UM ESTUDO SOBRE A IMPLEMENTAÇÃO DE ÍNDICES ATRAVÉS DE ÁRVORES N-ÁRIAS

Elisabeth Fátima Torres

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

Aprovada por:

Prof. JANO MOREIRA DE SOUZA

Profa. SUELI MENDES DOS SANTOS

Prof. JAYME SZWARCFITER

Prof. LUIZ A. C. DA CUNHA COUCEIRO

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 1981

TORRES, ELISABETH FÁTIMA

Um Estudo sobre a Implementação de Índices Através de Árvores N-Árias (Rio de Janeiro) 1981

VII, 194 p. 29,7cm (COPPE-UFRJ, M.Sc. Engenharia de Sistemas e Computação, 1981)

Tese - Univ. Fed. Rio de Janeiro. Faculdade de Engenharia.

1. Banco de Dados I. COPPE/UFRJ II. Um Estudo Sobre Implementação de Índices Através de Árvores N-Árias.

À minha mãe

AGRADECIMENTOS

Ao meu orientador, JANO MOREIRA DE SOUZA, pelos ensinamentos e atenção; ao acadêmico HÉLIO MARÇOLA JUNIOR, pela colaboração na programação; ao EDSON SHIGUERU TAKEDA, pelo cuidado com a datilografia; ao diretor do Núcleo de Processamento de Dados da Universidade Estadual de Maringá, DILVO PAUPTIZ, por ter facilitado o uso do computador.

À minha família e amigos pelo apoio moral e compreensão.

Aos professores do Programa de Engenharia de Sistemas e Computação da COPPE-UFRJ, pela formação aqui adquirida.

Ao CNPQ - Conselho Nacional de Desenvolvimento Científico e Tecnológico, à CAPES - Coordenação Nacional de Aperfeiçoamento do Pessoal do Ensino Superior e à FINEP - Financiadora de Projetos e Pesquisas, pelas bolsas recebidas durante o curso.

RESUMO

Consiste num estudo, sobre a organização e estruturação de índices para acesso aleatório e sequencial a arquivos residentes em discos magnéticos ou discos magnéticos flexíveis e, na escolha e definição de uma estrutura para a implementação. A estrutura escolhida é uma variação da árvore B e permite ao usuário construir e acessar índices com chaves numéricas e alfanuméricas, de tamanho constante ou variável, através das operações de busca, inserção, remoção, busca ao próximo, busca ao anterior e busca por posição.

A implementação foi feita num computador IBM 1130, sendo usada para a programação a linguagem FORTRAN (salvo subrotinas auxiliares de manipulação de bytes, que estão em Assembler), apresentando uma boa portabilidade.

Preparou-se um manual para orientação aos usuários do sistema.

ABSTRACT

This work involves the study concerning the organization and structure of indexes for the randomic access to the existing files in magnetic disks, and in the choice of a definition about a structure for implementation. The chosen structure is a variation of a B tree and permits the user to build up and to have access to indexes with numerical and alphanumeric Keys, of a constant and variable size, through retrieval, insertion, deletion, search for the next, search for the previous and search for position.

An implementation was made in a IBM 1130 computer being used for the program dealing with the FORTRAN language (with the exception of auxiliary subrotines of manipulation of bytes, which are in assembler), offering a good portability.

It prepares a manual for the guidance of those that use the system.

ÍNDICE

Pág.

I - INTRODUÇÃO

O trabalho desenvolvido trata do problema do projeto de um sistema para organização e utilização de índices para acesso aleatório e sequencial a arquivos residentes em discos magnéticos ou discos magnéticos flexíveis. O nosso trabalho consistiu na pesquisa das referências bibliográficas sobre o assunto, no estudo da revisão da literatura, sistematização da nomenclatura usada para as árvores n-árias, definição de um projeto de estruturação de índices que aceitasse o uso de chaves alfanuméticas, de tamanho constante e variável, e na implementação dessa estrutura de índice.

