivo de bloco

FAT incorreta

AL = unidade lógica (0 = A:)

se dispositivo de caracter [2]

erro em leitura ou escrita

[2] somente a partir da versão 2

[3] somente a partir da versão 3

A pilha contém os seguintes valores (do mais recente ao mais antigo):

IP referente ao INT 24h executado pelo DOS

CS idem

Flags idem

AX registradores quando da chamada ao DOS por INT 21h

BX idem

CX idem

DX idem

SI idem

DI idem

BP idem

DS idem

ES idem

IP idem

CS idem

Flags idem

Normalmente, a rotina de tratamento de erro crítico retorna ao DOS (via IRET) com os registradores SS, SP, DS, ES, BX, CX e DX inalterados e com AL contendo:

0 Se o erro deve ser ignorado

O DOS prossegue, desconsiderando o erro. Esta operação é "peri-gosa", visto o conteúdo da memória (na leitura) ou do disco (na escrita) serem indeterminados.

1 Se a operação deve ser retentada

O DOS tenta executar a operação novamente

2 Se o programa deve ser abortado

O DOS aborta o programa, forçando a sua terminação anormal

3 Se a função do DOS deve retornar com indicação de falha

Esta opção só está disponível na versão 3 e permite que o DOS prossiga, terminando a função que estava sendo executada com indicação de falha. Isto simplifica a continuação do programa em caso de erro não recuperável.

Excepcionalmente, o programa pode não retornar ao DOS via IRET, por exemplo para não abortar o programa em caso de erro não recuperável. Neste caso deve-se ter o cuidado de "limpar" a pilha e chamar uma função de disco (sugere-se a "reinicia disco") para colocar o DOS novamente em estado conhecido.

O DOS só chama INT 24h, em caso de erro de disco, após ter realizado cinco tentativas.

5.3 LEITURA E ESCRITA ABSOLUTA EM DISCO

Estas funções permitem o acesso a um dispositivo de bloco em termos de setores lógicos, não gerando INT 24h em caso de erro.

Nestas funções temos a principal incompatibilidade entre a versão 1 do DOS e as posteriores. Na versão 1 os setores eram numerados a partir do primeiro setor da primeira trilha da primeira face e avançando na mesma face até o último setor da última trilha; a a partir daí, a numeração prosseguia no primeiro setor da segunda face da primeira trilha. Na versão 2 e seguintes a numeração segue o padrão já visto no estudo de tratadores: a numeração prossegue na mesma trilha até o último setor da última face. Por exemplo, para um disco de 5 1/4", oito setores/trilha e duas faces temos:

	DOS 1			DOS 2 e 3		
setor lógico	**F**	**T**	**S**	**F**	**T**	**S**
0	0	0	1	0	0	1
1	0	0	2	0	0	2
...						
7	0	0	8	0	0	8
8	0	1	1	1	0	1
...						
319	0	39	8	1	19	8
320	1	0	1	0	20	1
...						
639	1	39	8	1	39	8

Na chamada a estas funções devemos ter:

AL	unidade lógica (0 = A:)
CX	número de setores a ler ou escrever
DX	setor lógico inicial
DS:BX	endereço de transferência

Não é permitida a transferência de mais de 64 K bytes (128 setores de 512 bytes).

Na saída, CF = 0 indica que a transferência ocorreu corretamente; CF = 1 indica erro, com AL contendo o código de erro devolvido pelo tratador (idêntico ao devolvido em DI no caso de erro crítico) e AH contendo um código de erro semelhante ao devolvido pelo BIOS:

80h	unidade sem disquete ("time-out")
40h	erro de posicionamento ("seek")
04h	setor não encontrado
03h	tentativa de escrita em disco protegido
02h	outros erros

No caso do tratador padrão do DOS, temos uma série curiosa de conversões: o código do BIOS é convertido no código padrão para tratador, que posteriormente é reconvertido no de BIOS...

O DOS efetua até cinco tentativas antes de indicar um erro.

Existe uma "pegadinha" nesta função: o conteúdo original dos flags é mantido na pilha! Aparentemente, o programador que fez este ponto de entrada do DOS não sabia como retornar de um INT com um flag (o CF) alterado e a pilha limpa (bastava usar RET 2). Deve-se, portanto, lembrar de retirar os flags da pilha (um POP BX ou ADD SP,2 é suficiente). De quebra, todos os registradores, exceto DS, ES, SS e SP, podem ser alterados por esta função.

Existe ainda um outro cuidado a tomar, que é garantir que o tratador saiba qual o tipo de disco presente na unidade. Estas funções não tomam nenhuma providência a respeito, fazendo com que o tratador acredite que o disco não foi trocado desde o último acesso. Deve-se, portanto, realizar antes um acesso através das funções normais, que testam troca de disco (busca no diretório, informação do diretório atual etc.).

As excentricidades e as incompatibilidades entre a versão 1 e as demais fizeram com que estas funções fossem preteridas por chamadas diretas ao BIOS. Isto é uma pena, pois elas têm as seguintes vantagens:

- Permitem acesso a unidades controladas por tratadores instalados;
- Permitem acesso a setor sem o problema das "fronteiras de DMA" (o BIOS não permite que os quatro bits mais significativos de endereço mudem numa transferência de disco);
- Permitem acesso multissetor com mudança automática de face e trilha; e
- Trabalham somente na partição DOS de discos fixos, dispensando a determinação do seu setor físico inicial.

5.4 AGUARDANDO CARACTER

Quando o DOS está aguardando ou escrevendo um caracter nas funções de caracter, ele executa INT 28h de forma a dar uma oportunidade de outro programa chamar o DOS. Este recurso (não documentado) é utilizado pelo PRINT e está explicado em mais detalhes no Capítulo 10.

5.5 CHAMADA AO COMMAND

É possível a um programa chamar o COMMAND para que ele execute um comando.

Para isto deve-se carregar DS:SI com o endereço de uma área contendo no primeiro byte o número de caracteres no comando, nos bytes seguintes o comando, e no final um caracter carriage return (que não é contado no número de caracteres) e executar INT 2Eh. No retorno, o conteúdo de todos os registradores deve ser considerado indefinido.

O mesmo efeito pode ser obtido (com muito mais trabalho) disparando a execução de uma segunda "encarnação" do COMMAND, como veremos no Capítulo 7.

Este recurso (não documentado) é útil quando se deseja alterar o tamanho do "environment mestre" (que será discutido no Capítulo 6). Esta área pertence ao COMMAND e só pode ser alterada por ele próprio - daí o uso desta função.

5.6 MULTIPLEXAÇÃO

O vetor 2Fh foi introduzido na versão 3 do DOS e é utilizado para estabelecer uma interface padrão com programas residentes. As chamadas a INT 2Fh têm a seguinte convenção de parâmetros:

<E> AH número do programa

01h PRINT

02h ASSIGN

10h SHARE

demais números de 0 a 7Fh estão reservados ao DOS

AL função

0 informa instalação

1 inclui arquivo

2 cancela arquivo

		4	informa estado
		5	termina informação de estado
			demais códigos variam conforme o programa
<S>	CF	0	função bem sucedida
		1	ocorreu erro, AX contém o código do erro
	AX		código de erro (se CF = 1)
		1	função inválida
		2	arquivo não achado
		3	subdiretório inexistente
		4	atingiu limite de arquivos abertos
		5	acesso ao arquivo negado
		8	fila de impressão cheia
		9	ocupado
		12	nome muito grande
		15	unidade de disco inválida.
			outros registradores podem ser usados conforme o programa

Um programa que assume INT 2Fh deve testar se AH contém o seu número; se não, deve passar a chamada à rotina que ocupava o vetor anteriormente. Deve ser levado em conta que durante o tratamento de uma interrupção 2Fh outra pode vir a ocorrer; o início do tratamento (teste do número do programa) deve ser reentrante.

Veremos agora alguns detalhes sobre os utilitários do DOS que utilizam INT 2Fh.

PRINT

O utilitário PRINT permite a impressão de arquivos simultaneamente à execução de outros programas. Introduzido na versão 2 do DOS, ele recebeu grandes aperfeiçoamentos na versão 3, dentre os quais o acesso através de INT 2Fh.

O uso do PRINT é iniciado pela sua instalação na memória. A partir daí ele mantém uma fila de arquivos a imprimir e utiliza as interrupções 1Ch (tempo real) e 28h (aguardando caracter) para assumir temporariamente o controle do micro e ler dados do disco e enviá-los à impressora.

O acesso a funções do PRINT via INT 2Fh é feito colocando-se 0 em AH e em AL o código da função desejada:

0		Informa instalação
		retorna em AL:
	0	se não instalado e pode vir a ser
	1	se não instalado e não pode vir a ser

(usado por programas que assumem INT 2Fh e não desejam que PRINT seja instalado)

FFh instalado

1 Coloca arquivo na fila de impressão

DS:DX aponta para área contendo:

nível (1 byte) - atualmente 0

ponteiro para nome (dword)

O nome é especificado através de uma cadeia ASCIIZ não ambígua.

2 Retira arquivo na fila de impressão

DS:DX aponta para uma cadeia ASCIIZ (pode ser ambígua) , que determina os arquivos a serem retirados.

3 Limpa fila de impressão

retira todos os arquivos da fila de impressão

4 Informa estado da fila de impressão

Esta função bloqueia a fila de impressão e retorna em DX o número de erros ocorridos e DS:SI aponta para a fila de impressão. A fila de impressão é uma tabela onde cada entrada tem 64 bytes e contém um nome de arquivo (cadeia ASCIIZ). O primeiro nome é o do arquivo sendo impresso e o fim da tabela é marcado por um nome vazio (isto é, tem 0 no primeiro caracter). A fila será liberada na próxima chamada a INT 2Fh

5 Libera fila de impressão após informação de estado

Esta função na realidade não faz nada; a sua chamada simplesmente libera a fila de impressão, o que também é feito nas demais (exceto na informa estado da fila).

ASSIGN

Este utilitário redireciona os acessos feitos a uma unidade de disco para outra. Apenas uma função está documentada:

0 Informa instalação

AL 0 se não instalado e pode vir a ser

1 se não instalado e não pode vir a ser

FFh instalado

SHARE

Este utilitário controla o compartilhamento de arquivos, sendo normalmente utilizado em ambiente de rede. Apenas uma função está documentada:

0 Informa instalação

AL 0 se não instalado e pode vir a ser

1 se não instalado e não pode vir a ser

FFh instalado

6 PROGRAMAS

Neste capítulo serão discutidos conceitos relacionados a programas no DOS. O sistema DOS suporta dois formatos de armazenamento de programas em disco, o ".COM" e o ".EXE", e fornece um conjunto de funções relacionadas a eles (como carga e termino).

Na versão 1 do DOS o conceito de programa estava firmemente ligado ao COMMAND, que era o responsável pela carga e terminação de programas. Na versão 2 a carga de programas continuou sendo feita pelo COMMAND, porém de forma integrada ao núcleo do DOS; a terminação passou para o núcleo do DOS e novas funções foram introduzidas. Na versão 3 o núcleo do DOS passou a conter todas as funções relacionadas a programas, exceto a manipulação do "environment" (que será discutido em breve).

Desde a primeira versão do DOS, uma estrutura de dados esteve relacionada a programas: a PSP - Prefixo de Segmento de Programa ("*Program Segment Prefix*"). Esta estrutura é montada pelo DOS sempre que um programa é carregado. No DOS 1 a finalidade principal desta estrutura era simular os primeiros 256 bytes de um sistema CP/M-80. A partir da versão 2, a PSP passou a armazenar uma série de informações do DOS (por exemplo, os handles em uso pelo programa). O endereço inicial da PSP (que é sempre da forma XXXXh:0) passou a ser tratado pelo DOS como uma identificação do programa. Embora as versões atuais do DOS não suportem a presença de vários programas em execução paralela (multiprogramação), elas suportam a presença de vários programas na memória, identificando cada um deles pelo segmento da sua PSP. Daí, às vezes, ser este valor denominado de "identificação do processo" ("*process id*").

6.1 "ENVIRONMENT"

O "*environment*" (ambiente) é uma área de memória, com endereço na forma XXXXh:0 e existente a partir da versão 2 do DOS, na qual são armazenadas cadeias ASCIIZ com o seguinte formato:

nome=valor

O final das cadeias é indicado por uma cadeia vazia, isto é, um byte contendo zero. No DOS 3, após as cadeias temos um word (atualmente contendo 1) e uma cadeia ASCIIZ que contém o nome completo do arquivo do qual o programa foi carregado (pena que esta informação esteja inacessível na versão 2).

A finalidade do environment é conter uma série de parâmetros de interesse de vários programas, de forma a eliminar a necessidade de fornecê-los quando da execução.

O environment é inicialmente criado pelo COMMAND, contendo sempre uma cadeia do tipo 'COMSPEC=nome de arquivo'; os utilitários PATH e PROMPT colocam cadeias do tipo 'PATH=diretório; diretório;...' e 'PROMPT=sequência'. Novas cadeias são normalmente acrescentadas através do utilitário SET.

Quando um programa é carregado, "herda" uma cópia do environment do seu "pai". As alterações realizadas pelo programa no seu environment afetarão os seus "filhos", porém não o seu "pai". O endereço do environment de um programa está armazenado na sua PSP.

Por este motivo o environment do COMMAND é denominado "environment mestre", sendo direta ou indiretamente transmitido a todos os programas. O tamanho do environment mestre é inicialmente de 160 bytes. O COMMAND pode aumentar o environment se a memória seguinte a ele estiver livre, ou seja, se

não existirem programas residentes na memória. O tamanho inicial pode ser alterado através do parâmetro /Ennn do COMAND. Nas versões anteriores à 3.2 este parâmetro não estava documentado e nnn era o tamanho em parágrafos (blocos de 16 bytes). Na versão 3.2 o parâmetro foi oficializado, porém nnn passou a ser o tamanho em bytes. O mínimo permitido é 160 bytes e o máximo 32K; o uso do parâmetro é feito acrescentando a seguinte linha no arquivo CONFIG.SYS:

SHELL=COMMAND.COM/Ennn

Não existe nenhuma maneira documentada de um programa acessar o environment mestre. Através do INT 2Eh é possível pedir ao COMMAND a execução de um comando SET; como isto não é documentado, esta facilidade pode vir a ser retirada em versões futuras.

A rotina abaixo localiza uma cadeia no environment:

```
; LOC_ENV - localiza cadeia em environment
; <E> ES segmento do environment
; BX nome a procurar (cadeia ASCIIZ)
; <S> DX inicio da cadeia (0FFFFh se nao achou)
;

         SUB     DI,DI
         CLD
 L1:
         MOV     DX,DI
         MOV     SI,BX                ;inicio da cadeia
         CMP     BYTE PTR ES:[DI],0   ;testa fim do environment
         JE      L4
L2:
         CMP     BYTE PTR [SI],0      ;fim do nome ?
         JE      L5                   ;sim: achou
         CMPSB                        ;nao: compara
         JE      L2                   ; repete enquanto igual
L3:
         INC     DI                   ;achou diferenca
         CMP     BYTE PTR ES:[DI-1],0 ;avanca p/ proxima cadeia
         JE      L1
         JMP     SHORT L3
L4:
         MOV     DX,0FFFFH
L5:
         RET
```

6.2 PREFIXO DE SEGMENTO DE PROGRAMA (PSP)

Conforme já dito, uma PSP é construída para cada programa carregado e contém diversos parâmetros de interesse do DOS e do programa:

Deslocamento	tamanho	conteúdo
00h	2 bytes	instrução INT 20h um desvio para PSP:0 causa a terminação do programa, ver Capítulo 7
02h	1 word	tamanho da memória, em parágrafos
04h	1 byte	reservado
05h	5 bytes	chamada ao DOS para facilitar a adaptação de programas escritos para o CP/M-80, o DOS permite o acesso ao seu grupo principal de funções através de um "call near" para PSP:5. Estas posições possuem um "call far" para o DOS, com endereço montado de forma a que o offset (word em PSP:6) indique o número de bytes livres no segmento da PSP (normalmente 0FFFxh).
0Ah	1 dword	endereço de terminação o conteúdo destas posições é copiado para o vetor de interrupção 22h ao final da execução do programa (ver Capítulo 7).
0Eh	1 dword	endereço de interrupção o conteúdo destas posições é copiado para o vetor de interrupção 23h ao final da execução do programa (ver Capítulo 7).
12h	1 dword	endereço da rotina de erro crítico o conteúdo destas posições é copiado para o vetor de interrupção 24h ao final da execução do programa (ver Capítulo 7).
16h	1 word	endereço da PSP do "processo pai"
18h	20 bytes	tabela de handles do programa [2] * cada byte nesta tabela contém 0FFh, se o handle associado estiver fechado, ou o índice do arquivo na tabela de FCBs interna ao DOS, se o handle estiver aberto.
2Ch	1 word	segmento do environment [2]
2Eh	1 dword	endereço da pilha do usuário [2] * sempre que o DOS é chamado, o conteúdo de SS e SP são salvos nestas posições.
32h	1 word	tamanho da tabela de handles [3] * esta posição contém normalmente 20 (14h), porém pode (teoricamente) ser alterada pelo usuário.
34h	1 dword	endereço da tabela de handles [3] * esta posição normalmente contém o endereço PSP:18h, porém pode (teoricamente) ser alterada pelo usuário.
38h	24 bytes	reservado
50h	3 bytes	chamada ao DOS [2] estas posições contém as instruções "INT 21h" e "RET far", permitindo chamar o DOS através de um "call far" para PSP:50h.
53	9 bytes	reservado
5Ch	16 bytes	FCB do primeiro parâmetro quando da carga do programa, estas posições são preenchidas com a unidade, o nome e a extensão do nome do primeiro parâmetro fornecido.
6Ch	20 bytes	FCB do segundo parâmetro quando da carga do programa, estas posições são preenchidas com a unidade, o nome e a extensão do nome do segundo parâmetro for-necido.
80h	128 bytes	linha de comando

	o primeiro byte contém o número de caracteres na linha, e os demais os caracteres seguintes ao nome do programa na linha que causou a sua execução. Na versão 2 e posteriores os caracteres relacionados a redirecionamento ("< nome de arquivo", "> nome de arquivo" e "| comando") não são incluídos nesta área.

[2] somente a partir da versão 2

[3] somente a partir da versão 3

* campos não documentados

6.3 PROGRAMAS .COM

Um programa .COM é armazenado em disco na forma de imagem de memória, isto é, o arquivo contém exatamente os bytes do programa. O seu tamanho não pode ser superior a 64 Kbytes - 256 bytes (ocupados pela PSP).

A carga de um programa .COM é extremamente simples:

1. O DOS monta uma PSP para o programa;
2. O programa é carregado logo após a PSP, ou seja, no endereço PSP:100H, através da sua leitura para a memória;
3. Os registradores de segmento (CS, DS, ES e SS) são carregados com o segmento da PSP; o registrador SP fica com 0FFF0h ou aponta para o fim da memória, se a memória livre for inferior a 64 Kbytes;
4. Os registradores AL e AH são carregados com 0FFh ou 000h, conforme os dois primeiros parâmetros sejam, respectivamente, nomes válidos ou inválidos para arquivos;
5. É colocado na pilha um word com zeros, de forma a que um "RET near" passe o controle para o INT 20H na PSP, causando o término do programa; e
6. O programa é iniciado pela instrução em 100H

Um programa .COM deve independer do segmento em que foi carregado, visto não sofrer nenhuma relocação quando da carga. Deve também ter sua primeira instrução no primeiro byte do arquivo, que será carregado no offset 100H.

Normalmente, um programa assembler .COM é gerado através de três passos:

1. O programa fonte é convertido em objeto relocável pelo Assembler
2. O objeto relocável é convertido em objeto executável .EXE pelo LINK
3. O objeto .EXE é convertido em .COM pelo programa EXE2BIN

O programa EXE2BIN faz algumas exigências quanto ao programa:

1. Não pode ter segmento STACK definido
2. Não pode conter referências intersegmento
3. Tem que ter endereço inicial definido no offset 100h

Tipicamente um programa assembler .COM tem a seguinte estrutura:

```
; DEFINICAO DE CONSTANTES E SEGMENTOS EM ENDERECOS FIXOS, SEM
; DECLARACAO DE DADOS COM CONTEUDO DEFINIDO

UNICO    SEGMENT
         ASSUME CS:UNICO, DS:UNICO, ES:UNICO, SS:UNICO
         ORG 100H

INICIO:
         JMP COMECO

; DECLARACAO DOS DADOS, PARA QUE O ASSEMBLER SAIBA QUAIS OS
; SEUS TIPOS - ISTO NAO DEVERIA SER NECESSARIO, POREM AS
; PRIMEIRAS VERSOES DO MASM TINHAM DIFICULDADES EM TRATAR
; REFERENCIAS A VARIAVEIS DEFINIDAS APOS O SEU USO

COMECO:
; PROGRAMA PROPRIAMENTE DITO; SUBROTINAS TEM QUE SER "NEAR"

         ÚNICO   ENDS
         END     INICIO
```

Algumas variantes são possíveis. Uma bastante comum é ter um segmento para código e um para dados, localizado após o de código. Neste caso a relocação em relação ao segmento de carga consiste na soma do offset final do código ao conteúdo de DS (que é iniciado pelo DOS apontando para o segmento de código).

A estrutura .COM é simples e propicia uma carga rápida, sendo apropriada para pequenos programas.

6.4 PROGRAMAS .EXE

Um programa .EXE é constituído de duas partes:

- Um cabeçalho, que contém informações sobre o endereço de início de execução, relocações intersegmento necessárias etc.; e
- O módulo do programa, que é uma imagem do programa, supondo que ele vá ser carregado no início da memória (segmento 0).

O cabeçalho, por sua vez, divide-se em duas partes: a descrição e a tabela de relocação. O cabeçalho contém sempre um número inteiro de páginas de 512 bytes. O formato da descrição é o seguinte:

Deslocamento	tamanho	conteúdo
0	1 word	04D5Ah - identifica programa .EXE
2	1word	número de bytes ocupados na última página de 512 bytes do arquivo
4	1 word	número de páginas de 512 bytes que compõem o arquivo
6	1 word	número de ítens na tabela de relocação
8	1 word	tamanho do cabeçalho em parágrafos (blocos de 16 bytes)
0Ah	1 word	número mínimo de parágrafos de memória livre após o programa
0Ch	1 word	número máximo de parágrafos de memória livre após o programa. Normalmente 0FFFFh, indicando que o programa deve ser carregado no menor endereço possível
0Eh	1 word	valor inicial para SS

10h	1 word	valor inicial para SP
12h	1 word	soma de todos os words do arquivo, com "vai-um" ignorado
14h	1 word	valor inicial para IP (início da execução)
16h	1 word	valor inicial para CS (início da execução)
18h	1 word	deslocamento do primeiro item da tabela de relocação em relação ao início do arquivo, normalmente 1Ch
1Ah	1 word	número do "overlay". Normalmente 0, indicando programa principal

Cada item da tabela de relocação é um dword, apontando para um word do programa que deve ser relocado em relação ao endereço de carga.

Para entendermos melhor, vejamos um programa assembler e o seu respectivo objeto .EXE:

```
DADOS    SEGMENT 'DATA'            ;SEGMENTO DE DADOS
VAR      DW      55H
DADOS    ENDS

STACK    SEGMENT STACK 'STACK'     ;SEGMENTO DA PILHA
         DW      64 DUP (0)
STACK    ENDS

COD1     SEGMENT 'CODE'            ;PRIMEIRO SEGMENTO DE CODIGO
         ASSUME  CS:COD1
ROTINA   PROC    FAR
         RET
ROTINA   ENDP
COD1     ENDS

COD2     SEGMENT 'CODE'            ;SEGUNDO SEGMENTO DE CODIGO
         ASSUME  CS:COD2
INICIO:
         MOV     AX,DADOS
         MOV     DS,AX
         CALL    ROTINA
COD2     ENDS

         END     INICIO
```

Processando o objeto através do LINK, obtemos o seguinte posicionamento dos segmentos, supondo carga no segmento 0:

Endereço	segmento	tamanho
00000h	DADOS	02h bytes
00010h	STACK	80h bytes
00090h	COD1	01h bytes
000A0h	COD2	0Ah bytes

O cabeçalho fica, portanto, com os seguintes valores:

```
4D5Ah 00AAh 0002h 0002h 0020h 0000h FFFFh

0001h 0080h 7968h 0000h 000Ah 001Ch 0000h

0001h 000Ah 0008h 000Ah 0000h 0000h ...
```

O módulo do programa fica com:

55h, 15 bytes com 00h (DADOS)

128 bytes com 00h (STACK)

CBh, 15 bytes com 00h (COD1)

B8h, 00h, 00h, 8Eh, D8h, 9Ah, 00h, 00h, 09h, 00h (COD2)

A carga de um programa .EXE processa-se da seguinte forma:

1. O DOS monta uma PSP para o programa;
2. A descrição do programa é lida do arquivo;
3. Se o número máximo de parágrafos livres após o programa for 0FFFFh, o módulo do programa é carregado no menor endereço possível, normalmente no segmento seguinte ao da PSP. Caso contrário, o programa é carregado no final da memória;
4. O programa carregado é relocado, em função da tabela de relocação. Notar que os itens da tabela de relocação também têm que ter os segmentos relocados;
5. Os registradores de segmento DS e ES são carregados com o segmento da PSP;
6. Os registradores SS e SP são carregados com os valores contidos na descrição, devidamente relocados em função do endereço de carga;
7. Os registradores AL e AH são carregados com 0FFh ou 000h, conforme os dois primeiros parâmetros sejam, respectivamente, nomes válidos ou inválidos para arquivos; e
8. A execução é iniciada no endereço especificado para CS e IP na descrição, devidamente relocado em função do endereço de carga

A estrutura .EXE é mais complexa, fornecendo mais informações sobre o programa. A carga deste tipo de programa é mais demorada, porém o programa pode ter mais de 64 Kbytes. Os programas .EXE são normalmente gerados por compiladores; os programas tipicamente codificados em assembler não necessitam dos recursos adicionais.

7 FUNÇÕES DO DOS

O DOS possui um grande número de funções que podem ser chamadas por programas. A maioria delas está agrupada em um único ponto de chamada, o Grupo Principal. Existem ainda algumas funções acessíveis por outros pontos, já apresentadas no Capítulo 5.

Normalmente os parâmetros de entrada e saída são passados através de registradores.

7.1 FORMAS DE CHAMAR O DOS

O grupo principal de funções do DOS pode ser chamado de três formas:

1) INT 21h

   Esta é a forma mais comum de se chamar o DOS. O número da função desejada é carregado em AH, e uma instrução INT 21h é executada.

2) CALL NEAR para PSP:5

   Esta maneira existe apenas para facilitar a conversão de programas escritos para o CP/M-80. A PSP possui no offset 5 uma instrução CALL FAR que transfere o controle a uma rotina do DOS que altera a pilha, de forma a simular um INT 21h, e passa o controle à rotina normal de entrada do DOS. O número da função desejada deve ser colocado em CL. Este método só permite acesso às funções de número 0 a 24h e altera sempre o conteúdo de AX.

3) CALL FAR para PSP:50h

   Esta maneira existe somente a partir da versão 2 do DOS e tem pouca utilidade prática. O número da função desejada deve ser colocado em AH.

As funções são apresentadas no restante deste capítulo, organizadas por uso (funções de arquivo, de gerenciamento de memória etc.). Uma outra classificação importante é pelos seus números:

00h a 2Eh	Funções "tradicionais", presentes em todas as versões do DOS. Estas funções preservam todos os registradores exceto os usados para retorno de resultados;
2Fh a 58h	Funções introduzidas na versão 2. O registrador AX é sempre alterado, contendo código de erro quando CY = 1; e
59h a 62h	Funções introduzidas na versão 3. O registrador AX é sempre alterado, contendo código de erro quando CY = 1.

Dentro destes grupos de funções existem várias que são documentadas pela MicroSoft como "reservadas" ou "de uso interno". Algumas delas serão discutidas neste capítulo; porém o seu uso deve ser evitado, visto não existir compromisso da MicroSoft quanto à sua manutenção em versões futuras.

Com exceção de AX e dos registradores para retorno de resultado, os registradores são mantidos inalterados. Para tanto, o DOS coloca na sua entrada todos os registradores na pilha do usuário. Em seguida, ele altera SS e SP para uma de suas três pilhas:

- Uma usada para as funções de número 1 a 0Ch, 50h, 51h e 62h, chamadas normalmente;
- Outra usada para as funções de número 1 a 0Ch, 50h, 51h e 62h, chama-das durante o tratamento de um erro crítico;
- E, finalmente, uma usada para as demais funções.

Este esquema (aparentemente estranho) de três pilhas permite que as funções de 1 a 0Ch, 50h, 51h e 62h sejam chamadas durante o tratamento de um erro crítico que tenha sido causado por uma outra chamada ao DOS. Permite também que durante uma espera por dispositivo nas chamadas às funções de 1 a 0Ch as demais funções sejam chamadas (o que pode causar um erro crítico e uma terceira entrada no DOS).

Com exceção destes casos o DOS só suporta uma chamada em curso de cada vez (ou, dito de outra forma, ele não é reentrante).

A chamada ao DOS durante uma interrupção de hardware não é também aconselhável, visto não existirem muitas precauções no DOS e no BIOS para evitar a interrupção de trechos críticos (que não poderiam ser interrompidos).

No Capítulo 10 a questão de reentrada e chamada ao DOS durante uma interrupção é discutida mais a fundo.

7.2 CÓDIGOS DE ERRO

Conforme já dito, as funções introduzidas a partir da versão 2 indicam erro retornando com CY=1 e com AX contendo o código do erro.

Caso não ocorra erro, CY retorna com 0. Na versão 2, AX retorna com zero (se não contiver resultado); na versão 3 o conteúdo de AX é indefinido. Deve-se, portanto, testar o flag CY antes de testar o conteúdo de AX.

Na versão 3 foi criada uma nova função, "Fornece código de erro expandido", que fornece informações adicionais sobre o erro ocorrido.

