DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UM SISTEMA DE ARQUIVO EM DISCO PARA ACESSO ALEATÓRIO

Pedro Nogueira Cruz

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 CIÊNCIA (M.Sc.)

Aprovada por:

Célio Guimarães

Nelson Maculan Filho

Pierre Jean Lavelle

RIO DE JANEIRO

ESTADO DO RIO DE JANEIRO - BRASIL

DEZEMBRO DE 1975

A DORA,
DANIELA e
MEUS PAIS

A G R A D E C I M E N T O

Ao professor Célio Guimarães pelo assunto e orientação deste trabalho; ao colega Sálvio Calichio Sobrinho tanto pelo apoio em nosso trabalho como pelo simples companheirismo; aos funcionários do NCE e em especial ao professor Ivan da Costa Marques que nos permitiu a utilização de máquinas indispensáveis ao desenvolvimento de nossos estudos; ao professor Nelson Maculan Filho e funcionários da Secretaria do Programa de Sistemas; aos demais colegas da COPPE que de alguma forma colaboraram com o nosso projeto.

Finalmente agradeçemos a Superintendência do Desenvolvimento do Nordeste - SUDENE pelo financiamento do curso.

R E S U M O

O presente trabalho constitui-se na elaboração e implementação de algorítmos que permitem a geração e recuperação de informação em arquivos de acesso aleatório, residindo em dispositivos de memória externa (discos magnéticos), tendo como objetivo minimização do número de acessos para buscas em arquivos com alto fator de ocupação e utilização eficiente de memória externa.

Esses algorítmos foram implementados no mini-computador PDP-11/10, podendo os mesmos serem implementados em qualquer outro computador.

A utilização é independente das linguagens. Nessa implementação foram testados com Fortran e Assembler.

ABSTRACT

This work consists in the design and implementation of algorithms for generation and retrieval of information in randon access files stored in external memory (magnetic disks), with the objectives of minimizing the average number of accesses for searches in files with high load factor and efficient external memory utilization.

These algorithms have been implemented on a PDP11/10 and can also be implemented in other machines.

The usage is independent of programming languages. In this implementation they were tested in Fortran and Assembler.

ÍNDICE

I. INTRODUÇÃO 1

II. TÉCNICAS DE GERAÇÃO DE ENDEREÇOS E TRATAMENTO DE COLISÕES. 3

2.1 - Métodos de transformação de chaves em endereços ... 3

2.2 - Colisão e Overflow 7

2.3 - Método utilizado 17

III. UM SISTEMA DE ARQUIVO PARA ACESSO ALEATÓRIO 19

3.1 - Arquivo 19

3.2 - Descrição dos algorítmos 23

IV . IMPLEMENTAÇÃO NO PDP 11/10 31

4.1 - Requisitos necessários 31

4.2 - Manual de uso do sistema 33

V. CONCLUSÕES 47

APÊNDICES

A - Fluxogramas 56

B - Listagem dos programas 61

I. INTRODUÇÃO

Tanto em aplicações comerciais como em aplicações científicas é fundamental uma boa organização de arquivo que permita rapidez na recuperação de informações solicitadas de forma aleatória.

Esses tipos de arquivos podem ser utilizados em aplicações onde o tempo de resposta é importante, tais como:

- reserva de passagem aérea;
- controle de estoque;
- consulta bancária.

Podem ser usados também em outras aplicações onde o acesso requerido é aleatório.

Por acesso aleatório neste trabalho entendemos como uma transação com um registro através do fornecimento de uma chave (que identifica univocamente esse registro) escolhida uniformemente dentre todas as chaves do arquivo. Supõe-se também que não existe nenhuma correlação entre sucessivas transações do mesmo tipo; em particular esta organização não é apropriada, por exemplo, para recuperação dos registros em ordem sequencial das chaves.

O desenvolvimento deste trabalho teve como objetivo principal a minimização do tempo médio de acesso ao arquivo e ma

ximização da ocupação da área reservada.

A facilidade de utilização dos algorítmos e a independência em termos de linguagem foram fatores preponderantes no estudo desta implementação, permitindo que o usuário faça uso deste sistema tanto para gerar como para recuperar uma informação no arquivo.

Os testes realizados na fase de implementação mostraram que os objetivos acima citados foram alcançados (vide capítulo V).

II. TÉCNICAS DE GERAÇÃO DE ENDEREÇOS E TRATAMENTO DE COLISÕES

2.1 - Métodos de transformação de chaves em endereços

Seja

$K = \{k_1, k_2, \ldots\ldots, k_c\}$ um conjunto de c chaves distintas, e

$S = \{0, 1, 2, \ldots\ldots, m-1\}$ um conjunto de m endereços lógicos no disco.

Vamos expor resumidamente alguns métodos de transformação de chave em endereço através de uma função h de K em S (função "hash").

A aplicação da função h em uma chave k_i ($h(k_i)$) resultará no endereço do bloco físico no disco ("home address") onde se encontra o registro correspondente a chave k_i.

O ideal de uma função "hash" seria que os valores $h(k_i)$ fossem uniformemente distribuidos entre $\emptyset$ e m-1. A distribuição de endereços gerados depende:

a) - da função h

b) - do conjunto K de chaves.

Dado k_i e k_j, $i \neq j$, se $h(k_i) = h(k_j)$, dizemos que houve uma colisão, isto é, os endereços gerados para duas chaves distintas são idênticos.

Não se conhece um modelo matemático que permita obter uma função "hash" que evite colisão com certeza. A "qualidade"

dos métodos a serem descritos em seguida foi obtida através de resultados experimentais (6,7).

Desse modo uma boa função "hash" deve minimizar o número de colisões e ser rápida de calcular. Essas funções são aplicadas tanto para gerar endereço em memória como em disco. No entanto, em nosso trabalho vamos procurar dar maior ênfase a aplicações de busca externa, onde o endereço é gerado a nível de bloco ("bucket", registro físico, etc.), e cada bloco pode conter um ou mais registros. Nesse caso m é o número de blocos, e a capacidade do arquivo é de m×r, em que r é o número de registros por bloco.

Principais métodos:

- Divisão: neste método definimos a função "hash" como $h(k_i) = k_i \bmod m$. m deve ser um número inteiro positivo de preferência um número primo. Segundo Dodd (6) é suficiente que m seja ímpar e que não seja múltiplo de nenhum número primo menor que 20.

- Mid-Square: seja $k_i \in K$. Este método utiliza a função "hash", onde o valor $h(k_i)$ é os n bits tomados do meio do produto $k_i \times k_i$; n deve satisfazer a condição $2^n = m$. (Fig.1).

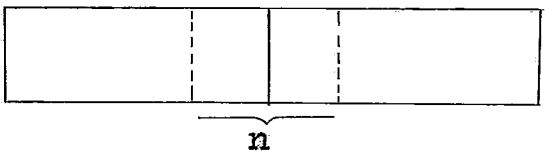


Fig. 1

- Multiplicação (5): escolher A um inteiro primo relativo ao tamanho da palavra do computador. O valor $h(k_i)$ é tomado como sendo os n bits menos significativos da metade do produto. (Fig.2).

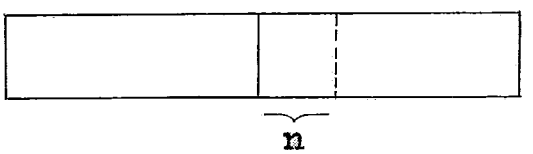


Fig. 2

- Folding (6): é um método bastante útil em aplicações onde o tamanho da chave é maior que uma palavra. A chave é dobrada em um número de partes, cada uma das quais, exceto a última, tem o tamanho do campo de endereçamento do arquivo. A dobra pode ser feita através de soma ou "ou exclusivo" das partes. O valor $h(k_i)$ é o resultado da composição da dobra da chave, com o de outra função (mid-square, divisão, etc.) se esse resultado estiver fora do intervalo $\emptyset$ - (m-1). Este método é chamado "Fold-shifting" ou "Fold-boundary". (Fig. 3).

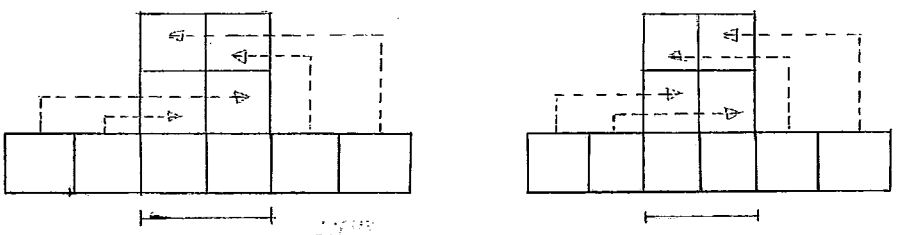


Fig. 3

De acordo com diversos experimentos citados foi constatado que o método de divisão apresentou melhores resultados em relação aos demais. Com base nesses experimentos utilizamos o método de divisão para os casos onde a chave seja menor ou igual a uma palavra de computador. Caso isso não ocorra, antes de aplicar a divisão a chave é reduzida a uma palavra pelo método de "folding", que é um bom método para esses tipos de chave.

Utilizamos a redução da chave para uma palavra como na figura abaixo:

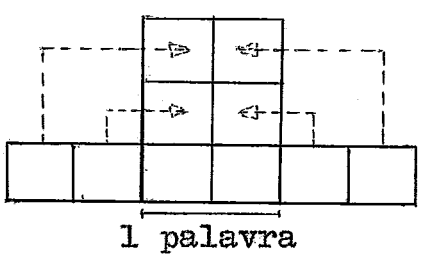


Fig. 4

A composição da chave foi feita através da soma de palavras, uma vez que a soma de bytes dá resultados pequenos, dificultando o espalhamento.

2.2 - Colisão e Overflow

Em geral o arquivo é composto de blocos com vários registros quando se trabalha com busca externa. A vantagem é que as colisões são resolvidas na memória, enquanto existir lugar disponível no bloco cujo endereço foi gerado. Caso o bloco já esteja cheio a inserção de um novo registro provocará "overflow".

"Overflow" pode ser resolvido inserindo o registro em outro bloco que tenha lugar disponível. Os blocos que contêm registros de "overflow" podem ser encadeados ou não, e podem estar na área principal ou em área separada.

Veremos a seguir alguns métodos para resolver "over - flow" sem encadeamento. Esses métodos utilizam a área principal como área de "overflow":

- Busca Linear: se estamos querendo inserir um novo registro e seu "home address" é um bloco cheio, pegamos o próximo, se não tiver lugar disponível prosseguimos sequencialmente até encontrar lugar vazio, ou chegar ao ponto onde a pesquisa de um lugar vago em um bloco foi iniciada (tabela cheia). A vantagem deste método é a facilidade de programação, porém tem uma grande desvantagem, principalmente quando o arquivo está quase cheio, que é a formação de agrupamento primário, (fig. 5), como veremos. Este ocorre quando o método usado para tratamento

de colisões escolhe endereços vizinhos ao do "home address" aumentando a probabilidade condicional de que haja nova colisão naquela região, havendo a tendência de formar "bolsões" de áreas ocupadas no arquivo ("cluster"); o fenômeno é chamado de agrupamento primário e tende a aumentar rapidamente o número médio de acessos a medida que mais registros são inseridos. De um modo geral podemos definir tais políticas de resolver colisão através da notação seguinte:

Seja $h_i(k)$ uma sequência de funções "hash" a serem usadas no tratamento de colisões com a seguinte convenção:

$h_0(k) = h(k)$

$h_j(k)$ = j ésima função "hash" para resolver colisão;

$h_j(k)$ é utilizado se e somente se as posições $h_0(k)$, $h_1(k)$,... $h_{j-1}(k)$ estão ocupadas.

Exemplo: endereçamento aberto com busca linear:

$h_i(k) = (h_0(k)+ci) \bmod m \quad i = 1,2,...$

- Random Probing (9): usa gerador de número pseudo-aleatórios para resolver colisão. É um eficiente método para gerar sucessivos endereços, podendo gerar todo endereço da tabela exatamente uma vez. Cada novo endereço a ser gerado depende apenas do atual e não dos anteriores. Este método elimina agrupamento ("cluster") primário, porém permite formação de agrupamento

secundário, uma vez que todos os registros de "overflow" de mesmo "home address" percorrem a mesma sequência: $h_o(k_i)$, $h_1(k_i)$ ... $h_j(k_i)$. (Fig. 6)

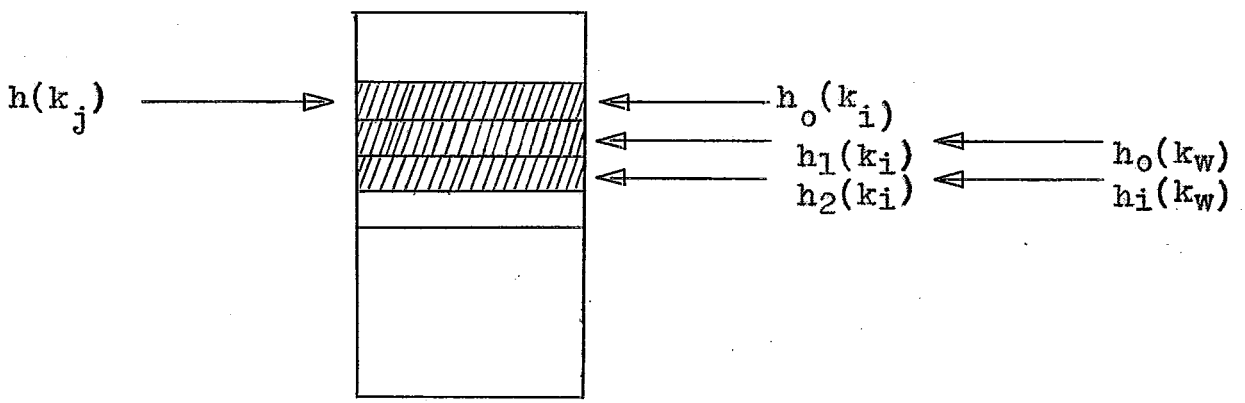


Fig. 5 - Agrupamento primário

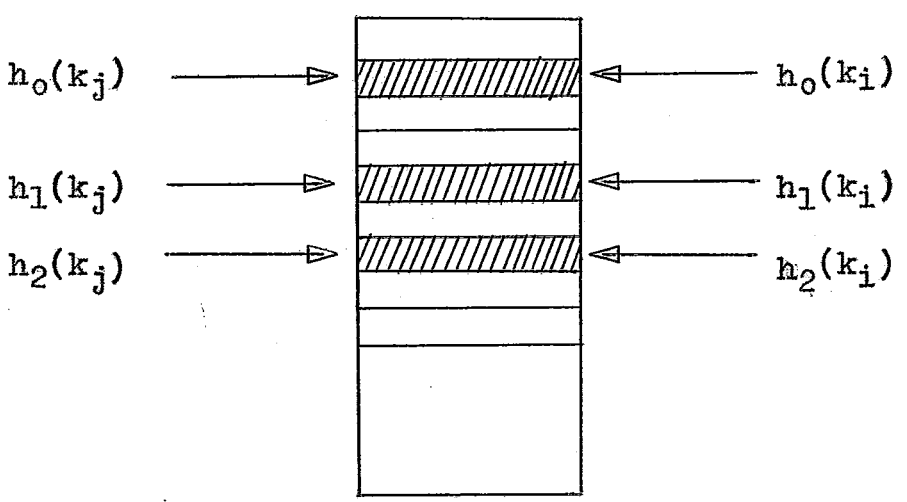


Fig. 6 - Agrupamento secundário

- Quadratic Search (9): dado $k_i \in K$ define $h(k_i) = h_o(k_i) = R$. Caso haja colisão os novos endereços a serem gerados são dados por: $h_j(k_i) = (h_o(k_i)+a\times j+b\times j^2) \bmod m$ onde a e b são constantes. Este método elimina o "cluster" primário, pois os endereços gerados a partir de $h_o(k_i)$ não são sequenciais, porém forma "cluster" secundário.

- Quadratic Quotient (1): este método é modificação no "quadratic search" para que os "clusters" secundários sejam eliminados. Isto é feito tomando como função quadrática $a\times j+b(k_i)\times j^2$, onde o coeficiente $b(k_i)$ depende de k_i. Assim:

$$h(k_i) = h_o(k_i) = R \qquad \text{e}$$

$$h_j(k_i) = (h_o(k_i)+a\times j+b(k_i)\times j^2) \bmod m$$

A função b deve ser escolhida para que elimine "cluster" e seja fácil de calcular. Como no primeiro endereço gerado já fizemos a divisão k_i por m para obtermos o resto, podemos utilizar o quociente para $b(k_i)$, sem nenhum cálculo adicional. Além disso, caso haja colisão de k_i com k_j $(k_i \neq k_j)$ temos $b(k_i) \neq b(k_j)$, uma propriedade desejável para a função b.

Uma das desvantagens dos métodos de busca quadrática propostos na literatura até recentemente é o fato de que o período da busca (número de endereços distintos gerados pela função quadrática de tratamento de colisões) era sempre menor que o ta -

manho da tabela. O método de RADKE (ACM Fev. 70) garante per - correr todo o arquivo porém exige o uso de duas funções quadráticas. Recentemente, no entanto, V. BATAGELJ (CACM abril 75 pp 216-219) mostrou que há funções quadráticas bastante simples que garantem percorrer todo o arquivo.

- Linear Quotient (2): seja R o resto e Q o quociente da divisão de uma chave $k_i \in K$ por m. Este método toma $h(k_i) = h_o(k_i) = R$. Caso haja colisão, endereços subsequentes serão dados por: $h_{j+1}(k_i) = (h_j(k_i)+Q) \bmod m$. A degradação em relação ao método do quociente quadrático é cerca de 4%.

- Método de BRENT (3): este é um método para reorganizar uma tabela de "Hashing" durante a inserção. A idéia de Brent consiste em mudar as posições de chaves na tabela durante a inser - ção, a fim de minimizar o número de visitas durante a busca com sucesso. Deixaremos de dar maiores detalhes por se tratar de um método para aplicações em busca interna (em memória).

Métodos com encadeamento

- Área de "overflow" separada

Este método deve ser usado em aplicações onde a probabilidade de "overflow" é baixa.

As áreas podem ser alocadas de várias maneiras:

- uma área comum para todos os registros de "overflow";
- uma área para cada seção (vários blocos) da área principal;
- uma área em cada cilindro do disco.

