LABORATÓRIO DE LINGUAGEM DE MONTAGEM

OPERAÇÕES COM DISCOS E ARQUIVOS

RESUMO: conceitos importantes do **CAPÍTULO 19** do livro-texto, pags. 395-420

- O comando **FORMAT** do **DOS** reparte cada superfície de um disco em:
    - áreas concêntricas e circulares denominadas **TRILHAS** (*tracks*).
    - cada **TRILHA** é repartida em **SETORES** (*sectors*) de 512 bytes.
    - define-se **CILINDRO** (*cylinder*) ao conjunto de **TRILHAS** de mesma posição em **todas as superfícies** do disco ou conjunto de discos.

- A quantidade de TRILHAS e SETORES de um disco depende de sua construção.

Disco	Cilindros ou Trilhas	Setores por Trilha	Superfícies	Capacidade
Flexível 5 ¼ high density	80	15	2	1.228.800
Flexível 3 ½ double density	80	9	2	737.280
Flexível 3 ½ high density	80	18	2	1.474.560
Rígido 10 Mbytes	306	17	4	10.653.696
Rígido 560 Mbytes				
Rígido 4.3 Gbytes				

- Quando o DOS escreve dados no disco, todas a TRILHAS de um CILINDRO vão sendo preenchidas enquanto houver espaço, antes de mover as cabeças de leitura/escrita para o CILINDRO adjacente.

- As características de um arquivo estão contidas num diretório de arquivos.

- Cada arquivo possui uma **entrada** (*file entry*) de **32 b**ytes, que contém:

Byte	Informação
0 - 7	Nome do arquivo
8 - 10	Extensão
11	Atributo
12 - 21	Reservado para o DOS
22 - 23	Hora-minuto-segundo de criação
24 - 25	Ano-mês-dia de criação
26 - 27	Número inicial do *cluster*
28 - 31	Tamanho do arquivo em bytes

- O **atributo** especifica se o arquivo é somente para leitura (*read-only*), escondido (*hidden*), do sistema, rótulo de volume, subdiretório, ou se foi modificado. Seu valor usual é 20h.

- DOS reserva espaço para arquivos em **CLUSTERS**.

  - Um CLUSTER é uma quantidade fixa de SETORES.
  - Exemplos:

  - Disco flexível de 5¼ double density: 1 CLUSTER = 2 SETORES = 1.024 bytes
  - Disco flexível de 3½ high density: 1 CLUSTER = 1 SETOR = 512 bytes
  - Disco rígido de 4.3 Gbytes: 1 CLUSTER = 128 SETORES = 65.536 bytes

- O primeiro arquivo do disco é posicionado no CLUSTER 2.

- Se o arquivo for menor do que a quantidade de bytes do CLUSTER, o espaço restante fica sem uso. O desperdício típico pode ser de 50 % do tamanho do CLUSTER: 32.768 bytes num disco de 4.3 Gbytes !

- A **Tabela de Alocação de Arquivos** (FAT – *file allocation table*) provê um mapa de como os arquivos estão armazenados no disco.

- Para o DOS:
  - nos discos flexíveis a entrada da FAT (FAT entry) é de 12 bits
  - nos discos rígidos, a entrada da FAT é de 16 ou 32 bits.

- Windows 95 e 98 utilizam entrada da FAT de 32 bits

- Windows NT 4.0 utiliza entrada da FAT de 16 bits

- Como um arquivo é localizado?
  - O diretório de arquivos contem o número **N1** do CLUSTER onde se inicia este arquivo.
  - Na FAT, a posição **N1** indica se o arquivo continua no CLUSTER **N2**, a posição **N2** indica se há continuação para o CLUSTER **N3**, e assim por diante.
  - OU, se for o último CLUSTER, tal posição contem **FFFh** ou **FFFFh**

- **INTERRUPÇÃO INT 21h :** funções que manipulam arquivos

  File handling functions

  Proporcionam um modo conveniente de realizar operações com arquivos.