Os códigos de erro podem ser divididos em três grupos:

01 a 18	Códigos de erro retornados na versão 2
19 a 31	Códigos de erro referentes a erro crítico
32 a 88	Códigos de erro adicionais na versão 3

Para garantir a compatibilidade com a versão 2, a versão 3 retorna somente erros do primeiro grupo para as funções introduzidas a partir do DOS 2.00. Os demais códigos são obtidos nas funções introduzidas com o DOS 3.00 ou através da "Fornece Código de Erro Expandido".

Os códigos de erros definidos são:

1 - número inválido de função

2 - arquivo não encontrado

3 - caminho não encontrado

4 - não há handle disponível

5 - acesso negado

6 - handle inválido

7 - blocos de controle de memória danificados

8 - memória insuficiente

9 - endereço inválido de bloco de memória

10 - "environment" inválido

11 - formato inválido

12 - código de acesso inválido

13 - dados inválidos

15 - unidade inválida

16 - tentativa de suprimir diretório atual

17 - nomes devem especificar mesma unidade

18 - não há mais arquivos (busca concluída)

19 - tentativa de escrita em disco protegido

20 - unidade desconhecida

21 - unidade sem disco

22 - comando desconhecido

23 - erro de leitura/gravação (CRC)

24 - tamanho inválido da requisição

25 - erro de posicionamento

26 - tipo de disco desconhecido

27 - setor não encontrado

28 - impressora sem papel

29 - falha em escrita

30 - falha em leitura

31 - falha genérica

32 - violação de compartilhamento

33 - violação de bloqueio ("lock")

34 - troca inválida de disco

35 - não há FCB disponível

36 - "buffer" de compartilhamento cheio

50 - requisição de rede não suportada

51 - computador remoto não responde

52 - nome já existente na rede

53 - nome não encontrado na rede

54 - rede ocupada

55 - dispositivo não mais existente na rede

56 - excedido limite de comando do "NetBIOS"

57 - falha no hardware de rede

58 - resposta incorreta da rede

59 - erro inesperado da rede

60 - adaptador remoto incompatível

61 - fila de impressão cheia

62 - não há espaço para arquivo de impressão

63 - arquivo de impressão foi suprimido

64 - nome foi suprimido da rede

65 - acesso negado

66 - tipo de dispositivo na rede incorreto

67 - nome não encontrado na rede

68 - excedido limite de nome na rede

69 - excedido limite de sessão do "NetBIOS"

70 - suspenso temporariamente

71 - requisição de rede não aceita

72 - redirecionamento suspenso

80 - arquivo já existe

82 - não foi possível criar entrada no diretório

83 - falha devido a erro crítico

84 - excesso de redireções

85 - redirecionamento duplicado

86 - senha inválida

87 - parâmetro inválido

88 - falha em dispositivo da rede

Os códigos que podem ser retornados pela funções introduzidas na versão 2 são:

Função	Códigos	Função	Códigos	Função	Códigos
38h	2	40h	5,6	48h	7,8
39h	3,5	41h	2,3,5	49h	7,9
3Ah	3,5,15	42h	1,6	4Ah	7,8,9
3Bh	3	43h	1,2,34	4Bh	1,2,3,5,8,10,11
3Ch	3,4,5	44h	1,3,5,6	4Eh	2,3,18
3Dh	2,3,4,5,12	45h	4,6	4Fh	18
3Eh	6	46h	4,6	56h	2,3,5,17
3Fh	5,6	47h	15	57h	1,6

7.3 FUNÇÕES DE CARACTER

As funções neste grupo acessam, implicitamente, os seguintes handles:

0 - "entrada padrão", inicialmente o dispositivo CON

1 - "saída padrão", inicialmente o dispositivo CON

3 - "dispositivo auxiliar padrão", inicialmente o dispositivo AUX

4 - "impressora padrão", inicialmente o dispositivo PRN

Todos estes dispositivos são acessados normalmente no modo "ASCII" (ver Capítulo 4). Os handles 0 e 1 podem ser redirecionados a nível da linha de comando.

As seguintes observações se aplicam a estas funções:

- Interrupção de programa ("Control Break")

  A maioria das funções testa se foi requisitada a interrupção do programa (via "control break" ou "control c").

- Eco da leitura

  Na maioria das funções de leitura é feito o ecodos caracteres lidos, isto é, eles são enviados também à saída padrão

- Auto-impressão

  Na maioria das escritas na saída padrão, o código pode também ser enviado à impressora padrão. Este efeito é ligado e desligado através da digitação de Control P na entrada padrão.

- Suspensão da escrita na saída padrão

  A digitação de Control S na entrada padrão suspende a escrita na saída padrão, através da maioria das funções, até que um outro código seja digitado.

- Códigos não ASCII

O tratador padrão do console devolve dois códigos quando da digitação de determinadas combinações de teclas: zero (indica que tem segundo código) e o código expandido("extended code"). As combinações e códigos são:

Combinação	código
Ctrl @	3
Shift Tab	15
Alt Q,W,E,R,T,Y,U,I,O,P	16 a 25
Alt A,S,D,F,G,H,J,K,L	30 a 38
Alt Z,X,C,V,B,N,M	44 a 50
F1 a F10	59 a 68
Home	71
Flecha para cima	72
Pg Up	73
<-	75
->	77
End	79
Flecha para baixo	80
Pg Dn	81
Ins	82
Del	83
Shift F1 a F10	84 a 93
Ctrl F1 a F10	94 a 103
Alt F1 a F10	104 a 113
Ctrl PrtSc	114
Ctrl <-	115
Ctrl ->	116
Ctrl End	117
Ctrl PgDn	118
Ctrl Home	119
Alt 1,2,3,4,5,6,7,8,9,0	120 a 131
Ctrl PgUp	132

Um cuidado importante a ser tomado é sempre ler o segundo código quando o primeiro for 0; caso contrário, o segundo código poderá vir a ser considerado como uma tecla.

- Códigos de controle na saída padrão

O tratador padrão do console trata apenas os seguintes caracteres como controle:

07h (BEL)	aciona o alto-falante
08h (BS)	recua o cursor uma posição, sem mudar de linha e sem afetar o caracter na nova posição
09h (TAB)	avança o cursor para a próxima "tabulação". As posições "tabuladas" são espaçadas de oito colunas a partir da coluna 9 (9, 17, 25, ...,73,1 da linha seguinte etc.)
0Ah (LF)	move o cursor para a linha de baixo, sem mudar de coluna. Se estiver na última linha da tela, ela é "rolada" para cima
0Dh (CR)	move o cursor para a primeira coluna da linha atual

Todos os demais códigos são escritos no vídeo, sendo o cursor avançado uma posição. Ao final da linha o cursor passa para o início da linha de baixo; ao final da tela ela é "rolada" para cima.

O programador CP/M sentirá a falta de códigos para efetuar funções como posicionamento de cursor, limpeza total ou parcial da tela etc. Isto deve ser resolvido de duas maneiras:

- Acesso ao console via BIOS
- Instalação de um tratador de console em substituição ao padrão (por exemplo, o ANSI.SYS ou o descrito no capítulo 11).

7.3.1 LEITURA DO CONSOLE (FUNÇÃO 01H)

<S> AL código lido

Aguarda um caracter estar disponível na entrada padrão, ecoa-o e trata interrupção de programa, suspensão de escrita e auto-impressão.

7.3.2 ESCRITA NO CONSOLE (FUNÇÃO 02H)

<E> DL código a escrever

Escreve o código fornecido na saída padrão, tratando interrupção de programa, suspensão de escrita e auto-impressão.

7.3.3 LEITURA DO DISPOSITIVO AUXILIAR (FUNÇÃO 03H)

<S> AL código lido

Aguarda um caracter estar disponível no dispositivo auxiliar padrão; trata interrupção de programa.

7.3.4 ESCRITA NO DISPOSITIVO AUXILIAR (FUNÇÃO 04H)

<E> DL código a escrever

Escreve o código fornecido no dispositivo auxiliar padrão, tratando interrupção de programa.

7.3.5 ESCRITA NA IMPRESSORA (FUNÇÃO 05H)

<E> DL código a escrever

Escreve o código fornecido na impressora padrão, tratando interrupção de programa.

7.3.6 ENTRADA E SAÍDA DIRETA NO CONSOLE (FUNÇÃO 06H)

<E> DL 0FFh leitura,
00h a 0FEh código a escrever

<S> somente se leitura:
ZF 1 se não há código disponível
ZF 0 se há código disponível
AL - código lido

Esta função não efetua tratamentos especiais (interrupção de programa, suspensão de escrita, eco e auto-impressão). Ao contrário da função "leitura sem espera" de tratadores, um código é lido se disponível, isto é, ele é retirado da fila.

7.3.7 ENTRADA DIRETA DO CONSOLE, SEM ECO (FUNÇÃO 07H)

<S> AL código lido

Esta função lê um código da entrada padrão, sem efetuar tratamentos especiais (interrupção de programa, eco e auto-impressão).

7.3.8 ENTRADA DO CONSOLE, SEM ECO (FUNÇÃO 08H)

<S> AL código lido

Esta função lê um código da entrada padrão, tratando interrupção de programa, porém não ecoa o caracter lido.

7.3.9 ESCRITA DE CADEIA NO CONSOLE (FUNÇÃO 09H)

<E> DS:DX endereço da cadeia

Esta função escreve uma cadeia de caracteres na saída padrão, tratando interrupção de programa, suspensão de escrita e auto-impressão. O fim da cadeia é marcado pelo caracter '$' (24h).

Os dois erros mais comuns no uso desta função são: tentar escrever uma cadeia que contém '$' no meio (o que termina a escrita antes do espera-do) e esquecer de colocar '$' no final (o que causa a escrita do que estiver na memória, até que um '$' seja encontrado).

7.3.10 LEITURA DO CONSOLE COM EDIÇÃO (FUNÇÃO 0AH)

<E> DS:DX endereço da área de leitura

Esta função permite a leitura de uma cadeia de caracteres da entrada padrão, oferecendo várias facilidades de edição do texto.

A área de leitura deve conter no primeiro byte o número máximo de caracteres a serem lidos; o segundo byte irá conter o número de caracteres efetivamente colocados na área (a partir do terceiro byte). Após o último caracter, é colocado o código CR (0Dh).

A edição propriamente dita não é feita sobre a área do usuário, mas sim sobre uma área interna do DOS. A área do usuário é utilizada como uma área auxiliar ("template", na nomenclatura da MicroSoft). Inicialmente é verificado se o segundo byte é menor que o primeiro e se a posição final da área contém um CR. Se isto não ocorre, a área do usuário é marcada como vazia. Durante a edição existem comandos (teclas F1, F2, F3 etc.) que possibilitam a cópia de caracteres da área do usuário para a área de edição.

Desta forma, se a área do usuário não for alterada, o texto fornecido na leitura anterior estará disponível ao operador.

Exemplificando: suponhamos que inicialmente a área contenha 13 no primeiro byte e zero nos demais. Ao chamarmos a função de leitura com edição, a área será considerada vazia. Se o operador digitar XIS.DAT <enter>, a área conterá:

```
0Dh 07h 'X' 'I' 'S' '.' 'D' 'A' 'T' 0Dh
```

Se a mesma área for utilizada para outra leitura com edição e o operador digitar F3, o conteúdo anterior (XIS.DAT) será apresentado.

Notar que o CR não é contado no segundo byte; se o primeiro byte contiver 13, a área deverá ter 16 bytes (13 + os dois primeiros + 1 para o CR). Perceber também a importância de não se reaproveitar a mesma área para leitura de respostas distintas (por exemplo, nome de arquivo e números), pois isto diminuirá o aproveitamento do texto anterior.

Interrupção de programa e auto-impressão são tratados.

7.3.11 INFORMA ESTADO DO CONSOLE (FUNÇÃO 0BH)

<S> AL 0FFh= existe código disponível para leitura

000h = não existe código disponível para leitura

Esta função informa a disponibilidade de caracteres na entrada padrão, tratando interrupção de programa.

7.3.12 LEITURA DO CONSOLE COM ESPERA (FUNÇÃO 0CH)

<S> AL função desejada:

01h - leitura do console

06h - entrada direta do console

07h - leitura direta do console, sem eco

08h - leitura do console, sem eco

0Ah - leitura do console, com edição

Esta função consome todos os códigos disponíveis para leitura na entrada padrão, de forma a obrigar a espera de uma digitação, e efetua a função especificada em AL.

Consultar os demais parâmetros necessários nas descrições das funções.

O uso comum desta função é para introduzir uma pausa ou evitar que um texto digitado anteriormente seja tratado como resposta a uma pergunta inesperada pelo operador (como uma mensagem de erro).

7.4 FUNÇÕES DE ARQUIVO/DIRETÓRIO

Estas funções permitem acesso a dispositivos, arquivos e diretórios, já descritos em detalhes no Capítulo 4.

Um conceito importante é o da busca de uma entrada em um diretório. Frequentemente o DOS recebe do usuário uma especificação de arquivo (eventualmente ambígua) e (opcionalmente) um atributo, e procura no diretório uma entrada que satisfaça estas especificações:

1) Se o bit de rótulo estiver ativo, apenas o rótulo (se existir) é encontrado; este bit prevalece sobre os demais;
2) Se um ou mais dos bits de diretório, sistema ou "escondido" estiver ativo, são encontrados os arquivos que tiverem estes bits ativos ou inativos, isto é, a busca inclui os arquivos normais. Os arquivos que tiverem ativos bits inativos no atributo fornecido não são encontrados; e
3) Os bits de alterado e protegido são ignorados.

Os atributos normalmente utilizados são:

00h - para acessar arquivos normais

06h - para acessar arquivos normais ou "escondidos" ou de sistema

08h - para acessar o volume

10h - para acessar subdiretórios e arquivos normais

16h - para acessar quaisquer entradas, exceto volume

Notar que não é possível procurar apenas subdiretórios.

7.4.1 REINICIA DISCO (FUNÇÃO 0DH)

Esta função, que não tem parâmetros nem retorna resultados, comanda a gravação de todos os buffers de disco que tenham gravações pendentes.

Isto nãosignifica que alterações de tamanho em arquivos serão gravadas no diretório, pois esta gravação só é comandada pela função de fechamento de arquivo.

Atenção: Nas primeiras versões do manual do DOS está indicado que esta função altera a unidade padrão e a DTA; na verdade isto nãoocorre em nenhuma versão do DOS.

7.4.2 SELECIONA UNIDADE PADRÃO (FUNÇÃO 0EH)

<E> DL nova unidade padrão (0 = A:, 1 = B: etc.)

<S> AL número de unidades de disco (ver a seguir)

A unidade especificada passa a ser a padrão; notar que a numeração difere da usada em outras funções (na qual 0 corresponde à unidade padrão).

O resultado retornado tem pouco significado. Nas versões 1 e 2 do DOS, se existir uma única unidade de disquete ela é contada duas vezes (por estar associada às unidades A: e B:); na versão 3 o valor retornado é o configurado como "LASTDRIVE", independendo do número real de unidades.

Para descobrir o número de disquetes e winchesters existentes devem ser utilizadas as funções do BIOS (INT 11h e INT 13h respectivamente).

A unidade padrão é informada pela função # 19h.

7.4.3 ABERTURA DE ARQUIVO VIA FCB (FUNÇÃO 0FH)

<E>	DS:DX	endereço da FCB	
<S>	AL	00h	se operação bem sucedida
		FFh	se ocorreu erro

Esta função é utilizada para iniciar um acesso a um dispositivo ou a um arquivo (já existente), utilizando FCB.

Inicialmente, o DOS procura o nome contido na FCB na cadeia de tratadores, ignorando a unidade e a extensão.

Se encontrou, os campos reservados são iniciados, os campos bloco atual e tamanho de arquivo são preenchidos com zero, o campo tamanho do registro com 128 (80h) e os campos de data e hora com a data e hora atuais.

Se não se trata de dispositivo, o DOS verifica se a unidade especificada é 0, substituindo-a pela padrão (1 = A:, 2 = B: etc.) em caso afirmativo. Desta forma, alterações posteriores da unidade padrão não afetarão a FCB já aberta. O nome de arquivo é então procurado no diretório corrente da unidade especificada. Caso seja encontrado, os campos reservados são iniciados, o campo bloco atual é preenchido com zero, o campo tamanho do registro com 128 (80h) e os campos de tamanho do arquivo, data e hora com os valores contidos no diretório.

Observações:

1) A partir da versão 3, o DOS passou a controlar o compartilhamento de arquivos; mesmo que o arquivo exista, pode ser negada a sua abertura, com a geração de um erro crítico - ver descrição da funcão 3Dh.
2) Caso seja desejado um tamanho de registro diferente de 128, este campo deve ser alterado após a chamada desta função. Antes de realizar um acesso sequencial, o campo número de registro no bloco deve ser iniciado; no caso de acesso randômico, deve-se iniciar o número relativo de registro.
3) A FCB pode conter um nome ambíguo; neste caso o DOS acessa o primeiro nome que encontrar no diretório que satisfaça a máscara fornecida, atualizando o nome na FCB. Isto não vale para dispositivos.
4) Na versão 2 é possível (utilizando FCB estendida) abrir um subdiretório através desta função. Entretanto, como o tamanho indicado no diretório é 0, ele não pode ser lido e não deve ser alterado. Alterando na FCB o tamanho para o valor correto (que pode ser obtido analisando a FAT), o subdiretório pode ser acessado como arquivo comum. Na versão 3 esta facilidade foi retirada.
5) Normalmente o retorno de 0FFh indica que o arquivo não foi encontrado.

7.4.4 FECHAMENTO DE ARQUIVO VIA FCB (FUNÇÃO 10H)

<E>	DS:DX	endereço da FCB	
<S>	AL	00h	se operação bem sucedida
		FFh	se ocorreu erro

Esta função deve ser sempre chamada ao final de operações que alterem um arquivo, pois ela atualiza o diretório e a grava os buffers pendentes referentes à unidade em que está o arquivo.

No caso de arquivos acessados apenas para leitura, esta chamada não é obrigatória, porém muito aconselhada (principalmente sob o DOS 3.1 e posteriores).

O retorno de 0FFh indica que o DOS não encontrou o arquivo no diretório, normalmente devido a uma troca ilícita de disco.

7.4.5 BUSCA PRIMEIRA ENTRADA, VIA FCB (FUNÇÃO 11H)

<E>	DS:DX	ponteiro para FCB	
<S>	AL	00h	se encontrou, DTA contém a entrada
		FFh	se não encontrou

Esta função inicia uma busca de entradas que obedeçam a uma determinada máscara. A FCB contém a máscara e é utilizada pelo DOS para armazenar, na área reservada, informações sobre onde deve ser continuada a busca. Por este motivo, ela não deve ser alterada durante as operações de busca.

Caso a FCB contenha 0 no campo de unidade, o DOS realiza a busca na unidade padrão, porém não altera a FCB, permitindo que a mesma FCB seja utilizada em outras buscas, mesmo que a unidade padrão seja alterada.

Se a FCB for estendida, o atributo especifica que tipos de entradas são procuradas. Se o bit de rótulo estiver ativo, a busca é realizada no diretório raiz; caso contrário, ela é realizada no diretório corrente da unidade especificada.

O resultado de uma busca bem sucedida é colocado na DTA, da seguinte forma:

1) Se a FCB de busca era estendida, é colocado no início da DTA 0FFh, 5 bytes com 0 e o atributo de busca;
2) Em seguida, é colocada a unidade em que foi realizada a busca (1 = A:, 2 = B: etc.); e,
3) Finalmente, nos 32 bytes seguintes é copiada a entrada do diretório.

Notar que, desta forma, obtém-se uma FCB do mesmo tipo que a de busca; no caso de FCB estendida, o atributo do arquivo encontrado deve ser obtido da entrada(deslocamento de 12h em relação ao início da DTA).

No caso de se especificar um dispositivo, AL retorna com 0 e a DTA é preenchida considerando-se a entrada de diretório como sendo o nome do dispositivo seguido de zeros.

Na versão 2, as entradas '.' e '..' de um subdiretório não são encontradas; a versão 3 corrige este problema.

7.4.6 BUSCA PRÓXIMA ENTRADA, VIA FCB (FUNÇÃO 12H)

<E>	DS:DX	ponteiro para FCB	
<S>	AL	00h	se encontrou uma entrada,
			DTA contém a entrada
		FFh	se não encontrou

Esta função procura uma nova entrada, a partir da informada em uma chamada a "Busca Primeira" ou a ela própria. Tipicamente as funções de busca são usadas da seguinte forma:

1) Monta-se a FCB de busca;
2) Chama-se a função "Busca Primeira";
3) Se não for encontrada entrada, isto significa que nenhum arquivo atende à máscara fornecida;
4) Efetua-se o processamento desejado no arquivo descrito na DTA;
5) Chama-se a função "Busca Próxima"; e
6) Se mais uma entrada foi encontrada, volta-se ao passo 4.

7.4.7 SUPRIME ARQUIVO, VIA FCB (FUNÇÃO 13H)

<E>	DS:DX	ponteiro para FCB	
<S>	AL	00h	se suprimiu pelo menos um arquivo
		FFh	se não encontrou nenhum arquivo

Esta função suprime os arquivos que correspondam à máscara e atributo especificados na FCB. Arquivos protegidos não são suprimidos, mesmo que seja ativado o bit correspondente do atributo; o arquivo deve ser desprotegido antes. Na versão 2 era possível suprimir subdiretórios, porém os clusters alocados aos arquivos descritos neles não eram liberados; a versão 3 não permite esta operação.

7.4.8 LEITURA SEQUENCIAL, VIA FCB (FUNÇÃO 14H)

<E>	DS:DX	endereço da FCB	
<S>	AL	00	leitura bem sucedida
		01	atingiu fim do arquivo, nada lido
		02	DTA insuficiente (ver abaixo)
		03	atingiu fim do arquivo, um registro incompleto foi lido e completado com zeros

Lê o registro indicado pelos campos "bloco atual" e "registro no bloco" da FCB. Assume que todos os registros no arquivo tenham o tamanho dado pelo campo "tamanho do registro". O registro lido é colocado na DTA e os campos "bloco atual" e "registro no bloco" são incrementados para apontar o registro seguinte.

Considerando o endereço da DTA como seg:off, o DOS considera que o seu tamanho é de 10000h - off bytes, retornando AL = 2, sem transferir nenhum byte, se este tamanho for insuficiente.

Não esquecer de abrir o arquivo e iniciar o campo "registro no bloco" antes de acessá-lo.

7.4.9 ESCRITA SEQUENCIAL, VIA FCB (FUNÇÃO 15H)

<E> DS:DX endereço da FCB

<S> AL 00 escrita bem sucedida

01 disco cheio

02 DTA insuficiente (ver abaixo)

Escreve no registro indicado pelos campos "bloco atual" e "registro no bloco" da FCB. Assume que todos os registros no arquivo tenham o tamanho dado pelo campo "tamanho do registro". O registro lido é escrito a partir dos dados na DTA e os campos "bloco atual" e "registro no bloco" são incrementados para apontar o registro seguinte.

Considerando o endereço da DTA como seg:off, o DOS considera que o seu tamanho é de 10000h - off bytes, retornando AL = 2, sem transferir nenhum byte, se este tamanho for insuficiente.

Não esquecer de abrir o arquivo e iniciar o campo "registro no bloco" antes de acessá-lo.

7.4.10 CRIA ARQUIVO, VIA FCB (FUNÇÃO 16H)

<E> DS:DX endereço da FCB

<S> AL 00h arquivo criado

FFh erro

Se já existe uma entrada com o nome e atributo fornecidos na FCB, ela é reaproveitada, sendo liberados os clusters a ela alocados (isto não ocorre no CP/M-80, no qual seriam criadas duas entradas com um mesmo nome). Se já existe o arquivo, porém com atributo diferente, é indicado erro.

Se não existe uma entrada com mesmo nome, o DOS procura uma entrada vazia. Se não encontrar (diretório cheio), retorna com AL = 0FFh.

Se foi encontrada uma entrada (anterior ou nova), ela é atualizada com o nome e atributo especificados na FCB, o tamanho do arquivo é preenchido com zero e os campos de data e hora recebem os valores atuais. Em seguida, o diretório é atualizado e o arquivo aberto, de forma análoga à função # 0Fh.

Se for especificado um dispositivo, esta função é análoga à função #0Fh. Não é permitido criar subdiretório.

7.4.11 RENOMEIA ARQUIVOS, VIA FCB (FUNÇÃO 17H)

<E> DS:DX endereço da FCB

<S> AL 00h arquivos renomeados

FFh erro

Esta função utiliza uma FCB alterada, que possui um segundo nome 6 bytes a partir do primeiro:

Deslocamento	tamanho	conteúdo
0	1 byte	unidade (0 = padrão, 1 = A:, etc.)
1	11 vytes	primeiro nome (nome antigo)
17	11 bytes	segundo nome (nome novo)

Todas as entradas que satisfaçam o primeiro nome e o atributo fornecido têm seu nome alterado para o segundo nome. As posições do segundo nome que contém '?' não são alteradas. Por exemplo:

Arquivos existentes:	XIS
	LAPIS
	XUXU
Nome antigo:	X?????????? (só afetar nomes que começam com X)
Nome novo:	Y?ABC (manter o segundo caracter do nome antigo)
Resultado:	YIABC
	LAPIS
	YUABC

O DOS retorna AL = 0FFh se

1) Não encontrou entrada que satisfaça o primeiro nome; ou
2) Já existe entrada com o nome novo

A lógica usada pelo DOS é bastante simples:

1) Procura a primeira entrada que satizfaça ao nome antigo; se não encontrou, retorna erro;
2) Constrói o novo nome e o procura no diretório. Se encontrar, retorna erro.;
3) Atualiza o nome da entrada; e
4) Procura a próxima entrada que satisfaça ao nome antigo; se encontrou volta ao passo 2).

Como consequência, temos que

1) O diretório é percorrido uma vez para cada entrada que atende ao nome antigo (procurando o nome novo) e uma vez para encontrar as entradas que atendem ao nome antigo;
2) Em caso de erro, não ficamos sabendo o motivo exato (se não encontrou nome antigo ou se encontrou nome novo); e
3) Antes de acusar erro por encontrar nome novo, alguns nomes podem ter sido trocados.

Esta função permite alterar o nome de um subdiretório.

7.4.12 INFORMA UNIDADE PADRÃO (FUNÇÃO 19H)

<S> AL unidade padrão (0 = A:, 1 = B: etc.)

A unidade padrão é alterada pela função número 0Eh.

7.4.13 ALTERA DTA (FUNÇÃO 1AH)

<E> DS:DX endereço da DTA

Na versão 1 não é possível ler o endereço da DTA; a partir da versão 2 temos a função #2Fh.

7.4.14 OBTÉM INFORMAÇÕES DA FAT DA UNIDADE PADRÃO (FUNÇÃO 1BH)

<S> AL número de setores por cluster

CX número de bytes por setor

DX número de clusters

DS:BX ponteiro para "media byte"

Na versão 1, DS:BX apontava para a FAT da unidade, que estava sempre residente na memória em endereço fixo; na versão 2 e posteriores, DS:BX aponta para o campo que contém o "media byte", na tabela de parâmetros de disco interna ao DOS.

A partir da versão 2, existe a função número 36h (Informa Espaço em Disco) que é mais útil para obter informações sobre a FAT.

7.4.15 OBTÉM INFORMAÇÕES DA FAT (FUNÇÃO 1CH)

<E> DL unidade de disco (0 = padrão, 1 = A:, 2 = B: etc.)

<S> AL número de setores por cluster

CX número de bytes por setor

DX número de clusters

DS:BX ponteiro para "media byte"

Esta função fornece as mesmas informações que a anterior, para qualquer unidade de disco.

7.4.16 LEITURA RANDÔMICA, VIA FCB (FUNÇÃO 21H)

<E> DS:DX endereço da FCB

<S> AL 00 leitura bem sucedida

01 atingiu fim do arquivo, nada lido

02 DTA insuficiente (ver abaixo)

03 atingiu fim do arquivo, um registro incompleto foi lido e completado com zeros

Lê o registro indicado pelo campo "registro relativo" da FCB. Assume que todos os registros no arquivo tenham o tamanho dado pelo campo "tamanho do registro". O registro lido é colocado na DTA e os campos "bloco atual" e "registro no bloco" são alterados para apontar o registro lido.

Considerando o endereço da DTA como seg:off, o DOS considera que o seu tamanho é de 10000h - off bytes, retornando AL = 2, sem transferir nenhum byte, se este tamanho for insuficiente.

Esta função existe por compatibilidade com o CP/M-80; a função 27h (Leitura de Bloco) é muito mais útil.

Notar que o acesso randômico ou sequencial seguinte irá acessar o mesmo registro, a não ser que alteremos a FCB. Notar também que as funções sequenciais não alteram o campo "registro relativo".

Não esquecer de abrir o arquivo e iniciar o campo "registro relativo" antes de acessá-lo.

7.4.17 ESCRITA RANDÔMICA, VIA FCB (FUNÇÃO 22H)

<E> DS:DX endereço da FCB

<S> AL 00 escrita bem sucedida

01 disco cheio

02 DTA insuficiente (ver abaixo)

Escreve no registro indicado pelo campo "registro relativo" da FCB. Assume que todos os registros no arquivo tenham o tamanho dado pelo campo "tamanho do registro". O registro é escrito com o dados na DTA e os campos "bloco atual" e "registro no bloco" são alterados para apontar o registro escrito.

Considerando o endereço da DTA como seg:off, o DOS considera que o seu tamanho é de 10000h - off bytes, retornando AL = 2, sem transferir nenhum byte, se este tamanho for insuficiente.

Esta função existe por compatibilidade com o CP/M-80; a função 28h (Escrita de Bloco) é muito mais útil.

Notar que o acesso randômico ou sequencial seguinte irá acessar o mesmo registro, a não ser que alteremos a FCB. Notar também que as funções sequenciais não alteram o campo "registro relativo".

Não esquecer de abrir o arquivo e iniciar o campo "registro relativo" antes de acessá-lo.

7.4.18 INFORMA TAMANHO DE ARQUIVO, VIA FCB (FUNÇÃO 23H)

<E> DS:DX endereço da FCB

<S> AL 00h encontrou arquivo

FFh não encontrou

Antes de chamar esta função, o campo "tamanho do registro" deve ser iniciado com o valor desejado. O DOS procura uma entrada que satisfaça o nome e atributo fornecidos e coloca no campo "registro relativo" o tamanho do arquivo, em registros, arredondado para cima.