A revisão da literatura começou pelo estudo do artigo "Organization and Maintenance of Large Ordered Indexes" de BAYER e McCREIGHT[02], onde é apresentado o conceito de árvore B. Continuando com os estudos de revisão da literatura, encontramos variações surgidas em torno do conceito de árvore B, inclusive havendo multiplicidade nos nomes atribuidos a essas, e dados comparativos entre os desempenhos de implementações feitas com variações diversas.

Nesta implementação usamos a árvore apresentada neste trabalho como sendo árvore B^+ com prefixo simples. A estrutura desenvolvida, além dos procedimentos básicos de busca, inserção e remoção, oferece outras facilidades ao usuário como a possibilidade de ser feita inicialmente uma carga maciça do índice, em vez da inserção um a um, e a busca ao anterior, a busca ao próximo e a busca por posição, sendo também possível a construção de mais de um índice para um mesmo arquivo.

O objetivo foi possibilitar aos usuários da linguagem FORTRAN, que apresenta tão poucas opções para o trabalho com arquivos, as facilidades oferecidas pela utilização de índices associados aos arquivos. O sistema todo, salvo subrotinas auxiliares de manipulação de bytes que estão em assembler, foi desenvolvido com a linguagem FORTRAN do IBM 1130, o que acarreta uma boa portabilidade.

Consideramos o desenvolvimento deste trabalho importante por ser de utilidade prática a outros usuários. Particularmente, tem importância pela própria concretização da nossa proposta de trabalho e o consequente aperfeiçoamento adquirido com o seu desenvolvimento.

Na seção II, descreveremos as principais soluções encontradas na literatura, já utilizadas na prática ou em estudo. Na seção III apresentamos o sistema implementado, com todos os procedimentos desenvolvidos e os algoritmos correspondentes, que são apresentados usando-se uma pseudo-linguagem algorítmica em português. Na seção IV apresentamos as conclusões; na V as referências bibliográficas e na VI temos quatro anexos: no primeiro, apresentamos algumas características do computador em que foi feita a implementação, um IBM 1130 alocado na Universidade Estadual de Maringá, no Paraná; no segundo, temos a maneira de determinar corretamente o K, um dos parâmetros característicos de uma árvore B; no terceiro anexo temos as listagens do sistema e como quarto anexo temos o Manual do Usuário com orientações quanto ao uso do sistema e onde apresentamos programas exemplificando sua utilização.

II - REVISÃO DA LITERATURA

. 1 - Justificativas

. 2 - Tipos de índices

.. 1 - quanto à estabilidade da sua estrutura

.. 2 - quanto à localização das informações

. 3 - Índices estáticos - ISAM

.. 1 - definição

.. 2 - adequação à estrutura física do dispositivo de armazenamento

.. 3 - área de extensão

. 4 - Índices dinâmicos

.. 1 - VSAM

... 1 - definição

... 2 - organização do índice

... 3 - operações

.... 1 - busca

.... 2 - inserção

.... 3 - remoção

.. 2 - árvore B

... 1 - definição

... 2 - operações permitidas

.... 1 - busca

.... 2 - busca o próximo

.... 3 - inserção

.... 4 - remoção

... 3 - custo das operações

.... 1 - custo da busca

.... 2 - custo da busca ao próximo

.... 3 - custo da inserção

.... 4 - custo da remoção

II - 1 JUSTIFICATIVAS

A utilização de grandes arquivos em processamento onde se necessita de acesso aleatório exige uma estrutura de índice associada ao arquivo. O conceito de índice aqui é o mesmo usado em linguagem comum: consiste de um conjunto ordenado de informações que facilitam a busca a um conjunto maior de informações associadas às primeiras.

Dentro dos registros de um arquivo, organizado de maneira não sequencial, há informações que por si só, ou combinadas a outras, podem identificar univocamente os registros. Esses identificadores de registro são conhecidos como a sua chave e é possível identificar o registro r_i uma vez conhecida sua chave K_i. Em geral chaves obedecem a alguma relação de ordem o que permite definir-se uma sequência lógica entre elas, no arquivo, e é essa sequência lógica que permite a estruturação do índice.