Todos os casos acima apresentam mais desvantagens do que vantagens. Quando se tem uma área comum para todos os registros de "overflow" e, se a quantidade de "overflow" é grande, o caminho a ser percorrido será cada vez maior, a medida que os registros são ali inseridos. Usando seções separadas o aproveitamento da área pode se tornar bastante ineficiente. E no caso de listas independentes (mantendo em cada lista apenas registro de "overflow" de mesmo "home address") é usado um bloco da área separada para cada bloco onde tenha ocorrido "overflow" na área principal, a ocupação do espaço poderá ser ineficiente, pois a área de "overflow" pode estar sendo ocupada, mesmo existindo espaços vazios na área principal. No entanto, é possível estimar o número médio de registros de "overflow" (5, pp 537) a fim de se calcular o tamanho da área necessária para os mesmos. O uso de uma área em cada cilindro também torna ineficiente o aproveitamento de espaço do arquivo, pois se a distribuição dos registros não for uniforme as áreas de alguns cilindros não serão usadas. No entanto, essa alternativa parece ser melhor que as demais por

eliminar a movimentação do braço do disco (tempo de "seek").

A principal desvantagem de área separada é que restringe a área endereçável do arquivo, além de provocar grande movimento do braço do disco.

A alocação de área para o arquivo é difícil de ser otimizada porque a área separada é destinada a armazenar apenas os registros de "overflow".

- Área de "overflow" dentro da área principal

Podemos usar o encadeamento como usado com área separada, porém usando como área de "overflow" a própria área principal. O caso aqui se complica um pouco mais porque há uma mistura de registros que pertencem a blocos da área principal e registros de "overflow".

A idéia será criar uma lista de espaço disponível de todos os blocos que ainda disponham de lugar. Cada bloco terá um contador, que será decrementado toda vez que houver inserção. Quando esse contador for zero o bloco será desligado da lista (5, pp 536).

Sempre que ocorrer "overflow" será requisitado um bloco da lista de espaço disponível, onde o novo registro será inserido e o bloco requisitado passará a fazer parte de uma lista iniciada no "home address" do registro inserido. Essas listas

dentro da área principal podem ser independentes ou não. Se as listas são independentes o caminho a ser percorrido será menor do que nos casos em que as listas são coalescentes (listas formadas por registros de diferentes "home address" - ver capítulo V) e ainda, por está sendo usado apenas a área principal, há não só uma minimização de espaço como também de tempo para busca da informação.

Vale salientar que a organização de listas independentes dentro da área principal implica em certa complexidade de programação e de um pouco mais de tempo para fazer inserções, mas as listas são em média curtas e uma vez formadas a busca é rápida.

Resumindo, se ao invés de uma área de "overflow" separada a área principal ficar acrescida dessa área, o intervalo de distribuição das chaves fica maior, proporcionando, assim, um melhor espalhamento e consequentemente menos registros de "overflow". Com isso queremos dizer que é mais conveniente alocar uma área de 120% para o arquivo sem usar área separada (os casos de "overflow" seriam resolvidos na própria área principal) do que se trabalhar com uma área de 100% para área principal e 20% como área de "overflow".

Para escolha de um bom método de tratamento de "overflow" deve ser levado em conta a minimização do número de aces -

sos ao dispositivo usado.

Medidas de Eficiência

Um dos meios usados como medida de eficiência para arquivo de acesso aleatório é o tamanho médio da busca ou número médio de acessos para localizar um registro. Se não existe "overflow" em um arquivo o número médio de acessos é 1,0. A finalidade do tratamento de "overflow" é reduzir o tamanho da busca. Para comparar a eficiência de diversos métodos deve ser levado em conta o número médio de acessos para arquivos com mesmo fator de ocupação (número de registros que caberiam no arquivo) e mesmo fator de bloco (número de registro por bloco).

A eficiência de um arquivo se reduz quando se aumenta o fator de ocupação.

Outra medida seria o tempo médio para recuperar um registro escolhido aleatoriamente

$$\text{número médio de acessos} \times (T/2 + t_b) + t_s$$

onde o número médio de acessos é o resultado teórico como função do tamanho do "bucket" e fator de ocupação (5); e

T = tempo de revolução do disco;

t_b = tempo para transferir um bloco físico;

t_s = tempo médio de "seek",

supondo-se que blocos encadeados estão no mesmo cilindro e que o tempo de busca dentro do bloco (em memória) é desprezível. É possível através dessa fórmula calcular o tamanho ótimo de um "bucket" que minimiza o tempo médio por transação (busca).

Tabela 1

Tempo médio de busca x Fator de bloco

Fator de bloco	Número médio de acessos *	T/2 (ms)	t_b (ms)	t_s (ms)	Tempo médio para recuperar um registro (ms)
1	1,37	20	3,0	20	51,2
2	1,30	20	6,0	20	53,3
3	1,23	20	9,0	20	54,9
4	1,22	20	12,0	20	58,1
5	1,18	20	15,0	20	60,1

* Valor obtido experimentalmente com fator de ocupação = 95%

Tabela 2

Tempo médio de busca x Fator de bloco

Fator de bloco	Número médio de acessos *	T/2 (ms)	t_b (ms)	t_s (ms)	Tempo médio para recuperar um registro (ms)
1	1,5	20	3,0	20	54,5
2	1,4	20	6,0	20	56,4
3	1,4	20	9,0	20	60,6
4	1,3	20	12,0	20	61,6
5	1,3	20	15,0	20	65,5

* Valor teórico obtido com fator de ocupação = 95%

As tabelas 1 e 2 construidas respectivamente com número médio de acessos experimental e teórico, mostram que o fator de bloco ótimo é 1. As tabelas 1 e 2 foram construidas com resultados onde o fator de ocupação é de 95%. Desse modo justificamos em nossa implementação o uso de bloco igual a um setor, uma vez que o disco é setorizado. (Parágrafo 4.1).

2.3 - Método utilizado

Usamos em nossos algorítmos as funções Divisão e "Folding" combinadas, sendo que "Folding" é usada apenas para reduzir o tamanho da chave a uma palavra. Após a dobra da chave aplicamos a função divisão onde o divisor é o número de blocos do arquivo. Usamos o menor número primo para o número de blocos reservados que seja maior que o número requerido pelo usuário.

Como nosso trabalho é com busca externa, procuramos desenvolver um método para resolver os casos de "overflow" de maneira eficiente, ou seja, através de listas independentes. O método a ser usado é, em essência, devido a Lampson (5), não publicado. A área de "overflow" é constituida de blocos da área principal que ainda não estão totalmente ocupados. Esses blocos são ligados através de uma lista duplamente encadeada para maior facilidade no desligamento de setores requisitados. Por outro lado as listas são constituidas por elementos do mesmo "home address"

(listas independentes) a fim de permitir maior rapidez na busca. Desse modo um registro que não pertence ao bloco em que se encontra é deslocado para o fim da lista, quando surge um registro daquele bloco.

E assim, com o aproveitamento desses espaços ociosos, considerando que se eles não fossem usados por registro para ele endereçado nunca seriam ocupados por outros, é possível uma ocupação de quase todo espaço do arquivo. Se em determinado bloco for colocado apenas um registro o espaço restante poderá ser utilizado por registros de outro "home address".

No final de cada setor são controlados os registros através de quatro bytes, onde sabemos se existe registro de outro "home address", se o setor ainda está em disponibilidade, se existe lugar marcado por registro previamente removido e campo para um link no caso do setor fazer parte de uma lista de sinônimos (registros de mesmo "home address").

III. UM SISTEMA DE ARQUIVO EM DISCO PARA ACESSO ALEATÓRIO

A organização do arquivo que daremos a seguir foi definida em função do computador PDP-11/10 onde os algorítmos foram implementados.

3.1 - Arquivo

A área reservada para o arquivo é constituida de blocos do tamanho de um setor logicamente numerados de ∅ a m. Esta organização foi estabelecida visando aproveitar as facilidades de "hardware" do PDP-11 com relação a organização do disco (setorizado), onde a unidade básica de transferência é o setor (512 bytes), e possibilitar o uso de bloco com mais de um registro. O tamanho de um registro está limitado a 512-4 bytes, nesta implementação.

O setor zero é o "header" onde são gravadas informações:

- tamanho do registro;
- tamanho da chave;
- localização da chave;
- tamanho do arquivo (número de setores);
- número de registro por bloco;
- apontador da lista de espaço disponível;

- número de blocos disponíveis.

De cada setor, exceto o "header", são reservadas as duas últimas palavras para controle dos registros, ficando 254 palavras ou 508 bytes para os registros. Na geração do arquivo é calculado o número de registros por setor, em função do tamanho do registro. O arquivo terá uma melhor performance se o espaço útil do setor for múltiplo do tamanho do registro. As duas últimas palavras de cada setor tem as seguintes atribuições:

- 18 bits para controlar os registros no setor, indicando se este é o "home address" do registro ou não. É designado 1 bit para cada registro, sendo, obviamente, permitido um máximo de 18 registros por setor;

- 12 bits para link, utilizado para encadeamento de setores quando ocorrer "overflow". O link nulo indica a não existência de "overflow" no setor;

- 1 bit para indicar a existência de registro anteriormente removido no setor, ou não;

- 1 bit para indicar se o setor atual pertence ou não a lista de espaço disponível.

- Lista de Espaço Disponível

Ao ser gerado o arquivo, todos os setores, exceto o "header", formam uma lista de espaço disponível. (Fig.7).

Um setor deixa de fazer parte da lista de espaço disponível quando ficar totalmente ocupado por seus registros ou quando solicitado por registros de outro "home address". Como vemos é possível ocupar os espaços vazios de um setor por registros que não lhe pertencem. O que não é permitido é requisitar setor da lista de espaço disponível sem o setor requisitante está totalmente cheio, ou está cheio mas com registros removidos ou de outro "home address". Neste caso o registro removido ou de outro "home address" dá lugar ao registro que pertence aquele endereço. Isto nos garante listas independentes.

A lista de espaço disponível é circular e duplamente encadeada. Os apontadores ficam nas primeiras palavras no lugar do último registro. Por ocasião da gravação do último registro, não tendo mais finalidade esses apontadores, eles são removidos e o setor desligado da lista. Assim, nenhum espaço útil é perdido com a utilização dos citados apontadores.

- Limitações do Registro

O tamanho máximo permitido do registro é 508 bytes. Essa limitação, como vimos antes, é para melhor performance na uti

lização do arquivo. O tamanho mínimo é limitado pelo número máximo de registros permitidos no bloco, que é de 18 registros (28 bytes).

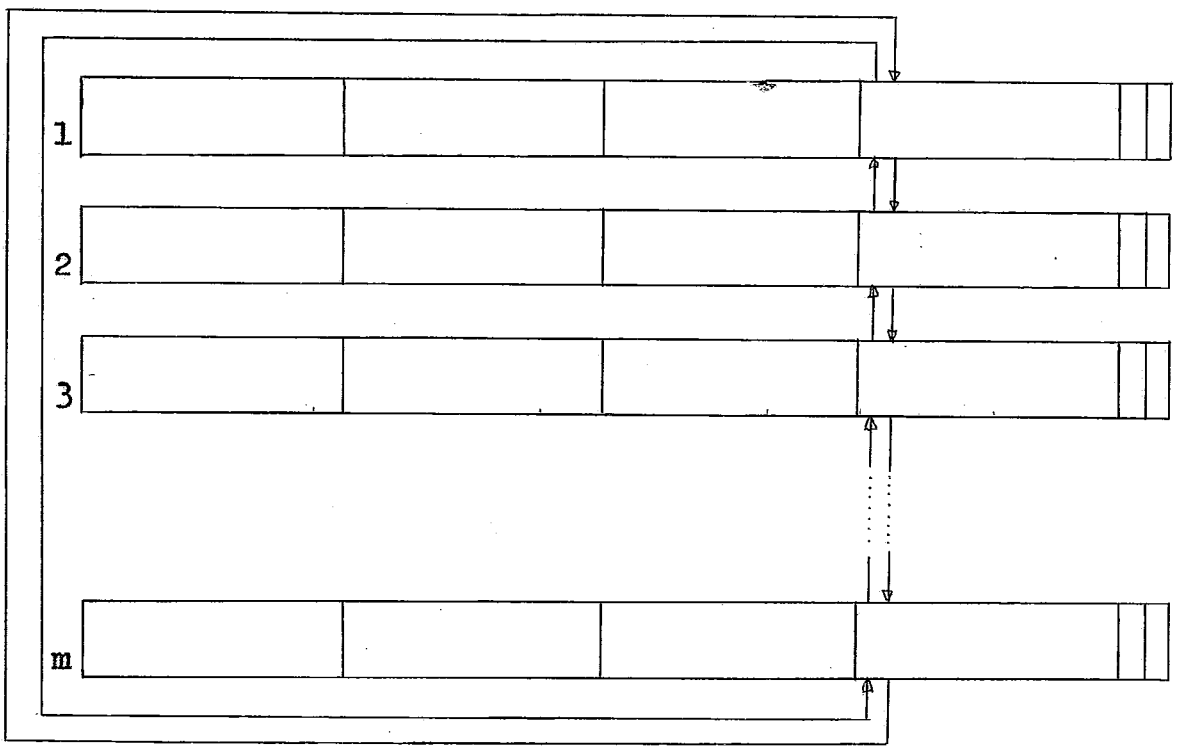


Fig. 7

- Informações sobre a chave

A chave pode ser numérica ou alfa numérica. Se numérica positiva (1 palavra), nenhum tratamento é necessário e a função "hash" (divisão) é aplicada diretamente sobre a chave a, fim de gerar um índice no intervalo (0-(m-1)). Como o setor ∅ é usado para controle do arquivo, ao resto da divisão é somado 1. Se alfa numérica, a chave é dobrada a fim de reduzi-la a uma pala -

vra e o bit de sinal é desligado (tornar positiva). Em seguida é aplicada a divisão.

3.2 - Descrição dos algorítmos

- Geração

O algorítmo de geração tem as seguintes atribuições:

a) - critica os parâmetros do usuário;

b) - calcula tamanho do arquivo e aloca área para o mesmo;

c) - grava "header";

d) - inicializa lista de espaço disponível.

Crítica dos parâmetros: os parâmetros são criticados e em caso de erro, será dada uma advertência ao usuário e o programa termina (detalhes de utilização no parágrafo 4.2).

Alocação de área para o arquivo: o usuário fornece o tamanho do arquivo (número de registros atual + expansão). O algorítmo inicialmente calcula o número de setores suficientes para o arquivo e em seguida, através de um algorítmo de geração de números primos, é procurado o menor número primo que seja maior que o tamanho do arquivo. Isto é importante para se conseguir uma distribuição uniforme na transformação de chaves em endereços.

"Header": o "header" consiste de informações sobre o arquivo que

serão usadas sempre que necessário. As informações são inicializadas em tempo de geração, sendo que algumas não sofrem alteração para um mesmo arquivo (tamanho do arquivo, tamanho do registro, tamanho da chave, localização da chave e número de registros por setor). As informações sobre a lista de espaço disponível (apontador para lista e número de setores disponíveis) são atualizados sempre que um novo setor é requisitado dessa lista.

Lista de espaço disponível: é inicializada nesta etapa e a cabeça da lista aponta para o setor 1.

- Inserção

O algorítmo de inserção tem como atribuições principais:

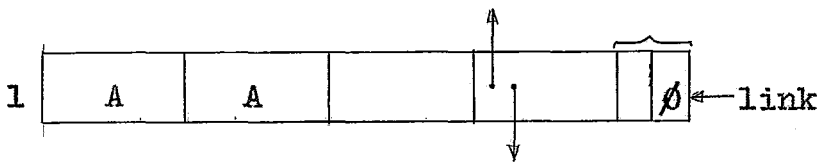
a) - verificar se o registro já existe;

b) - localizar espaço no setor e fazer inserção.

Verificar se o registro já existe: caso o registro já exista a inserção é considerada errônea, e será comunicado ao usuário através de um código de retorno. (Ver parágrafo 4.2).

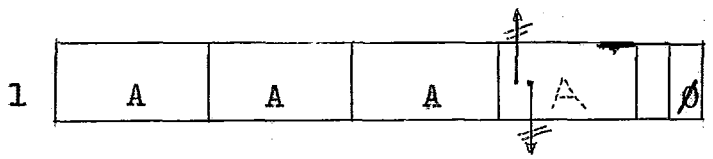
Localizar espaço no setor e fazer inserção: a localização do espaço vazio e inserção pode ocorrer de diversas maneiras:

- existe lugar vazio no seu "home address"



A inserção é feita normalmente no primeiro lugar vazio do seu setor. (Exemplo: setor 1 é o "home address" de A). O setor continua fazendo parte da lista de espaço disponível. Neste caso o link é nulo e os bits de situação de registro e de registro deletado permanecem zerados.

- o único lugar disponível no setor está ocupado pelos apontadores da lista de espaço disponível (lista disponível)



O setor é desligado da lista disponível implicando em:

a) - ler os setores anterior e posterior, alterar seus links e regravá-los;

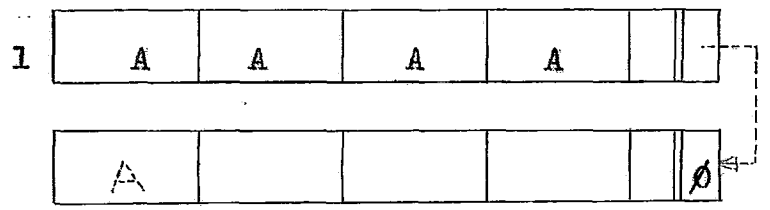
b) - atualizar número de setores disponíveis e cabeça da lista disponível se necessário, e gravar "header" atualizado;

c) - mover o registro do usuário para o lugar onde estavam os apontadores;

d) - ligar o bit para indicar que o setor não pertence a lista

de espaço disponível. Os demais bits das palavras de controle permanecem inalterados.

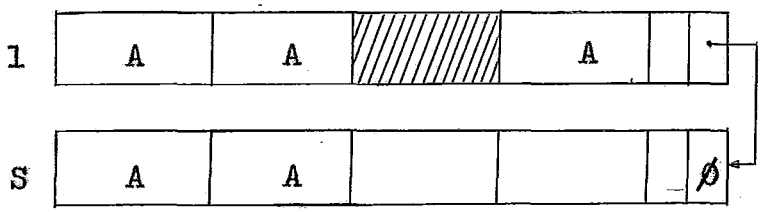
- não há lugar no seu setor que será ocupado por registros ati - vos cujo "home address" é o setor em consideração.



