DOTE - UM EDITOR DE TEXTOS ORIENTADO PARA VÍDEO

SILVIA HELENA MACHADO DE OLIVEIRA

ORIENTADOR

Profº Dr. Tomasz Kowaltowski

Dissertação apresentada no Instituto de Matemática, Estatística e Ciência da Computação, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

DEZEMBRO - 1982

RESUMO

O trabalho apresentado é relativo a definição e implementação de um editor de textos voltado para terminal de vídeo (DOTE) para o sistema COBRA-400.

Primeiramente faz-se um estudo sobre editores de texto já existentes; temos um apanhado geral dos mais representativos.

São destacadas, também, as características do sistema COBRA-400 relevantes à implementação do editor.

A seguir são discutidas as facilidades propostas para o editor.

Também é descrito a estrutura de dados e as operações primitivas para que se possa implementar os comandos.

Finalmente há sugestões de ampliaçao para funcionamento para os 4 terminais disponíveis no sistema (o sistema atual é monoprogramado).

Os Apêndices trazem o manual do usuário de DOTE bem como: lista de constantes e variáveis mais importantes do editor, exemplos de rotinas em linguagem de montagem do INTEL 8080.

ABSTRACT

We present in this dissertation the definition and the implementation of a display oriented text editor (DOTE) for the COBRA-400 System.

Initially we present a survey of existing text editors, including the most significant ones.

We describe then the characteristics of the COBRA-400 system which are relevant for this implementation.

Next we introduce the main features of our editor.

The implementation of the editor commands is described through its data structures and primitive operations.

Finally we present suggestions for transforming the editor into multiprogrammed system in order to include the four available terminals (presently DOTE is monoprogrammed).

Appendices include the DOTE user's manual, a list of the most important constants and variables used in the implementation, and examples of some routines written in the INTEL 8080 Assembly Language.

ÍNDICE

CAPÍTULO V

CAPÍTULO VI

CAPÍTULO I

1 - INTRODUÇÃO

1.1 - Generalidades

A maioria dos editores de texto são orientados para linha, no sentido de se ter acesso direto a apenas uma única linha. Esse tipo de editor é apropriado para terminais do tipo máquina de escrever, mas quando estamos em um terminal de vídeo devemos usar o fato de visualizarmos várias linhas e de termos acesso direto (através de controles do cursor) à tela toda.

Em um editor de textos convencional o usuário visualiza eventuais trechos de arquivo misturados com comandos para o editor, correção de comandos para o editor, mensagens de erro e outros caracteres mais. Um editor de textos orientado para vídeo deve mostrar a maior parte de texto possível a um dado momento, e este texto mostrado deve refletir o que está correntemente no arquivo, com o menor número possível de caracteres estranhos ao mesmo. Mensagens de erro e comandos para o editor devem estar em lugar separado do texto.

É muito comum para um usuário de editor de textos convencional o comando "**lis**te N linhas a partir da linha atual", este tipo de comando não é absolutamente necessário em um editor de textos voltado para vídeo: a linha atual e suas vizinhas estão automaticamente na tela.

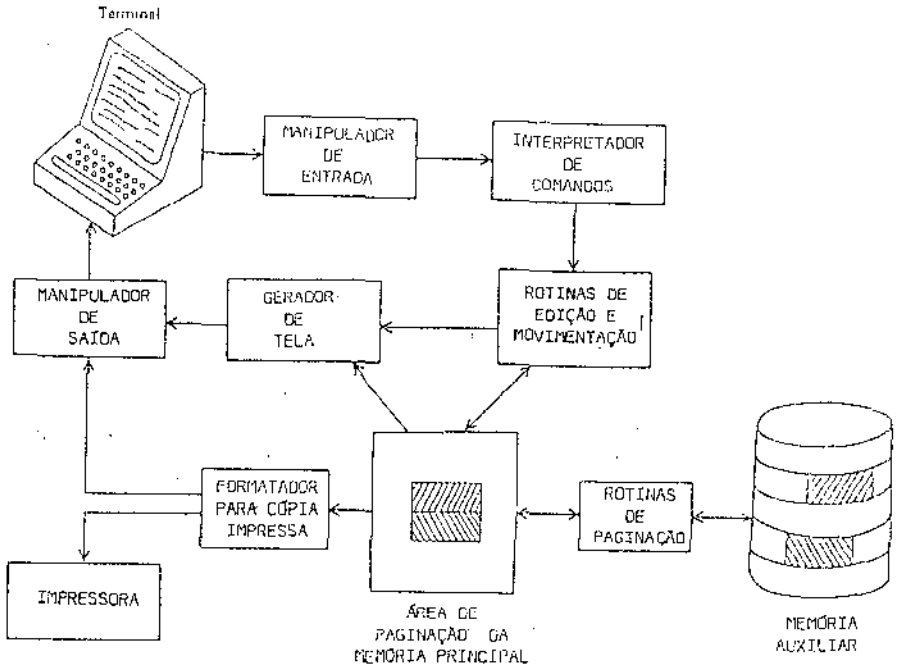
Mesmo o comando de substituição local em um editor convencional é muito usado para se definir corretamente a que parte do texto o usuário quer se referir e nem sempre com sucesso. Em um editor voltado para vídeo o usuário localizaria visualmente o trecho de texto pretendido através de posicionamento do cursor e então reescreveria o texto. Pela visualização automática da linha atual e sua vizinhança, diminui consideravelmente a probabilidade de erro.

Partindo-se para uma visão mais geral, nota-se que o uso de computadores está cada vez mais saindo do reduto dos "iniciados na arte" para se popularizar cada vez mais. A tendência de se facilitar ao máximo a comunicação usuário/máquina é necessária e pode-se mesmo especular que logo a maioria dos sistemas operacionais e mesmo algumas linguagens (vide "EMILY" seção 1.2.5 deste trabalho) serão inteiramente orientadas para terminal de vídeo. Não há motivo para não se usar facilidades de movimentos do cursor e reescrita de textos para entrada de dados, comandos para o sistema operacional e em todo tipo de comunicação usuário/máquina.

1.2 - Características de alguns editores

A fim de se ter uma idéia do que já havia sido desenvolvido na área de edição de textos foram consultados vários artigos sobre tal assunto. Os mais significativos serão aqui mencionados.

A figura a seguir mostra, de uma maneira geral e simplificada, o fluxo funcional da maioria dos editores de texto.



O usuário está sentado em frente a um terminal de vídeo, ou tipo máquina de escrever, onde é visível uma ou mais linhas de um arquivo. Os comandos são fornecidos via teclado (ou chaves, "light pen",...), e então a interrupção gerada é tratada pelo "MANIPULADOR DE ENTRADA" e mandada ao "INTERPRETADOR DE COMANDO". O comando, já em código interno, é passado para a "ROTINA DE EDIÇÃO" que realmente faz a operação pedida. A forma interna do texto pode ser alterada e o "GERADOR DE TELA" reformata a saída para o terminal de vídeo. O "FORMATADOR P/ CÓPIA IMPRESSA", (impressora, fotocomposito-ra, etc) faz o mesmo.

O usuário quer:

- resposta rápida para um grande número de terminais.
- comandos concisos e mnemônicos.

- comandos poderosos com pequenas restrições e exceções.
- comandos que usam as capacidades do computador

  Ex: - procurar a 1ª ocorrencia de um padrão.
  - substituição uniforme de um padrão por outro
    - sempre
    - na 1ª ocorrência
    - só se estiver numa dada coluna

O método de armazenamento e as funções de edição dependem se o editor é voltado para programas ou texto. Também as funções dependem do tipo de terminal.

1.2.1 - CMS

CMS [16] foi desenvolvido para o IBM 360/67 com sistema operacional CP/CMS.

A maneira de se armazenar o texto depende se é programa (registros de tamanho fixo: 80 caracteres) ou não (registros de tamanho variável: no máximo 130 caracteres). Na memória principal é como se todos os registros tivessem 130 caracteres, quando necessário preenche com brancos. Como o computador usado tem memória virtual o arquivo sendo editado é mantido "todo" na memória principal, não sendo necessário que o editor trate com entrada/saída para memória secundária; o próprio sistema operacional já faz isto.

Há duas maneiras de se modificar um programa. O modo de "entrada" quando o novo texto é contínuamente teclado uma linha por

vez. É usado para criar programas ou inserir várias linhas contíguas num programa. O modo de "edição" quando pode-se apagar uma ou mais linhas, uma linha pode ser modificada ou mesmo inserir caracteres em uma linha, trocar uma cadeia de caracteres por outra, não necessariamente do mesmo tamanho. Quando o tamanho máximo da linha é alcançado os caracteres a mais são truncados. (Se essa restrição não é muito importante quando o texto é um programa passa a ser desastrosa quando o texto é livre). Há dois tipos de comando de busca:

- busca uma cadeia de caracteres começando numa dada coluna.
- busca uma cadeia de caracteres em qualquer lugar da linha.

1.2.2 - WILBUR

Wilbur [16] foi implementado em um IBM 360/67 e usa os terminais 2741; e é orientado para linha. Um arquivo é dividido em linhas de Ø a 133 caracteres, mas normalmente, programas tem no máximo 72 caracteres. Para cada linha armazena nº da linha e comprimento.

Geralmente WILBUR opera sobre um intervalo de linhas. Esse intervalo é definido como toda linha que tenha uma característica comum. Exemplos de especificação de intervalo:

explícita
- 3/12 ... as linhas 3,4,5,6,7,...,12
- 4,7,9 ... as linhas 4,7,9

associativa { "< cadeia >" ... todas as linhas que contêm < cadeia >

ambos { "CAO" IN 6/10 ... todas as linhas que contêm "CAO", entre as linhas 6,7,8,9,10

Os comandos "DELETE", "MOVE", "CHANGE" e "COPY" trabalham com intervalo de linhas. O comando "MODIFY" trabalha numa dada linha. A linha é impressa no terminal e o usuário tecla caracteres especiais embaixo da linha impressa modificando aquela linha. Quando uma linha excede o tamanho máximo de linha é truncada.

Tem facilidades para interagir com o sistema: pode mandar compilar um programa, listar o estado de um programa em execução.

1.2.3 - "TEXT EDITOR and CORRECTOR (TECO)"

TECO [16] corre em um PDP1Ø. Segundo o autor, os comandos de TECO são primitivos, no entanto esses comandos são usados em blocos para determinar operações elaboradas de edição.

Além dos comandos usuais de edição (SEARCH e SUBSTITUTE, por exemplo) tem ainda um comando de "execução condicional", muito útil para fazer programas de edição.

TECO não é orientado para linha, mas sim para caracteres. Um arquivo é uma sequência de caracteres normalmente quebrada em páginas, o fim de uma página é representado internamente pelo caráter "FORM FEED".

Num dado momento pode-se editar o que está no "buffer" cujo tamanho normal é equivalente ao de uma página de um terminal tipo máquina de escrever. O usuário pode aumentar o tamanho deste "buffer". Depois que um "buffer" foi editado é escrito sequencialmente em outro arquivo e a próxima página é trazida do arquivo que está sendo editado. Por essa sequencialidade não é permitido voltar para uma página anterior; deve-se reler o arquivo novo do começo.

A posição corrente no "buffer" é determinada por um apontador que aponta no meio de dois caracteres. Esse apontador pode ser posicionado por uma busca de cadeia ou por posicionamente absoluto ou relativo de linha (relativo à "linha atual"). Existem registradores onde se podem efetuar contas aritméticas ou copiar partes do texto. Para se tirar parte do texto de um lugar e colocar em outro basta salvar o texto desejado num registrador e apagá-lo do "buffer" então reposicionar o apontador para o lugar desejado e copiar o conteúdo do registrador no "buffer".

Exemplo de "programa" para inserir uma linha com '*****' antes da linha que contém a cadeia "EXEMPLO". Isso é feito para todas as linhas até que se encontre um "buffer" vazio ou um "buffer" onde não ocorre a cadeia; nesse caso insere-se no início do "buffer" a mensagem "NENHUM *EXEMPLO* NESTE BUFFER" e não procura mais no resto do arquivo.

```
!START!ØUA$<SEXEMPLO$;ØL$I*****
$%A$L$>QA"NPZ"NOSTART$'OFIM$'
INENHUM*EXEMPLO*NESTE BUFFER$!FIM!$$
```

1.2.4 - TVEDIT

TVEDIT [16] foi desenvolvido na Universidade de Stanford em 1965.

Este editor foi projetado para um computador com tempo compartilhado e com terminais de vídeo. Mostra várias linhas do texto, assim o usuário vê continuamente a mais recente versão do seu texto. O tamanho máximo de uma linha é de 128 caracteres. Como o tamanho da linha do terminal é de 50 caracteres, linhas com mais de 50 caracteres aparecerão truncadas com uma marca indicando isto.

Os comandos de TVEDIT são fornecidos através da pressão das teclas "CONTROL" e uma outra tecla qualquer. Se antes de um comando for teclado um número o comando será repetido aquele número de vezes. Os comandos são executados à medida que são teclados, não havendo necessidade do usuário pressionar nenhuma tecla especial para indicar fim de cadeia de comando.

Há um apontador na tela que tanto pode servir para identificar uma linha como estar entre dois caracteres de uma linha.

Para se alterar partes do texto basta teclar o novo texto. Linhas ou caracteres são apagados movendo-se o apontador convenientemente e teclando-se um caráter de controle. Quando um caráter é apagado do texto todos os caracteres seguintes a ele são movidos uma posição à esquerda. Quando um caráter é inserido acontece a mesma coisa só que para a direita.

Não existe um comando explícito para procurar uma cadeia de caracteres, isso deve ser feito visualmente.

1.2.5 - EMILY

Em todos os editores vistos até agora, o usuário entrava com o texto caráter por caráter. Com Emily [16] o usuário cria e altera programas selecionando, através de uma "light pen", entre várias alternativas fornecidas pelo sistema de acordo com a descrição sintática da linguagem, no caso, PL-I.

O usuário interage com Emily através de um terminal gráfico IBM 2250 ligado a um IBM/360. A tela do terminal é assim dividida:

Em TEXTO aparece o texto já selecionado,
em MENU aparece o conjunto de possibilidades para o não terminal corrente
em MENSAGEM tem mensagens de erro, estado e também serve para a entrada de identificadores.

Não-terminais são sublinhados e o corrente é marcado com um retângulo.

Exemplo do que acontece em TEXTO durante a criação de um comando IF:

1) [<STMT>]
2) IF [<EXPR>] THEN <STMT>
3) IF [<VAR>] THEN <STMT>
4) IF FIRST-TIME THEN [<STMT>]

```
5)  IF  FIRST-TIME  THEN  DO;
        <STMT >
    END;
6)  IF FIRST-TIME THEN DO;
        FIRST-TIME = FALSE;
    END
```

quando em TEXTO temos IF <VAR> THEN <STMT> em MENU temos os nomes das variáveis definidas e acessíveis nesse momento.

Cada seleção do MENU gera um nó com espaço para um apontador para cada não terminal na cadeia. Quando um não terminal é substituido o espaço correspondente é preenchido com um apontador para o nó gerado por aquela substituição.

O usuário pode mudar sua visão do texto, assim a cadeia gerada por um não terminal pode ser representada por um único símbolo chamado "holophrost" (abreviatura). Por exemplo ao invés do que aparece na linha 6) pode ser mostrado o que aparece na linha 4). Em programas grandes isso é bom para se ver a estrutura do programa sem se preocupar com detalhes e também pode-se descer na estrutura e ver somente alguns detalhes.

As críticas a este editor são:

- dificuldade de uso da "light pen"; é lento (precisa esperar escrever o MENU todo) e incômodo.
- dificuldade para escolher um identificador, a sua posição no MENU vai sendo alterada a medida que são acrescidos nomes de variáveis.

1.2.6 - HES e FRESS

HES e FRESS [16] correm em IBM 360 usando terminais IBM 2250, 2260 ou outros mais poderosos.

O primeiro, HES, é um sistema flexível e foi baseado em terminal gráfico (IBM 2250) fornecendo comandos para edição e formatação de textos. Os comandos são fornecidos através de "light pen" e teclas especiais de função. Inicialmente aperta-se a tecla de função desejada então com o auxílio da "light pen" indica-se a porção de texto ao qual a função se aplica. Por exemplo para se apagar parte do texto pressiona-se a tecla "DELETE" com a "light pen", indica--se o início e o fim da parte a ser apagada e então o sistema vai colocar brancos na região do texto afetada. Se estiver correto pressiona-se uma tecla para realmente modificar o texto.

Entre as funções de edição e formatação estão: insira, apague, substitua, rearranje e copie. O sistema dá mensagens especificando a cada momento o que é disponível. Produz texto com letras maiúsculas e minúsculas.

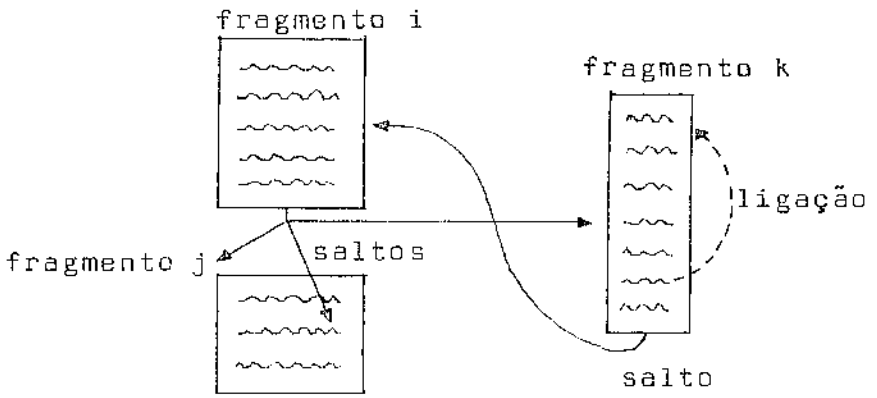
A estrutura de dados e as operações de edição são completamente independentes do que está na tela ou na impressora. O usuário divide o texto em fragmentos chamados "áreas de texto". Essas áreas podem ser interligadas e com referências cruzadas. Existem dois tipos de referência cruzada:

— Saltos: desvios incondicionais entre dois fragmentos que forçam o usuário a, através da "light pen", escolher onde prosseguir (exemplo típico: MENU).

— ligações: desvios condicionais que o usuário pode acionar, através da "light pen", ou não (exemplo típico: RODAPÉ).

O resultado móvel dos fragmentos de texto é chamado hipertexto. Um exemplo prático de hipertexto pode ser uma enciclopédia.

O sistema lembra a sequência de saltos e ligações que o usuário escolheu, permitindo ao usuário percorrer o caminho inverso. Para se ter acesso aleatório ao texto deve-se usar rótulos e depois, através da "light pen", saltar para o rótulo.



HES foi um sistema experimental a partir do qual surgiu FRESS que não tem restrição a tamanho de cadeia para edição, tem busca por conteúdo, saída para fotocompositora, edição entre arquivos e proteção de arquivos e blocos de texto através de senhas.

1.2.7 - EDITOR DE IRONS E DJORUP

Foi implementado [11] em um CONTROL DATA 6600, com o sistema "time-sharing" IDA-CRD, por volta de 1972.

É um sistema de edição de textos que usa terminal de vídeo com

teclado normal e treze teclas com funções especiais. Tem facilidades para se editar vários arquivos numa só sessão e conexão com o resto do sistema. Pode-se compilar o programa sendo editado ou executar um outro tendo como entrada o arquivo sendo editado.

Para se corrigir partes do texto basta posicionar o cursor no lugar desejado e bater novamente o texto correto. Essas funções de posicionamento do cursor são feitas independentemente do computador e as alterações feitas estão somente na tela. Para que o arquivo seja modificado é preciso que seja explicitamente dado um comando. Para se inserir dados deve-se inicialmente "abrir" espaço e depois agir como se se estivesse alterando um texto. A movimentação no arquivo parece ser inteiramente permitida tanto para avançar como para recuar.

1.2.8 - TYPER

TYPER [14] foi projetado para rodar em qualquer computador DGC com sistema NOVA ou ECLIPSE. Foi escrito em um superconjunto do ALGOL 60, por volta de 1976.

A idéia de TYPER surgiu da necessidade do departamento de publicações da DATA GENERAL CORPORATION para produção e atualização de manuais. Mostra a inconveniência da maioria dos editores para pessoas que não são da área de computação:

- difíceis para quem quer aprender só para usar o editor.
- complicados para se fazer o desejado.

Seu objetivo principal é a simplicidade para a produção e atualização de um texto por um não-programador. Foi projetado para composição imediata ("on line") do texto editado. Tem dois conceitos centrais:

— o que está na tela é o que está na memória.
— "fechamento do texto". Reformata o texto pondo tantas palavras completas quanto possível em cada linha é, portanto, não só um editor de textos como tem também características de um formatador de textos.

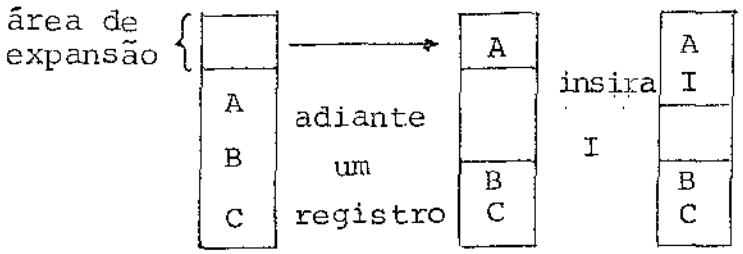
Usa as 20 linhas de um terminal de vídeo como uma "janela" no texto do usuário. Tem um cursor que se move na janela e serve como apontador para as modificações a serem feitas no texto.

Tem comandos para inserção e eliminação de caracteres, eliminação de cadeia de caracteres e reescrita de texto. O autor sentiu a necessidade de poder voltar no arquivo todo, comando de busca de cadeia de caracteres e também mover blocos de uma parte para outra do arquivo.

