UM SISTEMA OTIMIZADO DE ORDENAÇÃO EXTERNA COM UM DISCO. IMPLEMENTAÇÃO NO PDP11/10

Sálvio Calicchio Sobrinho

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.)

Aprovada por:

Prof. Célio Guimarães
(Presidente)

Prof. Jayme Luiz Szwarcfiter

Prof. Pierre Jean Lavelle

Prof. Nelson Maculan Filho

RIO DE JANEIRO, RJ - BRASIL
AGOSTO DE 1976

Ã Cristina e Meus Pais

AGRADECIMENTOS

Ao professor Célio Guimarães pela orientação deste trabalho; ao colega Pedro Nogueira Cruz pela sua colaboração e apoio; aos funcionários do NCE e em especial a equipe de trabalho do PDP11/10 que nos incentivou aos estudos; ao professor Nelson Maculan e funcionários do Programa de Sistemas; aos colegas da COPPE.

Agradecemos também ao CEPEL (Centro de Pesquisa de Energia Elétrica) pela facilidade que nos deram para o término da tese.

RESUMO

Este trabalho apresenta a teoria para construção de um sistema de ordenação externa em um só disco magnético, visando minimizar o tempo total de ordenação. Este sistema leva em consideração o tamanho da memória interna disponível, os tempos de movimentação do braço, de transferência de dados para/ do disco e o tamanho do arquivo a ser ordenado.

Esta teoria para discos setorizados mostrou que apenas a memória interna disponível e o produto do tempo de movimentação do braço pela razão de transferência são fatores importantes na minimização do tempo total de ordenação.

O sistema foi implementado e testado no computador PDP11/10. Alguns algorítmos foram implementados no B6700.

ABSTRACT

A theory for building a extern sort system in only one magnetic disk is presented. This theory has the objetive of minimizing the total sort time. This system into account the available internal memory size, the seek time, the data transfer time and the file size to be sort.

This theory for sectors structured magnectics disks shown that only the available internal memory size and the product of the seek time by the transfer ratio are the important factors in the minimization of the total sort time.

This system was implemented and tested on PDP11/10 computer and algorithms were implemented on B6700.

INDICE

I - INTRODUÇÃO 1

II - TEORIA 3

2.1 Definições 3

2.2 Definição do problema de ordenação externa em um disco 5

2.3 Fase de criação de corridas 7

2.4 Obtenção dos parâmetros de intercalação 11

2.5 Tempo total de ordenação 13

2.6 Sistema de ordenação externa em um disco 17

III - IMPLEMENTAÇÃO 25

3.1 Características do computador PDP11/10 25

3.2 Características do sistema implantado 28

3.3 Minimização do tempo de processamento 42

3.4 Estrutura de "Overlay" 42

3.5 Manual de utilização 44

IV - RESULTADOS EXPERIMENTAIS 51

CONCLUSÕES

APÊNDICES

A - Tabelas 63

B - Fluxograma 79

C - Listagens dos programas 83

I - INTRODUÇÃO

A eficiente utilização do computador para ordenação de grandes quantidades de dados é um passo importante para se obter uma minimização no tempo de processamento de uma aplicação científica ou de um problema comercial.

Os vários métodos de ordenação são classificados em métodos internos e métodos externos. O método interno é aquele que pode ser aplicado para quantidades de dados que possam estar inteiramente contidos em memória interna. O método externo trata com quantidades de dados que não possam estar inteiramente em memória interna, utilizando então memória externa (fitas, disco, tambores, etc.) durante a fase de ordenação.

A escolha do método de ordenação deve levar em conta vários parâmetros que possam afetar a performance da ordenação. Existem situações em que o mais primitivo método de ordenação poderá ser adequado, porque os tempos da ordenação poderão ser sobrepostos ou ser tão pequeno que não se torna importante. Já em outros casos, o tipo do dado, as saídas requeridas, a natureza do dispositivo, a memória interna disponível, etc. são considerações envolvidas na escolha do método.

A crescente redução dos custos dos minicomputadores e dos dispositivos de memória externa (disco de cabeça móvel) fez com que pequenos sistemas baseados em um minicomputador com uma unidade de disco se tornassem cada vez mais comuns.

O presente trabalho apresenta a teoria para construção de um sistema de ordenação externa em um só disco visando minimizar o tempo total da ordenação de um arquivo sequencial levando em consi-

deração o tamanho da memória interna disponível, os tempos de movimentação de braço, de transferência de dados para/do disco e o tamanho do arquivo a ser ordenado.

A aplicação desta teoria a discos setorizados mostrou que apenas a memória interna disponível e o produto do tempo de movimentação do braço pela razão de transferência são fatores importantes na minimização do tempo total de ordenação. Esta teoria pode ser facilmente estendida a um sistema com dois discos.

O sistema foi implementado e testado no minicomputador PDP11/10 do NCE da UFRJ.

II - TEORIA

2.1 Definições

2.1.1 Ordenação e Intercalação

Dado um conjunto (arquivo) A com n registros:

$$A = \{r_1, r_2, \ldots, r_n\}$$

Seja K_i um campo do registro r_i, que chamaremos de chave. K_i pertence ao conjunto K das chaves possíveis. Suporemos que K é linearmente ordenado através das relações $\leqslant$ e $\geqslant$.

- Ordenação é o processo de encontrar uma permutação $p(1), p(2), \ldots, p(n)$ dos registros, a qual coloque a chave que define o campo para a ordenação em ordem ascendente:

$$K_{p(1)} \leqslant K_{p(2)} \leqslant \ldots \leqslant K_{p(n)}$$

ou em ordem descendente:

$$K_{p(1)} \geqslant K_{p(2)} \geqslant \ldots \geqslant K_{p(n)}$$

Consideremos t subconjuntos ordenados de A segundo uma determinada ordem:

$$A_1 = \{r_{p_1(1)}, r_{p_1(2)}, \ldots, r_{p_1(n_1)}\}$$

$$A_2 = \{r_{p_2(1)}, r_{p_2(2)}, \ldots, r_{p_2(n_2)}\}$$

$$\vdots$$

$$A_t = \{r_{pt(1)}, r_{pt(2)}, \ldots, r_{pt(nt)}\}$$

Onde:

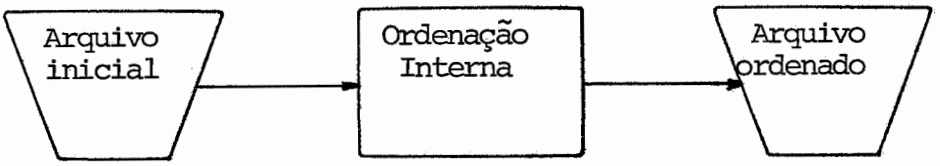
$$n_1 + n_2 + \ldots + n_t = n \quad \text{e} \quad p_{i(1)}, p_{i(2)}, \ldots, p_{i(n_i)}$$

uma permutação de n_i registros.

- Intercalação é o processo de intercalar os t subconjuntos de tal forma que se obtenha um só conjunto ordenado. Os subconjuntos A_i, $1 \leqslant i \leqslant t$ chamaremos de corridas. Nos métodos de intercalação nem sempre é possível ou eficiente intercalar todos os t subconjuntos de uma só vez. Desta forma, os subconjuntos serão intercalados em grupos de p, onde p chamaremos de ordem de intercalação, necessitando também de novas passagens sobre o arquivo que chamaremos de passos.

2.1.2 Ordenação Interna e Externa

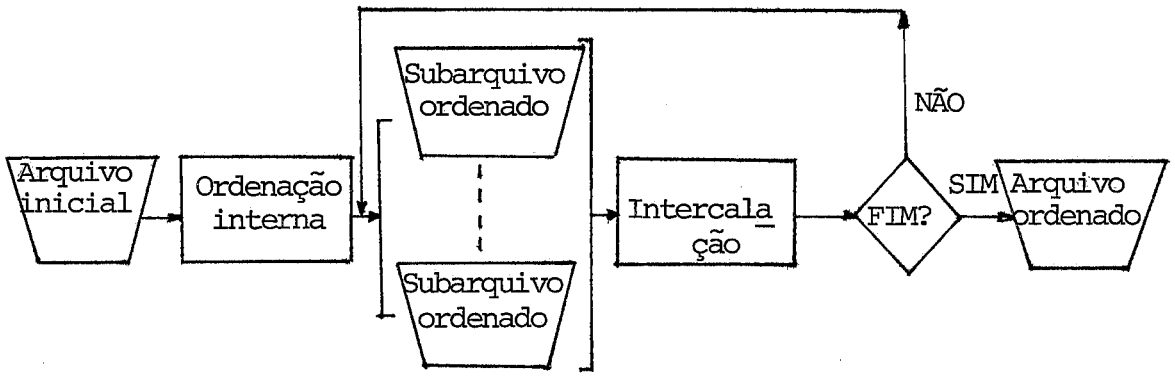
Se a memória interna disponível para os registros na ordenação for suficiente para alocar os n registros do arquivo A, o processo de ordenação será todo feito em memória interna, neste caso teremos uma ordenação interna (Fig. 1).



Ordenação Interna
Figura 1

Caso a memória interna disponível para registros não for suficiente para alocar os n registros,a ordenação constará de duas etapas , sendo que a primeira criará subarquivos ordenados (corridas),que serão armazenados em dispositivos de memória externa e em uma etapa

posterior fará a intercalação dos subarquivos (corridas), obtendo um só arquivo ordenado. Este procedimento chamaremos de ordenação externa (Fig. 2).



Ordenação Externa
Figura 2

2.2 Definição do Problema de Ordenação Externa com um Disco

Ao contrário da ordenação interna, onde a necessidade é escrever ótimas rotinas que minimizem o tempo de CPU, para a ordenação externa tem-se a necessidade de escrever ótimas rotinas que minimizem não só o tempo de CPU, mas também, os tempos de entrada/saída do dispositivo utilizado como memória auxiliar. Como estamos considerando o disco como tal dispositivo, os seguintes tempos serão considerados:

- tempo de movimento de braço ("seek"): tempo para a escolha de um cilindro;
- tempo de "switch": tempo para a escolha de outra trilha dentro do cilindro;
- tempo de espera ("latency time"): tempo de espera para a obten-

ção do bloco dentro da trilha e que corresponde ao tempo decorrido entre posicionamento de braço na trilha escolhida e a chegada do bloco sob a cabeça de leitura/gravação;

- tempo de transferência: tempo de rotação enquanto o dado passa sobre a cabeça de leitura/gravação.

O tempo total de ordenação será quase todo dominado pelos tempos de entrada e saída, dentre estes o tempo que deve ser considerado seriamente é o tempo de movimento do braço, isto porque, em se tratando de um só disco, estando o braço posicionado em uma corrida, para a leitura ou gravação de dados de outra corrida haverá um deslocamento de algumas trilhas. Outro fato importante é de não se poder sobrepor operações de entrada e saída.

Um sistema com múltiplas unidades, com "drives" independentes, poderá utilizar um processo de distribuição de corridas em unidadees distintas, de modo que o tempo de movimento de braço de uma corrida para outra seja completamente eliminado.

Das várias fases de um sistema de ordenação externa as principais são:

- fase de criação das corridas;
- fase de intercalação.

A meta deste sistema será de balancear o tempo da fase de criação das corridas e o tempo de intercalação de tal forma que minimizem o tempo total de ordenação. Esta meta não é alcançada apenas considerando um algoritmo que otimize a fase de criação das corridas ou a fase de intercalação, isto porque, existe um grande relacionamento entre elas.

A escolha do algoritmo para a fase de criação das corridas deve estar baseada nos seguintes itens:

- número de corridas produzidas;
- tamanho das corridas;

já que estes serão responsáveis para a definição do número de pas - sos necessários para a intercalação.

O propósito da fase de intercalação é a obtenção do número de passos e a ordem de intercalação em cada passo, que minimize o tempo total de intercalação. Três parâmetros do sistema são fundamentais para esta minimização: a memória disponível para armaze - nar registros do arquivo, o tempo médio de movimento de braço e a razão nominal de transferência do disco. Surpreendentemente, o tamanho do arquivo tem importância apenas marginal, como será visto a seguir, sendo utilizado para obter uma estimativa do tempo médio de movimento de braço através de uma aproximação linear. A técnica utilizada para intercalar os arquivos é derivada da intercalação Balanceada para fitas magnéticas. Técnicas como Polifasica e Cascata que são mais eficientes que a Balanceada para fitas magnéticas, não seriam uma boa maneira de intercalar as corridas no disco. Necessitariam de áreas separadas simulando as fitas e isto acarretaria em maior movimento de braço e as ordens de intercalação ficariam res - tritas a quantidade destas áreas.

Com estes aspectos será possível então, a comparação e obtenção de métodos que acoplados atinjam os objetivos desejados.

2.3 Fase de Criação das Corridas

A idéia básica nesta fase é a utilização de um algoritmo que minimize o tempo de CPU e o tempo de entrada e saída do sistema de ordenação. Como o tempo de entrada e saída é muito superior ao tempo de CPU, será melhor então a escolha de um algoritmo que gaste um pouco mais com tempo de CPU, mas que minimize tempo de entrada e

saída.

Uma diminuição no número de corridas formado nesta fase , provavelmente acarretará uma redução no número de passos de intercalação. Um passo a menos na fase de intercalação significa a diminuição de tempo de trasferência equivalente à leitura e gravação de todo o arquivo, e de inúmeros movimentos de braço.

2.3.1 Características do Método

Dos vários métodos existentes de ordenação interna, como por exemplo "Replacement Selection", "Binary Insertion", "Sifting", "Internal Merge" (1), o que mais se adapta às características do problema é o "Replacement Selection".

Várias propriedades tornam este método atrativo:

- o método de seleção de um registro é uniforme, desta maneira o tempo gasto pode ser razoavelmente fixado. Permitindo, assim, uma análise do tamanho das áreas de entrada e saída para a fase de criação das corridas.

- o número de comparações para a seleção de um registro é no máximo $1 + \lceil \log_2 P \rceil$ onde P é o número de registros com os respectivos apontadores que podem ser armazenados em memória interna.

- Se o arquivo estiver quase ordenado tem-se a possibilidade da total ordenação somente nesta fase.

- Cada registro será movido somente duas vezes, todas as demais movimentações são de chaves e endereços.

- O tamanho médio das corridas é da ordem de 2P, para dados randômicos (1,2).

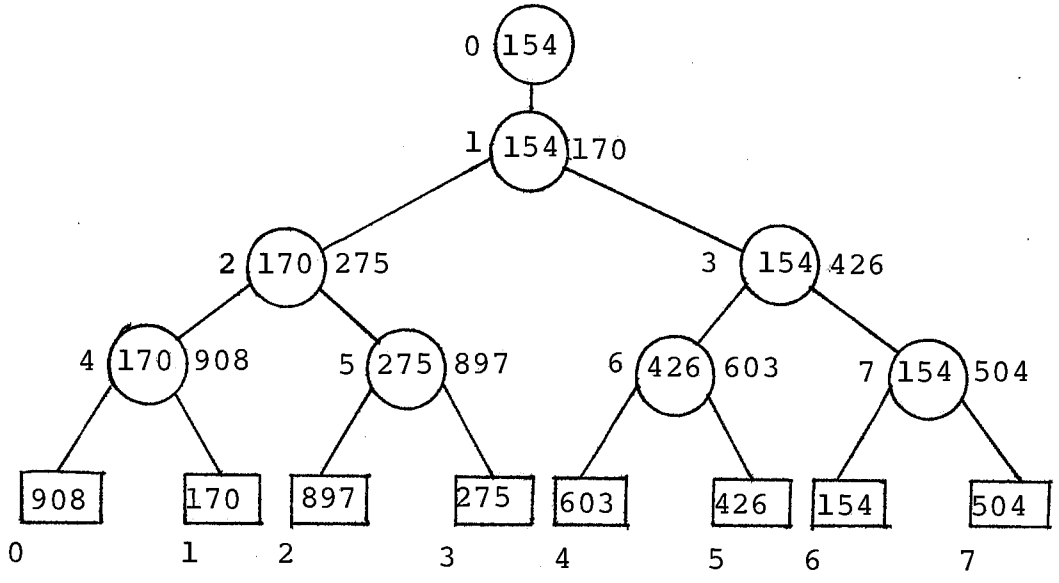
- Como poderá ser observado em 2.3.2, a disposição dos re-

gistros no arquivo inicial é que determinará o tamanho das corridas e consequentemente o número destas. Este fato é, às vezes, apontado como desvantagem do método, já que um conhecimento "a priori" do número de corridas facilitará a organização da memória disponível para esta fase. Vários processos de intercalação tem a necessidade de que o tamanho das corridas sejam fixos.

Alguns dos outros métodos podem apresentar um número menor de comparações de registros, mas em termos de número de corridas formadas o "Replacement Selection" possui as vantagens descritas anteriormente e que um algoritmo que gera um tamanho de corrida constante não apresenta.

2.3.2 Técnica de Formação das Corridas

O processo de formação das corridas pelo algoritmo de "Replacement Selection" utiliza a estrutura de árvore binária (Fig.3), perfeitamente balanceada.



Árvore Binária

Figura 3

Um nó na posição j tem como antecedente $\lfloor \frac{j}{2} \rfloor$ e como descendentes 2j e 2j + 1. Para os nós cujos descendentes são as folhas, temos: um nó j tem as folhas 2j - P e 2j - P + 1 como descendentes. Neste esquema, as folhas representam os registros e os nós internos apontadores. Para a obtenção do registro de menor chave (se a ordenação for ascendente) o procedimento é o seguinte: pares de chaves de registros adjacentes são comparadas. O registro de menor chave de um par terá sua chave comparada com a de outro registro de menor chave de um par adjacente (outro nível da árvore). Este processo segue a árvore de baixo para cima até se obter o registro de menor chave. Este registro é colocado na área de saída e um novo registro da área de entrada toma o seu lugar. O ramo da árovre onde foi colocado o novo registro terá todos os ponteiros atualizados. Uma corrida termina quando todos os registros em memória interna tenham chaves que sejam menores que a do último registro transferido para a área de saída. Nas comparações efetuadas somente ponteiros são atualizados (nós internos), nenhuma movimentação de registro é necessária. Uma modificação neste procedimento foi apresentada por W.D. Frazer e C. K. Wong (1), para aumentar o tamanho das corridas: se a chave do novo registro que deverá ocupar uma posição na árvore for menor que a chave do último registro que saiu,este novo registro não deve ocupar a posição na árvore e sim em uma área externa que comporta P' registros. Com isto o próximo registro da fila de entrada terá a chance de pertencer à corrida que está se formando. Este processo continua até que a área de tamanho P' esteja cheia, os seus registros serão utilizados como entrada para a formação da próxima corrida. Com esta modificação o tamanho das corridas, com dados randômicos, é aproximadamente eP, onde e $\simeq$ 2.718 e se P = P'.

2.4 Obtenção dos Parâmetros para Intercalação

Tratando-se de disco como memória externa e não fita magnética, a intercalação torna-se um tanto diferente porque todas as corridas são igualmente acessíveis, permitindo intercalação de qualquer ordem, removendo-se assim o limite que se faz necessário com a utilização de fitas magnéticas como memória externa.

Levando-se em consideração que:

- o tempo de transferência é reduzido pelo aumento da ordem de intercalação;

- o tempo de acesso ao disco (tempo de rotação + tempo de movimento de braço) é acrescido pelo aumento da ordem de intercalação, pois o tamanho das áreas para entrada e saída são diminuidos , o problema então é escolher uma ordem de intercalação ótima para que estas duas características sejam balanceadas, e contrariamente ao que se esperava a ordem ótima, em geral, não será a mais alta ordem possível. Seria desejável determinar uma simples fórmula ou um simples algoritmo que calcule o número de passos e a ordem de intercalação em cada passo. Vários métodos apresentam uma solução para o problema, dentre eles:

- Algoritmo de árvore ótima (1)
- Algoritmo de N. A. Black (3)
- Modelos Heurísticos (4)

O algoritmo de N. A. Black trata o problema de uma forma simples e obtem uma boa aproximação da solução ótima, inclusive uma extensão deste métodos foi utilizada no modelo heurístico citado acima. Neste modelo a intercalação é feita em P passos, utilizando somente duas ordens de intercalação,d e d-1. No primeiro passo q intercalações serão feitas com a ordem d e r intercalações com a ordem d-1.

Para $P-\overline{S}$ passos a intercalação é feita com a ordem d-1 e para os S-1 passos restantes ordem d. A diferença básica para o método de N.A. Black está no fato de utilizar no primeiro passo as duas ordens de intercalação, com isto ganhando em movimento de braço neste passo. Por esta característica supõe-se que o tamanho das corridas iniciais sejam aproximadamente iguais. Isto porque se fossem de tamanhos bem diferentes o tempo de movimento do braço seria variável no primeiro passo, trazendo assim certas dificuldades na obtenção dos parâmetros d, q, p, r, e s. Este modelo apresentou uma diferença de 1% do ótimo. O algoritmo apresentado por N.A. Black tem as seguintes características:

- cada passo será feito segundo uma certa ordem de intercalação;
- todos os registros participarão da intercalação em cada passo;
- no máximo, duas ordens de intercalação P e P-1 serão utilizadas. Uma justificação sobre a otimilidade desta escolha se encontra no artigo de Black (3).

Estas características dão uma forma particular para o caso geral, onde somente parte das corridas poderiam ser intercaladas em um determinado passo. Desta maneira, certos registros participam infrequentemente da intercalação. A simplicidade do método oferece certas vantagens:

- facilidade de programação;
- facilidade de controle dos arquivos;
- possibilidade de recomeçar a intercalação em um certo passo, se alguma interrupção ocorrer.

Considerando as características do método, o tempo total de interca

lação será dado pela soma dos tempos de intercalação em cada passo. Torna-se fácil obter uma fórmula que expresse este tempo, já que todos os registros participam em cada passo da intercalação.

2.5 Tempo Total de Ordenação Externa

Considerando:

L - tamanho do arquivo de registros.

l - tamanho do registro em palavras.

S - número de corridas.

M - memória interna disponível, em palavras.

V - número de passos.

P_j - ordem de intercalação no passo j

b --tamanho da área de entrada e saída em palavras (fase ou criação das corridas

b_{ej}-tamanho da área de entrada em palavras (fase de intercalação)

onde $b_{ej} = \left\lfloor \frac{M}{P_j + 1} \right\rfloor$

b_{sj}-tamanho da área de saída em palavras (fase de intercalação),

onde $b_{sj} = M - P_j \times b_{ej}$

B - inverso da razão de transferencia em ms/palavra

A - tempo médio de movimento de braço mais o tempo de meia revolução em ms/área de entrada e saída.

Uma observação a ser feita aqui é que o tempo de movimento de braço depende do tamanho do arquivo, e como veremos mais adiante na alocação dos arquivos em disco, este será aproximadamente o tempo que o braço gastaria para atravessar o número de cilindros correspondentes ao tamanho do arquivo inicial.

O tempo total de ordenação será dado pela soma dos tempos CPU e de entrada e saída. O tempo predominante na ordenação to-

tal é o de entrada e saída, cuja ordem de grandeza é bastante superior à ordem de grandeza do tempo de CPU. O tempo de CPU é mais acentuado na fase de criação das corridas. Tempos relativos à alocação de arquivos e processamento dos algoritmos para obtenção de parâmetros não serão considerados, uma vez que não são passíveis de minimização. O tempo total da fase de criação de corridas será dado aproximadamente por:

$$TCT = K_1 L \log \left(\frac{M - 2.b}{l}\right) + 2.T_m + 2.(B.L.l. + \frac{A.L.l}{b}) \qquad (I)$$

onde:

- $K_1.L.\log \left(\frac{M - 2.b}{l}\right)$ representa a parcela relativa ao tempo gasto em comparações no algoritmo de "Replacement Selection", sendo K_1 uma constante de proporcionalidade que depende dos tamanhos das chaves e seus tipos.
- T_m representa o tempo de CPU gasto em movimentações de registros da área de entrada para a área de trabalho e daí à área de saída, sendo portanto duas movimentações por registro. Este tempo é proporcional ao tamanho do arquivo.
- B.L.l representa o tempo gasto em transferência de todo o arquivo do disco para a memória interna ou vice-versa.
- $\frac{A.L.l}{b}$ representa o tempo gasto para posicionamento da cabeça de leitura/gravação (movimentação do braço + tempo de espera pelo setor), para todo o arquivo.

Os tempos B.L.l e $\frac{A.L.l}{b}$ são multiplicados por dois, já que todo o arquivo é lido e gravado uma vez nesta fase.

As seguintes modificações foram feitas no algoritmo de N. A. Black para a obtenção dos parâmetros de intercalação:

- O número de corridas será aproximadamente $\frac{L.l}{2.(M - 2.b)}$ e não

$\frac{L.l}{M}$;

- considerar os tempos de leitura e gravação e não somente o tempo de leitura.

Uma aproximação do tempo total de intercalação será dada por:

$$TTI = \sum_{j=1}^{v} \{K_2.P_j + T_m + 2.B.L.l + A.L.l\ (\frac{1}{b_{ej}} + \frac{1}{b_{sj}})\} \qquad (II)$$

onde:

$b_{ej} = \left\lfloor \frac{M}{P_j + 1} \right\rfloor$ e $b_{sj} = M - P_j \cdot b_{ej}$, com os P_j satisfazendo $P_1.P_2 \ldots P_v \geq S$, sendo S o número de corridas

$K_2.P_j$ - é o tempo relativo às comparações que dependem da ordem de intercalação em um dado passo j.

T_m - é o tempo de CPU gasto em movimentação de todos os registros do arquivo que serão movimentados uma vez em cada passo.

B.L.l - é o tempo de transferência disco/memória, que corresponde a uma leitura ou a uma gravação de todo o arquivo em cada passo.

A.L.l $(\frac{1}{b_{ej}} + \frac{1}{b_{sj}})$ - corresponde ao tempo gasto para posicionamento de cabeça de leitura/gravação em relação às áreas de entrada e saída.

Considerando:

$D = \frac{T_m}{2.B.L.l}$, $\hat{K}_1 = K_1.L.\log\ (\frac{M - 2.b}{l})$ e $\hat{K}_2(j) = K_2.P_j$, o tempo total esperado de ordenação será estimado por: TO = TCT + TTI

$$E(TO) = \hat{K}_1 + (1 + 2.D).2.B.L.l + 2.\frac{A.L.l}{b} + \sum_{j=1}^{v} \{\hat{K}_2(j) +$$

$$+ (1 + D).2.B.L.l + A.L.l\ (\frac{1}{b_{ej}} + \frac{1}{b_{sj}}) \qquad (III)$$

com $P_1.P_2 \ldots P_v \geq E(S)$ onde,

$E(S) = \frac{L.l}{2.(M - 2.b)}$ é o número esperado de corridas a serem geradas.