Um índice pode estar fisicamente integrado ao arquivo ou à parte, e por si só o índice já é um arquivo. Se o índice for muito extenso poder-se-á ter um índice de nível superior associado a ele.

Embora os índices tenham sido usados em computadores praticamente desde o começo, é com o desenvolvimento dos dispositivos de disco magnético, que permitem o acesso direto ao dado, que a sua utilização é acentuada.

Os índices extensos são mantidos em dispositivos de memória auxiliar e por isso o tempo gasto com o seu processamento é importante na avaliação dos custos. O tempo requerido para acessar o dispositivo secundário de armazenamento é o principal fator considerado nessa análise de custos, outros são: o tempo necessário para processar um dado uma vez colocado na memória, a ocupação de espaço no dispositivo de

armazenamento e a razão entre o espaço requerido pelo índice e o espaço requerido pelas informações associadas (os registros completos).

Os índices geralmente apresentam alguma ou todas das seguintes características:

1) habilidade para acessar diretamente um registro através de sua chave,
2) habilidade para processar sequencialmente os registros do arquivo pela sequência das chaves,
3) habilidade para inserir ou remover registros; em particular, permitir o desenvolvimento do arquivo num determinado sentido.

A estrutura básica de representação dos índices considera-os como um conjunto de chaves K_i, i=1, ..., n (n = tamanho do arquivo) e um conjunto de apontadores p_j, j = 0, ... n (j ε N).

II - 2 TIPOS DE ÍNDICES

1. Quanto à estabilidade de sua estrutura os índices são classificados por HELD e STONEBRAKER[07] como estáticos ou dinâmicos.

Um índice é dito dinâmico quando está sujeito à reorganização de sua estrutura sempre que uma operação de inserção ou remoção é feita, podendo haver alterações em todos os seus níveis e até do próprio número de níveis. Nesse tipo de índice, a flexibilidade é conseguida através de uma grande utilização de ponteiros.

Ex. de índice desse tipo: implementações com árvore-B (a ser vista com detalhes na seção II-4-2)

Outro problema decorrente do uso de índices dinâmicos, além do cuidado exigido para a sua frequente reestruturação, diz respeito ao uso de um mesmo arquivo por processos concorrentes.

Neste caso precauções devem ser tomadas para que não ocorram interferências entre os diversos processos.

Índices estáticos são estruturas de armazenamento rígidas, nas quais não são permitidas reorganizações estruturais significativas durante o processamento. Como exemplo temos o ISAM,da IBM ou da Digital, (a ser visto com detalhes na seção II-3). No ISAM da IBM, os registros inseridos após a criação do arquivo só interferem na disposição física dos demais registros da mesma trilha, podendo fazer com que o registro de maior valor seja colocado numa área de extensão à parte, chamada área de "overflow".

2. Quanto à localização das informações

Uma das maneiras de implementar estruturas de índices é deixar as informações, (ou apontadores para as informações), no nível de mais baixa ordem (folha) e usar os níveis superiores simplesmente para ordenação do índice. Para implementações desse tipo pode-se usar outros elementos que não as chaves, para a parte superior, o que em implementações mais sofisticadas pode ser feito com compressão, de chaves e apontadores.

A parte formada pelos níveis superiores é muitas vezes denominada na literatura por índice (ou "index set") e a formada pelos elementos do nível inferior por arquivo (ou "sequence set").

Frequentemente, um arquivo para o qual se deve construir um índice não está fisicamente ordenado conforme a sequência das chaves usadas para o índice. Neste caso, cada chave do arquivo deve também aparecer no índice. Estes índices são conhecidos como índices densos ou secundários, conforme BAYER e UNTERAUER[04]. Caso contrário, se nem todas as chaves precisam

participar do índice ele é dito não denso.

É interessante notar que um arquivo só pode ter um índice não denso, visto que não é possível ordená-lo fisicamente segundo mais de uma relação de ordem.

Se a condição para as chaves participarem do índice for outra que não a sua localização pode-se classificar os índices como seletivos ou exaustivos. No índice seletivo somente algumas das chaves devem participar do índice, baseado em algum atributo, geralmente outro que não a localização. No índice exaustivo, todas as chaves devem participar do índice baseado no atributo indicado.