1.2.9 - SITAR

SITAR [13] foi implementado, por volta de 1976, em um PDP-11 com sistema "time-sharing" e com terminais de vídeo com função de edição microprogramadas. É um sistema para manipular e analisar textos, também visando os usuários não especializados em computação.

Combina os poderes de comandos simples, orientação para cadeia, estrutura de arquivo circular, terminal de vídeo com memória local e processamento interativo. O texto é considerado como uma cadeia contínua de caracteres. Para evitar a criação de um arquivo temporário SITAR usa o arquivo circular com uma área de expansão no início que se move no arquivo juntamente com a edição de texto. Exemplo:

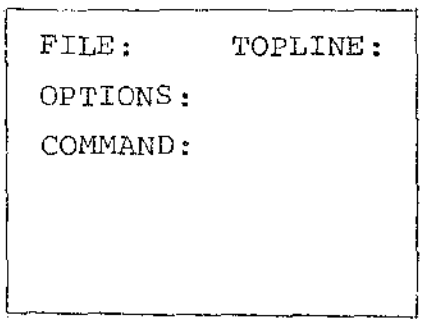


Com isso andar no arquivo para frente ou para trás ("voltar" no arquivo) é facilmente implementado. Tem facilidades para se editar vários arquivos numa só sessão, busca de partes do texto que começam com uma cadeia e terminam com outra (SEARCH CADEIA1...CADEIA2) e substituição de cadeias de caracteres.

1.2.10 - GODOT

GODOT [8] corre sobre o sistema operacional UNIX de um sistema de PDP-11.

GODOT foi projetado para terminal de vídeo com cursor endereçável. O texto pode ser entrado diretamente na tela, apagando, re-escrevendo ou inserindo caracteres. As três primeiras linhas do terminal são reservadas ao editor.



Tem endereçamento absoluto (o número da linha só é atualizado no fim da sessão de edição), relativo ao cursor e relativo ao fim do arquivo.

Tem dois tipos de comando: simples e o de linha. Os simples são fornecidos através de caracteres de controle e normalmente não tem argumentos. Entre os comandos de linha estão SEARCH FORWARD e BACKWARD, MOVE e COPY.

1.2.11 - VED

A noção básica deste editor, Ved, é que o que normalmente se faz é entrada e revisão de texto. Ved [7] foi implementado, em 1982, na linguagem Pascal e corre em um PDP-10.

A tela tem 2 linhas reservadas para o editor e todo início de linha tem antes o endereço da linha. Uma das linhas é para mensagem de erro. A outra é para o editor pedir argumentos, etc. O endereço da linha não faz parte do arquivo. Cada linha da tela mostra uma linha do arquivo; se a linha do arquivo for maior do que a da tela posiciona-se o cursor no último caráter mostrado e tecla-se o comando que move o cursor para a direita. Repete-se quantas vezes necessário. Os comandos são fornecidos através de teclas es-

peciais e caracteres de controle. Um mesmo comando pode ter significados distintos caso o cursor esteja no texto ou no endereço da linha.

1.3 - Características gerais do sistema COBRA-400

A configuração do sistema COBRA-400 disponível na UNICAMP consta do seguinte:

64 K bytes de memória principal

Unidade Central de Processamento - UCP (microprocessador INTEL 8080)

4 terminais de vídeo

1 unidade de disco fixo de 5,3 M bytes

1 unidade de disco flexível

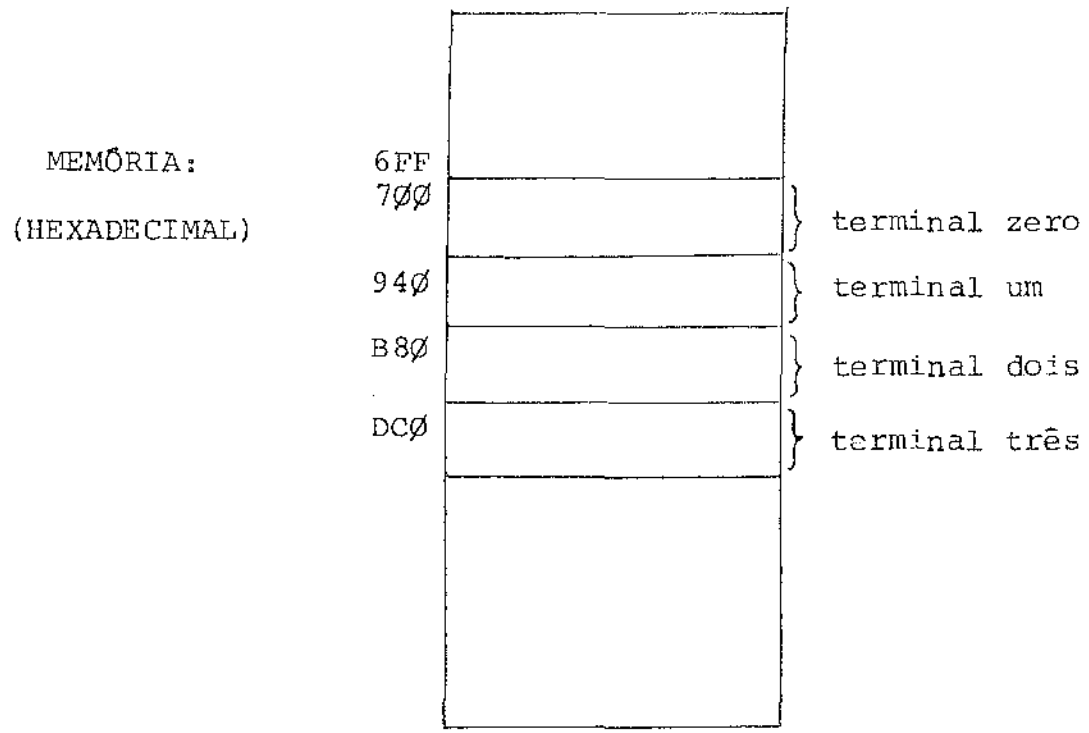
1 unidade de fita magnética

1 impressora serial

O terminal é constituído de uma tela e um teclado.

A tela possui nove linhas endereçáveis, sendo que a primeira delas é separada das oito restantes pelo espaço equivalente a uma linha em branco. Aproveitando essa característica, a primeira linha será reservada para o editor e será chamada de linha de controle. Nas futuras referências ao tamanho da tela serão consideradas apenas oito linhas. O texto visível no vídeo é imagem direta de posições de memória, o que equivale a dizer que por e tirar caracteres da tela consiste em por e tirar caracteres de uma região pré-

-fixada de memória.



O teclado (figura 1) possui todos os caracteres alfabéticos, numéricos e os seguintes caracteres especiais:

! ? . , ; : @ & # + - * / = < > $ % ( ) [ ] ' "

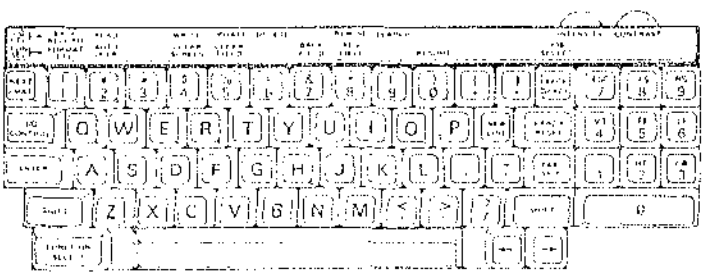


Figura 1

Tem as seguintes teclas auxiliares:

(estas só fazem sentido se pressionadas juntamente com uma tecla não-auxiliar)

SHIFT , I/O CTRL (*), FUNCTION SELECT

Possui ainda as seguintes teclas especiais:

ENTER, NEWLINE, ERROR RESET, ← , → , NEXT FMAT (*), BACK SPACE, TAB/ SKIP, ESC (*), FS (*), RS (*), VT, FF (*), LF (*), HT, CR (*).

Cada vez que uma tecla é pressionada seu código é gerado e ocorre uma interrupção no sistema. O código gerado é do tamanho de um "byte" de oito "bits", onde os dois "bits" mais significativos indicam se havia uma das teclas auxiliares pressionadas ou não e os seis restantes indicam a posição da tecla no teclado. Quando a rotina que trata das interrupções detectar uma interrupção (**) do teclado deve pegar o dado, decodificá-lo e convertê-lo no caráter ASCII apropriado se possível.

O mecanismo de interrupção do COBRA-400 é o seguinte:

quando um pedido de interrupção é aceito, a UCP inibe interrupções, desvia para a posição 8 da memória principal e salva o contador de instruções. Na posição 8 de memória temos então o desvio para a rotina de interrupções.

(*) Esta tecla não será considerada como tecla especial pelo editor. (NÃO CAUSARÃO NENHUM EFEITO COLATERAL)

(**) A rotina de interrupção nunca é interrompida.

O COBRA-400 dispõe de um relógio de tempo real, não programável, que gera uma interrupção a cada 50 ms. O sistema fornece repetição automática de teclas, que é feito por programa durante o tratamento da interrupção do relógio.

O disco fixo [3] é controlado por um micro computador INTEL 8080 que dispõe de memória particular independente da memória principal. A comunicação entre a UCP e o controlador de disco é feita através de comandos enviados ao controlador em uma região pré-fixada da memória principal: o DCB - "DISK CONTROL BLOCK". O DCB está localizado nas posições de 40H a 4CH, e seu formato é dado a seguir:

endereço (hexadecimal)	conteúdo
40	comando
41	"STATUS"
42-43	endereço do "buffer"
44-45	endereço do setor do disco
4 A	nº de setores a serem transferidos
4 B	nº de setores alternativos restante
4 C	"CHECKSUM"

O "buffer" acima referido deve se localizar nos primeiros 32k da memória principal, que correspondem aos últimos 32k da memória particular do controlador de disco.

Em "STATUS" temos o estado do controlador: se já terminou ope

ração, se houve erro e o tipo do erro.

Atualmente o controlador de disco interrompe a UCP quando do término de uma operação de entrada/saída, porém tal facilidade não pode ser utilizada por não se conhecer completamente o mecanismo de tratamento dessa interrupção.

O problema de E/S em disco foi resolvido utilizando-se "STATUS". Essa posição é contínuamente testada pela UCP ("busy wait").

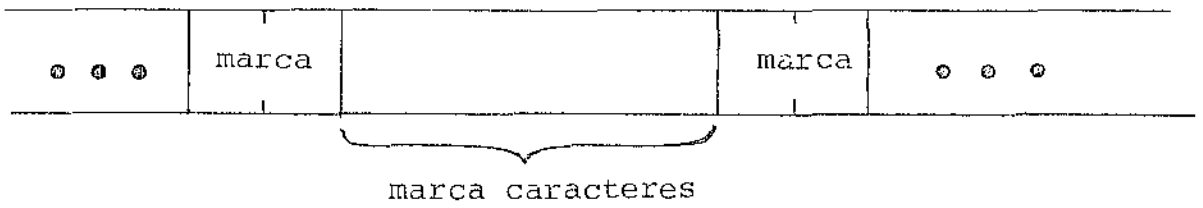
O disco fixo é organizado em setores de 512 "bytes". Quando se faz leitura ou gravação em disco isso é feito para um número inteiro de setores.

Os dados que estão em disco estão organizados em forma de arquivos. Cada arquivo, por sua vez, é dividido em registros. Esses registros podem ter tamanho fixo ou variável. No momento da criação de um arquivo é especificado se seus registros serão de tamanho fixo ou não, caso sejam de tamanho fixo deverá também ser especificado o tamanho do registro.

O tamanho máximo de um registro é de 511 "bytes", que equivale a uma tela do terminal menos um caráter, que é usado para a visualização do separador de registros (■) na tela. Esse separador de registros não faz parte do arquivo.

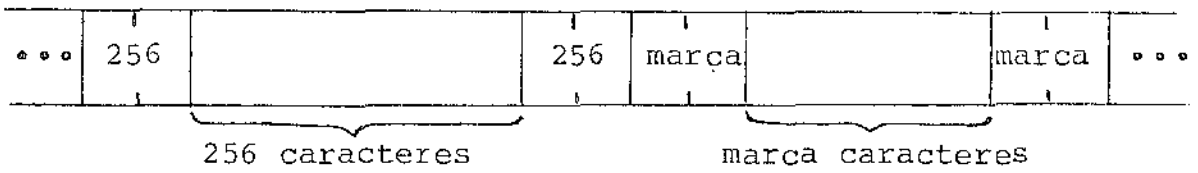
O primeiro registro de um arquivo deve começar no início de um setor. Para os demais registros não existe essa exigência; eles serão colocados nos setores sequencialmente preenchendo todos os espaços disponíveis.

Cada registro tem uma marca de início e fim de registro. Se o registro tem até 255 caracteres é armazenado da seguinte maneira:



onde marca é o número de caracteres do registro.

Se o registro tiver mais do que 255 caracteres será armazenado do seguinte modo:



onde marca é igual ao número de caracteres do registro menos 256. (marca pode ser zero).

Cada marca ocupa sempre dois "bytes".

Os setores de 0 a 1 do disco contém o carregador de mini disco. Nos setores de 3 a 32 está o "HEADER" do disco. Os setores de 32 a 83 contém as entradas do diretório dos arquivos em disco. Nos setores restantes estão os arquivos e espaços livres.

Para cada arquivo em disco temos uma entrada no diretório de arquivos. Uma entrada no diretório contém as seguintes informações:

posição (hexadecimal)	conteúdo
0	tamanho da entrada (64 caracteres)
1-8	nome do arquivo
A-B	tipo do arquivo
C-D	nº do setor inicial do arquivo (BOE)

E-F	nº do setor final do arquivo (EOE)
10-11	nº do último setor lido ou gravado (CDA)
12-13	deslocamento do último setor lido ou gravado (CDAR) (em "bytes")
14-15	nº do primeiro setor com espaço disponível (NAR)
16-17	deslocamento no primeiro setor com espaço disponível (NARD)
18-19	tamanho do registro

Existem quatro tipos de estrutura de arquivos disponíveis:

— sequencial
— indexado
— circular
— aleatório

— arquivos sequenciais

Podem ter registros de tamanho fixo ou variável. Os registros são gravados um após o outro e lidos na ordem em que foram gravados.

— arquivos indexados

Cada registro tem uma única chave associada a ele. As chaves com os endereços dos registros são armazenados em um arquivo separado em disco. Os registros podem ser de tamanho fixo ou variável.

A quantidade de espaço em disco necessária para cada registro de índice é de (20 + 2 x (tamanho de chave)) "bytes".

Os arquivos indexados não podem ser criados a partir do teclado. Somente através de utilitários específicos do sistema ou um programa especial de controle de dados pode-se criar tais arquivos.

— Arquivos circulares

Usado pelo sistema para armazenar dados temporariamente. Quando um registro é lido o espaço ocupado por ele torna-se disponível para um novo registro.

— Arquivos aleatórios

Normalmente são usados em COBOL. A cada registro está associado um endereço em disco. Os registros devem ter tamanho fixo.

Se um arquivo sequencial tem registros de tamanho fixo pode-se usá-lo como aleatório em COBOL, apenas associando-se uma "ACTUAL KEY" a ele.

Existem dois utilitários para se criar e alterar arquivos: DE e EDIT. Em ambos é mostrado um registro de cada vez.

Com DE pode-se fazer entrada de dados e alterações em um registro caso estas alterações não modifiquem o tamanho do registro. DE altera o próprio arquivo. Não se consegue inserir um registro no meio de um arquivo usando-se DE, as inserções são feitas sempre no fim do arquivo. Somente com o uso de DE pode-se fazer entrada de dados com formato fixo e com consistência dos dados na entrada: se é alfabético, se é numérico, se está dentro de uma certa faixa etc. Pode-se avançar ou recuar livremente no arquivo.

Com EDIT pode-se criar e alterar arquivos sem restrições de tamanho do registro. EDIT usa um arquivo para entrada e outro para saída. Pode-se inserir registros no meio do arquivo, mas não se pode voltar no arquivo.

1.4 - Roteiro da tese

No capítulo 1 temos uma breve visão de editor de texto voltado para vídeo, um apanhado de alguns editores de propósito geral e as características do sistema COBRA-400 que nos interessa.

No capítulo 2 estão as características do editor proposto. Este será melhor compreendido com o auxílio do anexo 2 onde está definido DOTE.

Os capítulos seguintes referem-se à implementação do editor. O capítulo 3 mostra a estrutura de dados proposta para suportar o editor.

O capítulo 4 mostra as operações primitivas referentes à estrutura de dados.

No capítulo 5 temos os algorítmos que descrevem a implementação dos comandos do editor.

No capítulo 6 temos quais os comandos do editor estão em funcionamento atualmente e sugestões para futuras ampliações.

oooo°oooo

CAPÍTULO II

2 - CARACTERÍSTICAS DO DOTE

(Display Oriented Text Editor)

DOTE é um editor de textos orientado para vídeo. Foi, portanto, projetado para permitir edição direta de todo o texto visível na tela de um terminal.

O cursor pode ser diretamente posicionado no lugar do texto onde as alterações serão feitas. Além dessas facilidades, tem a vantagem, como já foi citado, de que automaticamente é mostrado o contexto no qual as correções serão feitas.

DOTE pretende também conciliar os dois seguintes fatos:

— apresentação de várias facilidades para o usuário.
— simplicidade de uso.

que o tornam satisfatório tanto a profissionais de programação de computadores como a pessoas que usam o computador somente como ferramenta auxiliar.

De uma maneira geral a filosofia de DOTE é a seguinte:

A tela é uma "janela" através da qual se vê o arquivo; portanto deve refletir o mais fielmente possível a imagem do arquivo.

Como o tamanho da linha do terminal é de 64 caracteres e o

tamanho máximo do registro é de 511 caracteres serão utilizadas tantas linhas quantas forem necessárias para mostrar um registro inteiro. Para que se visualize o fim do registro é usado o caráter: "■". Este é o único caráter que aparece na janela e não está realmente no arquivo.

Sempre que possível, o cursor apontará para um caráter do arquivo. Por exemplo se a janela for a seguinte:

```
AQUI ESTÁ
UM EXEMPLO■
DE ■
TEXTO ■
```

onde "_" é o cursor,

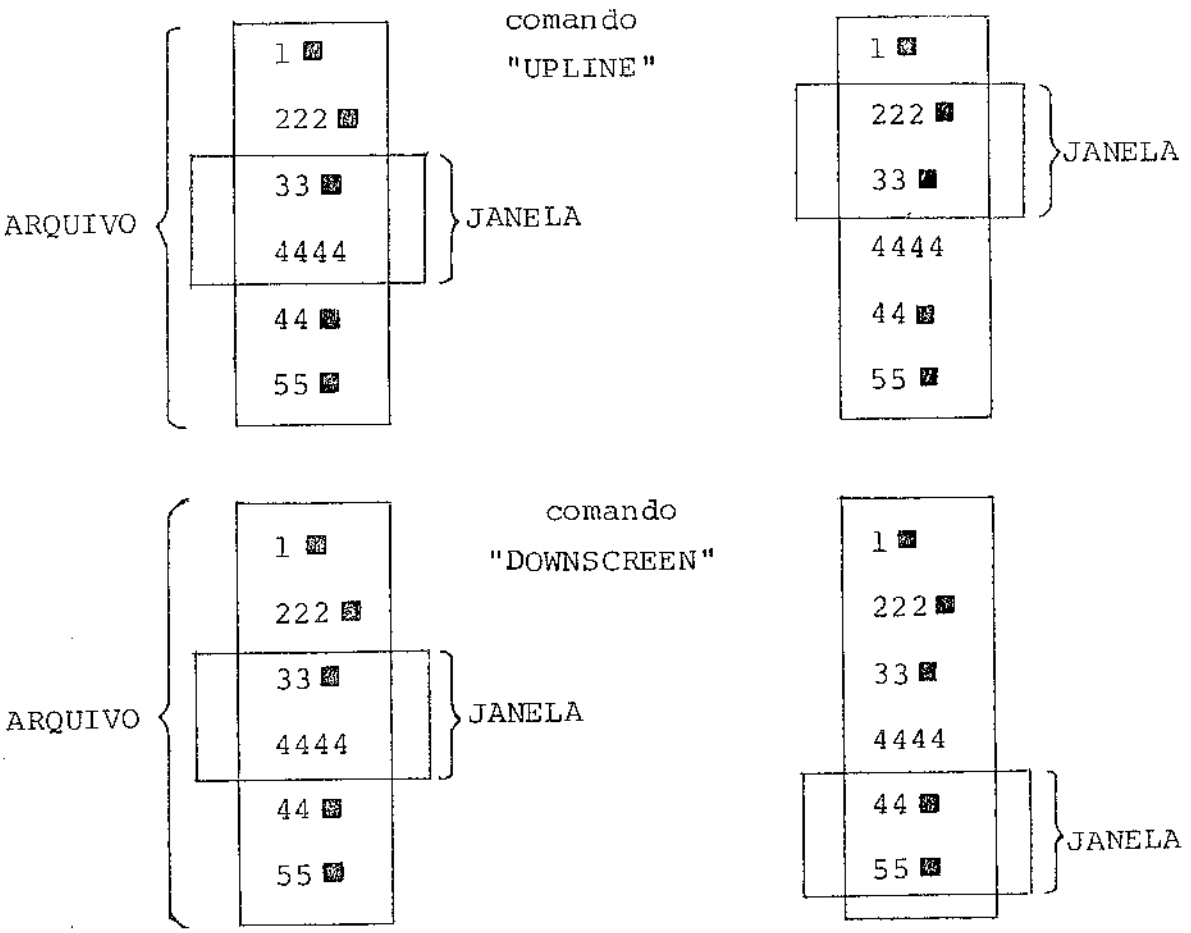
e for pressionada a tela ← (volte um caráter) a situação da janela passará a ser:

```
AQUI ESTÁ
UM EXEMPLO ■
DE ■
TEXTO ■
```

Como a inserção é feita à esquerda do cursor precisa-se ter acesso ao "■" caso se queira inserir caracteres no fim do registro. Isso é feito usando-se a tecla → (avance um caráter) na situação do exemplo anterior, que passaria a ser:

```
AQUI ESTÁ
UM EXEMPLO ■
DE ■
TEXTO ■
```

A janela pode se mover pelo arquivo tanto linha a linha (efeito de se "enrolar" o texto) como janela a janela. Assim, por exemplo:



Para melhor conseguir implementar a filosofia de DOTE foram usados dois tipos de comando:

— comandos imediatos

— comandos normais

A um dado momento ou se está em estado de comando normal ou em estado de comando imediato. Na linha de controle do editor tere

mos uma indicação de qual o estado atual do editor ("IMMEDIATE" ou "*").