É solicitado um setor da lista de espaço disponível*. O link do setor "home address" do registro do usuário aponta para o setor requisitado. A inserção é feita e é ligado o bit correspondente ao novo registro inserido, indicando que aquele registro é de outro "home address".

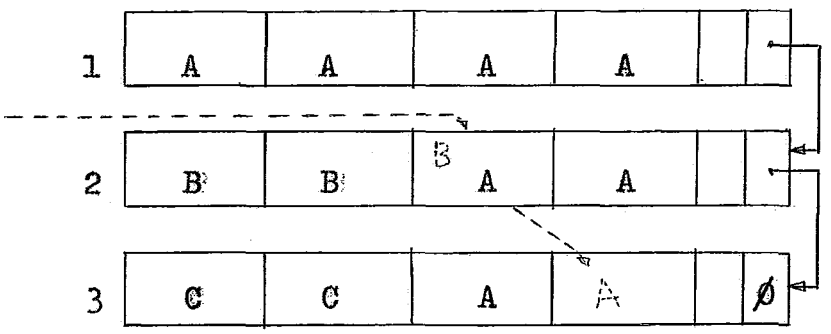
- não há lugar no seu setor mas há lugar marcado por remoção feita anteriormente.

* Quando um setor é solicitado da lista disponível, os apontadores são removidos e esse setor será utilizado apenas por regis - tros desse "home address" e os registros que fazem parte da lisa que esse setor pertence.



A inserção é feita no lugar do registro removido. Se não existe mais registros removidos no setor em manipulação o bit que indica presença de registro removido é zerado.

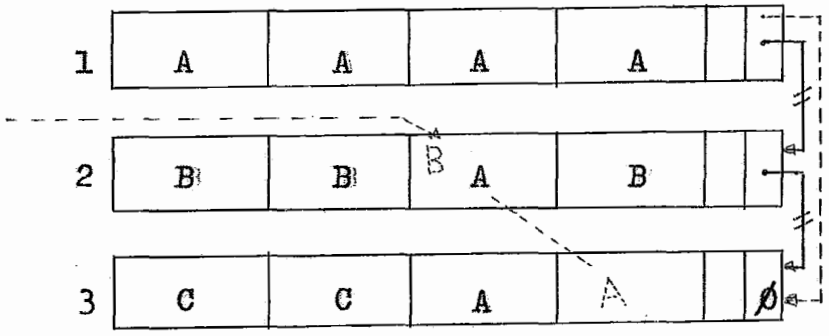
- não há lugar no seu setor. Não há registros marcados por remoção, porém existem registros de outro endereço



No exemplo acima a inserção de um registro "B" se processará após a remoção de um registro "A" que é de outro "home address". O registro de outro setor no caso "A" será colocado no final da lista. Se o final da lista não tiver lugar disponível será requisitado um setor da lista disponível. O novo registro "B" será colocado onde foi removido "A" e os bits de

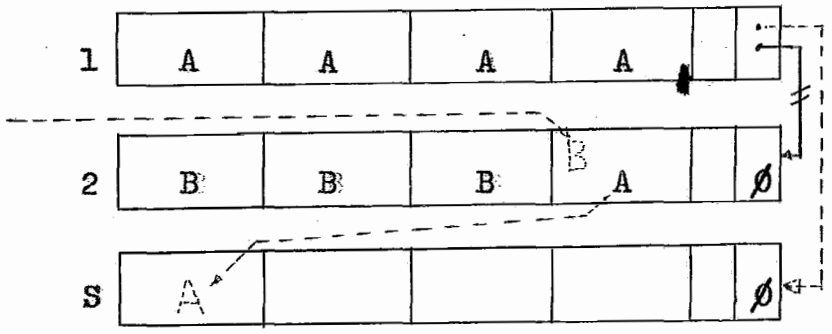
controle serão atualizados de acordo.

- não há lugar no seu setor. Não há registros deletados, existe apenas um registro de outro "home address"



A inserção de um registro "B" acarretaria a remoção do registro "A" que não pertence ao setor. Após essa remoção, como não existe mais registro de outro "home address" o setor de "B" será então desligado. O link do setor de "B" é zerado e os bits alterados de acordo. É importante observar que uma lista só é criada quando o setor onde ela se inicia está cheio, sem registros deletados ou registros de outro "home address". Com isso garantimos a independência das listas.

- um exemplo semelhante ao caso anterior:



Neste caso, para inserção de "B" necessitamos remover o registro de outro setor "A". Como o link do setor "B" é nulo e não há lugar para onde remover "A", requisitamos um setor da lista disponível, colocamos o registro "A" e inserimos o "B" onde "A" foi removido e como não há mais registro de outro "home-address" no setor de "B", então fazemos a ligação do setor de "A" com o novo setor. Os bits de controle são atualizados.

- Busca

É utilizada para recuperar informação. O usuário fornece a chave e recebe o registro solicitado, se for encontrado, e um código de retorno indicando se houve ou não sucesso na busca.

O algorítmo aplica a função "hash" obtendo o endereço do setor onde o registro deve ser procurado. A busca é iniciada por esse setor. Se a chave não for encontrada o link é testado, e, se for nulo, a busca é "sem sucesso" (registro não existe). Caso contrário a busca continua até o fim da lista. O registro sendo encontrado é colocado no "buffer" do usuário.

- Alteração

Esse algorítmo pode ser usado para alterar um determinado registro. O usuário fornece o novo registro. A procura do

registro a ser alterado é feita da mesma maneira que o algorítmo de busca. Se o registro for encontrado é substituido pelo novo registro e ao usuário será comunicado que a alteração foi feita, através de um código devolvido no "buffer" do usuário (detalhes no Manual de uso do sistema). Se o registro não for encontrado será comunicado ao usuário através do "buffer", a fim de que o mesmo, se necessário, tome alguma providência.

- Remoção

A remoção é feita através do algorítmo "REMOVE". O usuário fornece a chave e recebe um código no "buffer" indicando se houve ou não remoção. A localização do registro é feito como no algorítmo de busca.

Se o registro for encontrado será marcado, ou seja, todos os bits de cada palavra do registro serão ligados. Essa configuração é representada por - 1 (menos um), código que não deve ser usado pelo usuário, nas chaves.

O bit de controle de registros removidos será ligado.

IV. IMPLEMENTAÇÃO NO PDP-11/10

4.1 - Requisitos Necessários

Os algorítmos deste sistema foram programados em Assembler e implementados no computador PDP-11/10 do Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro.

A linguagem assembler foi escolhida para programação dos algorítmos por apresentar vantagens tais como:

- economia de memória;
- maiores recursos de programação (programação a nível de bit);
- comunicação com outras linguagens;
- maior rapidez.

O fortran também foi estudado para ligação com o assembler a fim de permitir que usuários fortran tenham acesso a este sistema.

O sistema operacional sob o qual foi implementado o sistema de acesso aleatório foi o DOS/BATCH versão V09 de mono programação. As transferências de entrada e saída são manuseadas pelo sistema em três níveis:

- READ/WRITE
- RECORD/BLOCK
- TRAN

READ/WRITE: neste nível de entrada e saída a transferência dos dados é feita para ou do "buffer" do usuário através do "buffer" do monitor. O usuário lê ou escreve sempre o registro seguinte ao anteriormente lido ou escrito.

RECORD/BLOCK: somente para arquivos estruturados. Usado para acesso por número de registro, conforme explicado a seguir:

- RECORD: lê ou grava um registro pelo seu número de sequência, especificado no arquivo. O programa sempre tem acesso a qualquer registro no arquivo. É permitido usar registro de tamanho menor, igual ou maior que um bloco (setor), porém o processamento será mais rápido se o setor for múltiplo de registro. Os registros são numerados logicamente de $\emptyset$ a α-1. Apesar de o usuário necessitar apenas de um registro de cada vez, o sistema operacional lê ou grava o bloco onde se encontra esse registro, coloca no "buffer" do monitor, transferindo apenas o registro para o "buffer" do usuário.
- BLOCK: lê ou grava um especificado bloco no arquivo. O programa sempre tem acesso a qualquer bloco no arquivo. Os blocos também são numerados logicamente de $\emptyset$ a m-1. O bloco é sempre colocado no "buffer" do monitor, e não é transferido para o "buffer" do usuário. A diferença para o RECORD é que no BLOCK o usuário tem acesso ao "buffer" do monitor, o que é possível

acessar qualquer registro dentro do setor lido.

TRAN: lê ou grava uma determinada quantidade de palavras. Nenhuma estrutura de arquivo é respeitada. O dado é transferido diretamente do dispositivo para o "buffer" do usuário (usa endereço absoluto). A gravação com TRAN é bastante perigosa, pois pode destruir todos os dados de um dispositivo. Se BLOCK pode ser usado ao invés de TRAN, é recomendável. TRAN pode transferir um ou mais blocos, porém toda transferência deve começar em início de setor.

Implementamos os algorítmos com BLOCK pela facilidade de endereçamento, pois, como já vimos, esse nível de acesso usa endereços relativos de disco, coincindindo com os endereços gerados da função "hash" (0-(m-1)).

Este nível de acesso facilita o controle dos registros contidos no bloco (setor) já que trabalha com número exato de registro.

4.2 - Manual de uso do sistema

Os algorítmos foram implementados em forma de subrotinas visando principalmente a facilidade de uso. Foi criada uma biblioteca privada com todas as subrotinas. O usuário terá apenas que "link-editar" seu programa com essa biblioteca. Outras

subrotinas foram também colocadas nessa biblioteca, para abrir e fechar os arquivos. A primeira ("ABRE") tem como finalidade abrir todos os arquivos, ler o "header" e colocar as informações em áreas comuns a todas as subrotinas (global), e deve ser usada antes de qualquer outra subrotina ser chamada. A subrotina "FE - CHA" fecha todos os arquivos usados. Deve ser colocada após a última chamada a subrotina deste sistema. Ambas só necessitam ser usadas uma única vez em cada programa. A exceção ao uso dessas subrotinas é na geração do arquivo.

A programação dessas subrotinas visa principalmente a sua utilização através de programas fortran. A chamada de qual - quer subrotina tem como parâmetros um código de retorno (uma palavra) e o endereço de memória do registro ("buffer" do usuário).

```
Exemplo: CALL ABRE(NOME)
         CALL BUSCA(COD,REG)
         CALL FECHA
```

onde NOME é o nome do arquivo do usuário (ver geração do arquivo), BUSCA é o nome da subrotina de busca, COD é a variável onde é colocado o código de retorno, informando ao usuário se houve ou não sucesso no seu pedido, e REG é o endereço de memória (em fortran, nome do vetor) onde deve ser devolvido o registro, caso seja encontrado. Mesmo na subrotina de Remoção o registro será colocado em REG, a fim de que o usuário tome conhecimento

dos registros removidos, se interessar. Os números usados no código de retorno, serão os seguintes:

1 - com sucesso;

2 - sem sucesso.

As subrotinas podem ser usadas através de programas de qualquer linguagem, desde que os parâmetros sejam fornecidos na mesma sequência. Testamos com Fortran e Assembler. Serão dados exemplos das diversas subrotinas, com ambas as linguagens citadas. Detalhes sobre cada utilização veremos mais adiante.

As operações de entrada e saída devem ser feitas pelo usuário. Achamos que assim será mais conveniente porque deixará o usuário independente das subrotinas. Por exemplo: se está sendo feita uma busca e se esta for com sucesso, o usuário receberá o registro na localização de memória definida pelo parâmetro indicado.

Na chamada de subrotinas através de programas assembler deve ser usado o registrador R5, e o número de parâmetros fornecidos deve ser colocado na palavra seguinte a chamada da subrotina, uma vez que a chamada do programa fortran se processa dessa maneira.

Exemplo:

```
        JSR   R5, BUSCA
NPARAM: .WORD ∅
COD:    .WORD ∅
REG:    .WORD ∅
```

onde NPARAM é a palavra onde deve ser colocado o número de parâmetros neste exemplo 2. COD a palavra onde deve ser colocado o código de retorno e REG o endereço do registro ("buffer" do usuário). Nos exemplos que se seguem usaremos esses mesmos mnemônicos, que tem as mesmas atribuições.

Geração do arquivo

A chamada é feita através do nome da subrotina e dos parâmetros correspondentes.

Exemplo: CALL GERA(TREG,TCHAV,LCHAV,TARQ,NOME),

onde:

GERA é o nome do algorítmo de geração.

TREG (tamanho do registro). O tamanho do registro deve ser dado em caracteres (bytes).

TCHAV (tamanho da chave). O tamanho da chave deve ser dado em bytes.

Exemplo: CHAVE∅∅2

Tamanho da chave: 8

LCHAV (localização da chave). Localização é a posição do primeiro byte da chave dentro do registro. As posições são logicamente numeradas de 1 a N dentro do registro.

Exemplo: 543CHAVE1768
↑ ↑

A localização de CHAVE1 é 4

TARQ (tamanho do arquivo). Este parâmetro compreende o número atual de registros expansão. A expansão deve ser prevista pelo usuário em função da movimentação que o arquivo terá. (Inserção e Remoção).

Exemplo: arquivo atual tem 1.000 registros

expansão de 30% 300

parâmetro fornecido como tamanho do arquivo:

1.300

NOME (nome do arquivo). É permitido a utilização de seis caracteres para nome do arquivo mais três caracteres para a extensão do nome. No entanto o usuário deve reservar espaço para dez caracteres, mesmo que utilize menos, para nome do seu arquivo. E no caso em que o nome tenha menos de 6 (seis) caracteres, a extensão do nome deve ser colocada na sétima posição.

Exemplo:

O usuário poderá colocar os parâmetros para chamar geração de duas maneiras:

a) - com os argumentos separados. Exemplo:

```
CALL GERA(COD,TREG,TCHAV,LCHAV,TARQ,NOME)
```

b) - colocando-se em vetor. Exemplo: lêr um cartão ou digitar pela keyboard

```
0 7 2 0 0 6 0 0 2 5 0 0 A R Q V 0  Ø Ø 1
```

A chamada por Fortran poderia ser:

```
      INTEGER INT(4),NOME(5)
      READ(6,10)INT,NOME
   10 FORMAT(413,5A2)
      CALL GERA(INT,NOME)
      CALL EXIT
      END
```

Para maior facilidade do usuário serão colocados os cartões de controle mais adiante.

Exemplo de geração de arquivo através de um programa Assembler:

```
:
:
```

```
        :
        JSR   R5, GERA
NPARAM: .WORD ∅
TREG:   .WORD ∅
TCHAV:  .WORD ∅
LCHAV:  .WORD ∅
TARQ:   .WORD ∅
NOME:   .WORD ∅
```

O número de parâmetros neste exemplo 5, deve ser colocado em NPARAM. Nos demais parâmetros devem ser colocados os respectivos endereços, antes de ser chamada a subrotina GERA.

Insere

Esta subrotina faz inserção de registros em um arquivo. Exemplo:

```
        INTEGER COD,REG(35)
        CALL ABRE(NOME)
        :
        CALL INSERE(COD,REG)
        :
        CALL FECHA
        CALL EXIT
        END
```

onde,

INSERE é o nome da subrotina de inserção.

Exemplo de inserção de registros no arquivo através de um programa Assembler:

```
        :
        JSR   R5, ABRE
NPARAM: .WORD Ø
NOME:   .WORD Ø
        :
        JSR   R5, INSERE
NPARAM: .WORD Ø
COD:    .WORD Ø
REG:    .WORD Ø
        JSR   R5, FECHA
```

Busca

Esta subrotina busca um registro através da chave fornecida pelo usuário. O registro, se encontrado, é colocado no "buffer" do usuário.

Exemplo:

```
      INTEGER COD,REG(30)
      CALL ABRE(NOME)
      :
      CALL BUSCA(COD,REG)
      :
      CALL FECHA
      CALL EXIT
      END
```

onde, BUSCA é o nome da subrotina de busca. A chamada a ser fornecida pelo usuário para localização do registro, deve ser colocada nas primeiras posições do "buffer" (primeiras posições de REG), mesmo que sua posição dentro do registro no arquivo seja outra.

Exemplo de busca de um registro através de um programa Assembler:

```
          :
          JSR   R5, ABRE
NPARAM:   .WORD ∅
NOME:     .WORD ∅
          :
```

```
        .
        .
        .
        JSR    R5, BUSCA
NPARAM: .WORD  ∅
COD:    .WORD  ∅
REG:    .WORD  ∅
        JSR    R5, FECHA
```

Altera

Esta subrotina altera registros no arquivo. A alteração é feita pela substituição de um registro por outro, ou seja, após o registro ser localizado é substituido pelo novo registro do "buffer" do usuário. Se o usuário desejar fazer alteração em apenas um campo, ela chama BUSCA que localiza e coloca o registro no "buffer" do usuário, aí então o campo pode ser alterado, em seguida chama ALTERA que substitue o registro que está no arquivo pelo alterado. Exemplo:

```
        INTEGER COD,REG(20)
        CALL ABRE(NOME)
        .
        .
        .
        CALL ALTERA(COD,REG)
        .
        .
        .
```

```
          :
          CALL FECHA
          CALL EXIT
          END
```

onde, ALTERA é o nome da subrotina de alteração.

O endereço do registro ("buffer") deve ser movido para REG antes de chamar a subrotina ALTERA.

Exemplo de alteração de um registro através de um programa Assembler:

```
          :
          JSR  R5, ABRE
PARAM:    .WORD Ø
NOME:     .WORD
          :
          JSR   R5, ALTERA
NPARAM:   .WORD Ø
COD:      .WORD Ø
REG:      .WORD Ø
          JSR   R5, FECHA
```

Remove

Esta subrotina remove registros de um arquivo. A remoção é feita marcando o lugar do registro com - 1 (menos 1). Não devem existir chaves com essa numeração.

Exemplo:

```
INTEGER COD,REG(30)
CALL ABRE(NOME)
:
CALL REMOVE(COD,REG)
:
CALL FECHA
CALL EXIT
END
```

onde, REMOVE é o nome da subrotina de Remoção. A chave, através da qual o registro será removido, deve ser colocada nas primeiras posições do "buffer", mesmo que a posição da chave dentro do registro no arquivo não seja no início.