Tipicamente, coloca-se 1 no tamanho do registro, para se obter o tamanho em bytes. A mesma informação pode ser obtida através da função "Busca Primeira" (# 11h) ou "Abre Arquivo" (#0Fh).

No caso de se especificar um dispositivo, o tamanho indicado é 0.

7.4.19 POSICIONA REGISTRO RELATIVO (FUNÇÃO 24H)

<E> DS:DX - endereço da FCB

Atualiza o campo "registro relativo" para apontar para o mesmo registro que os campos "bloco atual" e "registro no bloco". Facilita a passagem de acesso sequencial para randômico.

7.4.20 LEITURA DE BLOCO, VIA FCB (FUNÇÃO 27H)

<E> DS:DX endereço da FCB

CX número de registros a ler

<S> AL 00h leitura bem sucedida

01h atingiu fim do arquivo, nenhum registro lido parcialmente

02h DTA insuficiente (ver abaixo)

03h atingiu fim do arquivo; último registro foi lido parcialmente e completado com zeros.

CX número de registros lidos

Esta função é a mais flexível para leitura através de FCBs. Lê um bloco com até CX registros, do tamanho especificado na FCB. O registro inicial é definido pelo campo "registro relativo" da FCB, supondo que todos os registros tenham o mesmo tamanho. Os dados são colocados na DTA. Um máximo de 64K - 1 bytes podem ser transferidos.

Os campos "bloco atual", "registro no bloco" e "registro relativo" são atualizados para apontar para o registro seguinte ao último lido.

Considerando o endereço da DTA como seg:off, o DOS considera que o seu tamanho é de 10000h - off bytes, retornando AL = 2, sem transferir nenhum byte, se este tamanho for insuficiente.

Veja na função seguinte um exemplo de uso.

7.4.21 ESCRITA DE BLOCO, VIA FCB (FUNÇÃO 28H)

<E> DS:DX endereço da FCB

CX número de registros a escrever

<S> AL 00h escrita bem sucedida

01h disco cheio, nenhum registro foi escrito

02h DTA insuficiente (ver abaixo)

CX número de registros escritos

Esta função é a mais flexível para escrita através de FCBs. Escreve um bloco com até CX registros, do tamanho especificado na FCB. O registro inicial é definido pelo campo "registro relativo" da FCB, supondo que todos os registros tenham o mesmo tamanho. Os dados são obtidos da DTA. Um máximo de 64K - 1 bytes podem ser transferidos.

Os campos "bloco atual", "registro no bloco" e "registro relativo" são atualizados para apontar para o registro seguinte ao último escrito.

Considerando o endereço da DTA como seg:off, o DOS considera que o seu tamanho é de 10000h - off bytes, retornando AL = 2, sem transferir nenhum byte, se este tamanho for insuficiente.

Caso seja especificado CX = 0, esta função altera o tamanho do arquivo para o número de registros especificado pelo campo "registro relativo", alocando ou liberando clusters conforme necessário.

Vejamos um exemplo de uso, no qual copiamos o arquivo ARQ1 para o arquivo ARQ2, supondo ambos existentes:

```
        MOV     DX,OFFSET BUFFER
        MOV     AH,1AH
        INT     21H ;INICIA DTA

        MOV     DX,OFFSET FCB1
        MOV     AH,0FH
        INT     21H             ;ABRE ARQ1
        MOV     FCB1.TREG,1     ;REGISTROS DE 1 BYTE
        MOV     FCB1.REGLO,0    ;PRIMEIRO REGISTRO
        MOV     FCB1.REGHI,0
        MOV     DX,OFFSET FCB2
        MOV     AH,0FH
        INT     21H             ;ABRE ARQ2
        MOV     FCB2.TREG,1     ;REGISTROS DE 1 BYTE
        MOV     FCB2.REGLO,0    ;PRIMEIRO REGISTRO
        MOV     FCB2.REGHI,0
COP:
        MOV     DX,OFFSET FCB1
        MOV     CX,TAMBUF
        MOV     AH,27H          ;LE ATE' TAMBUF BYTES
        INT     21H             ; PARA BUFFER
        MOV     DX,OFFSET FCB2
        MOV     AH,28H          ;GRAVA BYTES LIDOS
        INT     21H             ; NO FINAL DE ARQ1, CX = 0
        OR      CX,CX           ; E ACERTA O TAMANHO DE ARQ2
        JNZ     COP

        MOV     DX,OFFSET FCB1
        MOV     AH,10H
        INT     21H             ;FECHA ARQ1
        MOV     DX,OFFSET FCB2
        MOV     AH,10H
        INT     21H             ;FECHA ARQ2
```

7.4.22 ANALISA NOME DE ARQUIVO (FUNÇÃO 29H)

<E> DS:SI endereço da linha a ser analisada

ES:DI área onde será montada uma FCB

AL controle:

bit 0 = 1 ignora delimitadores antes do nome

bit 1 = 1 altera unidade na FCB somente se uma foi forne-cida

bit 2 = 1 altera nome na FCB somente se um foi fornecido

bit 3 = 1 altera extensão na FCB somente se uma foi forne-cida

demais bits devem ser 0

<S>	AL	00h encontrado nome não ambíguo
		01h encontrado nome ambíguo
		FFh nome inválido
	DS:SI	endereço do caractere seguinte ao nome na linha

Esta função é muito útil na montagem de FCBs a partir de nomes fornecidos pelo operador.

A FCB montada consiste da unidade, nome e extensão do nome do arquivo; em AL pode-se indicar que a FCB já contém valores padrão que não devem ser alterados se não forem fornecidos. Caracteres '*' são expandidos no número apropriado de '?'.

Os delimitadores iniciais que podem ser ignorados são: ":", ".", ";", ",", "=", "+", TAB (09h) e espaço. Os terminadores de nome são estes caracteres, mais ">", "<", "|", "/", """", "[", "]" e caracteres de controle (inferiores a 20h). Notar que um nome lido pela função "Leitura do Console com Edição" possui um CR no final que serve como terminador.

7.4.23 LIGA/DESLIGA OPÇÃO DE VERIFICAÇÃO (FUNÇÃO 2EH)

<E>	AL	00h para desligar a opção
		01h para ligar a opção
	DL	deve conter 0 para as versões anteriores a 3

Quando ativa, a opção de verificação faz com que o DOS chame a função de escrita com verificação do tratador ao invés da de escrita. Normalmente, isto faz com que seja verificado se os dados escritos foram gravados corretamente.

A partir da versão 2 está disponível a função 54h que fornece o estado desta opção.

7.4.24 INFORMA ENDEREÇO DA DTA (FUNÇÃO 2FH)

(Somente a partir da versão 2)

<S>	ES:BX	endereço da DTA atual

O endereço da DTA é alterado pela função # 1Ah.

7.4.25 INFORMA ESPAÇO DISPONÍVEL EM DISCO(FUNÇÃO 36H)

(Somente a partir da versão 2)

<E>	DL	unidade (0 = padrão, 1 = A:, 2 = B:, etc)
<S>	AX	número de setores por cluster
		(0FFFFh se unidade inválida)
	BX	número de clusters disponíveis
	CX	bytes por setor
	DX	número total de clusters

Esta função é semelhante a "Obtém Informações da FAT" (# 1Ch). Para obter o número de bytes livres em uma unidade podemos proceder da seguinte forma:

```
        MOV     DL,UNID         ;UNIDADE DESEJADA
        MOV     AH,36H
        INT     21H
        CMP     AX,0FFFFh
        JE      INVAL           ;DESVIA SE INVALIDA
        MUL     CX              ;AX = BYTES POR CLUSTER
        MUL     BX              ;DXAX = ESPACO LIVRE
```

7.4.26 CRIA SUBDIRETÓRIO (MKDIR) (FUNÇÃO 39H)

(Somente a partir da versão 2)

<E> DS:DX endereço de cadeia ASCIIZ

<S> CY 0 se bem sucedida

1 se ocorreu erro, AX contém código do erro

A cadeia fornecida não pode conter nomes ambíguos, e todos os nomes, exceto o último, devem existir (apenas o último é criado).

7.4.27 REMOVE SUBDIRETÓRIO (RMDIR) (FUNÇÃO 3AH)

(Somente a partir da versão 2)

<E> DS:DX endereço de cadeia ASCIIZ

<S> CY 0 se bem sucedida

1 se ocorreu erro, AX contém código do erro

A cadeia fornecida não pode conter nomes ambíguos e deve especificar o nome de um subdiretório existente, vazio (somente com as entradas "." e "..") e que não pode ser o diretório atual.

Cuidado! O DOS retorna o código de erro 5 (acesso negado) ao invés de 16 (tentativa de remover diretório atual) se tentamos remover o diretório atual.

7.4.28 ALTERA DIRETÓRIO ATUAL (CHDIR) (FUNÇÃO 3BH)

(Somente a partir da versão 2)

<E> DS:DX endereço de cadeia ASCIIZ

<S> CY 0 se bem-sucedida

1 se ocorreu erro, AX contém código do erro

A cadeia ASCIIZ deve ter no máximo 64 caracteres e não pode conter nomes ambíguos. O diretório atual pode ser obtido através da função # 47h.

7.4.29 CRIA ARQUIVO, VIA CADEIA ASCIIZ (CREAT) (FUNÇÃO 3CH)

(Somente a partir da versão 2)

<E>	DS:DX	endereço de cadeia ASCIIZ
	CX	atributo do arquivo
<S>	CY	0 se bem-sucedida, AX contém handle
		1 se ocorreu erro, AX contém código do erro

Cria o arquivo especificado (se não existir) ou o esvazia; em seguida abre-o para leitura/escrita (ver função "Abre Arquivo, via Handle" a seguir). O atributo não pode especificar diretório.

7.4.30 ABRE ARQUIVO, VIA HANDLE (FUNÇÃO 3DH)

(Somente a partir da versão 2)

<E>	DS:DX	endereço de cadeia ASCIIZ
	AL	código de acesso (ver adiante)
<S>	CY	0 se bem-sucedida, AX contém handle
		1 se ocorreu erro, AX contém código do erro

Esta função abre arquivos normais, "escondidos" e de sistema; note que não é possível especificar o atributo.

O ponteiro de leitura/escrita é posicionado no primeiro byte do arquivo (ele pode ser alterado através da função 42h). A data e hora do arquivo pode ser obtida através da função 57h e o atributo através da 43h.

Na versão 2, o código de acesso tem três valores possíveis:

0 - abertura para leitura

1 - abertura para escrita

2 - abertura para leitura/escrita

Na versão 3, o código de acesso contém:

bit 7 - indicador de "herança"

se 1, os processos disparados pelo programa (via função EXEC) não "herdarão" este handle.

bits 6 a 4 - modo de compartilhamento

0 - compatibilidade

1 - impede leitura/escrita

2 - impede escrita

3 - impede leitura

4 - impede nada

bit 3 - deve ser 0

bits 2 a 0 - tipo de acesso

0 - leitura

1 - escrita

2 - leitura/escrita

O compartilhamento de arquivos não é controlado pelo DOS propriamente dito, mas sim pelo utilitário SHARE. Uma descrição detalhada do assunto foge do escopo deste livro; a seguir é dada apenas a idéia geral do mecanismo.

O modo "compatibilidade" corresponde a arquivos abertos através das funções de FCB, das de criação, ou das de handle que especifiquem este modo (por exemplo, as escritas para a versão 2 onde os bits 7 a 3 deviam ser 0). Nenhum controle de compartilhamento é feito se todos os acessos ao arquivo forem feitos no modo "compatibilidade"; este controle surge quando de uma segunda abertura.

Se o arquivo estiver protegido, o modo "compatibilidade" é automaticamente convertido pelo DOS em "impede escrita".

Caso um arquivo já esteja aberto em um modo diferente de "compatibilidade", tentativas de abri-lo neste modo ocasionarão erro crítico com indicação de "unidade sem disquete"; o código de erro expandido indicará "violação de compartilhamento".

Um arquivo aberto no modo "impede leitura e escrita" não pode ser aberto novamente, sendo portanto exclusivo.

Um arquivo só pode ser aberto no modo "impede escrita" se não estiver aberto para escrita nem no modo "compatibilidade" e só poderá ser aberto novamente para leitura em modo:

1. "impede nada", independente da primeira abertura;
2. "impede escrita" se a primeira abertura foi para leitura;
3. "impede leitura" se a primeira abertura foi para escrita.

Um arquivo só pode ser aberto no modo "impede leitura" se não estiver aberto para leitura nem no modo "compatibilidade" e só poderá ser aberto novamente para escrita em modo:

1. "impede nada", independente da primeira abertura;
2. "impede escrita" se a primeira abertura foi para leitura;
3. "impede leitura" se a primeira abertura foi para escrita.

Finalmente, um arquivo só pode ser aberto no modo "impede nada" se não estiver aberto no modo "compatibilidade" e poderá ser aberto novamente no modo:

1. "impede nada", independente da primeira abertura;
2. "impede escrita" se a primeira abertura foi para leitura;
3. "impede leitura" se a primeira abertura foi para escrita.

7.4.31 FECHA ARQUIVO, VIA HANDLE (FUNÇÃO 3EH)

(Somente a partir da versão 2)

<E> BX handle

<S> CY 0 se bem-sucedida

1 se ocorreu erro, AX contém código do erro

7.4.32 LEITURA VIA HANDLE (FUNÇÃO 3FH)

(Somente a partir da versão 2)

<E> BX handle

CX número de bytes a ler

DS:DX endereço da área de transferência

<S> CY 0 se bem-sucedida, AX contém o número de bytes lidos

1 se ocorreu erro, AX contém código do erro

Lê CX bytes a partir do apontado pelo ponteiro de leitura/escrita, que é avançado para o byte seguinte ao último lido. O endereço da área em que serão colocados os dados é passado explicitamente. O DOS "normaliza" internamente este endereço, permitindo, por exemplo, a leitura de 1000h bytes para 2000h:FF00h (o endereço é convertido para 2FF0h:0h e os dados colocados de 2FF0h:0000h a 2FF0h:0FFFh).

O ponteiro de leitura/escrita pode ser movimentado através da função # 42h.

7.4.33 ESCRITA VIA HANDLE (FUNÇÃO 40H)

(Somente a partir da versão 2)

<E> BX handle

CX número de bytes a escrever

DS:DX endereço da área de transferência

<S> CY 0 se bem-sucedida, AX contém o número de bytes escritos

1 AX contém código do erro

Escreve CX bytes a partir do apontado pelo ponteiro de leitura/escrita, que é avançado para o byte seguinte ao último escrito. O endereço da área em que serão colocados os dados é passado explicitamente. O DOS "normaliza" internamente este endereço, permitindo, por exemplo, a escrita de 1000h bytes a partir de 2000h:FF00h (o endereço é convertido para 2FF0h:0h e os dados obtidos de 2FF0h:0000h a 2FF0h:0FFFh).

Se o handle corresponder a um arquivo e AX retornar com número inferior ao contido em CX, o disco está cheio. No caso de escrita em dispositivo em modo ASCII, AX diferente de CX indica que existia o código 1Ah (Control-Z) nos dados.

Se for especificado CX = 0, o tamanho do arquivo é ajustado para a posição do ponteiro de leitura/escrita, sendo alocado ou liberado clusters conforme necessário.

O ponteiro de leitura/escrita pode ser movimentado através da função # 42h.

7.4.34 REMOVE ARQUIVO, VIA CADEIA ASCIIZ (UNLINK) (FUNÇÃO 41H)

(Somente a partir da versão 2)

<E> DS:DX endereço de cadeia ASCIIZ

<S> CY 0 se bem-sucedida

1 se ocorreu erro, AX contém código do erro

A cadeia ASCIIZ não pode conter nomes ambíguos. Esta função remove arquivos normais, "escondidos" e de "sistema", desde que não protegidos. Para remover subdiretórios, usar a função 3Ah.

7.4.35 MOVE PONTEIRO DE LEITURA/ESCRITA (LSEEK) (FUNÇÃO 42H)

(Somente a partir da versão 2)

<E> AL modo de movimentação (ver adiante)

BX handle

CXDX deslocamento a ser feito

<S> CY 0 se bem-sucedida,

DXAX contém a nova posição do ponteiro

1 se ocorreu erro,

AX contém código do erro

O ponteiro de leitura/escrita é movido conforme o método especificado em AL e o deslocamento fornecido em CX (word mais significativo) e DX (word menos significativo):

AL 0 movimentação em relação ao início do arquivo

1 movimentação em relação à posição atual

2 movimentação em relação ao final do arquivo

As operações de movimentação são feitas somando-se o deslocamento com o valor especificado (início, atual ou final), ignorando o vai-um do word mais significativo. Podemos, portanto, codificar deslocamentos negativos em complemento de dois (exemplo: deslocamento -2 corresponde a CX=0FFFFh e DX=0FFFEh).

Os usos mais comuns desta função são:

1. AL = 0 e CXDX = posição absoluta desejada
2. AL = 1 e CXDX = 0 para obter a posição atual
3. AL = 2 e CXDX = 0 para posicionar ponteiro no final e obter o tamanho do arquivo

7.4.36 INFORMA/ALTERA ATRIBUTO DE ARQUIVO (CHMOD) (FUNÇÃO 43H)

(Somente a partir da versão 2)

<E> DS:DX endereço de cadeia ASCIIZ

AL 0 para obter atributo

1 para alterar,

CX contém o novo atributo

<S> CY 0 se bem-sucedida,

CX contém o atributo

1 se ocorreu erro,

AX contém código do erro

A cadeia ASCIIZ não pode conter nomes ambíguos. Não é permitido alterar os bits de rótulo e diretório; quando AL = 1 estes bits têm de ser zero.

7.4.37 CONTROLE DE ENTRADA/SAÍDA (IOCTL) (FUNÇÃO 44H)

(Somente a partir da versão 2)

Esta função contém várias subfunções (uma delas com sub-subfunções) que envolvem o acesso a informações associadas a handles ou a passagem de comandos especiais a tratadores de dispositivos:

AL	função	
00h	obtém informações de handle	
01h	altera informações de handle	
02h	lê sequência de controle via handle	
03h	escreve sequência de controle via handle	
04h	lê sequência de controle via unidade	
05h	escreve sequência de controle via unidade	
06h	obtém estado de leitura	
07h	obtém estado de escrita	
08h	informa se disco removível (DOS 3)	
09h	informa se unidade local ou remota (DOS 3.1)	
0Ah	informa se handle local ou remoto (DOS 3.1)	
0Bh	altera repetição em falha de compartilhamento (DOS 3)	
0Dh	IOCTL genérico (DOS 3.2)	
	CL	**função**
	40h	altera parâmetros
	60h	obtém parâmetros
	41h	grava trilha em unidade lógica
	61h	lê trilha de unidade lógica
	42h	formata e verifica trilha em unidade lógica
	62h	verifica trilha em unidade lógica
0Eh	informa unidade lógica (DOS 3.2)	
0Fh	altera unidade lógica (DOS 3.2)	

Em todas as subfunções, em caso de erro é retornado CF = 1 e código do erro em AX.

Obtém/Altera Informações de Handle

<E>	AL	0	obtém informações
		1	altera informações,
	DH		deve conter 0
	DL		novas informações
	BX		handle
<S>	DX		informação

Estas funções permitem acessar as informações referentes ao dispositivo associado a um handle:

bit 14	1	se tratador suporta sequências de controle
		(funções chamadas por AL = 2, 3, 4 e 5)
bit 7	1	se dispositivo de caracter
	0	se dispositivo de bloco (arquivo)
se dispositivo de caracter		
bit 6	1	se ocorreu fim de arquivo na leitura
bit 5	1	se operando em modo binário,
	0	se operando em modo ASCII
bit 3	1	se dispositivo de relógio
bit 2	1	se dispositivo nulo
bit 1	1	se dispositivo de escrita no console
bit 0	1	se dispositivo de leitura no console
se dispositivo de bloco		
bit 6	1	se foi efetuada escrita
bits 5 a 0		contém o número do dispositivo (0 = A:, 1 = B: etc.)

Os bits 15, 13 a 8 e 4 são reservados e não devem ser alterados.

Transfere Sequências de Controle via Handle

<E>	AL	2	leitura
		3	escrita
	BX		handle
	CX		número de bytes a transferir
	DS:DX		área de dados para a transferência
<S>	AX		número de bytes transferidos

Transfere Sequências de Controle via Unidade

<E>	AL	4	leitura
		5	escrita
	BL	unidade (0 = padrão, 1 = A:, 2 = B: etc.)	
	CX	número de bytes a transferir	
	DS:DX	área de dados para a transferência	
<S>	AX	número de bytes transferidos	

Informa Estado de Dispositivo

<E>	AL	6	estado de leitura
		7	estado de escrita
	BX	handle	
<S>	AX	para arquivo:	
		00h	atingiu fim do arquivo
		FFh	não atingiu fim do arquivo
		para dispositivo de caracter:	
		00h	ocupado
		FFh	pronto

Esta função pode, por exemplo, ser usada para verificar se a impressora padrão (handle 4) está pronta antes de enviar um caracter via função 5

Informa se Disco Removível (DOS 3)

<E>	AL	8	
	BL	unidade (0 = padrão, 1 = A:, 2 = B: etc.)	
<S>	AX	00h	se disco removível
		01h	se disco fixo
		FFh	se unidade fornecida é inválida

Esta função permite sutilezas como, em não encontrando um arquivo, fornecer mensagens diferentes conforme o disco seja removível ("Arquivo não encontrado - coloque o disco correto e digite uma tecla") ou fixo ("Arquivo não encontrado - forneça outra unidade").

Informa se Unidade Local ou Remota (DOS 3.1)

<E>	AL	9
	BL	unidade (0 = padrão, 1 = A:, 2 = B: etc.)
<S>	DX	se local, atributo do cabeçalho do tratador
		se remoto, bit 12 está ativo

Esta função só tem sentido em ambientes de rede e deve ser evitada, visto ser desejável que programas acessem unidades locais e remotas de forma indistinta.

Informa se Handle Local ou Remota(DOS 3.1)

<E>	AL	0Ah
	BX	handle
<S>	DX	se local, atributo do cabeçalho do tratador
		se remoto, bit 15 está ativo

Esta função só tem sentido em ambientes de rede e deve ser evitada, visto ser desejável que programas acessem dispositivos locais e remotos de forma indistinta.

Altera Repetição em caso de Erro de Compartilhamento (DOS 3)

<E>	AL	0Bh
	CX	número de vezes a executar pausa
	DX	número de repetições

Toda vez que ocorre algum conflito de compartilhamento, o DOS repete a tentativa várias vezes. Esta função permite alterar o número de tenta-tivas e o tempo entre elas. O tempo é especificado pelo número de vezes que será executada a seguinte pausa:

```
        XOR CX,CX
PAUSA:
        LOOP PAUSA              ; REPETE 64 K VEZES
```

O DOS assume três repetições e execução de uma única pausa.

IOCTL genérico (DOS 3.2)

<E>	AL	0Dh
	BL	unidade de disco (0 = padrão, 1 = A:, 2 = B: etc.)
	CH	08H
	CL	função desejada:
		40h altera parâmetros
		60h informa parâmetros
		41h escreve trilha

61h lê trilha

42h formata e verifica trilha

62h verifica trilha

DS:DX endereço de bloco de parâmetros

Estas funções permitem a escrita de programas de acesso a disco trilha a trilha (como os utilitários FORMAT, DISKCOPY e DISKCOMP), sem que haja necessidade de chamar diretamente o BIOS (o que impede o seu uso com tratadores instalados).

Altera/Informa Parâmetros

Bloco de Parâmetros:

controle (byte)

Informa: bit 0 1 retorna BPB gerada por "Constrói BPB"

0 retorna BPB padrão

demais bits devem ser 0

Altera: bit 0 1 próximas chamadas a "Constrói BPB" devem retornar a BPB fornecida

0 próximas chamada a "Constrói BPB" devem retornar a BPB correspondente ao disco na unidade. A BPB fornecida é a nova BPB padrão

bit 1 1 ignorar todos os campos do bloco, exceto o formato da trilha

0 considerar todos os campos

bit2 1 todos os setores têm o mesmo tamanho e são numerados sequencialmente a partir de 1

0 setores podem ter tamanhos diferentes e/ou numeração não sequencial a partir de 1

demais bits devem ser 0; os bits 0 e 1 não podem ser ativados simultaneamente

tipo de dispositivo (byte)

0 disquete 5 1/4" 320/360 Kbytes

1 disquete 5 1/4" 1.2 Mbytes

2 disquete 3 1/2" 720 Kbytes

3 disquete 8" densidade simples

4 disquete 8" densidde dupla

5 disco rígido

6 unidade de fita

7 outro

este campo é preenchido pelo tratador

atributos do dispositivo (word)

bit 0 1 disco removível

0 disco fixo

bit 1 1 tem sensor de porta

0 não tem sensor de porta

demais bits reservados

este campo é preenchido pelo tratador

número de trilhas (word)

número máximo de trilhas suportado pelo dispositivo

este campo é preenchido pelo tratador

tipo de disco (byte)

utilizado em dispositivos que suportam mais de um tipo de disco, quando não existe outra forma de determinar o tipo. O tipo 0 é sempre o padrão. O caso comum é a unidade de disco de alta densidade do PC-AT que suporta dois tipos de disco:

0 - disquete de 1.2 Mbyte (padrão)

1 - disquete de 320/360 Kbytes

BPB (28 bytes)

Desloc.	tamanho	conteúdo
00h	1 word	número de bytes por setor
02h	1 byte	número de setores por cluster
03h	1 word	número de setores reservados
05h	1 byte	número de cópias da FAT
06h	1 word	número de entradas no diretório raiz
08h	1 word	número total de setores no disco
0Ah	1 byte	"media descriptor"
0Bh	1 word	número de setores por cópia da FAT
0Ch	1 word	número de setores por trilha
0Eh	1 word	número de faces
10h	1 word	número de setores "escondidos"
12h	1 dword	reservado
16h	6 bytes	reservado, devem conter 0

formato de uma trilha

O tratador não necessita manter o formato de trilha para cada unidade; é responsabilidade do DOS (quando do acesso normal a arquivos) e do programador (quando do acesso trilha a trilha) atualizar o formato quando necessário. Altera atualiza sempre este parâmetro; Informa não utiliza este campo. O primeiro word deste campo deve conter o número de setores por trilha; para cada setor devem ser fornecidos dois words: o número do setor e o seu tamanho em bytes.

Lê/Escreve Trilha

Bloco de Parâmetros:

controle (byte)

deve conter zero

face (word)

cilindro (word)

setor inicial (word)

os setores são numerados a partir de 0

número de setores (word)

endereço de transferência (dword)

Formata/Verifica Trilha

Bloco de Parâmetros:

controle (byte)

<E>	bit 0	1	testa se suporta formato
		0	formata ou verifica
<S>	bit 0	1	não suporta formatos de trilha não padrão
		0	suporta formatos de trilha não padrão
	bit 1	1	o formato atual (definido por Altera Parâmetros) não é suportado

face (word)

trilha (word)

Informa Unidade Lógica

<E>	AL	0Eh	
	BL	unidade (0 = default, 1 = A:, 2 = B: etc.)	
<S>	AL	0	só existe uma unidade lógica associada à unidade física
		<> 0	existe mais de uma unidade lógica associada à unidade física, AL contém a última acessada (1 = A:, 2 = B: etc.)

O exemplo típico de uma unidade física com mais de uma unidade lógica associada é quando temos uma única unidade de disquete; ela corresponde à unidade A: e B:, com o DOS pedindo automaticamente a colocação do disco apropriado.

Altera Unidade Lógica

<E>	AL	0Fh	
	BL	unidade (0 = default, 1 = A:, 2 = B: etc.)	
<S>	AL	0	só existe uma unidade lógica associada à unidade física
		<> 0	existe mais de uma unidade lógica associada à unidade física, AL contém a nova unidade lógica (1 = A:, 2 = B: etc.)

O exemplo típico de uma unidade física com mais de uma unidade lógica associada é quando temos uma única unidade de disquete; ela corresponde à unidade A: e B:, com o DOS pedindo automaticamente a colocação do disco apropriado.

Esta função deve ser utilizada quando foi realizada uma troca não conhecida pelo DOS (por exemplo, para acesso trilha a trilha).

7.4.38 DUPLICA HANDLE (DUP) (FUNÇÃO 45H)

(Somente a partir da versão 2)

<E>	BX	handle a duplicar	
<S>	CY	0	se bem-sucedida,
			AX contém novo handle
		1	se ocorreu erro,
			AX contém código do erro

O novo handle aponta para a mesma entrada na tabela de arquivos (interna ao DOS) que o fornecido. Desta forma, toda ação efetuada sobre um deles afeta o outro; notar que existe um único ponteiro de leitura/escrita para ambos.

7.4.39 DUPLICAÇÃO FORÇADA DE HANDLE (FORCDUP) (FUNÇÃO 46H)

(Somente a partir da versão 2)

<E>	BX	handle original	
	CX	handle duplicata	
<S>	CY	0	se bem-sucedida
		1	se ocorreu erro,
			AX contém código do erro

Esta função é semelhante à anterior, porém o programador especifica qual o handle "duplicata" (se ele já estava aberto, é fechado antes da duplicação).

Um uso para esta função é no redirecionamento:

```
ARQ     DB      'ERROS',0

        MOV     DX,OFFSET ARQ
        MOV     CX,0
        MOV     AH,3CH
        INT     21H             ;CRIA ARQUIVO, AX = HANDLE AUXILIAR
        JC      ERRO
        MOV     BX,AX
        MOV     CX,2            ;DISPOSITIVO DE ERRO PADRAO
        MOV     AH,46H
        INT     21H             ;HANDLE 2 PASSA A REFERIR AO ARQ.
        MOV     AH,3EH
        INT     21H             ;FECHA HANDLE AUXILIAR

; ESCRITAS NO HANDLE 2 VAO PARA ARQUIVO AO INVES DE CONSOLE
```

7.4.40 INFORMA DIRETÓRIO CORRENTE (FUNÇÃO 47H)

(Somente a partir da versão 2)

<E> DS:SI endereço de área do usuário

DL unidade (0 = padrão, 1 = A:, 2 = B: etc.)