Os comandos imediatos não tem parâmetro (com exceção do comando "INSERT") e são fornecidos através da pressão das teclas FUNC SELECT, representada por ↑, juntamente com uma outra. Quando um comando imediato é fornecido, este não é ecoado no terminal, apenas são tomadas as ações necessárias. Todos os comandos existentes no modo imediato são também disponíveis no modo normal.

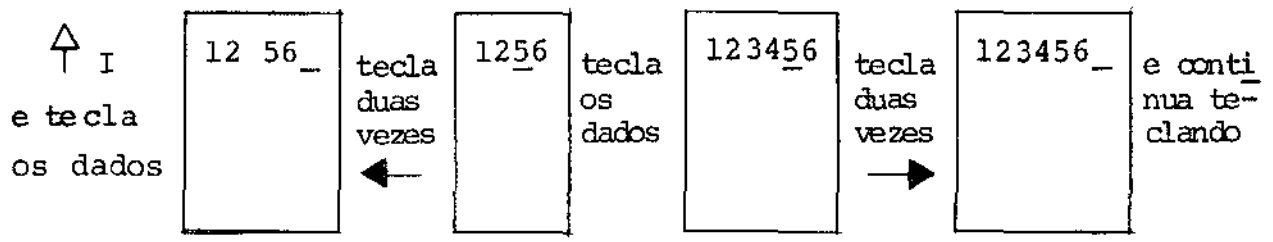
Os comandos normais podem ter parâmetros e são fornecidos ao editor via teclas normais. Os comandos teclados só serão executados após um comando explícito para tal; enquanto este não for fornecido o texto entrado será ecoado na linha de controle. Por essas características recomenda-se que as inserções sejam feitas através de comando imediato pois será imediatamente visível o efeito causado no arquivo.

Usou-se a tecla "TAB/SKIP" como indicador de fim de registro, no lugar de "LINE FEED" por exemplo, por ser de fácil acesso no teclado. Pelo mesmo motivo usou-se a tecla "ENTER" para indicar fim de inserção imediata, no lugar do delimitador (o padrão é $).

Assim, por exemplo, quando se quer criar um arquivo basta fornecer o nome do arquivo-saída, dar o comando ↑ I (inserção imediata) e começar a teclar o texto. Caso o usuário cometa algum erro de datilografia deve-se sair do comando de inserção pressionando-se a tecla ENTER, reposicionar o cursor no texto através do comando ← (volte um caráter) e reescrever o texto correto.

Os movimentos do cursor (←, →, "UPLINE", "DOWNLINE") são

sempre permitidos. Mesmo durante uma inserção. Se, no exemplo anterior, o usuário se esqueceu de teclar parte do registro ele poderá reposicionar o cursor, inserir o que foi esquecido, reposicionar o cursor e continuar teclando, sem sair do comando de inserção. Assim:



Durante uma inserção os movimentos do cursor não podem ultrapassar os limites do registro atual.

Caso se queira alterar um arquivo já existente deve-se fornecer os nomes do arquivo-entrada e o do arquivo-saída e imediatamente serão mostradas as primeiras oito linhas do arquivo-entrada.

Quando um comando de busca de cadeia de caracteres é executado a janela só será alterada quando necessário. Podem existir 3 casos:

— Se a cadeia pretendida já estiver na janela, esta permanecerá inalterada apenas o cursor será mudado de lugar, apontando para o fim da cadeia.
— Se a cadeia pretendida estiver na super-janela ("buffer" da janela) e não na janela, esta só será alterada para mostrar a cadeia desejada.
— Se a cadeia procurada não estiver na super-janela, será trazida uma linha do arquivo-entrada para a super-janela,

será então feita a busca nessa linha e em caso de fracasso a operação será repetida.

Para manter as facilidades do COBRA-400 DOTE também possui repetição automática de teclas (anexo 1). Isso é fornecido também para teclas de controle. Além disso é acessível o uso de letras minúsculas, cuja utilidade prática imediata está no fato de a impressora do sistema permitir a impressão de tais caracteres.

Visando a compatibilidade com o resto do sistema DOTE opera sobre arquivos sequenciais. Os arquivos criados ou alterados por DOTE tem o mesmo formato que os criados por utilitários do COBRA-400 (cap. 1.3, pág.21). Assim um arquivo pode ser criado ou alterado tanto por DOTE quanto por qualquer outro utilitário.

DOTE é monoprogramado, ou seja, apenas o terminal zero está interagindo com um usuário os demais estão inativos. Nº capítulo 6 são apresentadas sugestões para transformá-lo em multiprogramado.

Os comandos imediatos são:

(para melhor compreensão vide anexo 2: "MANUAL DO USUÁRIO DE DOTE")

a - relativos ao cursor

UP.....cursor sobe uma linha

DOWN.....cursor desce uma linha

LEFT.....cursor volta um caráter

RIGHT.....cursor avança um caráter

TOP.....desloca o cursor p/ canto superior esquerdo.

BOTTOM.....desloca o cursor p/ canto inferior esquerdo

b - relativos à tela

UPSCREEN.....troca a atual janela, pela imediatamente anterior

DOWN SCREEN...troca a atual janela pela imediatamente sequinte

UPLINE.....a janela sobe de uma linha

DOWNLINE.....a janela desce de uma linha

c - relativos ao controle

QUIT.....termina sessão de edição

MODE.....troca de modo de comando

SAVE.....termina sessão de edição e grava todos os arquivos necessários

d - relativos à edição

INSERT.....inserção de texto

DELETE CHAR...apaga o caráter apontado pelo cursor

DELETE LINE...apaga a linha atual

DUPLICATE LINE..duplica a linha atual

Os comandos normais, além dos já citados, são:

EXECUTE ..... executa os comandos normais já fornecidos

SEARCH BACKWARD.... procura uma cadeia para trás

SEARCH FORWARD .... procura uma cadeia para frente

REPLACE BACKWARD .. substitui uma cadeia por outra para trás

REPLACE FORWARD ... substitui uma cadeia por outra para frente

COPY ..... copia linhas do texto em outra parte do arquivo

MOVE ..... retira parte do texto de um lugar e coloca em outra parte do arquivo

REPEAT ..... repete um certo número de vezes os comandos dados

ANCHOR ON ..... "ancora" a busca a uma dada coluna

ANCHOR OFF ..... a busca será feita livremente pelo texto

DELIMITER ..... altera o atual delimitador.

A combinação dos comandos e facilidades existente em DOTE dá uma certa versatilidade ao editor.

oooo°oooo

CAPÍTULO III

3 - ESTRUTURA DE DADOS

A estrutura de dados aqui proposta visa minimizar o tempo gasto para edição de textos e também permitir uma certa maleabilidade para o usuário como, por ex., voltar no arquivo.

3.1 - Relativa ao teclado

À medida que o usuário vai teclando, os caracteres correspondentes às teclas pressionadas devem ser obtidos e analisados para então tomar-se as ações necessárias referentes ao caráter pressionado.

Basicamente, existem três tipos diferentes de ações a serem tomadas.

a - quando se referir a texto inserido através de comando imediato (C.I.)

b - quando se referir a comandos normais (C.N.) e seus argumentos

c - quando se referir a comandos imediatos.

No caso a , o caráter será ecoado na tela e deverá ser introduzido antes do caráter apontado pelo cursor. Para tal, o caráter apontado pelo cursor e todos os que estão à sua direita até o final do registro deverão ser deslocados de uma posição à direita.

No caso b o caráter será ecoado na linha de controle.

No caso c o caráter não será ecoado na tela, apenas serão tomadas as ações necessárias.

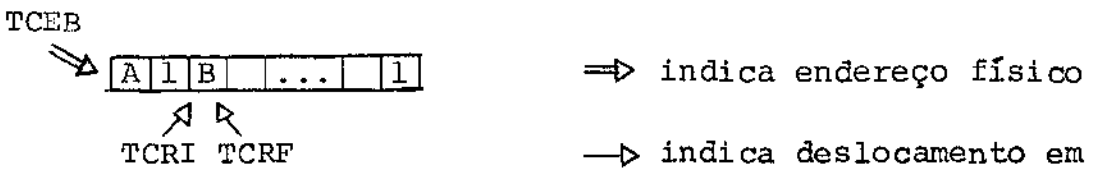
Para que não se restrinja a velocidade do usuário teclar ao tempo necessário para a obtenção, análise e tratamento do caráter foi utilizado um "buffer" para os dados que vem do teclado ("buffer" do teclado).

Devido ao fato do "buffer" do teclado comportar-se como uma fila, e de ser imprevisível a sequência de operações "põe um caráter "e" tira um caráter" tomou-se a decisão de fazê-lo circular, pois assim não haveria necessidade de movimentação de dados; apenas alteração dos apontadores para início e fim do "buffer". Sua capacidade padrão é de uma linha da tela menos um caráter (no caso a capacidade de linha é NOCARLIN = 64 caracteres). O acesso ao "buffer" do teclado é uma região crítica [9] na medida em que a rotina de interrupção irá incluir caracteres no "buffer" e o programa principal irá retirar caracteres do mesmo. Para que o tempo de acesso à região crítica pelo programa principal seja o menor possível optou-se pela utilização de dois "bytes" para cada caráter, sendo que um deles corresponde ao caráter ASCII, sempre que possível, e o outro ao código referente a teclas auxiliares (no caso SHIFT, IO CTRL, FUNCTION SELECT).

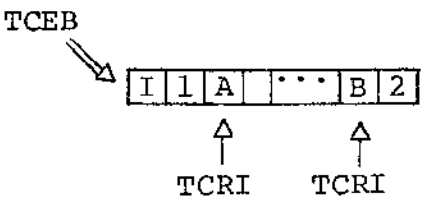
O "buffer" do teclado (TCBF) tem o tamanho físico de NOLNTC linhas (no caso NOLNTC = 2), em termos de caracteres isto seria TCTM = (NOLNTC x NOCARLIN)/2. Seus apontadores para início e fim (TCRI e TCRF) são relativos ao início do "buffer". Como o TCBF é circular e seus apontadores são relativos não há necessidade de en

dereço limite, apenas de endereço base (TCEB), pois temos TCTM.

Exemplo de configuração do TCBF vazio:



Exemplo de configuração do TCBF com elementos:



(TCTM-2) elementos

As operações que serão efetuadas sobre este "buffer" são:

a - "Pegar" um caráter do teclado e por no TCBF

b - Tirar um caráter do TCBF para ser analisado pelo editor

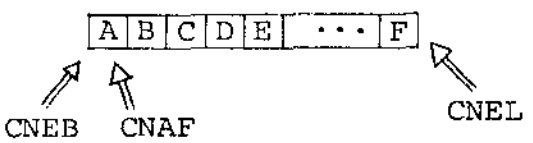
3.2 - Relativa aos Comandos Normais

Como os comandos normais serão ecoados na linha de controle e a filosofia do editor quanto a C.N. é de só executá-los após a indicação de término dos mesmos (pressão da tecla ENTER) decidiu-se usar "buffer" para comandos normais para não limitá-los ao compri

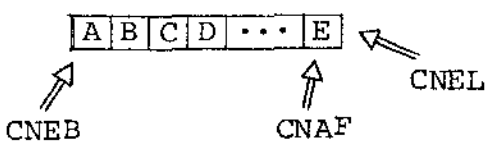
mento de uma linha. O "buffer" de C.N. também se comporta como uma fila, mas sabe-se que sempre haverá uma sequencia de comandos "põe um caráter" e só depois haverá uma sequencia de comandos "tira um caráter". Constatada essa regularidade de comportamento foi utilizado um "buffer" simples (com início fixo). Em cada posição há o código ASCII de um caráter. Na utilização deste "buffer" nunca será necessário uma região crítica dada a peculiaridade de que durante uma sequencia de "põe um caráter" é que será detectado o comando que causará o início de uma outra sequencia de "tira um carácter".

O "buffer" de comandos normais (CNBF) tem o tamanho de NOTLCN telas (no caso NOTLCN = 1), ou seja (NOTLCN x TLTM) caracteres. Seu apontador para fim (CNAF) é absoluto. Aqui temos necessidade de endereço base e limite (CNEB e CNEL).

Configuração única do CNBF vazio:



Exemplo de configuração do CNBF com elementos:



NOTLCN x TLTM elementos

As operações que serão efetuadas sobre este "buffer" são:

a - TIRAR DO TCBF e por no CNBF

b - tirar do CNBF um caráter para ser analisado pelo analisador de Comandos Normais

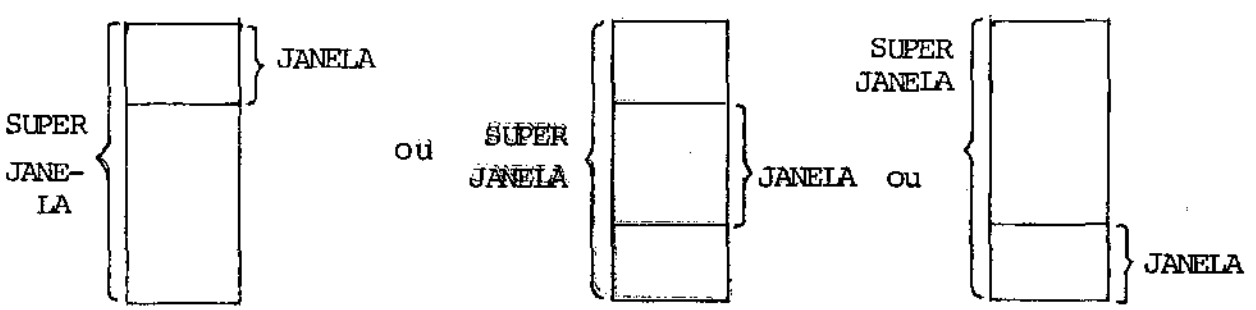
3.3 - Relativa à tela

A tela pode ser considerada uma "janela" através da qual se vê o arquivo.

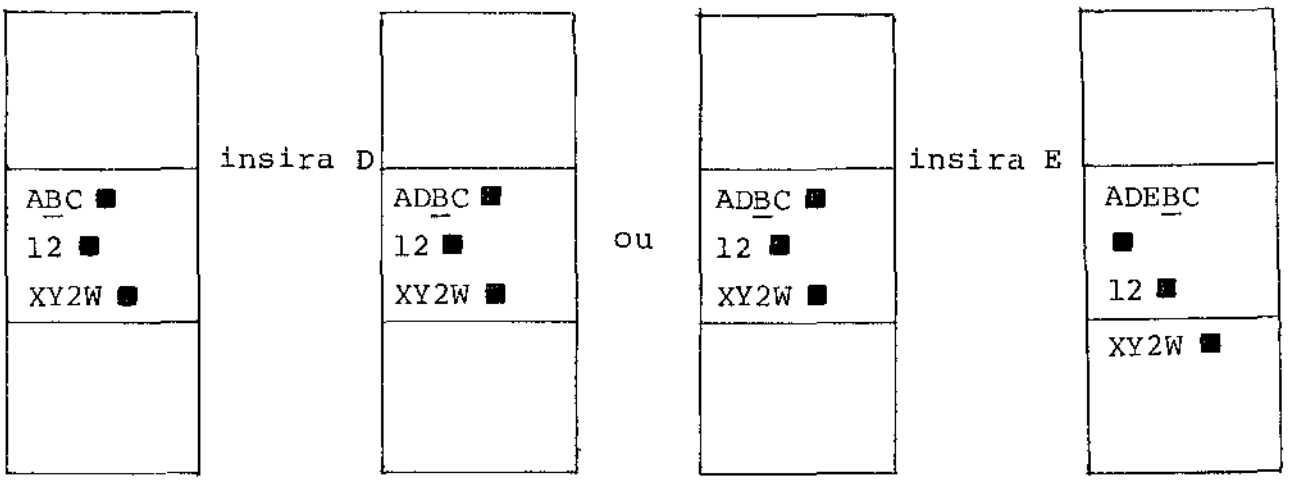
Um arquivo não pode ser editado sobre ele mesmo. Deverá sempre existir um arquivo diferente do de entrada, onde será colocado o arquivo resultante da sessão de edição. A partir do instante que uma porção de texto for gravada no arquivo de saída, esta não poderá mais ser alterada. A cada momento pode se alterar a parte do arquivo vista através da tela.

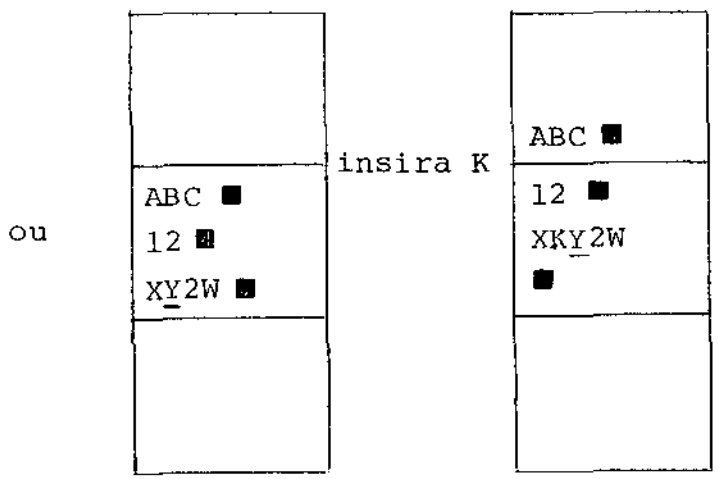
Para não se **restringir** o usuário à edição de apenas o que está na janela foi utilizado um "buffer" para a mesma (super-janela), que permite ao usuário "voltar" a janela, desde que não ultrapasse a super-janela. O tamanho padrão da super-janela, e mínimo, é de 3 telas (3 x TLTM caracteres).

A janela faz parte da super-janela mas não numa posição fixa. Por exemplo poderíamos ter:



Ao se inserir um caráter, se existir espaço disponível no registro este será utilizado. Caso não exista, a primeira ou a última linha da janela irá para a super-janela, que deverá ser convenientemente deslocada, e se terá espaço para a nova linha. Por exemplo, poderíamos ter:

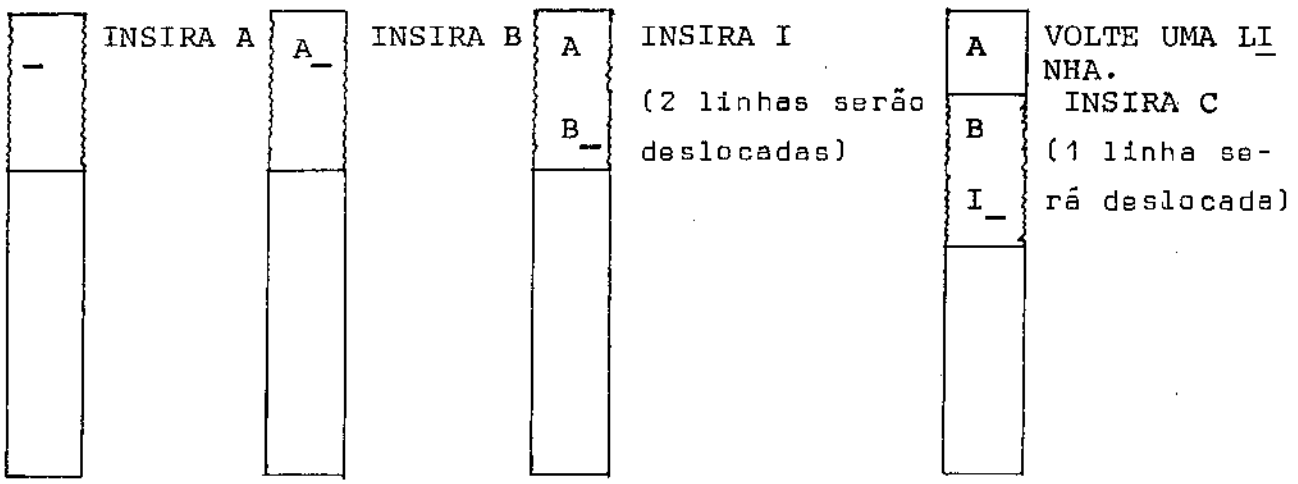


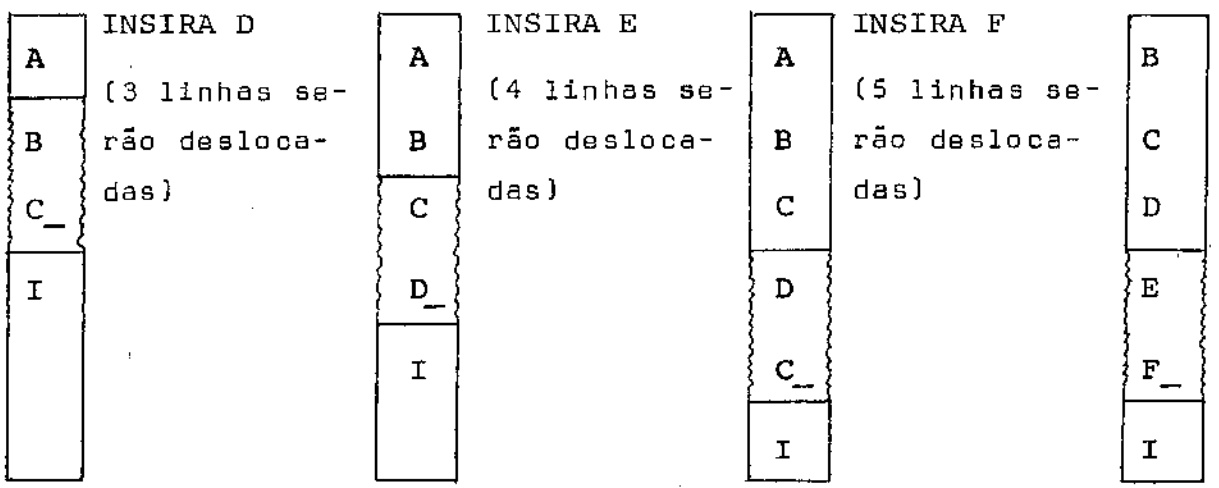


onde ■ indica fim de registro e _ indica posição de cursor.