- **RESUMO DAS FUNÇÕES**

Função	Descrição
3Ch	Abre um arquivo novo
3Dh	Abre um arquivo já existente
3Eh	Fecha um arquivo
3Fh	Lê um arquivo
40h	Escreve um arquivo
42h	Move o ponteiro de um arquivo
43h	Obtém/muda o atributo de um arquivo

- Quando um arquivo é aberto, o DOS assinala ao mesmo um número chamado ***file handle***. De modo que um programa pode **identificar** este arquivo através deste número.

- Números de ***file handles*** pré-definidos:

Número	Dispositivo ou arquivo
0	Teclado
1	Tela do monitor
2	Exibição de erro na tela
3	Dispositivo auxiliar
4	Impressora
5 - 7	Reservado para arquivos a serem abertos
8 - ...	Limite pode ser estendido: ver DOS

- As funções de manipulação de arquivos do INT 21h podem resultar em **erros**, que são identificados por um **código numérico**:

Código HEX	Significado
1	Número de função inválido
2	Arquivo não encontrado
3	Caminho (path) não encontrado
4	Todos os handles já em uso
5	Acesso negado
6	File handle inválido
C	Código de acesso inválido
F	Especificação de drive inválida
10	Tentativa de remover o diretório corrente
11	Não é o mesmo dispositivo
12	Não há mais arquivos a serem encontrados

Interrupção INT 21H – Função 3Ch

Abre um arquivo novo.

Entradas: AH = 3Ch
DS:DX = endereço do nome do arquivo sob forma de um string ASCIIZ
(string que termina com um byte 0)
CL = atributo

Saídas: Se bem sucedido, AX = file handle
Se houver erro, CF = 1 e o código de erro está em AX (3, 4 ou 5)

Exemplo:

```
...
.DATA
NOME      DB        'ARQUIVO1',0
HANDLE    DW        ?
...
.CODE
MOV AX,@DATA
MOV DS,AX           ;inicializa DS
MOV AH,3Ch          ;função para abertura de arquivo novo
LEA DX,NOME
MOV CL,1            ;atributo de leitura apenas
INT 21h
MOV HANDLE,AX       ;salva em AX o handle ou o código de erro
JC  ERRO_OPEN
...
```

Interrupção INT 21H – Função 3Dh

Abre um arquivo já existente.

Entradas: AH = 3Dh
DS:DX = endereço do nome do arquivo sob forma de um string ASCIIZ
(string que termina com um byte 0)
AL = código de acesso: 0 -> abrir para leitura
1 -> abrir para escrita
2 -> abrir para leitura e escrita

Saídas: Se bem sucedido, AX = file handle
Se houver erro, CF = 1 e o código de erro está em AX (2, 4, 5, 12)

Exemplo:

O exemplo anterior pode ser modificado para exemplificar o uso desta função. Faça as modificações necessárias!

Interrupção INT 21H – Função 3Eh

Fecha um arquivo anteriormente aberto
Libera um handle para que outros arquivos possam ser abertos

Entradas: AH = 3Eh
BX = file handle

Saídas: Se houver erro, CF = 1 e o código de erro está em AX (6)

Exemplo:

```
...
.DATA
...
HANDLE    DW   ?       ;já contem um handle válido
...
.CODE
MOV AX,@DATA
MOV DS,AX              ;inicializa DS
...
MOV AH,3Eh             ;função para fechar arquivo
MOV BX,HANDLE          ;obtém o handle armazenado
INT 21h                ;fecha o arquivo
JC  ERRO_CLOSE
...
```

Interrupção INT 21H – Função 3Fh

Lê uma quantidade especificada de bytes de um arquivo aberto e armazena na memória (.DATA)

Entradas: AH = 3Fh
BX = file handle
CX = quantidade de bytes a ser lida
DS:DX = endereço do ***buffer*** na memória

Saídas: Se bem sucedido, AX = quantidade de bytes realmente lidos
Se AX = 0 ou AX < CX, denota que o EOF foi encontrado
Se houver erro, CF = 1 e o código de erro está em AX (5, 6)