<S> CY 0 se bem-sucedida,

área contém nome do diretório

1 se ocorreu erro,

AX contém código do erro

Esta função coloca o nome do diretório corrente em uma área do usuário, que deve ter pelo menos 64 bytes. O nome é terminado por um zero e não contém a unidade nem a barra ('\') inicial; se o diretório corrente for o raiz, a área conterá apenal o zero final.

O diretório corrente é alterado através da função 3Bh. Na versão 2, se o nome fornecido para a função 3Bh continha letras minúsculas, a função 47h as devolve inalteradas; na versão 3 elas são convertidas para maiúsculas.

7.4.41 PROCURA PRIMEIRA ENTRADA, VIA CADEIA ASCIIZ (FUNÇÃO 4EH)

(Somente a partir da versão 2)

<E> DS:DX endereço de cadeia ASCIIZ

CX atributo de busca

<S> CY 0 se bem-sucedida, DTA contém informações

1 se ocorreu erro,

AX contém código do erro

Esta função é semelhante à "Busca Primeira Entrada, via FCB". A DTA, entretanto, contém informações para continuar a busca, além de informações sobre o diretório. Estas informações estão também em um formato mais compacto. O formato da DTA é:

Deslocamento	tamanho	conteúdo
00h	21 bytes	reservado ao DOS, contém informações para prosseguimento da busca
15h	1 byte	atributo do arquivo
16h	2 bytes	hora da última alteração
18h	2 bytes	data da última alteração
1Ah	1 dword	tamanho do arquivo
1Eh	13 bytes	nome do arquivo, terminada por um zero. Se existir extensao, ela é precedida por '.'

É importante destacar a diferença entre procurar a entrada correspondente a um subdiretório e procurar as entradas definidas dentro de um subdiretório. Por exemplo, se \SUBDIR é um subdiretório, para procurar as entradas definidas dentro dele devemos fornecer a cadeia \SUBDIR*.*

Na versão 2 o diretório raiz (cadeia ASCIIZ "\") não era encontrado; a versão 3 corrige este problema.

Veja exemplo na próxima função.

7.4.42 PROCURA PRÓXIMA ENTRADA, VIA CADEIA ASCIIZ (FUNÇÃO 4FH)

(Somente a partir da versão 2)

<E> DTA informação obtida por "Procura Primeira" ou "Procura Próxima"

<S> CY 0 se bem-sucedida, DTA contém informações

1 se ocorreu erro,

AX contém código do erro

Esta função prossegue a busca iniciada por "Procura Primeira". A seguir temos um exemplo de procedimento para processar um conjunto de arquivos a partir de uma especificação fornecida pelo usuário:

1) Acrescenta-se "*.*" no final do nome, nos seguintes casos especiais:
   a) foi fornecido nome vazio
   b) foi fornecida somente unidade
   c) foi fornecido nome terminado por "\"
2) Chama-se "CHMOD" (# 43h), passando o nome fornecido. Temos três possibilidades:
   a) foi encontrado um arquivo; o usuário forneceu um nome não ambíguo que deve ser processado;
   b) foi encontrado um diretório; o nome fornecido pelo usuário deverá ser concatenado com os nomes que serão obtidos no passo 3, usando como cadeia para a busca o nome fornecido acrescentado de "*.*"; ou
   c) nada foi encontrado; provavelmente foi fornecido um nome ambíguo, que será usado para realizar buscas no passo 3. O nome do diretório a ser concatenado com os nomes obtidos é o fornecido menos o último nome (percorrer da direita para a esquerda até encontrar uma '\').
3) Chama-se "Procura Primeira" com a cadeia de busca determinada no passo 2; se nada for encontrado é acusado erro. Para obter o nome do arquivo a ser processado, concatena-se o diretório determinado

no passo com o nome contido na DTA. Dá-se continuidade ao processo chamando-se "Procura Próxima" até não se encontrar mais arquivos.

7.4.43 INFORMA OPÇÃO DE VERIFICAÇÃO (FUNÇÃO 54H)

(Somente a partir da versão 2)

<S> AL 0 opção de verificação inativa

1 opção de verificação ativa

A opção de verificação pode ser alterada através da função 2Eh.

7.4.44 RENOMEIA ARQUIVO, VIA CADEIA ASCIIZ (FUNÇÃO 56H)

(Somente a partir da versão 2)

<E> DS:DX endereço de cadeia ASCIIZ contendo nome antigo

ES:DI endereço de cadeia ASCIIZ contento nome novo

<S> CY 0 se bem-sucedida

1 se ocorreu erro,

AX contém código do erro

As cadeias não podem conter nomes ambíguos. Podem ser especificados caminhos diferentes nos dois nomes; neste caso a entrada de diretório é movida de um diretório para outro.

Na versão 3 é possível alterar o nome de um subdiretório; esta facilidade não está disponível na versão 2. Em ambas as funções é possível alterar nomes de arquivos normais, escondidos e de sistema, desde que não protegidos.

7.4.45 INFORMA/ALTERA DATA E HORA DE ARQUIVO, VIA HANDLE (FUNÇÃO 57H)

(Somente a partir da versão 2)

<E> BX handle

AL 0 para obter data e hora

1 para alterar data e hora:

CX hora

DX data

<S> CY 0 se bem-sucedida,

CX hora

DX data

1 se ocorreu erro,

AX contém código do erro

A data e a hora estão no formato do diretório, isto é

CX = hora shl 11 + minuto shl 5 + segundo shr 2

DX = (ano - 1980) shl 9 + mes shl 5 + dia

Antes de chamar esta função o arquivo deve ser aberto; após alterar a data ou hora o arquivo deve ser fechado para que o diretório seja atualizado.

7.4.46 CRIA ARQUIVO ÚNICO (FUNÇÃO 5AH)

(Somente a partir da versão 3)

<E> DS:DX endereço de cadeia ASCIIZ

CX atributo do arquivo

<S> CY 0 se bem-sucedida,

AX handle

DS:DX nome completo do arquivo

1 se ocorreu erro,

AX contém código do erro

Esta função recebe o nome de um subdiretório, terminado por "\", e cria nele um arquivo de nome diferente dos já existentes. Exceto pelo nome "estranho", este arquivo é igual a todos os outros e esta função é semelhante à 3Ch.

7.4.47 CRIA ARQUIVO NOVO, VIA CADEIA ASCIIZ (FUNÇÃO 5BH)

(Somente a partir da versão 3)

<E> DS:DX endereço de cadeia ASCIIZ

CX atributo do arquivo

<S> CY 0 se bem-sucedida,

AX contém handle

1 se ocorreu erro,

AX contém código do erro

Esta função é idêntica à 3Ch, exceto que retorna erro se o arquivo já existir.

7.4.48 CONTROLA ACESSO A ARQUIVO (FUNÇÃO 5CH)

(Somente a partir da versão 3)

<E>	AL	0	Protege ("Lock")
		1	Desprotege ("Unlock")
	BX	handle	
	CXDX	deslocamento inicial	
	SIDI	tamanho	
<S>	CY	0	se bem-sucedida
		1	se ocorreu erro,
			AX contém código do erro

Esta função se destina a controlar o acesso a regiões de um arquivo, de forma a permitir o seu compartilhamento entre dois processos. Se um processo tenta ler ou escrever na região protegida por outro o DOS efetua algumas tentativas (o número pode ser controlado por IOCTL). Se a região continuar protegida é retornado erro.

Um programa deve manter o mínimo de regiões simultaneamente potegidas, e mantê-las protegidas o mínimo tempo necessário. As proteções têm que ser retiradas antes de fechar o arquivo e antes do programa terminar. O programa deve assumir as interrupções 23h e 24h para garantir a retirada de proteções em caso do programa ser abortado.

Quando da retirada de proteção, deve-se especificar a mesma região que foi especificada quando da sua colocação.

7.5 FUNÇÕES DE GERENCIAMENTO DE MEMÓRIA

A partir da versão 2 o DOS dispõe de um gerenciamento da memória, controlando quais os blocos livres e em uso.

Todo bloco de memória começa em um endereço múltiplo de 16 bytes e possui nos 16 bytes anteriores um bloco de controle:

Deslocamento	tamanho	conteúdo
0	1 byte	'Z' se último bloco 'M' no caso contrário
1	1 word	0 se bloco livre PSP de quem alocou se em uso
3	1 word	tamanho do bloco em parágrafos (1 parágrafo = 16 bytes)
5	11 bytes	reservado

O DOS guarda internamente o endereço do primeiro bloco, estabelecendo assim uma lista ligada:

Bloco	endereço	tamanho	identificação
1	e_1	t_1	'M'
2	$e_1 + t_1 + 16$	t_2	'M'
...			
N	$e_{(n-1)}+t_{(n-1)}+16$	t_n	'Z'

O usuário não "enxerga" os blocos de controle, sendo utilizado nas chamadas ao DOS o endereço do bloco de memória.

Normalmente, quando um programa é carregado pelo COMMAND, toda a memória disponível está alocada para ele (a não ser que seja .EXE e o cabeçalho indique em contrário). Uma providência saudável é liberar a memória que não será usada.

7.5.1 ALOCA MEMÓRIA (FUNÇÃO 48H)

(Somente a partir da versão 2)

<E> BX número de parágrafos desejado

<S> CY 0 se bem-sucedida,

AX:0 aponta para o bloco alocado

1 se ocorreu erro,

AX contém código do erro

BX contém tamanho do maior bloco disponível

Esta função procura um bloco livre com o tamanho maior ou igual ao desejado. Se encontrar, quebra-o em duas partes, alocando a primeira ao programa e marcando o restante como livre. Caso contrário, informa o tamanho do maior bloco disponível.

Para alocar a maior quantidade possível de memória contígua, pode-se utilizar a seguinte sequência:

```
        MOV     BX,0FFFFH
        MOV     AH,48H
        INT     21H             ;BX = TAMANHO DO MAIOR BLOCO
        MOV     AH,48H
        INT     21H             ;ALOCA: AX = ENDERECO, BX = TAMANHO
```

Normalmente a busca é feita a partir do endereço mais baixo (o chamado "first fit"). Na versão 3 existe a possibilidade (não documentada) de se escolher outros dois algoritmos: o "best fit" (aloca o bloco de tamanho mais próximo do desejado) e o "last fit" (aloca o bloco de endereço mais alto que satisfaça o pedido).

7.5.2 LIBERA BLOCO DE MEMÓRIA (FUNÇÃO 49H)

(Somente a partir da versão 2)

<E> ES:0 endereço inicial do bloco de memória

<S> CY 0 se bem-sucedida

1 se ocorreu erro,

AX contém código do erro

Libera totalmente um bloco de memória, marcando-o como livre. Para liberar parte de um bloco de memória utilize a função seguinte.

7.5.3 MODIFICA TAMANHO DE BLOCO DE MEMÓRIA (FUNÇÃO 4AH)

(Somente a partir da versão 2)

<E> ES:0 endereço inicial do bloco

BX novo tamanho desejado

<S> CY 0 se bem-sucedida

1 se ocorreu erro,

AX contém código do erro

se não conseguiu aumentar tamanho,

BX contém o máximo tamanho possível

Esta função permite diminuir ou aumentar o tamanho de um bloco. Só é possível aumentar se existir um bloco livre contíguo ao bloco em questão.

7.5.4 INFORMA/ALTERA ALGORITIMO DE ALOCAÇÃO (FUNÇÃO 58H)

(Somente a partir da versão 3, não documentada)

<E>	AL	0	Informa
		1	Altera
	BL	0	"first fit"
		1	"best fit"
		2	"last fit"
<S>	CY	0	se bem-sucedida; se Informa
			AL algoritimo (0, 1 ou 2)
		1	se ocorreu erro,
			AX contém código do erro

Esta função permite selecionar o algoritmo de escolha do bloco a alocar; veja descrição da função 48h

7.6 FUNÇÕES DE GERENCIAMENTO DE PROGRAMAS

7.6.1 TERMINA PROGRAMA (FUNÇÃO 0 OU INT 20H)

<E> CS Segmento da PSP

Esta função termina o programa sendo executado através dos seguintes passos:

1) Restaura os vetores 22h (endereço de terminação), 23h (Interrupção de Programa) e 24h (Erro Crítico), a partir dos valores contidos na PSP.
2) Fecha todos os handles ainda abertos; apesar disto, a documentação do DOS diz ser obrigatório fechar os arquivos alterados (somente a partir da versão 2).
3) Grava todos os buffers com gravação pendente.
4) Libera todos os blocos de memória alocados para o programa (somente a partir da versão 2).
5) Obtém da PSP atual o endereço da PSP "pai", que passa a ser a nova PSP atual (somente a partir da versão 2).
6) Passa o controle para o endereço contido no vetor 22h. Embora não documentado, a partir da versão 2, SS e SP são restaurados a partir da PSP atual e os registradores DS, ES, BP, SI, DI, DX, CX, BX, AX e Flags são desempilhados (foram empilhados pela chamada à função 4Bh que "disparou" a execução do programa que acaba de ser terminado.

Esta função exige que CS contenha o segmento da PSP. Isto ocorre nos programas .COM mas não nos programas .EXE; uma solução é mostrada abaixo:

```
DADOS    SEGMENT                 ;SEGMENTO DE DADOS
FIM      LABEL   DWORD           ;APONTA PARA INT 20h NA PSP
FIM_OFF  DW      0
FIM_SEG  DW      ?
DADOS ENDS

; INICIO DA EXECUCAO
         MOV     AX,DADOS
         MOV     DS,AX           ;PARA ACESSAR FIM
         MOV     FIM_SEG,ES      ;SALVA ENDERECO DA PSP

; TERMINO DA EXECUCAO
         JMP     FIM
```

A partir da versão 2, existe a função 4Ch ("Termina Processo"), que deve ser usada preferencialmente.

7.6.2 TERMINA E MANTÉM RESIDENTE (INT 27H)

<E> CS segmento da PSP

DX offset final utilizado

Esta função perdeu significado a partir da versão 2, quando foi introduzida a função 31h ("Termina Processo e Mantém Residente").

Em DX deve ser fornecido o offset (em relação à PSP) da primeira posiçao não utilizada, o que limita o programa a 64 Kbytes. O DOS termina o programa de forma semelhante à função 0, exceto que:

1) não fecha handles;
2) não libera memória alocada; e
3) "encolhe" o bloco onde está a PSP conforme o valor passado em DX.

Na versão 1 o gerenciamento de memória consistia apenas em uma variável que indicava em qual endereço eram carregados programas; nesta versão, esta função alterava o endereço conforme DX e prosseguia com uma terminação normal.

7.6.3 CRIA NOVA PSP (FUNÇÃO 26H)

<E> CS segmento da PSP

DX segmento onde será construída uma PSP

Esta é outra função "morta" pela versão 2 do DOS. Seu uso era para facilitar a carga de programas, criando uma PSP. A função 4Bh realiza esta tarefa de forma mais completa.

A PSP é criada copiando-se os 256 bytes de CS:0 para DX:0 e atualizando na nova PSP o tamanho do segmento da PSP e os endereços de Terminação, Interrupção e Erro Crítico.

7.6.4 TERMINA PROCESSO E MANTÉM RESIDENTE (FUNÇÃO 31H)

(Somente a partir da versão 2)

<E> AL Código de Retorno

DX Tamanho de Memória em Parágrafos

A partir da versão 2, esta é a forma recomendada de terminar um programa mantendo-o residente. Seu funcionamento é idêntico ao descrito no "Termina e Mantém Residente", exceto que DX contém o tamanho em parágrafos, permitindo manter residente mais de 64 Kbytes, e não é necessário que CS aponte para a PSP; o DOS utiliza a PSP atual.

O código de retorno pode ser obtido pelo programa "pai", através da função 4Dh.

Os handles e áreas de memória desnecessárias devem ser liberados; em particular pode-se liberar o bloco que contém o "environment" (chamar a função 49h com ES contendo o valor em PSP:2Ch).

Veja no Capítulo 10 uma discussão mais completa sobre programas residentes.

7.6.5 CARREGA E EXECUTA PROGRAMA (EXEC) (FUNÇÃO 4BH)

(Somente a partir da versão 2)

<E> AL 0 Carrega e Executa

3 Carrega apenas

DS:DX cadeia ASCIIZ com nome do arquivo a carregar

ES:BX endereço de bloco de parâmetros

<S> CY 0 se bem-sucedida

1 se ocorreu erro,

AX contém código do erro

Esta função permite carregar um programa e executá-lo. O programa pode ser .COM ou .EXE (independente do nome do arquivo); se ele tiver mais de 28 bytes e os dois primeiros forem 4Dh e 5Ah, é considerado .EXE.

Na versão 2, o carregador de programas não está no DOS, mas sim na parte transiente do COMMAND. Para usar esta função nesta versão é necessário cerca de 2 Kbytes para o carregador; além disso, ele pode ter que ser recarregado de disco.

Temos duas subfunções:

1) Carrega e Executa

Esta função é utilizada para disparar um programa "filho". O bloco de parâmetros tem o seguinte formato:

Deslocamento	tamanho	conteúdo
00h	1 word	segmento do "environment" a ser copiado (0 = endereço obtido da PSP do pai)
02h	1 dword	endereço da linha de comando a ser colocada em PSP:80h
06h	1 dword	endereço da FCB a ser copiada para PSP:5Ch
0Ah	1 dword	endereço da FCB a ser copiada para PSP:6Ch

Os seguintes passos são seguidos pelo DOS:

a) determina o tamanho do "environment" a ser copiado, aloca um bloco de memória deste tamanho e efetua a cópia;
b) aloca um bloco no qual serão colocados a PSP e o programa;
c) cria a PSP do filho, copiando a PSP do pai. Atualiza os endereços de Terminação, Interrupção e Erro Crítico a partir dos vetores 22h, 23h e 24h. Atualiza os valores em PSP:2 e PSP:6 para corresponder ao tamanho do bloco alocado. Atualiza o endereço da PSP pai e do "environment". Copia para PSP:5Ch e PSP:6Ch os 11 bytes apontados pelos respectivos endereços no bloco de parâmetros. Copia para PSP:80h os 128 bytes indicados pelo bloco de parâmetros;
d) anota a duplicação de todos os handles (na versão 3 os handles abertos com o bit de "herança" = 1 não são duplicados, sendo acertada a tabela de handles na PSP);
e) anota nos dois blocos alocados o endereço da PSP do filho, que será portanto o "dono" destes blocos. Esta PSP passa a ser a PSP atual;
f) põe no vetor 22h o endereço da instrução seguinte à chamada ao DOS; e
g) inicia os registradores e desvia para o início do programa filho.

O programa "pai" deve, portanto, liberar memória para a carga do "filho". Ao final da execução do filho, o pai prossegue na instrução seguinte à chamada ao DOS. Deve-se admitir que, se CY = 0, todos os registradores podem ter sido alterados (na realidade o DOS procura mantê-los, porém isto não é documentado).

2) Carrega apenas

Esta função se destina a carregar um "overlay". O bloco de parâmetros tem o seguinte formato:

Deslocamento	tamanho	conteúdo
00h	1 word	segmento onde o programa será carregado
02h	1 dword	valor para relocação de .EXE

Neste caso, o "pai" é quem aloca (ou já contém) a área em que o "filho" será carregado. O programa carregado deve ser visto como uma subrotina, não como um "processo filho". Normalmente o valor de relocação é igual ao segmento de carga; só precisam ser diferentes se o programa será movido na memória antes de ser executado.

Atenção: A versão 2.00 possui um erro de lógica ao salvar o conteúdo de SS e SP na PSP. Estes valores são salvos em SS:2Eh, o que obriga SS a apontar para a PSP.

7.6.6 TERMINA PROCESSO (EXIT) (FUNÇÃO 4CH)

(Somente a partir da versão 2)

<E> AL Código de Retorno

Esta função é equivalente à 00h, exceto que CS não precisa apontar para a PSP (o DOS utiliza a PSP atual) e permite fornecer um código de retorno (que o programa "pai" pode obter através da função 4Dh).

7.6.7 INFORMA CÓDIGO DE TÉRMINO (WAIT) (FUNÇÃO 4DH)

(Somente a partir da versão 2)

<S> AH Tipo de Terminação:

0 Normal

1 Interrompido ("Ctrl Break")

2 Erro Crítico

3 Termina e Mantém Residente

AL Código de Retorno

Esta função informa (uma única vez)o código de retorno do processo anterior. As chamadas seguintes à primeira retornam AX = 0. Caso o programa tenha terminado por uma das funções "antigas" (00h ou INT 27h), o código de retorno informado será 0.

O COMMAND utiliza esta função para obter o código de retorno dos programas que executou e armazená-lo para teste através de "ERRORLEVEL".

7.6.8 ALTERA PSP ATUAL (FUNÇÃO 50H)

(Somente a partir da versão 2, não documentada)

<E> BX nova PSP atual

Esta função permite alterar a PSP atual, não devendo ser utilizada normalmente. Veja uma discussão a respeito no Capítulo 10.

7.6.9 INFORMA PSP ATUAL (FUNÇÃO 51H)

(Somente a partir da versão 2, não documentada)

<S> BX PSP atual

Na versão 3 existe uma versão "oficial" desta função, a de número 62h.

7.6.10 INFORMA PSP ATUAL (FUNÇÃO 62H)

(Somente a partir da versão 3)

<S> BX PSP atual

Esta função é idêntica à 51h (não documentada).

7.7 OUTRAS FUNÇÕES

7.7.1 ALTERA VETOR DE INTERRUPÇÃO (FUNÇÃO 25H)

<E> AL vetor a alterar

DS:DX endereço a colocar no vetor

Esta função é a forma "limpa" de alterar um vetor de interrupção. A partir da versão 2 existe a função 35h, que informa o conteúdo de um vetor.

7.7.2 INFORMA DATA ATUAL (FUNÇÃO 2AH)

<S> AL Dia da semana (0 = domingo, 1 = segunda etc.)

CX Ano (1980 a 2099)

DH Mês (1 a 12)

DL Dia (1 a 31)

Na versão 1, a data atual é mantida pelo próprio DOS; a partir da versão 2, é obtida do tratador de relógio.

7.7.3 ALTERA DATA ATUAL (FUNÇÃO 2BH)

<E> CX Ano (1980 a 2099)

DH Mês (1 a 12)

DL Dia (1 a 31)

<S> AL 00h se data válida

FFh data inválida

Na versão 1, a data atual é mantida pelo próprio DOS; a partir da versão 2, é obtida do tratador de relógio.

7.7.4 INFORMA HORA ATUAL (FUNÇÃO 2CH)

<S>	CH	Hora (0 a 23)
	CL	Minuto (0 a 59)
	DH	Segundo (0 a 59)
	DL	Centésimo de segundo (0 a 99)

Na versão 1, a hora atual é obtida diretamente do BIOS; a partir da versão 2, é obtida do tratador de relógio.

7.7.5 ALTERA HORA ATUAL (FUNÇÃO 2DH)

<E>	CH	Hora (0 a 23)
	CL	Minuto (0 a 59)
	DH	Segundo (0 a 59)
	DL	Centésimo de segundo (0 a 99)
<S>	AL	00h se hora válida
		FFh se hora inválida

Na versão 1, a hora atual é obtida diretamente do BIOS; a partir da versão 2, é obtida do tratador de relógio.

7.7.6 INFORMA VERSÃO DO DOS (FUNÇÃO 30H)

<S>	AL	00h se versão anterior a 2
		número da versão, se 2 ou posterior,
	AH	número da revisão
	BX	0
	CX	0

Exemplificando com as versões do PC-DOS:

Versão	AL	AH
1.00	00h	---
1.10	00h	---
2.00	02h	00h
2.10	02h	0Ah
3.00	03h	00h
3.10	03h	0Ah
3.20	03h	14h

7.7.7 INFORMA/ALTERA OPÇÃO DE INTERRUPÇÃO (FUNÇÃO 33H)

(Somente a partir da versão 2)

<E> AL 0 para informar

1 para alterar,

DL 0 para desativar

1 para ativar

<S> DL 0 opção inativa

1 opção ativa

O DOS sempre testa se ocorreu pedido de interrupção de programa ("Control Break) quando de determinadas chamadas às funções de caracter; quando a opção de interrupção está ativa, este teste é estendido às demais funções do DOS.

7.7.8 INFORMA ENDEREÇO DO INDICADOR DE CHAMADA EM CURSO (FUNÇÃO 34H)

(Somente a partir da versão 2, não documentada)

<S> ES:BX aponta para indicador de chamada em curso

O endereço retornado aponta para um byte que é incrementado na entrada do DOS (exceto EXEC e terminação) e decrementado na sua saída, o que permite testar (quando de uma interrupção) se existe uma chamada ao DOS em curso. No Capítulo 10 este assunto é discutido mais a fundo.

7.7.9 INFORMA VETOR DE INTERRUPÇÃO (FUNÇÃO 35H)

(Somente a partir da versão 2)

<E> AL vetor desejado

<S> ES:DX endereço no vetor

Esta função é a forma "limpa" de ler um vetor de interrupção. A função 25h permite alterar o conteúdo de um vetor.

7.7.10 INFORMA/ALTERA "SWITCH CHAR"/"AVAIL DEV" (FUNÇÃO 37H)

(Somente a partir da versão 2, não documentada)

<E> AL 0 Informa "Switch Car"

1 Altera "Switch Car"

DL novo valor

2 Informa "Avail Dev"

3 Altera "Avail Dev"

DL 00h desativa

FFh ativa

<S> Se Informa "Switch Char"

DL valor atual

Se Informa "Avail Dev"

DL 00h inativo

FFh ativo

Esta função deve ser considerada uma "curiosidade" do DOS. Foi introduzida de forma a permitir uma maior compatibilidade com o sistema XENIX, através do controle de dois parâmetros:

"Switch Char" - Caracter de Opção

É o caracter utilizado para indicar opções em uma linha de comando ("/" no DOS e "-" no XENIX). Teoricamente, os programas deveriam consultar esta chamada antes de analisar uma linha de comando (os utilitários do DOS, exceto BACKUP e RESTORE, fazem isto). Se for selecionado um caracter diferente de "/" o DOS passa a usar "/" para separar diretórios (ao invés de "\").

"Avail Dev"

Quando ativo, os dispositivos estão sempre disponíveis quando da busca em qualquer diretório (condição normal). Quando inativo, os dispositivos só podem ser acessados através do diretório fictício "\DEV".

Na versão 2, existiam comandos para controlar estes parâmetros no CONFIG.SYS ("SWITHCHAR" e "AVAILDEV"). Na versão 3, estes parâmetros "saíram de moda": os comandos não são aceitos no CONFIG.SYS e a função "Altera AvailDev" foi desimplementada.

7.7.11 INFORMAÇÕES DEPENDENTES DE PAÍS (FUNÇÃO 38H)

(Somente a partir da versão 2)

Versão 2:

<E> AL 0

DS:DX endereço de área com 32 bytes para resultado

<S> CY 0 se bem-sucedida

1 se ocorreu erro,

AX contém o código do erro

A área cujo endereço foi fornecido é preenchida com:

Deslocamento	tamanho	conteúdo
00h	1 word	formato de data e hora: 0 = americano (m/d/a h:m:s) 1 = europeu (d/m/a h:m:s) 2 = japonês (a:m:d h:m:s)
02h	2 bytes	símbolo da moeda seguido de zero (por exemplo '$',0)
04h	2 bytes	separador de milhares seguido de zero (por exemplo ',',0)

06h	2 bytes	separador de decimal seguido de zero (por exemplo '.',0)
08h	24 bytes	reservado

Versão 3, para obter informações:

<E> AL 00h para obter informação do país corrente

FFh para obter informações para país com código > 254,

BX código do país

código do país (1 a 254)

DS:DX endereço de área com 34 bytes para resultado

<S> CY 0 se bem-sucedida,

BX contém o código do país

1 se ocorreu erro,

AX contém o código do erro

A área cujo endereço foi fornecido é preenchida com:

desloc.	tamanho	conteúdo
00h	1 word	formato de data: 0 = americano (m d a) 1 = europeu (d m a) 2 = japonês (a m d)
02h	5 bytes	símbolo da moeda seguido de zero (por exemplo '$',0)
07h	2 bytes	separador de milhares seguido de zero (por exemplo ',',0)
09h	2 bytes	separador de decimal seguido de zero (por exemplo '.',0)
0Bh	2 bytes	separador de data seguido de zero (por exemplo '/',0)
0Dh	2 bytes	separador de hora seguido de zero (por exemplo ':',0)
0Fh	1 byte	uso do símbolo da moeda: 0 - antes do valor, sem espaços $1.2 1 - depois do valor, sem espaços 1.2$ 2 - antes do valor, com 1 espaço $ 1.2 3 - depois do valor, com 1 espaço 1.2 $ 4 - no lugar do separador de decimal 1$2
10h	1 byte	número de dígitos decimais em valores (2 -> $ 1.00, 3 -> $ 1.000 etc.)
11h	1 byte	formato de hora bit 0 0 formato 12 horas 1 formato 24 horas
12h	1 dword	endereço de rotina para converter para maiúscula códigos acima de 07Fh. Deve ser chamada com o caracter a converter em AL, retorna resultado também em AL.
16h	2 bytes	separador de ítens seguido de zero (por exemplo ',',0)
18h	10 bytes	reservado

Versão 3, para alterar país corrente:

<E> AL FFh para país com código > 254,
BX código do país
código do país (1 a 254)
DX FFFFh
<S> CY 0 se bem-sucedida
1 se ocorreu erro,
AX contém o código do erro

Os códigos de país são os seus códigos de DDI.

7.7.12 FORNECE CÓDIGO DE ERRO EXPANDIDO (FUNÇÃO 59H)

(Somente a partir da versão 3)

<E> BX 0

<S> AX Código de erro expandido
BH Classe do erro
BL Ação sugerida
CH Local do erro

Importante: Os registradores CL, DX, SI, DI, ES e DS são alterados.

Esta função pode ser usada no tratamento de erro crítico ou após uma chamada mal-sucedida (indicada por CY=1 ou por 0FFh em uso de FCB), para obter maiores informações sobre o erro.