Exemplo de remoção de um registro através de um programa Assembler:

```
:
JSR   R5, ABRE
```

```
        .
        .
        .
        JSR   R5, REMOVE
NPARAM: .WORD ∅
COD:    .WORD ∅
REG:    .WORD ∅
        JSR   R5, FECHA
```

Mais de uma subrotina pode ser chamada de um mesmo programa. As subrotinas ABRE e FECHA só serão usadas uma só vez, como vimos nos exemplos anteriores. Apenas uma subrotina deve ser usada fora desse esquema, a de Geração, que deve ser colocada antes da subrotina ABRE (ou em um programa específico para geração), pois a finalidade de GERA é criar o arquivo, e o arquivo não pode ser aberto antes de ser criado. Exemplo:

```
        CALL GERA(TREG,TCHAV,LCHAV,TARQ,NOME)
        CALL ABRE(NOME)
        .
        .
        .
        CALL INSERE(COD,REG)
        .
        .
        .
        CALL BUSCA(COD,REG)
        .
        .
        .
        CALL FECHA
        CALL EXIT
        END
```

Cartões de Controle

Daremos a seguir os cartões de controle necessários para "compilação" e "link-edição" de programas Fortran (BATCH).

Como o Fortran do PDP-11 também trabalha com duas palavras, para compatibilidade com outros computadores, o usuário deve usar a "switch"/ON na "compilação" (semelhante a ONE WORD INTEGER no computador IBM 1130). A passagem dos parâmetros está prevista com palavras simples.

```
$JOB PROG
$RUN FORTRN
#PROG,LP:<BI:/ON
    :
PROGRAMA FONTE
    :
$RUN LINK
#PROG,LP:<PROG,HASLIB/CC,FTNLIB/L/E
$RUN PROG
```

onde HASLIB é o nome da biblioteca onde estão catalogadas as seguintes subrotinas:

GERA
ABRE
INSERE
BUSCA
ALTERA
REMOVE
FECHA

V. CONCLUSÕES

Os resultados experimentais obtidos na implementação deste trabalho encontram-se na tabela 3. As chaves utilizadas nesses experimentos foram obtidas através do seguinte gerador de números pseudo-aleatórios:

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

(um melhor gerador não foi usado por conveniência de programação para evitar "overflow" na multiplicação).

De um modo geral, os resultados empíricos foram melhores que os teóricos. Convém observar que nas tabelas 4 e 5 os fatores de ocupação não incluem o espaço de "overflow" (área separada). Caso isto fosse feito o fator de ocupação dessas tabelas seria menor que o apresentado o que implicaria que os nossos resultados dariam uma comparação mais favorável ainda.

Verificamos em nossos testes que o caminho máximo para o maior fator de ocupação alcançado (coluna 9 da tabela 3) foi 6 (número de blocos lidos para localizar um registro), o que justifica o uso deste sistema, mesmo nos casos onde há necessidade de tempo de resposta rápido. Pois, em testes realizados com

2.400 registros verificou-se que o tempo médio de uma busca foi de 50 ms, o que representa 300 ms para o pior caso.

Foi constatado também que o tempo médio para inserir um registro foi 2,5 vezes maior que para buscar.

Outro ponto que julgamos importante nesta análise é que o fator de ocupação aumenta quando é reduzido o fator de bloco, atingindo 100% quando o fator de bloco é 1. Neste caso, apesar do arquivo cheio, o caminho máximo foi de 4 acessos e o número médio não passou de 1,4. Com isso podemos concluir que os objetivos definidos inicialmente para este trabalho foram atingidos.

Poderíamos neste ponto fazer algumas sugestões para implementações futuras:

a) - uso de listas coalescentes quando não for mais possível manter listas independentes. Desse modo, todo espaço reservado para o arquivo poderia ser utilizado. No entanto é bom salientar que o caminho máximo deve ser maior do que com listas independentes, quando o arquivo estiver cheio.

b) - alocação de uma área de "overflow" adicional quando o arquivo for considerado "cheio", isto é, quando não for mais possível manter as listas in-

dependentes.

c) - a combinação dessas duas sugestões

Por outro lado é conveniente, em implementações em disco magnético, não setorizado, usar área útil de bloco múltiplo de registro, bem como escolher bloco de tamanho ótimo.

Tabela 3

Resultados dos Testes - Número médio de acessos por busca

Fator de bloco	Fator de ocupação %							Capac. máxima	Caminho máximo
	70	75	80	85	90	95	100		
1	1,25	1,28	1,31	1,32	1,35	1,37	1,40	100,0	4
2	1,18	1,20	1,22	1,25	1,27	1,30	-	97,7	6
3	1,11	1,13	1,15	1,19	1,20	1,23	-	96,8	5
4	1,09	1,12	1,13	1,16	1,18	1,22	-	96,5	5
5	1,07	1,09	1,13	1,14	1,16	1,18	-	95,8	5
10	1,02	1,05	1,07	1,09	-	-	-	88,2	-

Tabela 4

Número médio de acessos para busca - experimental (6)

Fator de bloco	Fator de ocupação %					
	70	75	80	85	90	95
1	1,28	1,31	1,34	1,41	1,38	1,41
2	1,19	1,19	1,26	1,32	1,30	1,34
3	-	-	-	-	-	-
4	-	-	-	-	-	-
5	1,09	1,11	1,13	1,20	1,24	1,25
10	1,03	1,07	1,08	1,25	1,16	1,20

Tabela 5

Número médio de acessos para busca - teórico (5, pp. 535)

Fator de bloco	Fator de ocupação %					
	70	75	80	85	90	95
1	1,350	-	1,400	-	1,450	1,5
2	1,238	-	1,299	-	1,364	1,4
3	1,181	-	1,246	-	1,319	1,4
4	1,145	-	1,211	-	1,290	1,3
5	1,119	-	1,186	-	1,286	1,3
10	1,056	-	1,115	-	1,206	1,3

B I B L I O G R A F I A

1. BELL, J. R. "The quadratic quotient method: a hash code eliminating secondary clustering," Comm. ACM 13, 2 (Fev. 1970), pp. 107-109.

2. BELL, J. R.; e KAMAN, C. M. "The linear quotient hash co - de", Comm. ACM 13, 11 (Nov. 1970) pp. 675-677.

3. BRENT, R. P. "Reducing the retrieval time of scatter storage techniques," Comm. ACM 16, 2 (Fev. 1973), pp. 105-109.

4. GUIMARÃES, Célio. Notas de aula do Curso "Buscas em Arquivos", COPPE 1974.

5. KNUTH, D. E. "The art of computer programming, Vol. III: Sorting and searching", Addison-Wesley, Reading, Mass., 1973.

6. LUM, V. Y.; YUEN, P. S. T.; e DODD, M. "Key to address transform techniques: a fundamental performance study on large existing formatted files," Comm. ACM 14, 4 (abril 1971), pp 228-239.

7. LUM, V. Y.; e YUEN, P. S. T. "Additional results on key-to-address transform techniques," Comm. ACM 15, 11 (Nov. 1972) pp. 996-997.

8. LUM, V. Y. "General performance analysis of key-to-address transformation methods using an abstract file concept," Comm. ACM 16, 10 (Out. 1973), pp. 603-612.

9. MAURER, W. D. "An improved hash code for scatter storage," Comm. ACM 11, 1 (Jan. 1968), pp. 35-38.

10. MAURER, W. D.; e LEWIS, T. G. "Hash table methods," ACM computing surveys, Vol. 7, nº 1 (março 1975).

11. MORRIS, R. "Scatter storage techniques," Comm. ACM 11, 1 - (Jan. 1968), pp. 38-43.

12. TAINITER, M. "Addressing for random access storage with multiple bucket capacities," J. ACM 10, 3 (julho 1963), pp. 307-315.

13. KEITH, R. L. "Techniques for direct access," Auerbach Publi shers Inc., Philadelphia 1973 - Primeira edição.

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

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

16. DOS/BATCH Fortran Compiler and Object Time System, Progra - mmer's Manual, DEC-11-LFRTA-A-D.

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

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

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

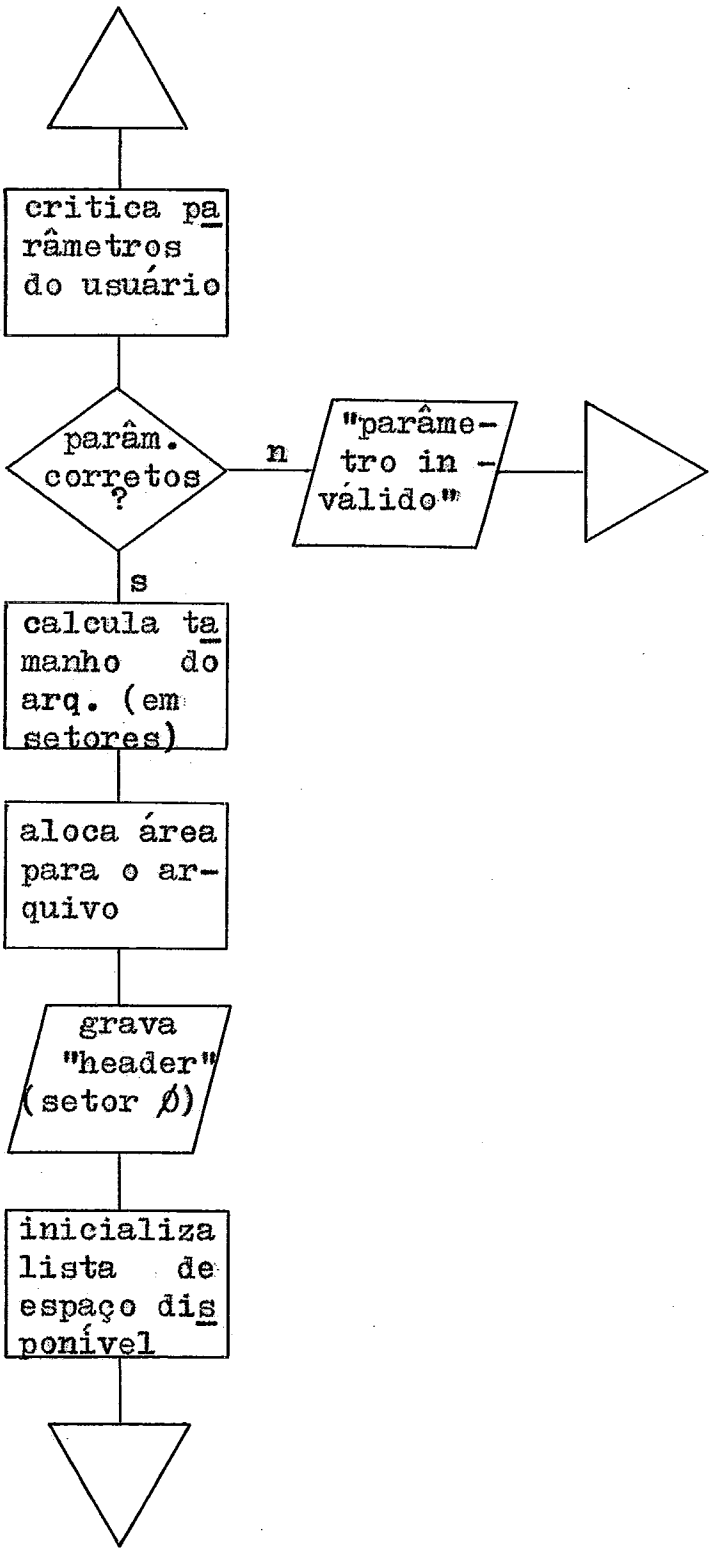
20. DOS/BATCH Librarian (LIBR), Programmer's Manual, DEC-11 - ULBAA-A-D.

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

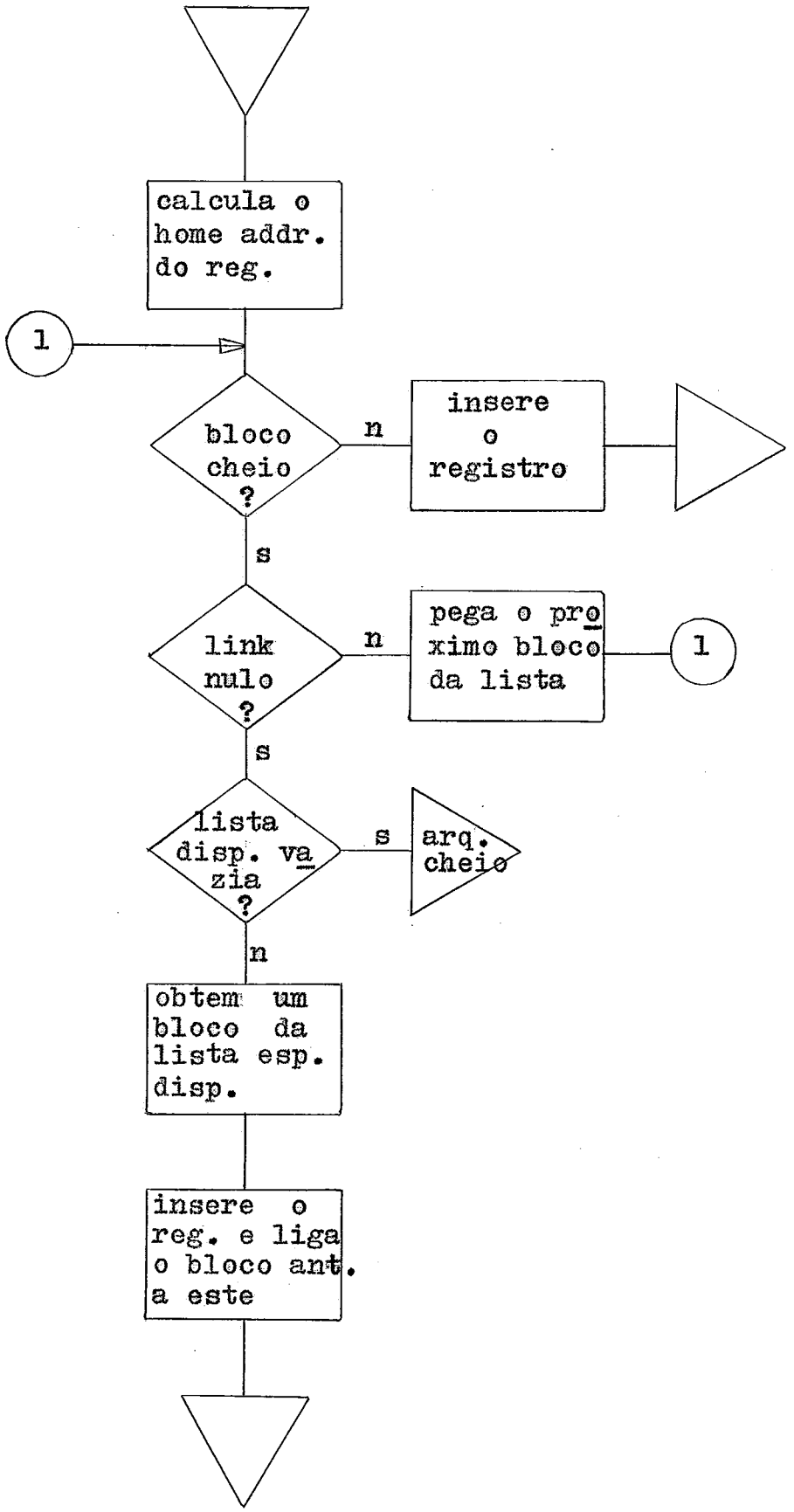
22. PERIPHERALS AND INTERFACING HANDBOOK, Digital Equipment Corporation, 1971.

23. PROCESSOR HABDBOOK, Digital Equipment Corporation, 1973.

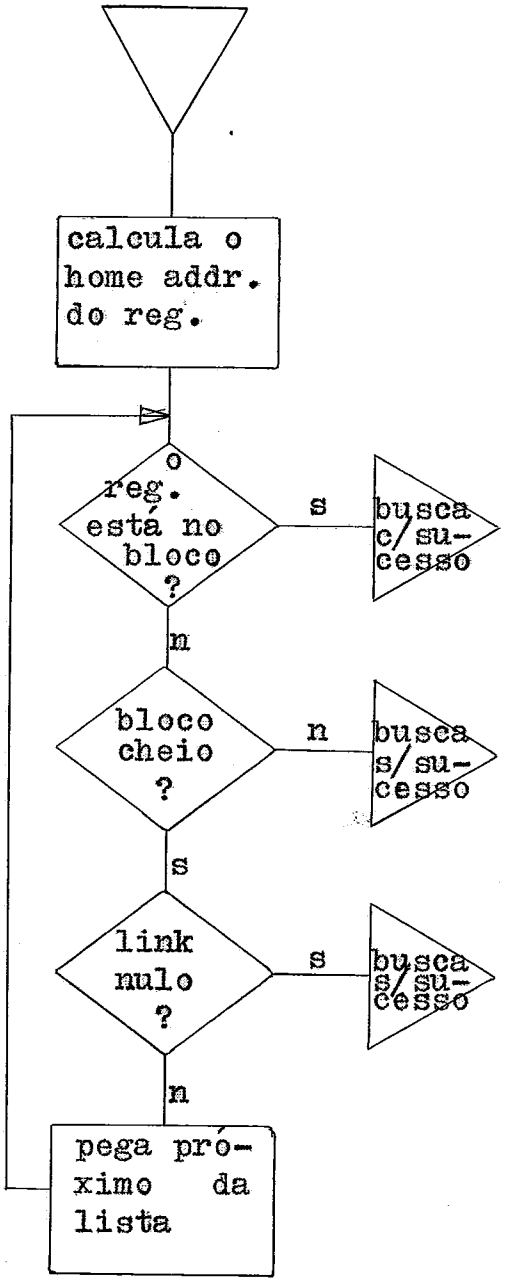
Geração


critica parâmetros do usuário
parâm. corretos ?
n
"parâmetro inválido"
s
calcula tamanho do arq. (em setores)
aloca área para o arquivo
grava "header" (setor ∅)
inicializa lista de espaço disponível