Um índice exaustivo é um índice denso e eventualmente, conforme o atributo testado para o arquivo, um índice seletivo pode ser denso.

II - 3 ÍNDICES ESTÁTICOS - ISAM

.. 1 - Definição

Diversas implementações de arquivos do tipo Sequencial Indexado optaram por uma estrutura estática, entre elas temos o ISAM (Indexed Sequential Access Method).

Um arquivo ISAM é um arquivo sequencial ao qual se associa um índice formado por uma coleção de pares, cada um deles associando um valor da chave de acesso a um endereço do arquivo.

A característica de sequencialidade física do arquivo é aproveitada para a definição de blocos aos quais é associada uma única entrada no índice, ao invés de uma para cada registro individual. Assim, sendo N o número de registros e m o número de blocos, o índice possuirá em seu último nível apenas m entradas, em vez de N.

É empregado em arquivos que estão sujeitos a um grande nú-

mero de acessos aleatórios, bem como a frequentes processamentos sequenciais. Neste caso há necessidade de utilização de uma estrutura de acesso, associada ao arquivo, que ofereça maior eficiência na localização de um registro, identificado por uma chave, que os métodos vistos para arquivos sequenciais.

A figura (2.1) apresenta um exemplo de arquivo do tipo sequencial indexado estático onde podemos observar a existência de uma área de extensão usada para evitar a reorganização da estrutura do arquivo quando, em uma operação de inserção de um registro, a trilha correspondente já não tiver espaços livres. No caso de operações de remoção é suficiente assinalar os registros removidos com alguma marca especial, de forma a que eles não sejam considerados no processamento do arquivo.

.. 2 - Adequação à estrutura física do dispositivo de armazenamento.

Para a definição dos níveis de um índice ISAM procura-se adequá-lo aos níveis físicos dos meios de armazenamento, procurando obter uma maior eficiência, como no caso de discos magnéticos onde podemos considerar os níveis de disco, cilindro e trilha.

O índice de cilindros contém, para cada cilindro ocupado pelo arquivo, uma entrada, que indica o maior valor da chave de ordenação nele contido. A cada cilindro do arquivo é associado um índice de trilhas que indica, para cada trilha do cilindro, o valor da maior chave nela contido. A cada trilha do cilindro é associado um par de entradas no índice de trilhas, ficando aí registrado o valor da maior chave presente na trilha e o valor da maior chave lógicamente pertencente a esta mesma tri