Definindo $TO_n = \frac{TO}{2.B.L.l}$ teremos:

$$E(TO_n) = \underbrace{\frac{\hat{K}_1}{2.B.L.l} + 1 + 2.D + \frac{2.G}{b}}_{TC_n} + \underbrace{\sum_{j=1}^{V} \{\frac{\hat{K}_2(j)}{2.B.L.l} + (1 + D) + G.(\frac{1}{b_{ej}} + \frac{1}{b_{sj}})\}}_{TI_n} \quad (IV)$$

onde $G = \frac{A}{2.B}$ e:

TC_n - tempo de criação das corridas normalizado;

TI_n - tempo de intercalação normalizado.

O objetivo do sistema de ordenação é minimizar $E(TO_n)$. A minimização estará dividida em duas partes: a primeira será de encontrar um valor para b que minimize $E(TO_n)$, usando para isto o número esperado de corridas. A segunda será de encontrar as ordens de intercalação e o número de passos que minimize TI_n, desta vez levando em consideração o número exato de corridas geradas na primeira fase.

O algoritmo abaixo escolhe o número de passos e a intercalação em cada passo tal que o tempo total de intercalação seja minimizado. Considerando S o número de corridas, P a ordem de intercalação e V o número de passos, temos: ALGORITMO 1 (N. A. Black).

1 - Para cada V, $1 \leqslant V \leqslant \lceil(\log_2 S)\rceil$

1.1 Calcule o menor P t.q $P^V \geqslant S$

1.2 Calcule o maior r, $0 \leqslant r < V$ t.q

$P^{(V-r)} \cdot (P-1)^r \geqslant S$

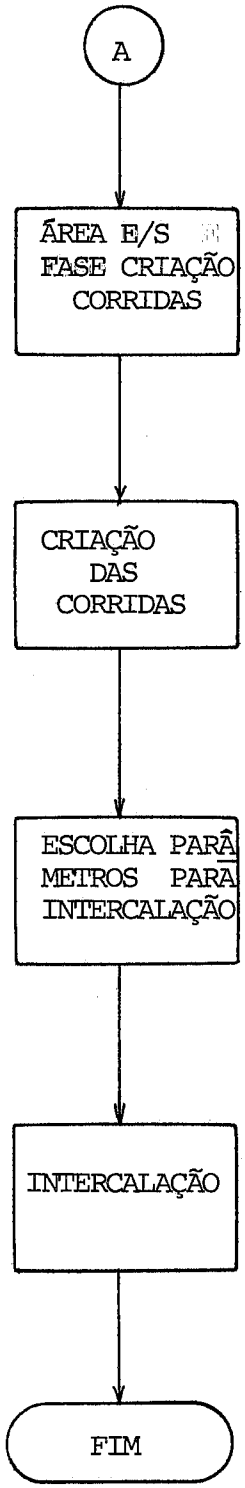
2 - Obtenha V, P e r que dê o menor tempo normalizado de intercalação (TI_n), discutido acima. Com a obtenção de V, P e r a intercalação será feita da seguinte maneira:

1 - Execute r passos com a ordem de intercalação P-1

2 - Execute V-r passos com a ordem de intercalação P

2.6 Sistema de Ordenação Externa em Um Disco

Além das fases de criação das corridas e obtenção dos parâmetros para a intercalação, que constituem as principais fases de um sistema de ordenação externa, há necessidade de outras fases que completarão o sistema. A estrutura básica do nosso sistema de ordenação externa é apresentada na Figura 4.



Estrutura Básica do Sistema de Ordenação Externa

Figura 4

2.6.1 Verificação dos Parâmetros do Usuário

Nesta fase uma crítica dos parâmetros do usuário é feita para verificação de suas validades.

2.6.2 Gerência de Memória Interna

A memória interna, sendo um dos fatores de maior importância em um sistema de ordenação externa, seu índice de aproveitamento deve ser o melhor possível. Isto somente se consegue com uma boa interface com o sistema operacional que está sendo utilizado.

2.6.3 Alocação de Arquivos

Com a escolha do método de N. A. Black para a fase de intercalação, se faz necessário a alocação em três arquivos.

- Arquivo de entrada
- Arquivo de saída
- Arquivo de trabalho

O arquivo de trabalho tem duas vezes o tamanho do arquivo de entrada, pode-se ter a opção do arquivo de saída ser o mesmo de entrada.

2.6.4 Criação das Corridas

Como ficou claro, a minimização do tempo total de ordenação externa não depende somente de escolher de um algoritmo que minimize a fase de criação das corridas e de um algoritmo que minimize a fase de intercalação. A memória interna disponível nesta fase é dividida em espaço para áreas de entrada e saída e área para armazenar registros para ordenação. Um problema importante neste ponto é a escolha do tamanho b das áreas de entrada da fase de criações de corridas. Uma alteração em b, estaremos alterando a memória disponível para registros, desta forma modificando o número de corridas que consequentemente al

terará a intercalação. Devemos estimar b de tal modo que os tempos TC e TI sejam balanceados e nos dêem o menor TO. Para isto o número esperado de corridas E(S) será dado por: $\frac{L \cdot 1}{2.(M - 2.b)}$ para arquivos randômicos. Uma característica importante aqui é o fato de que mesmo não sendo necessária a fase de intercalação, o tempo de criação das corridas será minimizado. Considerando b_{inf} o tamanho mínimo da área de entrada e saída e b_{sup} o tamanho máximo, o seguinte algoritmo nos fornecerá o valor estimado de b.

ALGORITMO 2

1 - Para cada b, $b_{inf} \leq b \leq b_{sup}$

1.1 Faça $E(S) = \frac{L.1}{2.(M - 2.b)}$

1.2 Para cada V, $1 \leq V \leq \lceil \log_2 E(S) \rceil = V_{max}$

1.2.1 Calcule o menor P t.q. $P^V \geq E(S)$

1.2.2 Calcule o maior r, $0 \leq r \leq V$
t.q. $P^{(V-r)} \cdot (P - 1)^r \geq E(S)$

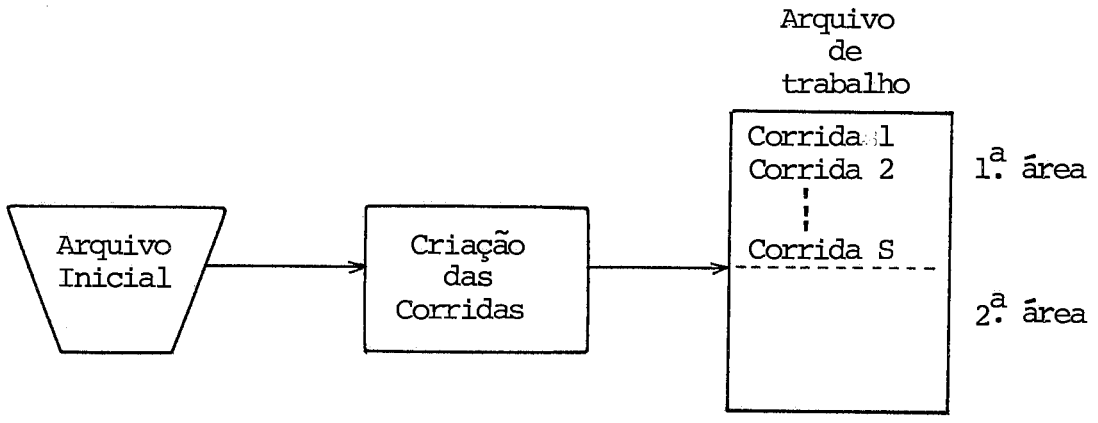
1.2.3 Calcule TO_n como função de b e V e dado por (IV).

1.3 Calcule $TO_n(b, V_{min}) = \min TO_n (b,V), 1 \leq V \leq V_{max}$

2 - Calcule $TO_n (b_{min}, V_{min}) = \min TO_n (b,V_{min}), b_{inf} \leq b \leq b_{sup}$

O valor b_{min} será utilizado como o tamanho para as áreas de entrada e saída da fase de criação das corridas.

A medida que as corridas forem sendo formadas serão dispersas na primeira parte do arquivo de trabalho sequencialmente como mostra a Fig. 5. Concorrentemente será formada uma tabela com dados sobre as corridas que serão utilizadas na fase de intercalação.



Dispersão das corridas

Figura 5

2.6.5 Obtenção dos Parâmetros da Intercalação

Com o número exato S das corridas formadas, a obtenção dos parâmetros V, P e r que definirão a fase de intercalação são escolhidos de tal forma que o tempo TI seja minimizado de acordo com o Algoritmo 1, apresentado 2.4.

2.6.6 Intercalação

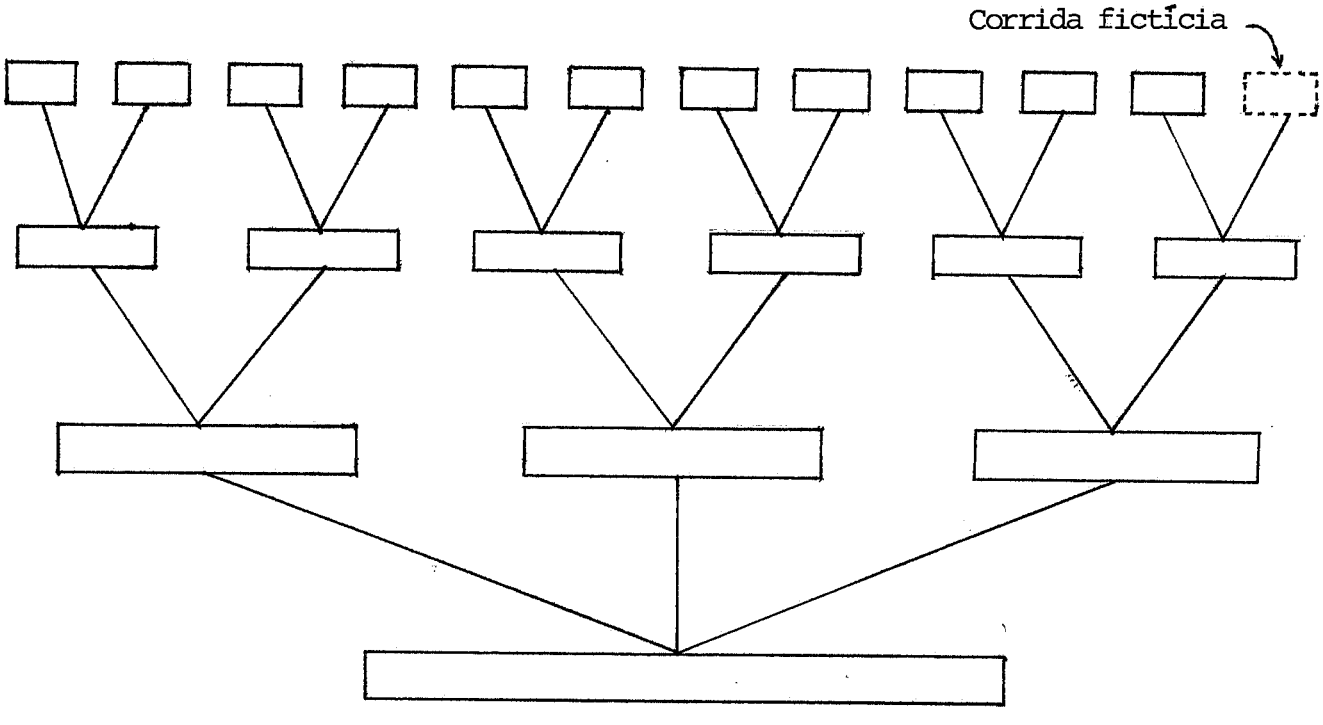
O gerenciamento das áreas de entrada e saída e do arquivo de trabalho são as características principais desta fase.

Seja S_i o número de corridas no passo V_i, P_i a ordem de intercalação para este passo e M a memória interna disponível em palavras.

A área de memória interna disponível será dividida em $P_i + 1$ sub-áreas de tamanho $\left\lfloor \frac{M}{P_i + 1} \right\rfloor$ para P_i áreas que serão consideradas como áreas de entrada e uma de tamanho $M - P_i \cdot \left\lfloor \frac{M}{P_i + 1} \right\rfloor$ que será considerada como área de saída. Um estudo foi feito por

E. Ferguson, S. J. Waters e Ewings Walker (1), baseado no fato que se fazendo o tamanho da área de saída diferente do tamanho da área de entrada, obtem-se uma minimização no tempo de movimento de braço, como por exemplo: considerando a ordem de intercalação 4, M memória disponível e L_o tamanho de cada corrida em caracteres, teremos: para uma intercalação desta ordem, se tomarmos B = M/5 como área de entrada e saída teríamos necessidade de $\frac{L_o}{B}$ movimentos de braço para cada área de entrada e $4.\frac{L_o}{B}$ para a área de saída, totalizando $\frac{8.L_o}{B} = \frac{40.L_o}{M}$ movimentos de braço para a intercalação. Ao passo que se utilizarmos as áreas de entrada de tamanho M/6 e a de saída M/3 seriam necessários $4 . \frac{6.L_o}{M} + 4. \frac{3.L_o}{M} = \frac{36.L_o}{M}$. Desta forma este método é melhor para alocar as áreas de entrada e de saída.

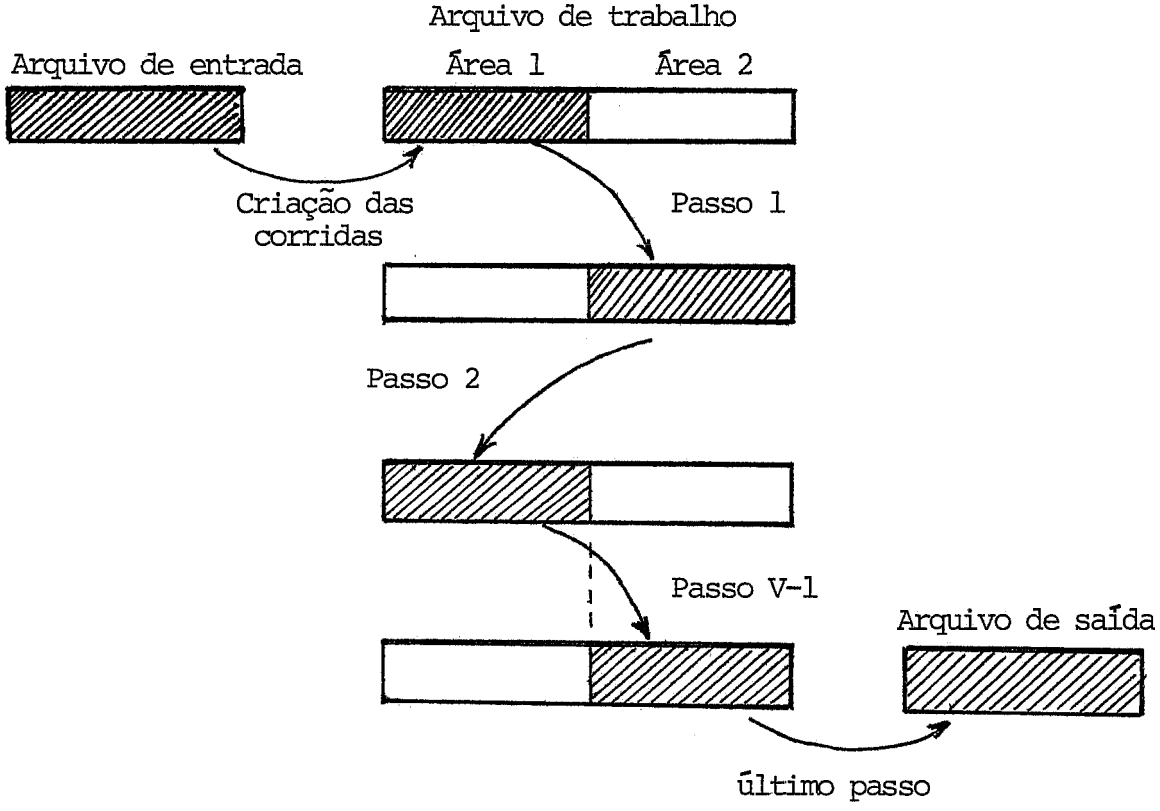
O agrupamento das corridas em cada passo é como mostra a figura 6.



Intercalação para S=11, V=3, P=3 e r=2

Figura 6

Observemos que se P_i não divide S_i a última intercalação do passo V_i será da ordem de S_i mod P_i. Outra observação é o fato de utilizar a área de entrada de uma corrida que terminou como extensão de outra área de entrada ou saída. As novas corridas que estão sendo formadas em cada passo V_i serão gravadas em uma das áreas do arquivo de trabalho, com exceção do último passo que será gravado diretamente no arquivo da saída do usuário. O procedimento é como mostra a Figura 7.



Gravação das Corridas em cada Passo

Figura 7

O algoritmo para obtenção das menores ou maiores chaves, dependendo do sentido da ordenação, pode ser um de comparação sequencial, já

que as ordens de intercalação em geral não ultrapassam oito.

III - IMPLEMENTAÇÃO

A implementação de um modelo teórico depende das características de "hardware" e "software" do computador utilizado. Nesta fase surgem alternativas para melhoria do modelo, bem como a possibilidade para testá-lo com dados reais.

3.1 Características do Computador

O computador utilizado na implementação foi o PDP 11/10 do Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro.

O sistema computacional é o DOS/BATCH monoprogramável. Ocupa uma área residente de memória interna de 4 K palavras de 16 "bits".

3.1.1 Disco

O disco utilizado é o RKØ5. Contem duas faces e suas trilhas são divididas em setores ou blocos. O endereçamento é feito a nível de setor.

Suas características são:

- Capacidade	1.228.800 palavras	
- Organização	2 faces	
	200 trilhas / face	
	12 setores / trilha	
	256 palavras/ setor	
- Tempos	transferência	3.3ms / setor
	meia revolução	20 ms
	velocidade de rotação	1500 RPM

Posicionamento na trilha:

- Movimento de uma trilha 10 ms

- Média 50 ms
- Máximo 85 ms

O sistema operacional suporta dois tipos de arquivos em disco:

- Contíguo: os setores são alocados adjacentes
- Não contíguo: os setores não são alocados adjacentes , tendo-se a necessidade de apontadores.

3.1.2 Linguagem Utilizada

A linguagem de programação utilizada foi o MACRO ASSEMBLER, a menos de alguns programas em ALGOL rodados no BURROUGHS B.6700 para geração de tabelas.

Foi necessária a construção de subrotinas de multiplicação e divisão de números inteiros, já que estas instruções não fazem parte do conjunto de instruções da linguagem ASSEMBLER. As subrotinas de ponto flutuante não estão implementadas neste computador.

Existem três níveis de entrada e saída nesta linguagem:

READ / WRITE

RECORD / BLOCK

TRAN

- READ / WRITE: A leitura e gravação são feitas sequencialmente, de cada registro. A transferência, tanto na leitura como na gravação, passa na área de entrada e saída do sistema operacional, para posterior transferência para a área de entrada e saída do usuário no caso de leitura e para o disco, no caso de gravação.

- RECORD / BLOCK: Este nível só é permitido para arquivos contíguos. Os registros ou blocos são numerados logicamente de

0 a n-1, onde n é o número de registros ou blocos do arquivo. Para o RECORD, a trasnferência de um registro, nos dois sentidos de memó rai/disco, é feita diretamente para ou da área de entrada e saída do usuário, enquanto que o BLOCK, a transferência de um bloco é feita diretamente para ou da área de entrada e saída do sistema operacional. No caso de BLOCK há necessidade da transferência entre a área do usuário e do sistema, se isto for conveniente. Dado o número ló gico de um registro ou bloco o acesso é direto.

- TRAN: Transfere um número de palavras começando numa fron teira de setor estipulado pelo usuário e é o mais rápido dos níveis . Para esta transferência tem-se necessidade de dois endereços absolu tos.

  - Endereço absoluto do bloco do disco onde iniciará a trans ferência.
  - Endereço absoluto de memória interna.

A transferência é feita diretamente entre o disco e a á- rea de entrada e saída do usuário. A gravação neste nível é bastan te perigosa, pois a estrutura do dispositivo é ignorada, uma vez que o sistema não testa se os endereços dos blocos pertencem ao arquivo que se quer ter acesso. Só pode ser utilizado para arquivos contí- guos. Muitos dos programas auxiliares do sistema operacional utili zam este nível. Este nível foi o escolhido para implementação do sistema de ordenação externa, já que é o mais rápido na transferên- cia, não limita o número de palavras a transferir e é feito direta- mente entre o disco e a área do usuário.

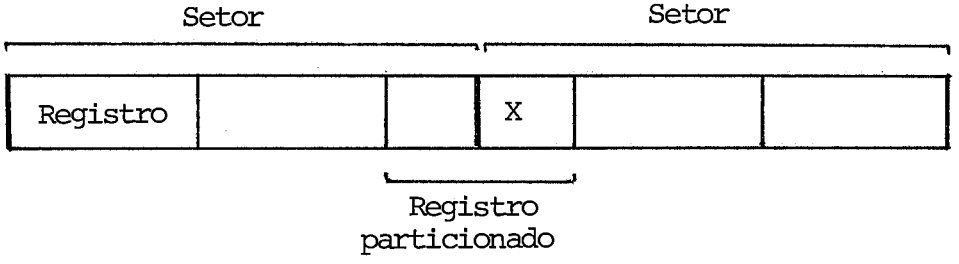
Vários testes foram feitos com TRAN para a verificação de aspectos importantes para o sistema de ordenação, dentre eles:

- Se na leitura ou gravação de vários setores, tempo de

rotação é perdido de um setor para outro. Verificou-se que este tempo não é perdido.

- Se na passagem de uma trilha para outra, dentro do mesmo cilindro, uma rotação é perdida. Não se perde esta rotação.

- Se na gravação de arquivos contíguos, onde o tamanho do setor não é múltiplo do tamanho do registro, o sistema operacional particiona o registro ou inicia em um novo setor (testes com a linguagem FORTRAN também foram feitos). Neste caso, o registro é particionado (Fig. 8), para a utilização de toda a área do disco, pois considerando o caso onde o tamanho do registro fosse 130 palavras, quase metade do setor estaria perdida.



Particionamento de Registro
Figura 8

Há dois pontos negativos neste tipo de utilização. O primeiro é em relação ao tempo perdido para a leitura da parte X (Fig.8) duas vezes, já que nas operações de leitura e gravação de uma quantidade de palavras iniciam na fronteira de um setor. Dependendo do tamanho do registro este tempo é razoável. O segundo é a dificuldade de programação para controle do registro particionado.

3.2 Características do Sistema Implantado

Como o endereçamento das operações de entrada e saída é a nível de setor, foi necessário utilizá-lo como unidade de trabalho e não a palavra como descrito no modelo teórico. Deste fato e de outros que descreveremos nos próximos ítens, o que obtemos foi uma aproximação do modelo teórico.

3.2.1 Verificação dos Parâmetros do Usuário e Possibilidade de Ordenação.

Uma crítica dos parâmetros fornecidos pelo usuário é feita para testes de validades de limitações. Caso todos os parâmetros estejam coerentes, um estudo é feito à parte deste para verificação da possibilidade de ordenação em termos de área disponível no disco e memória interna.

3.2.2 Alocação de Arquivos

Todos os arquivos alocados são contíguos. Tem-se a necessidade de três que descreveremos abaixo:

- Arquivo de trabalho: este arquivo deve suportar duas vezes o arquivo inicial. O controle da divisão deste arquivo em dois fica a cargo do sistema de ordenação externa, evitando assim a alocação de dois arquivos distintos, o que acarretaria em mais movimentos de braço.

- Arquivo de informações das corridas: o tamanho deste arquivo é estimado no início, já que podemos estimar o número de corridas. Neste arquivo constará as seguintes informações:

  - endereço da corrida no disco
  - número de registros das corridas.

O endereço da corrida no disco é dado pelo endereço do setor onde a corrida inicia e o "byte" indicativo do início da corri-

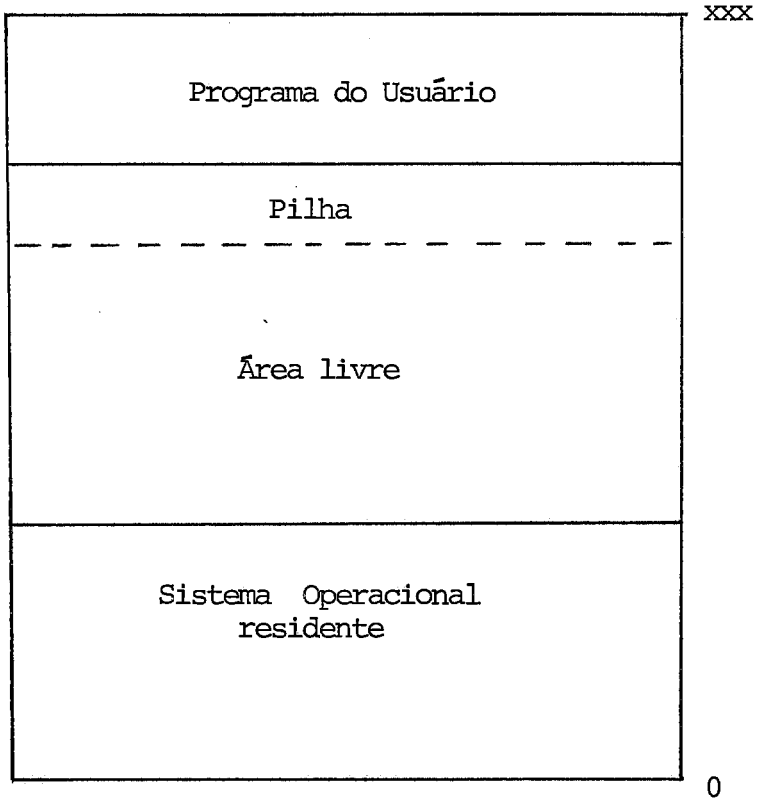
da dentro do setor.

- Arquivo para saída: este arquivo será opcional, já que a saída poderá ser no arquivo inicial que é alocado pelo usuário(vide manual de utilização).

Ao final da ordenação o arquivo de trabalho e o de informações sobre as corridas serão removidos.

3.2.3 Gerência de Memória

A estrutura de memória interna mantida pelo sistema operacional é como mostra a Fig. 9.