Os códigos de erro já foram vistos no ítem 7.2; os demais resultados estão descritos nas tabelas a seguir:

Classe do erro	
01h	Recurso esgotado
02h	Situação temporária
03h	Problema de permissão
04h	Interno (erro de lógica no sistema)
05h	Falha de hardware
06h	Falha no sistema
07h	Erro no programa (chamada incorreta)
08h	Não encontrado
09h	Formato inválido
0Ah	Problema de compartilhamento: item protegido
0Bh	Erro de disco
0Ch	Já existe
0Dh	Outros tipos

Ação sugerida	
01h	Tente de novo, imediatamente
02h	Tente de novo, após algum tempo
03h	Se dados obtidos do usuário, pedir novamente
04h	Termine o programa, pode tentar finalizações (fechar arquivos, liberar memória etc.)
05h	Aborte o programa, sem tentar finalizações
06h	Ignore o erro
07h	Tente de novo, após intervenção do usuário

Local do erro	
01h	Não específico
02h	Dispositivo de bloco
03h	Rede
04h	Dispositivo de caracter
05h	Memória

8 FORMATO DE ARQUIVOS OBJETO

Uma característica do DOS é a inclusão de um ligador de módulos objeto (o utilitário LINK.EXE) no sistema padrão, ao contrário, por exemplo, do CP/M-80, no qual o ligador é um programa comercializado à parte. Isto incentiva a adoção de um padrão único de formato de arquivo objeto para os diversos compiladores existentes no mercado.

Na realidade, a MicroSoft não criou um formato novo, mas simplesmente seguiu um formato definido pelo INTEL (fabricante do 8088).

8.1 CONCEITOS

Vamos denominar de **módulo objeto** ao arquivo gerado por um programa tradutor (um compilador ou assembler). Um **programa** é obtido através da ligação de um ou mais módulos, o que é feito pelo LINK.

Alguns conceitos utilizados pelo LINK têm relação direta com diretivas assembler:

- **segmento** é uma região de memória com, no máximo, 64 Kbytes; a toda variável ou rotina corresponde um "offset" (de 16 bits) em relação a um segmento (o segmento no qual ela foi definida)
- **grupo** é um conjunto de segmentos que residem em uma mesma área de memória com, no máximo, 64 Kbytes. Um nome de grupo é utilizado para acessar, com um único valor em um registrador de segmento, dados e/ou rotinas declarados em segmentos diferentes.
- **classe** é um conjunto de segmentos que residirão em endereços contíguos, podendo ocupar mais de 64 Kbytes. O conceito de classe é utilizado para especificar o posicionamento relativo dos vários segmentos na memória.

Outros conceitos não têm correspondente na linguagem assembler:

- **Overlay** é um trecho de programa que pode compartilhar a memória com outros trechos, sendo trazidos para a memória conforme necessário. Por exemplo, suponhamos que um programa tenha três grandes partes: um programa principal, uma iniciação e rotinas de impressão e queremos rodar este programa em 128 Kbytes de memória, apesar de cada parte ter cerca de 60 Kbytes. Se admitirmos que a iniciação e impressão nunca são necessárias conjuntamente, podemos utilizar a mesma região de memória para as duas partes, criando assim dois overlays.
- **Frame** é a terminologia INTEL para uma área de memória de 64 Kbytes iniciada em um endereço múltiplo de 16 bytes (um parágrafo). Um frame é normalmente referenciado pelo seu endereço inicial dividido por 16 (por exemplo, o frame que começa no endereço 12340H é o frame 1234H).

8.2 FORMATO BÁSICO DE UM REGISTRO

Um arquivo objeto é composto por uma série de registros, que seguem um formato básico:

Deslocamento	tamanho	conteúdo
0	1 byte	tipo do registro
1	1 word	tamanho do conteúdo + *checksum* ("n")
2	n-1 bytes	depende do tipo do registro
n+2	1 byte	*checksum*: complemento de 2 da soma de todos os demais bytes do registro, ou seja, a soma de todos os bytes do registro deve resultar em 00H (somas ignorando "vai um")

A INTEL definiu 30 tipos de registro para arquivos objeto, de número 6Eh a AAh (só existem os de número par), porém a MicroSoft só utiliza atualmente 15 tipos:

Número	nome	descrição
7Ah	BLKDEF	início de bloco de programa
7Ch	BLKEND	fim de bloco de programa
80h	THEADR	início de módulo
88h	COMENT	comentário
8Ah	MODEND	fim de módulo
8Ch	EXTDEF	definição de símbolos externos
8Eh	TYPDEF	descreve um tipo de dados
90h	PUBDEF	definição de símbolos públicos
94h	LINNUM	associa números de linha do programa fonte ao código objeto
96h	LNAMES	nomes de segmentos, classes, overlays e grupos
98h	SEGDEF	descreve um segmento
9Ah	GRPDEF	descreve um grupo
9Ch	FIXUPP	indentifica posições a serem relocadas
A0h	LEDATA	dados não repetitivos
A2h	LIDATA	dados repetitivos

Os nomes contidos acima são os definidos pela INTEL (também utilizados pela MicroSoft).

8.3 DESCRIÇÃO DOS PRINCIPAIS REGISTROS

Neste ítem serão descritos os registros atualmente em uso pela MicroSoft, fornecendo-se a descrição do campo de conteúdo de cada um deles.

THEADR - tipo 80h

Este registro é sempre o primeiro de um arquivo objeto. O seu conteúdo é o nome do módulo, sendo de tamanho variável. O primeiro byte indica o número de caracteres no nome e é seguido pelo nome propriamente dito. Por exemplo, um módulo de nome "XIS" teria o seguinte registro THEADR:

```
80h                 tipo
05h 00h             tamanho (5 bytes)
03h                 tamanho do nome (3 bytes)
58h 49h 53h         nome ("XIS")
54h                 checksum
```

COMENT - tipo 88h

Este registro contém informações não relacionadas diretamente ao programa, como o nome da linguagem e do compilador utilizados.

O primeiro byte do conteúdo informa como este registro deve ser tratado por utilitários:

bit 7 1 ("No Purge")

indica que o registro não deve ser retirado

bit 6 1 ("No List")

indica que o registro não deve ser listado

Os demais bits devem conter 0.

O segundo byte indica o tipo de comentário (o tipo 0 contém a identificação do compilador e o tipo 81h a identificação da linguagem).

Os demais bytes do conteúdo contém o texto do comentário.

LNAMES - tipo 96H

Este registro contém uma lista de nomes de segmentos, classes, grupos e overlays, que serão referenciados em outros registros. A referência a estes nomes é feita numerando-os a partir de 1 (que corresponde ao primeiro nome do primeiro registro LNAMES do módulo).

Cada nome é colocado no registro precedido pelo seu tamanho. Vários compiladores geram um nome vazio, indicado por um tamanho 0. Por exemplo, um programa contendo os segmentos DADOS (classe DATA), STACK (classe STACK), COD1 e COD2 (classe CODE) e o nome vazio, teria um registro LNAMES com o seguinte conteudo:

```
96h                 tipo
22h 00h             tamanho (34 bytes)
00h                 nome #1 (vazio)
04h                 tamanho do nome (4 bytes)
43h 4Fh 44h 31h     nome #2 ("COD1")
04h                 tamanho do nome (4 bytes)
43h 4Fh 44h 32h     nome #3 ("COD2")
04h                 tamanho do nome (4 bytes)
43h 4Fh 44h 45h     nome #4 ("CODE")
05h                 tamanho do nome (5 bytes)
44h 41h 44h 4Fh 53h nome #5 ("DADOS")
04h                 tamanho do nome (4 bytes)
44h 41h 54h 41h     nome #6 ("DATA")
05h                 tamanho do nome (5 bytes)
53h 54h 41h 43h 4Bh nome #7 ("STACK")
09h                 checksum
```

SEGDEF - tipo 98h

Este registro descreve um segmento, tendo o seguinte formato:

1) um byte de controle:

bits 7 a 5	alinhamento do segmento
	000 Absoluto
	001 relocável, pode começar em qualquer endereço (alinhamento em byte)
	010 relocável, deve começar em endereço par (alinhamento em word)
	011 relocável, deve começar em endereço múltiplo de 16 (alinhamento em parágrafo)
	100 relocável, deve começar em endereço múltiplo de 256 (alinhamento em página)
bits 4 a 2	combinação do segmento
	000 segmento privativo, não será combinado com segmentos de outros módulos
	010 segmento público, será combinado com segmentos de outros módulos que tenham o mesmo nome e classe
	101 egmento stack
	110 segmento comum ("common"). Todos os segmentos de mesmo nome e classe são sobrepostos (cada módulo define o segmento a partir do seu offset 0)
bit 1	contém 1 se o segmento tiver exatamente 64 Kbytes
bit 0	contém 0

2) se o segmento for absoluto, seguem-se dois words contendo, respectivamente o "frame" e o "offset" inicial do segmento.
3) um word contendo o tamanho (em bytes) do segmento. Se o segmento contiver exatamente 64 Kbytes (que é o máximo possível) este campo contém 0 e o bit 1 do byte de controle é ativado.
4) três bytes contendo, respectivamente, os índices dos nomes do segmento, classe e overlay (em referência aos registros LNAMES).

Por exemplo, supondo que o segmento DADOS mencionado no item anterior tenha um tamanho de 2 bytes e seja alinhado em parágrafos e privativo, teríamos o seguinte registro SEGDEF:

```
98h                 tipo
07h 00h             tamanho (7 bytes)
60h                 controle (011 000 0 0)
02h 00h             tamanho (2 bytes)
05h                 nome do segmento (DADOS)
06h                 classe (DATA)
01h                 overlay (nome vazio)
F3h                 checksum
```

GRPDEF - tipo 9Ah

Este registro define um grupo. Ele possui no primeiro byte o índice do nome do grupo e nas posições restantes, os índices dos nomes dos segmentos que o compõem (2 bytes por cada nome, o primeiro contendo FFh e o segundo o índice).

Ainda seguindo o exemplo dado na descrição de LNAMES, teríamos o seguinte registro GRPDEF para um grupo de nome CODE que contivesse COD1 e COD2:

```
9Ah                 tipo
06h 00h             tamanho (6 bytes)
04h                 nome do grupo (CODE)
FFh 02h             nome de segmento (COD1)
FFh 03h             nome de segmento (COD2)
59h                 checksum
```

TYPDEF - tipo (8Eh)

Este registro define um tipo de variável, de forma a fornecer informações a depuradores, sendo referenciado por outros registros. No padrão MicroSoft é utilizado um registro TYPDEF vazio, exceto na definição de variáveis contidas "commom". Estas variáveis (utilizadas somente na linguagem FORTRAN) podem ser definidas com tamanho diferente em vários módulos. O LINK aloca uma área com o maior tamanho utilizado e todas as referências utilizam o mesmo endereço inicial.

O registro TYPDEF para uma variável "commom" é composto de dois bytes contendo 0 e um descritor ("leaf descriptor") que pode ser de dois tipos:

1) descritor de variável near

   É iniciado por um byte contendo 62h, seguido de um byte contendo o tipo de variável (77h - matriz, 79h - estrutura, 7Bh - escalar) e um byte contendo o tamanho em bits. Pode aindar existir mais um byte contendo o subtipo da variável, porém este byte é ignorado no padrão MicroSoft.

2) descritor de variável far

   É iniciado por um byte contendo 61h, seguido de um byte contendo o tipo de variável (o formato MicroSoft permite apenas 77h - matriz), um byte com o número de elementos e um byte com o índice do TYPDEF que define o tipo do elemento da matriz.

PUBDEF - tipo 90h

Este registro descreve os símbolos públicos do módulo, isto é, aqueles que poderão ser referenciados em outros módulos.

O primeiro byte do registro contém o índice do nome do grupo ao qual pertencem os símbolos, referenciando um registro LNAMES; o valor 0 indica que os símbolos não pertencem a nenhum grupo.

O segundo byte contém o indice do segmento, referenciando um registro SEGDEF (1 corresponde ao primeiro SEGDEF, 2 ao segundo, etc.).

Se os dois índices forem 0, segue-se um word contendo o frame no qual está o símbolo.

Em seguida, temos uma lista de definições de símbolos, com o seguinte formato:

1) O nome do símbolo, com o primeiro byte indicando tamanho e os seguintes os caracteres do nome
2) O offset do símbolo (um word) no segmento ou frame
3) O índice (um byte) do tipo de símbolo, referenciando um registro TYPDEF (1 corresponde ao primeiro TYPDEF, 2 ao segundo, etc).

Por exemplo, vamos supor que o módulo contenha dois símbolos públicos, XIS e YPS, no segmento (não pertencente a nenhum grupo) definido no segundo SEGDEF e que tenhamos apenas um TYPDEF vazio. O registro PUBDEF ficaria:

```
90h                 tipo
11h 00h             tamanho (17 bytes)
00h                 índice do grupo
02h                 índice do segmento
03h                 tamanho do nome do 1. símbolo
58h 49h 53h         "XIS"
01h 00h             offset do símbolo (0001h) no segmento
01h                 índice do tipo
03h                 tamanho do nome do 2. símbolo
59h 50h 53h         "YPS"
00h 01h             offset do símbolo (0100h) no segmento
01h                 índice do tipo
63h                 checksum
```

EXTDEF - tipo 8Ch

Este registro define os símbolos externos aos quais este módulo se refere, isto é, símbolos definidos em outros módulos e utilizados neste.

O registro contém uma lista de símbolos; cada um deles consiste em um byte com o tamanho do nome, os caracteres do nome, e um byte contendo o índice do tipo do símbolo (referenciando um registro TYPDEF).

Supondo que os símbolos XIS e YPS sejam externos e que tenhamos apenas um registro TYPDEF vazio, teríamos o seguinte registro EXTDEF:

```
8Ch                 tipo
0Bh 00h             tamanho (11 bytes)
03h                 tamanho do nome do 1. símbolo
58h 49h 53h         "XIS"
01h                 índice do tipo
03h                 tamanho do nome do 2. símbolo
59h 50h 53h         "YPS"
01h                 índice do tipo
71h                 checksum
```

LEDATA - tipo A0

Este registro contém dados (que na realidade podem ser dados, constantes, código, etc) a serem carregados para a memória. Os dados podem ainda vir a ser relocados e são enumerados, isto é, a cada byte da memória corresponde um byte no registro. Por isto a INTEL denomina o registro de "*Logical Enumerated Data*" (dados lógicos enumerados).

O formato do registro consiste em um byte contendo o índice do segmento (referenciando um SEGDEF), um word contendo o offset inicial no segmento e os dados propriamente ditos.

Exemplo:

```
A0h                 tipo
07h 00h             tamanho (7 bytes)
02h                 índice do segmento
00h 02h             offset (200h)
41h 42h 43h         dados ("ABC")
8Fh                 checksum
```

LIDATA - tipo A2

Este registro contém dados (que na realidade podem ser dados, constantes, código, etc) a serem carregados para a memória. Os dados podem ainda vir a ser relocados e são iterativos, isto é, bytes consecutivos com valores repetidos são compactados. Por isto a INTEL denomina o registro de "*Logical Iterated Data*" (dados lógicos iterativos).

O formato do registro é o seguinte:

1) Um byte contendo o índice do segmento (referenciando um SEGDEF)
2) Um word contendo o offset inicial no segmento
3) Um bloco de dados iterativo:
   a) número de repetições do bloco (word)
   b) número de sub-blocos; 0 indica que contém dados (word)
   c) os sub-blocos (no mesmo formato) ou dados (com o tamanho na primeira posição)

Conforme se percebe, é possível termos blocos dentro de blocos; o LINK da MicroSoft impõe a limitação de um máximo de 17 níveis e um máximo de 512 bytes no campo de bloco do registro. Alguns exemplos mostram como funciona este registro:

1) código assembler:

```
DB 100 DUP ('ABC')
```

código objeto:

```
A2h                 tipo
0Ch 00h             tamanho (12 bytes)
02h                 índice do segmento
00h 02h             offset (200h)
64h 00h             número de repetições (100)
```

```
00h                  00h número de sub-blocos
03h                  tamanho dos dados (3 bytes)
41h                  42h 43h dados ("ABC")
21h                  checksum
```

2) código assembler:

```
DB '(', 100 DUP ('ABC'), ')'
```

código objeto:

```
A2h                  tipo
1Ch 00h              tamanho (28 bytes)
02h                  índice do segmento
00h 02h              offset (200h)
01h 00h              número de repetições (1)
03h 00h              número de sub-blocos(3)
01h 00h              número de repetições (1)
00h 00h              número de sub-blocos (0)
01h                  tamanho dos dados (1 byte)
28h                  dados ("(")
64h                  00h número de repetições (100)
00h                  00h número de sub-blocos (0)
01h                  tamanho dos dados (3 bytes)
41h 42h 43h          dados ("ABC")
01h 00h              número de repetições (1)
00h 00h              número de sub-blocos (0)
01h                  tamanho dos dados (1 byte)
29h                  dados (")")
BAh                  checksum
```

FIXUP - tipo 9Ch

O registro de fixup descreve a relocação de dados. Ele contém a informação de que uma posição (denominada **local**) contém uma referência a um símbolo (denominado **alvo**) e que deve, portanto, ser corrigida conforme o valor do símbolo após a carga. Os registros de FIXUP estão sempre associados ao registro anterior de dados mais próximo.

Algumas relocações dependem apenas do posicionamento relativo dos segmentos, podendo ser efetuadas pelo LINK. Outras dependem do segmento em que o programa for carregado; neste caso, elas são anotadas no objeto executável (.EXE) e serão efetuadas quando da carga do programa pelo DOS.

Existem dois tipos básicos de registros FIXUP: o de **elo** (*thread*) e o de **relocação** propriamente dita.

Os registros de elo de FIXUP definem até quatro elos de frame e quatro elos de alvo, que são utilizados para simplificar os registros de relocação. Consistem em uma lista de itens com o seguinte formato:

1) um byte de controle

   bit 7 contém 0 para indicar elo de FIXUP

   bit 6 tipo de elo

   0 alvo

   1 frame

   bit 5 contém 0

   bits 4 a 2 método de identificação

   se alvo

   000 índice de SEGDEF e offset (A0)

   001 índice de GRPDEF e offset (A1)

   010 índice de EXTDEF e offset (A2)

   011 número de frame e offset (A3)

   100 índice de SEGDEF offset=0 (A4)

   101 índice de GRPDEF offset=0 (A5)

   110 ndice de EXTDEF offset=0 (A6)

   111 número de frame offset=0 (A7)

   se frame

   000 índice de SEGDEF (F0)

   001 índice de GRPDEF (F1)

   010 índice de EXTDEF (F2)

   011 número de frame (F3)

   100 frame do local (F4) *

   101 frame do alvo (F5)

   * obtido do LEDATA

   bits 1 a 0 número do elo (0 a 3)

2) um byte com índice ou número de frame, desde que o método não seja F4 ou F5

Os registros FIXUP de relocação consistem em uma lista de ítens com o seguinte formato:

1) 3 bytes de controle:

   1 bit contém 1 para indicar FIXUP de relocação

   1 bit modo de relocação

   0 "*self relative*", indica que a relocação deve afetar apenas um offset de 8 ou 16 bits

   1 "*segment relative*", indica que a relocação envolve um valor de segmento

1 bit contém 0

3 bits local a ser relocado

000 "*Lobyte*" (byte menos significativo de um word)

001 "*Offset*" (word menos significativo de ponteiro)

010 "*Base*" (word mais significativo de ponteiro)

011 "*Pointer*" (ponteiro)

100 "*Hibyte*" (byte mais significativo de um word)

10 bits deslocamento inicial do valor a ser relocado, em relação ao inicio da área de dados do registro LEDATA ou LIDATA anterior

1 bit forma de especificação do frame do local (F):

0 explícita

1 através de elo

3 bits se F = 0, especifica o método de definição do frame:

000 índice de SEGDEF (F0)

001 índice de GRPDEF (F1)

010 índice de EXTDEF (F2)

011 número de frame (F3)

100 frame do local (F4) (obtido do LEDATA)

101 frame do alvo (F5)

se F = 1, contém o número do elo de frame

1 bit forma de especificação do alvo (A):

0 explícita

1 através de elo

1 bit tipo de alvo (P):

1 primário (metodos A0 a A3), exige índice e offset

0 secundário (metodos A4 a A7), dispensa offset

2 bits se A = 0 método de especificação:

se P = 1

00 índice de SEGDEF e offset (A0)

01 índice de GRPDEF e offset (A1)

10 índice de EXTDEF e offset (A2)

11 número de frame e offset (A3)

se P = 0

00 índice de SEGDEF offset=0 (A4)

01 índice de GRPDEF offset=0 (A5)

10 índice de EXTDEF offset=0 (A6)

11 número de frame offset=0 (A7)

se A = 1 contém número do elo de alvo

2) se o frame do local for especificado explicitamente (F=0), segue-se um byte contendo o índice para o SEGDEF, GRPDEF ou EXTDEF (conforme o método) que identifica o frame do local

3) se o frame do alvo for especificado explicitamente (A=0), segue-se um byte contendo o índice para o SEGDEF, GRPDEF ou EXTDEF (conforme o método) que identifica o frame do alvo

4) finalmente, se o frame do alvo por especificado explicitamente de forma primária (A=0, P=1) segue-se um word contendo o offset do alvo.

Vejamos alguns exemplos. Vamos supor definidos:

SEGDEF # 1 - segmento de dados DADOS

EXTDEF # 5 - variável externa VAREXT

Um elo de FIXUP poderia ser:

```
9Ch                 tipo
04h 00h             tamanho (4 bytes)
43h                 controle:
                    0     elo
                    1     frame
                    0
                    000   definido por SEGDEF
                    11    elo número 3
01h                 índice do frame em SEGDEF
18h                 controle
                    0     elo
                    0     alvo
                    0
                    110   definido por EXTDEF
                    00    elo número 0
05h                 índice do alvo em EXTDEF
FFh                 checksum
```

Estes elos poderiam ser usados no seguinte FIXUP de relocação:

```
9Ch                 tipo
04h 00h             tamanho (4 bytes)
82h 09h BCh         controle:
                    1          relocação
                    0          "self-relative"
                    0
                    001        relocar "Offset"
                    0000001001 posição no LEDATA
                    1          frame do local definido por elo
                    011        elo número 3
                    1          alvo definido por elo
                    1          secundário
                    00         elo número 0
19h                 checksum
```

Este FIXUP deve ser interpretado da seguinte maneira: "o word localizado a partir do nono byte do LEDATA anterior, que será colocado no frame do segmento DADOS, deve ser relocado para conter o offset da variável VAREXT". Notar como o uso dos elos e referências a outros registros permitiu a compactação do código objeto.

BLKDEF - tipo 7Ah

Este registro indica o início de um bloco de programa (subrotina, "loop", etc). Suas informações se destinam principalmente a depuradores; no padrão INTEL existem registros adicionais que aparecem dentro de um bloco para indicar variáveis locais e outros dados.

O seu formato é o seguinte:

1) um byte com o índice do grupo em que o bloco está
2) um byte com o índice do segmento em que o bloco está
3) se os dois primeiros bytes forem 0, trata-se de um bloco em endereço absoluto; segue-se um word com o número do frame.
4) um byte com o tamanho do nome do bloco, seguido dos caracteres do nome
5) um word com o offset inicial do bloco
6) um word com o tamanho do bloco
7) um byte de controle: o bit 7 contém 1 se o bloco for uma subrotina; neste caso o bit 6 contém 1 se for uma subrotina far. Os demais bits contém 0.
8) se for uma subrotina, segue um word contendo o deslocamento do endereço de retorna na pilha
9) finalmente, temos um byte (opcional) que é um índice para o tipo da subrotina (referindo-se a um TYPDEF).

Por exemplo, vamos supor que exista uma rotina de nome WORDC, com 238 bytes, localizada a partir do offset 0001h do segmento CODE descrito no primeiro SEGDEF. Teríamos o seguinte BLKDEF:

```
7Ah                  tipo
11h 00h              tamanho (17 bytes)
00h                  índice do grupo (não pertence a grupo)
01h                  índice do segmento
05h                  tamanho do nome (5 bytes)
57h 4Fh 52h 44h 43h  nome do bloco ("WORDC")
01h 00h              offset inicial (0001h)
EEh 00h              tamanho do bloco (238 bytes)
60h                  controle: subrotina far
00h 00h              deslocamento (0) do endereço de retorno
00h                  índice do tipo (não tem tipo associado)
A1h                  checksum
```

BLKEND - tipo 7Ch

Este registro indica o fim de um bloco, não contendo nenhuma informação adicional:

```
7Ch                  tipo
01h 00h              tamanho (1 byte, o checksum)
83h                  checksum
```

LINNUM - tipo 94h

Este registro contém uma lista de associações de número de linha fonte a endereço de objeto. Esta informação, se colocada pelo compilador, é listada pelo LINK quando especificamos a opção /LINE.

O registro contém dois bytes no início, os quais contém os índices para o grupo e o segmento, nesta ordem, onde está o código. A seguir, temos a lista de associações, composta de pares de words. O primeiro word de cada par contém o número de uma linha do programa fonte e o segundo o offset inicial do código correspondente á linha.

Por exemplo, vamos supor as seguintes associações entre linhas fontes e código objeto:

Linha	Endereço inicial
5	0000h
6	0010h
10	0122h
11	0126h

Teríamos o seguinte registro LINNUM:

```
94h                 tipo
13h 00h             tamanho (19 bytes)
00h                 índice do grupo (não pertence a grupo)
01h                 índice do segmento
05h 00h             número de linha (5)
00h 00h             offset inicial do código (0000h)
06h 00h             número de linha (6)
10h 00h             offset inicial do código (0010h)
0Ah 00h             número de linha (10)
22h 01h             offset inicial do código (0122h)
0Bh 00h             número de linha (11)
26h 01h             offset inicial do código (0126h)
DEh                 checksum
```

MODEND - tipo 8Ah

Este registro assinala o fim de um módulo, tendo o seguinte formato:

1) um byte de contrôle, contendo um atributo nos dois bits mais significativos e 000001 nos restantes. O atributo pode ser:

   00 módulo não principal, endereço inicial não especificado

   01 módulo não principal, endereço inicial especificado

   10 módulo principal, endereço inicial não especificado

   11 módulo principal, endereço inicial especificado

2) se o atributo no byte de controle indicar presença do endereço inicial, este endereço é fornecido da mesma forma que um alvo de FIXUP: um byte de controle (que informa como ele é especificado), até dois bytes com o índices para SEGDEF, GRPDEF ou EXTDEF (se especificados explicitamente), e um word com o offset (se for especificado explicitamente de forma primária).

Exemplo:

```
8Ah                 tipo
07h 00h             tamanho (7 bytes)
C1h                 controle: módulo principal com endereço
                    inicial definido
00h                 controle:
                    0     explícito
                    000   frame definido por índice de SEGDEF
                    0     explícito
                    0     primário
                    00    definido por índice de SEGDEF e offset
04h                 índice de SEGDEF
04h                 índice de SEGDEF
00h 00h             offset
A6h                 checksum
```

9 INTERPRETADOR DE COMANDOS

Neste capítulo vamos estudar o COMMAND, que é o interpretador de comandos do DOS.

9.1 FUNÇÕES DO COMMAND

O COMMAND é a parte do DOS que realiza a interação com o usuário. Suas funções básicas são:

1) Obter um comando (do usuário ou de arquivo "batch")
2) Analisar o comando
3) Executá-lo
4) Fornecer as rotinas padrão de tratamento de Interrupção de Programa e de Erro Crítico

Algumas funções variam conforme a versão do DOS:

Versão 1 O COMMAND é o responsável pela carga de programas e gerenciamento da memória.

Versão 2: O COMMAND é o responsável pela carga de programas e manutenção do "environment mestre".

Versão 3: O COMMAND é o responsável pela manutenção do "environment mestre".

9.2 ESTRUTURA DO COMMAND

O COMMAND pode ser dividido nas seguintes partes:

1) Iniciação

   Executada somente na carga do sistema e descartada posteriormente

2) Residente

   Sempre presente na memória, contém o tratamento de Terminação, Interrupção e Erro Crítico, bem como recarga da parte transiente

3) Transiente

   Fica no final da memória, podendo ser superposta por outros programas. Trata da obtenção, análise e execução de comandos, particularmente os comandos internos

4) Carregador de Programas

   Somente presente na versão 2, implementa a função EXEC do DOS

A figura a seguir mostra o posicionamento das partes em disco e na memória:

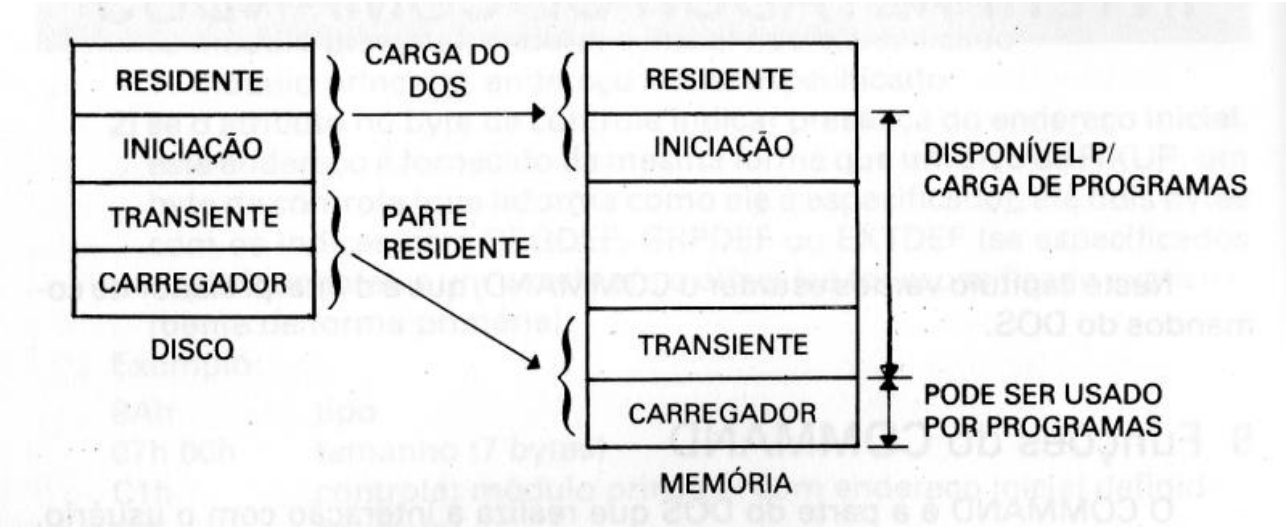


Figura 6: Estrutura do COMMAND

Nos próximos ítens vamos analisar cada uma destas partes.