Obs.: EOF -> *end of file*

Exemplo: Escrever um trecho para ler 512 bytes (1 setor) de um arquivo especificado

```
...
.DATA
HANDLE    DW   ?       ;já contem um handle válido
BUFFER    DB   512  DUP  (0)
...
.CODE
MOV AX,@DATA
MOV DS,AX             ;inicializa DS
MOV AH,3Fh            ;função para ler bytes de um arquivo
MOV BX,HANDLE         ;obtém o handle armazenado
MOV CX,512
INT 21h               ;efetua leitura no arquivo
JC  ERRO_READ
...
```

Interrupção INT 21H – Função 40h

Escreve uma quantidade especificada de bytes em um arquivo aberto.

Entradas: AH = 40h
BX = file handle
CX = quantidade de bytes a ser escrita
DS:DX = endereço dos dados: ***buffer*** na memória ou string

Saídas: Se bem sucedido, AX = quantidade de bytes realmente escritos
Se AX < CX, denota que o disco está cheio = erro
Se houver erro, CF = 1 e o código de erro está em AX (5, 6)

Exemplo: Escrever um trecho para mostrar uma mensagem na tela do monitor

```
...
.DATA
MSG  DB     'TESTE DE ESCRITA'
...
.CODE
MOV  AX,@DATA
MOV  DS,AX              ;inicializa DS
MOV  AH,40h             ;função para escrever bytes
MOV  BX,1               ;file handle para a tela
MOV  CX,16              ;quantidade de bytes de MSG
LEA  DX,MSG             ;aponta para a mensagem
INT  21h                ;efetua escrita da mensagem
JC   ERRO_WRITE
...
```

Interrupção INT 21H – Função 42h

Move o ponteiro do arquivo.
Normalmente, quando um arquivo é aberto, o ponteiro é posicionado no início do arquivo.

Entradas: AH = 42h
AL = código de movimento
0 -> move em relação ao início do arquivo
1 -> move em relação à posição corrente do ponteiro
2 -> move em relação ao fim do arquivo
BX = file handle
CX:DX = quantidade de posições a ser movida (número sinalizado)

Saídas: Se bem sucedido, DX:AX = nova posição do ponteiro em bytes a partir do início
Se houver erro, CF = 1 e o código de erro está em AX (1, 6)

Exemplo: Escrever um trecho para determinar o tamanho de um arquivo

```
...
.DATA
HANDLE    DB    ?       ;file handle válido já definido
...
.CODE
MOV AX,@DATA
MOV DS,AX               ;inicializa DS
MOV AH,42h              ;função para mover ponteiro
MOV BX,handle           ;obtém file handle para o arquivo aberto
XOR DX,DX
XOR CX,CX               ;valor de DX:CX é nulo -> mover zero bytes
MOV AL,2                ;mover o ponteiro para o fim
INT 21h                 ;calcula a deslocamento final correspondente
                        ;ao tamanho do arquivo que estará no
                        ;conjunto DX:AX
JC  ERRO_MOVE
...
```

Interrupção INT 21H – Função 43h

Obtém/muda o atributo de um arquivo

Entradas: AH = 43h
AL = 0 para obter atributo
= 1 para mudar o atributo
CX = novo atributo para o arquivo, se AL = 1
DS:DX = endereço do nome do arquivo (e caminho) em string ASCIIZ

Saídas: Se bem sucedido, CX = atributo corrente
Se houver erro, CF = 1 e o código de erro está em AX (2, 3 ou 5)

Exemplo: Escrever um trecho para mudar o atributo de um arquivo para escondido (*hidden*)

```
...
.DATA
NOME            DB    'A:\TESTE\ARQ1.TXT'
...
.CODE
MOV AX,@DATA
MOV DS,AX           ;inicializa DS
MOV AH,43h          ;função atributo de um arquivo
MOV AL,1            ;opção de mudar atributo
LEA DX,NOME         ;aponta para o inicio do string NOME que é um path
MOV CX,1            ;atributo de arquivo escondido (verificar!)
INT 21h
JC  ERRO_ATRIB
...
```

INTERRUPÇÕES DO DOS QUE OPERAM DISCOS DIRETAMENTE :

Interrupção INT 25H - Usada para ler diretamente setores especificados de um disco.