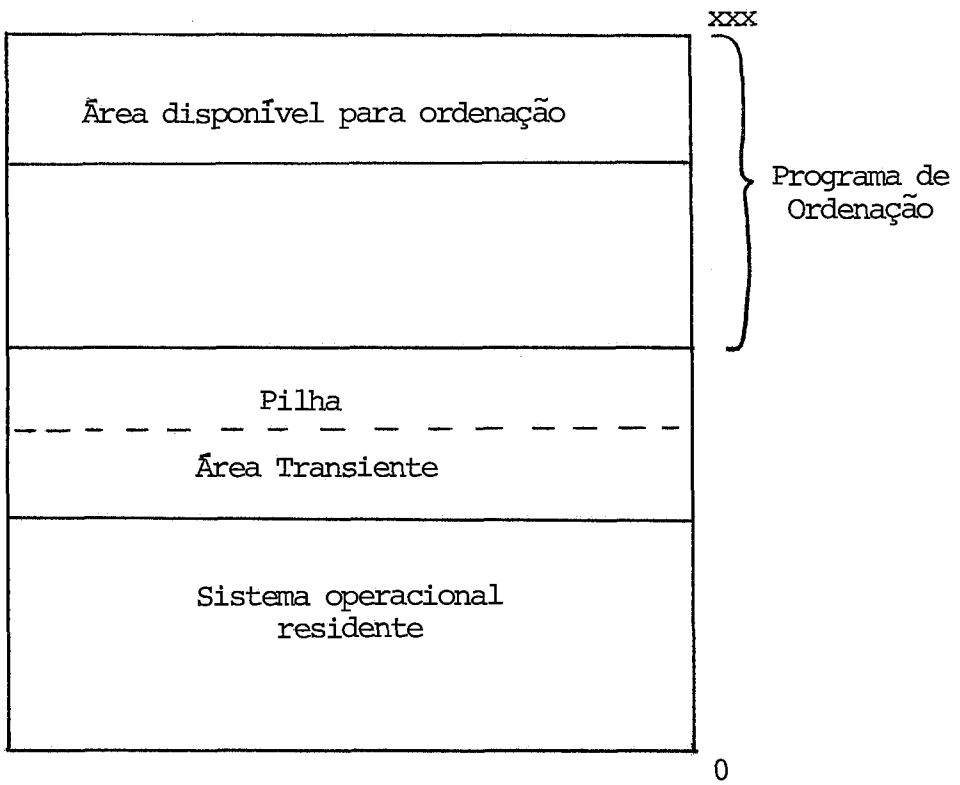
Estrutura da Memória Interna

Figura 9

Existem duas possibilidades de utilização de memória disponível para a ordenação:

- Área reservada dentro do próprio programa (Fig. 10)
- Área de memória livre que está sobre o controle do sistema operacional (Fig. 11)

Na primeira possibilidade tem-se a vantagem de proteção total da área, pois está alocada para o programa. Mas, em compensação não estaríamos utilizando de fato toda a memória disponível num dado momento. Isto porque dentro da área livre (Fig. 11) o sistema operacional possui uma área transiente, que é utilizada conforme a chamada de certas rotinas do sistema operacional para o programa do usuário. Ainda mais quando houvesse expansão de memória do computador a área reservada dentro do programa teria que ser atualizada.



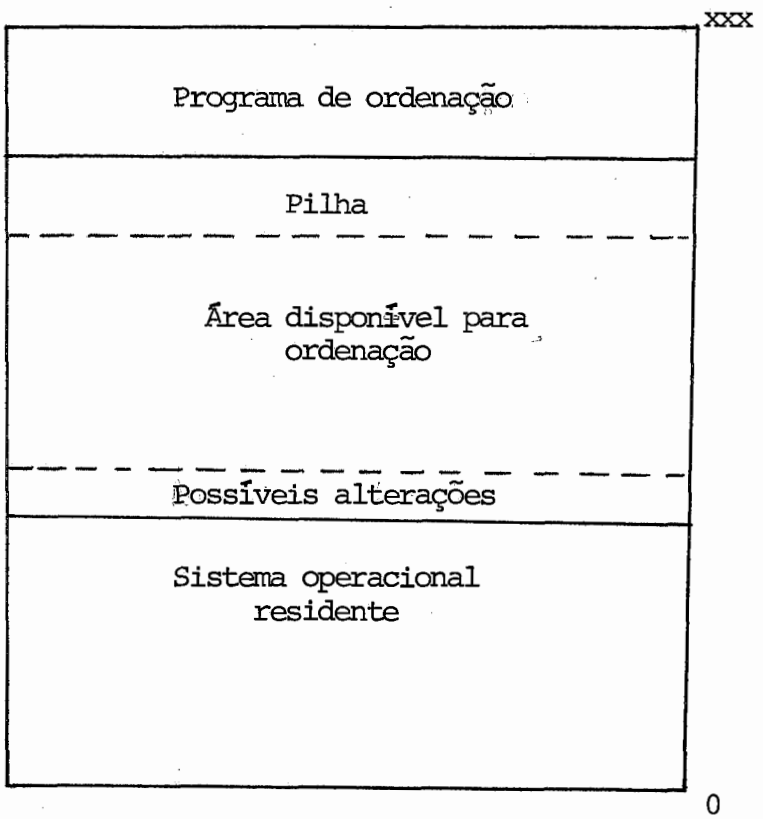
Área de Trabalho dentro do Programa de Ordenação

Figura 10

A segunda possibilidade foi utilizada na implementação.Um levantamento das instruções do sistema de ordenação externa que acarretam o carregamento de subrotinas do sistema operacional e o comportamento da área utilizada pela pilha, foi feito para o controle da área livre como área de trabalho para a ordenação. Com estas informações mais:

- endereço de carga do programa de ordenação
- endereço da última palavra utilizada pelo sistema operacional residente

pode-se prever a área disponível.



Área de Trabalho fora do programa

Figura 11

3.2.4 Áreas de Entrada e Saída

Como foi visto anteriormente, dependendo do tamanho do registro pode ocorrer particionamento no final do setor. Por este motivo, o tamanho das áreas de entrada e saída, tanto para a fase de criação das corridas como para a fase de intercalação, ficou na dependência desta característica. Foi considerado então dois tipos de áreas de entrada e saída:

- Áreas de tamanho de um número exato de setores. Isto quando o tamanho do setor é múltiplo do tamanho do registro;

- Área do tamanho de um número exato de setores mais o tamanho de um registro. Isto quando o tamanho do setor não é múltiplo do tamanho do registro. Na transferência do disco para esta área, a parte final do último registro será lida duas vezes, uma vez que o início de uma leitura ou gravação é a nível de setor. Uma outra inconveniência é que na gravação desta área somente um número exato de setores será transferido e o restante do último registro deverá ser movido para o início da área para ser transferido na próxima gravação.

3.2.5 Criação das Corridas

3.2.5.1 Obtenção do tamanho das áreas de entrada e saída

A teoria descrita no Capítulo II usa como unidade a palavra. Nas implementações dos algoritmos foi necessário a utilização do setor como unidade, isto porque, não seria capaz o acesso a informações se os tamanhos das áreas de entradas e saída fossem menores que um setor, ou também que não fossem múltiplo de setor. As implementações dos algoritmos em outro computador é devido a não existência das rotinas de ponto flutuante no PDP11/10 utilizado.

O programa 1 em ALGOL anexo foi utlizado para a obtenção do valor ótimo de b_{min}, é uma implementação do algoritmo 2 do CapítuloII. Na fórmula do tempo total de ordenação (TO_n) não se considerou o tempo gasto em comparações, já que este tempo é muito particular das características das chaves a serem utilizadas na ordenação. A unidade utilizada é o setor.

O programa foi rodado variando o tamanho da memória interna disponível em setores e para cada variação na memória interna foi feita uma variação no tamanho do arquivo em setores. A cada tamanho de arquivo foi associado um valor de A, já que A varia com este tamanho. O valor de A foi tomado a partir de:

$$A = \text{tempo de meia revolução} + V_o + K.nc$$

O tempo de meia revolução para este disco é da ordem de 20ms. V_o representa o tempo gasto para a inicialização do movimento de braço (10ms). K será dado pela divisão do tempo máximo de movimento de braço menos V_o, pelo número total de cilindros do disco. nc representa o número de cilindros que o arquivo inicial ocupará. Vários testes foram feitos para apurar o valor real de A. O valor de K deu aproximadamente 0.8 enquanto que utilizando os valores nominais dados pelo fabricante, teríamos aproximadamente 0.4.

O valor de B foi estimado em 4ms, considerando que para a transferência de um setor existem palavras para "gaps" e que quando a transferência for de mais de um setor que não estejam no mesmo cilindro, existe um tempo de movimento de braço e revolução.

O valor de D foi calculado a partir dos tempos das instruções utilizadas para transferência de palavras na memória. Considerando que estamos trabalhando com o setor como unidade temos:

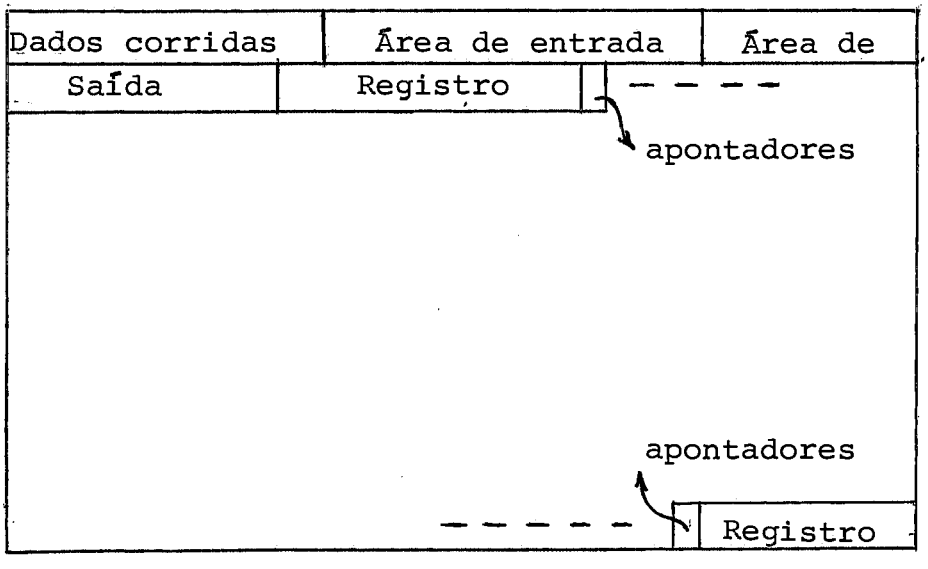
$$D = \frac{Tts \ . \ nc}{2 \ . \ B \ . \ nc}$$

Tts representa o tempo gasto para transferir um setor em memória e é aproximadamente 4ms. Desta forma D é aproximadamente 0.5.

Um estudo sobre os resultados obtidos deste programa ALGOL, possibilitou-nos a obter uma tabela dos b_{min} por tamanho de memória, enquanto que o normal seria obtermos valores de b_{min} relativos aos tamanhos dos arquivos correspondentes à cada tamanho de memória. A tabela 1 apresenta o resultado reduzido, onde associado a cada valor de b_{min} temos a maior diferença percentual. Esta diferença representa o tempo perdido em usarmos um outro valor para b_{min} diferente do calculado pelo programa, para um certo tamanho de arquivo. Como por exemplo na tabela 1-a, temos o que foi calculado pelo programa. Os tempos assinalados representam o tempo mínimo e sua coluna identifica o tamanho de b_{min}. Somente para arquivos de 4 setores b_{min} será igual a 4. Para obtermos a tabela reduzida transformamos este valor para 3. Obtivemos assim uma diferença percentual de 17.5. Esta diferença apresenta-se grande apenas para arquivos pequenos. Uma vez que serão ordenadas sem a utilização da fase de intercalação, a tendência será de utilizar o máximo para o tamanho das áreas de entrada e saída, para minimizar o tempo de ordenação. Para arquivos maiores a diferença não chega a 10% do tempo total de ordenação. A pesquisa nesta tabela é feita sequencialmente, e o valor encontrado será utilizado para tamanho de área de entrada e da área de saída.

3.2.5.2 Controle de memória interna e final de corridas

Nesta fase a memória disponível para a ordenação será subdividida como mostra a figura 12. A área reservada para dados das corridas, armazena o endereço das corridas no disco e a quantidade de elementos de cada corrida. A cada espaço reservado para registro está também o espaço reservado para os quatro apontadores que são necessários, para a implementação do algoritmo "Replecement Selection". Dois destes apontadores poderiam ser calculados a medida que fossem solicitados, não sendo preciso espaço para estes (1).



Estrutura da memória interna fase de criação das corridas

Figura 12

O final de corrida pode ser detectado utilizando-se uma das alternativas:

- gravar uma marca de final de corrida;
- testar a chave de cada registro com a de seu sucessor. Se a do sucessor for menor (ou maior, caso a ordem for descendente) indica final de corrida;
- usar um contador de registro para cada corrida.

A última alternativa é que foi utilizada na implementação.

3.2.6 Obtenção dos Parâmetros para a Intercalação

O programa 2 em ALGOL anexo foi utilizado para a obtenção dos parâmetros para a intercalação, é uma implementação do algoritmo 1 do Capítulo II. Como no programa 1 em ALGOL não foi considerado o tempo gasto em comparações. Os valores estimados para A, B e D são os mesmo utilizados para o programa 1. Foi feita uma variação na memória interna disponível em setores e para cada variação de memória variou-se o número de corridas formadas. A tabela 2 apresenta os resultados obtidos. As colunas relativas à diferença percentual mostra a maior diferença obtida para a troca de certos números de passos. Isto porque entre dois limites de números de corridas havia pontos onde o número de passos não era coerente com os demais, tendo portanto, a necessidade de uma transformação para uniformizar o intervalo. Como por exemplo, o caso de M = 10 como mostra a tabela 2-a. Para 81 e de 101 a 108 corridas o número de passos calculados seria 4, e para uniformizarmos o intervalo de 65 a 160 corridas para três passos, transformamos todos os números de passos iguais a 4. Para esta transformação a maior diferença percentual obtida foi de 3.9. Os tempos assinalados representam os tempos mínimos para a fase de intercalação. Os espaços onde não há tempo

especificado é porque com estes números de passos não é possível a intercalação. O programa de ordenação utiliza uma tabela como a tabela 2, a menos das colunas das diferenças percentuais.

O seguinte algoritmo é utilizado para a pesquisa na tabela: Considerando S o número de corridas geradas, M a memória disponível em setores, T a tabela e V o número em que será feita a intercalação.

1. j=1
2. Se $S<T_{M,j}$ então V=j e pare.
3. Incremente j e vá para 2.

Este algoritmo não é completo, pois a memória disponível M depende do número de passos V por causa da necessidade de se reservar um registro para cada área de entrada e saída quando o tamanho do setor não é múltiplo do tamanho do registro. Um algoritmo que leve isto em consideração será apresentado em seguida.

Com a obtenção do número V de passos que será feita a intercalação, resta saber a ordem de intercalação em cada passo. Estas ordens serão obtidas da tabela 3 que é simplesmente uma tabela de multiplicação ordenada em ordem crescente de produtos. A tabela apresentada é para no máximo quatro passos. Considerando T_j a subtabela correspondente ao número de passos j e S_j o campo de cada entrada da subtabela T_j contendo o produto das ordens de intercalação de cada entrada, o seguinte procedimento é utilizado para a obtenção das ordens de intercalação.

1 - Com o número de passos V obtidos anteriormente, escolha a subtabela T_V.

2 - Escolha a primeira entrada desta subtabela satisfazendo $S_V \geq S$, onde S é o número de corridas geradas. As or

dens de intercalação relativas a esta entrada serão utilizadas.

Do fato das áreas de entrada e saídaterem que usar o tamanho de um registro a mais, quando o tamanho do setor não for múltiplo do tamanho do registro, levou-nos a uma pesquisa nestas tabelas como mostra o procedimento abaixo.

ALGORITMO 3

Seja l - tamanho do registro em "bytes"

S - número de corridas

P_{max} - maior ordem de intercalação em um passo V_i

MS - memória disponível em setores

M - memória disponível em "bytes"

N - número de áreas do tamanho de um registro que sobrou quando a memória foi dividida em setores.

NM - nova memória disponível em setores

V - número de passos

NV - novo número de passos.

P1: Entra com MS na tabela 2 e retira V.

P2: Se V=1 então:

P2.1: Calcule $A = \left\lfloor \frac{M - (S + 1) \cdot 512}{l} \right\rfloor + n$

P2.2: Se $A \geq (S + 1)$ vá para FIM, senão P3

Senão P4

P3: Faça V=2

P4: Calcule P_{max}

P5: $P_{max} + 1 < n$ vá para FIM

senão $P_{max} = P_{max} + 1 - n$

P6: $NM = MS - \left\lceil \frac{P_{max} \cdot l}{512} \right\rceil$

P7: Entre com NM na tabela 2 e retire o novo número de passos NV

P8: Se NV<V vá para P10

P9: Se NV>4 faça NV=4 e vá para FIM, senão FIM.

P10: Faça V=NV e vá para P4

FIM: Faça a intercalação em NV passos

3.2.7 Divisão da Memória Interna Disponível na Fase de Intercalação

A memória interna disponível na fase de intercalação estará dividida em uma área para armazenar informações das corridas, várias áreas de entrada e uma área de saída. Considerando P_i a ordem de intercalação no passo V_i e M a memória interna disponível em setores teremos:

- P_i áreas de entrada de tamanho $EN = \left\lfloor \frac{M}{(P_i + 1)} \right\rfloor$
- Uma área de saída de tamanho $SA = M - EN \times P_i$

Como o setor é a unidade utilizada em geral o tamanho da área de entrada será diferente do tamanho da área de saída. Por exemplo:

Para M = 30 setores de memória disponível para a ordenação e a ordem de intercalação igual a 3 teríamos EN = 7 e SA = 9. Ao invés deste método da divisão poderíamos ter utilizado como valor de EN o mínimo da função: $F(EN) = \frac{1}{EN} + \frac{1}{M - P.EN}$, onde P é a ordem de intercalação. Estudando o comportamento desta função, o mínimo é obtido no ponto $EN = \frac{M}{P + \sqrt{P}}$ para os EN tal que F(EN) 0. Como não podemos trabalhar com valores de EN que não sejam inteiros e como a função é unimodal pode-se provar que o inteiro EN que satisfaz a condição de dar o menor F(EN) seria o maior inteiro menor ou igual a EN ou o menor inteiro maior ou igual a EN. No caso do exemplo acima EN seria igual a 6.34. O maior inteiro menor ou igual a EN seria 6 e o

menor inteiro maior ou igual a EN seria 7. O utilizado para EN seria 6 já que F(6) < F(7). Desta forma este método apresenta uma melhora sobre o método da divisão, onde o valor de EN encontrado foi 7. O programa 3 em ALGOL anexo gerou os valores mínimos de EN que pode ser colocado em forma reduzida como mostra a tabela 4. O que se pode notar do resultado, é que em raras vezes o valor de EN pelo método da divisão deu diferente dos dois inteiros próximos do ponto de mínimo de F e neste caso estaríamos incorrendo em uma diferença de menos de 10% para o valor de F no ponto ótimo. O seguinte procedimento encontraria o valor de EN: seja E_{ij} a tabela 4, d a ordem de intercalação, M a memória disponível e l_i o número de colunas da linha i.

1 - Faça i = d - 1

2 - j=1

3 - Se $M<E_{ij}$ então EN = j e FIM

4 - j=j + 1

5 - Se $j>l_i$ então ERRO

6 - Vá para 3.

3.2.8 Algoritmo para a Intercalação de Registros

Para a escolha da menor ou maior chave para a intercalação, o algoritmo descrito abaixo satisfaz as exigências, uma vez que, em geral as ordens de intercalação são menores que oito.

ALGORITMO 4

Seja P_i a ordem de intercalação

E1: Compare as P_i chaves sequencialmente e encontre as duas menores (maiores)

E2: Mova para a área de saída o registro relativo a menor (maior) das duas chaves escolhidas

E3: Teste a chave do novo registro a entrar com a segunda me-

nor (maior) chave

E4: Se for menor (maior) mova este registro para a área de saída e vá para E3. Caso contrário E5.

E5: Mova o registro relativo a segunda menor (maior) chave para a área de saída e vá para E1

3.3 Minimizando o Tempo de Processamento

A sequência de instruções para uma operação de entrada ou saída são as macro instruções:

. TRAN

. WAIT

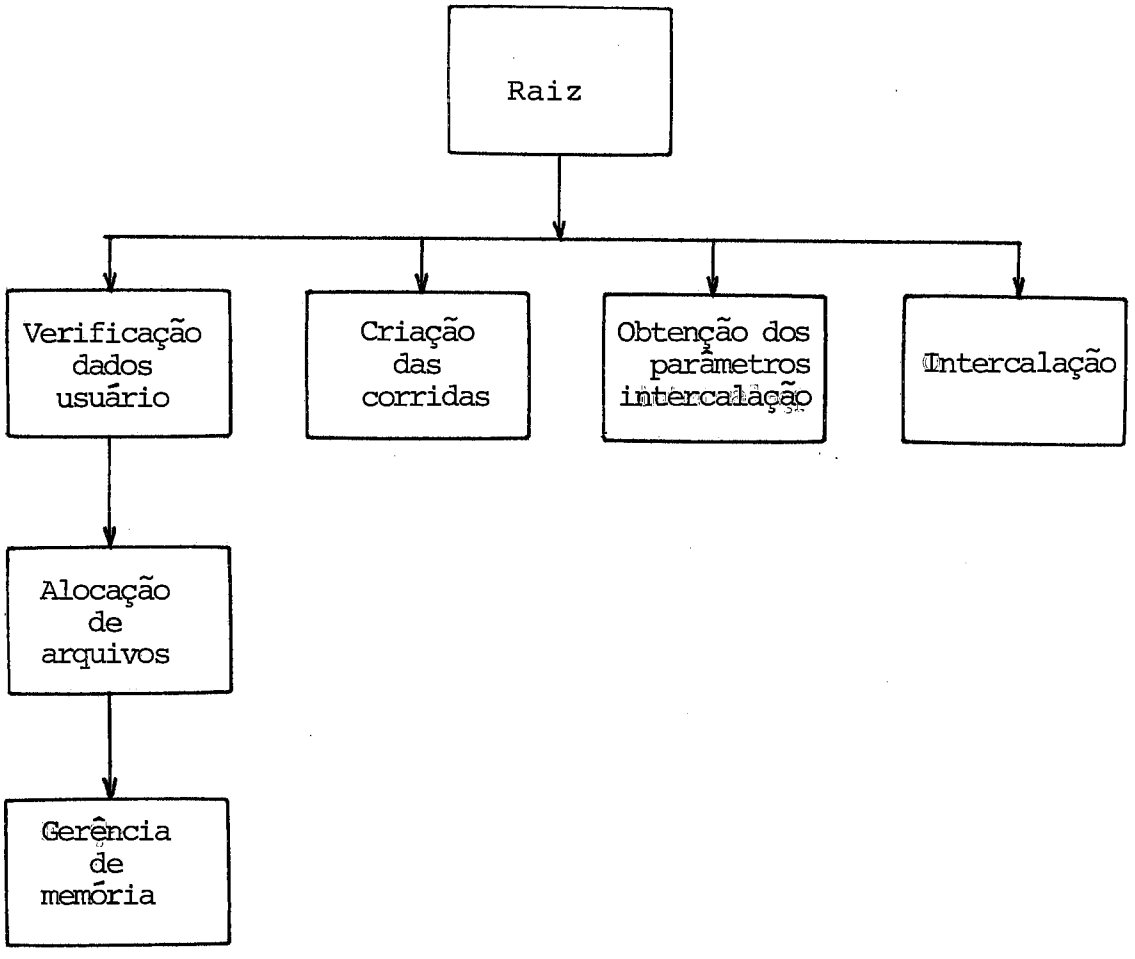
A macro .WAIT é para a espera de complementação integral da transferência de uma operação de entrada ou saída, pois sem esta macro pode-se tentar ter acesso a um dado, sem que este esteja completo. Entre estas duas macros podemos fazer uma série de processamentos que não dependiam dos dados que estariam na área de entrada e saída,com isto minimizando o tempo de processamento.

Como a rotina de teste de chaves é a mais solicitada no sistema de ordenação procuramos diminuir o máximo possível o número de instruções. Para isto a partir das informações do usuário sobre as chaves, o sistema de ordenação monta uma rotina própria para estes tipos de chaves. Isto evita de quando testar, ter que saber se a chave é numérica ou alfanumérica para seguir um procedimento.

3.4 Estrutura de "Overlay"

A independência em termos de sequência de processamento das diversas fases do sistema de ordenação externa, permite a estrutura de "overlay" descrita abaixo, obtendo-se assim mais memória in

terna disponível para as diversas fases. Na implementação não utili_
zou-se estrutura de "Overlay".



Estrutura de "Overlay"

Figura 13

3.3 Manual de Utilização

O sistema de ordenação é independente da linguagem que se utiliza. Trabalhará com um "step" do programa do usuário ou com um único "step".

Existem certas restrições quanto ao uso do sistema:

- O tamanho máximo do registro é de 256 palavras (1 setor)

- O arquivo de entrada deve estar em disco e gravado sem formato. Com isto ocupa-se uma menor área do disco, já que com formato há necessidade de várias palavras de controle.

- O número de chaves para a ordenação é no máximo de três, podendo ser chaves numéricas e alfanuméricas.

Dados para o Sistema

Os dados para a ordenação são lidos da leitora de cartões e é necessário somente um cartão com os seguintes campos (fig. 14):

- Nome do arquivo de entrada: podera ter até dez colunas, sendo as três últimas para extensão do nome do arquivo;

- número de registros do arquivo: número aproximado ou se possível exato. Poderá ter até 5 colunas.

- tamanho do registro em palavras: como foi limitado em 256 o tamanho máximo, poderá utilizar até 3 colunas.

- número de chaves para a ordenação: o número máximo de chaves foi estipulado em 3 chaves, mas chaves alfanuméricas e da mesma ordem de ordenação poderão ser agrupadas em uma só chave, basta que o usuário as coloque consecutivamente no registro.