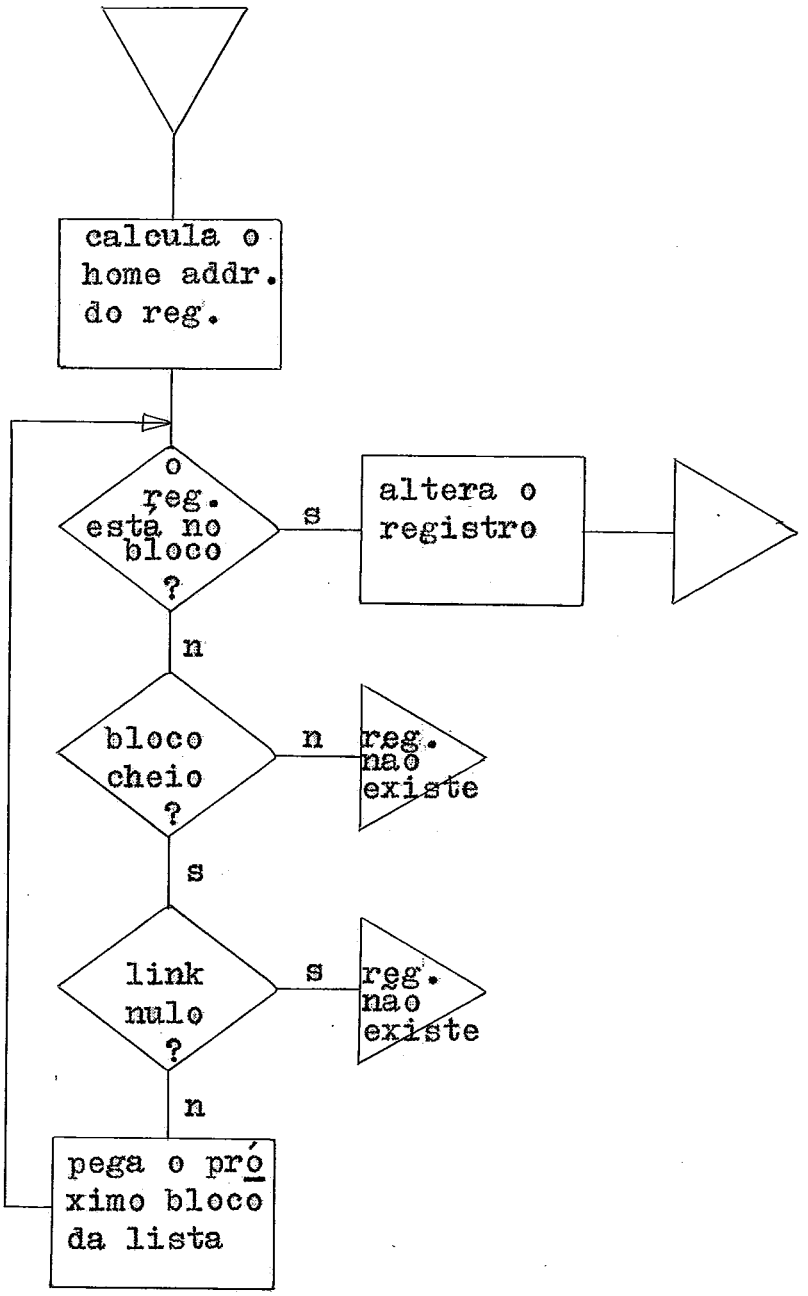
Inserção


calcula o home addr. do reg.
1
bloco cheio ?
n
insere o registro
s
link nulo ?
n
pega o próximo bloco da lista
1
s
lista disp. vazia ?
s
arq. cheio
n
obtem um bloco da lista esp. disp.
insere o reg. e liga o bloco ant. a este

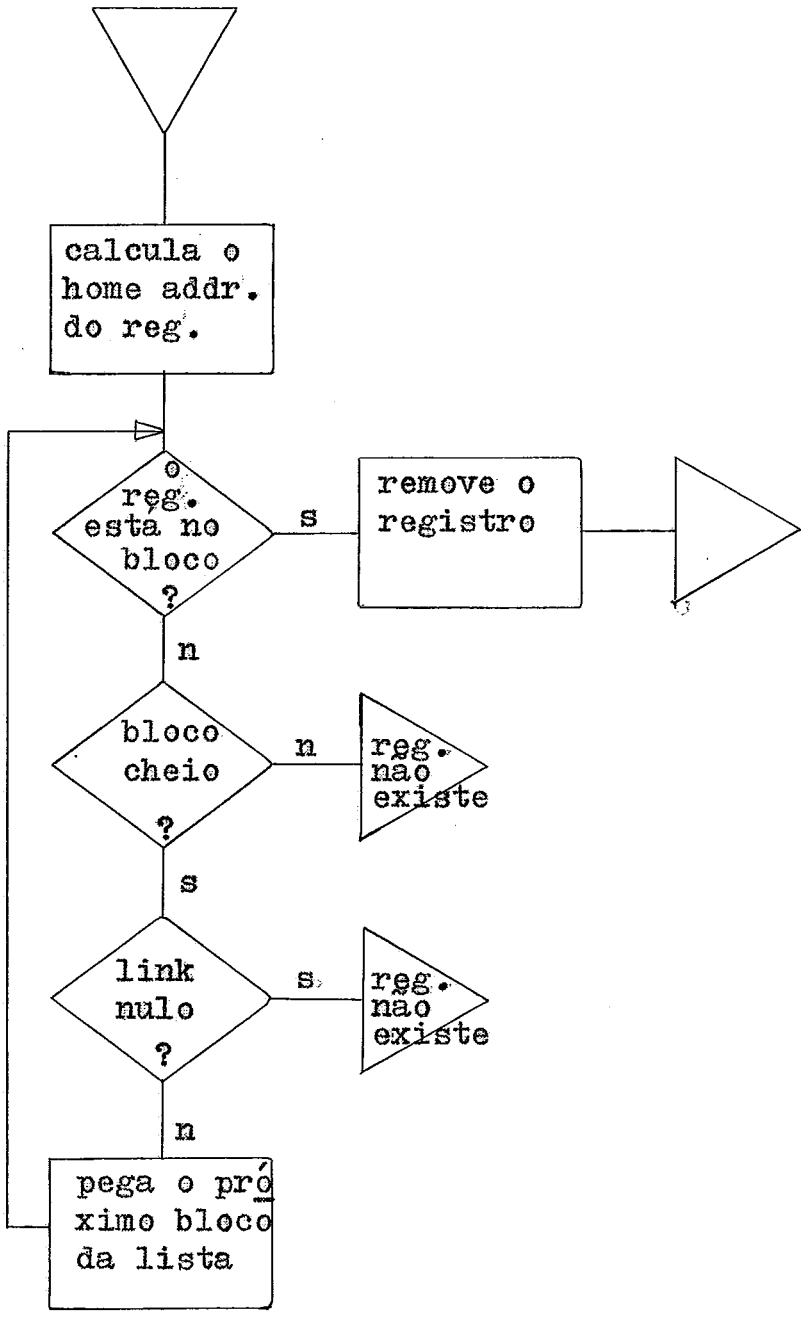
Busca


calcula o home addr. do reg.
o reg. está no bloco ?
s
busca c/ sucesso
n
bloco cheio ?
n
busca s/ sucesso
s
link nulo ?
s
busca s/ sucesso
n
pega próximo da lista

Alteração


calcula o home addr. do reg.
o reg. esta no bloco ?
s
altera o registro
n
bloco cheio ?
n
reg. nao existe
s
link nulo ?
s
reg. nao existe
n
pega o próximo bloco da lista