9.3 PARTE RESIDENTE

Esta parte possui 4 rotinas principais:

1) Disparo e Terminação de Programa Externo

   A execução de um programa externo precisa ser comandada pela parte residente, pois a parte transiente pode ser sobreposta durante a carga. Neste ponto é também efetuada uma soma de controle da parte transiente, para verificar depois se é necessário recarrega-la. Na terminação de um programa externo, o COMMAND obtem e salva o seu código de retorno (para uso em ERRORLEVEL), recarrega a parte transiente e passa o controle a ela.

2) Interrupção de Programa

   Se está executando um arquivo "batch", pergunta se o interrompe. O programa em execução é terminado, através de retorno ao DOS com CF=1.

3) Tratamento de Erro Crítico

   Apresenta a mensagem apropriada e pergunta ao operador qual a ação a ser tomada. Esta ação é informada ao DOS.

4) Recarga da Parte Transiente

   Verifica se a parte transiente foi alterada, através de uma soma de controle. Em caso de alteração, recarrega a parte transiente de disco.

Na parte residente ficam também as variáveis da parte transiente que devem ser mantidas intactas em caso de recarga.

9.4 PARTE TRANSIENTE

Esta parte tem a seguinte lógica:

1) Apresenta o "prompt"
2) Se executando arquivo "batch"
   a) obtém proximo comando do arquivo
   b) se fim do arquivo, desliga o indicador de "batch"
3) Se não está executando arquivo "batch"
   a) lê comando do console
4) isola o nome do comando e os dois primeiros parâmetros
   a) se for só unidade: altera unidade padrão e volta ao passo 1
   b) se for comando interno: executa o comando e volta ao passo 1
   c) Procura comando.BAT; se achou, ativa indicador de "batch" e volta ao passo 1
   d) Procura comando.COM e comando.EXE (nesta ordem); se achou, passa para a rotina de disparo (na parte residente)
   e) Apresenta mensagem de comando inválido e retorna ao passo 1

9.5 INICIAÇÃO

Esta parte executa as seguintes tarefas:

1) Assume os vetores 22h, 23h e 24h.
2) Altera o tamanho da memória alocada ao COMMAND, liberando as partes referentes a iniciação, transiente e carregador.
3) Procura o arquivo AUTOEXEC.BAT, ativando indicador de "batch", se encontrá-lo.
4) Recarrega parte transiente e passa o controle a ela.

9.6 SUBSTITUINDO O COMMAND

O interpretador de comandos é um módulo estanque, com estrutura de um programa comum, de forma a permitir a sua substituição por programadores experientes.

A substituição do COMMAND permite a criação de novos comandos internos, alteração na interface homem/máquina, criação de sistemas "fechados", etc.

10 PROGRAMAS RESIDENTES

10.1 INTRODUÇÃO

Um tipo de programa muito popular atualmente é aquele que fica residente na memória, sendo ativado durante a execução de outro (através, por exemplo, de uma combinação especial de teclas).

Neste capítulo veremos os problemas envolvidos no projeto de programas deste tipo e como resolvê-los.

10.2 PROGRAMAS SIMPLES

Programas residentes simples são aqueles que não utilizam funções do DOS.

Estes programas simplesmente assumem alguns vetores de interrupção e terminam através de INT 27h ou da função 31h do INT 21h. Quando ativados (por uma interrupção), eles executam sua função e terminam por IRET.

Um exemplo é o utilitário GRAPHICS, que assume INT 5 (impressão da tela - "hardcopy") para permitir a impressão de telas gráficas em impressoras idem.

10.3 REENTRÂNCIA DO DOS

Quando um programa residente deseja acessar o DOS, ele encontra problemas, pois o DOS foi projetado para ter apenas um programa ativo de cada vez.

Conforme descrito no capítulo 7, a estrutura de pilhas internas do DOS permite, no máximo, três chamadas "simultâneamente" em curso:

- Uma normal às funções de caractere
- Uma às funções de caractere, durante o tratamento de erro crítico
- Uma às demais funções

Para que o DOS permitisse mais de uma função em curso, "simultaneamente", ele precisaria, além de usar uma pilha para cada chamada, ter todas as suas variáveis na pilha, controlar regiões críticas (para evitar a entrada no momento errado), etc.

É, portanto, necessário que se coloque um controle para impedir que o programa residente chame o DOS enquanto ele estiver processando uma chamada do programa "principal".

Realizar externamente este controle não é simples, pois uma chamada não retorna necessariamente por "IRET". As funções de terminação e EXEC, bem como aborto por erro crítico ou interrupção são exemplos destas dificuldades.

Felizmente, este controle já é feito pelo DOS, embora não documentado. A função 34h devolve em ES:BX o endereço de um byte que contém zero somente se não existir chamada em curso.

A **regra número 1** para programas residentes é, portanto: "Não chamar o DOS se o indicador de chamada em curso estiver ativo".

Esta regra é, entretanto, forte demais. Ela impede a ativação do programa residente durante a espera nas funções de caractere (por exemplo, quando o COMMAND colocou o "prompt" e está executando a função 0Ah para ler um comando do operador).

Para isto devemos utilizar outro recurso não documentado: INT 28h. Nas esperas em funções de caractere o DOS fica executando continuamente INT 28h. No tratamento desta interrupção podemos chamar funções do DOS que não as de caractere (1 a 0Ch).

Obtemos assim a **regra número 2**: "Assumir INT 28h e ativar o programa (se requisitado) independente do indicador de chamada em curso".

Existe (é claro) uma excessão. Não podemos chamar o DOS durante uma chamada a INT 28h se ela ocorrer durante um erro crítico causado por uma chamada a função de disco.

Este problema não tem solução simples! O famoso Sidekick obtém a informação de estar em erro crítico acessando uma variável interna (obviamente não documentada) do DOS. O endereço desta variável muda conforme a versão do DOS e não existe uma função (como a 34h) para obtê-lo. O mais simples é incluir no manual (e não no código) do programa a **regra número 3**: "Não ativar o programa enquanto estiver respondendo a um erro crítico".

Finalmente, cabe lembrar que o INT 28h não é milagroso; ele serve somente para acessos a dispositivos via funções de caractere. Ele não ocorre no acesso a disco via estas funções, nem no acesso a dispositivos via funções de arquivo. Por exemplo, o utilitário "type" utiliza a função 40h para mostrar um arquivo na tela (mandando até 64K - 1 bytes de uma só vez), o que impede a ativação de um programa residente enquanto o arquivo é mostrado.

Conforme visto nos capítulos 4 e 6, o DOS armazena na PSP informações sobre os handles utilizados pelo programa. É necessário, portanto, mudar a "PSP atual" durante a execução do programa residente. Para isto, utilizam-se as funções (não documentadas) "Informa PSP Atual" (51h) e "Altera PSP Atual" (50h).

10.4 REENTRÂNCIA DO BIOS

O BIOS não possui pilha própria, utilizando a pilha do programa que o chama.

Apesar disto, ele não é totalmente reentrante, pelos seguintes motivos:

1) Algumas funções do BIOS possuem variáveis em posição fixa de memória; uma reentrância pode ocorrer num instante em que as variáveis estão inconsistentes
2) Existem operações que devem forçosamente ser executas em sequência, sem acesso ao mesmo periférico entre elas. Um primeiro exemplo é no acesso ao disco, onde exitem um posicionamente da cabeça, uma programação do controlador de DMA e um operação de leitura/escrita. Um segundo exemplo é no acesso ao controlador de vídeo 6845, ao qual devemos fornecer o número de um registrador e um dado.

No acesso a disco, a solução normalmente adotada é interceptar as chamadas ao BIOS (assumindo INT 13h) e não ativar o programa durante a execução destas chamadas.

No acesso ao vídeo, a maioria dos programs ignora o problema, confiando em ser baixa a probabilidade de ocorrer. Os mais rigorosos podem adotar procedimento idêntico ao do acesso ao disco, assumindo INT 10h.

Ficamos assim com a **regra número 4**: "Não ativar o programa durante os acessos a disco (ou ao vídeo se formos rigoros) via BIOS".

10.5 SALVANDO E RESTAURANDO O CONTEXTO

Um programa residente está, de certa forma, "roubando" a máquina de um outro programa. Para que este "roubo" não seja percebido é necessário prestar atenção na "arrumação da casa" e colocar tudo de volta na hora de sair. É o que se chama de *preservar o contexto*.

O contexto inclui tudo aquilo que o programa principal pode estar usando e o programa residente pode alterar:

- registradores (inclusive flags)
- "variáveis" do DOS: DTA, unidade padrão, diretório corrente e opções de verificação e interrupção
- tela (modo, conteudo, posição e formato do cursor)

A **regra número 5** fica: "Preservar o contexto do programa principal"

Um outro cuidado relativo a contexto é o tratamento do ero crítico: não podemos permitir que um erro que ocorreu devido ao nosso programa residente seja tratado pelo programa principal. Daí a **regra número 6**: "Assumir o tratamento do erro crítico ao ativar o programa e devolvê-lo ao desativá-lo)".

10.6 QUANDO ATIVAR UM PROGRAMA RESIDENTE

Já vimos que existem várias ocasiões em que não podemos ativar um programa residente, devido ao DOS e o BIOS não estarem disponíveis. Vejamos agora quando tentar ativá-lo.

1) Na interrupção de teclado (INT 9)

   Normalmente o pedido de ativação do programa residente é detectado na interrupção de teclado. Podemos, portanto, tentar ativá-lo nesta hora. Temos, porém, dois problemas:

   a) se o DOS ou o BIOS não estiverem disponíveis, teremos que anotar a pedência de ativação e tentar novamente em outra ocasião. Aguardar uma nova interrupção do teclado não é conveniente.
   b) estamos no meio de uma interrupção de hardware, que pode ter ocorrido em quase qualquer instante. Teoricamente, os programs desabilitam as interrupções quando estão realizando operações críticas. Na realidade, o próprio BIOS do PC IBM deixa as interrupções habilitadas em alguns momentos críticos (como quando está fornecendo endereço e dado ao 6845).

   A melhor providência é, portanto, anotar a pendência de ativação e testá-la em outro momento.

2) Na interrupção de relógio (INT 1Ch)

   Neste caso, interceptamos INT 1Ch e ativamos o programa se o DOS e BIOS estiverem disponíveis e houver pedido pendente.

   As desvantagens deste método são que, novamente, estamos em uma interrupção de hardware e existem programas que assumem INT 1Ch sem passar a interrupção adiante.

Devemos tomar o cuidade de indicar para o controlador de interupções que novas interrupções de relógio podem ser aceitas, escrevendo 20h no "port" de endereço 20h.

3) Na chamada ao INT 21h

   Uma terceira maneira é ficar "de tocaia" na entrada (ou na saída) do DOS. Este método tem a vantagem de evitar automaticamente a reentrância do DOS, porém só permite interromper programas que o estejam chamando frequentemente.

Além disso, devemos tirar proveito do INT 28h, que nos permite ativar o programa de forma segura durante as esperas nas funções de caractere.

10.7 MANTENDO COMPATIBILIDADE COM OUTROS PROGRAMAS RESIDENTES

Uma característica importante é ser compatível com outros programas residentes, permitindo a ativação do nosso programa durante a execução dos outros e vice-versa.

Para isto deve-se tomar os seguintes cuidados:

1) Usar pilha própria, com espaço para o nosso uso e mais o de outros programas.
2) Passar adiante as interrupções interceptadas, isto é, chamar no nosso tratamento o anterior.
3) Chamar a rotina anterior de tratamento de INT 28h enquanto estiver esperando uma tecla, para dar oportunidade para outro programa residente entrar.
4) Carregar o nosso programa (que passa adiante as interrupções) antes dos outros (que podem não passá-las).

10.8 OBSERVAÇÕES FINAIS

O acesso ao vídeo e teclado deve ser feito via BIOS (ou acessando direto o hardware) pois não podemos acessar as funções de caractere durante o tratamento de INT 28h e queremos gerar INT 28h enquando aguardamos uma digitação.

Vimos que não existem soluções perfeitas, que garantam a ativação imediata e sem problemas. Isto é de se esperar, pois estamos fazendo algo não previsto pelo DOS. Apesar disto, as vantagens proporcionadas pelos programas residentes superam de longe os problemas que eles (às vezes) acarretam, o que explica o seu grande sucesso.

Ao tornar um programa residente devemos liberar todos os recursos que não serão mais necessários. Em particular é importante reduzir a memória alocada ao mínimo e fechar os handles que não serão mais utilizados.

Um exemplo de programa residente é mostrado no capítulo 11.

11 PROGRAMAS EXEMPLO

11.1 LISTADOR DE OBJETOS

Este programa lista, em ASCII e hexadecimal, um módulo objeto relocável. A listagem é organizada por registros, sendo destacados o tipo, tamanho e *checksum*; o byte de checksum é descontado do tamanho indicado no arquivo, de forma a corresponder ao conteúdo.

Para a geração do programa em um PC, o programa fonte deve ser digitado (por exemplo, no arquivo LOBJ.ASM), "assemblado", "linkado" e convertido para .COM:

```
MASM LOBJ;                (gera LOBJ.OBJ)
LINK LOBJ;                (gera LOBJ.EXE)
EXE2BIN LOBJ LOBJ.COM     (gera LOBJ.COM)
```

O programa recebe como parâmetro o nome do arquivo a listar, aceitando nomes completos (com especificação de diretório).

Deixo como sugestão para o leitro a alteração do programa para decodificar o conteúdo dos registros.

11.2 TRATADORES INSTALÁVEIS

Este programa contém dois tratadores instaláveis, que tratam os dispositivos "LST" e "TELA".

O dispositivo LST permite acessar a impressora normal (LPT1), saltando automaticamente o "picote" do formulário. Isto é feito pulando seis linhas adicionais a cada 60 linhas impressas; estes valores podem ser falcilmente alterados.

O dispositivo TELA permite acesso ao vídeo do PC de forma mais sofisticada que o tratador padrão, reconhecendo os seguintes códigos de controle:

BEL	(07H)	soa o alto-falante
BS	(08h)	recua o cursor uma coluna, parando no início da linha e não apagando caracteres
HT	(09H)	avança para a próxima coluna múltipla de 8
VT	(0BH)	altera a linha do cursor para o valor fornecido em seguida (0 a 23)
FF	(0CH)	limpa a tela e coloca o cursor no início
CR	(0DH)	coloca o cursor no início da linha
DLE	(10h)	altera a coluna do cursor para o valor fornecido em seguida (0 a 79)
CAN	(18H)	limpa da posição do cursor até o final da tela

Para a geração do programa em um PC, o programa fonte deve ser digitado (por exemplo, no arquivo TRAT.ASM), "assemblado", "linkado" e convertido para o formato de imagem de memória:

```
MASM TRAT;                (gera TRAT.OBJ)
LINK TRAT;                (gera TRAT.EXE)
EXE2BIN TRAT TRAT.SIS     (gera TRAT.SIS)
```

Para instalar os tratadores é necessário incluir a linha "DEVICE=TRAT.SIS" no arquivo CONFIG.SYS e recarregar o DOS. Após a recarga, os dispositivos LST e TELA estarão disponíveis, podendo ser usados como os demais. Por exemplo, para listar o programa podemos usar o comando "COPY TRAT.ASM LST".

O dispositivo TELA pode ser usado temporariamente como console, através do comando "CTTY TELA". Para instalá-lo na carga do sistema, altere o nome do tratador para COM e ative os bits correspondentes a console no atributo do cabeçalho. A escrita em TELA se processa de forma nitidamente mais lenta que no COM padrão; isto é devido ao não uso da "escrita rápida". Experimente acrescentar esta rotina, não esquecendo de assumir INT 29H e acertar o atributo no cabeçalho.

11.3 PROGRAMA RESIDENTE

Este programa ilustra as técnicas descritas no Capítulo 10, permitindo a obtenção da unidade padrão, do diretório corrente e da hora atual, durante a execução de outros programas.

Para a geração do programa em um PC, o programa fonte deve ser digitado (por exemplo, no arquivo RESID.ASM), "assemblado", "linkado" e convertido para .COM:

```
MASM RESID;                (gera RESID.OBJ)
LINK RESID;                (gera RESID.EXE)
EXE2BIN RESID RESID.COM    (gera RESID.COM)
```

O programa deve então ser instalado na memória, o que é feito chamando-o sem parâmetros. A partir deste instante, o pressionamento do "shift" esquerdo com o "shift" direito apertado, fará surgir no topo do vídeo uma linha reversa contendo a unidade, o diretório e a hora. A digitação de qualquer tecla restaura o vídeo, encerrando a ativação do programa residente.

Por simplificação, o programa não entra se o vídeo estiver em modo gráfico. Em caso de erro de disco, o alto-falante é soado e a operação repetida.

LISTADOR DE OBJETOS

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

                             ;
                             ; PROGRAMA EXEMPLO #1 - LISTADOR DE OBJETOS
                             ;
                             ; Uso:  LOBJ arquivo
                             ;

0000                         LOBJ    SEGMENT
                                     ASSUME CS:LOBJ, DS:LOBJ

0081                                 ORG     81H
0081                         CMD     LABEL   BYTE            ; linha de comando

0100                                 ORG     100H            ; inicio de programa .COM
0100                         INICIO:
0100  E9 01A1 R                      JMP     COMECO

                             ;
                             ; CONSTANTES
                             ;

= 008A                       MODEND  EQU     08AH            ; registro de fim de modulo

= 0021                       DOS     EQU     021H            ; int de sw p/ chamar DOS
= 0009                       MOSMEN  EQU     009H            ; mostra mensagem
= 3D00                       OPEN    EQU     03D00H          ; funcao abre para leitura
= 003F                       READ    EQU     3FH             ; funcao le arquivo
= 004C                       EXIT    EQU     04CH            ; funcao termina programa

= 000D                       CR      EQU     00DH            ; carriage return
= 000A                       LF      EQU     00AH            ; line feed

                             ;
                             ; VARIAVEIS
                             ;

0103  ??                     TIPO    DB      ?               ; tipo do registro
0104  ????                   TAM     DW      ?               ; tamanho da area de dados
0106  ??                     CHECK   DB      ?               ; "checksum"
0107  ????                   HANDLE  DW      ?               ; handle p/ ler objeto

                             ;
                             ; MENSAGENS
                             ;

0109  41 72 71 75 69 76      ERRO1   DB      'Arquivo nao encontrado !$'
      6F 20 6E 61 6F 20
      65 6E 63 6F 6E 74
      72 61 64 6F 20 21
      24
0122  41 72 71 75 69 76      ERRO2   DB      'Arquivo invalido !$'
      6F 20 69 6E 76 61
      6C 69 64 6F 20 21
      24
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

0135  0D 0A                   CABEC   DB      CR, LF
0137  54 69 70 6F 3A 20               DB      'Tipo: '
013D  58 58                   CAB_TIP DB      'XX'
013F  20 20 54 61 6D 61               DB      '  Tamanho: '
      6E 68 6F 3A 20
014A  58 58 58 58             CAB_TAM DB      'XXXX'
014E  20 20 43 68 65 63               DB      '  Check: '
      6B 3A 20
0157  58 58                   CAB_CHK DB      'XX'
0159  0D 0A                           DB      CR, LF
015B  24                              DB      '$'

015C     10 [                 SAIDA   DB      16 DUP ('XX ')
             58 58 20
                      ]

018C  3A                              DB      ':'
018D     10 [                 SAI_ASC DB      16 DUP ('A')
              41
                 ]

019D  3A                              DB      ':'
019E  0D 0A                           DB      CR, LF
01A0  24                              DB      '$'

01A1                          COMECO:
01A1  E8 01BB R                       CALL    ABRE            ; tenta abrir o arquivo
01A4  72 11                           JC      FIM             ; fim se erro
01A6                          PRINC:
01A6  E8 01E9 R                       CALL    LE_REG          ; le registro
01A9  72 0C                           JC      FIM             ; fim se erro
01AB  E8 022D R                       CALL    MOS_REG         ; mostra registro
01AE  80 3E 0103 R 8A                 CMP     TIPO,MODEND
01B3  75 F1                           JNE     PRINC           ; repete ate' fim do modulo
01B5  B0 00                           MOV     AL,0            ; fim normal
01B7                          FIM:
01B7  B4 4C                           MOV     AH,EXIT
01B9  CD 21                           INT     DOS             ; retorna ao DOS

                              ;
                              ; ABRE  abre o arquivo especificado na linha de comando
                              ;       retorna CY = 0 se conseguiu,
                              ;                CY = 1 se nao encontrou, AL = 01
                              ;

01BB                          ABRE    PROC    NEAR

01BB  FC                              CLD
01BC  BE 0081 R                       MOV     SI,OFFSET CMD
01BF                          ABRE1:
01BF  AC                              LODSB
01C0  3C 20                           CMP     AL,' '
01C2  74 FB                           JE      ABRE1           ; pula brancos iniciais
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------

01C4  8D 54 FF                        LEA     DX,[SI-1]     ; DX = inicio do nome
01C7                          ABRE2:
01C7  AC                              LODSB
01C8  3C 0D                           CMP     AL,CR
01CA  74 04                           JE      ABRE3
01CC  3C 20                           CMP     AL,' '
01CE  75 F7                           JNE     ABRE2         ; procura fim do nome
01D0                          ABRE3:
01D0  C6 44 FF 00                     MOV     BYTE PTR [SI-1],0 ; marca fim com 0
01D4  B8 3D00                         MOV     AX,OPEN
01D7  CD 21                           INT     DOS           ; tenta abrir
01D9  A3 0107 R                       MOV     HANDLE,AX
01DC  73 0A                           JNC     ABRE4

01DE  BA 0109 R                       MOV     DX,OFFSET ERRO1
01E1  B4 09                           MOV     AH,MOSMEN
01E3  CD 21                           INT     DOS           ; avisa que nao encontrou
01E5  B0 01                           MOV     AL,1
01E7  F9                              STC
01E8                          ABRE4:
01E8  C3                              RET

01E9                          ABRE    ENDP

                              ;
                              ; LE_REG le o proximo registro do objeto
                              ;        atualiza TIPO, TAM, CHECK e REG
                              ;        retorna CY = 0 se leu,
                              ;                CY = 1 se erro, AL = 02
                              ;

01E9                          LE_REG  PROC    NEAR

01E9  BA 0103 R                       MOV     DX,OFFSET TIPO
01EC  B9 0003                         MOV     CX,3
01EF  8B 1E 0107 R                    MOV     BX,HANDLE
01F3  B4 3F                           MOV     AH,READ
01F5  CD 21                           INT     DOS           ; tenta ler TIPO e TAM
01F7  72 29                           JC      LEREG1
01F9  3B C1                           CMP     AX,CX
01FB  75 25                           JNE     LEREG1

01FD  BA 02C7 R                       MOV     DX,OFFSET REG
0200  8B 0E 0104 R                    MOV     CX,TAM
0204  8B 1E 0107 R                    MOV     BX,HANDLE
0208  B4 3F                           MOV     AH,READ
020A  CD 21                           INT     DOS           ; tenta ler dados e check
020C  72 14                           JC      LEREG1
020E  3B C1                           CMP     AX,CX
0210  75 10                           JNE     LEREG1

0212  8B 1E 0104 R                    MOV     BX,TAM
0216  8A 87 02C6 R                    MOV     AL,REG [BX-1]
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------

021A  A2 0106 R                        MOV     CHECK,AL     ; retira CHECK
021D  FF 0E 0104 R                     DEC     TAM          ; desconta de TAM
0221  C3                              RET
0222                           LEREG1:
0222  BA 0122 R                       MOV     DX,OFFSET ERRO2
0225  B4 09                          MOV     AH,MOSMEN
0227  CD 21                          INT     DOS          ; avisa erro
0229  B0 02                          MOV     AL,2
022B  F9                              STC
022C                           LEREG2:
022C  C3                              RET

022D                           LE_REG  ENDP


                               ;
                               ; MOS_REG mostra o registro lido
                               ;

022D                           MOS_REG PROC    NEAR

022D  FC                              CLD
022E  BF 013D R                       MOV     DI,OFFSET CAB_TIP
0231  A0 0103 R                       MOV     AL,TIPO
0234  E8 02AE R                       CALL    HEXA
0237  BF 014A R                       MOV     DI,OFFSET CAB_TAM
023A  A0 0105 R                       MOV     AL,BYTE PTR (TAM+1)
023D  E8 02AE R                       CALL    HEXA
0240  A0 0104 R                       MOV     AL,BYTE PTR TAM
0243  E8 02AE R                       CALL    HEXA
0246  BF 0157 R                       MOV     DI,OFFSET CAB_CHK
0249  A0 0106 R                       MOV     AL,CHECK
024C  E8 02AE R                       CALL    HEXA
024F  BA 0135 R                       MOV     DX,OFFSET CABEC
0252  B4 09                          MOV     AH,MOSMEN
0254  CD 21                          INT     DOS          ; mostra cabecalho

0256  BE 02C7 R                       MOV     SI,OFFSET REG  ; SI - prox byte
0259  2B D2                          SUB     DX,DX          ; DX - # de bytes
025B                           MREG1:
025B  BF 015C R                       MOV     DI,OFFSET SAIDA
025E  B9 0030                         MOV     CX,16*3
0261  B0 20                          MOV     AL,20H
0263  F3/ AA                      REP STOSB
0265  BF 018D R                       MOV     DI,OFFSET SAI_ASC
0268  B9 0010                         MOV     CX,16
026B  F3/ AA                      REP STOSB
026D  BF 015C R                       MOV     DI,OFFSET SAIDA
0270  BB 018D R                       MOV     BX,OFFSET SAI_ASC ; limpa SAIDA
0273                           MREG2:
0273  3B 16 0104 R                    CMP     DX,TAM
0277  74 29                          JE      MREG5        ; fim se mostrou todos
0279  AC                              LODSB                ; pega proximo byte
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

027A  8B 07                            MOV     [BX],AL      ; mostrar em ASCII
027C  3C 24                            CMP     AL,'$'
027E  74 0B                            JE      MREG3        ;  "$",
0280  3C 20                            CMP     AL,20H
0282  72 04                            JB      MREG3        ;   abaixo de 20h e
0284  3C 7F                            CMP     AL,7FH
0286  72 03                            JB      MREG4        ;   acima de 7Fh
0288                           MREG3:
0288  C6 07 2E                         MOV     BYTE PTR [BX],'.' ; mostra "."
028B                           MREG4:
028B  43                               INC     BX
028C  E8 02AE R                        CALL    HEXA         ; coloca em hexa
028F  47                               INC     DI           ; espaco entre hexas
0290  42                               INC     DX
0291  F7 C2 000F                      TEST    DX,0FH
0295  75 DC                            JNZ     MREG2        ; linha cheia ?
0297  52                               PUSH    DX
0298  BA 015C R                        MOV     DX,OFFSET SAIDA
029B  B4 09                            MOV     AH,MOSMEN
029D  CD 21                            INT     DOS          ; sim: mostra
029F  5A                               POP     DX
02A0  EB B9                            JMP     MREG1
02A2                           MREG5:
02A2  0B D2                            OR      DX,DX        ; fim do registro:
02A4  74 07                            JZ      MREG6
02A6  BA 015C R                        MOV     DX,OFFSET SAIDA
02A9  B4 09                            MOV     AH,MOSMEN
02AB  CD 21                            INT     DOS          ; mostra linha parcial
02AD                           MREG6:
02AD  C3                               RET

02AE                           MOS_REG ENDP


                               ;
                               ; HEXA  coloca AL em hexadecimal na posicao apontada
                               ;       por DI; avanca DI
                               ;

02AE                           HEXA    PROC    NEAR

02AE  50                               PUSH    AX
02AF  D0 E8                            SHR     AL,1
02B1  D0 E8                            SHR     AL,1
02B3  D0 E8                            SHR     AL,1
02B5  D0 E8                            SHR     AL,1
02B7  E8 02BB R                        CALL    NIBBLE       ; primeiro digito hexa
02BA  58                               POP     AX
02BB                           NIBBLE:
02BB  24 0F                            AND     AL,0FH       ; isola 4 bits
02BD  04 30                            ADD     AL,30H       ; converte se digito
02BF  3C 39                            CMP     AL,39H
02C1  76 02                            JBE     NIB1
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------

02C3  04 07                          ADD     AL,7        ; acerto para letras
02C5                         NIBi:
02C5  AA                             STOSB               ; coloca na saida
02C6  C3                             RET