Cada vez que uma linha sai da janela e vai para a parte não-visível da super-janela devem ser alteradas algumas das linhas da super—janela. Se esta última já está totalmente ocupada então a primeira linha da super-janela deve ser mandada para o arquivo de saída e assim cria-se uma linha livre na super-janela.

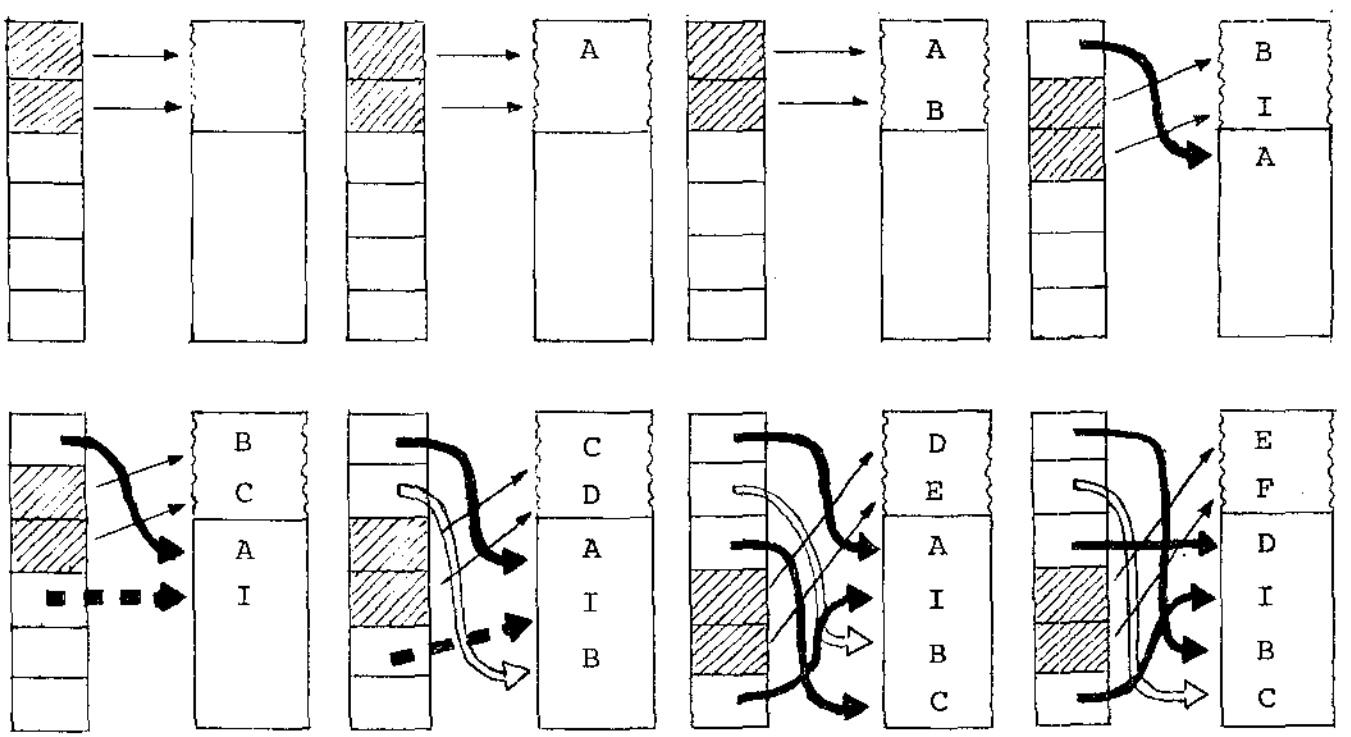
Exemplo de comportamento da janela e super-janela durante a inicialização de um arquivo.





Como era de se esperar de um editor de textos, a movimentação de dados é intensa, porém somente na janela temos a necessidade de que as linhas estejam em posições de memória sequenciais e pré-fixadas. Na parte não visível da super-janela esta sequencialidade não é necessária. Para diminuir o movimento real de dados nesta parte da super-janela decidiu-se usar linhas "lógicas" e um vetor que faz o ajustamento dessas linhas lógicas para as físicas. Assim o movimento de linhas na parte não-visível da super-janela será feito através da mudança apenas dos apontadores de linhas físicas e não das próprias linhas físicas.

No exemplo anterior teríamos:



onde ▨ indica endereços fixos e □ indica endereços quaisquer.

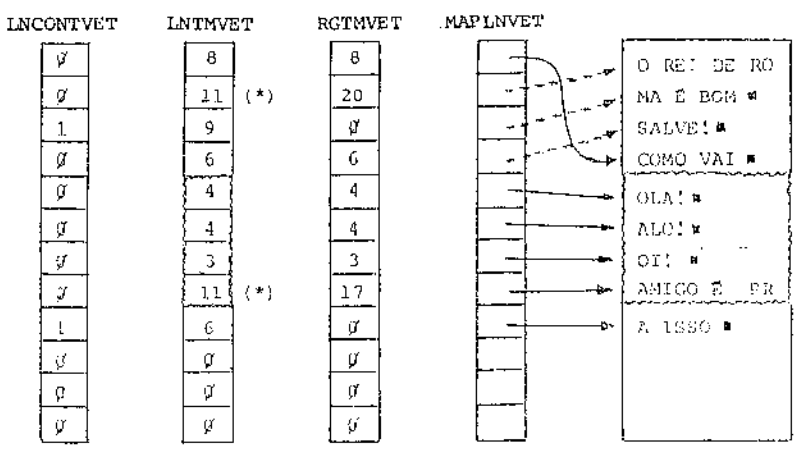
OBS.: Na representação pictórica as linhas da parte não-visível da super-janela foram mostrados sequencialmente para melhor visualização do processo, mas na implementação não há necessidade disto.

A janela tem TLTM (no caso, 512) caracteres e seus endereços estão fixos (são os endereços de memória equivalentes ao terminal em uso). O apontador para a posição atual é absoluto e é visualizado pelo piscar do cursor.

Como já foi visto, é preciso saber-se quantos caracteres temos em um registro. Para tal é usado um vetor (RGTMVET) que deverá conter esta informação para cada registro da super-janela. Já que cada linha é uma "candidata" a ser um registro, teremos no máximo SJTM (SJTM = nº de linhas da super-janela) registros em uma super-janela.

Para agilizar as tomadas de decisões pelo editor, decidiu-se também ter mais dois vetores de SJTM elementos, onde cada elemento de um dos vetores (LNTMVET) teria o nº de caracteres na linha correspondente e cada elemento do outro vetor (LNCONTVET) informaria se a linha correspondente é continuação da anterior ou não.

Já foi assinalada a necessidade de um vetor para fazer o ajuste das linhas lógicas para as físicas. Esse vetor tem SJTM elementos, onde cada elemento é o endereço do início da linha correspondente. Exemplo de uma configuração da super-janela e seus vetores de suporte:



(*) o valor de NOCARLIN no exemplo é 11.

As operações que serão efetuadas sobre a super-janela são:

a - "enrolar" o texto para cima

b - "enrolar" o texto para baixo

c - "abrir" uma linha em branco numa dada linha da janela

d - "matar" uma linha da janela

e - trazer N linhas da super-janela para a janela

f - tirar o 1º registro da super-janela e POR NO OUTBF.

3.4 - Relativa ao arquivo de saída

Quando uma linha sai da super-janela deve ir para o arquivo de saída.

Para manter a compatibilidade com o sistema de arquivos do COBRA-400 deve se ter registros de no máximo 511 caracteres (isso corresponde ao tamanho da tela menos um caráter, e ao tamanho de um setor do disco fixo (STTM) menos um caráter). Pelo mesmo motivo é necessário colocar-se no início e fim de cada registro o número de caracteres existentes no mesmo. Dada esta última peculiaridade do sistema de arquivos do COBRA-400 e querendo-se usar um "buffer" para arquivo do menor tamanho possível decidiu-se que cada vez que uma linha de um registro tiver que ir para arquivo será levado para arquivo o registro todo.

No "buffer" de arquivo (OUTBF) serão colocados os registros ,

que vem da super-janela, acrescidos das marcas de início e fim de registro. Esse "buffer" é do tamanho de um setor do disco fixo (STTM = 512 caracteres), deve ser fisicamente contínuo e estar localizado nos primeiros 32k da memória principal.

O "buffer" de arquivo-saída tem comportamento semelhante ao CNBF. Por isso temos apontador absoluto para fim de arquivo (OUTAF), endereços base e limite.

As operações que serão efetuadas sobre este "buffer" são:

a - Por elementos em OUTBF

b - Tirar elementos de OUTBF (STTM caracteres de cada vez)

3.5 - Relativa ao arquivo de entrada

Para se corrigir um arquivo já existente deve se trazer dados do arquivo de entrada para a super-janela. Para tal é usado um "buffer" (INPBF) para o arquivo de entrada. Seu tamanho é de STTM caracteres. Este "buffer", como o do arquivo de saída, deve ser fisicamente contínuo e estar localizado nos primeiros 32K da memória principal. Também para este "buffer" como para o do arquivo de saída, temos apontador absoluto para fim de arquivo (INPAF), endereços base e limite. Como já foi citado, para cada registro temos uma "marca" (número de caracteres do registro) de início e fim. Neste "buffer" teremos os registros e suas marcas, que deverão ser devidamente retiradas quando forem para a super-janela.

As operações que serão efetuadas sobre este "buffer" são:

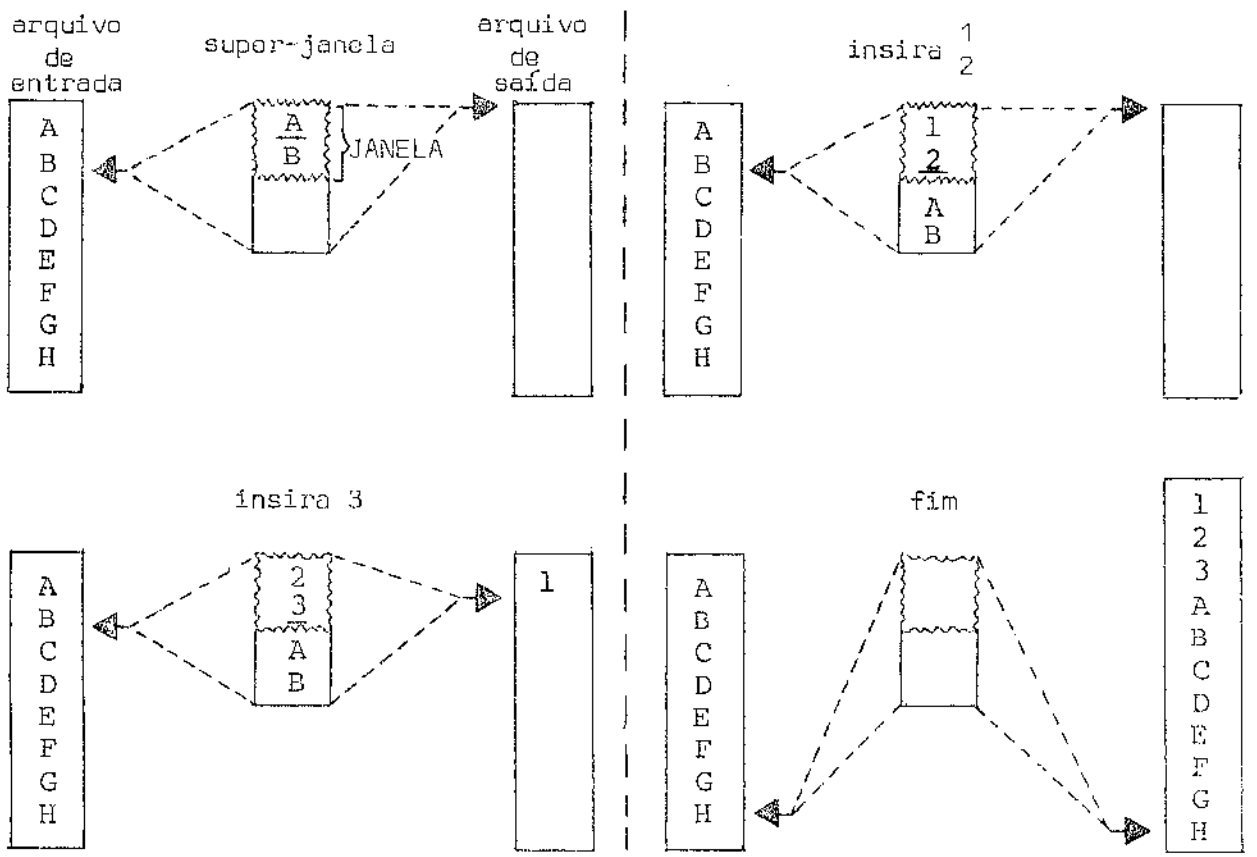
a - Por elementos em INPBF

b - Tirar elementos de INPBF (NOCARLIN caracteres de cada vez).

A movimentação de dados entre arquivo de entrada, super-janela e arquivo de saída é inter-ligada da seguinte maneira:

Sempre que se retira dados do arquivo de entrada estes vão para a super-janela, a maioria das vezes que se retira dados da janela estes vão para a super-janela, sempre que se retira dados da super-janela estes vão para o arquivo de saída.

Por exemplo a alteração de um arquivo já existente, a grosso modo, seria assim:



CAPÍTULO IV

4 - OPERAÇÕES PRIMITIVAS

Neste capítulo serão descritas as operações primitivas com as quais o editor vai operar. Essas primitivas estão diretamente ligadas à estrutura de dados proposta e constituem a base que opera diretamente sobre tal estrutura de dados.

A descrição das operações é feita em linguagem algorítmica com a seguinte convenção: chamadas de rotinas são representadas pelo nome da rotina grifado e em letras maiúsculas. O nome da rotina virá entre asteriscos, grifado e em letra maiúscula. Nome de variáveis também virão em maiúsculas. Para que se tenha idéia de como foram implementadas temos no anexo 4 a primitiva "ENROLAR O TEXTO PARA CIMA" em linguagem de montagem do INTEL 8080.

4.1 - Relativas ao teclado

Usando-se semáforos e os procedimentos P e V de Dijkstra para tratamento de interrupções [vide 9], teremos os seguintes procedimentos:

*** Pegar um caráter do teclado e POR NO TCBF ***

P(teclado) (*entrando na região crítica*)

"Lê" do teclado

```
        Se o buffer não está cheio
        então põe no TCBF
                TCRF: = (TCRF + 1) módulo (TCTM)
        senão soa o alarme do teclado do terminal
                e o caráter é ignorado
V (teclado)         (*saindo da região crítica*)

*** TIRAR um caráter DO TCBF ***
P (teclado)         (*entrando na região crítica*)
Se o "buffer" não está vazio
        então tira do buffer
                TCRI: = (TCRI + 1) módulo (TCTM)
V (teclado)         (*saindo da região crítica*)
```

4.2 - Relativas aos Comandos Normais

Temos os seguintes procedimentos:

```
*** Tirar do TCBF e POR NO CNBF ***
TIRAR DO TCBF
enquanto não é comando EXECUTE faça
        Se cabe no CNBF
            então por no "buffer"
                    incrementa 1 em CNAF
            senão mensagem ("COMMAND BUFFER IS FULL")
                    fim da rotina
            TIRAR DO TCBF
(*quando for EXECUTE volta ao programa principal para
            executar os comandos normais*)
```

*** TIRAR todos os elementos DO CNBF ***

enquanto CNBF tem elementos faça

tirar um elemento do "buffer"

verificar qual comando é

executar

4.3 - Relativos à tela

*** ENROLAR o texto PARA CIMA ***

Se pode enrolar

então transfere para SJ a 1ª linha da tela

(1) transfere as linhas da tela de 2 a NOLNTELA para a anterior

(2) transfere para a última linha da tela a 1ª linha posterior à janela

atualizar MAPLNVET

---

(1) equivalente à:

TRANSFERE (NOLNTELA - 1) LINHAS FÍSICAS A PARTIR DA LINHA (POSL1 + 1) PARA A PARTIR DA LINHA (POSL1) onde POSL1 é a posição da linha 1 da tela na SJ.

(2) equivalente à:

TRAZER 1 LINHA DA SJ PARA JANELA A PARTIR DA LINHA (POSL8 + 1) DA SJ E DA (POSL8) DA JANELA onde POSL8 é a posição da linha NOLNTELA da tela na SJ.

*** ENROLAR o texto PARA BAIXO ***

semelhante ao anterior com os devidos ajustes,
por ex: 1ª linha fica última linha, etc.

*** ABRIR UMA LINHA EM BRANCO numa dada linha I da tela ***

Se não há uma linha livre na SJ
então PÕE EM OUTBF o 1º registro da SJ
(*deixando livre pelo menos uma linha*)
Se I = (última linha da tela + 1)
então transfere p/ SJ a 1ª linha da tela p/ a anterior
(vide pág. anterior(1))transfere as linhas de 2 a NOLNTELA da tela p/ a anterior
põe ' ' na última linha da tela
atualizar MAPLNVET, LNCONTVET, LNTMVET, RGTMVET
senão transfere p/ SJ a última linha da tela para a seguinte
transfere as linhas de I a NOLNTELA - 1 p/ a seguinte
põe ' ' na linha I
atualiza MAPLNVET, LNCONTVET, LNTMVET, RGTMVET

*** "MATAR" uma dada LINHA I DA TELA ***

Se I não é a última linha da tela
então transfere as linhas de (I+1) até NOLNTELA
para a anterior

transfere para a última linha da tela a 1ª linha

posterior à janela

atualiza MAPLNVET, LNCONTVET, LNTMVET, RGTMVET

*** TRAZER N LINHAS DA SJ PARA JANELA

A PARTIR DA LINHA K DA SJ E DA I DA TELA ***

Se existem as N linhas pedidas na SJ

então enquanto N > Ø faça

transfere para tela da linha K p/ linha I

decrementa 1 de N

incrementa 1 em K

incrementa 1 em I

senão M: = nº linhas disponíveis (*M < N*)

N: = N-M

enquanto M > Ø faça

transfere p/ tela da linha K p/ linha I

decrementa 1 de M

incrementa 1 em K

incrementa 1 em I

Se não tentou ultrapassar a 1ª linha da SJ

então enquanto N > Ø faça

TIRAR DO INPBF E POR NA LINHA I

decrementa 1 de N

incrementa 1 em I

senão MENSAGEM ("ATTEMPT TO OVER CROSS

THIS BUFFER")

```
*** TRANSFERE N LINHAS FÍSICAS A PARTIR DA LINHA I PARA A
                PARTIR DA LINHA J ***
    para cada linha A: = N, 1, -1 faça
                para cada caráter B: = 1, NOCARLIN faça
                        SJ [J,B]: = SJ [I,B]
                incrementa 1 em I
                incrementa 1 em J
```

4.4 - Relativas ao arquivo saída

```
*** POR elementos EM OUTBF ***
POE MARCA DE REGISTRO    (*INICIO DE REGISTRO*)
Para cada caráter I do 1º registro da SJ faça
     Se não cabe em OUTBF
             então TIRA DE OUTBF
     OUTBF [OUTAF]: = I
     incrementa 1 em OUTAF
POE MARCA DE REGISTRO     (*FIM DE REGISTRO*)
atualiza MAPLNVET, LNCONTVET, LNTMVET, RGTMVET

*** TIRAR elementos DE OUTBF ***
Se ainda cabem registros no arquivo-saída
  então gravar OUTBF em disco
        OUTAF: = OUTEB
  senão mensagem ("END OF OUTPUT FILE REACHED")
```

*** POE MARCA de início ou fim DE REGISTRO ***

Se OUTBF cheio

então TIRAR DE OUTBF

põe a 1ª e a 2ª partes da marca

senão põe a 1ª parte da marca

Se OUTBF cheio

então TIRAR DE OUTBF

põe a 2ª parte da marca

4.5 - Relativas ao arquivo entrada

*** POR elementos EM INPBF ***

Se ainda há elementos no arquivo entrada

então ler de disco

INPAI: = INPEB

senão mensagem ("END OF INPUT FILE REACHED")

*** TIRAR elementos DE INPBF E POR NA LINHA I ***

Se não há registro já começado

então TIRA MARCA (* do fim de registro *)

TIRA MARCA (* do início de registro *)

atualizar RGTMVET, LNCONTVET

no. carac. reg: = marca

senão atualizar LNCONTVET

N: = mínimo (NOCARLIN, no. carac. reg.)

```
      para cada J: = 1, N faça
      Se INPBF vazio
          então PÕE EM INPBF
      SJ [I,J]: = INPBF [INPAI]
J: = N+1
enquanto J < NOCARLIN faça
      SJ [I,J]: = ' '
      incrementa 1 em J
atualizar LNTMVET
no.carac.reg: = no.carac.reg - N
```

*** TIRA MARCA de fim ou início de registro ***

Se INPBF não está vazio
então pega a 1ª parte da marca
Se INPBF não está vazio
então pega a 2ª parte da marca
Se INPBF está vazio
então PÕE EM INPBF
senão PÕE EM INPBF
pega a 2ª parte da marca
senão PÕE EM INPBF
pega a 1ª e a 2ª parte da marca

oooo°oooo

CAPÍTULO V

5 - IMPLEMENTAÇÃO DE COMANDOS

Os algoritmos que descrevem os comandos do editor são feitos com base nas operações primitivas descritas no capítulo anterior.

Um exemplo de comando traduzido para linguagem de montagem do INTEL 8080 está no anexo 5, com o comando "INSERT".