Remoção


calcula o home addr. do reg.
o reg. esta no bloco ?
s
remove o registro
n
bloco cheio ?
n
reg. nao existe
s
link nulo ?
s
reg. nao existe
n
pega o próximo bloco da lista

```
$JOB ASSMBL *100,100*
$RUN PIP
 GERA.MAC BI*/FA
         .MCALL .INIT,.RLSE
         .MCALL .BIN2D,.BLOCK,.ALLOC,.LOOK
         .MCALL .OPEN,.CLOSE,.WRITE,.WAIT
         .GLOBL GERA
* ****************************
* *** SUBROTINA DE GERACAO ***
* ****************************
GERA*    .INIT   LNK1
         .INIT   LNK3
         MOV     (R5),R4
         CMP     (R4), 1          * COMPARA NO. DE PARAMETROS C/1
                                  * (TODOS OS PARAM. EM 1 VETOR.)
         BEQ     PARAM1
         CMP     (R4), 5          * COMPARA NO. DE PARAM. C/5
         BEQ     PARAM5
* ************************************************
* *** ROTINA DE ERRO - N. DE PARAMETROS INVALIDOS ***
* ************************************************
         MOV      16,R2
         MOV      MENSG2,R0
         MOV      BUF+6,R1
1$       MOV     (R0)+,(R1)+
         DEC     R2
         BNE     1$
         .WRITE   LNK3, BUF
         .WAIT    LNK3
         RTS     R5
* ************************************************************
* *** O USUARIO FORNECEU TODOS OS PARAMETROS EM 1 VETOR ***
* ************************************************************
PARAM1* MOV      2(R5),R1         *R1 CONTEM O ENDERECO DO VETOR
         MOV      TREG,R3
1$*      MOV     (R1)+,(R3)+
         CMP     R3, NREG
         BLO     1$
         MOV      3,R4
         MOV      FIL1,R3
         JSR     PC,TRADIX
         BR      TESTE
* ************************************************************
* *** O USUARIO FORNECEU OS PARAMETROS (5) SEPARADAMENTE. ***
* ************************************************************
PARAM5* MOV      2(R5),TREG
         MOV     4(R5),TCHAV
         MOV     6(R5),LCHAV
         MOV     10(R5),TARQ
         MOV     12(R5),R1
```

```
        MOV     3,R4
        MOV     FIL1,R3
        JSR     PC,TRADIX
*
* *** TESTA VALIDADE DOS PARAMETROS LIDOS
*
TESTE*  TST     TREG            *TESTA TAMANHO DO REGISTRO
        BEQ     ERRO1           *SE IGUAL A ZERO, ERRO1
        CMP     TREG, 774       *COMPARA TAMANHO DO REG. C/ 508
                                * DECIMAL
        BHI     ERRO1           *SE MAIOR, ERRO1
        TST     TCHAV           *TESTA TAMANHO DA CHAVE
        BEQ     ERRO1           *SE IGUAL A ZERO, ERRO1
        CMP     TCHAV, 22       *COMPARA TAMANHO DA CHAVE C/ 18
                                * DECIMAL
        BHI     ERRO1           *DESVIA SE MAIOR
        CMP     TCHAV,TREG      *COMPARA TAM. DA CHAVE C/TAM.DO
                                * REGISTRO
        BHI     ERRO1           *DESVIA SE MAIOR
        MOV     TREG,R0         *MOVE TAMANHO DO REGISTRO P/ R0
        SUB     TCHAV,R0        *SUBTRAI TAMANHO DA CHAVE DE R0
        CMP     LCHAV,R0        *COMPARA LOCALIZACAO DA CHAVE
                                * COM R0
        BHI     ERRO1
*
* *** CALCULA NO. DE REGISTROS POR SETOR
*
        MOV     774,R1          *MOVE 508 DECIMAL P/ R1
SUB1*   INC     NREG            *INCREMENTA NO. DE REGISTRO
        SUB     TREG,R1         *SUBTRAI TAM. DO REG. DE R1
        CMP     TREG,R1
        BLOS    SUB1
        BR      OK
*
* *** ROTINA DE ERRO - PARAMETRO INVALIDO
*
ERRO1*  MOV     11,CONT1        *MOVE . DECIMAL P/ CONT1
        MOV     MENSG1,R0       *MOVE END. DE MENSG1 P/ R0
        MOV     BUF+6,R1        *MOVE ENDERECO DO BUF DO USUARIO
                                * P R0
MOV2*   MOV     (R0)+,(R1)+     *MOVE MENSAGEM P/ BUFFER DO
                                * USUARIO
        DEC     CONT1           *DECREMENTA 1 DE CONT1
        TST     CONT1           *TESTA CONT1
        BNE     MOV2            *DESVIA SE DIFERENTE DE ZERO
        .WRITE  LNK3, BUF
        .WAIT   LNK3
        RTS     R5              *RETORNA AO USUARIO
*
* *** CALCULA TAMANHO DO ARQUIVO EM SETORES
```

```
* *** ALOCA ARQUIVO
* *** OBTEM INFORMACOES SOBRE O ARQUIVO
*
OK*     MOV     TARQ,DENDO          *MOVE TARQ PARA DENDO
                                    * (DIVIDENDO)
        MOV     NREG,DVSOR          *MOVE NREG PARA DIVISOR
        JSR     PC,DIVIDE          *CHAMA ROTINA DE DIVISAO
        MOV     QUOC,TARQ           *TAMANHO DO ARQUIVO EM SETORES
        MOV     QUOC,SALVA          *SOLVA QUOC
* *** PROCURA O MAIOR NUMERO PRIMO MENOR QUE O TAMANHO DO
                                    * ARQUIVO ***
        MOV      3,R0               *INICIALIZA R0 C/ 3
        BIT      1,TARQ             *TESTA SE TAMANHO DO ARQUIVO E
                                    * IMPAR
        BNE     1$                  *DESVIA SE IMPAR
        INC     TARQ                *INCREMENTA DE 1 (SE PAR)
1$*     MOV     TARQ,R1             *SALVA TARQ
        ASR     R1                  *DIVIDE TAMANHO DO ARQUIVO P/ 2
2$*     CMP     R0,R1               *TESTA SE CHEGOU A METADE DO
                                    * TARQ
        BHIS    4$                  *DESVIA SE SIM
        MOV     TARQ,DENDO          *MOVE TARQ P/R3 (DIVIDENDO)
        MOV     R0,DVSOR            *MOVE R0 P/R4 (DIVISOR)
        JSR     PC,DIVIDE           *CHAMA ROTINA DE DIVISAO
        TST     RESTO               *TESTA SE RESTO ZERO
        BEQ     3$                  *DESVIA SE IGUAL A ZERO
        ADD      2,R0               *SOMA 2 EM R0
        BR      2$                  *DESVIA P/2$
3$*     ADD      2,TARQ             *SOMA 2 EM ARQ. (PEGA PROX. NO.
                                    * IMPAR MAIOR)
        BR      2$                  *DESVIA P/ 2$
4$*     INC     TARQ
        ASL     TARQ                *MULTIPLICA TARQ POR 2
        ASL     TARQ                *MULTIPLICA TARQ POR 2
        MOV     TARQ,R0
        .ALLOC   LNK1, FIL1,R0
        .LOOK    LNK1, FIL1,1       *OBTEM INICIO E TAMANHO DO
                                    * ARQUIVO
        MOV     (SP)+,INICIO        *SALVA PARAMETROS
        MOV     (SP)+,NBLOC         *    DA PILHA
        CLR     (SP)+
        MOV     SALVA,TARQ
        .OPEN    LNK1, FIL1         *ABRE O ARQUIVO MESTRE
        .BLOCK   LNK1, BLK1
        .WAIT    LNK1
        MOV     BLK1+4,R0
        MOV     TREG,(R0)+          *MOVE TAM. DO REG. P/BUFM
        MOV     TCHAV,(R0)+         *MOVE TAM. DA CHAVE P/BUFM
        DEC     LCHAV
        MOV     LCHAV,(R0)+         *MOVE LOCAL. DA CHAVE P/BUFM
```

```
          DEC     NBLOC
          MOV     NBLOC,(R0)+       *MOVE NO. DE SETORES P/BUFM
SIGA1*    MOV     NREG,(R0)+        *MOVE NO. DE REGS. P/ BUFM
          MOV      1,(R0)+          *MOV END. DO AVAIL P/ BUFM
          MOV     NBLOC,(R0)+       *NUMERO DE SETORES DISPONIVESS
                                    * (SDISP)
          MOV      2,BLK1
          MOV      0,BLK1+2
*
* *** GRAVA SETOR ZERO  (HEADER)
*
          .BLOCK  LNK1, BLK1
          .WAIT   LNK1
*
* *** LIMPA BUFFER DO MONITOR
*
          MOV      376,CONT1        *MOVE 254 DEC. P/ CONT1
          MOV     BLK1+4,R0
1$*       CLR     (R0)+
          DEC     CONT1             *DECREMENTA 1 DE CONT1
          TST     CONT1             *TESTA CONT1
          BNE     1$
* *** INICIALIZA LISTA DE ESPACO DISPONIVEL
*
          CLR     R2
          CLR     R4
          DEC     NREG
          TST     NREG
          BEQ     3$
2$*       ADD     TREG,R2
          INC     R4
          CMP     R4,NREG
          BLO     2$
3$*       MOV     BLK1+4,R3
          ADD     R2,R3
          MOV     R3,SALVA
          CLR     R2
          CLR     R0
          MOV     NBLOC,(R3)+       *O PRIMEIRO SETOR APONTA P/O
                                    * ULTIMO
          BR      ADD2
MOVE3*    DEC     R0
          MOV     R0,(R3)+
ADD2*     ADD      2,R0
          MOV     R0,(R3)
          INC     R2
          MOV     R2,BLK1+2
          .BLOCK   LNK1, BLK1       *GRAVA SETOR DE AVAIL
          .WAIT    LNK1
          MOV     SALVA,R3
```

```
        CMP     R0,NBLOC
        BLO     MOVE3
        DEC     R0
        MOV     R0,(R3)+
        MOV      1,(R3)             *FAZ O ULTIMO SETOR APONTAR P/ O
                                    * PPRIM.
        INC     R2
        MOV     R2,BLK1+2
        .BLOCK  LNK1, BLK1
        .WAIT   LNK1
* ********************************************************************
* ** INFORMA AO USUARIO*NO. DE BLOCO* E NO. DE REG. POR BLOCO **
* ********************************************************************
        MOV      10,R0
        MOV      MENSG3,R1
        MOV      BUF+6,R2
1$*     MOV     (R1)+,(R2)+
        DEC     R0
        BNE     1$
        .BIN2D  BUF+32,  NBLOC     *NO. *E BLOCOS ALOCADOS
        .WRITE  LNK3, BUF
        .WAIT   LNK3
        MOV      12,R0
        MOV      MENSG4,R1
        MOV      BUF+6,R2
2$*     MOV     (R1)+,(R2)+
        DEC     R0
        BNE     2$
        .BIN2D  BUF+32,  NREG      * NO. DE REG. P/BLOCO.
        .WRITE  LNK3, BUF
        .WAIT   LNK3
        .CLOSE  LNK1
        .CLOSE  LNK3
        .RLSE   LNK1
        .RLSE   LNK3
        RTS     R5                 *RETORNA AO USUARIO
* ********************************************************************
* *** ROTINA QUE CONVERTE CARACTERES ASCII EM RAD50 ***
* ********************************************************************
TRADIX* MOV     R1,-(SP)
1$*     CLR     -(SP)
        EMT     42
        MOV     (SP)+,(R3)+
        DEC     R4
        TST     R4
        BNE     1$
        TST     (SP)+
        RTS     PC
DIVIDE* MOV     DVSOR,AUX
        CLR     QUOC               *ZERA QUOC
```

```
COMP1*  CMP     DVSOR,DENDO     *COMP. DIVISOR C/DIVIDENDO
        BHI     COMP2           *DESVIA SE MAIOR
        ASL     DVSOR           *MULTIPLICA DVSOR P/ 2
        BR      COMP1           *DESVIA P/ COMP1
COMP2*  CMP     DVSOR,AUX       *COMP DVSOR C/AUX
        BEQ     SAIDA           *DESVIA SE IGUAL
        ASL     QUOC            *MULTIPLICA QUOC P/2
        ASR     DVSOR           *DIVIDE DVSOR P/2
        CMP     DVSOR,DENDO     *COMP DVSOR C/ DENDO
        BHI     COMP2           *DESVIA SE MAIOR
        SUB     DVSOR,DENDO     *SUBTRAI DVSOR DE DENDO
        INC     QUOC            *INCREMENTA QUOC DE 1
        BR      COMP2           *DESVIA P/ COMP2
SAIDA*  MOV     DENDO,RESTO
        RTS     PC
MENSG1* .ASCII  /PARAMETRO/
        .BYTE   40
        .ASCII  /INVALIDO/
MENSG2* .ASCII  /NO./
        .BYTE   40
        .ASCII  /DE/
        .BYTE   40
        BYTE    40
        .ASCII  /PARAMETROS/
        .BYTE   40
        .BYTE   40
        .ASCII  /INVALIDO/
MENSG3* .ASCII  /SETORES/
        .BYTE   40
        .ASCII  /ALOCADOS/
MENSG4* .ASCII  /REGISTROS/
        .BYTE   40
        .ASCII  /POR/
        .BYTE   40
        .ASCII  /SETOR/
        .BYTE   40
TREG*   .WORD   0
TCHAV*  .WORD   0
LCHAV*  .WORD   0
TARQ*   .WORD   0
NREG*   .WORD   0
INICIO* .WORD   0
NBLOC*  .WORD   0
SALVA*  .WORD   0
CONT1*  .WORD   0
POSIT*  .WORD   100000
AUX*    .WORD   0
DENDO*  .WORD   0
DVSOR*  .WORD 0
RESTO*  .WORD   0
```

```
QUOC*     .WORD   0
          .WORD   0
LNK1*     .WORD   0
          .RAD50  /DK0/
          .BYTE   1,0
          .RAD50  /DK/
*
          .WORD   0
          .BYTE   1,0
FIL1*     .RAD50  /ARQV01/
          .RAD50  /    /
          .WORD   0,0
*
BLK1*     .WORD   4,0,0,0
*
*
          .WORD   0
LNK3*     .WORD   0
          .RAD50  /LP0/
          .BYTE   1,0
          .RAD50  /LP/
*
          .WORD   0
          .BYTE   2,0
FIL3*     .WORD  0,0,0,0,0
*
BUF*      .WORD   122
          .BYTE   0,0
          .WORD   122
          .=.+120
          .WORD   12
          .END    GERA
$RUN MACRO
 GERA,LP*/NL*TTM GERA
$FINISH
```

```
$JOB ASSMBL *100,100*
$RUN PIP
 ABRE.MAC BI*/FA
        .MCALL .BLOCK,.WAIT
        .MCALL .INIT,.OPEN
        .GLOBL LINK
        .GLOBL LNK1
        .GLOBL LNK3
        .GLOBL FIL1
        .GLOBL FIL3
        .GLOBL BLK1
        .GLOBL BUF
        .GLOBL MENSG1
        .GLOBL TREG
        .GLOBL TCHAV
        .GLOBL LCHAV
        .GLOBL TARQ
        .GLOBL NREG
        .GLOBL AVAIL
        .GLOBL SDISP
        .GLOBL ABRE
ABRE*   .INIT   LNK1
        .INIT   LNK3
        MOV     (R5),R4
        CMP     (R4), 1
        BEQ     RADX
*  ***********************************************************
*  *** ROTINA DE ERRO - N. DE PARAMETROS INVALIDOS ***
*  ***********************************************************
        MOV      16,R2
        MOV      MENSG2,R0
        MOV      BUF+6,R1
1$      MOV     (R0)+,(R1)+
        DEC     R2
        BNE     1$
        .WRITE   LNK3, BUF
        .WAIT    LNK3
        RTS     R5
RADX*     MOV     2(R5),R1
        MOV      3,R4
        MOV      FIL1,R3
        JSR     PC,TRADIX
        .OPEN    LNK1, FIL1
        .OPEN    LNK3, FIL3
        MOV      0,BLK1+2          *BLK1+2 (SETOR A SER LIDO)
        .BLOCK   LNK1, BLK1        *LER HEADER DO ARQUIVO
        .WAIT    LNK1              *ESPERA O FIM DA TRANSFERENCIA
        MOV     BLK1+4,LINK        *COLOCA O END. DA ULTIMA PALAVRA
        ADD      510.,LINK         *DO BUFFER DO MONITOR EM LINK
        MOV     BLK1+4,R0          *MOVE END. DO BUFM P/ R0
```

```
          MOV     TREG,R1
_OOP*     MOV     (R0)+,(R1)+       *MOVE INFO. DO BUFM P/VETOR ARGS
          CMP     R1, SDISP
          BLOS    LOOP
          RTS     R5                *RETORNA AO USUARIO
* ********************************************************
* *** ROTINA QUE CONVERTE CARACTERES ASCII EM RAD50 ***
* ********************************************************
TRADIX* MOV     R1,-(SP)
1$*       CLR     -(SP)
          EMT     42
          MOV     (SP)+,(R3)+
          DEC     R4
          TST     R4
          BNE     1$
          TST     (SP)+
          RTS     PC
MENSG2*   .ASCII  /NO./
          .BYTE   40
          .ASCII  /DE/
          BYTE    40
          BYTE    40
          .ASCII  /PARAMETROS/
          .BYTE   40
          .BYTE   40
          .ASCII  /INVALIDO/
          .WORD   0
LNK1*     .WORD   0
          .RAD50  /DK0/
          .BYTE   1,0
          .RAD50  /DK/
*
*
          .WORD   0
LNK3*     .WORD   0
          .RAD50  /LP0/
          .BYTE   1,0
          .RAD50  /LP/
*
          .WORD   0
          .BYTE   1,0
FIL1*     .RAD50  /ARQV01/
          .RAD50  /   /
          .WORD   0,0
*
*
          .WORD   0
          .BYTE   2,0
FIL3*     .WORD   0,0,0,0,0
*
```

```
*
BLK1*    .WORD   4,0,0,0
*
BUF*     .WORD   122
         .BYTE   0,0
         .WORD   122
         .=.+120
         .WORD   12                    *LINE FEED
LINK*    .WORD 0
TREG*    .WORD   0
TCHAV*   .WORD   0
LCHAV*   .WORD   0
TARQ*    .WORD   0                     *USAR TARQ P/CALCULAR O NO.
                                       * P/ALOCAR ARQ
NREG*    .WORD   0
AVAIL*   .WORD   0
SDISP*   .WORD   0
         .END    ABRE
$RUN MACRO
 ABRE,LP*/NL*TTM ABRE
$FINISH
```

```
$JOB ASSMBL *100,100*
$RUN PIP
 INSERE.MAC BI*/FA
        .TITLE SUBROTINA DE INSERCAO
        .PAGE
        .SBTTL MASCARAS
WORD17 = 16384.     *BIT QUE CONTROLA O 17. REG. DO SETOR
WORD18 = 32768.     *BIT QUE CONTROLA O 18. REG. DO SETOR
MASK1 = 4095.       *12 PRIMEIROS BITS LIGADOS
MASK3 = 65535.      *TODOS OS BITS DE UMA PALAVRA LIGADOS
MASK4 = 57344       *
DELETA = 4096.      *13. BIT LIGADO (REG. REMOVIDO)
        .PAGE
        .SBTTL CORPO DO PROGRAMA
        .MCALL .OPEN,.CLOSE,.RLSE,.EXIT
        .MCALL .ALLOC,.LOOK,.READ,.WRITE
                                    *
        .MCALL .WAIT,.BLOCK,.D2BIN,.INIT,.BIN2D
        .GLOBL INSERE
INSERE* MOV    4(R5),R2             *MOVE END. DO SEGUNDO PARAM.
                                    * P/R2
        JSR    PC,HASH
        MOV    R1,SALVA2
        JSR    PC,BUSCA
LEBLK*  MOV    SALVA2,BLK1+2        *MOVE ENDERECO CALCULADO P/
                                    * BLK1+2
        .BLOCK  LNK1, BLK1          *LER BLOCO INDICADO
        .WAIT   LNK1
        MOV    BLK1+4,W512          *MOV  END. DO BUFFER DO MONITOR
                                    * P/R1
        ADD     510.,W512           *W512 CONTEM END. DA ULT. PAL.
                                    * DO SETOR
        MOV    W512,R1
        SUB     2,R1
        MOV    R1,W510              *W510 CONTEM END. DA PENULT. PAL
                                    * DO SETOR
        MOV     AREATB,WA512
        ADD     510.,WA512          *END. DA ULT. PAL. DA AREATB
        MOV    WA512,R1
        SUB     2,R1
        MOV    R1,WA510             *END. DA PENULT. PAL. DA AREATB
        MOV    NREG,R2
        DEC    R2
        MOV    BLK1+4,R1
        TST    R2                   *TESTA SE SO TEM 1 REG. POR
                                    * BLOCO
        BEQ    2$
1$*     ADD    TREG,R1
        DEC    R2
        TST    R2
```

```
         BNE     1$
2$*      MOV     R1,ENDLK1
         TST     (R1)+
         MOV     R1,ENDLK2
         BIT      DELETA, W512      *TESTA BIT DE REG. REMOVIDO
         BEQ     TLINK1            *DESVIA SE NAO TEM REMOVIDO
         JSR     PC,DELET          *CHAMA ROTINA QUE OCUPA LUGAR DE
                                   * REG.
                                   *PREVIAMENTE REMOVIDO
         JSR     PC,GRAVAR         *CHAMA ROTINA QUE GRAVA UM SETOR
         JMP     FIMSUB
TLINK1*  BIT      MASK1, W512      *TESTA LINK
         BNE     2$
         JSR     PC,LOCAL          *CHAMA ROTINA QUE LOCALIZA
                                   * ESPACO
                                   *NO SETOR
         TST     FLAG
         BEQ     1$
         JSR     PC,INSER
         JSR     PC,GRAVAR
         JMP     FIMSUB
1$*      JSR     PC,TST5
         TST     FLAG1             *TESTA SE INSERIU
         BNE     TESTHA
         JMP     FIMSUB
2$*      BIT      MASK3, W510      *W510 CONTEM PENULTIMA PAL. DO
                                   * SETOR
         BNE     TLINK3            *DESVIA SE TEM OUTRO H.A.
         BIT      49152., W512
         BNE     TLINK3
         JSR     PC,LEPRX          *CHAMA ROTINA P/PERCORRER TODA
                                   * LISTA
         JSR     PC,LOCAL
         TST     FLAG
         BEQ     3$
         JSR     PC,INSER
         JSR     PC,SETHA
         JSR     PC,GRAVAR
         JMP     FIMSUB
3$*      JSR     PC,COPIA
         JSR     PC,AVAIL1
         JSR     PC,LOCAL
         TST     FLAG
         BNE     4$
         HALT
4$*      JSR     PC,SETHA
         JSR     PC,INSER
         JSR     PC,GRAVAR
         MOV     BLK1+2,R1
         BIS     R1, WA512
```

```
          JSR     PC,DEVOLV
          JSR     PC,GRAVAR