02C7                         HEXA    ENDP


                             ; O registro e' lido na area apos o programa

02C7                         REG     LABEL   BYTE

02C7                         LOBJ    ENDS

                                     END     INICIO
```

TRATADORES INSTALÁVEIS

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

                             ;
                             ; PROGRAMA EXEMPLO #2 - TRATADORES INSTALAVEIS
                             ;
                             ; Este programa contem dois tratadores:
                             ; - LST  impressora com formatacao
                             ; - TELA console com funcoes especiais
                             ;

0000                         TRAT    SEGMENT
0000                                 ORG     0H
                                     ASSUME  CS:TRAT, DS:TRAT, SS:TRAT

                             ; Cabecalhos dos Tratadores

0000  0012 R                 CAB_LST DW      OFFSET CAB_TEL          ; proximo cabecalho
0002  ????                           DW      ?                       ; preenchido pelo DOS
0004  8000                           DW      1000000000000000B       ; dispo de caractere
0006  01A4 R                         DW      OFFSET ESTRAT           ; rot. de estrategia
0008  01AF R                         DW      OFFSET INT_LST          ; rot. de interrupcao
000A  4C 53 54 20 20 20              DB      'LST     '              ; nome
      20 20
0012  FF FF FF FF            CAB_TEL DD      -1                      ; proximo cabecalho
0016  8000                           DW      1000000000000000B       ; dispo de caractere
0018  01A4 R                         DW      OFFSET ESTRAT           ; rot. de estrategia
001A  01B5 R                         DW      OFFSET INT_TEL          ; rot. de interrupcao
001C  54 45 4C 41 20 20              DB      'TELA    '              ; nome
      20 20

                             ; Constantes

                                                                     ; Chamadas ao BIOS
= 0010                       VIDEO   EQU     10H                     ;   Video
= 0016                       TECLADO EQU     16H                     ;   Teclado
= 0017                       PRINTER EQU     17H                     ;   Impressora

                                                                     ; Status
= 0100                       ST_LIV  EQU     0100H                   ; Fim normal, livre
= 0300                       ST_OCUP EQU     0300H                   ; Fim normal, ocupado

                             CAB_REQ STRUC
0000  ??                     TAM     DB      ?                       ; Tamanho
0001  ??                     UNIT    DB      ?                       ; Unidade
0002  ??                     CMD     DB      ?                       ; Comando
0003  ????                   STATUS  DW      ?                       ; Resultado
0005     08 [                RESERV  DB      8 DUP (?)               ; Reservado ao DOS
            ??
               ]
000D                         CAB_REQ ENDS

                             RQ_INIT STRUC                           ; Req. de Iniciacao
0000     0D [                        DB      13 DUP (?)              ; Cabecalho
            ??
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

000D  ??                     N_UNIT  DB      ?                       ; # de unidades
000E  ????                   FIM_OFF DW      ?                       ; Fim do tratador
0010  ????                   FIM_SEG DW      ?
0012  ????????               TAB_BPB DD      ?                       ; Tabela de BPBs
0016                         RQ_INIT ENDS

                             RQ_IO   STRUC                           ; Req. de E/S
0000     0D [                        DB      13 DUP (?)              ; Cabecalho
            ??
               ]

000D  ??                     MEDIA   DB      ?                       ; "Media Byte"
000E  ????????               DTA     DD      ?                       ; Area de transf.
0012  ????                   CNT     DW      ?                       ; # de bytes/setores
0014  ????                   SET_INI DW      ?                       ; Setor inicial
0016                         RQ_IO   ENDS

                             RQ_INP  STRUC                           ; Requisicao de
                                                                     ;   Leitura sem espera
0000     0D [                        DB      13 DUP (?)              ; Cabecalho
            ??
               ]

000D  ??                     CAR     DB      ?                       ; Proximo caractere a
                                                                     ;  ser lido
000E                         RQ_INP  ENDS


                             ; Variaveis


0024                         REQUIS  LABEL   DWORD                   ; end. da requisicao
0024  ????                   REQ_OFF DW      ?
0026  ????                   REQ_SEG DW      ?

0028  ????                   PIL_SEG DW      ?                       ; end. da pilha do DOS
002A  ????                   PIL_OFF DW      ?

002C  ??                     LIN     DB      ?                       ; contador de linhas
                                                                     ; (LST)

002D  ????                   ESTADO  DW      ?                       ; estado da rotina de
                                                                     ;   escrita (TELA)

002F  ??                     TEC_ANT DB      ?                       ; tecla lida e ainda
                                                                     ;  nao informada
                                                                     ;  (0 = nao tem)

0030     80 [                        DW      128 DUP (?)             ; area para pilha
            ????
               ]

0130                         PILHA   LABEL   WORD
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

                              ; Tabelas de enderecos das rotinas que
                              ; executam os comandos

0130  021A R                  ROT_LST DW      OFFSET INI_LST          ; Iniciacao
0132  0216 R                          DW      OFFSET INVAL            ; Teste de troca
0134  0216 R                          DW      OFFSET INVAL            ; Constroi bpb
0136  0216 R                          DW      OFFSET INVAL            ; IOCTL leitura
0138  022D R                          DW      OFFSET LE_LST           ; Leitura
013A  0212 R                          DW      OFFSET OCUPADO          ; Leitura sem espera
013C  020E R                          DW      OFFSET LIVRE            ; Status de leitura
013E  020A R                          DW      OFFSET INOCUA           ; Flush de leitura
0140  0237 R                          DW      OFFSET ESC_LST          ; Escrita
0142  0237 R                          DW      OFFSET ESC_LST          ; Escrita com verif.
0144  020E R                          DW      OFFSET LIVRE            ; Status de escrita
0146  020A R                          DW      OFFSET INOCUA           ; Flush de escrita
0148  0216 R                          DW      OFFSET INVAL            ; IOCTL escrita

014A  0298 R                  ROT_TEL DW      OFFSET INI_TEL          ; Iniciacao
014C  0216 R                          DW      OFFSET INVAL            ; Teste de troca
014E  0216 R                          DW      OFFSET INVAL            ; Constroi bpb
0150  0216 R                          DW      OFFSET INVAL            ; IOCTL leitura
0152  03AA R                          DW      OFFSET LE_TEL           ; Leitura
0154  0391 R                          DW      OFFSET LE2_TEL          ; Leitura sem espera
0156  03D2 R                          DW      OFFSET STA_TEL          ; Status de leitura
0158  03E7 R                          DW      OFFSET FLS_TEL          ; Flush de leitura
015A  02B1 R                          DW      OFFSET ESC_TEL          ; Escrita
015C  02B1 R                          DW      OFFSET ESC_TEL          ; Escrita com verif.
015E  020E R                          DW      OFFSET LIVRE            ; Status de escrita
0160  020A R                          DW      OFFSET INOCUA           ; Flush de escrita
0162  0216 R                          DW      OFFSET INVAL            ; IOCTL escrita

                              ; Tabela de rotinas de tratamento de escrita de
                              ; codigos de controle em TELA

0164  02DD R                  TAB_FUN DW      OFFSET NADA     ; NUL
0166  02DD R                          DW      OFFSET NADA     ; SOH
0168  02DD R                          DW      OFFSET NADA     ; STX
016A  02DD R                          DW      OFFSET NADA     ; ETX
016C  02DD R                          DW      OFFSET NADA     ; EOT
016E  02DD R                          DW      OFFSET NADA     ; ENQ
0170  02DD R                          DW      OFFSET NADA     ; ACK
0172  02D7 R                          DW      OFFSET ETBIOS   ; BEL  - soa altofalante
0174  02D7 R                          DW      OFFSET ETBIOS   ; BS   - recua cursor
0176  0340 R                          DW      OFFSET TABULA   ; HT   - tabulacao
0178  02D7 R                          DW      OFFSET ETBIOS   ; LF   - muda de linha
017A  02E5 R                          DW      OFFSET POS_VER  ; VT   - posic. vertical
017C  02EC R                          DW      OFFSET CLS      ; FF   - limpa tela
017E  02D7 R                          DW      OFFSET ETBIOS   ; CR   - volta a coluna 0
0180  02DD R                          DW      OFFSET NADA     ; SO
0182  02DD R                          DW      OFFSET NADA     ; SI
0184  02DE R                          DW      OFFSET POS_HOR  ; DLE  - posic. horizontal
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

0186  02DD R                          DW      OFFSET NADA     ; DC1
0188  02DD R                          DW      OFFSET NADA     ; DC2
018A  02DD R                          DW      OFFSET NADA     ; DC3
018C  02DD R                          DW      OFFSET NADA     ; DC4
018E  02DD R                          DW      OFFSET NADA     ; NAK
0190  02DD R                          DW      OFFSET NADA     ; SYN
0192  02DD R                          DW      OFFSET NADA     ; ETB
0194  030A R                          DW      OFFSET CLR_EOS  ; CAN  - limpa ate fim da tela
0196  02DD R                          DW      OFFSET NADA     ; EM
0198  02DD R                          DW      OFFSET NADA     ; SUB
019A  02DD R                          DW      OFFSET NADA     ; ESC
019C  02DD R                          DW      OFFSET NADA     ; FS
019E  02DD R                          DW      OFFSET NADA     ; GS
01A0  02DD R                          DW      OFFSET NADA     ; RS
01A2  02DD R                          DW      OFFSET NADA     ; US


                                      ; Rotina de estrategia

01A4                                  ESTRAT  PROC    FAR

01A4  2E: 8C 06 0026 R                        MOV     CS:REQ_SEG,ES           ; salva endereco
01A9  2E: 89 1E 0024 R                        MOV     CS:REQ_OFF,BX           ;   da requisicao
01AE  CB                                      RET

01AF                                  ESTRAT  ENDP

                                      ; Rotinas de interrupcao

01AF                                  INTER   PROC    FAR

01AF                                  INT_LST:
01AF  57                                      PUSH    DI
01B0  BF 0130 R                               MOV     DI,OFFSET ROT_LST
01B3  EB 04                                   JMP     SHORT DISTRIB

01B5                                  INT_TEL:
01B5  57                                      PUSH    DI
01B6  BF 014A R                               MOV     DI,OFFSET ROT_TEL
                                      ;       JMP     SHORT DISTRIB


                                      ; Rotina de distribuicao de controle as funcoes
                                      ; (E) DI original salvo na pilha
                                      ;     endereco da tabela de rotinas em DI
                                      ; As rotinas recebem ES:BX apontando para a requisicao
                                      ;           devem devolver status em AX

01B9                                  DISTRIB:
01B9  56                                      PUSH    SI
01BA  55                                      PUSH    BP
01BB  52                                      PUSH    DX
01BC  51                                      PUSH    CX
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

01BD  53                          PUSH    BX
01BE  50                          PUSH    AX
01BF  06                          PUSH    ES
01C0  1E                          PUSH    DS                      ; salva registradores
01C1  8C C8                       MOV     AX,CS
01C3  8E D8                       MOV     DS,AX
01C5  8C 16 0028 R                MOV     PIL_SEG,SS
01C9  89 26 002A R                MOV     PIL_OFF,SP              ; salva pilha do DOS
01CD  FA                          CLI
01CE  8E D0                       MOV     SS,AX
01D0  BC 0130 R                   MOV     SP,OFFSET PILHA         ; usa pilha local
01D3  FB                          STI
01D4  C4 1E 0024 R                LES     BX,REQUIS
01D8  26: 8A 47 02                MOV     AL,ES:[BX].CMD
01DC  2A E4                       SUB     AH,AH
01DE  3D 000C                     CMP     AX,12
01E1  76 05                       JBE     INT_1                   ; comando valido ?
01E3  B8 8103                     MOV     AX,8103H                ; nao: status de erro
01E6  EB 06                       JMP     SHORT INT_2
01E8                        INT_1:                                ; sim:
01E8  03 F8                       ADD     DI,AX
01EA  03 F8                       ADD     DI,AX
01EC  FF 15                       CALL    [DI]                    ;   chama funcao
01EE                        INT_2:
01EE  C4 1E 0024 R                LES     BX,REQUIS
01F2  26: 89 47 03                MOV     ES:[BX].STATUS,AX       ; atualiza status
01F6  FA                          CLI
01F7  8E 16 0028 R                MOV     SS,PIL_SEG
01FB  8B 26 002A R                MOV     SP,PIL_OFF              ; volta a pilha do DOS
01FF  FB                          STI
0200  1F                          POP     DS
0201  07                          POP     ES
0202  58                          POP     AX
0203  5B                          POP     BX
0204  59                          POP     CX
0205  5A                          POP     DX
0206  5D                          POP     BP
0207  5E                          POP     SI
0208  5F                          POP     DI                      ; restaura regs
0209  CB                          RET

020A                        INTER   ENDP


                            ; INOCUA  - Retorna sem fazer nada
                            ; LIVRE   - Indica dispositivo livre
                            ; OCUPADO - Indica dispositivo ocupado
                            ; INVAL   - Indica funcao invalida

020A                        INOCUA  PROC    NEAR
020A  B8 0100                       MOV     AX,ST_LIV
020D  C3                            RET
020E                        INOCUA  ENDP
```

```
--- Objeto ----------------  --- Programa Fonte ----------------------------------------

020E                         LIVRE   PROC    NEAR
020E  B8 0100                        MOV     AX,ST_LIV
0211  C3                             RET
0212                         LIVRE   ENDP

0212                         OCUPADO PROC    NEAR
0212  B8 0300                        MOV     AX,ST_OCUP
0215  C3                             RET
0216                         OCUPADO ENDP

0216                         INVAL   PROC    NEAR
0216  B8 8103                        MOV     AX,8103H
0219  C3                             RET
021A                         INVAL   ENDP


                             ; INI_LST - inicia tratador de LST

021A                         INI_LST PROC    NEAR
021A  C6 06 002C R 00                MOV     LIN,0
021F  26: 8C 4F 10                   MOV     ES:[BX].FIM_SEG,CS
0223  26: C7 47 0E 03FC R            MOV     ES:[BX].FIM_OFF,OFFSET FIM_TRT
0229  B8 0100                        MOV     AX,ST_LIV
022C  C3                             RET
022D                         INI_LST ENDP


                             ; LE_LST - leitura em LST

022D                         LE_LST  PROC    NEAR
022D  26: C7 47 12 0000              MOV     ES:[BX].CNT,0           ; indica nada lido
0233  B8 0100                        MOV     AX,ST_LIV
0236  C3                             RET
0237                         LE_LST  ENDP    NEAR

                             ; ESC_LST - escrita em LST

0237                         ESC_LST PROC    NEAR
0237  26: 8B 4F 12                   MOV     CX,ES:[BX].CNT      ; CX = # de bytes por
                                                                 ;      escrever
023B  26: C4 5F 0E                   LES     BX,ES:[BX].DTA      ; ES:BX = dados
023F                         ELST_1:
023F  E3 46                          JCXZ    ELST_7              ; enquanto nao acabar
0241  26: 8A 07                      MOV     AL,ES:[BX]          ;   pega caractere
0244  43                             INC     BX
0245  3C 0C                          CMP     AL,0CH
0247  75 07                          JNE     ELST_2              ;   Form Feed ?
0249  C6 06 002C R 00                MOV     LIN,0               ;   Sim: vai p/
024E  EB 08                          JMP     SHORT ELST_3        ;     primeira linha
0250                         ELST_2:
0250  3C 0A                          CMP     AL,0AH
0252  75 04                          JNE     ELST_3              ;   Line Feed?
0254  FE 06 002C R                   INC     LIN                 ;   Sim: muda de linha
```

```
--- Objeto ----------------  --- Programa Fonte ----------------------------------

0258                        ELST_3:
0258  E8 028B R                     CALL    IMP                 ;   imprime o caractere
025B  72 1E                         JC      ELST_6              ;   fim se erro
025D  49                            DEC     CX
025E  80 3E 002C R 3C               CMP     LIN,60
0263  75 DA                         JNE     ELST_1              ;   fim da pagina?
0265  C6 06 002C R 00               MOV     LIN,0               ;   Sim: primeira linha
026A  51                            PUSH    CX
026B  B9 0006                       MOV     CX,6                ;       pula 6 linhas
026E                        ELST_4:
026E  B0 0A                         MOV     AL,0AH
0270  E8 028B R                     CALL    IMP
0273  72 05                         JC      ELST_5
0275  E2 F7                         LOOP    ELST_4
0277  59                            POP     CX
0278  EB C5                         JMP     ELST_1
027A                        ELST_5:
027A  59                            POP     CX
027B                        ELST_6:                             ; Erro:
027B  C4 1E 0024 R                  LES     BX,REQUIS
027F  26: 29 4F 12                  SUB     ES:[BX].CNT,CX      ;   Atualiza contagem
0283  B8 810A                       MOV     AX,810AH            ;   Status de erro
0286  C3                            RET
0287                        ELST_7:                             ; Fim normal:
0287  B8 0100                       MOV     AX,ST_LIV           ;   Status normal
028A  C3                            RET

028B                        ESC_LST ENDP

                            ; IMP - escreve (AL) em LPT1:
                            ; Retorna com CY=1 se erro

028B                        IMP     PROC    NEAR
028B  2B D2                         SUB     DX,DX
028D  8A E2                         MOV     AH,DL
028F  CD 17                         INT     PRINTER             ; Escreve via BIOS
0291  80 E4 29                      AND     AH,29H
0294  74 01                         JZ      IMP_1               ; Testa erro
0296  F9                            STC
0297                        IMP_1:
0297  C3                            RET
0298                        IMP     ENDP

                            ; INI_TEL - inicia tratador de TELA

0298                        INI_TEL PROC    NEAR
0298  C7 06 002D R 02C9 R          MOV     ESTADO,OFFSET EST_NOR
029E  C6 06 002F R 00               MOV     TEC_ANT,0
02A3  26: 8C 4F 10                  MOV     ES:[BX].FIM_SEG,CS
02A7  26: C7 47 0E 03FC R           MOV     ES:[BX].FIM_OFF,OFFSET FIM_TRT
02AD  B8 0100                       MOV     AX,ST_LIV
02B0  C3                            RET
02B1                        INI_TEL ENDP
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

                             ; ESC_TEL - escrita em TELA

02B1                         ESC_TEL PROC    NEAR
02B1  26: 8B 4F 12                   MOV     CX,ES:[BX].CNT          ; CX = # de bytes por
                                                                     ;      escrever
02B5  26: C4 77 0E                   LES     SI,ES:[BX].DTA          ; ES:SI = dados
02B9  E3 0A                          JCXZ    ETEL_2
02BB                         ETEL_1:                                 ; enquanto nao acabar
02BB  26: 8A 04                      MOV     AL,ES:[SI]              ;   pega caractere
02BE  46                             INC     SI
02BF  FF 16 002D R                   CALL    ESTADO                  ;   trata
02C3  E2 F6                          LOOP    ETEL_1
02C5                         ETEL_2:
02C5  B8 0100                        MOV     AX,ST_LIV               ;   Status normal
02C8  C3                             RET
02C9                         ESC_TEL ENDP


                             ; EST_NOR - Estado normal
                             ; ETBIOS  - Escreve (AL) na tela (via BIOS)
                             ; NADA    - Faz nada
                             ; POS_HOR - Trata HT
                             ; POS_VER - Trata VT
                             ; CLS     - Limpa tela
                             ; CLR_EOL - Limpa ate final da tela
                             ; TABULA  - Avanca para proxima tabulacao

02C9                         EST_NOR PROC    NEAR

02C9  3C 20                          CMP     AL,20H
02CB  73 0A                          JAE     ETBIOS                  ;   controle ?
02CD  2A E4                          SUB     AH,AH                   ;   sim:
02CF  8B D8                          MOV     BX,AX
02D1  D1 E3                          SHL     BX,1
02D3  FF A7 0164 R                   JMP     TAB_FUN [BX]            ;      chama funcao
02D7                         ETBIOS:                                 ;   nao:
02D7  B3 03                          MOV     BL,3
02D9  B4 0E                          MOV     AH,14
02DB  CD 10                          INT     VIDEO                   ;      escreve via BIOS
02DD                         NADA:
02DD  C3                             RET


02DE                         POS_HOR:
02DE  C7 06 002D R 0378 R            MOV     ESTADO,OFFSET EST_HT
02E4  C3                             RET


02E5                         POS_VER:
02E5  C7 06 002D R 035F R            MOV     ESTADO,OFFSET EST_VT
02EB  C3                             RET
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------

02EC                        CLS:
02EC  51                            PUSH    CX
02ED  B4 0F                         MOV     AH,15
02EF  CD 10                         INT     VIDEO           ; Estado do video
02F1  53                            PUSH    BX              ; BH - pagina ativa
02F2  FE CC                         DEC     AH
02F4  8A D4                         MOV     DL,AH           ; DL - ultima coluna
02F6  B6 18                         MOV     DH,24           ; DH - ultima linha
02F8  2B C9                         SUB     CX,CX           ; CX - primeira linha
                                                            ;      e coluna
02FA  B7 07                         MOV     BH,07H          ; b & p
02FC  B8 0600                       MOV     AX,0600H        ; limpa regiao
02FF  CD 10                         INT     VIDEO
0301  5B                            POP     BX              ; BH - pagina ativa
0302  2B D2                         SUB     DX,DX           ; DX - inicio da tela
0304  B4 02                         MOV     AH,2            ; posiciona cursor
0306  CD 10                         INT     VIDEO
0308  59                           POP     CX
0309  C3                           RET

030A                        CLR_EOS:
030A  51                            PUSH    CX
030B  B4 0F                         MOV     AH,15
030D  CD 10                         INT     VIDEO           ; Estado do video
030F  8A DC                         MOV     BL,AH           ; BL - # de colunas
0311  53                            PUSH    BX              ; BH - pagina ativa
0312  B4 03                         MOV     AH,3
0314  CD 10                         INT     VIDEO           ; le posicao do curso
0316  5B                            POP     BX
0317  53                            PUSH    BX
0318  52                            PUSH    DX
0319  8A CB                         MOV     CL,BL
031B  2A CA                         SUB     CL,DL
031D  2A ED                         SUB     CH,CH           ; CX - # de colunas ;
031F  B3 07                         MOV     BL,07H          ;      limpar
0321  B8 0920                       MOV     AX,0920H
0324  CD 10                         INT     VIDEO           ; limpa ate o final
0326  59                           POP     CX              ;   da linha
0327  5B                           POP     BX
0328  80 FD 18                      CMP     CH,24
032B  74 11                         JE      CLR_1           ; pronto se na ultim
                                                            ; linha
032D  8A D3                         MOV     DL,BL
032F  FE CA                         DEC     DL              ; DL - ultima coluna
0331  B6 18                         MOV     DH,24           ; DH - ultima linha
0333  2A C9                         SUB     CL,CL           ; CL - primeira colu
0335  FE C5                         INC     CH              ; CH - linha
0337  B7 07                         MOV     BH,07H          ; b & p
0339  B8 0600                       MOV     AX,0600H        ; limpa regiao
033C  CD 10                         INT     VIDEO
033E                        CLR_1:
033E  59                            POP     CX
033F  C3                            RET
```

```
--- Objeto -----------------   --- Programa Fonte ----------------------------------------

0340                           TABULA:
0340  51                               PUSH    CX
0341  B4 0F                            MOV     AH,15
0343  CD 10                            INT     VIDEO                   ; Estado do video
0345  8A DC                            MOV     BL,AH                   ; BL - # de colunas
0347  B4 03                            MOV     AH,3
0349  CD 10                            INT     VIDEO                   ; le posicao do cursor
034B  80 E2 F8                         AND     DL,0F8H
034E  80 C2 08                         ADD     DL,8                    ; tabula de 8 em 8
0351  3A D3                            CMP     DL,BL
0353  72 04                            JB      TAB_1                   ; para na ultima coluna
0355  8A D3                            MOV     DL,BL
0357  FE CA                            DEC     DL
0359                           TAB_1:
0359  B4 02                            MOV     AH,2
035B  CD 10                            INT     VIDEO                   ; posiciona cursor
035D  59                               POP     CX
035E  C3                               RET

035F                           EST_NOR ENDP

                               ; EST_VT - posicionamento vertical

035F                           EST_VT  PROC    NEAR
035F  51                               PUSH    CX
0360  50                               PUSH    AX
0361  B4 0F                            MOV     AH,15
0363  CD 10                            INT     VIDEO                   ; Estado do video
0365  B4 03                            MOV     AH,3
0367  CD 10                            INT     VIDEO                   ; le posicao do cursor
0369  58                               POP     AX
036A  8A F0                            MOV     DH,AL                   ; altera linha
036C  B4 02                            MOV     AH,2
036E  CD 10                            INT     VIDEO                   ; reposiciona
0370  59                               POP     CX
0371  C7 06 002D R 02C9 R             MOV     ESTADO,OFFSET EST_NOR
0377  C3                               RET
0378                           EST_VT  ENDP

                               ; EST_HT

0378                           EST_HT  PROC    NEAR
0378  51                               PUSH    CX
0379  50                               PUSH    AX
037A  B4 0F                            MOV     AH,15
037C  CD 10                            INT     VIDEO                   ; Estado do video
037E  B4 03                            MOV     AH,3
0380  CD 10                            INT     VIDEO                   ; le posicao do cursor
0382  58                               POP     AX
0383  8A D0                            MOV     DL,AL                   ; altera coluna
0385  B4 02                            MOV     AH,2
0387  CD 10                            INT     VIDEO                   ; reposiciona
0389  59                               POP     CX
```

```
--- Objeto ----------------  --- Programa Fonte ----------------------------------------

038A  C7 06 002D R 02C9 R          MOV     ESTADO,OFFSET EST_NOR
0390  C3                           RET
0391                         EST_HT  ENDP


                             ; LE2_TEL - Leitura sem espera em TELA


0391                         LE2_TEL PROC    NEAR
0391  A0 002F R                     MOV     AL,TEC_ANT
0394  0A C0                        OR      AL,AL
0396  74 08                        JZ      LT2_2                   ; Tecla pendente ?
0398                         LT2_1:                                 ; Sim: informa-a
0398  26: 88 47 0D                 MOV     ES:[BX].CAR,AL
039C  B8 0100                      MOV     AX,ST_LIV               ;      Status de livre
039F  C3                           RET
03A0                         LT2_2:
03A0  B4 01                        MOV     AH,1                    ; Nao: Consulta BIOS
03A2  CD 16                        INT     TECLADO                 ;      Tecla pendente?
03A4  75 F2                        JNZ     LT2_1                   ;      Sim: informa-a
03A6  B8 0300                      MOV     AX,ST_OCUP              ;      Nao: Status de
                                                                   ;           ocupado
03A9  C3                           RET
03AA                         LE2_TEL ENDP


                             ; LE_TEL - Leitura em TELA


03AA                         LE_TEL  PROC    NEAR
03AA  26: 8B 4F 12                 MOV     CX,ES:[BX].CNT          ; CX - # de caracteres
                                                                   ;      por ler
03AE  26: C4 7F 0E                 LES     DI,ES:[BX].DTA          ; ES:DI - dados
03B2  FC                           CLD
03B3  E3 19                        JCXZ    LTEL_3
03B5                         LTEL_1:                               ; enquanto nao acabar
03B5  2A C0                        SUB     AL,AL
03B7  86 06 002F R                 XCHG    AL,TEC_ANT
03BB  0A C0                        OR      AL,AL                   ;   tem tecla anterior ?
03BD  75 0C                        JNZ     LTEL_2                  ;   Sim: consome
03BF  8A E0                        MOV     AH,AL
03C1  CD 16                        INT     TECLADO                 ;   Nao: Le via BIOS
03C3  0A C0                        OR      AL,AL
03C5  75 04                        JNZ     LTEL_2                  ;   "Extended Code" ?
03C7  88 26 002F R                 MOV     TEC_ANT,AH              ;   Sim: guarda segundo
                                                                   ;        codigo
03CB                         LTEL_2:
03CB  AA                           STOSB                           ;   coloca no buffer
03CC  E2 E7                        LOOP    LTEL_1
03CE                         LTEL_3:
03CE  B8 0100                      MOV     AX,ST_LIV               ; Status de fim normal
03D1  C3                           RET
03D2                         LE_TEL ENDP
```

```
-- Objeto ----------------   --- Programa Fonte ----------------------------------------

                            ; STA_TEL - Status de leitura em TELA

03D2                        STA_TEL PROC    NEAR
03D2  A0 002F R                     MOV     AL,TEC_ANT
03D5  0A C0                         OR      AL,AL
03D7  74 04                         JZ      ST_2                ; Tecla pendente ?
03D9                        ST_1:                               ; Sim:
03D9  B8 0100                       MOV     AX,ST_LIV           ;       Status de livre
03DC  C3                            RET
03DD                        ST_2:
03DD  B4 01                         MOV     AH,1                ; Nao: Consulta BIOS
03DF  CD 16                         INT     TECLADO             ;       Tecla pendente?
03E1  75 F6                         JNZ     ST_1                ;       Sim: Status de
                                                                ;            livre
03E3  B8 0300                       MOV     AX,ST_OCUP          ;       Nao: Status de
                                                                ;            ocupado
03E6  C3                            RET
03E7                        STA_TEL ENDP


                            ; FLS_TEL - Limpa fila de leitura em TELA

03E7                        FLS_TEL PROC    NEAR
03E7  C6 06 002F R 00               MOV     TEC_ANT,0           ; Cancela tecla pendente
03EC                        FLS_1:
03EC  B4 01                         MOV     AH,1
03EE  CD 16                         INT     TECLADO
03F0  74 06                         JZ      FLS_2               ; Enquanto tiver no BIOS
03F2  B4 00                         MOV     AH,0
03F4  CD 16                         INT     TECLADO             ;   Le do BIOS
03F6  EB F4                         JMP     FLS_1
03F8                        FLS_2:
03F8  B8 0100                       MOV     AX,ST_LIV
03FB  C3                            RET
03FC                        FLS_TEL ENDP

03FC                        FIM_TRT LABEL   BYTE

03FC                        TRAT    ENDS
                                    END
```

PROGRAMA RESIDENTE

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

                             ;
                             ; PROGRAMA EXEMPLO #3 - PROGRAMA RESIDENTE
                             ;
0000                         RESID   SEGMENT
                                     ASSUME CS:RESID, DS:RESID

002C                                 ORG     02CH
002C  ????                   ENVRN   DW      ?               ; segmento do environment

0100                                 ORG     100H            ; inicio de programa .COM

0100                         PILHA   LABEL   WORD            ; pilha na area padrao de DTA
                                                             ; (PSP:80h a PSP:FFh)

0100                         INICIO:
0100  E9 0364 R                      JMP     COMECO

                             ;
                             ; CONSTANTES
                             ;

= 0021                       DOS     EQU     021H            ; int de sw p/ chamar DOS
= 0009                       MOSMEN  EQU     009H            ; mostra mensagem
= 0019                       UNIPAD  EQU     019H            ; informa unidade padrao
= 0025                       ALTVET  EQU     025H            ; altera conteudo de vetor
= 002C                       HORATL  EQU     02CH            ; informa hora atual
= 0031                       KEEP    EQU     031H            ; termina e mantem residente
= 0034                       INFIND  EQU     034H            ; informa end. do indicador de
                                                             ;   chamada em curso
= 0035                       INFVET  EQU     035H            ; informa conteudo de vetor
= 003E                       CLOSE   EQU     03EH            ; fecha handle
= 0047                       DIRCUR  EQU     047H            ; informa diretorio corrente
= 0049                       LIBMEM  EQU     049H            ; libera memoria
= 004C                       EXIT    EQU     04CH            ; funcao termina programa

= 0016                       TEC     EQU     016H            ; int de sw p/ chamar BIOS
= 0000                       TEC_LE  EQU     000h            ; le tecla
= 0001                       TEC_INF EQU     001H            ; informa se tecla apertada
= 0002                       TEC_ST  EQU     002H            ; fornece estado do teclado
= 0010                       VIDEO   EQU     010H            ; int de sw p/ chamar BIOS
= 000E                       VID_TTY EQU     00EH            ; escreve no video
= 000F                       VID_ST  EQU     00FH            ; fornece estado do video

= 0007                       BEL     EQU     007H            ; soa alto-falante
= 000D                       CR      EQU     00DH            ; carriage return
= 000A                       LF      EQU     00AH            ; line feed

= 0070                       ATRIB   EQU     070H            ; atributo de video

= 0060                       PORTA   EQU     060H            ; port A da 8255
= 0020                       INTCTR  EQU     020H            ; controlador de interrupcoes
= 0020                       EOI     EQU     020H            ; comando EOI
```

```
-- Objeto ----------------   --- Programa Fonte ----------------------------------------

= 002A                      SHF_ESQ EQU     02AH        ; "scan code" do shift esquerdo
= 0001                      SHF_DIR EQU     001H        ; bit de shift direito no
                                                        ;   estado do teclado

                            ;
                            ; VARIAVEIS
                            ;

0103  ????                  SEG_VID DW      ?           ; segmento do video
0105  00                    ATIVO   DB      0           ; 0 se programa ativo
0106  00                    ATIVAR  DB      0           ; 1 se ativacao pendente
0107  00                    NOBIOS  DB      0           ; 1 se acessando disco via BIOS

0108                        NODOS   LABEL   DWORD       ; endereco do indicador
0108  ????                  ND_O    DW      ?           ;   de DOS em uso
010A  ????                  ND_S    DW      ?