5.1 - Relativos ao cursor

*** CURSOR UP ***

```
Se precisa "enrolar" o texto p/ baixo
     então Se pode enrolar
               então ENROLA PARA BAIXO
               Senão MENSAGEM ("ATTEMPT TO OVERCROSS THIS BUFFER")
Se não existe tal coluna na linha anterior
     então nova coluna é a última coluna da linha anterior
decrementa 1 do contador de linha atual
atualiza variáveis para piscar o cursor
```

*** CURSOR DOWN ***

semelhante ao anterior com os devidos ajustes,

por ex: "enrolar" para baixo fica "enrolar" para cima,...

```
*** CURSOR LEFT ***
    Se precisa mudar de linha
       então se precisa "enrolar" o texto p/ baixo
              então Se pode "enrolar"
                     então ENROLA PARA BAIXO
                     senão MENSAGEM ("ATTEMPT TO
                           OVERCROSS THIS BUFFER")
                           FIM DA ROTINA
                 atualiza endereço do cursor
              senão decrementa 1 do endereço do cursor
                    decrementa 1 do contador de linha atual
  atualiza variáveis para piscar o cursor

*** CURSOR RIGHT ***
    Se precisa mudar de linha
       então Se precisa "enrolar" o texto p/ cima
              então Se não pode enrolar
                     então POE EM OUTBF (*1º REGISTRO DA SJ*)

                           ENROLA PARA CIMA
                           atualiza endereço do cursor
                     senão atualiza endereço do cursor
              atualiza variáveis para piscar o cursor
              incrementa 1 no contador de linha atual
       senão  incrementa 1 no endereço do cursor
              atualiza variáveis para piscar o cursor
```

*** TOP ***

endereço do cursor: = MAPLNVET [posição da linha 1 da tela].

atualiza variáveis para piscar o cursor

contador linha atual: = posição linha 1 da tela

*** BOTTOM ***

semelhante ao anterior com os devidos ajustes, por exemplo "posição linha 1 da tela" fica "posição linha 8 da tela"

5.2 - Relativos à tela

*** UPSCREEN ***

Se há uma tela anterior na SJ

então para cada posição I da tela faça

ACC: = tela [I]

tela [I] = SJ [f(I)] (1)

SJ [f(I)] = ACC

TOP

senão MENSAGEM ('ATTEMPT TO OVER CROSS THIS BUFFER')

*** DOWNSCREEN ***

N: = nº linhas da SJ posterior a janela

---

(1) f(I): = endereço do caráter que está TLTM caracteres atrás do caráter de endereço I.

Se há uma tela posterior na SJ

então para cada posição I da tela faça

ACC: = tela [I]

tela [I]: = SJ[g(I)] (2)

SJ[g(I)]: = ACC

TOP

senão para cada posição I da SJ posterior a janela faça

ACC: = tela [h(I)] (2)

tela [h(I)]: = SJ[I]

SJ[I]: = ACC

TIRAR de INPBF (MOLNTELA -N) linhas

*** UPLINE ***

ENROLAR PARA BAIXO

*** DOWNLINE ***

ENROLAR PARA CIMA

5.3 - Relativos ao Controle

*** QUIT ***

para cada endereço I da tela faça

tela [I]: = ' '

desassinala o terminal da sessão de edição

*** MODE ***

ESTADO: = 'I'

---

(2) g(I), h(I) semelhante a f(I)

*** SAVE ***

para cada registro da SJ faça

POR EM OUTBF (*1º REGISTRO DA SJ*)

enquanto arq. entrada não está vazio faça

I: = 1

enquanto não terminar o registro faça

TIRA DO INPBF E PÕE NA LINHA I

incrementa 1 em I

POR EM OUTBF (*1º REGISTRO DA SJ*)

QUIT

5.4 - Relativos à edição

*** INSERT ***

TIRA DO TCBF

enquanto não for delimitador faça

se é comando imediato

então se é comando de cursor

então executa (se possível)

senão soa o alarme do terminal

senão se o registro ocupa um nº inteiro

de linhas físicas

então ABRIR UMA LINHA EM BRANCO

NA LINHA (LINATUAL + 1)

desloca todos os caracteres a frente

do cursor (inclusive) de uma

posição à direita

coloca o caráter lido na posição

indicada pelo cursor

TIRAR DO TCBF

*** DELETE CHAR ***

desloca todos os caracteres seguintes ao cursor de

uma posição à esquerda

Se tal registro ocupava um nº inteiro

de linhas físicas mais um caráter (o " ■ ")

então MATAR A LINHA (ÚLTIMA DO REGISTRO ATUAL)

*** DELETE LINE ***

MATAR A LINHA (ATUAL)

*** DUPLICATE LINE ***

ABRIR UMA LINHA EM BRANCO NA LINHA (ATUAL + 1)

para cada caráter I da linha atual faça

SJ(linha atual + 1, I) ← SJ(linha atual, I) (1)

atualizar LNCONTVET, LNTMVET, RGTMVET

*** TAB ***

verifica na tabela de tabulação a posição seguinte para coluna

atualizar variáveis para piscar o cursor

(1) Tem-se acesso a SJ aqui através de nº de linha e nº da coluna dentro de uma dada linha.

```
*** SEARCH BACKWARD ***
não achou: = TRUE
Se ANCOR ≠ ∅
   então I.: = endereço do cursor
        enquanto não achou faça
            se I  < início da SJ
               então MENSAGEM ("STRING NOT FOUND")
                     FIM  DA  ROTINA
               senão Se último caráter da cadeia = SJ[I]  (1)
                        então se o resto da cadeia é igual
                                 então não achou: = false
                     I: = endereço do caráter anterior ao atual
    senão j: = linha atual
          enquanto não achou faça
              Se j < 1
                 então MENSAGEM ("STRING NOT FOUND")
                       FIM DA ROTINA
                 senão se 1º caráter da cadeia = SJ [j, ANCOR]  (2)
                          então se o resto da cadeia é igual
                                   então não achou: = false
                       decrementa 1 em j
(*JÁ ACHOU A CADEIA*)
Se a linha onde ocorreu a cadeia não está na janela
     então ENROLA PARA BAIXO até que tal linha esteja na 1ª
                                                 linha da janela
atualizar o cursor na coluna sequinte ao último caráter da cadeia
```

(1) Tem-se acesso a SJ aqui pelo endereço de cada caráter.
(2) Tem-se acesso a SJ aqui através de nº de linha e nº de coluna dentro de uma dada linha.

*** SEARCH FORWARD ***

Semelhante a SEARCH BACKWARD com os devidos ajustes, por ex: "do cursor pra trás" fica "do cursor para frente"

ENROLA PARA BAIXO fica ENROLA PARA CIMA

*** REPLACE BACKWARD *** (RB S1 S2)

SEARCH BACKWARD S1

para cada caráter I de S1 faça

DELETE CHAR

para cada caráter I de S2 faça

INSERT I

*** REPLACE FORWARD *** (RF S1 S2)

SEARCH FORWARD S1

para cada caráter I de S1 faça

DELETE CHAR

para cada caráter I de S2 faça

INSERT I

*** COPY *** CP [NO] LINE

se LINE é acessível

então por num "buffer" auxiliar as NO linhas seguintes (inclusive)

inserir após LINE o que está no "buffer"

atualizar o cursor no caráter seguinte ao último caráter inserido

senão MENSAGEM (LINE, 'NOT ACCESSIBLE')

*** MOVE ***

semelhante à COPY, apenas que após o comando:

"por num buffer auxiliar as NO linhas seguintes (inclusive)"

existe o comando:

para cada L: = 1, NO faça

MATAR LINHA ATUAL

*** TABULATE ***

redefine o vetor de tabulação

*** REPEAT *** RE NO (C1$...CN$)

enquanto NO ≠ ∅ faça

executar C1

:

executar CN

decrementa 1 de NO

*** ANCHOR ON *** (A + NO)

ANCOR: = NO

*** ANCHOR OFF ***

ANCOR: = ∅

*** DELIMITER *** DM char

DELIM: = char

oooo o oooo

CAPÍTULO VI

6 - CONCLUSÃO E SUGESTÕES

6.1 - Estado atual de DOTE

DOTE é monoprogramado, ou seja usa apenas o terminal zero. Dos comandos propostos estão em funcionamento os seguintes: "CURSOR LEFT", "CURSOR RIGHT", "DELETE CHAR", "INSERT", "QUIT", "SAVE".

Esse sub conjunto ocupa aproximadamente 4k "bytes" de memória. A implementação dos comandos restantes supõe-se que ocupe outros 4k "bytes".

A maior dificuldade na implementação está no fato de ter sido feito um editor de textos e uma estrutura de dados razoavelmente sofisticados para ser descrito em linguagem de montagem do micro processador INTEL 8080. Como a estrutura de dados já está implementada não seria difícil implementar os comandos restantes.

O fato de ter sido feito o acesso a disco através de "busy wait" impõe um rítmo um pouco mais lento quando da ocasião de se ler ou escrever em arquivo. Esse fato poderá ser sanado quando se conhecer exatamente o tratamento da interrupção do disco. Com exceção desse fato a resposta do editor é rápida e comparável aos editores já existentes no COBRA-400. Enquanto não é preciso ter-se acesso a disco, DOTE responde mais prontamente por estar tudo em memória principal.

A repetição automática de teclas é fornecida também para comandos imediatos o que facilita e agiliza nos casos dos comandos de cursor, e de apagar um caráter.

6.2 - Sugestões para futuras extensões

Uma ampliação natural do sistema seria torná-lo multiprogramado para que os quatro terminais possam usar DOTE simultaneamente. Outra ampliação desejável seria a interação com um arquivo auxiliar, de onde se poderia ler ou escrever trechos de texto.

6.3 - DOTE multiprogramado

O editor implementado funciona para só um terminal (monoprogramado). Para que se possa utilizar os 4 terminais com o editor (multiprogramado) deveriâmos ter pelo menos 4 variáveis a mais RBVAR (um "BYTE") que daria o endereço inicial do conjunto de variáveis relativas ao terminal atualmente sendo atendido RBPILHA (uma "WORD") que seria o apontador da pilha do usuário atual e TERMATUAL (um "BYTE") que daria o nº do terminal sendo atualmente atendido e TIMESLICE (uma "WORD") que faria a contagem de porção de tempo para cada usuário.

Como o número de variáveis utilizadas não ultrapassa 256 "bytes" todas podem ser indicadas usando-se uma página de memória. Assim, um pequeno exemplo seria:

ANTES

```
SJCT:  DB 1
MAPLNVET:  DW 74ØH
           DW 78ØH
           DW 76ØH
RGATUAL: DB Ø
```

DEPOIS

```
SJCT  EQU Ø
MAPLNVET  EQU 1
RGATUAL  EQU 3
ORG  ØBØØØH
DB 1
DW ØB5ØØH
DB Ø
```

As posições de memória de ØBØØØH a ØBØFFH (256 posições) vão conter as variáveis relativas a um dos usuários ou apontadores para início de conjunto de variáveis, caso estes sejam vetores. O que antes era endereço das variáveis passa a ser o deslocamento em relação ao início da página que contém as variáveis. Se uma variável era do tipo vetor (nome terminando com VET) teremos o endereço do início do vetor e não o próprio vetor; o caso de MAPLNVET no exemplo começa em ØB5ØØH.

A única mudança na rotina de interrupções seria no atendimento do relógio que faria a mudança de usuário e do contexto do usuário (as 4 variáveis acrescidas e restaurar os registradores e apontadores de pilha) quando necessário. A interrupção do teclado deveria ser atendida para todos os terminais colocando o caráter lido no "buffer" conveniente.

O acesso a vetores era feito através dos comandos LXI e LHLD que deveriam passar a ser:

```
LXI H, RGTMVET ..............  PUSH   PSW
                               LDA    RBVAR
                               MOV    H,A
                               MVI    L, RGTMVET
                               MOV    A,M
                               INX    H
                               MOV    H,M
                               MOV    L,A
                               POP    PSW
```

```
LHLD  RGTMVET ..............  PUSH  PSW
                              LDA   RBVAR
                              MOV   H,A
                              MVI   L, RGTMVET
                              MOV   A,M
                              INX   H
                              MOV   H,M
                              MOV   L,A
                              MOV   A,M
                              INX   H
                              MOV   H,M
                              MOV   L,A
                              POP   PSW
```

O acesso às demais variáveis era feito através de comandos LHLD, SHLD, LDA, STA, LXI que deveriam passar a ser:

```
LHLD INTENDER ..............  PUSH  PSW
                              LDA   RBVAR
                              MOV   H,A
                              MVI   L, INTENDER
                              MOV   A,M
                              INX   H
                              MOV   H,M
                              MOV   L,A
                              POP   PSW
```

```
SHLD INTENDER ..... PUSH   D
                    PUSH   PSW
                    LDA    RBVAR
                    MOV    D,A
                    MUI    E, CRSEND
                    MOV    A,L
                    STAX   D
                    INX    D
                    MOV    A,H
                    STAX   D
                    POP    PSW
                    POP    D

LDA CRSAVX ........ PUSH   H
                    LDA    RBVAR
                    MOV    H,A
                    MUI    L, CRSAUX
                    MOV    A,M
                    POP    H

STA CRSAUX ........ PUSH   H
                    PUSH   PSW
                    LDA    RBVAR
                    MOV    H,A
                    MUI    L, CRSAUX
                    POP    PSW
                    MOV    M,A
                    POP    H
```

```
LXI H, CRSEND ..... PUSH   PSW
                    LDA    RBVAR
                    MOV    H,A
                    MVI    L, CRSEND
                    POP    PSW
```

6.2.2 - Arquivo Auxiliar

A idéia de arquivo auxiliar seria um lugar de onde se pode trazer trechos de texto ou guardar os mesmos, ou ambos.

Para que se possa interagir com um arquivo auxiliar deveríamos ter um parâmetro a mais na inicialização da sessão de edição:

/ SECONDARY FILE <nome>

que definiria qual o arquivo que servirá de auxiliar, e os seguintes comandos:

APPEND <NO > ..... insere NO linhas no final do arquivo auxiliar, que são copiadas do arquivo principal a partir da linha atual, inclusive. O arquivo principal não é alterado.

READ [<L1>]<L2> ..... insere após a linha atual a parte do arquivo auxiliar que vai das linhas L1 até L2. Se L1 não for for-

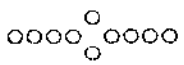
necido será desde o início do arquivo auxiliar. O arquivo auxiliar não é alterado. O cursor ficará posicionado no caráter seguinte ao último caráter copiado.

NUMBER ..... ao se entrar com este comando o editor devolverá, na linha reservada ao editor, o número da linha atual.

WRITE [<NO>]..... o arquivo auxiliar passará a ter NO linhas, se NO for omitido vale 1, que são copiadas do arquivo principal a partir da linha atual, inclusive. O arquivo principal não é alterado.

WATCH ..... serve para "espiar" o arquivo auxiliar. Limpa a tela e em seguida mostra a primeira janela do arquivo auxiliar (no arquivo auxiliar só se pode usar comandos de tela, o comando para voltar ao arquivo principal ou o comando NUMBER). O cursor ficará posicionado no canto superior esquerdo da janela.

COME BACK ..... volta para o arquivo principal. Limpa a janela e em seguida mostra a janela do arquivo principal que estava sendo mostrada antes de se dar o comando WATCH.



BIBLIOGRAFIA

1 - Benjamim, A. J. An Extensible Editor for a small machine with Disk storage. Communications of the ACM, 15,(8): 742 - 747, 1972.

2 - Bourne, S.R. A design for a text editor. Software - Practice and Experience, 1, 73-81, 1971.

3 - COBRA Computadores e Sistemas Brasileiros S/A
Manual do usuário COBRA-400. Rio de Janeiro 1977.

4 - COBRA Computadores e Sistemas Brasileiros S/A.
Mini Disk multi-sector BSH firmware Rio de Janeiro, 1977.

5 - Coulouris, G.F. at alli The design and implementation of an Interactive Document Editor. Software Practice and Experience, 6, 271-279, 1976.

6 - Deutsh, P. & Lampson, B.W. An online editor. Communications of the ACM, 10,(12): 793-803, 1967

7 - Elliot, B. Design of a simple screen editor. Software Practice and Experience, 12, 375-384, 1982.

8 - GODOT USER MANUAL, 1978.

9 - Guimarães, C.C.
Princípios de Sistema Operacionais, 1979.

10 - Hazel, P. A General-Purpose Text Editor for OS/360. Software Practice and Experience, 4, 389-399, 1974.

11 - Irons, E.T. & Djorup, F.M. A CRT Editing System Communications of the ACM, 15, (1): 16-20, 1972.

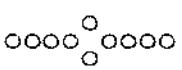
12 - Kerninghan, B.W. & Plauger, P.J.
Software tools, 1976.

13 - Schneider Jr, B.R. & Watts, R.M. SITAR: An Interactive Text Processing System for Small Computers. Communications of the ACM, 20 ,(6): 495-499, 1977.

14 - Sonderegger Jr, R.P. Typer an editor for the on line composition of text. Computer Graphics, 10, (2): 1-6, 1976

15 - Teperman, A. & Katzenelson, J. A Format Editor. Software Practice and Experience, 2, 219-230, 1972.

16 - Van Dam, A. & Rice, D.E. On line text editing: a survey. Computing Surveys, 3 (3): 93-114, 1971.



ANEXO 1

ROTINA DE INTERRUPÇÃO

```
INTER:
;*************************************************************
;ROTINA DE INTERRUPCAO
;A- DETECTA A INTERRUPCAO DO TECLADO CONVERTE AS COORDENADAS DO
   ;CARATER E POE NO BUFFER DE TECLADO (TCBF)
;B- SE A TECLA CONTINUA PRESSIONADA REPETE O CARATER NO TCBF
;C- QUANDO NECESSARIO PISCA O CURSOR

;*************************************************************
        PUSH H
        PUSH D
        PUSH B
        PUSH PSW
        IN INTPORT          ;SALVA OS REGISTRADORES
        CPI CLOCK           ;VE QUAL PERIFERICO PEDIU INTERRUPCAO
        JZ RELOG            ;SE NAO FOI O RELOGIO
        CPI CONJTERM
        JNZ OUTROS              ;ENTAO SE FOI O CONJUNTO DE TERMINAIS
        IN TERMPORT                     ;ENTAO DESLIGA A INTERRUPCAO
        MOV C,A
        OUT TERMPORT
        IN TECPORT                          ;LE DO TECLADO
        MOV B,A
        MOV A,C
        CPI TERMO
        JNZ OUTROS                          ;SE FOI O TERMINAL ZERO
        MOV A,B
        STA  RPTAUX
        LXI  H,RPTTIME
        MVI  M,RPTLAT
REPETE:
         CALL CONVERT                       ; ENTAO CONVERTE AS COORDENADAS NO CARCTER ASCII
                                                ;SE POSSIVEL
         PUSH  B                               ;SALVA B,C (O CARACTER ASCII E O CODIGO MAIUSC),
         LDA  TCRF
         MOV  C,A
         MVI  B,TCTM
         CALL MADDB                             ;TCRF (--TCRF+1 MODULO TCTM
         LDA  TCRI
         CMP  C                                 ;SE O BUFFER NAO ESTA CHEIO
          JZ  OVERFLOW                              ;ENTAO POE NO TCBF O CODIGO DO CARATER
         LDA  TCRF
         MOV  E,A
         MVI  D,0
         LXI  H,TCEB
          DAD  D
         POP  D
         MOV  M,D
         MVI  B,0
         LXI  H,TCEB
         DAD  B
         MOV  M,E
         MVI  B,TCTM
         CALL MADDB
         MOV  A,C
         STA  TCRF                                      ;ATUALIZA TCRF
```

```
OUTROS: POP  PSW
        POP  B
        POP  D
        POP  H
        EI
        RET
OVERFLOW: MVI  A,APITO
        OUT TECPORT
        JMP  OUTROS
RELOG:  XRA  A
        OUT CLOCPORT
        LXI  H,CRSCT
        INR  M
        MVI  A,PISCTIME
        CMP  M
        JNZ  VESEREPETE
        MVI  M,0
        LHLD  CRSEND
        MOV  A,M
        CPI  CURSOR
        JNZ  POECURSOR
        LDA  CRSAUX
        MOV  M,A
VESEREPETE:
        IN    TERMPORT
        MOV  C,A
        OUT  TERMPORT
        IN   TECPORT
        MOV  B,A
        MOV  A,C
        CPI  TERMO
        JNZ  VESEREPETE
        LDA RPTAUX
        CMP  B
        JNZ  OUTROS
        LXI  H, RPTTIME
        DCR  M
        JNZ  OUTROS
        MVI  M,1
        JMP REPETE
POECURSOR: MVI  M, CURSOR
        JMP  VESEREPETE
```

```
                                ;SENAO ESTOUROU TCBF:SOA ALARME DO TERM.

     ;RESTAURA OS REGISTRADORES
;SENAO DESLIGA A INTERRUPCAO DO RELOGIO


                                ;PISCA O CURSOR

                                ;SE A TECLA CONTINUA PRESSIONADA
                                    ;ENTAO VAI REPETIR A TECLA

                                ;RESTAURA OS REGISTRADORES
```

ANEXO 2

MANUAL DO USUÁRIO DE DOTE

MANUAL DO USUÁRIO DE DOTE

(Display Oriented Text Editor)

I - INTRODUÇÃO

A maioria dos editores de texto são orientados para linha, no sentido de se ter acesso direto a apenas uma única linha. Esse tipo de editor é apropriado para terminais do tipo máquina de escrever, mas quando estamos em um terminal de vídeo devemos usar o fato de termos acesso direto (através de controles do cursor) à tela toda.

DOTE foi projetado para permitir edição direta de todo o texto visível na tela de um terminal de vídeo. O cursor pode ser diretamente posicionado no lugar do texto onde as alterações serão feitas. Além dessas facilidades, tem a vantagem de que automaticamente é mostrado o contexto no qual as correções serão feitas.