- tipo de chave: consideraremos chaves numéricas e alfanuméricas. Chave numérica é quando estiver compactada em uma só

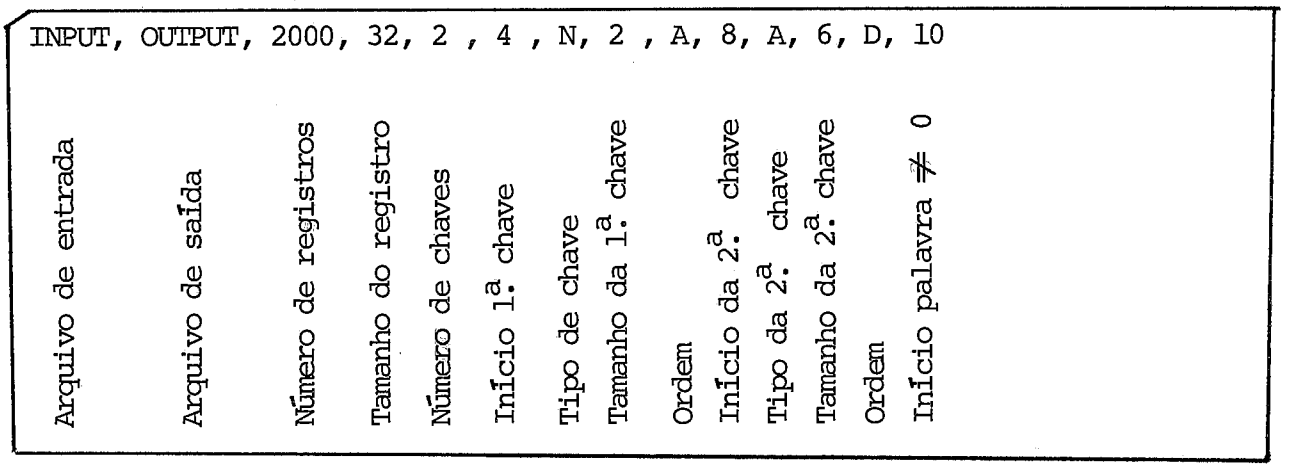
palavra. Isto será indicado com a letra "N" em uma coluna do cartão. Chave alfanumérica é quando cada caractere estiver em um "byte". Será indicado com a letra "A" em uma coluna do cartão.

- Tamanho da chave: este número deverá ser dado em "bytes". Se for chave numérica o tamanho será dois, se for alfanumérica poderá ter até 512 "bytes".

- Ordem para ordenação da chave: indicar com a letra "A" para ordem ascendente e com a letra "D" para descendente.

- Marca de final de arquivo: o usuário terá duas opções para a marca final de arquivo. A primeira será de dar o início da palavra em seu registro que nunca poderá ser zero. A segunda será a de gravar um último registro, tendo em uma de suas palavras um caractere diferente de qualquer outro utilizado em seus registros nesta posição. Neste caso o início da palavra deverá ser dado em uma coluna e em outra coluna o caractere diferente.

Todos os campos do cartão de dados serão separados por vírgula e sem espaço em branco. Caso haja mais de uma chave, os dados relativos a cada uma deverão estar em sequência.



Cartão de dados

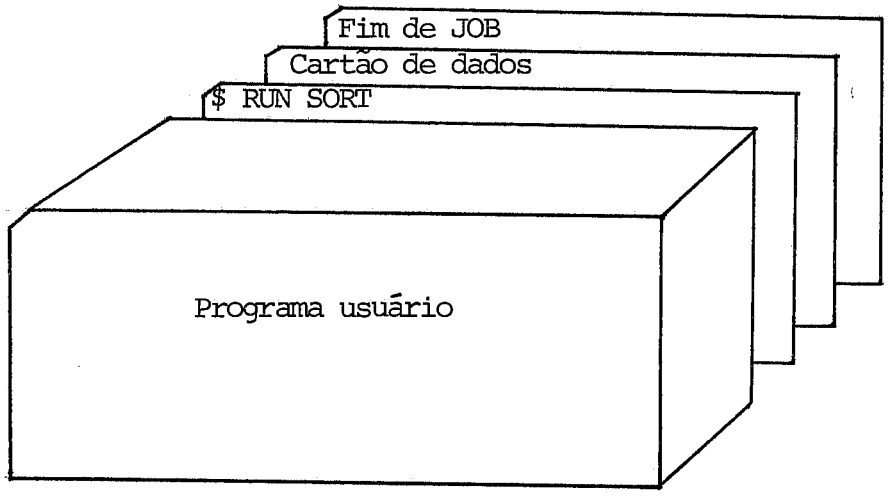
Figura 14

O usuário poderá ter a opção de utilizar o arquivo de entrada como arquivo de saída, bastando que os dois nomes dados no cartão coincidam, caso contrário serão criados dois arquivos.

Chamada do Sistema de Ordenação

Como "Step" do Programa do Usuário

O primeiro "step" como mostra a figura representa o programa do usuário que gravará o arquivo no disco. O segundo "step" apresenta a chamada do sistema de ordenação que está catalogado na biblioteca. Este tipo de utilização é uma chamada em "batch".

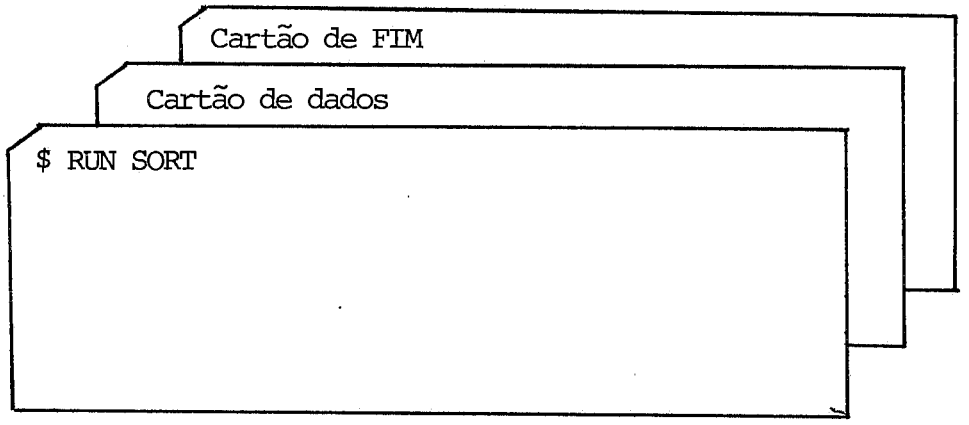


Chamada do Programa em "batch"
Figura 15

Como um Único "Step"

Neste caso o usuário já teria o seu arquivo gravado em disco. Para a chamada do sistema de ordenação terá duas opções:

- Chamada em "batch" como mostra a sequência de cartões indicada na figura 16 na página a seguir.



Sistema de ordenação como um único "step" (batch)

Figura 16

- Chamada pela console, onde na leitora basta ter o cartão de dados. A chamada é dada pelo comando $ RUN SORT.

Mensagens do Sistema de Ordenação

- Dados sobre o Arquivo

Uma verificação da validade dos dados será feita e em caso de erro será impressa a mensagem: ERRO NO CARTÃO DE DADOS.

- Espaço em Disco

Caso não dê para alocar todos os arquivos será impressa a mensagem: NÃO HÁ ESPAÇO EM DISCO. O usuário deverá verificar se existe algum arquivo que possa ser removido. Não esquecer que tem a possibilidade do arquivo de saída ser o mesmo de entrada.

Exemplos Utilizando FORTRAN

- Exemplo 1 -

A utilização do exemplo abaixo é em "batch" e o sistema de ordenação é um "step". O programa gera 3.000 registros aleatórios de tamanho de 32 palavras.

```
$ JOB PROG1
$ RUN FORTRN
  PROG1, LP : < BI : / ON

          DIMENSION INT (32)
          BYTE FILNAM (10)
C  LE NOME DO ARQUIVO DE ENTRADA
          READ (5, 6) FILNAM
       6  FORMAT (10A1)
          CALL SETFIL (1, FILNAM, IERR, "SY", 0)
          DEFINE FILE 1 (3000, 32, U, INDEX)
          K = 17
          INT (2) = K
          J = 1
C GRAVA O PRIMEIRO REGISTRO
          WRITE (1'J) INT
          DO 1  J = 2, 3000
          K = 5 * K + 1 - 4096  *  IFIX ((5 * K + 1)/4096)
          INT (2) = K
C GRAVA OS DEMAIS REGISTROS
       1  WRITE (1'J) INT
          CALL EXIT
          END
$ RUN LINK
  PROG1, LP :   < PROG1, FTNLIB/L/E
$ RUN PROG1
```

```
     Cartão de dados
$ RUN SORT
     Cartão de dados
$ FINISH
```

O cartão de dados para o programa em FORTRAN terá o nome do arquivo, que poderá ter até dez colunas, onde três colunas são para a extensão do nome se existir. O cartão de dados para o programa de ordenação tem os seguintes campos:

```
INPUT, INPUT, 3000 , 32 , 1, 3, N, 2, A, 10
```

Neste exemplo o final do arquivo é detectado verificando a existência de zero na palavra iniciando no "byte" dez.

- Exemplo 2 -

```
$ JOB PROG2
$ RUN FORTRN
  PROG2, LP : < BI : / ON

          DIMENSION NOME (40), IFIR (23)
          BYTE FILNAM (10)
          DATA IAST / * /
C LE NOME DO ARQUIVO DE ENTRADA
          READ (5, 6) FILNAM
        6 FORMAT (10A1)
          CALL SETFIL (1, FILNAM, IERR, "SY", 0)
          DEFINE FILE 1 (2000, 64, U, I)
          DO 1    I = 1, 2000
          READ (5, 7) NOME, IFIR, ISALA
        7 FORMAT (40A2, 23A2, I5)
          WRITE (1'I) NOME, IFIR, ISALA
```

```
          IF (NOME(1) - IAST) 1, 2, 1
        1 CONTINUE
        2 I = I - 1
C         IMPRIME O NÚMERO DE REGISTROS GRAVADOS
          WRITE (6, 11) I
       11 FORMAT (I5)
          CALL EXIT
          END
$ RUN LINK
  PROG2, LP : < PROG2, FTNLIB / L / E
$ RUN PROG2
    ¦   Cartões de dados
    ¦
$ FINISH
```

Este programa grava I registros. O primeiro cartão de dados contém o nome do arquivo de entrada, por exemplo INP, os demais cartões são os registros que serão gravados, sendo que o último cartão só terá um asterisco (*) na primeira coluna para indicar final de dados. O cartão de dados para a ordenação contém os seguintes campos:

INP, OUT, 4500, 64, 2, 1, A, 80, A, 127, N, 2, D, 1, *

Estamos supondo I igual a 4500 registros. A chamada do sistema de ordenação pode ser feita em "batch", colocando um cartão com $ RUN SORT antes do cartão de dados para a ordenação ou fora do "batch" dando este comando pela console.

IV - RESULTADOS EXPERIMENTAIS

Os testes feitos utilizaram o gerador de números aleatórios descritos abaixo para obtenção das chaves.

```
      K(1) = 17
      DO 1  J = 2,M
      A = 5. * K(J - 1) + 1.
      A = A - 4096. * IFIX (A/4096.)
    1 K(J) = A
```

Os tamanhos dos registros foram de 32 palavras para alguns arquivos e 40 palavras para outros. A ordenação foi segundo uma chave numérica (1 palavra). A memória interna utilizada foi de 10 e 20 setores de 256 palavras. Os resultados obtidos na implementação estão apresentados na tabela 5. Nesta tabela pode-se observar a importância da escolha ótima para b_{min} ao contrário de se utilizar um setor para área de entrada e um para área de saída. Outro ponto a observar é em relação ao decréscimo de tempo devido ao fato de se aumentar a memória interna disponível, onde duplicando-se a memória obteve-se um decréscimo em alguns casos em mais de 50% do tempo. A tabela 6 apresenta os tempos de transferência, movimento de braço e CPU em relação as fases de ordenação. Como era de se esperar o tempo de CPU predominante é na fase de criação de corridas. Este tempo e o tempo de movimento de braço se equivalem devido a escolha ótima para b_{min}. No caso de se utilizar um setor para tamanho das áreas de entrada e saída o tempo mais acentuado nesta fase seria o de movimento de braço. A tabela 8 apresenta os percentuais em relação ao tempo total de ordenação, do tempo de transferência , movimento de braço e CPU. O tempo predominante de fato é o tempo de movimento de braço, mas que com um aumento de memória já tende

a equipar-se com o tempo de CPU. O gráfico 1 apresenta os tempos de intercalação real e estimado. No tempo estimado, em relação ao tempo de CPU, só se considerou o tempo de movimentação interna de registros. O gráfico 2 apresenta a velocidade de ordenação em Kb/segs. Os tempos estimados foram obtidos à partir da fórmula IV, mas com o conhecimento exato do número de corridas. Os tempos dados na tabela 6 foram discriminados a partir da fórmula IV também, a não ser os tempos de CPU que foram tomados como o que faltou para completar o tempo real obtido.

CONCLUSÃO

Um dos problemas importantes a discutir nesta fase é sobre a estimativa de G na implementação do algoritmo 1 (Cap. II). Veremos o que uma variação nesta estimativa acarreta com relação ao tempo intercalação, uma vez que a estimativa feita leva em consideração parâmetros que nem sempre sabemos de suas totais validades (distância percorrida pelo braço, tempo para percorrer esta distância, etc). Para isto foram geradas as tabelas 9, 10 e 11 à partir do algoritmo 1 para G fixos 5, 6 e 7 respectivamente. Estes valores apresentam as variações no tempo de movimento de braço (médio) para arquivos médios (G=5) até para os maiores arquivos que possam armazenar no disco (G=7). Os espaços em branco nestas tabelas indicam a não possibilidade de intercalação do número de corridas correspondentes no especificado número de passos. Seja M = 15, TARQ = 910 e S =65 onde M é memória interna disponível em setores, TARQ tamanho do arquivo em setores S o número de corridas. Neste exemplo o valor de G que deve aproximar mais do valor real é 7, e da tabela correspondente a este G obtemos como número de passos para a intercalação 3.

Caso o valor de G fosse estimado em 5 teríamos da tabela correspondente, 2 como número de passos para a intercalação. Em termos de tempo de

intercalação podemos ver a partir da tabela 11-a que diferença acarretaria fazer a intercalação em 2 passos e não em 3. Os tempos sublinhados representam o menor tempo indicando o número de passos para o valor de S correspondente. Desta forma obtemos uma diferença percentual 2.7 em relação aos tempos correspondentes a 2 e 3 passos respectivamente, indicando que uma variação grande em G não leva a uma diferença de tempo considerável. A mesma variação de G (5, 6 e 7) foram aplicadas na implementação do algoritmo 2 (Cap. II) para obtenção do b_{min}. Os resultados estão apresentados na tabela 12 e como se pode observar para as diferentes variações de G os valores de b_{min} são identicos, com isto estamos cientes que grandes variações no valor estimado de G não acarreta grande diferença no tempo de ordenação.

A maioria das tabelas apresentadas na implementação que tem como entrada a memória interna disponível foram geradas para diferentes valores de memória prevendo uma expansão de memória interna. Poderia gerar novas tabelas quando houvesse uma expansão de memória e utilizar tabelas reduzidas para a configuração atual. Com este estudo somos levados a acreditar, que se tem mais eficiência na obtenção dos parâmetros de ordenação por meio de tabelas, ao contrário da obtenção pela execução dos algoritmos no momento da ordenação.

Um outro ponto importante é a verificação se esta teoria se aplicar a outros sistemas de computador. Como se pode notar esta teoria é perfeitamente adaptável a qualquer computador

com um disco setorizado ou não. Pode ser inclusive adaptada a um sistema que tenha dois discos, com o seguinte procedimento: na 1ª. fase teremos o arquivo inicial em um disco e um arquivo no outro disco para armazenamento das corridas. Tanto para um disco como para o outro nesta fase o braço se moverá sequencialmente . Desta forma para cada acesso teríamos somente um tempo para inicialização de movimento, mais o tempo de espera (meia revolução), já que o tempo de mudança de cilindro está incluido no tempo de transferência. O tempo total normalizado desta fase (TC_n) será dado por:

$$TC_n = \frac{\hat{R}_1}{2.B.L.l} + 1 + 2.D + \frac{2.G_1}{b} \qquad \text{onde}$$

$$G_1 \cong \frac{A_1}{2.B} \quad \text{e} \quad A_1 = \text{tempo de meia revolução}$$

Para a fase de intercalação teremos o arquivo de trabalho 1 em um disco e o arquivo de trabalho 2 no outro disco. Para o arquivo que irá trabalhar como saída o movimento de braço será sequencial utilizando para este caso o G_1 definido acima.

Para o arquivo que irá trabalhar como entrada, o movimento de braço dependerá dos tamanhos das corridas e da ordem de intercalação. Tem-se então a necessidade de estimar A_2 (tempo mé

dio de movimento de braço mais meia revolução) para este fim. O tempo normalizado de intercalação seria dado por:

$$TI_n = \sum_{j=1}^{V} \{\frac{\hat{R}_2(j)}{2.B.L.1} + (1 + D) + \frac{G_2}{b_{ej}} + \frac{G_1}{b_{sj}}\}, \text{ onde}$$

$$G_2 = \frac{A_2}{2.B}$$

As variáveis utilizadas são as mesmas da fórmula (IV) a menos de G_1, A_1, G_2 e A_2.

Alguns aspectos serão apresentados agora, como sugestões para implementações futuras.

A utilização do método de raiz discutido anteriormente, para a obtenção do tamanho das áreas de entrada e saída para fase de intercalação, trará uma redução no tempo de movimento de braço. Para sua implementação basta substituir na fórmula (IV) o valor de $b_e = \frac{M}{P+1}$ por $b_e = \frac{M}{P+\sqrt{P}}$ ~.

Na apresentação teórica (Cap. II), foi estipulado uma área de entrada e uma área de saída para a fase de criação de corridas. O fato de que no algoritmo de 'Replecement selection' a cada registro que é retirado da árvore, um novo registro da entra

da toma seu lugar, sugere a utilização de uma só área de entrada e saída. Considerando o disco setorizado a escolha do tamanho desta área única será feita pelo algoritmo 2 (Cap. II), apenas considerando como o número esperado de corridas $E(S) = \frac{L.\ 1}{2.(M-b)}$.

O tamanho do registro deverá ser considerado para a definição desta área. Supondo em primeiro lugar que o tamanho do setor seja um múltiplo do tamanho do registro deveremos ter:

1a - A área reservada para registros e ponteiros deverá armazenar um número de registros múltiplo do número que caibam em um setor.

2a - A área de entrada e saída deverá ser como mostra a figura 17.

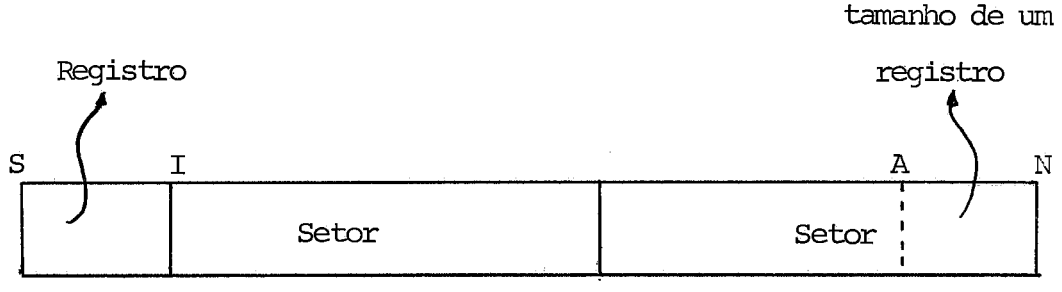


Figura 17

De S a A teremos a área de saída e ou I a N teremos a área de entrada. Com o registro e mais a esquerda sempre teremos condições de armazenar o registro que sai da árvore. Supondo agora que o tamanho do setor não seja múltiplo do tamanho do registro deveremos ter:

1b - A área reservada para registros e ponteiros deverá armazenar um número de registros que caibam em uma quantidade exata de setores e se necessário esta quantidade mais um registro.

2b - A área de entrada e saída deverá ser como mostra a figura 18.

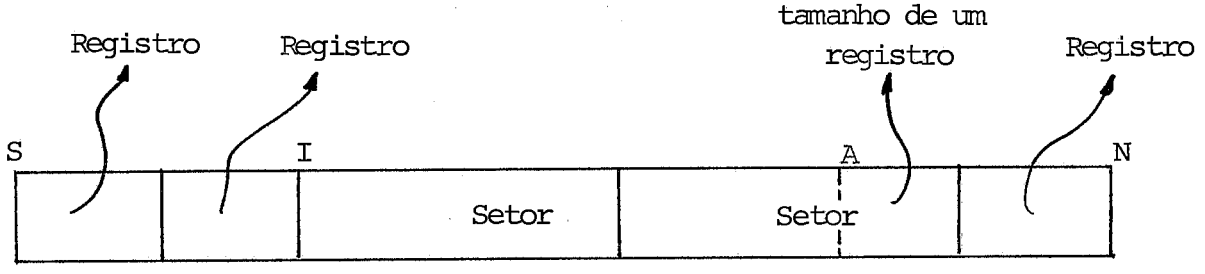
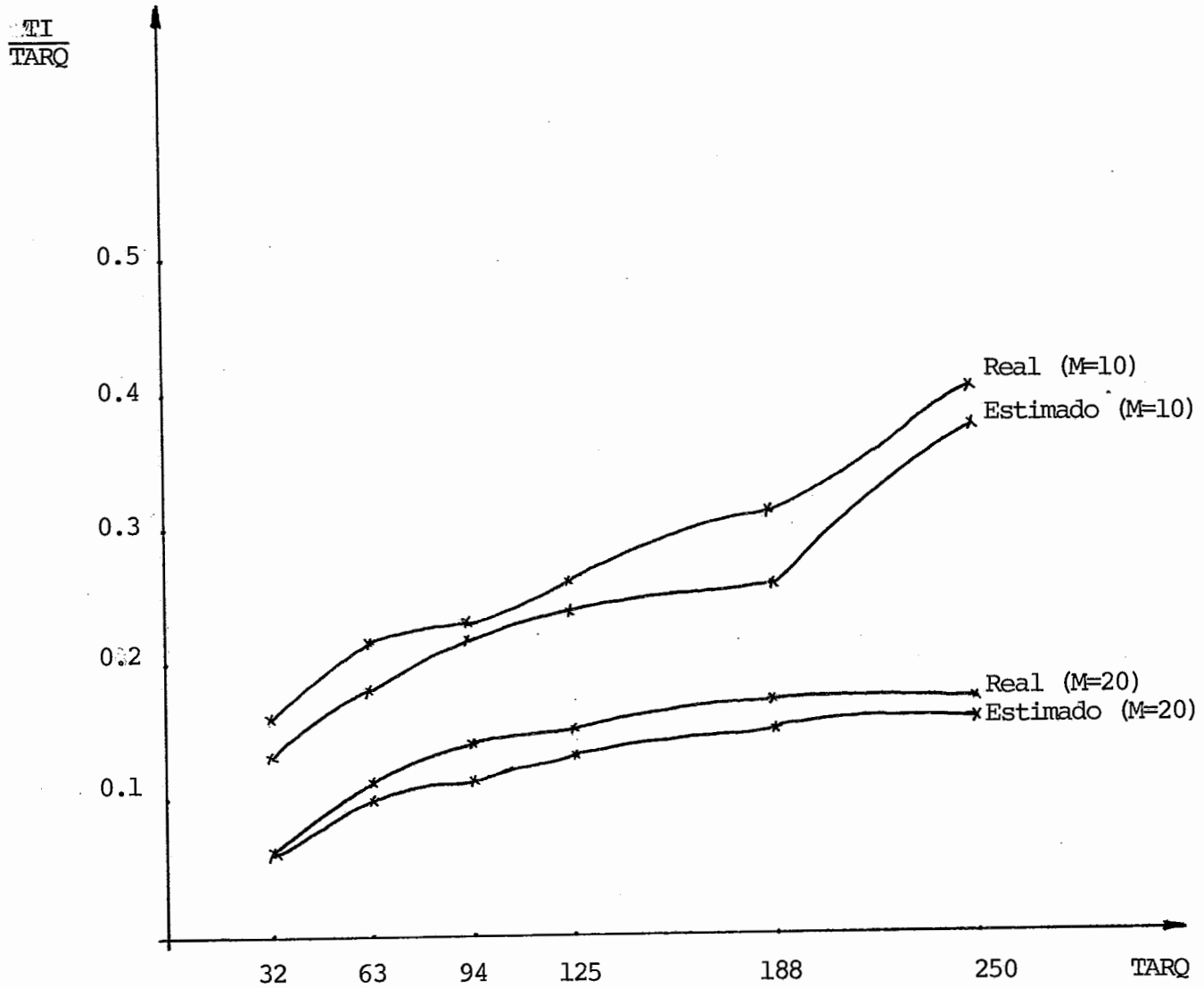


Figura 18

A área para saída será de S a A e a área de entrada de I a N. É necessário a utilização de dois registros a esquerda, devido ao fato

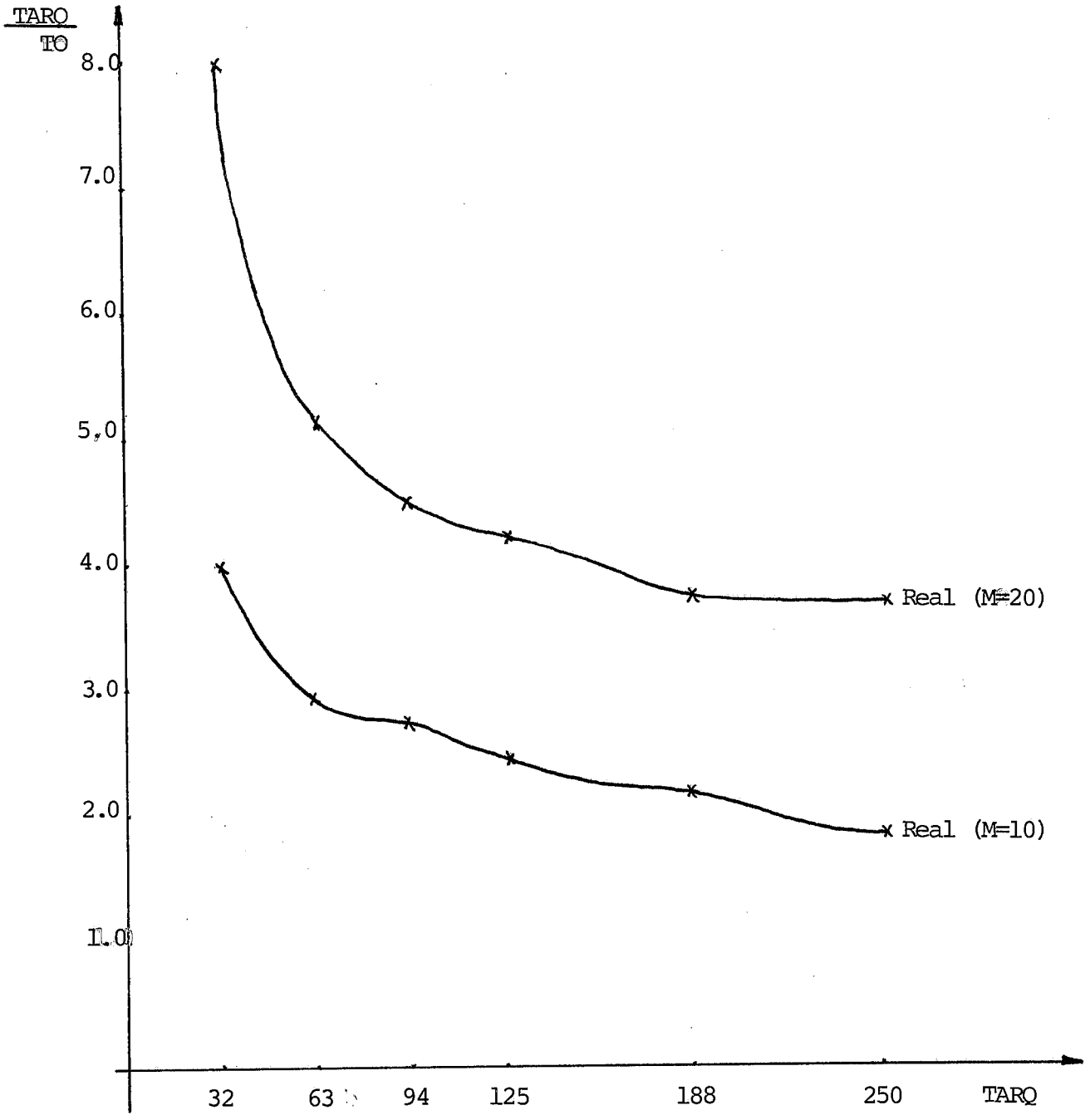
de que sempre gravamos um número exato de setores, e o restante do registro particionado será movido para o início da área de saída. Desta forma se tivessemos um só registro a esquerda e se na leitura o início do primeiro registro estivesse em I, não haveria lugar para o registro que saísse da árvore. A primeira gravação nos dois casos discutidos poderá ser de um número menor de setores do que o estipulado para esta área, isto porque para o preenchimento da árvore pode ter sido necessário várias leituras e nem todos os registros foram utilizados. As restrições de 1a e 1b garante neste caso, que o próximo registro é o primeiro de um setor. Sem estas restrições poderíamos ter a necessidade de uma nova leitura sem que a área ou saída estivesse cheia. Com este novo esquema de entrada e saída o tamanho desta área será bem maior, diminuindo o tempo de movimento de braço tanto para a entrada como para saída.