010C                        TECLADO LABEL   DWORD       ; endereco original de INT 9
010C  ????                  TEC_O   DW      ?
010E  ????                  TEC_S   DW      ?

0110                        DISCO   LABEL   DWORD       ; endereco original de INT 13
0110  ????                  DSC_O   DW      ?
0112  ????                  DSC_S   DW      ?

0114                        RELOGIO LABEL   DWORD       ; endereco original de INT 1C
0114  ????                  REL_O   DW      ?
0116  ????                  REL_S   DW      ?

0118                        ESPERA  LABEL   DWORD       ; endereco original de INT 28
0118  ????                  ESP_O   DW      ?
011A  ????                  ESP_S   DW      ?

011C  ????                  CRIT_O  DW      ?           ; endereco original de INT 24h
011E  ????                  CRIT_S  DW      ?

0120  ????                  PIL_O   DW      ?           ; endereco original da pilha
0122  ????                  PIL_S   DW      ?


                            ;
                            ; MENSAGENS DA PARTE RESIDENTE
                            ;

0124  ??                    UNID    DB      ?                   ; Unidade padrao
0125  3A 5C                         DB      ':\'
0127     40 [               DIRET   DB      64 DUP (?)          ; Diretorio corrente
            ??
               ]

0167  39 39 3A 39 39 3A     HORA    DB      '99:99:99'          ; Hora atual
      39 39
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------

                                ;
                                ; Programa RESIDENTE
                                ;

016F                            MOSTRA  PROC    NEAR

016F  50                                PUSH    AX
   9  2E: 89 26 0120 R                  MOV     CS:PIL_0,SP
   5  2E: 8C 16 0122 R                  MOV     CS:PIL_S,SS       ; Salva endereco da pilha
017A  8C C8                             MOV     AX,CS
017C  8E D0                             MOV     SS,AX
017E  BC 0100 R                         MOV     SP,OFFSET PILHA   ; Usa pilha local

0181  53                                PUSH    BX
0182  51                                PUSH    CX
0183  52                                PUSH    DX
0184  56                                PUSH    SI
0185  57                                PUSH    DI
0186  55                                PUSH    BP
0187  1E                                PUSH    DS
0188  06                                PUSH    ES                ; Salva registradores
0189  8E D8                             MOV     DS,AX
018B  8E C0                             MOV     ES,AX             ; Inicia DS e ES
018D  C6 06 0105 R 01                   MOV     ATIVO,1           ; Programa ativado
0192  C6 06 0106 R 00                   MOV     ATIVAR,0          ; Pendencia atendida
0197  FB                                STI                       ; Habilita interrupcoes

0198  8E 1E 0103 R                      MOV     DS,SEG_VID
019C  2B F6                             SUB     SI,SI
019E  BF 0339 R                         MOV     DI,OFFSET SALVA
01A1  B9 0050                           MOV     CX,80
01A4  FC                                CLD
01A5  F3/ A5                        REP MOVSW                     ; Salva 1a linha do video
01A7  8E D8                             MOV     DS,AX

01A9  B0 24                             MOV     AL,24H
01AB  B4 35                             MOV     AH,INFVET
01AD  CD 21                             INT     DOS                   ; le vetor 24h p/ ES:BX
01AF  89 1E 011C R                      MOV     CRIT_0,BX
01B3  8C 06 011E R                      MOV     CRIT_S,ES             ; Salva conteudo original
01B7  BA 02A5 R                         MOV     DX,OFFSET ERR_CRIT
01BA  B0 24                             MOV     AL,24H
01BC  B4 25                             MOV     AH,ALTVET
01BE  CD 21                             INT     DOS                   ; Assume INT 24h

01C0  B4 19                             MOV     AH,UNIPAD
01C2  CD 21                             INT     DOS
01C4  04 41                             ADD     AL,'A'
01C6  A2 0124 R                         MOV     UNID,AL           ; Obtem unidade padrao
01C9  2A D2                             SUB     DL,DL
01CB  BE 0127 R                         MOV     SI,OFFSET DIRET
01CE  B4 47                             MOV     AH,DIRCUR         ; Obtem diretorio corrente
01D0  CD 21                             INT     DOS
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

01D2  B4 2C                        MOV     AH,HORATL
01D4  CD 21                        INT     DOS             ; Obtem hora atual
01D6  BF 0167 R                    MOV     DI,OFFSET HORA
01D9  8A C5                        MOV     AL,CH
01DB  D4 0A                        AAM                     ; Converte hora em
01DD  05 3030                      ADD     AX,3030H        ;   dois digitos
01E0  88 25                        MOV     [DI],AH
01E2  88 45 01                     MOV     [DI+1],AL
01E5  8A C1                        MOV     AL,CL
01E7  D4 0A                        AAM                     ; Converte minutos em
01E9  05 3030                      ADD     AX,3030H        ;   dois digitos
01EC  88 65 03                     MOV     [DI+3],AH
01EF  88 45 04                     MOV     [DI+4],AL
01F2  8A C6                        MOV     AL,DH
01F4  D4 0A                        AAM                     ; Converte segundos em
01F6  05 3030                      ADD     AX,3030H        ;   dois digitos
01F9  88 65 06                     MOV     [DI+6],AH
01FC  88 45 07                     MOV     [DI+7],AL

01FF  BE 06 0103 R                 MOV     ES,SEG_VID
0203  2B FF                        SUB     DI,DI
0205  B9 0050                      MOV     CX,80
0208  B4 70                        MOV     AH,ATRIB
020A  B0 20                        MOV     AL,20H
020C  FC                           CLD
020D  F3/ AB                 REP   STOSW                   ; Limpa primeira linha

020F  2B FF                        SUB     DI,DI
0211  BE 0124 R                    MOV     SI,OFFSET UNID
0214                         MOS_1:
0214  AC                           LODSB
0215  0A C0                        OR      AL,AL
0217  74 04                        JZ      MOS_2
0219  AA                           STOSB
021A  47                           INC     DI              ; (pula atributo)
021B  EB F7                        JMP     MOS_1           ; Mostra unidade e diretorio
021D                         MOS_2:
021D  BF 0090                      MOV     DI,(80 - 8)*2
0220  BE 0167 R                    MOV     SI,OFFSET HORA
0223  B9 0008                      MOV     CX,8
0226                         MOS_3:
0226  A4                           MOVSB
0227  47                           INC     DI              ; (pula atributo)
0228  E2 FC                        LOOP    MOS_3           ; Mostra hora
022A                         MOS_4:
022A  9C                           PUSHF
022B  FF 1E 0118 R                 CALL    ESPERA          ; Simula INT 28h
022F  B4 01                        MOV     AH,TEC_INF
0231  CD 16                        INT     TEC
0233  74 F5                        JZ      MOS_4           ; Espera digitar uma tecla

0235  B4 00                        MOV     AH,TEC_LE
0237  CD 16                        INT     TEC             ; Le a tecla
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

0239  2B FF                        SUB     DI,DI
023B  BE 0339 R                    MOV     SI,OFFSET SALVA
023E  B9 0050                      MOV     CX,80
0241  FC                           CLD
0242  F3/ A5                   REP MOVSW                   ; Restaura 1a linha do video

0244  8B 16 011C R                 MOV     DX,CRIT_O
0248  8E 1E 011E R                 MOV     DS,CRIT_S
024C  B0 24                        MOV     AL,24H
024E  B4 25                        MOV     AH,ALTVET
0250  CD 21                        INT     DOS             ; Restaura INT 24h

0252  FA                           CLI                     ; Desabilita interrupcoes
0253  2E: C6 06 0105 R 00          MOV     CS:ATIVO,0      ; Programa desativado
0259  07                           POP     ES
025A  1F                           POP     DS
025B  5D                           POP     BP
025C  5F                           POP     DI
025D  5E                           POP     SI
025E  5A                           POP     DX
025F  59                           POP     CX
0260  5B                           POP     BX              ; Restaura registradores
0261  2E: 8E 16 0122 R             MOV     SS,CS:PIL_S
0266  2E: 8B 26 0120 R             MOV     SP,CS:PIL_O     ; Volta a pilha original
026B  58                           POP     AX
026C  C3                           RET

026D                       MOSTRA  ENDP


                           ;
                           ; Rotina que intercepta INT 9h
                           ;

026D                       INT_TEC:
026D  2E: 80 3E 0105 R 01          CMP     CS:ATIVO,1
0273  74 16                        JE      ITEC_2          ; Se nao ativo
0275  50                           PUSH    AX
0276  E4 60                        IN      AL,PORTA
0278  3C 2A                        CMP     AL,SHF_ESQ
027A  75 0E                        JNE     ITEC_1          ;   e digitou Shift Esquerdo
027C  B4 02                        MOV     AH,TEC_ST
027E  CD 16                        INT     TEC
0280  A8 01                        TEST    AL,SHF_DIR
0282  74 06                        JZ      ITEC_1          ;   e Shift Direito apertado
0284  2E: C6 06 0106 R 01          MOV     CS:ATIVAR,1     ;     Ativar prog. residente
028A                       ITEC_1:
028A  58                           POP     AX
028B                       ITEC_2:
028B  2E: FF 2E 010C R             JMP     CS:TECLADO      ; Trata tecla normalmente
```

```
--- Objeto ----------------  --- Programa Fonte ----------------------------------------

                             ;
                             ; Rotina que intercepta INT 13h
                             ;

0290                         INT_DSC PROC    FAR

0290  2E: C6 06 0107 R 01           MOV     CS:NOBIOS,1     ; Executando BIOS
0296  9C                            PUSHF
0297  2E: FF 1E 0110 R              CALL    CS:DISCO        ; Chama rotina original
029C  2E: C6 06 0107 R 00           MOV     CS:NOBIOS,0     ; Terminou de usr BIOS
02A2  CA 0002                       RET     2               ; Retorna c/ flags do BIOS

02A5                         INT_DSC ENDP


                             ;
                             ; Rotina que intercepta INT 24h
                             ;

02A5                         ERR_CRIT PROC   FAR

02A5  B4 0E                         MOV     AH,VID_TTY
02A7  B0 07                         MOV     AL,BEL
02A9  CD 10                         INT     VIDEO           ; Soa Alto-falante
02AB  B0 01                         MOV     AL,1            ; Tenta novamente
02AD  CF                            IRET

02AE                         ERR_CRIT ENDP


                             ;
                             ; Rotina que intercepta INT 1Ch
                             ;

02AE                         INT_REL:
02AE  2E: 80 3E 0106 R 01           CMP     CS:ATIVAR,1     ; Se ativacao pendente
02B4  75 44                         JNE     IREL_3
02B6  2E: 80 3E 0107 R 00           CMP     CS:NOBIOS,0     ;    e nao esta no BIOS
02BC  75 3C                         JNE     IREL_3
02BE  50                            PUSH    AX
02BF  53                            PUSH    BX
02C0  1E                            PUSH    DS
02C1  2E: C5 1E 0108 R              LDS     BX,CS:NODOS
02C6  80 3F 00                      CMP     BYTE PTR [BX],0
02C9  75 2C                         JNE     IREL_2          ;    e nem no DOS
02CB  B4 0F                         MOV     AH,VID_ST
02CD  CD 10                         INT     VIDEO           ;    e esta em modo 80x24
02CF  BB B000                       MOV     BX,0B000H
02D2  3C 07                         CMP     AL,7
02D4  74 0B                         JE      IREL_1
02D6  BB B800                       MOV     BX,0B800H
02D9  3C 02                         CMP     AL,2
02DB  74 04                         JE      IREL_1
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

02DD  3C 03                            CMP     AL,3
02DF  75 16                            JNE     IREL_2
02E1                           IREL_1:
02E1  2E: 89 1E 0103 R                 MOV     CS:SEG_VID,BX   ;       salva segto do video
02E6  B0 20                            MOV     AL,EOI
02E8  E6 20                            OUT     INTCTR,AL       ;       encerra interrupcao
02EA  1F                               POP     DS
02EB  5B                               POP     BX
02EC  58                               POP     AX
02ED  9C                               PUSHF
02EE  2E: FF 1E 0114 R                 CALL    CS:RELOGIO      ;       chama rot. anterior
02F3  E8 016F R                        CALL    MOSTRA          ;       ativa prog. residente
02F6  CF                               IRET
02F7                           IREL_2:
02F7  1F                               POP     DS              ; Senao
02F8  5B                               POP     BX
02F9  58                               POP     AX
02FA                           IREL_3:
02FA  2E: FF 2E 0114 R                 JMP     CS:RELOGIO      ;    desvia p/ rot. anterior

                                       ;
                                       ; Rotina que trata INT 28h
                                       ;

02FF                           INT_ESP:
02FF  2E: 80 3E 0106 R 01              CMP     CS:ATIVAR,1     ; Se ativacao pendente
0305  75 2D                            JNE     IESP_3
0307  2E: 80 3E 0107 R 00              CMP     CS:NOBIOS,0     ;    e nao esta no BIOS
030D  75 25                            JNE     IESP_3
030F  50                               PUSH    AX
0310  53                               PUSH    BX
0311  B4 0F                            MOV     AH,VID_ST
0313  CD 10                            INT     VIDEO           ;    e esta em modo B0x24
0315  B8 B000                          MOV     BX,0B000H
0318  3C 07                            CMP     AL,7
031A  74 0B                            JE      IESP_1
031C  B8 B800                          MOV     BX,0B800H
031F  3C 02                            CMP     AL,2
0321  74 04                            JE      IESP_1
0323  3C 03                            CMP     AL,3
0325  75 0B                            JNE     IESP_2
0327                           IESP_1:
0327  2E: 89 1E 0103 R                 MOV     CS:SEG_VID,BX   ;       salva segto do video
032C  5B                               POP     BX
032D  58                               POP     AX
032E  E8 016F R                        CALL    MOSTRA          ;       ativa prog. residente
0331  CF                               IRET
0332                           IESP_2:                         ; Senao
0332  5B                               POP     BX
0333  58                               POP     AX
0334                           IESP_3:
0334  2E: FF 2E 0118 R                 JMP     CS:ESPERA       ;    Desvia p/ rot. anterior
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

                              ;
                              ; A area abaixo sera utilizada pelo programa residente
                              ; para salvar a primeira linha do video
                              ;

0339                          SALVA   LABEL   BYTE

                              ;
                              ; MENSAGENS DA INICIACAO
                              ;

0339  50 72 6F 67 72 61       JA_RES  DB      'Programa ja instalado.$'
      6D 61 20 6A 61 20
      69 6E 73 74 61 6C
      61 64 6F 2E 24
0350  50 72 6F 67 72 61       INST_OK DB      'Programa instalado.$'
      6D 61 20 69 6E 73
      74 61 6C 61 64 6F
      2E 24

                              ;
                              ; INSTALACAO
                              ;

0364                          COMECO:
0364  B0 09                            MOV     AL,9H
0366  B4 35                            MOV     AH,INFVET
0368  CD 21                            INT     DOS                     ; le vetor 09h p/ ES:BX
036A  81 FB 026D R                     CMP     BX,OFFSET INT_TEC
036E  75 0D                            JNE     INSTAL

0370  BA 0339 R                        MOV     DX,OFFSET JA_RES
0373  B4 09                            MOV     AH,MOSMEN
0375  CD 21                            INT     DOS                     ;  'ja' instalado'
0377  B0 01                            MOV     AL,1
0379  B4 4C                            MOV     AH,EXIT                 ;  Termina,
037B  CD 21                            INT     DOS                     ;    ERRORLEVEL = 1
037D                          INSTAL:
037D  FA                               CLI
037E  89 1E 010C R                     MOV     TEC_O,BX
0382  8C 06 010E R                     MOV     TEC_S,ES                ; Salva valor original
0386  BA 026D R                        MOV     DX,OFFSET INT_TEC
0389  B0 09                            MOV     AL,9
038B  B4 25                            MOV     AH,ALTVET
038D  CD 21                            INT     DOS                     ; Assume INT 9h
038F  B0 13                            MOV     AL,13H
0391  B4 35                            MOV     AH,INFVET
0393  CD 21                            INT     DOS                     ; le vetor 13h p/ ES:BX
0395  89 1E 0110 R                     MOV     DSC_O,BX
0399  8C 06 0112 R                     MOV     DSC_S,ES                ; Salva valor original
039D  BA 0290 R                        MOV     DX,OFFSET INT_DSC
03A0  B0 13                            MOV     AL,13H
03A2  B4 25                            MOV     AH,ALTVET
03A4  CD 21                            INT     DOS                     ; Assume INT 13h
```

```
--- Objeto ----------------   --- Programa Fonte ----------------------------------------

03A6  B0 1C                        MOV     AL,1CH
03A8  B4 35                        MOV     AH,INFVET
03AA  CD 21                        INT     DOS                     ; le vetor 1Ch p/ ES:BX
03AC  89 1E 0114 R                 MOV     REL_O,BX
03B0  8C 06 0116 R                 MOV     REL_S,ES                ; Salva valor original
03B4  BA 02AE R                    MOV     DX,OFFSET INT_REL
03B7  B0 1C                        MOV     AL,1CH
03B9  B4 25                        MOV     AH,ALTVET
03BB  CD 21                        INT     DOS                     ; Assume INT 1Ch

03BD  B0 28                        MOV     AL,28H
03BF  B4 35                        MOV     AH,INFVET
03C1  CD 21                        INT     DOS                     ; le vetor 28h p/ ES:BX
03C3  89 1E 0118 R                 MOV     ESP_O,BX
03C7  8C 06 011A R                 MOV     ESP_S,ES                ; Salva valor original
03CB  BA 02FF R                    MOV     DX,OFFSET INT_ESP
03CE  B0 28                        MOV     AL,28H
03D0  B4 25                        MOV     AH,ALTVET
03D2  CD 21                        INT     DOS                     ; Assume INT 28h
03D4  FB                           STI

03D5  B4 34                        MOV     AH,INFIND
03D7  CD 21                        INT     DOS                     ; Salva endereco do
03D9  89 1E 0108 R                 MOV     ND_O,BX                 ;   indicador
03DD  8C 06 010A R                 MOV     ND_S,ES                 ;   de DOS em uso

03E1  BA 0350 R                    MOV     DX,OFFSET INST_OK
03E4  B4 09                        MOV     AH,MOSMEN
03E6  CD 21                        INT     DOS

03E8  BB 0005                      MOV     BX,5
03EB                       FECHA:
03EB  4B                           DEC     BX
03EC  B4 3E                        MOV     AH,CLOSE
03EE  CD 21                        INT     DOS
03F0  0B DB                        OR      BX,BX
03F2  75 F7                        JNZ     FECHA                   ; Fecha handles 0 a 4

03F4  8E 06 002C R                 MOV     ES,ENVRN
03F8  B4 49                        MOV     AH,LIBMEM               ; Libera memoria do
03FA  CD 21                        INT     DOS                     ;   environment

03FC  BA 03E8 R                    MOV     DX,OFFSET SALVA + 175
03FF  B1 04                        MOV     CL,4
0401  D3 EA                        SHR     DX,CL
0403  B0 00                        MOV     AL,0
0405  B4 31                        MOV     AH,KEEP                 ; Termina,
0407  CD 21                        INT     DOS                     ;   mantendo residente

0409                       RESID   ENDS

                                   END     INICIO
```

APÊNDICE A - INTERRUPÇÕES DO DOS

Vetor	Uso
20h	Terminação de Programas
21h	Chamada de Função do DOS
22h	Endereço de Terminação de Programa
23h	Interrupção de Programa ("Control Break")
24h	Tratamento de Erro Crítico
25h	Leitura Absoluta de Disco
26h	Escrita Absoluta em Disco
27h	Termina Programa, Mantendo Residente
28h	Aguardando Caractere
29h	Escrita Rápida em Dispositivo
2Eh	Chamada ao COMMAND
2Fh	Multiplexação (DOS 3.00)

APÊNDICE B - FUNÇÕES DO DOS, POR USO

Funções de Caractere

01h	Leitura do Console	
02h	Escrita no Console	
03h	Leitura do Dispositivo Auxiliar	
04h	Escrita no Dispositivo Auxiliar	
05h	Escrita na Impressora	
06h	Entrada e Saída Direta no Console	
07h	Entrada Direta do Console, Sem Eco	
08h	Entrada do Console, Sem Eco	
09h	Escrita de Cadeia no Console	
0Ah	Leitura do Console com Edição	
0Bh	Informa Estado do Console	
0Ch	Leitura do Console com Espera	

Funções de Arquivo/Diretório

0Dh	Reinicia Disco	
0Eh	Seleciona Unidade Padrão	
0Fh	Abertura de Arquivo, via FCB	
10h	Fechamento de Arquivo, via FCB	
11h	Busca Primeira Entrada, via FCB	
12h	Busca Próxima Entrada, via FCB	
13h	Suprime Arquivo, via FCB	
14h	Leitura Sequencial, via FCB	
15h	Escrita Sequencial, via FCB	
16h	Cria Arquivo, via FCB	
17h	Renomeia Arquivo, via FCB	
19h	Informa Unidade Padrão	
1Ah	Altera DTA	
1Bh	Obtém Informações da FAT da Unidade Padrão	
1Ch	Obtém Informações da FAT	
21h	Leitura Randômica, via FCB	
22h	Escrita Randômica, via FCB	
23h	Informa Tamanho de Arquivo, via FCB	
24h	Posiciona Registro Relativo	
27h	Leitura de Bloco, via FCB	
28h	Escrita de Bloco, via FCB	
29h	Analisa Nome de Arquivo	
2Eh	Liga/Desliga Opção de Verificação	
2Fh	Informa Endereço da DTA	(2)
36h	Informa Espaço Disponível em Disco	(2)
39h	Cria Subdiretório (MKDIR)	(2)
3Ah	Remove Subdiretório (RMDIR)	(2)
3Bh	Altera Diretório Atual (CHDIR)	(2)
3Ch	Cria Arquivo, via Cadeia ASCIIZ	(2)
3Dh	Abre Arquivo, via Handle	(2)
3Eh	Fecha Arquivo, via Handle	(2)
3Fh	Leitura via Handle	(2)
40h	Escrita via Handle	(2)

41h	Remove Arquivo, via cadeia ASCIIZ (UNLINK)	(2)
42h	Movimenta Ponteiro de Leitura/Escrita (LSSEK)	(2)
43h	Informa/Altera Atributo de Arquivo (CHMOD)	(2)
44h	Controle de Entrada/Saída (IOCTL)	(2)
45h	Duplica Handle (DUP)	(2)
46h	Duplicação Forçada de Handle (FDUP)	(2)
47h	Informa Diretório Corrente	(2)
4Eh	Procura Primeira Entrada, via cadeia ASCIIZ	(2)
4Fh	Procura Próxima Entrada, via cadeia ASCIIZ	(2)
54h	Informa Opção de Verificação	(2)
56h	Renomeia Arquivo, via Cadeia ASCIIZ	(2)
57h	Informa/Altera Data e Hora de Arquivo	(2)
5Ah	Cria Arquivo Único	(3)
5Bh	Cria Arquivo Novo	(3)
5Ch	Controla Acesso a Arquivo	(3)

Funções de Gerenciamento de Memória

48h	Aloca Memória	(2)
49h	Libera Bloco de Memória	(2)
4Ah	Modifica Tamanho de Bloco de Memória	(2)
58h	Informa/Altera Algorítimo de Alocação	(3) (*)

Funções de Gerenciamento de Programas

00h	Termina Programa	
26h	Cria Nova PSP	
31h	Termina Processo e Mantém Residente	(2)
4Bh	Carrega e Executa Programa (EXEC)	(2)
4Ch	Termina Processo (EXIT)	(2)
4Dh	Informa Código de Término (WAIT)	(2)
50h	Altera PSP Atual	(2) (*)
51h	Informa PSP Atual	(2) (*)
62h	Informa PSP Atual	(3)

Outras Funções

25h	Altera Vetor de Interrupção	
2Ah	Informa Data Atual	
2Bh	Altera Data Atual	
2Ch	Informa Hora Atual	
2Dh	Altera Hora Atual	
30h	Informa Versão do DOS	
33h	Informa/Altera Opção de Interrupção	(2)
34h	Informa End. do Indicador de Chamada em Curso	(2) (*)
35h	Informa Vetor de Interrupção	(2)
37h	Informa/Altera "SwitchChar"/"AvailDev"	(2) (*)
38h	Informações Dependentes de País	(2)
59h	Fornece Código de Erro Expandido	(3)

(2) somente a partir da versão 2

(3) somente a partir da versão 3

(*) não documentada

APÊNDICE C - FUNÇÕES DO DOS, POR NÚMERO

00h	Termina Programa	
01h	Leitura do Console	
02h	Escrita no Console	
03h	Leitura do Dispositivo Auxiliar	
04h	Escrita no Dispositivo Auxiliar	
05h	Escrita na Impressora	
06h	Entrada e Saída Direta no Console	
07h	Entrada Direta do Console, Sem Eco	
08h	Entrada do Console, Sem Eco	
09h	Escrita de Cadeia no Console	
0Ah	Leitura do Console com Edição	
0Bh	Informa Estado do Console	
0Ch	Leitura do Console com Espera	
0Dh	Reinicia Disco	
0Eh	Seleciona Unidade Padrão	
0Fh	Abertura de Arquivo, via FCB	
10h	Fechamento de Arquivo, via FCB	
11h	Busca Primeira Entrada, via FCB	
12h	Busca Próxima Entrada, via FCB	
13h	Suprime Arquivo, via FCB	
14h	Leitura Sequencial, via FCB	
15h	Escrita Sequencial, via FCB	
16h	Cria Arquivo, via FCB	
17h	Renomeia Arquivo, via FCB	
19h	Informa Unidade Padrão	
1Ah	Altera DTA	
1Bh	Obtém Informações da FAT da Unidade Padrão	
1Ch	Obtém Informações da FAT	
21h	Leitura Randômica, via FCB	
22h	Escrita Randômica, via FCB	
23h	Informa Tamanho de Arquivo, via FCB	
24h	Posiciona Registro Relativo	
25h	Altera Vetor de Interrupção	
26h	Cria Nova PSP	
27h	Leitura de Bloco, via FCB	
28h	Escrita de Bloco, via FCB	
29h	Analisa Nome de Arquivo	
2Ah	Informa Data Atual	
2Bh	Altera Data Atual	
2Ch	Informa Hora Atual	
2Dh	Altera Hora Atual	
2Eh	Liga/Desliga Opção de Verificação	
2Fh	Informa Endereço da DTA	(2)
30h	Informa Versão do DOS	
31h	Termina Processo e Mantém Residente	(2)
33h	Informa/Altera Opção de Interrupção	(2)
34h	Informa End. do Indicador de Chamada em Curso	(2) (*)
35h	Informa Vetor de Interrupção	(2)
36h	Informa Espaço Disponível em Disco	(2)
37h	Informa/Altera "SwitchChar"/"AvailDev"	(2) (*)

38h	Informações Dependentes de País	(2)
39h	Cria Subdiretório (MKDIR)	(2)
3Ah	Remove Subdiretório (RMDIR)	(2)
3Bh	Altera Diretório Atual (CHDIR)	(2)
3Ch	Cria Arquivo, via Cadeia ASCIIZ	(2)
3Dh	Abre Arquivo, via Handle	(2)
3Eh	Fecha Arquivo, via Handle	(2)
3Fh	Leitura via Handle	(2)
40h	Escrita via Handle	(2)
41h	Remove Arquivo, via cadeia ASCIIZ (UNLINK)	(2)
42h	Movimenta Ponteiro de Leitura/Escrita (LSSEK)	(2)
43h	Informa/Altera Atributo de Arquivo (CHMOD)	(2)
44h	Controle de Entrada/Saída (IOCTL)	(2)
45h	Duplica Handle (DUP)	(2)
46h	Duplicação Forçada de Handle (FDUP)	(2)
47h	Informa Diretório Corrente	(2)
48h	Aloca Memória	(2)
49h	Libera Bloco de Memória	(2)
4Ah	Modifica Tamanho de Bloco de Memória	(2)
4Bh	Carrega e Executa Programa (EXEC)	(2)
4Ch	Termina Processo (EXIT)	(2)
4Dh	Informa Código de Término (WAIT)	(2)
4Eh	Procura Primeira Entrada, via cadeia ASCIIZ	(2)
4Fh	Procura Próxima Entrada, via cadeia ASCIIZ	(2)
50h	Altera PSP Atual	(2) (*)
51h	Informa PSP Atual	(2) (*)
54h	Informa Opção de Verificação	(2)
56h	Renomeia Arquivo, via Cadeia ASCIIZ	(2)
57h	Informa/Altera Data e Hora de Arquivo	(2)
58h	Informa/Altera Algorítimo de Alocação	(3) (*)
59h	Fornece Código de Erro Expandido	(3)
5Ah	Cria Arquivo Único	(3)
5Bh	Cria Arquivo Novo	(3)
5Ch	Controla Acesso a Arquivo	(3)
62h	Informa PSP Atual	(3)

(2) somente a partir da versão 2

(3) somente a partir da versão 3

(*) não documentada