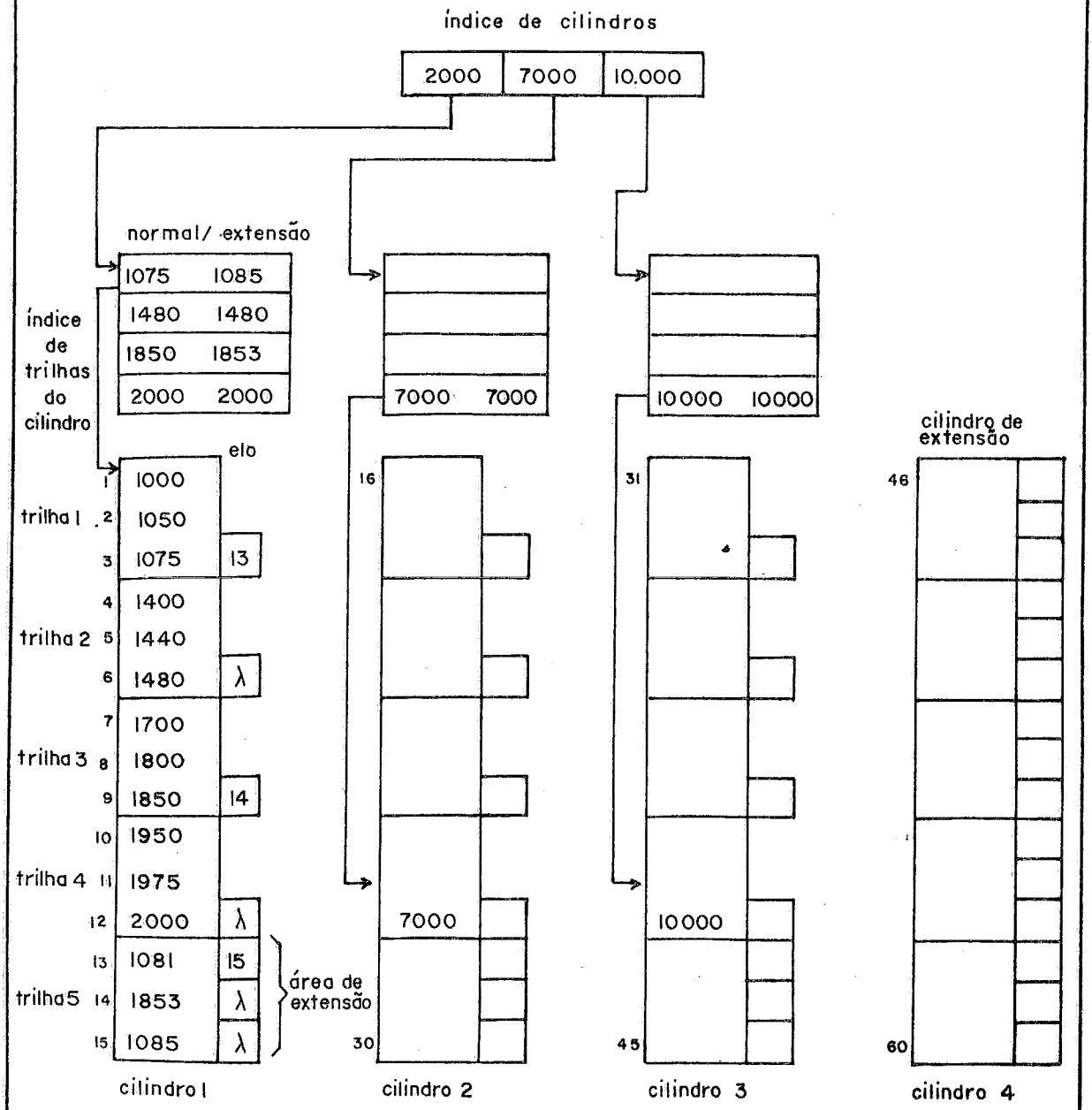


FIGURA 2.1 - ESTRUTURA DO ÍNDICE DE UM ARQUIVO **ISAM**

lha, presente na área de extensão. Se a área de extensão não for usada pela trilha as duas entradas registram o mesmo valor.

O mais alto nível do índice, acima do índice de cilindros, é chamado "índice mestre" e contém para cada disco (ou grupo de cilindros ocupado pelo arquivo) a identificação da sua chave de maior valor. Seu uso é opcional.

O índice mestre e o de cilindros, sempre que possível são mantidos residentes na memória principal, durante o processamento, de modo que durante o processo de pesquisa é determinado em que cilindro está o registro procurado sem necessidade de acessar o disco. Uma vez determinado o cilindro, este é acessado para a leitura do seu índice de trilhas, o qual permite a localização da trilha onde se encontra o registro. A trilha selecionada é, então, lida e dentro dela localizado, por meio de algum método de pesquisa, o registro desejado . Neste processo, apenas um cilindro é acessado.

Ressalte-se que não há necessidade de cada entrada no índice de trilhas conter o endereço da trilha, pois este é determinado pela sequência na qual aparece a entrada dentro do índice. O mesmo se aplica ao índice de cilindros.

.. 3 - Área de extensão

A área de extensão (também conhecida como área de "overflow") destina-se a possibilitar a inserção de novos registros quando já não houver espaços livres na trilha adequada. Ela é uma extensão da área principal de dados do arquivo.