DOTE pretende também conciliar os dois seguintes fatos:

— apresentação de várias facilidades para o usuário

— simplicidade de uso

que o tornam satisfatório tanto a profissionais de programação de computadores como a pessoas que usam o computador somente como ferramenta auxiliar.

II - CONCEITOS GERAIS

Algumas convenções:

— Janela é a porção do texto visível na tela.

— O delimitador padrão de DOTE é $.

— Todos os parâmetros entre colchetes ([ ]) são opcionais.

— O tamanho de um bloco de memória é de tantos caracteres quantos sejam necessários para se encher uma tela de terminal.

— Linha atual é aquela onde está o cursor.

DOTE deve ser implementado em um computador que tenha terminal de vídeo com cursor endereçável. A primeira linha do terminal é reservada para o editor.

Para se inicializar DOTE temos os seguintes parâmetros possíveis:

[/INPUTFILE] ..... nome do arquivo a ser editado

/OUTPUTFILE ..... nome do arquivo já editado

[/BUFFER NO] ..... define o tamanho do "BUFFER" em um número inteiro de blocos (sendo cada bloco equivalente a uma tela de terminal).

A inicialização de DOTE faz com que apareça na tela a primeira janela do arquivo de entrada e o cursor estará posicionado no

canto superior esquerdo da janela.

É obrigatório o fornecimento do nome do arquivo de saída (OUTPUTFILE).

O "BUFFER" padrão do DOTE é equivalente a três telas de terminal (/BUFFER 3). Esse é o tamanho mínimo do "BUFFER".

Todo comando que precisa "voltar" no arquivo não pode ultrapassar o "BUFFER".

Ao se tentar posicionar o cursor fora da janela, o texto será "enrolado" para cima ou para baixo, conforme a posição do cursor. Caso se tente ultrapassar o início do "BUFFER" ou o final do arquivo, será mostrada uma mensagem explicativa na linha reservada ao editor.

O efeito de se "enrolar" o texto para cima também será conseguido quando a janela estiver cheia e se tentar escrever além da última linha.

No DOTE o endereçamento pode ser feito tanto por conteúdo (comandos de busca) como por linha, relativo à linha atual (comandos do cursor).

Para melhor conseguir implantar a filosofia de DOTE (- usar a tela toda durante a edição - comandos de sintaxe simples, mas potentes) foram usados dois tipos de comandos:

— comandos imediatos

— comandos normais

A um dado momento ou se está em estado de comando imediato ou se está em estado de comando normal. No canto superior esquerdo

da tela, na linha reservada ao editor, teremos a palavra "IMMEDIATE" caso se esteja em estado de comando imediato. Quanto se estiver em estado de comando normal teremos um "*" na linha reservada ao editor indicando que DOTE está esperando dados.

Os comandos imediatos não tem parâmetros (com exceção do comando "INSERT") e são fornecidos ao editor quando se estiver em estado de comando imediato através da pressão das teclas CTRL juntamente com uma outra tecla. Não serão ecoados no terminal os caracteres pressionados, apenas serão imediatamente tomadas as ações necessárias. Por exemplo:

O comando "QUIT" será obtido pressionando-se

CTRL Q

e causará o término imediato da sessão de edição sem alterar ou gravar arquivo algum, voltando o comando do terminal ao monitor.

Os comandos normais podem ter parâmetros e são fornecidos ao editor quando se estiver em estado de comando normal através dos mnemônicos que representam cada comando (esses mnemônicos são todos compostos de duas letras). As teclas pressionadas serão ecoadas na linha reservada ao editor. Poderão ser fornecidos vários comandos, separados pelo delimitador, e só depois se manda executar a sequência de comandos pressionando-se a tecla ENTER. Por exemplo: Para trocar uma parte do texto que diz "mandar o empregado" por "pedir ao patrão" e procurar a próxima ocorrência de "mandar" devemos fazer:

RFmandar o empregado$pedir ao patrão$SFmandar$ENTER

Quando se está em modo de comando normal todo o texto batido será colocado na linha reservada ao editor. Após se encher a linha, e se continuar batendo, será deixado em branco da primeira coluna até a coluna correspondente a primeira posição de tabulação, e assim sucessivamente irá deixando em branco sempre até a próxima posição de tabulação. Os comandos serão guardados no "BUFFER" de comandos, que é do tamanho de um bloco (equivalente a uma tela de terminal.)

Quando o "BUFFER" de comandos estiver cheio e se tentar bater alguma coisa a mais será dada uma mensagem explicativa na linha reservada ao editor.

RECOMENDAÇÃO:

Como no modo normal o comando só será executado depois de inteiramente batido, recomenda-se fazer inserção no modo imediato onde os caracteres serão inseridos diretamente no local desejado.

Todos os comandos existentes no modo imediato são também disponíveis no modo normal. Quando se está em modo imediato pode-se alterar porções do texto, bastando para tal posicionar o cursor no local desejado e simplesmente começar a bater o novo texto. A cada tecla pressionada será apagado o caráter aonde está o cursor, escrito o caráter correspondente à tecla pressionada, e avançado o cursor de uma posição.

ATENÇÃO: nunca se poderá inserir texto dessa maneira, somente alterar um texto já existente.

Para se fazer inserção em modo imediato basta pressionar

CTRL I

e começar a entrar com o novo texto, que será colocado a partir da posição do cursor e deslocando todos os caracteres após o cursor (inclusive o apontado por ele) e até o fim do registro; até que seja pressionada a tecla ENTER.

Obs.: Quando se estiver usando o comando "INSERT" no modo imediato, aparecerá na linha reservada ao editor a palavra "INSERT".

Para se apagar caracteres em qualquer modo, basta se pressionar as teclas CTRL F, que causará a eliminação do caráter apontado pelo cursor e todos os caracteres seguintes a ele serão deslocados de uma posição à esquerda.

III - COMANDOS IMEDIATOS

Para melhor legibilidade convencionaremos que a tecla CTRL será representada por ↑.

1 - Relativos ao cursor

UP (↑U ou ↑) (*)

Faz com que o cursor "suba" uma linha, conservando a mesma coluna, caso esta coluna exista na linha anterior. Se a linha anterior for menor do que a atual, o cursor ficará na última coluna. Caso o cursor já se encontre na primeira linha da janela, o texto será "enrolado" para baixo e será mostrada a linha anterior, se esta estiver no "BUFFER"; senão será mostrada uma mensagem explicativa na linha reservada ao editor.

DOWN (↑D OU ↓) (*)

Faz com que o cursor "desça" uma linha, conservando a mesma coluna, caso esta coluna exista na linha seguinte. Se a linha seguinte for menor do que a atual, o cursor ficará na última coluna. Caso o cursor já se encontre na última linha da janela o texto será "enrolado" para cima e será mostrada a linha sequinte; se for detectado o final de arquivo, será mostrada uma mensagem explicativa na linha reservada ao editor.

LEFT (↑ L OU ←) (*)

Faz com que o cursor se desloque um caráter à esquerda. Se

o cursor estiver na primeira coluna de uma linha irá para a última coluna da linha anterior (se o cursor estiver na primeira coluna da primeira linha o efeito será semelhante a ↑U, só que o cursor ficará na última coluna).

RIGHT (↑R OU →) (*)

Faz com que o cursor se desloque um caráter à direita. Se o cursor estiver na última coluna de uma linha irá para a primeira coluna da linha sequinte (se o cursor estiver na última coluna da última linha o efeito será semelhante a ↑D, só que o cursor ficará na última coluna).

TOP (↑T)

Faz com que o cursor se desloque para o canto superior esquerdo da janela.

BOTTOM (↑B)

Faz com que o cursor se desloque para o canto inferior esquerdo da janela.

(*) os caracteres ↑, ↓, ←, → se referem a teclas especiais do terminal, se disponíveis.

2 - Relativos à tela

UP SCREEN (↑ -)

Troca a atual janela pela imediatamente anterior, que está no "BUFFER". O cursor ficará posicionado no canto superior esquerdo da janela. Caso se tente ultrapassar o início do "BUFFER" será mostrada uma mensagem explicativa na linha reservada ao editor.

DOWN SCREEN (↑ +)

Troca a atual janela pela imediatamente seguinte. O cursor ficará posicionado no canto superior esquerdo da janela. Caso se tente ultrapassar o final do arquivo será mostrada uma mensagem explicativa na linha reservada ao editor.

UP LINE (↑ X)

O texto será "enrolado" para baixo de uma linha. O cursor ficará posicionado no início da nova linha. Caso se tente ultrapassar o início do "BUFFER" será mostrada uma mensagem explicativa na linha reservada ao editor.

DOWN LINE (↑ Y)

O texto será "enrolado" para cima de uma linha. O cursor fi

cará posicionado no início da nova linha. Caso se tente ultrapas sar o final do arquivo será mostrada uma mensagem explicativa na linha reservada ao editor.

3 - Relativos ao controle

QUIT (↑ Q)

Termina a sessão de edição e não altera, nem cria arquivo al gum. Volta o comando do terminal ao monitor.

MODE (↑ M)

Troca de modo de comando, passando para modo de comando nor mal.

SAVE (↑ S)

Termina a sessão de edição, grava todos os arquivos necessá rios e volta o comando do terminal para o monitor.

4 - Relativos à edição

INSERT (↑ I)

Insere texto a partir da posição do cursor até que seja pres

sionada a tecla equivalente ao delimitador ou ENTER.

DELETE CHAR (↑ F (FORGET))

Apaga do arquivo o caráter apontado pelo cursor e desloca de uma posição à esquerda todos os caracteres seguintes ao caráter apagado.

DELETELINE (↑ K (KILL LINE))

Apaga do arquivo a linha atual (aquela aonde está o cursor) e desloca todas as linhas seguintes para cima. O cursor ficará posicionado no início da linha que era a seguinte, antes do comando ser dado.

DUPLICATE LINE (↑ P)

Duplica a linha atual e desloca todas as linhas seguintes para baixo. O cursor ficará posicionado no caráter seguinte ao último caráter da linha duplicada.

6 - Auxiliar

TAB (TAB)

Muda o cursor para a próxima posição de tabulação.

IV - COMANDOS NORMAIS

IMPORTANTE: Os parâmetros que forem omitidos serão considerados como sendo um. Cada comando deve ser seguido do delimitador. Os mnemônicos de cada comando são compostos de duas letras.

1 - Relativos ao cursor

UP CURSOR (UC [NO])

Faz com que o cursor "suba" NO linhas, conservando a mesma coluna, caso esta coluna exista na noésima linha anterior. Se a noésima linha anterior for menor do que a atual o cursor ficará na última coluna. Se necessário serão trazidas linhas do "BUFFER" (ver III 1 "UP").

DOWN CURSOR (DC [NO])

Faz com que o cursor "desça" NO linhas, conservando a mesma coluna caso esta coluna exista na noésima linha seguinte. Se a noésima linha seguinte for menor do que a atual o cursor ficará na última coluna. Se necessário serão trazidas linhas do arquivo principal (ver III 1 "DOWN").

LEFT CURSOR (LC [NO])

Faz com que o cursor se desloque NO caracteres à esquerda (ver III 1 "LEFT").

RIGHT CURSOR (RC [NO])

Faz com que o cursor se desloque NO caracteres à direita (ver III 1 "RIGHT").

TOP (TP)

Faz com que o cursor se desloque para o canto superior esquerdo da janela.

BOTTOM (BT)

Faz com que o cursor se desloque para o canto inferior esquerdo da janela.

2 - Relativos à tela

UP SCREEN (US [NO])

Troca a atual janela pela janela NO vezes anterior (ver III 2 "UP SCREEN"). O cursor ficará posicionado no canto superior esquerdo da janela.

DOWN SCREEN (DS [NO])

Troca a atual janela pela janela NO vezes seguinte (ver III 2 "DOWN SCREEN"). O cursor ficará posicionado no canto superior esquerdo da janela.

UP LINE (UL [NO])

O texto será "enrolado" para baixo de NO linhas (ver III 2 "UP LINE"). O cursor ficará posicionado no início da nova linha.

DOWN LINE (DL [NO])

O texto será "enrolado" para cima de NO linhas (ver III 2 "DOWN LINE"). O cursor ficará posicionado no início da nova linha.

3 - Relativos ao controle

QUIT (QT)

Termina a sessão de edição e não cria nem altera arquivo algum. Volta o comando do terminal ao monitor.

MODE (MD)

Troca de modo de comando, passando para modo de comando ime

diato. Depois desse comando não se pode dar mais nenhum comando normal, pois estes serão ignorados.

EXECUTE (ENTER) (**)

Executa os comandos normais que estão no "BUFFER" de comandos. Depois de executá-los, limpa o referido "BUFFER" e a linha reservada ao editor.

SAVE (SV)

Termina a sessão de edição, grava todos os arquivos necessários e volta o comando do terminal para o monitor.

4 - Relativos a edição

INSERT (IN STRING)

Insere texto a partir da posição do cursor até que seja pressionada a tecla relativa ao delimitador.

DUPLICATE LINE (DU [NO])

Insere NO cópias da linha atual, após a linha atual, e desloca todas as linhas seguintes. O cursor ficará posicionado no caráter seguinte ao último caráter da linha duplicada.

(**) O comando EXECUTE é fornecido através da tecla ENTER.

SEARCH BACKWARD (SB STRING)

Procura, a partir da posição do cursor para trás, a primeira ocorrência de STRING e coloca no meio da tela, se possível, a linha que a contém, O cursor ficará posicionado no caráter seguinte ao último caráter da STRING. A procura será feita somente até o início do "BUFFER".

SEARCH FORWARD (SF STRING)

Procura, a partir da posição do cursor para frente, a primeira ocorrência de STRING e coloca no meio da tela, se possível, a linha que a contém. O cursor ficará posicionado no caráter seguinte ao último caráter da STRING. A procura será feita até o final do arquivo.

REPLACE BACKWARD (RB S1 DELIMITADOR S2)

Procura, a partir da posição do cursor para trás, a primeira ocorrência de S1 e substitui por S2 (ver IV 4 SEARCH BACKWARD). O cursor ficará posicionado no caráter sequinte ao último caráter de S2.

REPLACE FORWARD (RF S1 DELIMITADOR S2)

Procura, a partir da posição do cursor para frente, a primeira ocorrência de S1 e substitui por S2 (ver IV 4 SEARCH FORWARD). O cursor ficará posicionado no caráter seguinte ao último caráter de S2.

DELETE LINE (DL [NO])

Apaga, a partir da linha atual inclusive, NO linhas (ver III 4 "DELETE LINE"). O cursor ficará posicionado no início da linha que era a seguinte, antes do comando ser dado.

DELETE CHAR (DC [NO])

Apaga, a partir da posição do cursor inclusive NO caracteres (ver III 4 "DELETE CHAR").

LINE BLANK (LN [NO])

Insere NO linhas em branco após a linha atual. O cursor ficará posicionado no caráter seguinte ao último caráter inserido.

COPY (CP [NO] LINE)

Copia NO linhas a partir da linha atual, inclusive, após a linha de número LINE. As linhas originais são apenas copiadas em outro lugar, mas não apagadas. Elas permanecem no arquivo. O cursor

ficará posicionado no caráter seguinte ao último caráter copiado.

MOVE (MV [NO] LINE)

Copia NO linhas a partir da linha atual, inclusive, após a linha de número LINE. As linhas originais são apagadas do arquivo. O cursor ficará posicionado no caráter seguinte ao último caráter copiado.

6 - Auxiliares

TABULATE (TB COL,...)

Define as novas posições de tabulação. As posições anteriores não são mais válidas.

REPEAT (RE [NO](COMANDO1$...COMANDON$))

Executa a sequência de comandos COMANDO1 até COMANDON NO vezes. Se NO for "*" repete os comandos até o final do arquivo ou o início do "BUFFER".

DEFINE (DF STRING (COMANDO1$...COMANDON$))

Define um "novo comando" de nome STRING que conterá a sequência de comandos COMANDO1,...COMANDON. A partir daí STRING poderá

ser usado como um comando normal, só que precedido do caracter "@". Por exemplo:

Para se definir o comando "BUSCA.ANCORADA":

DFBUSCA.ANCORADA(A+$SFPRISAO$)

Para se usar o comando "BUSCA.ANCORADA":

@BUSCA.ANCORADA (em estado de comando normal)

ANCHOR ON (A+ NO)

A partir deste comando todos os comandos de busca serão feitos ancorados (***) à noésima coluna.

ANCHOR OFF (A-)

A partir deste comando todos os comandos de busca serão feitos livremente em todas as colunas.

DELIMITER CHAR (DM CHAR)

(***) Ancorar uma busca a uma dada coluna significa fazer a busca começando sempre por essa coluna.

O delimitador padrão de DOTE é $. Este comando muda o delimitador para CHAR. Logo após DM deverá vir somente um caráter, que será o novo delimitador, seguido do delimitador atual. Esta será a última vez que o delimitador atual é válido. Por exemplo:

DM*$

Faz com que o novo delimitador seja *.

--0--

INDICE REMISSIVO

ANEXO 3

CONSTANTES E VARIÁVEIS MAIS IMPORTANTES DO EDITOR

CONSTANTES E VARIÁVEIS MAIS IMPORTANTES DO EDITOR

CONSTANTES:

```
APITØ       =   2    ; para soar a campainha do teclado do terminal Ø põe 2 no "port" do terminal Ø
CLOCK       =   8    ; constante que indica se a atual interrupção foi causada pelo relógio
CLOCPORT    =   2    ; número do "port" do relógio
CNEB                 ; endereço físico do "endereço base" do "buffer" de comandos normais
CNEL                 ; endereço físico do "endereço limite" do "buffer" de comandos normais
CONJTERM    =  1Ø    ; constante que indica que a atual interrupção foi causada por um dos quatro terminais
FUNCSELECT  =   1    ; código que indica que a tecla auxiliarFUNC SELECT estava presionada
INPEB                ; endereço físico do "endereço base" do "buffer" do arquivo entrada
INPEL                ; endereço físico do "endereço limite" do "buffer" do arquivo entrada
INTENDER    =   9    ; endereço físico da instrução de salto para a rotina de interrupções
INTPORT     =   Ø    ; número do "port" que vai indicar qual "periférico" interrompeu
```

```
IOCTL       =   2     ; código que indica que a tecla auxiliar
                        "I/O CTRL", estava pressionada
NOCARLIN    =  64     ; número de caracteres de uma linha do termi
                        nal
NOLNTC      =   2     ; número de linhas que o "buffer" do tecla-
                        do ocupa
NOLNTELA    =   8     ; número de linhas da tela do terminal
NOTLCN      =   1     ; número de telas que o "buffer" de comandos
                        normais ocupa
OUTEB                 ; endereço físico do início do "buffer"  de
                        arquivo saída
OUTEL                 ; endereço físico do fim do "buffer" de  ar
                        quivo saída
PISKTIME    =   8     ; tempo (em "clocks"  do relógio) de espera pa
                        ra por e tirar o cursor da tela
RPTLAT      =   8     ; tempo de latência (em "clocks" do relógio
                        interno) para início de repetição automáti
                        ca de teclas
SHIFT       =   Ø     ; código que indica que a tecla auxiliar
                        "SHIFT" estava pressionada
SJTM        = 3xNOLNTELA  ; número de linhas da super janela
STTM        = 512     ; número de caracteres de um setor do disco
                        fixo
TCEB        =         ; endereço físico do início do "buffer"  do
                        teclado
```

TCEL ; endereço físico do fim do "buffer" do teclado

TCTM = (NOLNTC x NOCARLIN)/2 ; número de "teclas" que o "buffer" do teclado comporta. CADA TECLA OCUPA 2 BYTES)

TECPORT = 3F hexadecimal ; número do "port" dos teclados

TERMØ = Ø ; código do terminal zero

TERMPORT = 3E hexadecimal ; número do "port" dos terminais: indica qual dos quatro interrompeu

TLTM = 512 ; número de caracteres de tela do terminal

VARIÁVEIS

	Nome	Nº bytes que ocupa	Descrição
	CARCT	2	; conta nº caracteres inseridos que ainda não completaram NOCARLIN
(*)	CNAF	2	; apontador para "buffer" de comandos normais
	CNBF	TLTM	; "buffer" de comandos normais
(*)	CRSAUX	1	; contém o caráter apontado pelo cursor para auxiliar a simula "piscar" o cursor
(*)	CRSCT	1	; contador que indica quando deve por o cursor na tela ou o caráter apontado por ele

	Nome	Nº bytes que ocupa	Descrição
(*)	CRSEND	2	; endereço do cursor na tela
(*)	ESTADO	1	; indica o estado atual do editor (Imediato ou Normal)
	FIMREG	1	; aponta para a última linha do registro atual, quando da inserção de caracteres
	INPAF	2	; endereço do último "byte" ocupado no arquivo entrada
(*)	LINATUAL	1	; deslocamento em relação ao início da super janela da linha atual
(*)	LNCONTVET	SJTM	; vetor que contém indicação se uma dada linha é continuação ou não da anterior
(*)	LNTMVET	SJTM	; vetor que contém o tamanho de cada linha da super janela
(*)	MAPLNVET	2xSJTM	; vetor que contém o endereço do início de cada linha de super janela
(*)	OUTAF	2	; endereço do ultimo "byte" ocupado no arquivo saída
(*)	POSL1	1	; deslocamento em relação ao início da super janela da 1ª linha da tela
(*)	POSL8	1	; semelhante ao anterior para a úl-

Nome	Nº bytes que ocupa	Descrição
		tima linha da tela
RGATUAL	1	; aponta para o início do registro atual
RGTMVET	2* SJTM	; vetor que contém o tamanho de cada registro da super janela
RPTAUX	1	; auxiliar para repetição automática de teclas: contém o último caráter lido
RPTTIME	1	; auxiliar para repetição automática de teclas: para esperar o tempo de latência
(*) SJCT	1	; indica 1ª linha livre da super janela
SYSINTER	2	; guarda o endereço da rotina de interrupção do sistema
TCBF	2 x NOCARLIN	; "BUFFER" de teclado
(*) TCRI	1	; aponta para o início do "buffer" de teclado (deslocamento em relação à base)
(*) TCRF	1	; semelhante ao anterior para o fim do "buffer"

(*) Esta variável tem um valor inicial pré-definido.