Interrupção INT 26H - Usada para escrever diretamente em setores especificados de um disco

Entradas:

AL = 0 para drive A
= 1 para drive B, etc.
DS:BX = segmento:offset do buffer na memória (destino ou fonte)
CX = quantidade de setores a serem lidos ou escritos
DX = número lógico do setor inicial

Saídas: Se bem sucedido, retornar os FLAGS da pilha
Se houver erro, CF = 1 e o código de erro está em AX

Exemplo: Números lógicos dos setores num disco flexível de 5 ¼ double density:

Lado	Trilha	Setor	Setor lógico	Informação
0	0	1	0h	Registro de Boot
0	0	2 - 5	1h – 4h	FAT
0	0	6 - 9	5h – 8h	Diretório de arquivos
1	0	1 - 3	9h – Bh	Diretório de arquivos
1	0	4 - 9	Ch – 11h	Dados
0	1	1 - 9	12h – 1Ah	Dados
1	1	...	...	...

Obs.: SETORES são numerados a partir de 1; mas SETORES LÓGICOS são numerados a partir de 0.

EXERCÍCIOS:

- **EXERCÍCIO 1**: ESTUDE O PROGRAMA DE EXEMPLO FORNECIDO COM O NOME DE **PROG1.ASM**. PROCURE COMPREENDER SEU FUNCIONAMENTO E DAS SUBROTINAS NELE CONTIDAS. EXECUTE E TESTE SEU FUNCIONAMENTO.

  MODIFIQUE ESTE PROGRAMA PARA QUE TAMBÉM SEJA EXIBIDO NA TELA O TAMANHO DO ARQUIVO EM BYTES. USE MENSAGENS ADEQUADAS. TALVÉS VOCÊ TENHA QUE UTILIZAR UMA SUBROTINA PARA EXIBIÇÃO DE NÚMEROS DECIMAIS NA TELA, POIS O TAMANHO DO ARQUIVO SERÁ FORNECIDO EM BINÁRIO. VEJA A PÁGINA 168 DO LIVRO.

  DICA: CRIE UM PEQUENO ARQUIVO TEXTO NUM DISQUETE PARA SERVIR COMO TESTE DO PROGRAMA.

- **EXERCÍCIO 2:** ESCREVA UM PROGRAMA QUE LEIA UM ARQUIVO QUALQUER CUJO NOME SEJA DIGITADO NO TECLADO E CRIE UMA CÓPIA EXATAMENTE IGUAL, COM O MESMO NOME PORÉM COM EXTENSÃO **.BAK**, PARA DESIGNAR UMA CÓPIA DE BACK-UP.

- **EXERCÍCIO 3:** ESCREVA UM PROGRAMA QUE LEIA O SETOR LÓGICO NÚMERO 5 DE UM DISCO FLEXÍVEL, TAL COMO O PROGRAMA **PGM19_3.ASM** DA PÁGINA 416 DO LIVRO. PARA TESTAR ESTE PROGRAMA, FORMATE UM DISQUETE E GRAVE NELE APENAS UM PEQUENO ARQUIVO TEXTO COM O SEU NOME. PROCURE IDENTIFICAR TODA A INFORMAÇÃO DO ***FILE ENTRY***, INCLUSIVE CALCULE OS DADOS DE **HH:MM:SS** E **DD:MM:AA** DE SUA CRIAÇÃO.

  DICA: USE A **INT 25h**.

  OBS: INICIANDO A LEITURA NO SETOR LÓGICO 1 (FAT), SE PODE VERIFICAR O SEU CONTEÚDO E VERIFICAR COMO OS SETORES ESTÃO SENDO OCUPADOS PELO ARQUIVO. ESTA É UMA APLICAÇÃO **MUITO IMPORTANTE** E SERÁ USADA NO FUTURO EM OUTRAS DISCIPLINAS. LEIA AS PÁGINAS 417 E 418 DO LIVRO PARA ENTENDER COMO SE DEVE PROCEDER.