          JMP     FIMSUB
TLINK3*   JSR     PC,COPIA
          MOV      W512,SALVA2
          JSR     PC,LEPRX
          JSR     PC,DESLIG
          JMP     FIMSUB
TESTHA*   CLR     FLAG1
          BIT      49152., W512      *TESTA SE TEM REG. DE OUTRO HOME
                                     * ADDRESS
          BNE     4$                *DESVIA SE TEM REG. DE OUTRO
                                     * HOME ADDRESS
          BIT      MASK3, W510       *MASK3 = TODOS OS BITS SETADOS
          BNE     4$                *DESVIA SE TEM REG. DE OUTRO
                                     * H.A.

* $** LOCALIZA REG. DE OUTRO H.A. NO BUFFER DO MONITOR

          JSR     PC,COPIA
          JSR     PC,AVAIL1
          BIS     BLK1+2, WA512     *FAZ A LIGACAO DO SETOR CHEIO
                                     * (AREATB)
                                     *C/ SETOR REQUISITADO DE AVAIL
          JSR     PC,LOCAL
          TST     FLAG
          BNE     1$
          HALT
1$*       JSR     PC,INSER
          JSR     PC,SETHA
          JSR     PC,GRAVAR
          JSR     PC,DEVOLV
          JSR     PC,GRAVAR
          JMP     FIMSUB
4$*       TST     SDISP             *TESTA SE AINDA TEM SETOR
                                     * DISPONIVEL
          BNE     5$                *DESVIA SE TEM
          JMP     FIM               *DESVIA P/FIM SE NAO TEM(ARQUIVO
                                     * CHEIO)
5$*       MOV     W510,R3
          MOV     W512,R4
          JSR     PC,LOCAL2         *CHAMA ROTINA QUE LOCALIZA REG.
                                     * DE OUTRO HA
          MOV     R0,SALVA          *SALVA END. DO REG. DE OUTRO HA
          JSR     PC,GRAVAR
          MOV     BLK1+2,SALVA2
          JSR     PC,AVAIL1         *REQUISITA UM SETOR DE AVAIL
          MOV     BLK1+2,SALVA3     *SALVA NO. DO SETOR REQUISIT. DE
                                     * AVAIL
          JSR     PC,GRAVAR
```

```
        MOV     SALVA2,BLK1+2       *MOVE NO. DO SETOR Q/TEM REG, DE
                                    * OUTRO H.A.
        .BLOCK  LNK1, BLK1
        .WAIT   LNK1
        BIS     SALVA3, W512        *ATUALIZA LINK.(APONTA P/SETOR
                                    * REQUISITA
                                    *DO DE AVAIL
        JSR     PC,COPIA            *COPIA SETOR Q/TEM OUTRO H.A.
                                    * P/AREATB
        MOV     SALVA3,BLK1+2       *MOVE NO. DO SETOR REQ. DE AVAIL
                                    * P/CBLK1+2
        .BLOCK  LNK1, BLK1
        .WAIT   LNK1

* *** O SETOR REQUISITADO DE AVAIL ESTA NO BUFFER DO MONITOR ***

        JSR     PC,LOCAL            *CHAMA ROTINA QUE LOCALIZA
                                    *ESPACO VAZIO NO BUFFER DO
                                    * MONITOR
        TST     FLAG                *TESTA SE ENCONTROU LUGAR VAZIO
        BNE     3$
        HALT
3$*     CMP     R3, 18.
        BNE     B17
        BIS      WORD18, W512
B17*    CMP     R3, 17.
        BNE     B16
        BIS      WORD17, W512
B16*    MOV      1,R2
        MOV      1,R1
1$*     CMP     R2,R3
        BEQ     2$
        INC     R2
        ASL     R1
        BR      1$
2$*     BIS     R1, W510
        MOV      AREATB,R4          *MOVE END. DA AREATB / P/ R4
        ADD     SALVA,R4            *R4 CONTEM END. DO REG. DE OUTRO
                                    * HA
        MOV     R0,AUX              *SALVA END. DO REG. NO BUFFER DO
                                    * MONITOR
        JSR     PC,COPREG           *CHAMA ROTINA QUE COPIA REGISTRO
        MOV     4(R5),R4            *MOVE END. DO PRIMEIRO PARAM.
                                    * P/R4
        MOV      AREATB,R0          *MOVE  END. DA AREATB P/R0
        ADD     SALVA,R0
        JSR     PC,COPREG           *COPIA REGISTRO DO USUARIO
                                    * P/AREATB
        MOV     BLK1+2,SALVA2
        JSR     PC,GRAVAR
```

```
        BIT     MASK3, WA510    *TESTA SE AINDA TEM REG. DE
                                * OUTRO H.A.
        BNE     5$              *DESVIA SE TEM
        BIT     49152., WA512   *TESTA SE AINDA TEM REG. DE
                                * OUTRO H.A.
        BEQ     DESL            *DESVIA P/DESLIGAR SETOR, SE NAO
                                * TEM
5$*     JSR     PC,DEVOLV       *COPIA AREATB P/BUFFER DO
                                * MONITOR
        MOV     ENDSET,BLK1+2   *MOVE NO. DO SETOR DA AREATB
                                * P/BLK1+2
        JSR     PC,GRAVAR       *GRAVA SETOR QUE ESTAVA NA
                                * AREATB
        JMP     FIMSUB
DESL*   MOV     AUX,R2          *R2 CONTEM END. DO REG. NO
                                * BUFFER DO MONITOR
        JSR     PC,HASH         *CHAMA ROTINA Q/CALCULA END. DO
                                * H.A. DO REG.
        MOV     R1,BLK1+2       *R1 CONTEM NO. DO SETOR (H.A.DO
                                * REG.)
7$*     .BLOCK  LNK1, BLK1      *LER SETOR CUJO NO. ESTA EM R1
        .WAIT   LNK1
        MOV     W512,R2         *MOVE CONTEUDO DA ULT. PAL. DO
                                * SETOR P/R2
        BIC     61440.,R2       *ZERA OS BITS QUE NAO PERT. AO
                                * LINK
        CMP     R2,ENDSET       *ENDSET = NO. DO SETOR QUE ESTA
                                * NA AREATB
        BEQ     8$
        MOV     R2,BLK1+2       *MOVE NO. DO PROX. SETOR A SER
                                * LIDO P/BLK1+2
        BR      7$              *DESVIA P/LER PROXIMO SETOR DA
                                * LISTA
8$*     MOV     SALVA2,R2
        BIC     170000,R2
        BIC     MASK1, W512
        BIS     R2, W512        *MOVE LINK DA AREATB P/LINK DO
                                * BUFM
                                *P/ LINK DO BUFFER DO MONITOR
        BIC     MASK1, WA512
        JSR     PC,GRAVAR       *CHAMA ROTINA DE GRAVACAO
        JSR     PC,DEVOLV       *COPAA SETOR DA AREATB P/BUFM
        JSR     PC,GRAVAR       *GRAVAR SETOR INDICADO EM BLK1+2
        JMP     FIMSUB
        .PAGE
        .SBTTL SUB-ROTINAS AUXILIARES
* ***********************************************************
* *** ESTA ROTINA FAZ A TRANSFORMACAO DE CHAVE EM ENDERECO ***
* ***********************************************************
HASH*   CLR     R1
```

```
        ADD     LCHAV,R2            *SOMA LOCALIZACAO DA CHAVE P/R2
                                    *R2 APONTA P/PRIMEIRA POSICAO DA
                                    * CHAVE
        CMP     TCHAV, 2
        BHI     1$
        MOV     (R2),R1
        BR      4$
1$*     MOV     R2,R3
        ADD     TCHAV,R3            *SOMA TAMANHO DA CHAVE EM R3
2$*     MOVB    -(R3),KEY+1
3$*     MOVB    (R2)+,KEY
        ADD     KEY,R1
        CMP     R2,R3
        BHI     4$
        BLO     2$
        BR      3$
4$*     BIC      100000,R1
        CMP     TARQ,R1
        BLOS    5$
        INC     R1
        RTS     PC
5$*     MOV     TARQ,R0
6$*     SUB     R0,R1
        CMP     R0,R1
        BLOS    6$
        INC     R1
        RTS     PC
BUSCA*  MOV     R1,BLK1+2
        .BLOCK   LNK1, BLK1         *LER SETOR DO END. GERADO
        .WAIT    LNK1
        MOV     BLK1+4,R1           *MOV END DO BUFM P/ R1
        MOV     R1,SALVA            *SALVA R1
ARGS*   MOV     4(R5),R2

        MOV     TCHAV,R0
1$*     CMPB    (R1)+,(R2)+
        BNE     2$
        DEC     R0                  *DECREMENTA R0 DE 1
        TST     R0                  *TESTA SE R0 IGUAL A ZERO
        BNE     1$
        MOV     2(R5),R2
        MOV      2,(R2)
        RTS     R5
2$*     MOV     SALVA,R2

* *** TESTA SE CHAVE IGUAL A ZERO
        MOV     TCHAV,R0
ZERO*   TSTB    (R2)+
        BNE     TEST3
        DEC     R0
```

```
         TST     R0
         BNE     ZERO
         RTS     PC

* *** TESTA SE CHEGOU AO FIM DO SETOR

TEST3*   INC     CONT1
         CMP     CONT1,NREG
         BHIS    TLINK
         ADD     TREG,SALVA
         MOV     SALVA,R1
         BR      ARGS

* *** TESTA SE LINK NULO

TLINK*   CLR     CONT1
         BIT      7777, LINK
         BNE     1$
         RTS     PC
1$*      MOV      LINK,R1
         BIC      170000,R1
         BR      BUSCA
*
* ************************************************************
* * DESLIGA SETOR DA LISTA DE REGISTROS DE MESMO 'HOME ADDRESS' *
* ************************************************************
*
DESLIG*  JSR     PC,LOCAL          *CHAMA ROTINA P/LOCALIZAR VAZIO
                                   * NO BUFM
         TST     FLAG              *TESTA SE HA ESPACO
         BNE     2$                *DESVIA SE TEM LUGAR DISP. NO
                                   * SETOR
         TST     SDISP             *TESTA SE AINDA EXISTE SETOR
                                   * DISPONIVEL
         BNE     1$                *DESVIA SE EXISTE
         JMP     FIM               *DESVIA SE ARQUIVO CHEIO
1$*      BIC      61440.,AVAIL     *ZERA BITS Q/NAO PERTENCEM AO
                                   * LINK
         BIS     AVAIL, W512       *LIGA BITS DE W512 (LINK)
         JSR     PC,GRAVAR         *CHAMA SUBROT. P/GRAVAR O SETOR
                                   * DO BUFM
         JSR     PC,AVAIL1         *REQUISITA UM SETOR DO AVAIL
         JSR     PC,LOCAL          *CHAMA ROTINA P/LOCALIZAR VAZIO
                                   * NO
                                   *SETOR REQUISITADO
         TST     FLAG              *TESTA SE HA ESPACO
         BNE     2$
         HALT
2$*      JSR     PC,SETHA          *SETA BIT DO REG. DE OUTRO H.A.
                                   * A SER INSERIDO NO BUFFER DO
```

```
                                    * MONITOR
        MOV     R0,AUX              *SALVA LOCAL DO REG. VAZIO NO
                                    * SETOR
                                    *REQUISITADO DE AVAIL
        MOV     WA510,R3
        MOV     WA512,R4
        JSR     PC,LOCAL2
        MOV      AREATB,R4
        ADD     R0,R4               *SOMA LOC. DO REG. DE OUTRO HA
                                    * EM R4
                                    *R4 APONTA P/REG. DE OUTRO H.A.
        MOV     R4,SALVA
        MOV     AUX,R0              *DEVOLVE O APONTADOR DO REG.
                                    * VAZIO P/R0
        JSR     PC,COPREG           *CHAMA ROTINA QUE COPIA REGISTRO
        JSR     PC,GRAVAR           *CHAMA ROTINA DE GRAVACAO
        MOV     SALVA,R0            *COLOCA EM R0 N END.DO REG. DE
                                    * OUTRO HA
        MOV     4(R5),R4
        JSR     PC,COPREG
        BIT      MASK3, WA510
        BNE     4$                  *DESVIA SE TEM AINDA REG. DE
                                    * OUTRO H.A.
        BIT      49152., WA512
        BNE     4$
        JMP     DESL
4$*     JSR     PC,DEVOLV
        JSR     PC,GRAVAR
        RTS     PC
* **********************
* *** COPIA REGISTRO ***
* **********************
COPREG* CLR     R1                  *ZERA R1
1$*     MOVB    (R4)+,(R0)+         *R4 CONTEM END. DO REG. NA
                                    * AREAT2
                                    *R0 CONTEM END. DO LUGAR ONDE O
                                    * REG. VAI
                                    *VAI SER INSERIDO
        INC     R1                  *INCREMENTA R1 DE 1
        CMP     R1,TREG             *COMP. C/TAMANHO DO REGISTRO
        BLO     1$                  *DESVIA SE MENOR
        RTS     PC
* ***********************************************
* *** LOCALIZA REGISTRO DE OUTRO HOME ADDRESS ***
* ***********************************************
LOCAL2* MOV      1,R0               *INICIALIZA R0 C/ 1
        CLR     R1                  *INICIALIZA R1 C/ ZERO
BITR0*  BIT     R0,(R3)
        BEQ     INCR1
        TST     FLAG
```

```
         BEQ     BICR0
         BIS     R0,(R3)
BICR0*   BIC     R0,(R3)
         BR      CLRR0
INCR1*   INC     R1                  *INCREMENTA R1 (NO. DO REG. DE
                                     * OUTRO H.A.)
         CMP     R0, 32768.          *TESTA SE R0 ESTA C/BIT 15
                                     * LIGADO
         BEQ     3$                  *DESVIA SE IGUAL
         ASL     R0                  *MULTIPLICA R0 POR 2
         BR      BITR0
3$*      BIT      WORD17,(R4)        *R4 ESTA APONTADO P/ULT. PAL. DO
                                     * SETOR
         BEQ     4$                  *DESVIA SE R4 ESTA C/BIT 14
                                     * LIGADO
         MOV      17.,R1             *MOVE  17 P/ R1
         BR      5$                  *DESVIA P/ 5$
4$*      BIT      WORD18,(R4)        *TESTA SE BIT UK  15 DE (R4)
                                     * ESTA LIGADO
         BNE     6$
         HALT
6$*      MOV      18.,R1             *MOVE 18 P/R1
5$*      TST     FLAG
         BEQ     BICR01
         BIS     R0,(R4)
BICR01*  BIC     R0,(R4)
CLRR0*   CLR     R0
         CLR     R2
         TST     R1                  *TESTA SE O PRIMEIRO REG E DE
                                     * OUTRO HA
         BNE     7$                  *DESVIA SE NAO
         RTS     PC
7$*      ADD     TREG,R0             *SOMA TAMANHO DO REGISTRO EM R0
         INC     R2                  *INCREMENTA R2 DE 1
         CMP     R2,R1               *COMP. R2 C/ R1(NO, DO REG. DE
                                     * OUTRO H.A.)
         BLO     7$                  *DESVIA SE MENOR
         RTS     PC
* ********************************************************************
* *** TESTA SE SETOR PERTENCE A LISTA DE ESPACO DISPONIVEL ***
* ********************************************************************
TST5*    CLR     FLAG1.
         BIT      8192., W512        *TESTA I BIT QUE INDICA SE SETOR
                                     *PERTENCE A LISTA AVAIL(1 - NAO
                                     * PERTENCE)
         BEQ     1$
         MOV      1,FLAG1            *SETA FLAG1 SE SETOR NAO
                                     * PERTENCE A AVAIL
         RTS     PC
1$*      JSR     PC,DESLG
```

```
          JSR     PC,INSER
          JSR     PC,GRAVAR
          RTS     PC
*
* ********************************************************************
* ** SUB-ROTINA LOCAL                                                **
* **          PROCURA LUGAR VAZIO NO SETOR,SE ENCONTRAR SETA UM **
*          'FLAG'.CASO NAO ENCONTRE (FLAG ZERO), O SETOR ESTA TO-**
* ** TALMENTE OCUPADO OU O ULTIMO REGISTRO ESTA COM LINKS DA    **
* ** LISTA AVAIL.                                                **
* ********************************************************************
*
LOCAL*    CLR     R2
          CLR     R3
          CLR     FLAG
          MOV     BLK1+4,R0          *MOVE END. DO BUFFER DO MONITOR
                                     * P/ R0
MOVR0*    MOV     R0,R1              *SALVA R0
          INC     R3                 *R3 CONTEM N NO. DO REG. VAZIO
COMPR1*   CMP     (R1)+, 0           *COMPARA REG. C/ZERO BINARIO
          BNE     COMPR3             *DESVIA DE DIFERENTE
          INC     R2                 * INCREMENTA R2 DE 1
          CMP     R2, 10             *COMP. SE CHEGOU A DECIMA
                                     * PALAVRA DO REG
          BLOS    COMPR1             *DESVIA SE R2 MENOR OU IGUAL
          MOV      1,FLAG            *EXISTE LUGAR NO SETOR
          RTS     PC
COMPR3*   CMP     R3,NREG            *TESTA SE E O ULTIMO REGISTRO DO
                                     * SETOR
          BLO     1$
          RTS     PC
1$*       ADD     TREG,R0            *SOMA TAMANHO DO REG. EM
                                     * R0(P/PEGAR
                                     *PROXIMO REGISTRO)
          BR      MOVR0              *DESVIA P/MOVR0
*
* ********************************************************************
* ** SUB-ROTINA INSERE                                               **
* **     A CHAMADA E FEITA APOS TER SIDO LOCALIZADO  ESPACO VA- **
* ** ZIO NO SETOR. ESSA ROTINA FAZ ATRANSFERENCIA DO REGISTRO  **
* ** DO USUARIO PARA O LUGAR DISPONIVEL.                          **
* ********************************************************************
*
INSER*    MOV     4(R5),R1
          CLR     R2
MOVB1*    MOVB    (R1)+,(R0)+
          INC     R2
          CMP     R2,TREG
          BLO     MOVB1
          RTS     PC
```

```
* *** SETA BIT DE REGISTRO QUE PERTENCE A OUTRO SETOR ***
SETHA*  CMP     R3, 18.             *R3 CONTEM O NO. DO REG.
                                    * INSERIDO
        CMP     R3, 18.             *R3 CONTEM O NO. DO REG.
                                    * INSERIDO
                                    *COMPARA SE E IGUAL A 18
        BNE     TST17               *SE DIFERENTE DESVIA P/TST17
        BIS      WORD18, W512       *SETA ULT.BIT DA PAL. 512 DO
                                    * SETOR
        RTS     PC
TST17*  CMP     R3, 17.             *COMPARA R3 COM 17
        BNE     TST16               *SE DIFERENTE DESVIA P/TST16
        BIS      WORD17, W512       *SETA PENULT.BIT DA PAL. 512 DO
                                    * SETOR
                                    *INDICA QUE O REGISTRO 17 E DE
                                    * OUTRO H.A
        RTS     PC
TST16*  MOV      1,CONT1            *MOVE 1 P/ CONT1
        MOV      1,R1               *MOVE 1 P/R1(R1 VAI INDICAR O
                                    * BIT A SER
                                    *LIGADO
CMPREG* CMP     R3,CONT1            *COMP. R3 C/CONT1(R3 CONTEM NO.
                                    * DO REG.
                                    *QUE FOI INSERIDO)
        BEQ     SETBIT              *DESVIA SE IGUAL P/SETBIT
        ASL     R1                  *MULTIPLICA O NO. EM R1 P/ 2
        INC     CONT1               *INCREMENTA CONT1 DE 1
        BR      CMPREG              *DESVIA P/CMPREG
SETBIT* BIS     R1, W510            *SETA O BIT NA PENULTIMA
                                    * PALAVRA,
                                    *QUE INDICA  REG. QUE FOI
                                    * INSERIDO
        MOV      0,FLAG
        RTS     PC

* *** OBTEM UM SETOR DA LISTA DE ESPACO DISPONIVEL ***

AVAIL1* TST     SDISP               *TESTA SE TEM SETOR DISPONIVEL
                                    * NA LISTA
        BNE     2$
        JMP     FIM
2$*     MOV     AVAIL,BLK1+2        *MOVE NO. DO PROX. SETOR A SER
                                    * LIDO
        .BLOCK   LNK1, BLK1         *LER PROX. SETOR (AVAIL)
        .WAIT    LNK1