TI = tempo da fase intercalação (segs)

TARQ = tamanho do arquivo em Kb

Gráfico 1



TO = tempo total de ordenação real (segs.)

TARQ = tamanho do arquivo

Velocidade de ordenação em Kb/seg

Gráfico 2

BIBLIOGRAFIA

1. KNUTH, D. E. - "The art of computer programming", Vol. III - Addison - Wesley Reading, Mass.

2. GASSNER, B. J. - "Sorting by Replacement Selection" Communications of the ACM, Vol. 10/nº 2/February 1967, pp. 89-93

3. BLACK, N. A. - "Optimum Merging from Mass Storage" Communications of the ACM - Vol 13/nº 12/December 1970, pp. 745-749

4. HYAFIL, L.; PRUSKER, F.; VUILLEMIN, J. - "Design of optimal merge on direct access devices" - Proceedings IFIP 74

5. SCHLUMBERGER, M.; VUILLEMIN, J. - "Optimal Disk Merge Patterns" Acta Informatica 3 - 1973, pp. 25-35

6. "SIMPOSIUM OF SORTING", Communications of the ACM, Vol. 6/nº 5/ May, 1963

7. DOS/BATCH Monitor, Programmer's Manual, DEC-11 - OMPMA-A-D

8. DOS/BATCH Assembler (Macro), Programmer's Manual DEC-11-LASMA-A-D

9. DOS/BATCH Fortran Compiler and Object Time Systems. Programmer's Manual . DEC-11-LFRTA-A-D

10. DOS/BATCH System Manager's Guide. DEC-11-OSMGA-A-D

11. DOS/BATCH Debugging Program (ODT-11R). Programmer's Manual DEC-11-UDEBA-A-D

12. DOS/BATCH Linker (LINK). Programmer's Manual. DEC-11-ULKAA-A-D

13. DOS/BATCH Librarian (LIBR). Programmer's Manual . DEC-11-ULBAA-A-D

14. DOS/BATCH Text Editor (EDIT). Programmer's Manual. DEC-11-UEDAA-A-D

15. PERIPHERALS AND INTERFACING HANDBOOK - Digital Equipment Corporation, 1971

16. PROCESSOR HANDBOOK - Digital Equipment Corporation, 1973

APÊNDICE A

Memória (setores)	Diferença %	b
10	17.5	3
11	17.5	3
12	36.0	3
13	30.5	3
14	41.0	3
15	21.0	4
16	28.0	4
17	28.0	4
18	20.0	5
19	20.0	5
20	13.0	6
21	13.7	6
22	25.5	6
23	25.0	6
24	40.0	7
25	40.0	7
26	52.0	7
27	52.0	7
28	60.0	8
29	58.0	8
30	68.0	8
31	68.0	8
32	19.0	8
33	19.0	8
34	16.0	9
35	16.0	9
36	12.0	10
37	12.0	10
38	12.0	11
39	12.0	11
40	12.0	12

Tabela do bmin

Tabela 1

M = 10 setores

Setor = 512 bytes

Tamanho do arquivo (setores)	G	Tamanho de b em setores			
		1	2	3	4
4	4.0	10.0	6.0	4.7	4.0
8	4.0	10.0	6.0	4.7	7.8
16	4.0	10.0	9.8	8.5	9.5
32	4.0	13.8	10.5	10.2	11.5
64	5.0	18.5	14.8	14.3	17.5
128	5.0	21.0	18.8	18.3	20.0
256	5.0	25.0	22.0	20.8	22.5
512	6.0	32.0	27.0	27.0	31.4
1024	7.0	40.0	37.3	36.7	37.4

$E\{TO_n\}$ como função do tamanho do buffer usado na fase de criação de corridas

Tabela 1-a

Memória em Setores	Diferença %	Limite Nº corridas 1 passo	Diferença %	Limite Nº corridas 2 passos	Diferença %	Limite Nº corridas 3 passos	Diferença %	Limite Nº corridas 4 passos
10		8		64	3.9	512		750
11		9		81	10.6	312		750
12		10		90	1.1	100		400
13		11		30	0.8	125		500
14		12		30		150		625
15		13		42		216		750
16		13		42		216		624
17		14		49		216		400
18		14		49	4.9	343		750
19		15		56		216	3.34	400
20		17		64		392		499
21		16		72		150	1.2	475
22	1.5	15		72	0.5	180		454
23		10		72	1.0	216		434
24		11		100	7.0	216		400
25	4.6	14		100		294		320
26		11		90	0.4	294	0.4	384
27		12		56	1.8	294	0.6	369
28		12		56		150		320
29		13		64		180		320
30		13		72	2.9	332		400
31		14		72		321		450
32		14	1.5	72	1.0	216		312
33		14		72	2.2	216		302
34		15		76	2.9	216		300
35		15		81		180		192
36		16	5.8	100	1.7	276		400
37		16	1.4	100	1.0	269		400
38		17		96	1.4	262		400
39		17		90		252		300
40		18		72	1.1	249		300

Obtenção do Número de Passos em Função de M(Memória disponível) e S(Número de corridas)

Tabela 2

Número de corridas	Número de passos	Tempo total normalizado de intercalação			
		1 passo	2 passos	3 passos	4 passos
2	1	3.7			
⋮					
8	1	7.5	9.0	11.5	
9	2	9.6	9.1	12.3	15.5
⋮					
64	2	-	20.6	22.2	23.7
65	3	-	23.7	23.4	23.8
⋮					
80	3	-	28.7	25.0	25.3
81	4	-	28.8	26.4	25.4
⋮					
100	3	-	-	28.6	29.0
101	4	-	-	30.1	29.1
⋮					
108	4	-	-	31.0	29.9
109	3	-	-	31.1	31.8
⋮					
160	3	-	-	38.1	40.1

Número de Passos para M = 10

Tabela 2-a

2 passos			3 passos				4 passos				
ordens de intercalação		número de corridas	ordens de intercalação			número de corridas	ordens de intercalação				número de corridas
2	2	4	2	2	2	8	2	2	2	2	16
2	3	6	2	2	3	12	2	2	2	3	24
3	3	9	2	3	3	18	2	2	3	3	36
3	4	12	3	3	3	27	2	3	3	3	54
4	4	16	3	3	4	36	3	3	3	3	81
4	5	20	3	4	4	48	3	3	3	4	108
5	5	25	4	4	4	64	3	3	4	4	144
5	6	30	4	4	5	80	3	4	4	4	192
6	6	36	4	5	5	100	4	4	4	4	256
6	7	42	5	5	5	125	4	4	4	5	320
7	7	49	5	5	6	150	4	4	5	5	400
7	8	56	5	6	6	180	4	5	5	5	500
8	8	64	6	6	6	210	5	5	5	5	625
8	9	72	6	6	7	252	5	5	5	6	750
9	9	81	6	7	7	297	5	5	6	6	900
9	10	90	7	7	7	343	5	6	6	6	1080
10	10	100	7	7	8	392	6	6	6	6	1296
10	11	110	7	8	8	448	6	6	6	7	1512
11	11	121	8	8	8	512	6	6	7	7	1764
11	12	132	8	8	9	576	6	7	7	7	2058
12	12	144	8	9	9	648	7	7	7	7	2401
13	12	156	9	9	9	729	7	7	7	8	2744
13	13	169	9	9	10	810	7	7	8	8	3136
13	14	182	9	10	10	900	7	8	8	8	3584

Ordens de Intercalação

Tabela 3

Ordem de intercalação d	Memória disponível M										
2	6	9	13	16	19	23	26	30	33	36	40
3	8	13	17	22	27	31	36				
4	10	16	22	28	34	40					
5	12	19	26	33	40						
6	14	22	30	39							
7	16	25	35								
8	18	29	39								
9	20	32									
10	22	35									
11	24	38									
12	26										
13	28										
14	30										
15	32										

Tabela de tamanhos ótimos de áreas de E/S

Tabela 4

Memória em Setores	b	Número de Passos	Tamanho Arquivo Registros	Tamanho Registro Palavras	Número de Corridas	Tempo fase Criação (seg)	Tempo fase Intercalação (seg)	Tempo Total (seg)
10	3	2	500	32	10	3	5	8
10	3	2	1000	32	16	8	14	22
10	3	2	1500	32	27	11	22	33
10	3	2	2000	32	37	16	34	50
10	3	2	3000	32	53	23	59	82
10	3	3	4000	32	71	35	99	134
10	1	1	500	32	5	8	4	12
10	1	2	1000	32	10	16	13	29
10	1	2	1500	32	14	23	21	44
10	1	2	2000	32	19	31	31	62
10	1	2	3000	32	27	46	48	94
20	6	1	500	32	5	2	2	4
20	6	1	1000	32	10	5	7	12
20	6	1	1500	32	14	8	13	21
20	6	2	2000	32	18	10	19	29
20	6	2	3000	32	27	18	32	50
20	6	2	4000	32	36	23	43	66
20	6	1	1000	40	12	6	11	17
20	6	2	2000	40	23	13	26	39
20	6	2	3000	40	35	22	44	66

Resultados Experimentais

Tabela 5

		Criação de corridas			Intercalação		
Memória em Setores	Tamanho Arquivo (Registros)	Tempo de Transferência	Tempo de Movimento de braço	Tempo de CPU	Tempo de Transferência	Tempo de Movimento de braço	Tempo de CPU
10	500	0.5	1.3	1.2	0.5	3.3	1.2
10	1000	1.0	2.8	4.2	2.0	8.5	3.5
10	1500	1.5	4.4	5.1	3.0	16.2	2.8
10	2000	2.0	6.4	7.6	4.0	23.6	6.4
10	3000	3.0	10.5	9.5	6.0	40.3	12.7
10	4000	4.0	15.4	15.6	12.0	73.6	13.4
20	500	0.5	0.6	0.9	0.5	1.0	0.5
20	1000	1.0	1.4	2.6	1.0	4.7	1.3
20	1500	1.5	2.2	4.3	1.5	7.8	3.7
20	2000	2.0	3.2	4.8	4.0	9.8	5.2
20	3000	3.0	5.2	9.8	6.0	18.3	7.7
20	4000	4.0	7.7	11.3	8.0	29.0	6.0

Tamanho do registro = 64 "bytes"

Tempo em segundos

Resultados Experimentais

Tabela 6

Memória em Setores	Tamanho Arquivo (Registros)	Criação de corridas			Intercalação		
		Tempo de Transferência	Tempo de movimento de braço	Tempo de CPU	Tempo de Transferência	Tempo de Movimento de braço	Tempo de CPU
10	500	17	43	40	10	66	24
10	1000	12	35	53	14	61	25
10	1500	14	40	46	14	74	12
10	2000	12	40	48	12	70	18
10	3000	13	46	41	10	68	22
10	4000	11	44	45	12	74	14
20	500	25	30	45	25	50	25
20	1000	20	28	52	14	67	19
20	1500	19	27	54	11	60	29
20	2000	20	32	48	21	52	27
20	3000	17	29	54	19	57	24
20	4000	17	33	50	19	67	14

Tamanho do registro = 64 "bytes"

Resultados Experimentais

Percentual em relação do tempo
total de cada fase

Tabela 7

Memória em Setores	Tamanho Arquivo (Registros)	Tempo Total		
		Transferência	Movimento de braço	CPU
10	500	18	57	25
10	1000	14	52	34
10	1500	13	63	24
10	2000	12	60	28
10	3000	11	62	27
10	4000	12	66	22
20	500	24	42	34
20	1000	17	51	32
20	1500	14	47	39
20	2000	21	45	34
20	3000	18	47	35
20	4000	18	55	27

Tamanho do registro = 64 "bytes"

Resultados Experimentais

Percentagem em relação ao tempo total real

Tabela 8

Memória em Setores	Diferença %	Limite Nº Corridas 1 passo	Diferença %	Limite Nº Corridas 2 passos	Diferença %	Limite Nº Corridas 3 passos	Diferença %	Limite Nº Corridas 4 passos
10		8		72	2.3	576		901
11		9		81	6.2	729		
12		10		100	4.7	811		
13		11	4.4	100		150		730
14		12	4.4	90		180		626
15		13	4.4	90		252		626
16		13		49		252		501
17		14		49		343		501
18		14		56	1.4	392	1.0	500
19	1.5	15		64		512		
20		17		72		449		
21		17		81	1.4	449		
22	3.0	14		81		449		
23		10		90		393		
24	6.1	18		110	2.8	393		
25	6.1	18		110		393		
26		11		121		344		
27		12	1.0	121		344		
28		12		144	1.0	344		
29		13		144		344		
30		13		169		295		
31		14		169		295		
32		14		169		295		
33		15	6.6	182		295		
34		15		196		253		
35		16		225		253		
36		16	2.1	225		253		
37		17		240		253		
38		17		240		253		
39		17	2.2	253				
40	1.2	18		253				

Obtenção do Número de Passos em Função de M e S para G fixo igual a 5.0

Tabela 9

Memória em Setores	Diferença %	Limite Nº Corridas 1 passo	Diferença %	Limite Nº Corridas 2 passos	Diferença %	Limite Nº Corridas 3 passos	Diferença %	Limite Nº Corridas 4 passos
10		8		64	3.8	576		901
11		9	0.6	73	8.0	729		
12		10		100	6.3	810		
13		11	6.4	100		150	4.5	730
14		11	3.0	30		180		626
15		13	6.5	100		252		626
16		13	3.5	49		252		501
17		13		49		294	3.0	501
18	2.5	14		56		392		500
19	5.2	14		64		448	2.6	513
20		16		72		449		
21	1.3	14	0.8	81	1.2	449		
22		10		81		449		
23		10		90		393		
24		10		110	4.5	393		
25		11		110		393		
26		11		121		344		
27		12	2.8	121		344		
28		12	1.8	144		344		
29		13	1.0	144		344		
30		13	1.0	156		295		
31		14		169		295		
32		14		169	1.5	295		
33		14	9.0	169		295		
34		15	2.0	182		253		
35		15		225		253		
36		16	4.8	225		253		
37		16		225		253		
38		17	1.0	225		253		
39		17		196	6.0	253		
40		18	5.3	241				

Obtenção do Número de Passos em Função de M e S para G Fixo igual a 6.0

Tabela 10

Memória em Setores	Diferença %	Limite Nº Corridas 1 passo	Diferença %	Limite Nº Corridas 2 passos	Diferença %	Limite Nº Corridas 3 passos	Diferença %	Limite Nº Corridas 4 passos
10		8		64	4.8	512		
11		9		81	5.5	648		
12		10	1.0	90	7.4	729		
13		10		30	0.9	125	3.6	729
14		11		30		150		626
15		13		42		252		626
16		13		42		252		500
17	1.0	13		49	1.0	343		501
18	1.0	13		49	2.2	392		
19		8		56	2.4	392		513
20		15		64		449		
21		9		72	2.8	449		
22		9		81		449		
23		10		81		392		
24		10	1.7	100		252	2.2	393
25		11		110		393		
26		11		110		344		
27		12	5.1	110		344		
28		12		121	3.1	344		
29		13		121	2.4	344		
30		13	2.5	156		295		
31		13		156		295		
32		14	1.7	156	3.5	294		
33		14		81	1.8	295		
34		14		90		253		
35		15	2.8	225		253		
36		16	7.0	210		253		
37		16	1.8	196		253		
38		16	1.0	150	0.9	253		
39	3.4	17		121		253		
40		17	1.0	121	0.9	226		

Obtenção do Número de Passos em Função de M e S para G Fixo igual a 7.0

Tabela 11

Número de Corridas	Tempo de Intercalação			
	1 passo	2 passos	3 passos	4 passos
2	4.3			
⋮				
13	12.0	12.3	14.3	17.2
14	15.5	12.3	14.3	17.2
⋮				
42		16.7	17.3	19.3
43		18.8	17.3	19.3
⋮				
65		19.2	18.7	20.0
⋮				
252			24.0	24.7
253			26.1	24.7
⋮				
626			28.8	26.5

Número de Passos para M=15

Tabela 11-a

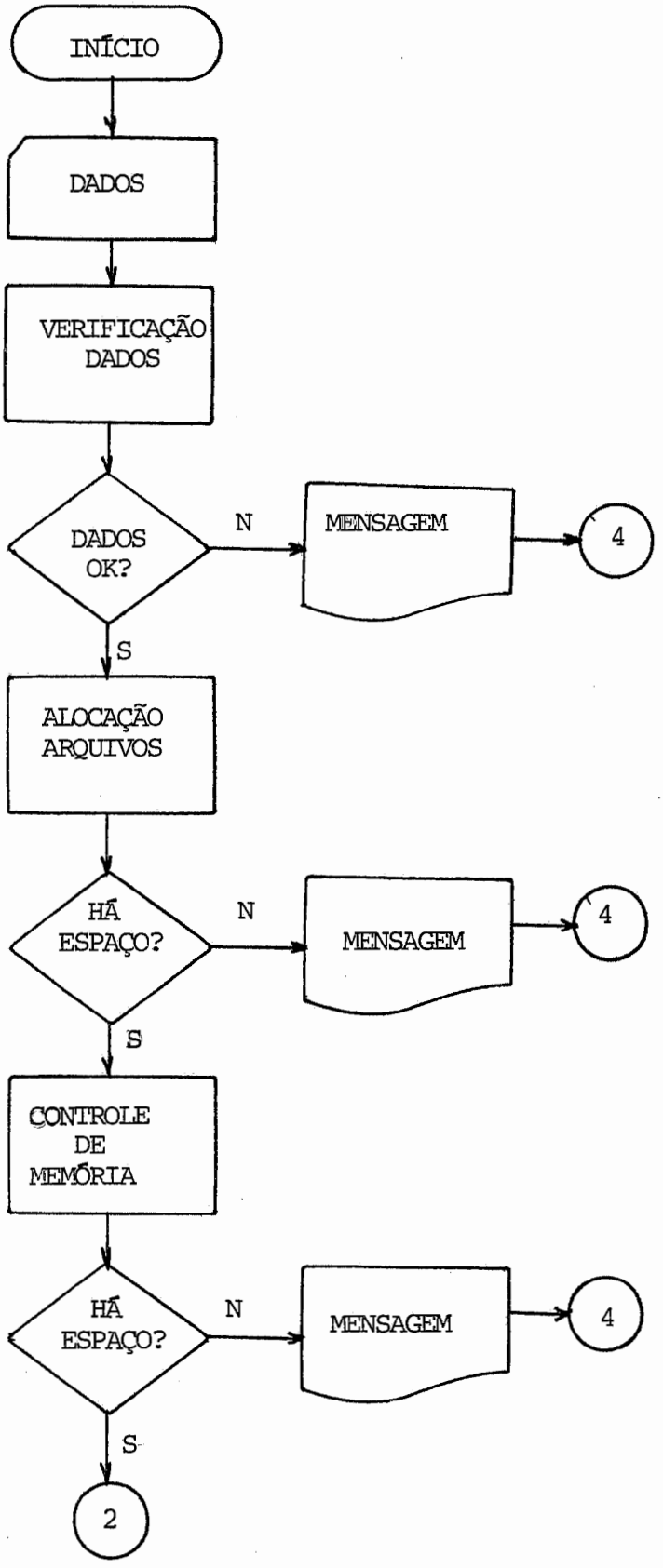
Memória em Setores	G = 5 Diferença %	G = 5 b	G = 6 Diferença %	G = 6 b	G = 7 Diferença %	G = 7 b
4		1		1		1
5		1		1		1
6	2.5	2	10.7	2	9.0	2
7	5.8	2	3.5	2	2.0	2
8	32.0	2	33.0	2	34.0	2
9	45.0	3	44.0	3	42.0	3
10	17.7	3	20.0	3	21.0	3
11	17.7	3	20.0	3	21.0	3
12	32.0	3	36.0	3	38.0	3
13	32.0	3	36.0	3	39.5	3
14	43.0	3	50.0	3	28.0	3
15	50.0	4	48.0	4	28.0	4
16	32.0	4	35.0	4	37.5	4
17	32.0	4	35.0	4	37.5	4
18	21.5	5	25.0	5	37.5	5
19	21.5	5	25.0	5	26.5	5
20	19.3	6	21.0	6	20.0	6
21	19.3	6	21.0	6	20.0	6
22	24.0	6	24.0	6	26.0	6
23	22.0	6	24.0	6	26.0	6
24	24.0	6	29.0	6	35.0	6
25	35.0	7	19.0	7	40.0	7

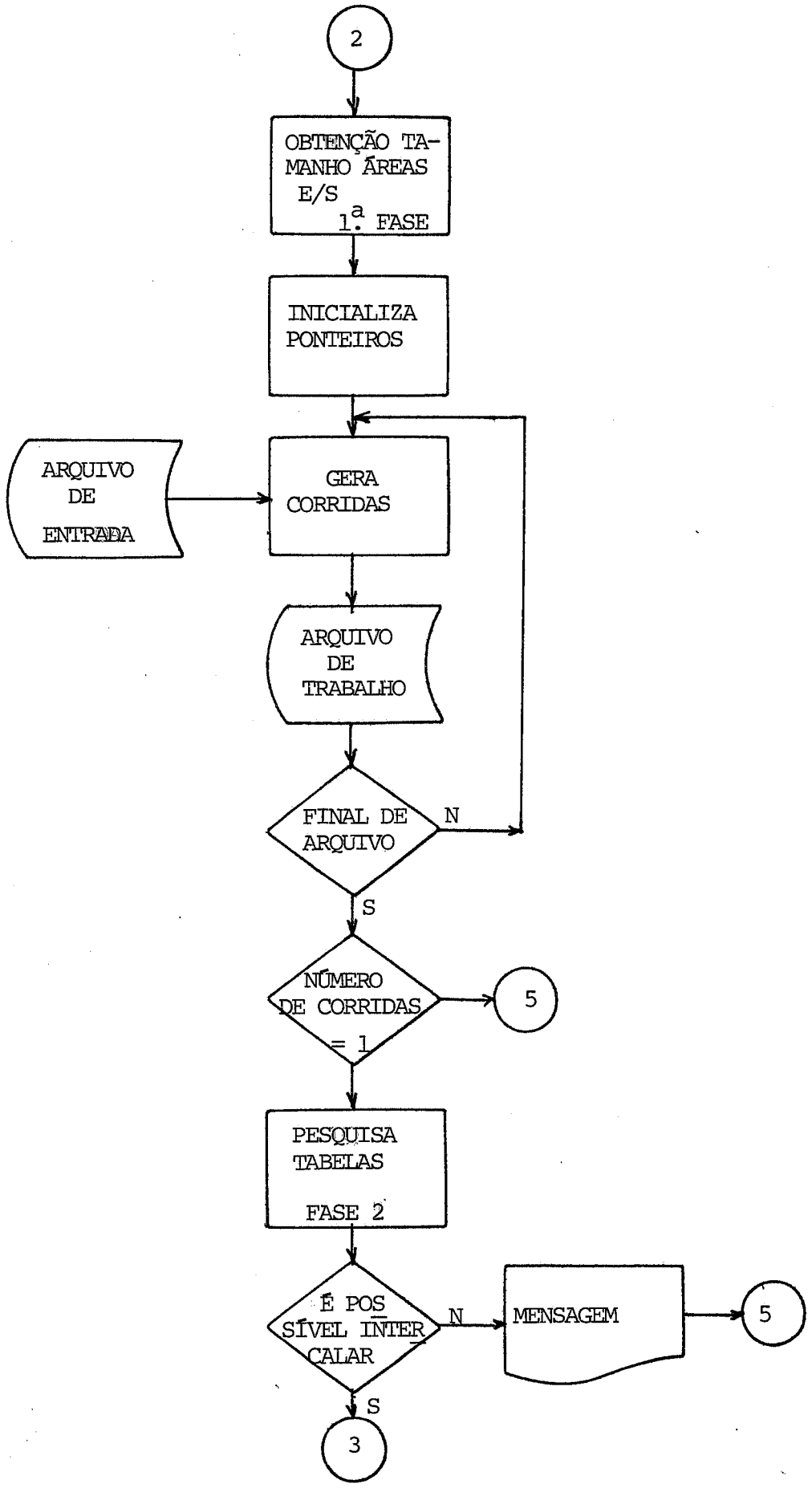
Tabela de b_{min} para G Igual a 5, 6 e 7