ANEXO 4

ROTINA "ENROLAR PARA CIMA" EM LINGUAGEM DE MONTAGEM DO INTEL 8080

```
;*****************************************************************
;ENROLAR PARA CIMA
;SE POSSIVEL A PRIMEIRA LINHA DA TELA VAI PARA A SUPERJANELA
;E SOBE TODAS AS LINHAS SEGUINTES
;USA OS REGISTRADORES A,B,C,D,E,H,L
;A.....AUXILIAR
;B,C...AUXILIAR
;D,E...AUXILIAR
;H,L...AUXILIAR
;*****************************************************************
ENRCIM: LDA  LINATUAL
        MOV  B,A
        LDA POSL8
        SUB  B
        INR  A
        STA  ACB1          ;ACB1:=POSL8-LINATUAL+1
        LDA    SJCT
        MOV     B,    A
        LDA    POSL8
        INR     A
        CMP     B          ; SE POS.LIN.B + 1 = SJCT
        JNZ    CS1
        MVI     A, SJTM          ;ENTAO INICIO
        INR     A
        CMP     B                ;SE SJCT = SJTM + 1
        JNZ    CS2
        CALL POUTBF                 ;ENTAO POR.NO.ARQBF.OUT
CS2:    MVI     B,    1
        LDA     SJCT
        MOV     C,   A
        CALL TINPBF                 ;TRAZER.DE.ARQIBF(1,SJCT)
CS1:    LDA   POSL1                 ;FIM
        MOV     L,    A
        LDA  SJCT          ;SE POSL1=SJCT
        CMP  L                 ;ENTAO RETORNE
        RZ

        MVI     H,    0
        DCX     H
        DAD     H
        LXI    D, MAPLNVET
        DAD     D
        MOV     C,    M
        INX     H
        MOV     B,    M    ; B:=MAPLNVET(POS.LIN.1)
        PUSH    B
        MVI    L,    SJTM
        MOV     L,    A
        MVI     H,    0
        DAD     H
                           ;ID:=ENDER DE MAPLNVET)
        DAD   D
        MOV    E,    M
        INX    H
        MOV    D,    M     ; D:=MAPLNVET(SJCT)
        LDA  POSL1         ;SE  POSL1=LINATUAL
```

```
        MOV  H,A
        LDA LINATUAL
        CMP  H
        JNZ  CS3                ;ENTAO ATUALIZA CARAC APONT.PELO CURSOR
        DI
        LHLD CRSEND
        LDA  CRSAUX
        MOV  M,A
        LDAX B
        STA  CRSAUX
        XCHG
        SHLD CRSEND             ;ATUALIZA VAR.PISCAR CURSOR
        XCHG
CS3:    MVI  H,NOCARLIN
                            ;PARA CADA CARACTER H:=NO.CARAC.LINHA, 1, -1  FACA
CFOR1:  LDAX    B               ;INICIO
        STAX    D               ;SJ(D) := SJ(B)
        INX     D               ;D := D+1
        INX     B               ;B := B+1
        DCR     H
        JNZ  CFOR1              ;FIM
        LDA  POSL1
        MOV     E,    A
        LDA  LINATUAL
        CMP     E
        JNZ     CSAI1
        EI
CSAI1:  POP     D
                            ;D := MAPLNVET(POSLIN1)
                            ;B := MAPLNVET( POS.LIN1  +  1) ;
        MVI     H,NOLNTELA-1;PARA CADA LINHA H :=NOLNTELA-1, 1, -1 FACA
CFOR2:  MVI     L,NOCARLIN      ;INICIO
        LDA  ACB1
        CMP  H                  ;SE NECESSARIO ATUALIZA CARAC APONT. CURSOR
        JNZ  CFOR3                      ;E ATUALIZA VAR.PISCAR O CURSOR
        DI
        LHLD CRSEND
        LDA  CRSAUX
        MOV  M,A
        LDAX B
        STA  CRSAUX
        XCHG
        SHLD CRSEND
        XCHG
        LDA  ACB1
        MOV  H,A
        MVI    L, NOCARLIN
CFOR3:  LDAX    B               ;PARA CADA CARACTER L := NO.CARAC.LINHA, 1 , -1 FACA
        STAX    D               ;INICIO
        INX     D               ;SJ(D) := SJ(B)
        INX     B               ;B := B +1
        DCR     L               ;D := D +1
        JNZ  CFOR3              ;FIM
        LDA  ACB1               ;D:=ENDER MAPLNVET(POS.LIN1 + H )
        CMP  H
        JNZ  CSAI2
```

```
        EI                      ;B:=ENDER MAPLNVET( POSLIN1 + H + 1)

CSAI2:  DCR     H
        JNZ     CFOR2           ;FIM
        LDA     POSL8
        MOV     L,      A       ;PARA A ULTIMA LINHA DA TELA FACA
        LDA     SJCT                    ;SE POSL8(=SJCT
        DCR     A                       ;ENTAO POE BRANCO NA ULTIMA LIN
        DCR     A
        CMP     L
        JP      CS4
        MOV     C,A
        MVI     B,0
        LXI     H,MAPLNVET
        DAD     B
        DAD     B
        MOV     E,M
        INX     H
        MOV     D,M
        MVI     L,NOCARLIN
        MVI     A,' '
CFOR5:  STAX    D
        INX     D
        DCR     L
        JNZ     CFOR5
        JMP     CSAI5
CS4:    MVI     H,      0
        DAD     H
        LXI     B,      MAPLNVET
        DAD     B
        MOV     C,      M
        INX     H
        MOV     B,      M
        MVI     L,      NOCARLIN        ;PARA CA CARACTER I:=1,NOCARLIN FACA
CFOR0:  LDAX    B                       ;INICIO
        STAX    D                       ;SJ(D):=SJ(B)
        INX     D                       ;D:=D+1
        INX     B                       ;H:=H+1
        DCR     L
        JNZ     CFOR0                   ;FIM
CSAI5:  LXI     D,      MAPLNVET
        MVI     H,      0
        LDA     POSL8
        MOV     L,A
        DAD     H
        DAD     D
        MOV     C,      M
        INX     H
        MOV     B,      M
        PUSH    B               ;SALVA CONTEUDO DE MAPLNVET(POSL8+1)
        MOV     D,      H
        MOV     E,      L
        DCX     D
        DCX     D               ;D:=ENDER DE  MAPLNVET ( POS.LIN8 )
        MVI     B,NOLNTELA      ;PARA CADA LINHA B:=NOLNTELA,1 ,-1 FACA
CFOR4:  LDAX    D                       ;INICIO
```

```
MOV     M,    A       ;MAPLNVET(H):=MAPLNVET(D)
DCX     H
DCX     D
LDAX    D
MOV     M,    A
DCX     H             ;H:=H-1
DCX     D             ;D:=D-1
DCR     B
JNZ     FOR4          ;FIM
XCHG

LXI     H,MAPLNVET
MVI  C,SJTM
MVI     B,    D
DAD     B
DAD     B
INX     H
MOV     A,    M
STAX    D
DCX     D
DCX     H
MOV     A,    M
STAX    D             ;MAPLNVET(D) := MAPLNVET(SJCT)

POP     B
MOV     M,    C
INX     H
MOV     M,    B       ;MAPLNVET(SJCT):=MAPLNVET(POS.LINB+1)
LDA     POSL1
INR     A
STA     POSL1         ;POS.LINHA1:=POS.LINHA1+1
LDA     POSL8
INR     A
STA     POSL8         ;POSL8:=POSL8+1
LXI  H,MAPLNVET
LDA  LINATUAL
MOV  C,A
MVI  B,0
INR  A
STA  LINATUAL
DAD  B
DAD  B
MOV  E,M
INX  H
MOV  D,M
DI
LHLD CRSEND
LDA  CRSAUX
MOV  M,A
XCHG
MOV  A,M
MVI  M,CURSOR
STA  CRSAUX
SHLD CRSEND           ;ATUALIZA VARIAVEIS PARA PISCAR O CURSOR
EI
RET
```