        JSR     PC,DESLG            *CHAMA ROTINA QUE DESLIGA O
                                    * SETOR DE AVAIL
        CLR      ENDLK1             *ZERA OS APONTADORES DA
        CLR      ENDLK2             * LISTA DE ESPACO DISPONIVEL
        RTS     PC
```

```
*  ************************************************
*  *** DESLIGA SETOR DA LISTA DE ESPACO DISPONIVEL ***
*  ************************************************
DESLG*   MOV      ENDLK1,LNKAV1
         MOV      ENDLK2,LNKAV2
         CMP      BLK1+2,AVAIL        *TESTA SE AVAIL APONTA P/SETOR
                                      * A SER DESLIGADO
         BNE      1$
         MOV      LNKAV2,AVAIL        *ATUALIZA APONTADOR DA LISTA
1$*      MOV      BLK1+2,SALVA3
         JSR      PC,ATHEAD
         MOV      LNKAV1,BLK1+2       *INDICA O SETOR A SER LIDO
                                      * (APONTA P/ANT
         .BLOCK   LNK1, BLK1          *LER SETOR ANTERIOR AO QUE ESTA
                                      * NA
         .WAIT    LNK1                *AREA DE TRABALHO P/ATUALIZAR
                                      * LINK
         MOV      LNKAV2, ENDLK2      *ATUALIZA LINK
         JSR      PC,GRAVAR
         MOV      LNKAV2,BLK1+2       *INDICA SETOR A SER LIDO
                                      * (SEGUINTE)
         .BLOCK   LNK1, BLK1          *LER SETOR SEGUINTE AO QUE ESTA
                                      * NA
         .WAIT    LNK1                *AREA DE TRABALHO P/ATUALIZAR
                                      * LINK DA LI
         MOV      LNKAV1, ENDLK1      *ATUALIZA LINK (COMPLETA DESL.
                                      * DO SETOR)
         JSR      PC,GRAVAR
         MOV      SALVA3,BLK1+2
         .BLOCK   LNK1, BLK1
         .WAIT    LNK1
         BIS      8192., W512         *SETA BIT - SETOR NAO PERTENCE A
                                      * LISTA AVAIL
         RTS      PC                  *FIM DE DESLIGA SETOR
*
*  ************************************************************
*  ** SUB-ROTINA DELET                                       **
*  **      LOCALIZA REGISTRO PREVIAMENTE DELETADO E FAZ A INSER- **
*  ** CAO. A BUSCA CONTINUA ATE O FINAL DO SETRO, E SE NAO MAIS **
*  ** EXISTIR OUTRO DELETADO SERA ZERADO O BIT DE CONTROLE.    **
*  ************************************************************
*
DELET*   CLR      R1                  *ZERA R1 (R1 USADO COMO CONT. DE
                                      * REGS)
         CLR      R0
         MOV      BLK1+4,R2           *MOVE END. DO BUFFER DO MONITOR
                                      * P/R2
         MOV      R2,R3               *SALVA R2
COMP11*  CMP      MASK3,(R3)
         BNE      INC10
```

```
        CLR     R1
        MOV     4(R5),R4
MOVB4*  MOVB     (R4)+,(R3)+
        INC     R1
        CMP     R1,TREG
        BLO     MOVB4
        * TESTA OS REGISTROS SEGUINTES DO SETOR
        * SE AINDA TEM ALGUM REMOVIDO, O BIT DE CONTROLE
                                        * CONTINUA LIGADO
1$*     INC     R0
        CMP     R0,NREG
        BHIS     CLEARD
        CMP      MASK3,(R3)
        BEQ     2$
        ADD     TREG,R3
        BR      1$
2$*     RTS     PC
INC10*  INC     R0
        CMP     R0,NREG
        BHIS    CLEARD
        ADD     TREG,R3             *SOMA TAMANHO DO REG. EM R3
        BR      COMP11              *DESVIA P/COMP11
CLEARD* BIC      DELETA, W512
        RTS     PC
* ***********************************************************************
* *** ROTINA QUE COPIA BUFFER DO MONITOR P/ AREA DE TRABALHO ***
* ***********************************************************************
COPIA*  MOV     BLK1+2,ENDSET       *SALVA NUMERO DO SETOR
        MOV      AREATB,R1          *MOVE END. DA AREA DE TRABALHO
                                    * P/R1
        MOV     BLK1+4,R2           *MOVE END. DO BUFFER DO MONITOR
                                    * P/R2
1$*     MOV     (R2)+,(R1)+         *MOVE BUF DO MONITOR P/AREA DE
                                    * TRAB
        CMP     R1, AREATB+510.
        BLO     1$
        MOV    (R2),(R1)
        RTS     PC
* ************************
* *** ATUALIZA HEADER ***
* ************************
ATHEAD* DEC     SDISP
        MOV      0,BLK1+2           *MOVE ZERO P/BLK1+2
        .BLOCK  LNK1, BLK1          *LER HEADER
        .WAIT   LNK1                *AGUARDA TERMINO DA
                                    * TRANSFERENCIA
        MOV     BLK1+4,R1           *MOVE END. DO BUFFER DO MONITOR
                                    * P/R1
        ADD      10.,R1             *SOMA 10. EM R1 P/PEGAR O END.
                                    * DA PAL.
```

```
                                        *QUE CONTEM AVAIL NO HEADER DO
                                        * ARQUIVO
        MOV     AVAIL,(R1)+             *ATUALIZA AVAIL (CABECA DA
                                        * LISTA)
        MOV     SDISP,(R1)              *ATUALIZA NO. DE SETORES DISP.NA
                                        * LISTA
        JSR     PC,GRAVAR
        RTS     PC
* ***************************************************
* *** COPIA AREA DE TRABALHO P/BUFFER DO MONITOR ***
* ***************************************************
DEVOLV* MOV     BLK1+4,R1               *MOVE END DO BUFFER DO MONITOR
                                        * P/R1
        MOV      AREATB,R4
1$*     MOV     (R4)+,(R1)+             *MOVE  SETOR DA AREA DE TRABALHO
                                        *P/BUFFER DO MONITOR
        CMP     R4, AREATB+510.
        BLO     1$
        MOV     (R4),(R1)
        MOV     ENDSET,BLK1+2           *MOVE NO. DO SETOR DA AREATB
                                        * P/BLK1+2
        RTS     PC                      *RETORNA - TERMINO DA
                                        * TRANSFERENCIA
* ***************************************************
* *** PERCORRE A LISTA ATE O ULTIMO SETOR ***
* ***************************************************
LEPRX*  MOV      W512,R3                *MOVE LINK DO SETOR P/R3
        BIC      61440.,R3              *ZERA OS BITS Q/NAO PERTENCEM AO
                                        * LINK
        MOV     R3,BLK1+2               *INDICA O PROXIMO SETOR A SER
                                        * LIDO
        .BLOCK   LNK1, BLK1             *LER SETOR INDICADO PELO LINK
                                        * (PROX)
        .WAIT    LNK1
        BIT      DELETA, W512           *TESTA SE TEM REG. REMOVIDO
        BEQ     1$
        JSR     PC,DELET                *CHAMA ROTINA P/OCUPAR LUGAR
                                        * REMOVIDO
        JSR     PC,GRAVAR
        JMP     FIMSUB
1$*      BIT      MASK1, W512           *TESTA SE LINK DO SETOR LIDO E
                                        * NULO
        BNE     LEPRX                   *DESVIA SE NAO NULO
        RTS     PC                      *RETORNA SE NULO (CHEGOU AO FIM
                                        * DA LISTA)
* ********************
* *** GRAVA UM SETOR ***
* ********************
GRAVAR* MOV      2,BLK1
        .BLOCK   LNK1, BLK1
```

```
          .WAIT    LNK1
          MOV      4,BLK1
          RTS      PC
*
* ********************************************************************
* **  INFORMA ATRAVES DO CODIGO DE RETORNO QUE HOUVE INSERCAO  **
* ********************************************************************
*
FIMSUB*   MOV      2(R5),R1
          MOV      1,(R1)
          RTS      R5                    *RETORNA AO USUARIO
* ********************************************************************
* *** INFORMA QUE O ARQUIVO ESTA CHEIO ATRAVES DE MENSAGEM ***
* ********************************************************************
FIM*      MOV      MENSG8,R1
          MOV      BUF+6,R2
1$*       MOV      (R1)+,(R2)+
          CMP      R1, MENSG8+12.
          BLO      1$
          MOV    (R1),(R2)
          .WRITE   LNK3, BUF
          .WAIT    LNK3
          MOV      2(R5),R1
          MOV      2,(R1)
          RTS      R5                    *RETORNA AO USUARIO

          .PAGE
          .SBTTL MENSAGENS
MENSG8*   .ASCII /ARQUIVO/
          .BYTE  40
          .ASCII /CHEIO/
          .BYTE  40
          .PAGE
          .SBTTL AREAS DE TRABALHO
*
ENDLK1*   .WORD  0
ENDLK2*   .WORD  0
LNKAV1*   .WORD  0
LNKAV2*   .WORD  0
AUX1*     .BYTE  0
          .EVEN
KEY*      .BYTE  0
          .BYTE  0
BYTE1*    .BYTE  0
BYTE2*    .BYTE  0
CONT*     .WORD  0
ENDSET*   .WORD  0                      *PAL. UTILIZADA P/SALVAR NO. DO
                                        * SETOR
AUX*      .WORD  0
FLAG*     .WORD  0
```

```
WA510*   .WORD   0
WA512*   .WORD   0
W510*    .WORD   0
W512*    .WORD   0
NBLOC*   .WORD   0                    *NO. DE BLOCOS OU SETORES
CONT1*   .WORD   0
SALVA*   .WORD   0
SALVA2*  .WORD   0
SALVA3*  .WORD   0
FLAG1*   .WORD   0
AREATB*  .BLKW 256.                   * AREA DE TRABALHO P/SALVAR BUF
                                      * DO MONITOR
         .END    INSERE

 INSERF.OBJ INSERE.MAC
$FINISH
```

```
$JOB ASSMBL *100,100*
$RUN PIP
 BUSCA.MAC BI*/FA
         .TITLE SUBROTINA DE BUSCA
         .GLOBL BUSCA
         .MCALL .INIT,.OPEN,.READ,.WAIT
         .MCALL .BLOCK,.CLOSE,.RLSE,.EXIT,.WRITE
         .PAGE
         .SBTTL CORPO DO PROGRAMA
*
* *** SUBROTINA DE BUSCA ***

* *** ROTINA DE HASH - DIVISAO

BUSCA*   MOV     4(R5),R2
         CLR     R1
         CLR     CONT1
         CMP     TCHAV, 2
         BHI     1$
         MOV     (R2),R1
         BR      SAI
1$*      MOV     R2,R3              *MOVE END DO INICIO DA CHAVE
                                    * P/R3
         ADD     TCHAV,R3           *SOMA TAMANHO DA CHAVE EM R3
MOV10*   MOVB    -(R3),TOT+1
         MOVB    (R2)+,TOT
         ADD     TOT,R1             *SOMA TOT EM R1 (DOBRA DA CHAVE)
         CMP     R2,R3
         BHI     SAI
         BLO     MOV10              *DESVIA P/MOV10 SE R2 MENOR
         BR      MOV10+4
SAI*     BIC      100000,R1         *TORNAR POSITIVO A CHAVE
         CMP     TARQ,R1            *COMP TAMANHO DO ARQ. C/ A CHAVE
         BHI     2$
         MOV     TARQ,R0
1$*      SUB     R0,R1
         CMP     R0,R1              *COMPARA TAMANHO DO ARQUIVO C/
                                    * CHAVE
         BLOS    1$
2$*      INC     R1
LEBLK*   MOV     R1,BLK1+2
         .BLOCK   LNK1, BLK1        *LER SETOR DO END. GERADO
         .WAIT    LNK1
         MOV     BLK1+4,R1          *MOV END DO BUFM P/ R1
         MOV     R1,SALVA           *SALVA R1
ARGS*    MOV     4(R5),R2
         MOV     TCHAV,R0
1$*      CMPB    (R1)+,(R2)+
         BNE     2$
         DEC     R0                 *DECREMENTA R0 DE 1
```

```
         TST     R0                  *TESTA SE R0 IGUAL A ZERO
         BEQ     ACHOU               *SE IGUAL DESVIA P/ ACHOU
         BR      1$
2$*      MOV     SALVA,R2
*
* *** TESTA SE CHAVE IGUAL A ZERO
         MOV     TCHAV,R0
ZERO*    TSTB    (R2)+
         BNE     TEST3
         DEC     R0
         TST     R0
         BEQ     NACHOU
         BR      ZERO

* *** TESTA SE CHEGOU AO FIM DO SETOR

TEST3*   INC     CONT1
         CMP     CONT1,NREG
         BHIS    TLINK
         ADD     TREG,SALVA
         MOV     SALVA,R1
         BR      ARGS

* *** TESTA SE LINK NULO

TLINK*   CLR     CONT1
         BIT      7777, LINK
         BEQ   NACHOU
         MOV    LINK,R1
         BIC      170000,R1
         BR    LEBLK
*    *** ACHOU REGISTRO / RETORNA AO BUFFER DO USUARIO
ACHOU*   CLR   R4
         MOV     4(R5),R1
         MOV   SALVA,R0
COPIA*   MOVB    (R0)+,(R1)+
         INC     R4
         CMP     R4,TREG
         BLO     COPIA
         MOV     2(R5),R1
         MOV      1,(R1)
         RTS     R5
NACHOU*  MOV     2(R5),R1
         MOV      2,(R1)
         RTS     R5
         .PAGE
         .SBTTL  AREAS DE TRABALHO
TOT*     .BYTE   0,0
TMAX*    .WORD   0
MAX*     .WORD   0
```

```
REGNE*   .WORD   0
SALVA*   .WORD  0
HEAD*    .WORD   0
CONT1*   .WORD   0
         .END    BUSCA
$RUN MACRO
 BUSCA,LP*/NL*TTM BUSCA
$FINISH
```

```
$JOB ASSMBL *100,100*
$RUN PIP
 ALTERA.MAC BI*/FA
        .TITLE SUBROTINA DE ALTERACAO
        .GLOBL ALTERA
        .MCALL .INIT,.OPEN,.READ,.WAIT
        .MCALL .BLOCK,.CLOSE,.RLSE,.EXIT,.WRITE
        .PAGE
        .SBTTL CORPO DO PROGRAMA

* *** ALGORITMO DE ALTERACAO

* *** ROTINA DE HASH - DIVISAO
ALTERA* MOV    4(R5),R2
        CLR    R1
        CLR    CONT1
        CMP    TCHAV, 2
        BHI    1$
        MOV    (R2),R1
        BR     SAI
1$*     MOV    R2,R3
        ADD    TCHAV,R3          *SOMA TAMANHO DA CHAVE EM R3
MOV10*  MOVB   -(R3),TOT+1
        MOVB   (R2)+,TOT
        ADD    TOT,R1            *SOMA TOT EM R1 (DOBRA DA CHAVE)
        CMP    R2,R3
        BHI    SAI
        BLO    MOV10             *DESVIA P/MOV10 SE R2 MENOR
        BR     MOV10+4
SAI*    BIC     100000,R1        *TORNAR POSITIVO A CHAVE
        CMP    TARQ,R1           *COMP TAMANHO DO ARQ. C/ A CHAVE
        BHI    2$
        MOV    TARQ,R0
1$*     SUB    R0,R1
        CMP    R0,R1             *COMPARA TAMANHO DO ARQUIVO C/
                                 * CHAVE
        BLOS   1$
2$*     INC    R1
LERBLK* MOV    R1,BLK1+2
        .BLOCK  LNK1, BLK1       *LER SETOR DO END. GERADO
        .WAIT   LNK1
        MOV    BLK1+4,R1         *MOV END DO BUFM P/ R1
        ADD    LCHAV,R1          *SOMA LOCAL DA CHAVE EM R1
        MOV    R1,SALVA          *SALVA R1
ARGS*   MOV    4(R5),R2
        ADD    LCHAV,R2
        MOV    TCHAV,R0
1$*     CMPB   (R1)+,(R2)+
        BNE    2$
        DEC    R0                *DECREMENTA R0 DE 1
```

```
         TST     R0                  *TESTA SE R0 IGUAL A ZERO
         BEQ     ACHOU               *SE IGUAL DESVIA P/ ACHOU
         BR      1$
2$*      MOV     SALVA,R2

* *** TESTA SE CHAVE IGUAL A ZERO
         MOV     TCHAV,R0
ZERO*    TSTB    (R2)+
         BNE     TEST3
         DEC     R0
         TST     R0
         BEQ     NACHOU
         BR      ZERO

* *** TESTA SE CHEGOU AO FIM DO SETOR

TEST3*   INC     CONT1
         CMP     CONT1,NREG
         BHIS    TLINK
         ADD     TREG,SALVA
         MOV     SALVA,R1
         BR      ARGS

* *** TESTA SE LINK NULO

TLINK*   CLR     CONT1
         BIT      7777, LINK
         BEQ   NACHOU
         MOV    LINK,R1
         BIC      170000,R1
         BR    LEBLK
*    *** ACHOU REGISTRO / RETORNA AO BUFFER DO USUARIO
ACHOU*   CLR   R4
         MOV     4(R5),R1
         MOV   SALVA,R0
         SUB     LCHAV,R0
COPIA*   MOVB    (R1)+,(R0)+
         INC     R4
         CMP     R4,TREG
         BLO     COPIA
          MOV      2,BLK1
         .BLOCK   LNK1, BLK1
         .WAIT    LNK1
         MOV      4,BLK1
         MOV     2(R5),R1
         MOV      1,(R1)
         RTS     R5
NACHOU*  MOV     2(R5),R1
         MOV      2,(R1)
         RTS     R5
```

```
          .PAGE
          .SBTTL  AREAS DE TRABALHO
TOT*      .BYTE   0,0
LINK*     .WORD   0
SALVA*    .WORD   0
HEAD*     .WORD    0
CONT1*    .WORD    0
          .END    ALTERA
$RUN MACRO
 ALTERA,LP*/NL*TTM ALTERA
$FINISH
```

```
$JOB ASSMBL *100,100*
$RUN PIP
 REMOVE.MAC BI*/FA
        .TITLE SUBROTINA DE REMOCAO
        .GLOBL REMOVE
        .MCALL .INIT,.OPEN,.READ,.WAIT
        .MCALL .BLOCK,.CLOSE,.RLSE,.EXIT,.WRITE
        .PAGE
        .SBTTL CORPO DO PROGRAMA

* *** SUBROTINA DE REMOCAO ***

* *** ROTINA DE HASH - DIVISAO
REMOVE* MOV    4(R5),R2
        CLR    R1
        CLR    CONT1
        CMP    TCHAV, 2          *COMPARE TAMANHO DA CHAVE COM 2
        BHI    1$
        MOV    (R2),R1
        BR     SAI
1$*     MOV    R2,R3             *MOVE END DO INICIO DA CHAVE
                                 * P/R3
        ADD    TCHAV,R3          *SOMA TAMANHO DA CHAVE EM R3
MOV10*  MOVB   -(R3),TOT+1
        MOVB   (R2)+,TOT
        ADD    TOT,R1            *SOMA TOT EM R1 (DOBRA DA CHAVE)
        CMP    R2,R3
        BHI    SAI
        BLO    MOV10             *DESVIA P/MOV10 SE R2 MENOR
        BR     MOV10+4
SAI*    BIC     100000,R1        *TORNAR POSITIVO A CHAVE
        CMP    TARQ,R1           *COMP TAMANHO DO ARQ. C/ A CHAVE
        BHI    2$
        MOV    TARQ,R0
1$*     SUB    R0,R1
        CMP    R0,R1             *COMPARA TAMANHO DO ARQUIVO C/
                                 * CHAVE
        BLOS   1$
2$*     INC    R1
LEBLK*  MOV    R1,BLK1+2
        .BLOCK  LNK1, BLK1       *LER SETOR DO END. GERADO
        .WAIT   LNK1
        MOV    BLK1+4,R1         *MOV END DO BUFM P/ R1
        MOV    R1,SALVA          *SALVA R1
ARGS*   MOV    4(R5),R2

        MOV    TCHAV,R0
1$*     CMPB   (R1)+,(R2)+
        BNE    2$
        DEC    R0                *DECREMENTA R0 DE 1
```

```
          TST     R0                  *TESTA SE R0 IGUAL A ZERO
          BEQ     ACHOU               *SE IGUAL DESVIA P/ ACHOU
          BR      1$
2$*       MOV     SALVA,R2

* *** TESTA SE CHAVE IGUAL A ZERO
          MOV     TCHAV,R0
ZERO*     TSTB    (R2)+
          BNE     TEST3
          DEC     R0
          TST     R0
          BEQ     NACHOU
          BR      ZERO

* *** TESTA SE CHEGOU AO FIM DO SETOR

TEST3*    INC     CONT1
          CMP     CONT1,NREG
          BHIS    TLINK
          ADD     TREG,SALVA
          MOV     SALVA,R1
          BR      ARGS

* *** TESTA SE LINK NULO

TLINK*    CLR     CONT1
          BIT      7777, LINK
          BEQ   NACHOU
          MOV    LINK,R1
          BIC      170000,R1
          BR    LEBLK
*    *** ACHOU REGISTRO / RETORNA AO BUFFER DO USUARIO
ACHOU*    CLR   R4
          MOV     4(R5),R1
          MOV   SALVA,R0
COPIA*    MOVB    (R0),(R1)+
          MOVB     255.,(R0)+         *LIGA TODOS OS BITS DO BYTE
          INC     R4
          CMP     R4,TREG
          BLO     COPIA
          BIS      4096., LINK
           MOV      2,BLK1
          .BLOCK   LNK1, BLK1
          .WAIT    LNK1
          MOV      4,BLK1
          MOV     2(R5),R1
          MOV      1,(R1)
          RTS     R5
NACHOU*   MOV     2(R5),R1
          MOV      2,(R1)
```

```
        RTS     R5
        .PAGE
        .SBTTL  AREAS DE TRABALHO
TOT*    .BYTE   0,0
LINK*   .WORD   0
SALVA*  .WORD   0
HEAD*   .WORD   0
CONT1*  .WORD   0
        .END    REMOVE
$RUN MACRO
 REMOVE,LP*/NL*TTM REMOVE
$FINISH
```

```
$JOB ASMBL *100,100*
$RUN PIP
 FECHA.MAC BI*/FA
          .MCALL .BIN2D,.WRITE,.WAIT
          .MCALL .CLOSE,.RLSE
          .GLOBL FECHA
FECHA*    .CLOSE  LNK1
          .CLOSE  LNK3
          .RLSE   LNK1
          .RLSE   LNK3
          RTS     R5
          .END    FECHA
$RUN MACRO
 FECHA,LP*/NL*TTM FECHA
$FINISH
```