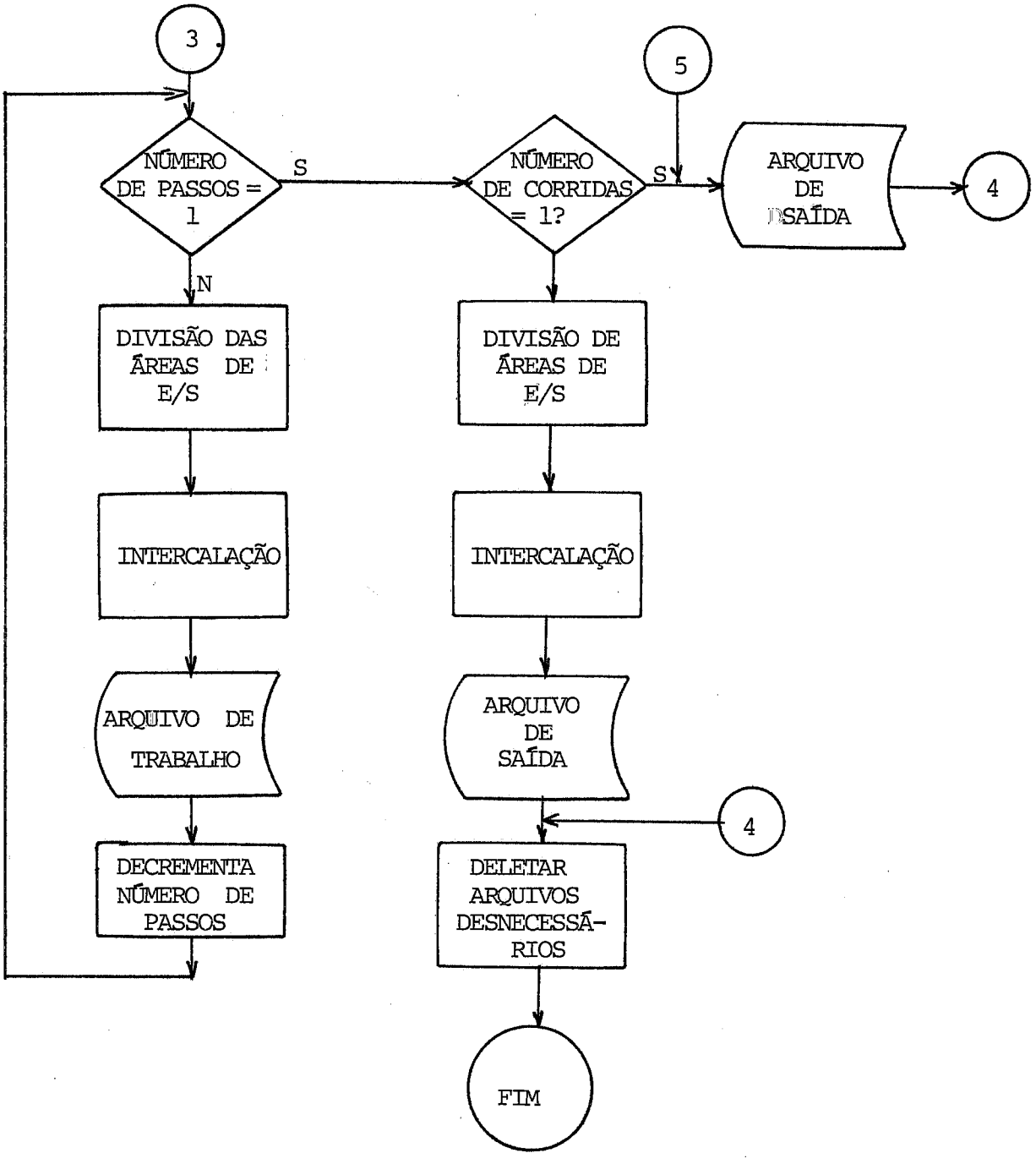
Tabela 12

APÊNDICE B

FLUXOGRAMA


INÍCIO
DADOS
VERIFICAÇÃO DADOS
DADOS OK?
N
MENSAGEM
4
S
ALOCAÇÃO ARQUIVOS
HÁ ESPAÇO?
N
MENSAGEM
4
S
CONTROLE DE MEMÓRIA
HÁ ESPAÇO?
N
MENSAGEM
4
S
2


2
OBTENÇÃO TAMANHO ÁREAS E/S 1ª FASE
INICIALIZA PONTEIROS
ARQUIVO DE ENTRADA
GERA CORRIDAS
ARQUIVO DE TRABALHO
FINAL DE ARQUIVO
N
S
NÚMERO DE CORRIDAS = 1
5
PESQUISA TABELAS FASE 2
É POSSÍVEL INTERCALAR
N
MENSAGEM
5
S
3


3
NÚMERO DE PASSOS = 1
S
N
NÚMERO DE CORRIDAS = 1?
5
S
ARQUIVO DE SAÍDA
4
DIVISÃO DAS ÁREAS DE E/S
INTERCALAÇÃO
ARQUIVO DE TRABALHO
DECREMENTA NÚMERO DE PASSOS
DIVISÃO DE ÁREAS DE E/S
INTERCALAÇÃO
ARQUIVO DE SAÍDA
4
DELETAR ARQUIVOS DESNECESSÁ-RIOS
FIM