ANEXO 5

ROTINA "INSERT" EM LINGUAGEM DE MONTAGEM DO INTEL 8080

```
;*****************************************************************
;SE POSSIVEL INSERE CARACTERES EM UM REGISTRO JA' EXISTENTE
;OU COMECA A INSERIR CARACTERES EM UM NOVO REGISTRO
;OS CARACTERES A DIREITA DO CURSOR SERAO DESLOCADOS CONVENIENTEMENTE
;USA OS REGISTRADORES A,B,C,D,E,H,L COMO AUXILIARES

;USA AS ROTINAS:
;TTCBF:      'PEGA' CHAR DO TECLADO
;TCHBF:      'PEGA' CHAR DO BUFFER DE COMANDOS NORMAIS
;ATUTM:      ATUALIZA RGTMVET(RGATUAL) E LNTMVET(FIMREG)
;PSJ(B):     ABRE, APOS A LINHA B, UMA LINHA EM BRANCO NA JANELA
;CMP(D,E;H,L) COMPARA DUAS PALAVRAS
;ATUCONT(C)  FAZ LNCONTVET(FIMREG+1):=C
;DSUB(B,C;D,E) FAZ A SUBTRACAO DE DUAS PALAVRAS
;TRNCAR      TRANSFERE L CARACTERES A PARTIR DE AIWPEG PARA A PARTIR DE AIWPG
;ENRCIM      ENROLA PARA CIMA
;ATUAIW1     ATUALIZA AIW1 COM O ENDERECO DA ULTIMA POS. DO REG. ATUAL
;POUTBF      POE O 1. REG.DA SUPERJANELA NO BUFFER D E SAIDA
;E AS ROTINAS DE COMANDOS DO CURSOR
;*****************************************************************
INSERT: LXI  H,LNCONTVET  ;(<<<< INICIALIZACAO DO COMANDO INSERT <<<<)
        LDA  LINATUAL
        DCR  A
        MOV  E,A          ;D,E:=LINATUAL
        MVI  D,0          ;
        DAD  D            ;H,L:=ENDER LNCONTVET(LINATUAL)
IE01:   MOV  A,M          ;ENQUANTO LNCONTVET(H)=1 (=> E' CONTINUACAO =>) FACA
        CPI  0                ;INICIO
        JZ   ISA1             ;
        DCX  H                ;H:=H-1
        DCX  D                ;D:=D-1
        JMP  IE01             ;FIM
ISA1:   LXI  H,RGTMVET    ;
        DAD  D            ;
        DAD  D            ;
        MOV  A,E          ;
        STA  RGATUAL      ;RGATUAL:=D
        MOV  C,M          ;
        INX  H            ;
        MOV  B,M          ;
        PUSH B            ;B:=RGTMVET(RGATUAL)
        MOV  A,B
        XRA  C
        LDA  LINATUAL
        MOV  E,A
        JZ   ISA2         ;
        MOV  E,A          ;
        LXI  H,LNCONTVET  ;
        DAD  D            ;
IE02:   LDA  SJCT         ;
        CMP  E            ;
        JZ   ISA2         ;
        MOV  A,M          ;
        CPI  0            ;
        JZ   ISA2         ;
        INX  H            ;
```

```
        INX  D          ;
        JMP  IE02       ;
ISA2:   DCR  E          ;
        MOV  A,E        ;FIMREG:=ULTIMA LINHA DO REG.
        STA  FIMREG     ;
        MOV  A,C        ;
        XRA  B          ;SE RGTMVET=0
        JNZ  IS0            ;ENTAO INICIO
        MVI  A,NOCARLIN         ;
        STA  AIB1               ;AIB1 :=NOCARLIN (* NO.CARAC.LIVRE ULT.LINHA REG. *)
        XRA  A                  ;
        STA  CARCT              ;CARCT:=0 (* NO.CARAC.INSERIDOS NAO CONTABILIZADOS *)
        LXI  B,0                ;B:=0
        LHLD CRSEND             ;H:=CRSEND-1
        DCX  H                  ;
        LDA  CRSAUX
        CPI  RUBOUT             ;SE TEM RS
        MVI  A,0                   ;ENTAO B:=1
        JNZ  IS02                  ;      AIB2:=1.
        INR  A                     ;
        INX  B                     ;
        INX  H                     ;      H:=H+1
        JMP  IS04                  ;
ISQ2:   PUSH PSW                   ;SENAO AIB2:=0
        LDA  SJCT                  ;      SJCT:=SJCT+1
        INR  A                     ;
        STA  SJCT                  ;
        POP  PSW                   ;
ISQ4:   STA  AIB2                  ;(***TELA SEMPRE FAZ PARTE DA SJ***)
        JMP  VEIN               ;FIM
IS0:    MOV  A,B
        CPI  1
        JNZ  IS06
        MOV  A,E
        CPI  STTM-1
        JNZ  IS06
        MVI  A,APITO
        OUT  TECPORT
        POP  B
        RET
IS06:   PUSH B              ;SENAO INICIO
        LXI  D,NOCARLIN         ;ENQUANTO B)NOCARLIN FACA
IE03:   MOV  H,B                   ;INICIO
        MOV  L,C                   ;
        CALL CMPW                  ;
        JP   ISA3                  ;
        CALL DSUB                  ;B:=B-NOCARLIN
        JMP  IE03                  ;FIM
ISA3:                           ;
        MOV  A,E                ;
        SUB  C                  ;
        STA AIB1                ;AIB1 :=NO.POS LIVRES NA ULT.LINHA DO REG.
        LXI  H,MAPLNVET         ;
        LDA  RGATUAL            ;
        MOV  E,A                ;
        MVI  D,0                ;
```

```
        DAD  D                  ;
        DAD  D                  ;
        MOV  E,M                ;
        INX  H                  ;
        MOV  D,M                ;D:=MAPLNVET(RGATUAL)
        POP  H                  ;H:=RGTMVET (RGATUAL)
        DAD  D                  ;H:=H+D (** POS. DO FIM DO REG.**)
        MOV  C,L                ;
        MOV  B,H                ;
        LHLD CRSEND             ;
        XCHG                    ;
        CALL DSUB               ;
        PUSH B                  ;B:=NO.CARAC DEPOIS DO CURSOR
        LDA  AIB2
        MOV  C,A
        MVI  A,NOCARLIN
        SUB  C
        MOV  C,A
        MVI  B,0
        LDA  FIMREG
        MOV  E,A
        MVI  D,0
        LXI  H,MAPLNVET
        DAD  D
        DAD  D
        MOV  A,M
        INX  H
        MOV  H,M
        MOV  L,A
        DAD  B                  ;H:=POS.FIM.REG.
        POP  B          ;
        MVI  H,RUBOUT           ;
        INX  B                  ;
         MVI  A,1
         STA  AIB2              ;AIB2:=1
        XRA  A                  ;CARCT:=0
        STA  CARCT              ;FIM (<<<< DA INICIALIZACAO DE INSERT<<<<)
VEIN:                   ;ENQUANTO NAO FOR DELIM FACA

        LDA  ESTADO     ;SE ESTADO = 'I'
        CPI  'I'        ;     ENTAO TTCBF
        JZ   INSI       ;     SENAO TCNBF
        CALL TCNBF      ;
        JMP  INSNJA     ;
INSI:   CALL TTCBF       ;(** PEGA DO BUFFER DO TECLADO EM D,E **)
INSNJA: MOV  A,E         ;VE SE FUNCSELECT NAO ESTAVA APERTADA
        CPI  FUNCSELECT     ;ENTAO
        JZ   IS1                ;
        MOV  A,D                ;SE CARC E' ALFANUMERICO
        CPI  ' '                    ;ENTAO
        JM   ISB0                       ;
        CPI  RUBOUT                     ;
        JP   IS90                       ;
        PUSH D                          ;
        LDA  AIB1
                                        ;AIB1 :=AIB1 -1
```

```
        LDA   CARCT
        INR   A
        STA   CARCT
        LDA   AIB1
        DCR   A
        STA   AIB1
        CPI   0
        JNZ   IS2
        PUSH  B
        PUSH  H
        CALL  ATUTM
        CPI   0
        JNZ   FIM5
        LDA   SJCT
        MOV   B,A
        LDA   FIMREG
        INR   A
        INR   A
        CMP   B
        JF    IS12

        DCR   A
        MOV   B,A
        CALL  PSJ
        CPI   0
        JZ    FIM6
        JMP   IS14
IS12:   DCR   A
        MOV   B,A
        MVI   A,SJTM
        CMP   B
        JNZ   IS16
        CALL  PSJ
        CPI   0
        JZ    FIM6
        JMP   IS14
IS16:   LHLD  CRSEND
        XCHG
        LHLD  ENDFIMTELA
        CALL  CMPW
        JZ    IS18
ISSY:   LDA   SJCT
        INR   A
        STA   SJCT
IS18:   LDA   FIMREG
        INR   A
        MOV   C,A
        MVI   B,0
        LXI   H,MAPLNVET
        DAD   B
        DAD   B
        MOV   C,M
        INX   H
        MOV   B,M
        MVI   A,' '
        MVI   H,NOCARLIN
```

```
;CARCT:=CARCT+1

;SE NO.CARAC.LIVRE = 0

    ;ENTAO
          ;ATUALIZA LNTMVET E RGTMVET
          /
          ;SE RGTMVET(RGATUAL) = STTM


              ;ENTAO VA P/ FIM5  (**REG.CHEIO**)
          }
          }

          ;B:=FIMREG+1
          ;SE NAO E' ULT.REG.JANELA

            ;ENTAO PSJ(B,E) (* ATUALIZA LNCONTVET*)




              ;SENAO SE NAO VAI PRECISAR ENROLAR
                 ;ENTAO SJCT:=SJCT+1
```

```
IS22:   STAX B
        INX  B
        DCR  H
        JNZ  IS22
IS14:   MVI  C,1                        ;ATUCONT(1)
        CALL ATUCONT
        LDA  FIMREG
        INR  A
        STA  FIMREG
        MOV  C,A
        MVI  B,0
        LXI  H,MAPLNVET
        DAD  B
        DAD  B
        MOV  A,M
        INX  H
        MOV  H,M
        MOV  L,A
        SHLD AIWPO
        SHLD AIW1
        MVI  A,NOCARLIN                 ;
        STA  AIB1                       ;AIB1 :=NOCARLIN
        POP  H                          ;
        POP  B                          ;L:=0
        SHLD AIWPEG                     ;ATUALIZA AIWPEG E AIWPO
        MOV  A,B
        XRA  C
        JNZ  IS24
        INX  H
        SHLD AIW1
IS24:   MVI  L,0
        JMP  IS116
IS2:    SHLD AIWPEG                     ;SENAO AIWPEG:=H,L
        INX  H                          ;AIWPO:=AIWPEG+1
        SHLD AIWPO                      ;AIW1:=AIWPO
        SHLD AIW1
        LDA  LINATUAL
        MOV  L,A
        LDA  FIMREG
        INR  A
        CMP  L
        JNZ  IS119
        MOV  L,C
        JMP  IS116
IS119:  LDA  AIB1                       ;L:=MIN(B,C E NOCARLIN-AIB1)
        MOV  L,A
        MVI  A,NOCARLIN
        SUB  L
        MOV  L,A
IS116:  LDA  FIMREG                     ;
        STA  AIB6                       ;AIB6:=FIMREG
        PUSH B
IEQ40:  MOV  A,B                        ;ENQUANTO B,C # 0 FACA
        XRA  C
        JZ   IS440
        MVI  H,0
```

```
        XCHG
        CALL DSUB
        XCHG
        CALL TRNCAR
        MOV  A,B
        XRA  C
        JZ   ISA40
        LDA  AIB6
        DCR  A
        STA  AIB6
        LXI  H,MAPLNVET
        MOV  E,A
        MVI  D,0
        DAD  D
        DAD  D
        MOV  E,M
        INX  H
        MOV  D,M
        LXI  H,NOCARLIN-1
        DAD  D
        DCX  B
        MOV  A,B
        XRA  C
        JNZ  IS127
        LDA  CRSAUX
        JMP  IS126
IS127:  MOV  A,M
IS126:  DCX  H
        SHLD AIWPEG
        XCHG
        LHLD AIWPO
        MOV  M,A
        XCHG
        INX  H
        SHLD AIWPO
        MVI  L,NOCARLIN-1
        MOV  A,B
        CPI  0
        JNZ  IE040
        MOV  A,C
        CMP  L
        JP   IE040
        MOV  L,C
        JMP  IE040
ISA40:  POP  B
        POP  D
        LHLD CRSEND
        DI
        MOV  M,0
        INX  H
        XCHG
        LHLD ENDFIMTELA
        INX  H
        CALL CMPW
        XCHG
        PUSH B
```

```
;B,C:=B,C-L

;TRNCAR (**TRANSFERE L CARAC**)


;SE B,C# 0
  ;ENTAO AIB6:=AIB6-1
 }
 }
 }
 }
 }
 }
 }
 }
 }
        ;CALCULA NOVO AIWPEG

 }
 }
 }
 }
 }
 }
        ;SJ(AIWPO):=SJ(AIWPEG)
        ;L:=NOCARLIN-1
 }
        ;AIWPO:=AIWPEG

        ;AIWPEG:=AIWPEG-1
}
}

}

;POE O CARAC LIDO NA TELA

;SE ESTA NO FIM TELA
  ;ENTAO ENRCIM
```

```
        JNZ   IS12D
        DCX   H
        MOV   A,M
        STA   CRSAUX
        CALL  ENRCIM
        CALL  ATUAIW1
        LDA   AIB1
        CPI   NOCARLIN
        JNZ   IS31
        LDA   SJCT
        INR   A
        STA   SJCT
IS12D:  SHLD  CRSEND                    ;
        MVI   M,CURSOR                      ;ATUALIZA VARIAEIS P/PISCAR O CURSOR
        EI                              ;
        LXI   H, MAPLNVET
        LDA   LINATUAL
        MOV   E,A
        MVI   D,0
        DAD   D
        DAD   D
        MOV   E,M
        INX   H
        MOV   D,M
        LHLD  CRSEND
        CALL  CMPW                          ;SE CRSEND=MAPLNVET(LINATUAL)
        JNZ   IS31
        LDA   LINATUAL                          ;ENTAO LINATUAL:=LINTUAL+1
        INR   A
        STA   LINATUAL
IS31:   POP   B                         ;
        LHLD  AIW1                              ;H:=POS.FIM REG.
                                                ;B:=NO.CARAC.DEPOIS DO CURSOR
        JMP   VEIN                      ;
IS9D:   PUSH  B
        PUSH  H                     ;SENAO
        CPI   RIGHT                     ;SE E' RIGHT
        JZ    IS4                           ;ENTAO VA P/ (** /R **)
        CPI   LEFT                      ;SENAO SE E' LEFT
        JZ    IS3                           ;ENTAO VA P/ (** /L **)
        CPI   SKIP                          ;SENAO SE E' (** RS **)
        JNZ   IS1D                            ;ENTAO
RSCHAR: PUSH  B
        MOV   A,M
        CPI   RUBOUT
        MVI   A,0
        JNZ   RS1
        INR   A
RS1:    STA   AIB2                                  ;AIB2:=1 SE TEM RS.O C.C.
        LDA   LINATUAL
        MOV   E,A
        MVI   D,0
        LDA   POSLB
        CMP   E
        JNZ   IS034
        LHLD  ENDFIMTELA
```

```
        INX  H
        MOV  B,H
        MOV  C,L
        JMP  IS035
IS034:  LXI  H,MAPLNVET
        DAD  D
        DAD  D
        MOV  C,M
        INX  H
        MOV  B,M
IS035:  LHLD CRSEND
        XCHG
        MOV  H,B
        MOV  L,C
        CALL DSUB
        MOV  A,C
        STA  AIB3
        MOV  E,C
       MVI  B,0
        LDA  FIMREG
        MOV  C,A
        LDA  LINATUAL
        DCR  A
        CMP  C
        JP   IS30
        LDA  AIB1
        CMP  E
        JM   IS30
        LDA  AIB3
         MOV  C,A
        MVI  A,NOCARLIN
        SUB  C
        MOV  C,A
        LXI  H,LNTHVET
        LDA  LINATUAL
        MOV  E,A
        MVI  D,0
        DCR  E
        DAD  D
        MOV  M,C
        LXI  H,LNTHVET
        LDA  FIMREG
        MOV  E,A
        DAD  D
        LDA  AIB3
        MOV  B,A
        LDA  AIB1
        MOV  C,A
        MVI  A,NOCARLIN
        SUB  C
        ADD B
        MOV  M,A
        POP  B
        POP  H
        PUSH H
        PUSH B
```

```
;SE FIMREG)LINATUAL
  ;ENTAO
    ;E:=NO.CARAC.ATE.FIM.LINHA
    ;SE AUXB1)=C
       ;ENTAO

          ;(**** CABE ULT.LINHA ****)









          ;LNTHVET(LINATUAL) E ATUALIZADA









          ;LNTHVET(FIMREG) E ATUALIZADA
```

```
        SHLD AIWPEG
        LDA  AIB3
        MOV  E,A
        MVI  D,0
        DAD  D
        SHLD AIWPO
        SHLD AIW1
        LDA  AIB1
        MOV  L,A
        MVI  A,NOCARLIN
        SUB  L
        MOV  L,A
        LDA  AIB2
        CPI  0
        JZ   IS104
        INR L
IS104:  LDA  FIMREG
        STA  AIB6
        MOV  A,L
        STA  AIB5
IE65:   MOV  A,B
        XRA  C
        JZ   ISA5
        MVI  H,0
        XCHG
        CALL DSUB
        XCHG
        CALL TRNCAR
        MOV  A,B
        XRA  C
        JZ   ISA5
        LDA  AIB6
        DCR  A
        STA  AIB6
        LXI  H,MAPLNVET
        MOV  E,A
        MVI  D,0
        DAD  D
        DAD  D
        MOV  E,M
        INX  H
        MOV  D,M
        LXI  H,NOCARLIN-1
        DAD  D
        SHLD AIWPEG
        PUSH H
        LDA  AIB3
        MOV  E,A
        MVI  D,0
        CALL DSUB
        XCHG
        CALL TRNCAR
        POP  H
        SHLD AIWPO
        LDA  AIB5
        MOV  L,A
```

```
;ATUALIZA AIWPEG
;ATUALIZA AIWPO

;SE TEM RS

   ;ENTAO L:=CARCT+1
  ;SENAO L:=CARCT

;AIB6:=FIMREG

;AIB5:=L
;ENQUANTO B,C # 0 FACA


   ;B,C:=B,C-L

   ;TRNCAR

   ;SE  B,C # 0

      ;ENTAO AIB6:=AIB6-1


                  ;CALC.NOVO AIWPEG


                  ;B,C:=B,C-L

                   ;TRNCAR

                  ;CALC.NOVO AIWPO

                   ;L:=AIB5
```

```
        MOV  A,B
        CPI  0
        JNZ  IE45
        MOV  A,C
        CMP  L
        JP   IE45
        MOV  L,C
        JMP  IE45
ISA5:   LDA  LINATUAL
        MOV  C,A
        LDA  POSLB
        CMP  C
        JNZ  IS106
        MVI  A,RUBOUT
        STA  CRSAUX
        CALL ENRCIM
        CALL ATUAIW1
        LDA  LINATUAL
        DCR  A
        STA  LINATUAL
        JMP  IS116
IS106:  LXI  H,MAPLNVET
        LDA  LINATUAL
        MOV  C,A
        MVI  B,0
        DAD  B
        DAD  B
        MOV  C,M
        INX  H
        MOV  B,M
        DI
        LHLD CRSEND
        MVI  M,RUBOUT
        MOV  H,B
        MOV  L,C
        SHLD CRSEND
        MVI  M,CURSOR
        EI
IS116:   LDA  LINATUAL
        MOV  C,A
        MVI  B,0
        LXI  H,LNCONTVET
         DAD  B
         MVI  M,0
ISA82:  LXI  H,RGTMVET
        LDA  RGATUAL
        MOV  E,A
        MVI  D,0
        DAD  D
        DAD  D
        LDA  CARCT
        MOV  E,A
        MOV  C,M
         INX  H
         MOV  B,M
        XCHG
```

```
                                ;SE L>B,C
                                  ;ENTAO L:=B,C

;SE LINATUAL = POSLB

     ;ENTAO ENRCIM

;B,C:=NOVA POS.CURSOR

;ATUALIZA VAR.P/PISCAR O CURSOR

;LNCONTVET(LINATUAL+1) E ATUALIZADO

;ATUALIZA RGTMVET(RGATUAL)
```

```
        DAD  E
        MOV  B,H
        MOV  C,L
        XCHG
        POP  D
        CALL DSUB
        LDA  AIB2
        CPI  0
        JZ   RS2
        INX  B
        DCX  D
RS2:    MOV  M,B
        DCX  H
        MOV  M,C

         LDA  LINATUAL
        STA    RGATUAL
        INR  A                          ;RGATUAL:=LINATUAL+1

         STA  LINATUAL                  ;LINATUAL:=LINATUAL+1
        MOV  B,A
        LDA  SJCT
        CMP  E                          ;SE SJCT = LINATUAL
        JZ   IS29
        JP   IS291
        MOV  A,B
        STA  SJCT
        JMP  IS29                       ;ENTAO ATUALIZA CARCT
IS291:  LDA  AIB1
        MOV  B,A
        MVI  A,NOCARLIN
        SUB  B
        MOV  B,A
        LDA  AIB3
        ADD  B
        CPI  NOCARLIN
        JM   ISAB
        SBI  NOCARLIN
ISAB:   STA  CARCT
        MOV  B,A
        MVI  A,NOCARLIN
        SUB  B
        STA  AIB1
        LXI  H,NOCARLIN
        CALL CMPW
        JM   ISA81
        MOV  A,B
        MOV  B,D
        MOV  C,E
        MOV  E,A
        MVI  D,0
        CALL DSUB
        LXI  H,RGTMVET
        LDA  LINATUAL
        DCR  A
        MOV  E,A
```

```
        MVI  D,0
        DAD  D
        DAD  D
        MOV  M,C
        INX  H
        MOV  M,B

ISA81:  POP  H
        POP  B

        LHLD AIW1

         JMP  VEIN
IS29:   LXI  H,RGTMVET
        LDA  LINATUAL
        DCR  A
        MOV  C,A
        MVI  B,0
        DAD  B
        DAD  B
        MOV  M,E
        INX  H
        MOV  M,D
        POP  B
        POP  B
        POP  H
        JMP  INSERT
IS30:   LDA  SJCT
        MOV  B,A
        LDA  FIMREG
        INR  A
        INR  A
        CMP  B
        JP   ISA71

        DCR  A
        MOV  B,A
        CALL PSJ
        CPI  0
        JZ   FIM6
        JMP  ISA72
ISA71:  POP  B
        PUSH B
        MOV  A,B
        XRA  C
        JZ   ISA74
        LDA  SJCT
        MVI  B,SJTM
        CMP  B
        JM   ISA73
        CALL POUTBF
        CPI 0
         JNZ  ISA73
         POP  B
         POP  B
         POP  B
```

```
                                ;ATUALIZA RGTMVET(LINATUAL)

                              ;SENAO ATUALIZ RGTMVET(LINATUA
                                    ;VA P/ INSERT

                              ;(**PRECISA NOVA LINHA ***)
                              ;SE NECESSARIO
                                 ;ENTAO PSJ
```

```
        POP   B
        MVI   A,APITO
        OUT   TECPORT
        JMP   INSERT
ISA73:  LDA   SJCT
        INR   A
        STA   SJCT
ISA72:  LXI   H,LNCONTVET
        LDA   FIMREG
        INR   A
        MOV   C,A
        MVI   B,0
        DAD   B
        MVI   M,1
ISA74:   LXI   H,LNTMVET                    ;SE TEM CHAR DEPOIS CURSOR
         LDA   FIMREG                       ;ENTAO LNCONTVET(FIMREG+1):=1
        MOV   C,A

        DAD   B
        MVI   M,NOCARLIN                    ;LNTMVET(FIMREG):=NOCARLIN
         INX   H
         POP   B
         PUSH B
         PUSH H
        LDA   AI82
        CPI   0
        JZ    RS3
        DCX   B
RS3:     LXI   H,NOCARLIN
IE97:    MOV   D,B
         MOV   E,C
         CALL CMPW                          ;ATUALIZA LNTMVET(LINATUAL)
        JM    ISA7
        XCHG
        CALL DSUB
        XCHG
        JMP   IE97
ISA7:   POP   H
        MOV   M,C
        LXI   H,LNTMVET
        LDA   LINATUAL
        DCR   A
        MOV   C,A
        DAD   B
        LDA   AI83
        MOV   B,A
        MVI   A,NOCARLIN
        SUB   B
        MOV   M,A
        LDA   FIMREG
        INR   A
        STA   FIMREG                        ;FIMREG:=FIMREG+1
        POP   B
        POP   H
        PUSH H
        PUSH B
```

```
        SHLD AIWPEG
        LDA  FIMREG
        STA  AIB6
        LXI  H,MAPLNVET
        MOV  E,A
        MVI  D,0
        DAD  D
        DAD  D
        MOV  E,M
        INX  H
        MOV  D,M
        PUSH D                          ;AIWPEG:=ENDER.ULTIMA POSICAO DO REG.
        MVI  L,NOCARLIN                 ;AIB6:=FIMREG
        MVI  A,' '
IF1:    STAX D
        INX  D
        DCR  L
        JNZ  IF1
        POP  D
        PUSH D
        LDA  AIB1
        MOV  E,A
        MVI  A,NOCARLIN
        SUB  E
        MOV  E,A
        LDA  AIB2
        CPI  0
        JZ   IS115                      ;EM E TEM CARCT JA COM RS
        INR  E
IS115:  CPI  0
        JZ   IS110
        MOV  A,B
        CPI  0
        JNZ  IS112                      ;SE ("CARCT"=0) OU (B,C < "CARCT")
        MOV  A,E
        CMP  C
        JM   IS112                         ;ENTAO L:=B,C
IS110:  MOV  L,C
        JMP  IS114
IS112:  LDA  AIB1
        MOV  L,A
        LDA  AIB3
        SUB  L                             ;SENAO L:=AIB3-AIB1  (+1 SE TEM RS)
        MOV  L,A
        LDA  AIB2
        CPI  0
        JZ   IS114
        INR  L
IS114:  MOV  A,E
        SUB  L                          ;AIB4:=CARCT-H (+1 SE TEM RS)
        STA  AIB4
        MOV  E,A
        MVI  A,NOCARLIN
        SUB  E
        STA  AIB5                    ;AIB5:=NOCARLIN-AIB4
        POP  D
```

```
        MVI  H,0
        PUSH H
        DAD  D
        DCX  H
        SHLD AIW1
        SHLD AIWPO
        POP  H
IE453!  MOV  A,B
        XRA  C
        JZ   ISA5
        MVI  H,0
        XCHG
        CALL DSUB
        XCHG
        CALL TRNCAR
        MOV  A,B
        XRA  C
        JZ   ISA5
        LDA  AIB6
        DCR  A
        STA  AIB6
        LXI  H,MAPLNVET
        MOV  E,A
        MVI  D,0
        DAD  D
        DAD  D
        MOV  E,M
        INX  H
        MOV  D,M
        LXI  H,NOCARLI-1
        DAD  D
        SHLD AIWPO
        LDA AIB4
        MOV  L,A
        MVI  H,0
        XCHG
        CALL DSUB
        XCHG
        CALL TRNCAR
        LXI  H,MAPLNVET
        LDA  AIB6
        DCR  A
        MOV E,A
        MVI  D,0
        DAD  D
        DAD  D
        MOV  E,M
        INX  H
        MOV  D,M
        LXI  H,NOCARLIN-1
        DAD  D
        SHLD AIWPEG
        LDA  AIB5
        MOV  L,A
        MOV  A,B
        CPI  0
```

```
;AIW1:=NOVO ENDER DE FIM REGISTRO
 ;AIWPO:=MAPLNVET(FIMREG)+H-1

 ;ENQUANTO B,C# 0 FACA



        ;B,C:=B,C-L

        ;TRNCAR

        ;SE  B,C # 0
           ;ENTAO AIB6:=AIB6-1

                    ;CALCULA NOVO AIWPO

                    ;L:=AIB4

          ;B,C:=B,C-L

          ;TRNCAR

                    ;CALCULA NOVO AIWPEG

                    ;L:=AIB5
```

```
        JNZ   IE@53
        MOV   A,C
        CMP   L
        JP    IE@53
        MOV   L,C
        JMP   IE@53

IS80:   PUSH B
        PUSH H
        CPI   8S
        JZ    RSCHAR
        POP   H
        POP   B
        JMP   VEIN
IS1:    PUSH B
        PUSH H
        MOV   A,D
        CPI   'L'
        JZ    IS3
        CPI   'R'
        JZ    IS4
        CPI   'U'
        JZ    IS5
        CPI   'D'
        JZ    IS6
        CPI   'T'
        JZ    IS7
        CPI   'B'
        JZ    IS8
        MVI   B,6
        JMP   ERII
IS8:    LDA   POSL8
        DCR   A
        MOV   C,A
        LDA   FIMREG
        CMP   C
        MVI   B,5
        JM    ERII
        CALL ATUTM
        CALL BOTTOM

        POP   H
        POP   H
        POP   B
        JMP   INSERT
IS7:     LDA  RGATUAL
         MOV  C,A
         LDA  POSL1
        DCR   A
        CMP   C
        MVI   B,5
        JM    ERII
        CALL ATUTM
        CALL TOP
```

```
                                                    ;SE  L)B,C
                                                     ;   ENTAO L:=B,C

                                                ;VA P/ ISA5

;SENAO INICIO (** FUNCSELECT APERTADA **)
      ;SE NAO FOI BATIDO /L
      }
          ;ENTAO INICIO
                ;SE NAO FOI BATIDO /R
                    ;ENTAO INICIO
                          ;SE NAO FOI BATIDO /U
                              ;ENTAO INICIO
                                    ;SE NAO FOI /D
                                        ;ENTAO INICIO
                                              ;SE NAO FOI /T
                                                  ;ENTAO INICIO
                                                        ;SE NAO /B
                                                            ;ENT ERRO(6)
                                                            }
                                                            ;
                                                            ;SEN INICIO
                                                                ;SE FIMREG<POSL8
                                                                  ;ENT ERRO(5)
                                                                  ;
                                                                  ;
                                                                  ;
                                                                  }
                                                                  ;SEN INICIO

                                                                          ;BOTTOM
                                                                ;FIM

                                                                ;

                                                ;SEN INICIO
                                                    ;ATUTM
                                                    ;TOP
```

```
        POP   H
        POP   H
        POP   B
        JMP   INSERT
IS6:    LDA   LINATUAL
        DCR   A
        MOV   C,A
        LDA   FIMREG
        CMP   C
        MVI   B,5
        JP    ERII
        CALL  ATUTM
        CALL  DOWNCURSOR
       POP   H
        POP   H
        POP   B
        JMP   INSERT
IS5:    LDA   RGATUAL
        MOV   C,A
        LDA   LINATUAL
        DCR   A
        CMP   C
        MVI   B,5
        JZ    ERII
        CALL  ATUTM
        CALL  UPCURSOR
       POP   H
        POP   H
        POP   B
        JMP   INSERT
IS4:    MOV   A,B
        XRA   C
        MVI   B,5
        JZ    ERII
        LDA   CRSAUX
        CPI   RUBOUT
        JZ    ERII
     CALL RIGTCURSOR
        POP   H
        POP   B
        DCX   B
        JMP   VEIN
IS3:    LHLD  CRSEND
        XCHG
        LDA   RGATUAL
        MOV   C,A
        MVI   B,0

        LXI   H,MAPLNVET
        DAD   B
        DAD   B
       MOV   A,M
       INX   H
       MOV   H,M
       MOV   L,A
        CALL CMPW
```

```
                                    ;FIM
                                ;FIM
                            ;SENAO INICIO
                                ;SE FIMREG=LINATUAL
                                    ;ENTAO ERRO(5)
                                    ;SENAO INICIO
                                    )
                                    )
                                    ;

                                    ;ATUTM
                                    ;DOWNCURSOR

                                    ;FIM


                        ;SENAO INICIO

                                ;SE LINATUAL = RGATUAL


                                ;ENTAO ERRO(5)

                                ;SENAO INICIO
                                    ;ATUTM

                                    ;UPCURSOR

                                    ;FIM
                    ;SENAO INICIO (** RIGHT **)
                        ;SE CRSEND = ULT.POS.REG.
                            ;ENTAO ERRO(5)



                        ;RIGTCURSOR
                        ;B:=NO.CARAC.DEPOIS DO CURSOR
                        ;H:=POS.FIM REG.
                        ;B:=B-1
                        ;VA P/ VEIN
                        ;FIM


;SENAO INICIO (** LEFT **)
    ;SE CRSEND = MAPLNVET(RGATUAL)
```

```
        MVI   B,5                       ;VENTRO CARACTER
        JZ    ERII
        POP   H                         ;LEFTCURSOR
        POP   B                         ;B:=NO.CARAC. DEPOIS DO CURSOR
        MOV   A,B                       ;H:=POS.ULTIMO CARAC DO REG.
        XRA   C
        JNZ   IS033
        MVI   A,RUBOUT
        STA   CRSAUX
        INX   B
        MVI   A,1
        STA   AIB2
        INX   H
IS033:  PUSH  B                         ;B:=B+1
        PUSH  H
        CALL  LEFTCURSOR
        POP   H
        POP   B
        INX   B
        JMP   VEIN                      ;VA P/ VEIN
IS10:   PUSH  B           ;(*** E' DELIMITADOR ***)
        LDA   CARCT
        CPI   0           ;
        JZ    IS100       ;SE CARCT>0
        PUSH  H           ;   ENTAO ATUTM
        CALL  ATUTM       ;
        POP   H           ;
IS100:  XCHG              ;SE RGTMVET<RGATUAL> H 0
        LXI   H,RGTMVET       ;ENTAO SE NAO TINHA COISAS DEPOIS DO CRSOR
        LDA   RGATUAL         ;
        MOV   C,A             ;
        MVI   B,0             ;
        DAD   B               ;
        DAD   B               ;
        MOV   C,M             ;
        INX   H               ;
        MOV   B,M             ;
        MOV   A,B             ;
        XRA   C               ;
        POP   B               ;
        JZ    IS101           ;
        MOV   A,B             ;
        XRA   C               ;
        JNZ   IS11            ;
        MVI   A,RUBOUT        ;
        DI                    ;
        DI                    ;
        STA   CRSAUX          ;ENTAO POE RS
        EI                    ;
        JMP   IS11            ;
IS101:  LDA   CRSAUX          ;SENAO SE NAO TEM RS
        CPI   RUBOUT          ;ENTAO SJCT:=SJCT-1
        JZ    IS11            ;
        LDA   SJCT            ;
        DCR   A               ;
        STA   SJCT            ;
```

```
IS11:   POP   H
        POP   H
        POP   H
        RET
FIM6:   POP   H
        POP   H
        POP   H
        POP   H
        MVI   A,APITO
        OUT   TECPORT
        JMP   INSERT
FIM5:   POP   H
        POP   H
        POP   H
        POP   H
        MVI   A,APITO
        OUT   TECPORT
        RET
FIM4:   POP   H
        POP   H
        POP   H
        POP   B
        JMP   VEIN
ERI1:   CALL  MENSAGEM
        POP   H
        POP   B
        JMP   VEIN
FIM2:   POP   H
        POP   D
        POP   B
        MVI   A,APITO
        OUT   TECPORT
        JMP   VEIN
FIM3:   POP   H
        POP   H
        POP   H
        MVI   A,APITO
        OUT   TECPORT
        JMP   INSERT
```