APÊNDICE C

```
          ****************************************
          *            PROGRAMA 3                *
          *         CALCULO DO TAMA-             *
          *NHO DAS AREAS DE E/S METODO DA RAIZ*
          ****************************************
BEGIN FILE PR(KIND=PRINTER);
   INTEGER M,D;
      FOR M:=4 STEP 1 UNTIL 40 DO
         FOR D:=2 STEP 1 UNTIL 15 DO
           BEGIN
            REAL F1,F2,DR;
            INTEGER BI1,BI2,BO1,BO2,BID,BOD,BMIN,BMON;
               DR:=D;
               BI1:=-ENTIER(-M/(DR+SQRT(DR)));
               BO1:=M-D*BI1;
               IF BO1 LEQ 0 THEN F1:=1000.
                            ELSE F1:=1.0/BI1 + 1.0/BO1;
               BI2:=BI1-1;
               BO2:=M-D*BI2;
               IF BI2 LEQ 0 OR BO2 LEQ 0
                            THEN F2:=1000.
                            ELSE F2:=1.0/BI2 + 1.0/BO2;
               BID:=M DIV (D+1);
               BOD:=M-D.BID;
               IF F1 < F2 THEN BMIN:=BI1
                          ELSE BMIN:=BI2;
               BMON:=M-D*BMIN;
               WRITE(PR,<7I4>,M,D,BMIN,BMON,BID,BOD,BI1);
           END;
END.
```

```
                    ********************
                    *     PROGRAMA 2     *
                    * CALCULO DO BMIN    *
                    ********************
BEGIN FILE PR(KIND=PRINTER);
INTEGER J,M,NU,NUMAX,NUMIN,S,SMAX,L,B,OLDNU,NRUNS,K;
REAL OLDT,INF,GAMMA,AL,BET;
BOOLEAN FIM;
AL:=48.0;BET:=4.0;SMAX:=164;NUMAX:=5;INF:=100000;
  BEGIN
    REAL ARRAY T 1:NUMAX ,G 1:9 ;
    INTEGER ARRAY D 1:NUMA );
FILL G *  WITH 4.0,4.0,4.0,4.0,5.0,5.0,5.0,6.0,7.0;
    % VARIACAO DA MEMORIA
     FOR M:=4 STEP 1 UNTIL 40 DO
        BEGIN
          K:=0;
          OLDT:=0;OLDNU:=0;
          FIM:=FALSE;
          NRUNS:=10000/M;
          WRITE(PR,<//,"M=",I4,"NRUNS=",I5>,M,NRUNS);
    % VARIACAO DO TAMANHO DO ARQUIVO EM SETORES
       FOR L:=4 STEP L WHILE L LEQ 1024 AND NOT FIM DO
          BEGIN
          REAL ARRAY TB 1:(M/2-1) ;
          REAL TBMIN;INTEGER BMIN;
          K:=K+1;
          GAMMA:=G K ;
          AL:=GAMMA*8.0;
    % VARIACAO DO TAMANHO DE B
        FOR B:=1 STEP 1 UNTIL (M/2-1) DO
           BEGIN
            GENERATE;
           END;
    % ESCOLHENDO B QUE DE O MENOR TEMPO
          TBMIN:=INF;
          FOR B:=1 STEP 1 UNTIL (M/2-1) DO
             BEGIN
              IF TB B  < TBMIN THEN
                                BEGIN
                                 TBMIN:=T B ;
                                 BMIN:=B;
                                END
             END;
          WRITE(PR,/,L,BMIN,TBMIN,GAMMA);
    % CALCULA TEMPO DE INTERCALACAO
    REAL PROCEDURE TAU(M,P);
     INTEGER M,P;
     BEGIN
       INTEGER BI,BO;
```

```
   BI:=IF P < 2 THEN 0 ELSE M DIV (P+1);
   BO:=M-P*BI;
   TAU:= IF BI > 0 THEN 1.5 + GAMMA*(1.0/BI + 1.0/BO)
                   ELSE INF;
 END TAU;
% CALCULA AS ORDENS DE INTERCALACAO PARA O NUMERO
  DE PASSOS IGUAL A J
PROCEDURE GETD(J);
 INTEGER J;
  BEGIN
   INTEGER K,P,Q,PROD;
   P:=ENTIER(S**(1.0/J));
   Q:=P+1;
   PROD:=P**J;
   FOR K:=0 STEP 1 UNTIL J-1 DO
    BEGIN
     IF PROD >= S THEN D J-K :=P;
     ELSE BEGIN
           D J-K :=Q;
           PROD:=PROD*Q DIV P;
          END;
    END;
  END GETD;
% CALCULA O NUMERO DE PASSOS E A ORDEM DE INTERCALACAO
  QUE MINIMIZE O TEMPO
PROCEDURE GENERATE;
 BEGIN
  REAL SUM;
  S:=-ENTIER((-L)/(2*(M-2*B)));
  T 1 :=(IF S = 1 THEN 0 ELSE TAU(M,S))+2.0+AL/(B*BET);
  FOR NU:=2 STEP 1 UNTIL NUMAX DO
    BEGIN
     SUM:=2.0+AL/(B*BET);
     GETD(NU);
     FOR J:=1 STEP 1 UNTIL NU DO SUM:=SUM + TAU(M,D J );
     T NU :=SUM;
    END;
  SUM:=6*INF;
  FOR NU:=1 STEP 1 UNTIL NUMAX DO
    IF T NU  < SUM THEN
                      BEGIN
                       SUM:=T NU ;
                       NUMIN:=NU;
                       TB B :=SUM;
                      END;
    WRITE(PR,<2I5,6F5.1,I10>,S,NUMIN,SUM,T * ,B);
  IF NUMIN=5 THEN FIM:=TRUE;
 END GENERATE;
      END LACO L,
    END LACO M;
```

```
  END LACO MAXIMO;
END.
```

```
                ****************************
                *        PROGRAMA 1        *
                *     CALCULO DOS PARA-    *
                *METROS PARA A INTERCALACAO*
                ****************************
BEGIN FILE PR (KIND=PRINTER);
INTEGER J,M,NU,NUMAX,NUMIN,S,SMAX,OLDNU,NRUNS,NSET,IB;
REAL INF,GAMMA,OLDT;
BOOLEAN FIM;
INTEGER ARRAY BMIN 1:37 ;
FILL BMIN *  WITH 1,1,2,2,2,3,3,3,3,3,3,4,4,4,4,5,5,6,6,6,6
                  7,7,7,7,8,8,8,8,8,8,9,9,10,10,11,11,12;
SMAX:=164;NUMAX:=5;INF:=100000;
 BEGIN
   REAL ARRAY T 1:NUMAX ;
   INTEGER ARRAY D 1:NUMAX ;
   IB:=0;
   % VARIACAO DA MEMORIA
     FOR M:=4 STEP 1 UNTIL 40 DO
        BEGIN
          OLDT:=0;OLDNU:=0;
          FIM:=FALSE;
          NRUNS:=10000/M;
          IB:=IB+1;
          WRITE(PR,</,"M=",I4,"NRUNS=",I5>,M,NRUNS);
   % VARIACAO DO NUMERO DE CORRIDAS
       FOR S:=2 STEP 1 WHILE S<NRUNS AND NOT FIM DO
            BEGIN
              NSET:=2*(M=2*BMIN IB )*S;
              GAMMA:=(30.+0.8*(NSET/24))/8.0;
              GENERATE;
            END LACO S;
   % CALCULA TEMPO DE INTERCALACAO
   REAL PROCEDURE TAU(M,P);INTEGER M,P;
    BEGIN
     INTEGER BI,BO;
       BI:=IF P< 2 THEN 0 ELSE M DIV (P+1);
       BO:=M=P*BI;
       TAU:=IF BI > 0 THEN 1.5+GAMMA*(1.0/BI + 1.0/BO)
            ELSE INF;
    END TAU;
   % CALCULA AS ORDENS DE INTERCALACAO PARA O NUMERO
     DE PASSOS IGUAL A J
   PROCEDURE GETD(J);
    INTEGER J;
     BEGIN
      INTEGER K,P,Q,PROD;
      P:=ENTIER(S**(1.0/J));
      Q:=P+1;
      PROD:=P**J;
```

```
      FOR K:=0 STEP 1 UNTIL J-1 DO
       BEGIN
         IF PROD >= S THEN D J-K :=P;
         ELSE BEGIN
                D J-K :=Q;
                PROD:=PROD*Q DIV P;
              END;
       END;
     END GETD;
   % CALCULA O NUMERO DE PASSOS,A ORDEM DE INTERCALACAO
     QUE MINIMIZE O TEMPO
   PROCEDURE GENERATE;
    BEGIN
     REAL SUM;
     T(1):=TAU M,S ;
     FOR NU:=2 STEP 1 UNTIL NUMAX DO
       BEGIN
        SUM:=0;GETD(NU);
        FOR J:=1 STEP 1 UNTIL NU DO
          SUM:=SUM + TAU(M,D J) ;
        T NU :=SUM;
       END;
     SUM:=INF;
     FOR NU:=1 STEP 1 UNTIL NUMAX DO
       IF T NU  < SUM THEN
                        BEGIN
                         SUM:=T N );
                         NUMIN:=NU;
                        END;
       IF OLDT NEQ SUM OR OLDNU NEQ NUMIN THEN
         BEGIN
          WRITE(PR,<2I5,7F5.1>,S,NUMIN,SUM,T * ,GAMMA);
          OLDT:=SUM;
          OLDNU:=NUMIN;
         END;
       IF NUMIN=5 THEN FIM:=TRUE;
    END GENERATE;
        END LACO M;
 END LACO MAXIMO;
END.
```

```
$JOB ASSMBL  1,5
$RUN PIP
#ORD.MAC<BI:/FA
        .MCALL .WRITE,.BIN2O,.OPEN,.CLOSE,.RLSE,.READ,.D2BIN
        .MCALL .INIT,.TRAN,.WAIT,.ALLOC,.LOOK,.EXIT,.MONF,.GTPLA
        .MCALL .DELETE
SORT:   .INIT #LNK1
        .INIT #LNK2
        .INIT #LNK3
        .OPEN #LNK2,#FIL2
*****************************************************************
*******       OBTENCAO DOS PARAMETROS DO USUARIO            *******
*******       VERIFICACAO DE SUAS VALIDADES                 *******
*******       OBTENCAO DE MEMORIA INTERNA LIVRE             *******
*******       VERIFICACAO DE AREA EM DISCO                  *******
*****************************************************************
LE:     .READ #LNK3,#BUF        ;LE DADOS DO USUARIO
        .WAIT #LNK3
        MOV   #BUF+6,R0
        MOV   #NOM,R2           ;NOME DO ARQUIVO DE ENTRADA
1$:     MOVB  (R0)+,(R2)+
        CMPB  (R0),#54
        BNE   1$
        MOV   #3,R4
        MOV   #FIL1,R3
        MON   #NOM,R1           ;TRANSFORMA NOME DO ARQUIVO DE EN-
        JSR   PC,TRADIX         ;TRADA EM RAD50
        .LOOK #LNK1,#FIL1,1
        MOV   (SP)+,EAFOR       ;ENDERECO ARQUIVO DE ENTRADA
        MOV   (SP)+NBLOCO       ;NUMERO DE SETORES DO ARQUIVO
        CLR   (SP)+
        MOV   EAFOR,FARQ        ;ENDERECO DO ULTIMO BLOCO DO AR-
        ADD   NBLOCO,FARQ       ;QUIVO DE ENTRADA
        DEC   FARQ
        MOV   #NOM,R1
        MOV   #10.,R2
20$:    CLRB  (R1)+
        DEC   R2
        BNE   20$
        TSTB  (R0)+
        MOV   R0,R1
        MOV   #BUF+6,R2
5$:     CMPB  (R1)+,(R2)+       ;TESTA SE ARQUIVO DE ENTRADA E O
        BNE   4$                ;MESMO PARA SAIDA
        CMPB(R1),#54
        BNE   5$
        CMPB  (R2),#54
        BEQ   6$
4$:     INC   FLAG              ;SE FOREM DISTINTOS FLAG=1
6$:     MOV   #NOM,R2
```

```
2$:     MOVB   (R0)+,(R2)+
        CMPB   (R0),#54
        BNE    2$
        TSTB   (R0)+
        TST    FLAG
        BNE    7$
        MOV    #3,R4
        MOV    #FIL1,R3
        MOV    #NOM,R1          ;TRANSFORMA NOME DO ARQUIVO DE
        JSR    PC,TRADIX        ;SAIDA EM RAD50
7$:     JSR    PC,BIN
        MOV    R3,CRF           ;NUMERO DE REGISTROS
        JSR    PC,BIN
        MOV    R3,TMREG         ;TAMANHO DOS REGISTROS EM PALAVRAS
        JSR    PC,BIN
        MOV    R3,NSETOR
        JSR    PC,BIN
        MOV    R3,NCHAV         ;NUMERO DE CHAVES
        MOV    NCHAV,R5
        MOV    #DCHAV,R4        ;OBTENDO DADOS SOBRE AS CHAVES
3$:     JSR    PC,BIN
        MOV    R3,(R4)+         ;INICIO DAS CHAVES
        MOVB   (R0)+,(R4)+      ;TAMANHO DAS CHAVES
        INC    R4
        TSTB   (R0)+
        JSR    PC,BIN
        MOV    R3,(R4)+         ;TIPOS DAS CHAVES
        MOVB   (R0)+,(R4)+      ;ORDEM DAS CHAVES
        TSTB   (R0)+
        INC    R4
        DEC    R5
        BNE    3$
        JSR    PC,BIN           ;OBTENDO MARCA DE FINAL DE ARQUIVO
        MOV    R3,IMARCA
        DEC    IMARCA
        CMPB   (R0)+,#54
        BNE    8$
        MOVB   (R0),MARCA       ;MARCA #0
        BR     FLE
8$:     MOV    #0,MARCA         ;MARCA E 0
FLE:    TST    FLAG             ;VERIFICA SE ALOCOU ARQUIVO DE
        BNE    1$               ;SAIDA
        MOV    EAFOR,EASAI
        MOV    NBLOCO,R4
        BR     5$
1$:     ASL    NBLOCO           ;SE NAO ALOCOU ESTA SENDO ALOCADO
        ASL    NBLOCO
        MOV    NBLOCO,-(SP)
        MOV    #FIL1,-(SP)
        MOV    #LNK1,-(SP)      ;ALOCANDO ARQOUT
```

```
        EMT   15
        INC   (SP)              ;VERIFICA SE HA ESPACO EM DISCO
        BEQ    2$
        MOV   #MENS2,R0
        JSR   PC,GRA
2$:     .LOOK #LNK1,#FIL1,1
        MOV   (SP)+,EASAI       ;ENDERECO ARQUIVO DE SAIDA
        MOV   (SP)+,R4          ;OBTENDO NUMERO DE SEGMENTOS DE
        CLR   (SP)+             ;64 PALAVRAS PARA ALOCAR ARQUIVO
5$:     MOV   R4,EASOR2         ;DE TRABALHO
        ASL   R4
        ASL   R4
        ASL   R4
        MOV   #FIL1,R1
        MOV   #3,R2
        MOV   #NOMARQ,R3        ;NOME DO ARQUIVO DE TRABALHO
4$:     MOV   (R3)+,(R1)+
        DEC   R2
        BNE   4$
        MOV   R4,-(SP)
        MOV   #FIL1,-(SP)       ;ALOCANDO ARQUIVO DE TRABALHO
        MOV   #LNK1,-(SP)
        EMT   15
        INC   (SP)              ;VERIFICA SE HA ESPACO EM DISCO
        BEQ    3$
        MOV   #MENS2,R0
        JSR   PC,GRA
3$:     .LOOK #LNK1,#FIL1,1     ;DIVIDINDO O ARQUIVO DE TRABALHO
        MOV   (SP)+,EASOR1      ;ENDERECO ARQUIVO DE TRABALHO 1
        CLR   (SP)+
        CLR   (SP)+
FAS2:   ADD   EASOR1,EASOR2     ;ENDERECO ARQUIVO DE TRABALHO 2
        MOV   TMREG,TRB
        ASL   TRB               ;TAMANHO DO REGISTRO EM BYTES
        MOV   TRB,TMRP
        ADD   #8.,TMRP          ;TAMANHO DO REGISTRO MAIS PONTEIRO
        MOV   #512.,R2          ;CABEM NO SETOR
        MOV   TRB,R3            ;NUMEX=0 INDICA QUE CABE UM NUME-
        JSR   PC,DIVID          ;RO EXATO DE REGISTROS POR SETOR
        MOV   R4,NUMEX
        MOV   #1536.,TMTAB      ;ESPACO PARA DADOS SOBRE CORRIDAS
        MOV   #FIL1,R1          ;ALOCANDO TABELA DADOS DAS CORRI-
        MOV   #NOTAB,R3         ;DAS NO DISCO
        MOV   #3,R2
13$:    MOV   (R3)+,(R1)+
        DEC   R2
        BNE   13$
        MOV   #12.,-(SP)
        MOV   #FIL1,-(SP)
        MOV   #LNK1,-(SP)
```

```
        EMT   15
        .LOOK #LNK1,#FIL1,1     ;ENDERECO DA TABELA NO DISCO
        MOV   (SP)+,TEND
        CLR   (SP)+
        CLR   (SP)+
        ADD   TMTAB,R1          ;OBTENDO MEMORIA DISPONIVEL EM
        ADD   TRB,R1            ;SETORES
        CMP   R1,#32767.
        BLOS  10$
        MOV   R1,-(SP)
        MOV   #512.,-(SP)
        JSR   PC,NEWDIV
        BR    11$
10$:    MOV   R1,R2
        MOV   #512.,R3
        JSR   PC,DIVID
11$:    MOV   #TBM,R2           ;PESQUISA DO BMIN
        SUB   #10.,QUOC
        ADD   QUOC,R2
        MOVB  (R2),NSETOR       ;NUMERO DE SETOR = BMIN
        MOV   NSETOR,R2
        MOV   #512.,R4
        JSR   PC,MULT
        MOV   R3,R0
        MOV   R0,TRNB2+4
        ASR   TRNB2+4           ;ESTAVA EM BYTES
        MOV   R0,TMBS           ;TAMANHO DA AREA DE ENTRADA SEM 1
        TST   NUMEX             ;REGISTRO
        BEQ   3$
        MOV   TRB,R1            ;OBTENDO QUANTIDADE DO ULTIMO RE-
        MOV   R0,R2             ;GISTRO QUE ULTRAPASSOU O SETOR
        MOV   R1,R3
        JSR   PC,DIVID          ;DIVIDINDO PARA SABER QUANTO PASSOU
        SUB   R4,R1
        MOV   R1,QPASI          ;QUANTIDADE QUE ULTRAPASSOU
        MOV   R0,TMBUF
        ADD   TRB,TMBUF         ;AREA DE ENTRADA MAIS 1 REGISTRO
        ADD   R1,R0
        BR    4$
3$:     MOV   R0,TMBUF
4$:     ASR   R0
        MOV   R0,TRNB1+4        ;QUANTIDADE PARA LER
        .GTPLA
        MOV   (SP)+,R1          ;PONTO DE CARGA DO PROGRAMA
        .MONF
        MOV   (SP)+,R2          ;ATE ONDE MONITOR OCUPA
        ADD   #512.,R2          ;ESPACO PARA AREA TRANSIENTE
        SUB   #200.,R1          ;100 PALAVRAS PARA A PILHA
        CMP   R2,R1
        BLO    7$
```

```
7$:     MOV   R2,INAREA         ;INICIO DA AREA LIVRE
        MOV   R1,FIAREA         ;FINAL DA AREA LIVRE
        SUB   R2,R1             ;TOTAL EM BYTES MEMORIA DISPONIVEL
        MOV   INAREA,R1
        MOV   INAREA,INTAB      ;INICIO DA TABELA NA MEMORIA
        ADD   TMTAB,R1
        MOV   R1,BUFIN          ;INICIO AREA DE ENTRADA
        ADD   TMBUF,R1
        MOV   R1,BUFOU          ;INICIO AREA DE SAIDA
        ADD   TMBUF,R1
        MOV   R1,INLAST         ;INICIO LASTKEY
        ADD   TRB,R1
        MOV   BUFIN,FBUFIN      ;FINAL DA AREA DE ENTRADA
        ADD   TMBS,FBUFIN
        ADD   QPASI,FBUFIN
        MOV   BUFOU,FBUFOU      ;FINAL DA AREA DE SAIDA
        ADD   TMBS,FBUFOU
        MOV   R1,IALIV          ;INICIO AREA LIVRE PRIMEIRA PARTE
        MOV   FIAREA,R2
        SUB   R1,R2             ;NUMERO DE BYTES PARA REGISTROS
        CMP   R2,#32767.
        BLOS  20$
        MOV   R2,-(SP)
        MOV   TRB,-(SP)
        ADD   #8.,(SP)
        JSR   PC,NEWDIV
        BR    21$
20$:    MOV   TRB,R3            ;OBTENDO NUMERO DE REGISTROS COM
        ADD   #8.,R3            ;PONTEIROS QUE CABEM NA AREA LIVRE
        JSR   PC,DIVID
21$:    MOV   QUOC,P
        JMP   RPLE
*****************************************************************
*******          ROTINA COMPACTA NUMERO                  ******
*******       TRANSFORMA CINCO CARACTERES ASCII          ******
*******       EM UM NUMERO DECIMAL                       ******
*****************************************************************
BIN:    MOV   R0,R2
1$:     CMPB  (R0)+,#54
        BNE   1$
        MOV   R0,R3
        DEC   R3
        SUB   R2,R3
        MOV   #ATRAN+5,R1
        MOV   R0,R2
        DEC   R2
2$:     MOVB  -(R2),-(R1)
        DEC   R3
        BNE   2$
        .D2BIN #ATRAN
```

```
        BCS     4$
        BCC     5$
4$:     MOV     #MENS1,R0
        JSR     PC,GRA
5$:     MOV     (SP)+,R3
        CLR     (SP)+
        MOV     #5,R2
        MOV     #ATRAN,R1
3$:     MOVB    #60,(R1)+
        DEC     R2
        BNE     3$
FBIN:   RTS     PC
*****************************************************************
*******           ROTINA TRANSFORMA EM RAD50                  ******
*******      TRANSFORMA CARACTERES ASCII EM RAD50             ******
*****************************************************************
TRADIX: MOV     R1,-(SP)
1$:     CLR     -(SP)
        EMT     42
        MOV     (SP)+,(R3)+
        DEC     R4
        TST     R4
        BNE     1$
        TST     (SP)+
FTRA:   RTS     PC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCC     FASE DE CRIACAO DAS CORRIDAS     CCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
*****************************************************************
*******               INICIALIZA LASTKEY                    ******
*******      VERIFICA OS TIPOS DE CHAVES E ORDENS           ******
*******      PARA INICIALIZAR LASTKEY                       ******
*****************************************************************
RPLE:   MOV     #TESTA+4,R2         ;MOVE O ENDERECO DA MEMORIA ONDE
        CLR     FLAG                ;SERA MONTADA A ROTINA DE TESTAR
        JSR     PC,MTEST            ;CHAVES
        MOV     #MLAS,R2            ;MOVE O ENDERECO DA MEMORIA ONDE
        INC     FLAG                ;SERA MONTADA A ROTINA DE MOVIMEN-
        JSR     PC,MTEST            ;TO PARA LASTKEY
        MOV     NCHAV,R5            ;VERIFICANDO OS TIPOS E ORDENS DAS
        MOV     #DCHAV,R0           ;CHAVES PARA PREENCHIMENTO DA LAS-
6$:     MOV     (R0)+,R2            ;TKEY
        DEC     R2
        ADD     INLAST,R2
        CMP     (R0)+,#116          ;COMPARA O TIPO DA CHAVE
        BNE     1$
        TST     (R0)+               ;ADIANTA PONTEIRO
        CMP     (R0)+,#101          ;COMPARA A ORDEM DA CHAVE
        BNE     2$
        MOV     #32767.,(R2)        ;MOVE O MAIOR INTEIRO (NUMERICA)
```

```
        BR      3$
2$:     MOV     #-32768.,(R2)       ;MOVE O MENOR INTEIRO (NUMERICA)
        BR      3$
1$:     MOV     (R0)+,R4            ;NUMERO DE BYTES
        CMP     (R0)+,#101
        BNE     4$
        MOV     #140,5$+2           ;MAIOR CONFIGURACAO DE BITS(ALFA-
        BR      5$                  ;NUMERICA) ORDEM ASCENDENTE
4$:     MOV     #41,5$+2            ;MENOR CONFIGURACAO DE BITS(ALFA-
5$:     MOVB    #140,(R2)+          ;NUMERICA) ORDEM DESCENDENTE
        DEC     R4
        BNE     5$
3$:     DEC     R5
        BEQ     FORA
        BR      6$
*****************************************************************
*******               INICIALIZACAO DOS PONTEIROS               ******
*****************************************************************
*****************************************************************
*******               ZERANDO TODAS AS POSICOES DE              ******
*******               CHAVES                                    ******
*****************************************************************
FORA:   MOV     TRB,R0              ;R0 APONTA PARA A LOCACAO DE RN
        MOV     R0,R1
        ADD     #2,R1               ;R1 APONTA PARA A LOCACAO DE LOSER
        MOV     R1,R2
        ADD     #2,R2               ;R2 APONTA PARA A LOCACAO DE FE
        MOV     R2,R3
        ADD     #2,R3               ;R3 APONTA PARA A LOCACAO DE FI
        MOV     R0,RET+2            ;MODIFICANDO AS INSTRUCOES INDEXA-
        MOV     R2,MOV1+2           ;DAS,POIS O INDICE DEPENDE DO TA-
        MOV     R3,MOV2+2           ;MANHO DO REGISTRO
        MOV     R1,RET+6
        MOV     R2,NOVT+6
        MOV     R1,VT1+2
        MOV     R1,VT3-2
        MOV     R1,VT3-10
        MOV     R0,VT1+22
        MOV     R0,VT5+2
        MOV     R0,VT5+10
        MOV     R3,VT4+2
        MOV     R1,R0
        MOV     IALIV,R1
IPON:   MOV     R1,Q                ;INICIALIZANDO RN,LOSER,FE,FI
        CLR     R5
        MOV     R1,R0               ;SALVA END 1 REG
RET:    MOV     R1,24(R1)           ;MOVE ENDERECO REGISTRO PARA RN
        CLR     26(R1)              ;ZERA LOSER
        MOV     P,R4
        ADD     R5,R4
```

```
        ASR     R4
        MOV     TMRP,R2
        JSR     PC,MULT
        ADD     R0,R3
MOV1:   MOV     R3,30(R1)       ;MOVE  (P+J)/2  PARA FE
        MOV     R5,R4
        ASR     R4
        MOV     TMRP,R2
        JSR     PC,MULT
        ADD     R0,R3
MOV2:   MOV     R3,32(R1)       ;MOVE  J/2  PARA FI
;ROTINA ZERA CHAVES
ZER:    MOV     R0,-(SP)        ;SALVA R0
        MOV     NCHAV,R3
        MOV     #DCHAV,R0       ;END DADOS A CHAVES
2$:     MOV     (R0)+,R2        ;POSICAO DENTRO DO REG
        DEC     R2
        ADD     R1,R2
        TST     (R0)+           ;PULA TIPO
        MOV     (R0)+,R4
1$:     CLRB    (R2)+
        DEC     R4
        BNE     1$
        DEC     R3
        BEQ     FZER
        TST     (R0)+           ;PULA ORDEM
        BR      2$
FZER:   MOV     (SP)+,R0
        ADD     TMRP,R1
        INC     R5
        CMP     R5,P
        BLO     RET
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
RRRRRRRRRRRR   ALGORITMO DE REPLECEMENT SELECTION   RRRRRRRRRRR
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
        CLR     RMAX            ;NUMERO DE CORRIDAS PRODUZIDAS
        CLR     RQ              ;INDICARA FINAL DA CRIACAO
        CLR     RC              ;NUMERO DA CORRIDA SENDO FORMADA
        CLR     NCR             ;NUMERO DE ELEMENTOS DA CORRIDA
ALG:    ADD     TMRP,R0         ;ENDERECO NO X(1)
        MOV     R0,LOCX1        ;NO FINAL TESTA COM T
        MOV     INTAB,R0
        MOV     BUFOU,R5        ;PONTEIRO AREA DE SAIDA
        MOV     EASOR1,TRNB2    ;END ARQ TRABALHO 1
        MOV     R5,TRNB2+2      ;ENDERECO AREA DE SAIDA
        MOV     EAFOR,TRNB1     ;END ARQ FORTRAN
        MOV     BUFIN,TRNB1+2   ;ENDERECO AREA DE ENTRADA
        .TRAN   #LNK1,#TRNB1    ;PRIMEIRA LEITURA
        MOV     BUFIN,R1        ;PONTEIRO AREA DE ENTRADA
        TST     NUMEX
```

```
        BNE     1$
        .WAIT   #LNK1
        BR      2$
1$:     MOV     TMBS,R2
        SUB     QPASI,R2
        MOV     TRB,R3
        JSR     PC,DIVID
        MOV     QPASI,QPASA         ;SALVA QUANT ANTERIOR
        MOV     TRB,QPASI
        SUB     R4,QPASI
        MOV     TMBS,R3
        ADD     QPASI,R3            ;NUMERO DE PAL PROXIMA LEITURA
        ASR     R3
        .WAIT   #LNK1
        MOV     R3,TRNB1+4
2$:     ADD     NSETCR,TRNB1        ;AJEITA SETOR
VOL1:   CMP     RQ,RC
        BEQ     1$
        CMP     RQ,RMAX
        BLOS    2$
        MOV     NCR,(RQ)+           ;MOVE QUANTIDADE DE REG ULTIMA CORRIDA
        CMP     R5,BUFOU            ;BUF VASIO?
        BEQ     3$
        SUB     BUFOU,R5            ;PALAVRAS PARA TRANSFERIR
        ASR     R5                  ;DIVIDE POR 2, ESTAVA EM BYTES
        MOV     R5,TRNB2+4
8$:     .TRAN   #LNK1,#TRNB2        ;GRAVACAO
        .WAIT   #LNK1
3$:     MOV     INTAB,TRNB2+2       ;GRAVAR TABELA
        MOV     TEND,TRNB2          ;END TABELA NO DISCO
        SUB     INTAB,R0
        ASR     R0                  ;ESTAVA EM BYTES
        MOV     R0,TRNB2+4          ;NUMERO DE PALAVRAS
        .TRAN   #LNK1,#TRNB2
        .WAIT   #LNK1
        .CVTDT  #1,#BUF+6
        .WRITE  #LNK2,#BUF
        .WAIT   #LNK2
        JMP     PTA                 ;PESQ TABELAS
2$:     TST     RC
        BNE     4$
        MOV     EASOR1,(R0)+        ;END PRIMEIRA CORRIDA
        CLR     (R0)+               ;POSICAO REG DENTRO DO SETOR
        BR      5$
4$:     MOV     NCR,(R0)+           ;NUMERO DE REGISTROS CORRIDA ANTERIOR
        MOV     R5,R4               ;CALCULANDO ENDERECO CORRIDA NO
        SUB     BUFOU,R4            ;DISCO
        MOV     R4,R2
        MOV     #512.,R3
        JSR     PC,DIVID
```

```
        MOV   TRNB2,(R0)
        ADD   QUOC,(R0)+        ;ENDERECO CORRIDA NO DISCO(SETOR)
        MOV   R4,(R0)+          ;POSICAO DENTRO DO SETOR
        CLR   NCR               ;ZERA CONTADOR DE ELEMENTOS
5$:     MOV   RQ,RC
1$:     TST   RQ                ;SE RQ#0 MOVE REGISTRO PARA SAIDA
        BNE   MOR1
        JMP   PULO
MOR1:   MOV   Q,R3
        MOV   TMREG,R2
1$:     MOV   (R3)+,(R5)+       ;MOVE REGISTRO PARA AREA DE SAIDA
        DEC   R2
        BNE   1$
        INC   NCR               ;INCREMENTA CONTADOR REG CORRIDAS
        CMP   R5,EBUFOU         ;ENCHEU AREA DE SAIDA?
        BLO   2$
5$:     .TRAN #LNK1,#TRNB2      ;GRAVACAO
        TST   NUMEX
        BNE   3$
        MOV   BUFOU,R5
        .WAIT #LNK1
        BR    6$
3$:     MOV   R5,R4
        SUB   FBUFOU,R4
        ASR   R4                ;QUANTIDADE QUE PASSOU EM PALAVRAS
        MOV   FBUFOU,R3
        MOV   BUFOU,R5
        .WAIT #LNK1
4$:     TST   R4
        BEQ   6$
        MOV   (R3)+,(R5)+       ;MOVE PARA INICIO AREA DE SAIDA
        DEC   R4
        BR    4$
6$:     ADD   NSETOR,TRNB2      ;INICIO GRAVACAO PROXIMA CORRIDA
2$:     MOV   Q,R2
        MOV   INLAST,R3         ;INICIO LASTKEY
; **** AREA ONDE MONTARA ROTINA PARA MOVER PARA LASTKEY ****
MLAS:   .WORD 432,431,430,427,426,425,424,423,422
        .WORD 421,420,417,416,415,414,413,412,411
        .WORD 410,407,406,405,404,403,402,401,400
PULO:   MOV   R1,-(SP)          ;SALVA R1
        ADD   IMARCA,R1         ;VERIFICANDO FINAL DE ARQUIVO
        CMP   (R1),MARCA        ;TESTA COM A MARCA DADA USUARIO
        BEQ   6$
        MOV   (SP)+,R1
        BR    1$
6$:     MOV   RMAX,RQ
        INC   RQ
        BR    NOVT
1$:     MOV   Q,R2
```

```
        MOV   TMREG,R4
2$:     MOV   (R1)+,(R2)+       ;MOVENDO REGISTRO PARA AREA DE
        DEC   R4                ;TRABALHO
        BNE   2$
        CMP   R1,FBUFIN         ;AREA DE ENTRADA VAZIA?
        BLO   3$
        CMP   TRNB1,FARQ        ;TESTA FIM DE ARQUIVO PELO NUMERO
        BHI   6$                ;DE SETORES
        .TRAN #LNK1,#TRNB1      ;LEITURA
        MOV   BUFIN,R1          ;ATUALIZA APONTADOR
        TST   NUMEX
        BNE   4$
        .WAIT #LNK1
        BR    5$
4$:     ADD   QPASA,R1
        MOV   TRNB1+4,FBUFIN
        ASL   FBUFIN
        ADD   BUFIN,FBUFIN      ;ATUALIZA FINAL AREA DE ENTRADA
        MOV   TMBS,R2           ;OBTENDO QUANTIDADE DE PALAVRAS PARA
        SUB   QPASI,R2          ;SEREM LIDAS
        MOV   TRB,R3
        JSR   PC,DIVID
        MOV   QPASI,QPASA       ;SALVA QUANT ANTERIOR
        MOV   TRB,QPASI
        SUB   R4,QPASI
        MOV   TMBS,R3
        ADD   QPASI,R3
        ASR   R3
        .WAIT #LNK1
        MOV   R3,TRNB1+4        ;ATUALIZA NUMERO PAL PROXIMA LEITURA
5$:     ADD   NSETOR,TRNB1      ;AJUSTA SETOR
3$:     MOV   Q,R2
        MOV   INLAST,R3
        JSR   PC,TESTA          ;COMPARA CHAVES REGISTRO APONTADO
        TST   VERD              ;POR RZ COM LASTKEY
        BEQ   NOVT
L1:     INC   RQ
        CMP   RQ,RMAX
        BLOS  NOVT
        MOV   RQ,RMAX
NOVT:   MOV   Q,R3
        MOV   104(R3),R4
VT1:    CMP   102(R4),RQ        ;AJUSTA NOVO PERDEDOR
        BLO   VT5
        BNE   VT3
        MOV   R4,-(SP)
        MOV   Q,R3
        MOV   100(R4),R2
        JSR   PC,TESTA
        TST   VERD
```

```
        BNE   VT2
VT6:    MOV   (SP)+,R4
        BR VT3
VT2:    MOV   (SP)+,R4
VT5:    MOV   100(R4),-(SP)
        MOV   Q,100(R4)
        MOV   (SP)+,Q
        MOV   RQ,-(SP)
        MOV   102(R4),RQ
        MOV   (SP)+,102(R4)
VT3:    CMP   R4,LOCX1          ;SOBE A ARVORE
        BNE   VT4
        JMP   VOL1
VT4:    MOV   106(R4),R4
        BR    VT1
; **** ROTINA MONTA TESTE E MOVIMENTO DE CHAVES ****
MTEST:  CLR   CONT1
        MOV   NCHAV,R1          ;NUMERO DE CHAVES
        MOV   #DCHAV,R0         ;ENDERECO DADOS SOBRE CHAVES
14$:    MOV   (R0)+,ICHA        ;INICIO CHAVE
        MOV   ICHA,R3
        CMP   R1,NCHAV
        BNE   1$
        DEC   R3
        BR    2$
1$:     SUB   R4,R3
2$:     MOV   #062702,(R2)+
        MOV   R3,(R2)+          ;DISTANCI A SOMAR
        MOV   #062703,(R2)+     ;INSTRUCAO ADD
        MOV   R3,(R2)+
        CMP   (R0)+,#116        ;CHAVE NUMERICA?
        BEQ   3$
        MOV   (R0)+,TAMACH      ;TAMANHO CHAVE
        TST   FLAG              ;TESTA SE E PARA TESTE OU MOVE
        BNE   7$
        MOV   #122223,R4        ;INSTRUCAO COMPARE
        BR    10$
7$:     MOV   #112223,R4        ;INSTRUCAO MOVE
10$:    BIT   #1,ICHA
        BNE   4$
6$:     MOV   TAMACH,R5
        BR   5$
4$:     BIT   #1,TAMACH
        BNE   6$
        MOV   TAMACH,R5
        ASR   R5
        BIC   #100000,R4
5$:     MOV   #012704,(R2)+
        MOV   R5,(R2)+
        MOV   R4,(R2)+
```

```
        ADD    #9.,CONT1
        TST    FLAG
        BEQ    8$
        MOVB   #-2,IBNE
        BR     9$
8$:     MOV    #IBLO,R5
        JSR    PC,SALT
        MOVB   #-5,IBNE
9$:     MOV    #005304,(R2)+     ;INSTRUCAO DEC
        MOV    IBNE,(R2)+        ; INSTRUCAO BNE
        TST    (R0)+
        ADD    #2,CONT1
        BR     13$
3$:     ADD    #7,CONT1
        MOV    (R0)+,TAMACH
        TST    FLAG
        BEQ    17$
        MOV    #012223,(R2)+     ;INSTRUCAO MOV
        TST    (R0)+
        BR     13$
17$:    MOV    #022223,(R2)+     ;INSTRUCAO COMP
        MOV    #IBLT,R5
        JSR    PC,SALT
13$:    MOV    TAMACH,R4
        ADD    ICHA,R4
        DEC    R1
        BNE    14$
FTES:   RTS    PC
; **** ROTINA ARRUMA INSTRUCAO DE JUMPS ****
SALT:   CMP    (R0)+,#101        ;COMPARA ORDEM
        BEQ    1$
        ADD    #2,R5
1$:     MOV    DSIM,R4
        SUB    CONT1,R4
        MOVB   R4,(R5)
        MOV    (R5),(R2)+
        INC    R4
        MOV    #IBNE,R5
        MOVB   R4,(R5)
        MOV    (R5),(R2)+
FSALT:  RTS    PC
; **** PALAVRAS RESERVADAS PARA MONTAGEM ROTINAS ****
DSIM:   .WORD 34.
ICHA:   .WORD 0
TAMACH: .WORD 0
FLAG:   .WORD 0
CONT1:  .WORD 0
IBLO:   .WORD 103400
IBHI:   .WORD 101000
IBNE:   .WORD 001000
```

```
IBLT:    .WORD 002400
IBGT:    .WORD 003000
; **** AREA ONDE SERA MONTADA ROTINA TESTA CHAVES ****
TESTA:   CLR   VERD
         .WORD 440,437,436,435,434,433,432,431,430,427,426
         .WORD 425,424,423,422,421,420,417,416,415,414,413
         .WORD 412,411,410,407,406,405,404,403,402,401,400
1$:      INC  VERD
FTESTA:   RTS  PC
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPP            PESQUISA  EM  TABELAS            PPPPPPPPPPPP
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
;  *** PESQUISA TABELAS PARA OBTENCAO PARAMETROS INTERCALACAO ***
;        EQUATES
         TEB1=8.     ;TAMANHO EM BYTES ENTRADA TAB 1
         TEB2=27.    ;NUMERO DE ENTRADAS PARA CADA PASSO
         TEB3=6      ;TAMANHO EM BYTES ENTRADA TAB 2
         TEB4=4      ;DISTANCIA EM BYTES ATE N. CORRIDAS
PTA:     .MONF                    ;OBTENDO MEMORIA INTERNA DISPONI-
         MOV   (SP)+,R2           ;VEL
         .GTPLA
         MOV   (SP)+,R1
         ADD   #512.,R2
         MOV   R2,INAREA
         ADD   TMTAB,R2
         SUB   #200.,R1
         MOV   R1,FIAREA
         SUB   R2,R1
         CMP   R1,#32767.
         BLOS   1$
         MOV   R1,-(SP)
         MOV   #512.,-(SP)
         JSR   PC,NEWDIV
         BR    2$
1$:      MOV   R1,R2
         MOV   #512.,R3
         JSR   PC,DIVID
2$:      MOV   QUOC,MEMS
         MOV   R4,R2
         MOV   TRB,R3
         JSR   PC,DIVID           ;QUANTIDADE DE REGISTROS QUE SO-
         MOV   QUOC,NRESO         ;BRARAM
P2:      JSR   PC,PESQ
         JMP   MERG               ;FASE DE MERGE
; **** ROTINA PESQZ PESQUISA ORDENS DE INTERCALACAO ****
PESQ2:   MOV   R1,R2
         MOV   #TEB2,R4
         JSR   PC,MULT
         MOV   R3,R2
         MOV   #TEB3,R4
```

```
        JSR     PC,MULT
        ADD     #INT2,R3
        MOV     #TEB2,R0
1$:     CMP     RMAX,TEB4(R3)       ;ACHA ORDENS PELO NUMERO DE CORRI-
        BLOS    2$                  ;DAS
        ADD     #TEB3,R3
        DEC     R0
        BNE     1$
2$:     MOV     R3,ENT              ;ENDERECO DA ENTRADA PARA RMAX '
FP2:    RTS     PC                  ;CORRIDAS
; **** ROTINA PESQ1 PESQUISA O NUMERO DE PASSOS ****
PESQ1:  SUB     #4,R2               ;TABELA COMECA COM MEMORIA EM SE-
        MOV     #TEB1,R4
        JSR     PC,MULT
        ADD     #INT1,R3
        CLR     R1
        MOV     #TEB1,R4
        ASR     R4
1$:     INC     R1
        CMP     RMAX,(R3)+          ;IDENTIFICA PELO NUMERO DE CORRI-
        BLOS    FP1                 ;DAS
        DEC     R4
        BNE     1$
FP1:    RTS     PC
; **** ROTINA MOVE AS ORDENS PARA AREA RESERVADA ****
MORD:   MOV     #ARORD,R0           ;ENDERECO AREA
        MOV     ENT,R2
1$:     MOVB    (R2)+,(R0)
        TST     (R0)+
        DEC     R1
        BNE     1$
FORD:   RTS     PC
; **** IMPLEMENTACAO DO ALGORITMO DE PESQUISA DE TABELAS ****
PESQ:   MOV     MEMS,R2             ;MEMORIA EM SETORES
        JSR     PC,PESQ1            ;CBTEM NUMERO DE PASSOS
        TST     NUMEX
        BNE     6$
        CMP     R1,#1               ;NUMERO DE PASSOS=1
        BEQ     7$
        MOV     R1,NPAS             ;NUMERO DE PASSOS
        SUB     #2,R1
        JSR     PESQ2               ;OBTEM ORDENS DE INTERCALACAO
        MOV     NPAS,R1
        JSR     PC,MORD             ;MOVE ORDENS
        JMP     FINAL
6$:     CMP     R1,#1
        BEQ     1$
        MOV     R1,NPAS
        SUB     #2,R1
        BR      2$
```

```
1$:     MOV   MEMS,R2          ;QUANDO NPAS=1 VERIFICA SE EXISTE
        MOV   RMAX,R5          ;POSSIBILIDADE DE INTERCALACAO
        INC   R5               ;UM A MAIS PARA SAIDA
        SUB   R5,R2
        MOV   #512.,R4
        JSR   PC,MULT
        MOV   R3,R2
        MOV   TRB,R3
        JSR   PC,DIVID
        ADD   NRESO,QUOC       ;SOMA REGISTROS QUE SOBRARAM
        CMP   QUOC,R5
        BLO   3$
7$:     MOV   R1,NPAS          ;DA PARA FASER EM UM PASSO
        MOV   RMAX,ARORD       ;ORDEM E O PROPRIO NUMERO DE
        JMP   FINAL            ;CORRIDAS
3$:     MOV   #2,NPAS          ;NAO DA PARA FAZER EM 1 PASSO
        CLR   R1
2$:     JSR   PC,PESQ2
        MOV   NPAS,R3          ;OBTENDO MAIOR ORDEM DE INTERCA-
        DEC   R3               ;LACAO
        ADD   ENT,R3
        MOVB  (R3),PMAX        ;MAIOR ORDEM
        MOV   PMAX,R2
        INC   R2
        CMP   R2,NRESO         ;VERIFICA SE NUMERO REGISTROS QUE
        BLOS  8$               ;SOBRARAM COBRE A FALTA
        SUB   NRESO,R2         ;SE NAO DEU VERIFICA QUANTOS SE-
        MOV   TRB,R4           ;TORES SERAO NECESSARIOS PARA SU-
        JSR   PC,MULT          ;PRIR
        MOV   R3,R2
        MOV   #512.,R3
        JSR   PC,DIVID
        TST   R4
        BEQ   4$
        ADD   #1,QUOC
4$:     MOV   MEMS,R2
        SUB   QUOC,R2          ;NOVA QUANTIDADE DE SETORES DISPO-
        JSR   PC,PESQ1         ;NIVEL
        CMP   R1,NPAS
        BHIS   5$
        MOV   R1,NPAS
        SUB   #2,R1
        BR    2$
5$:     MOV   R1,NPAS
        SUB   #2,R1
        JSR   PC,PESQ2
8$:     JSR   PC,MORD
FINAL:  RTS   PC
; **** PALAVRAS RESERVADAS PARA PESQUISA TABELAS ****
ENT:    .WORD 0
```

```
NRESO:   .WORD 0
PMAX:    .WORD 0
MEMS:    .WORD 0
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTT                    TABELAS                   TTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
; **** TABELA DE NUMERO DE PASSOS ****
INT1:    .WORD 3,7,19.,55.,4.,13.,49.,145.,5,21.,81.,321.,6,31.
         .WORD 126.,626.,7,37.,217.,1081.,7,50.,344.,1081.,8.,64.
         .WORD 512.,750.,9.,81.,312.,750.,10.,90.,100.,400.,11.
         .WORD 30.,125.,500.,12.,30.,150.,625.,13.,42.,216.,750.
         .WORD 13.,42.,216.,624.,14.,49.,216.,400.,14.,49.,343.
         .WORD 750.,15.,56.,216.,400.,17.,64.,392.,499.,16.,72.
         .WORD 150.,475.,15.,72.,180.,454.,10.,72.,216.,434.,11.
         .WORD 100.,216.,400.,14.,100.,294.,320.,11.,90.,294.,384.
         .WORD 12.,56.,294.,369.,12.,56.,150.,320.,13.,64.,180.
         .WORD 320.,13.,72.,332.,400.,14.,72.,321.,450.,14.,72.
         .WORD 216.,312.,14.,72.,216.,302.,15.,76.,216.,300.,15.
         .WORD 81.,180.,192.,16.,100.,276.,400.,16.,100.,269.,400.
         .WORD 17.,96.,262.,400.,17.,90.,252.,300.,18.,72.,249.
         .WORD 300.
; **** TABELA DE ORDENS PARA INTERCALACAO ****
INT2:    .BYTE 2,2,0,0        .BYTE 2,2,2,0         .BYTE 2,2,2,2
         .WORD 4              .WORD 8.              .WORD 16.
         .BYTE 2,3,0,0        .BYTE 2,2,3,0         .BYTE 2,2,2,3
         .WORD 6              .WORD 12.             .WORD 24.
         .BYTE 3,3,0,0        .BYTE 2,3,3,0         .BYTE 2,2,3,3
         .WORD 9.             .WORD 18.             .WORD 36.
         .BYTE 3,4,0,0        .BYTE 3,3,3,0         .BYTE 2,3,3,3
         .WORD 12.            .WORD 27.             .WORD 54.
         .BYTE 4,4,0,0        .BYTE 3,3,4,0         .BYTE 3,3,3,3
         .WORD 16.            .WORD 36.             .WORD 81.
         .BYTE 4,5,0,0        .BYTE 3,4,4,0         .BYTE 3,3,3,4
         .WORD 20.            .WORD 48.             .WORD 108.
         .BYTE 5,5,0,0        .BYTE 4,4,4,0         .BYTE 3,3,4,4
         .WORD 25.            .WORD 64.             .WORD 144.
         .BYTE 5,6,0,0        .BYTE 4,4,5,0         .BYTE 3,4,4,4
         .WORD 30.            .WORD 80.             .WORD 192.
         .BYTE 6,6,0,0        .BYTE 4,5,5,0         .BYTE 4,4,4,4
         .WORD 36.            .WORD 100.            .WORD 256.
         .BYTE 6,7,0,0        .BYTE 5,5,5,0         .BYTE 4,4,4,5
         .WORD 42.            .WORD 125.            .WORD 320.
         .BYTE 7,7,0,0        .BYTE 5,5,6,0         .BYTE 4,4,5,5
         .WORD 49.            .WORD 150.            .WORD 400.
         .BYTE 7,8.,0,0       .BYTE 5,6,6,0         .BYTE 4,5,5,5
         .WORD 56.            .WORD 180.            .WORD 500.
         .BYTE 8.,8.,0,0      .BYTE 6,6,6,0         .BYTE 5,5,5,5
         .WORD 64.            .WORD 216.            .WORD 625.
         .BYTE 8.,9.,0,0      .BYTE 6,6,7,0         .BYTE 5,5,5,6
         .WORD 72.            .WORD 252.            .WORD 750.
```

```
        .BYTE 9.,9.,0,0      .BYTE 6,7,7,0          .BYTE 5,5,6,6
        .WORD 81.            .WORD 294.             .WORD 900.
        .BYTE 9.,10.,0,0     .BYTE 7,7,7,0          .BYTE 5,6,6,6
        .WORD 90.            .WORD 343.             .WORD 1080.
        .BYTE 10.,10.,0,0    .BYTE 7,7,8.,0         .BYTE 6,6,6,6
        .WORD 100.           .WORD 392.             .WORD 1296.
        .BYTE 10.,11.,0,0    .BYTE 7,8.,8.,0        .BYTE 6,6,6,7
        .WORD 110.           .WORD 448.             .WORD 1512.
        .BYTE 11.,11.,0,0    .BYTE 8.,8.,8.,0       .BYTE 6,6,7,7
        .WORD 121.           .WORD 512.             .WORD 1764.
        .BYTE 11.,12.,0,0    .BYTE 8.,8.,9.,0       .BYTE 6,7,7,7
        .WORD 132.           .WORD 576.             .WORD 2058.
        .BYTE 12.,12.,0,0    .BYTE 8.,9.,9.,0       .BYTE 7,7,7,7
        .WORD 144.           .WORD 648.             .WORD 2401.
        .BYTE 13.,12.,0,0    .BYTE 9.,9.,9.,0       .BYTE 7,7,7,8.
        .WORD 156.           .WORD 729.             .WORD 2744.
        .BYTE 13.,13.,0,0    .BYTE 9.,9.,10.,0      .BYTE 7,7,8.,8.
        .WORD 169.           .WORD 810.             .WORD 3136.
        .BYTE 14.,13.,0,0    .BYTE 9.,10.,10.,0     .BYTE 7,8.,8.,8.
        .WORD 182.           .WORD 900.             .WORD 3584.
        .BYTE 14.,14.,0,0    .BYTE 10.,10.,10.,0    .BYTE 8.,8.,8.,8.
        .WORD 196.           .WORD 1000.            .WORD 4096.
        .BYTE 15.,14.,0,0    .BYTE 10.,10.,11.,0    .BYTE 8.,8.,8.,9.
        .WORD 210.           .WORD 1100.            .WORD 4608.
        .BYTE 15.,15.,0,0    .BYTE 10.,11.,11.,0    .BYTE 8.,8.,9.,9.
        .WORD 225            .WORD 1210.            .WORD 5184.
; **** TABELA DO BMIN ****
TBM:    .BYTE 3,3,3,3,3,4,4,4,5,5
        .BYTE 6,6,6,6,7,7,7,7,8.
        .BYTE 8.,8.,8.,8.,8.,9.
        .BYTE 9.,10.,10.,11.,11.,12.,0
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
IIIIIIIIIII          FASE  DE  INTERCALACAO          IIIIIIIIIIII
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
;       EQUATES
        TMTA=300.                  ;TAMANHO TABELA
        ECOR=74
        ENREG=170
        EBUF=264
        EFBUF=360
MERG:   MOV    #ARORD,PORD         ;PONTEIRO PARA ORDENS DO MERGE
        MOV    INAREA,P            ;APONTADOR TABELA DE CORRIDAS
        MOV    INAREA,PRET         ;PONTEIRO DE RETORNO
        MOV    INAREA,ENDTAB       ;ENDERECO PARA NOVAS CORRIDAS
        ADD     TMTAB,INAREA
        MOV    P,TRNB1+2           ;LEITURA DA TABELA
        MOV    TMTAB,TRNB1+4
        ASR    TRNB1+4
        MOV    TEND,TRNB1
        .TRAN #LNK1,#TRNB1
```

```
        .WAIT #LNK1
        CLR   Q                 ;CHAVE
; ****  DIVISAO DAS AREAS DE E/S  ****
DIBUF:  MOV   FIAREA,R1         ;OBTENDO  NUMERO DE BYTES DISPONIVEIS
        SUB   INAREA,R1
        TST   NUMEX
        BEQ   4$
        MOV   @PORD,R2          ;APONTA ORDEM DO MERGE
        INC   R2
        MOV   TRB,R4            ;CALCULA  NUMERO DE BYTES
        JSR   PC,MULT
        SUB   R3,R1
4$:     CMP   R1,#32767.
        BLOS  1$
        MOV   R1,-(SP)
        MOV   #512.,-(SP)
        JSR   PC,NEWDIV
        BR    3$
1$:     MOV   R1,R2             ;OBTENDO MEMORIA EM SETORES
        MOV   #512.,R3
        JSR   PC,DIVID
3$:     MOV   QUOC,R2           ;OBTENDO TAMANHO BUFS
        MOV   @PORD,R3
        INC   R3                ;MAIS O DE OUTPUT
        JSR   PC,DIVID
        TST   QUOC
        BNE   2$
2$:     MOV   QUOC,NSETI        ;NUMERO SETOR BUF INPUT
        ADD   R4,QUOC
        MOV   QUOC,NSETO        ;NUMERO SETOR BUF OUTPUT
        MOV   #512.,R2
        MOV   NSETI,R4
        JSR   PC,MULT
        MOV   R3,TMBUFI         ;TAMANHO BUF INPUT
        MOV   TMBUFI,TMBSRI     ;TAMANHO BUF SEM REG
        MOV   #512.,R2
        MOV   NSETO,R4
        JSR   PC,MULT
        MOV   R3,TMBUFO         ;TAMANHO BUF OUTPUT
        MOV   TMBUFO,TMBSRO     ;TAMANHO BUF SEM REGISTRO
        TST   NUMEX
        BEQ   CBUFO
        ADD   TRB,TMBUFI        ;TAMANHO BUF  INPUT COM 1 REG
        ADD   TRB,TMBUFO        ;TAMANHO BUF  OUTPUT COM 1 REG
CBUFO:  MOV   @PORD,R2
        MOV   TMBUFI,R4
        JSR   PC,MULT
        MOV   INAREA,IBUFO      ;INICIO BUF OUTPUT
        ADD   R3,IBUFO
        MOV   IBUFO,FBUFO       ;FINAL BUF OUTPUT
```

```
        ADD   TMBSRO,FBUFO
        MOV   IBUFO,TRNB2+2     ;INICIALIZA PARA GRAVAR
        MOV   TMBSRO,TRNB2+4
        ASR   TRNB2+4
        MOV   IBUFO,R5          ;R5 PERCORRE BUF OUTPUT
INMER:  MOV   RMAX,NCOR         ;NUMERO DE CORRIDAS
        CLR   RMAX              ;SERA NOVO NUMERO DE CORRIDAS
        CMP   #1,NPAS           ;NUMERO DE PASSOS E UM
        BNE   3$
        TST   Q
        BNE   4$
; ****  POSICIONANDO OS ARQUIVOS NO DISCO  ****
        MOV   EASAI,EASOR2      ;ULTIMO PASSO MOVE PARA ARQUIVO USUARIO
        BR    5$
4$:     MOV   EASAI,EASOR1
        BR    6$
3$:     TST   Q
        BEQ   5$
; ****  FAZENDO A INTERCALACAO  ****
6$:     MOV   EASOR2,TRNB1      ;INICIALIZA PARA LEITURA
        MOV   EASOR1,TRNB2      ;INICIALIZA PARA GRAVACAO
        CLR   Q
        BR    16$
5$:     MOV   EASOR1,TRNB1
        MOV   EASOR2,TRNB2
        INC   Q
16$:    MOV   TRNB2,ENDPC       ;GUARDA END PROMEIRA CORRIDA
        ADD   #4,PRET
7$:     CMP   NCOR,@PORD
        BHIS  8$
        TST   NCOR
        BEQ   21$
        CMP   #1,NCOR
        BNE   10$
        MOV   #1,ORD            ;ORDEM PARA O MERGE
        JSR   PC,MERGE
        BR    9$
10$:    MOV   NCOR,ORD          ;ORDEM PARA O MERGE
        JSR   PC,MERGE
        BR    9$
8$:     SUB   @PORD,NCOR
        INC   RMAX
        MOV   @PORD,ORD
        JSR   PC,MERGE
        JMP   7$
9$:     INC   RMAX
21$:    MOV   ENDPC,@ENDTAB     ;ENDERECO PRIMEIRA CORRIDA
        CMP   R5,IBUFO
        BEQ   40$
        SUB   IBUFO,R5
```

```
        ASR     R5
        MOV     R5,TRNB2+4
        .TRAN   #LNK1,#TRNB2
        .WAIT   #LNK1
40$:    DEC     NPAS
        BEQ     15$
        MOV     ENDTAB,P
        MOV     ENDTAB,PRET
        MOV     PORD,-(SP)
        ADD     #2,PORD
        CMP     @(SP)+,@PORD        ;MESMA ORDEM?
        BEQ     30$
        JMP     DIBUF
30$:    JMP     CBUFO
15$:    MOV     NCR,@P              ;NUMERO DE REGISTROS DA ULTIMA CORRIDA
FIM:    .DELETE #LNK1,#FIL1         ;REMOVER TABELA DISCO
        MOV     #FIL1,R1            ;REMOVER ARQUIVO DE TRABALHO
        MOV     #3,R2
        MOV     #NOMARQ,R3
50$:    MOV     (R3)+,(R1)+
        DEC     R2
        BNE     50$
        .DELETE #LNK1,#FIL1
        .CLOSE  #LNK2
        .RLSE   #LNK2
        .RLSE   #LNK1
        .RLSE   #LNK3
        .EXIT
; ****  ROTINA QUE FAZ A INTERCALACAO  ****
MERGE:  CLR     NCR
IN:     MOV     ORD,CONT
        MOV     P,R0
        MOV     #PTAB,R4
        MOV     INAREA,R1
1$:     MOV     R1,TRNB1+2          ;INICIO AREA DE ENTRADA
        MOV     (R0),TRNB1          ;END DISCO
        MOV     TMBUFI,TRNB1+4      ;PALAVRAS PARA TRANSFERIR
        TST     NUMEX
        BEQ     2$
        MOV     TMBUFI,R2
        SUB     2(R0),R2
        MOV     R4,-(SP)
        MOV     TRB,R3
        JSR     PC,DIVID
        SUB     R4,TRNB1+4
        MOV     (SP)+,R4
2$:     ASR     TRNB1+4
        .TRAN   #LNK1,#TRNB1        ;LEITURA
        MOV     R1,EBUF(R4)         ;INICIO AREA DE ENTRADA
        MOV     R1,EFBUF(R4)        ;FINAL AREA DE SAIDA
```

```
        ADD   TMBSRI,EFBUF(R4)
        MOV   (R0)+,ECOR(R4)    ;ENC CORRIDA NO DISCO
        ADD   NSETI,ECOR(R4)    ;POIS JA LEU
        MOV   (R0)+,(R4)        ;POSICAO DENTRO DO SETOR
        ADD   EBUF(R4),(R4)     ;RELATIVO A MEMORIA
        MOV   (R0)+,ENREG(R4)   ;NUMERO DE REGISTROS
        TST   (R4)+             ;ADIANTA PONTEIRO
        MOV   R0,P
        ADD   TMBUFI,R1
        DEC   CONT
        BEQ   FN
        .WAIT #LNK1
        BR    1$
FN:     .WAIT #LNK1
        MOV   ORD,FIMTAB
        ASL   FIMTAB
        ADD   #PTAB,FIMTAB
RETO:   CMP   #1,ORD
        BEQ   UMCO
        JMP   MER
UMCO:   MOV   #PTAB,R0
4$:     MOV   TMREG,R4
        MOV   (R0),R1
1$:     MOV   (R1)+,(R5)+
        DEC   R4
        BNE   1$
        INC   NCR
        CMP   R5,FBUFO
        BLO   2$
        .TRAN #LNK1,#TRNB2      ;GRAVACAO
        JSR   PC,GRACO
2$:     ADD   TRB,(R0)          ;ATUALIZA POSICAO PONTEIRO
        DEC   ENREG(R0)         ;ATUALIZA NUMERO DE REGI
        BNE   3$
        JMP   FMERG
3$:     CMP   (R0),EFBUF(R0)    ;TESTA AREA DE ENTRADA
        BLO   4$
        JSR   PC,LECO
        BR    4$
MER:    MOV   #PTAB,R1
        MOV   (R1),R2
        MOV   2(R1),R3
        JSR   PC,TESTA
        TST   VERD
        BNE   1$
        MOV   R1,-(SP)          ;O TOPO DA PILHA APONTA O SEGUNDO
        ADD   #2,(SP)
        MOV   R1,-(SP)          ;PALAVRA ANTERIOR APONTA PRIMEIRO
        ADD   #4,R1
        BR    2$
```

```
1$:     MOV   R1,-(SP)
        MOV   R1,-(SP)
        ADD   #2,(SP)
        ADD   #4,R1
2$:     CMP   R1,FIMTAB
        BEQ   5$
        MOV   @2(SP),R2
        MOV   (R1),R3
        JSR   PC,TESTA
        TST   VERD
        BNE   3$
        MOV   2(SP),(SP)        ;NOVO PRIMEIRO
        MOV   R1,2(SP)
        BR    4$
3$:     MOV   @(SP),R2
        MOV   (R1),R3
        JSR   PC,TESTA
        TST   VERD
        BNE   4$
        MOV   R1,(SP)
4$:     CMP   R1,FIMTAB
        BEQ   5$
        TST   (R1)+             ;ADIANTA PONTEIRO
        BR    2$
5$:     CLR   FLA1
18$:    MOV   @2(SP),R0
19$:    MOV   TMREG,R4
6$:     MOV   (R0)+,(R5)+       ;MOVE PARA AREA DE SAIDA
        DEC   R4
        BNE   6$
        INC   NCR               ;NUMERO DE REG CORRIDAS
        CMP   R5,FBUF0
        BLO   7$
        .TRAN #LNK1,#TRNB2      ;GRAVACAO
        JSR   PC,GRACO
7$:     MOV   2(SP),R0
        ADD   TRB,(R0)     ;ATUALIZA POSICAO PONTEIRO
        DEC   ENREG(R0)         ;ATUALIZA NUMERO DE REG
        BEQ   12$
        CMP   (R0),EFBUF(R0)    ;BUF VAZIO
        BLO   13$
        JSR   PC,LECO
13$:    TST   FLA1
        BEQ   17$
        TST   (SP)+             ;LIMPA PILHA
        TST   (SP)+             ;LIMPA PILHA
        JMP   MER
17$:    MOV   (R0),R3
        MOV   @(SP),R2
        JSR   PC,TESTA
```

```
        TST    VERD
        BEQ    18$
        MOV    @(SP),R0          ;SAI PARA OUTPUT O SEGUNDO
        MOV    (SP),2(SP)
        INC    FLA1
        BR     19$
12$:    SUB    #2,FIMTAB
        MOV    FIMTAB,R3         ;REARRUMANDO TABELA
        MOV    (R3),(R0)        ;POSICAO REG
        MOV    ECOR(R3),ECOR(R0)         ;END CORRIDA
        MOV    ENREG(R3),ENREG(R0)       ;NUMERO DE REG
        MOV    EBUF(R3),EBUF(R0)       ;END BUF
        MOV    EFBUF(R3),EFBUF(R0)       ;END FIM BUF
        DEC    ORD
        TST    (SP)+             ;LIMPA PILHA
        TST    (SP)+
        JMP    RETO
FMERG:  MOV    PRET,R1
        MOV    NCR,(R1)+       ;NUMERO REG CORRIDA ANTERIOR
        MOV    R5,R0
        SUB      IBUF0,R0
        MOV    R0,R2
        MOV    #512.,R3
        JSR    PC,DIVID
        MOV    TRNB2,(R1)
        ADD QUOC,(R1)+     ;END DISCO
        MOV    R4,(R1)+
        MOV    R1,PRET
        RTS    PC
; ****  ROTINA PARA LEITURA DE CORRIDAS  ****
LECO:   MOV    EBUF(R0),TRNB1+2 ;END BUF PARA LER
        MOV    TMBUFI,TRNB1+4   ;PALAVRAS PARA LER
        ASR    TRNB1+4
        MOV    ECOR(R0),TRNB1     ;END CORRIDA
        TST    NUMEX
        BEQ    1$
        MOV    (R0),R1
        SUB    EFBUF(R0),R1
        MOV    EBUF(R0),(R0)
        ADD    R1,(R0)           ;ATUALIZA POSICAO REG
        MOV    TMBUFI,R2
        SUB    R1,R2
        MOV    TRB,R3
        JSR    PC,DIVID
        ASR    R4
        SUB    R4,TRNB1+4
1$:     .TRAN  #LNK1,#TRNB1
        TST    NUMEX
        BNE    2$
        MOV    EBUF(R0),(R0)
```

```
        .WAIT #LNK1
        BR    3$
2$:     .WAIT #LNK1
3$:     ADD   NSETI,ECOR(R0)    ;ATUALIZA END DISCO
FLECO:  RTS   PC
; ****  ROTINA PARA SIMPLIFICAR GRAVACAO  ****
GRACO:  TST   NUMEX
        BNE   1$
        MOV   IBUFC,R5
        .WAIT #LNK1
        BR    2$
1$:     MOV   R5,R4
        SUB   FBUFC,R4
        ASR   R4
        MOV   FBUFC,R3
        MOV   IBUFC,R5
        .WAIT #LNK1
3$:     TST   R4
        BEQ   2$
        MOV   (R3)+,(R5)+
        DEC   R4
        BR    3$
2$:     ADD   NSETC,TRNB2
        RTS   PC
; ****   ROTINA DE IMPRESSAO ****
GRA:    MOV   #12.,R1
        MOV   #BUF+6,R2
1$:     MOV   (R0)+,(R2)+
        DEC   R1
        BNE   1$
        .WRITE #LNK2,#BUF
        .WAIT #LNK2
        JMP   FIM
FGRA:   RTS   PC
;*******************************************************************
;***********            ROTINA DE DIVISAO            **************
;*******************************************************************
DIVID:  MOV   R2,R4
        MOV   R3,-(SP)
        CLR   QUOC
1$:     CMP   (SP),R4
        BHI   2$
        ASL   (SP)
        BR    1$
2$:     CMP   (SP),R3
        BEQ   SADIV
        ASL   QUOC
        ASR   (SP)
        BIC   #100000,(SP)
        CMP   (SP),R4
```

```
        BHI     2$
        SUB     (SP),R4
        INC     QUOC
        BR      2$
SADIV:  CLR     (SP)+
        RTS     PC
*****************************************************************
*******         ROTINA DE DIVISAO NUMEROS MAIORES 32767  **********
*****************************************************************
NEWDIV: MOV     #32767.,R2
        MOV     (SP),R3
        JSR     PC,DIVID
        MOV     (SP)+,R3
        BIC     #100000,(SP)
        MOV     (SP)+,R2
        INC     R4
        ADD     R4,R2
        MOV     QUOC,-(SP)
        JSR     PC,DIVID
        ADD     (SP)+,QUOC
        RTS     PC
*****************************************************************
***********             ROTINA DE MULTIPLICACAO       **************
*****************************************************************
MULT:   CLR     R3
2$:     TST     R2
        BEQ     FOR
        BIT     #1,R2
        BEQ     1$
        ADD     R4,R3
1$:     ASR     R2
        ASL     R4
        BR      2$
FOR:    RTS     PC
; ****   MENSAGENS   ****
MENS1:  .ASCII  /ERRO/
        .BYTE 40
        .ASCII  /NO/
        .BYTE 40
        .ASCII  /CARTAO/
        .BYTE 40
        .ASCII  /DE/
        .BYTE 40
        .ASCII  /DADOS/
        .BYTE 40
MENS2:  .ASCII  /NAO/
        .BYTE 40
        .ASCII  /HA/
        .BYTE 40
        .ASCII  /ESPACO/
```

```
        .BYTE 40
        .ASCII /EM/
        .BYTE 40
        .ASCII /DISCO/
        .BYTE 40,40
; ****   PALAVRAS RESERVADAS CRIACAO CORRIDAS E INTERCALACAO ****
NBLOCO: .WORD 0
IALIV:  .WORD 0
TMTAB:  .WORD 0
AUX:    .WORD 0
VERD:   .WORD 0
NOM:    .BLKW 5
ATRAN:  .BYTE 60,60,60,60,60,60
DCHAV:  .BLKW 12.
NOMARQ: .RAD50 /SOR   /
        .RAD50 /   /
NOTAB:  .RAD50 /TABELA/
        .RAD50 /   /
NCHAV:  .WORD 0
EAFOR:  .WORD 0
CRF:    .WORD 0
TMREG:  .WORD 0
EASAI:  .WORD 0
EASOR1: .WORD 0
EASOR2: .WORD 0
TRB:    .WORD 0
QUOC:   .WORD 0
NUMEX:  .WORD 0
NSETOR: .WORD 0
TMBS:   .WORD 0
QPASI:  .WORD 0
QPASO:  .WORD 0
QPASA:  .WORD 0
TMBUF:  .WORD 0
INAREA: .WORD 0
FIAREA: .WORD 0
BUFIN:  .WORD 0
BUFOU:  .WORD 0
FBUFIN: .WORD 0
FBUFOU: .WORD 0
INTAB:  .WORD 0
INLAST: .WORD 0
P:      .WORD 0
TMRP:   .WORD 0
RMAX:   .WORD 0
RQ:     .WORD 0
RC:     .WORD 0
Q:      .WORD 0
LOCX1:  .WORD 0
NCR:    .WORD 0
```

```
TEMP:    .WORD 0
TEND:    .WORD 0
;        PALAVRAS REXERVADAS PARA MERGE
ARORD:   .BLKW 6
FBUFO:   .WORD 0
IBUFO:   .WORD 0
FLA1:    .WORD 0
FIMTAB:  .WORD 0
CONT:    .WORD 0
PTAB:    .BLKW TMTA
ORD:     .WORD 0
ENDPC:   .WORD 0
NPAS:    .WORD 0
NCOR:    .WORD 0
TMBUFI:  .WORD 0
TMBUFO:  .WORD 0
TMBSRI:  .WORD 0
TMBSRO:  .WORD 0
NSETI:   .WORD 0
NSETO:   .WORD 0
PORD:    .WORD 0
ENDTAB:  .WORD 0
PRET:    .WORD 0
MARCA:   .WORD 0
IMARCA:  .WORD 0
FARQ:    .WORD 0
; ****           DEFINICOES DOS BLOCOS DE E/S        ****
BUF:     .WORD 122
         .BYTE 0,0
         .WORD 122
         .=.+120
         .WORD 12
         .WORD 0
LNK3:    .WORD 0
         .RAD50 /CRD/
         .BYTE 1,0
         .RAD50 /CR/
         .WORD 0
         .BYTE 4,0
FIL3:    .WORD 0,0,0,0,0
         .WORD 0
LNK2:    .WORD 0
         .RAD50 /LDP/
         .BYTE 1,0
         .RAD50 /LP/
         .WORD 0
         .BYTE 2,0
FIL2:    .WORD 0,0,0,0,0
TRNB1:   .WORD 0
         .WORD 0
```

```
          .WORD 0
          .WORD 4
          .WORD 0
TRNB2:    .WORD 0
          .WORD 0
          .WORD 0
          .WORD 2
          .WORD 0
          .WORD 0
          .BYTE 1,0
FIL1:     .RAD50 /SAL   /
          .RAD50 /   /
          .WORD 0,0
          .WORD 0
LNK1:     .WORD 0
          .RAD50 /KDO/
          .BYTE 1,0
          .RAD50 /DK/
          .END SORT
$RUN MACRO
#ORD<ORD
$FINISH
```