Uma das soluções para o problema da manutenção da estrutura, quando são feitas inserções, é a adotada pela IBM. Dentro da trilha os registros devem manter uma sequência física segundo os valores de suas chaves o que exige que sempre haja

um deslocamento a cada inserção. Se a trilha já estiver cheia, isto é, sem espaços livres, o registro com a maior chave passa para a área de extensão. Para se determinar em qual trilha deverá ser feita a inserção procura-se no índice de trilhas , considerando tanto a entrada normal como a de "overflow", a menor das chaves superiores à que vai ser inserida. Se a chave do registro a ser iserida é maior que todas as do índice de trilhas então o arquivo deverá sofrer uma extensão.

No caso das implementações feitas com o uso de discos magnéticos onde os níveis do arquivo estão associados a trilhas, cilindros e discos há duas opções quanto ao uso das áreas de extensão: uma área de extensão junto ao cilindro e uma área de extensão independente.

A área de extensão junto ao cilindro é de uso comum às trilhas desse cilindro e evita que sejam feitos acessos extras quando se constata que o registro procurado deve estar na área de extensão. O inconveniente que apresenta é a reserva de espaço feita em cada cilindro, que pode ficar desperdiçada , caso não haja inserções suficientes.

No caso da área de extensão independente, cilindros especiais são reservados para ela. O seu uso minimiza os espaços necessários para a extensão do arquivo, mas, ocasiona buscas extras, isto é, acessos adicionais a outros cilindros.

Melhores detalhes podem ser encontradas na referência número onze (11) às paginas 170-189.

II - 4 - ÍNDICES DINÂMICOS

.. 1 - VSAM

... 1 - Definição

O Virtual Storage Access Method (VSAM) da IBM, foi desenvolvido para ser usado em sistemas operacionais com memória virtual. Êle surgiu da necessidade de haver um método de acesso que possibilitasse tanto o acesso direto através de uma chave de busca quanto o processamento sequencial segundo uma ordem definida entre as chaves, sem degradação do desempenho quando houvesse um grande número de inserções ou remoções. Os métodos de acesso do tipo sequencial indexado estático geralmente conseguem atender a essas duas formas de acesso mas o desempenho cai progressivamente à medida em que se realizam inser - ções ou remoções.

O VSAM foi planejado de maneira a evitar essa degradação. A estrutura apresenta dois conceitos lógicos , associados ao controle do espaço ocupado pelos dados , muito importantes: área de controle (CA) e intervalo de controle (CINV).

Uma área de controle é o espaço que contém um conjunto de intervalos de controle contíguos. A ordem dentro da CA não é necessariamente física. As CA's estão geralmente associadas a cilindros em discos magnéticos.

Um intervalo de controle é formado por um conjunto de registros armazenados em áreas contíguas. Os registros de um intervalo de controle são mantidos em sequência física segundo uma chave de ordenação. No caso de trabalhar-se com registros de tamanho variável o CINV deve ter informações sobre o endereço de início de cada regis

tro. O CINV é a unidade de informação que é transferida entre a memória virtual e a memória auxiliar. O tamanho do CINV é definido no momento da criação do arquivo.

... 2 - Organização do índice

Um índice é usado para endereçar os registros contidos nas áreas de controle e intervalos de controle. Pode ser estruturado com mais de um nível, cada um dos quais é um conjunto de informações cujas entradas dão a localização das informações no nível imediatamente abaixo. O nível mais baixo do índice é conhecido como "sequence set" : êle dá a localização dos CINV's. Os registros dos níveis superiores são coletivamente chamados de "index set". No nível mais alto haverá sempre um único registro. No caso de arquivos muito pequenos o índice pode ter apenas um nível, o do próprio "sequence set".

A figura (2.2) ilustra os níveis do índice e a formação das áreas de controle. Uma entrada no "index set" contém o valor máximo que pode aparecer no seu nível imediatamente inferior e um ponteiro para o primeiro valor do conjunto ao qual está associado. Uma entrada no "sequence set" contém a chave de maior valor do CINV, acompanhada de um ponteiro para o CINV. Cada registro do "sequence set" pode conter mais de uma entrada e o conjunto de CINV apontado por entradas de um mesmo registro do "sequence set" formam uma área de controle (CA), isto é, o número de entradas existentes em cada registro do "sequence set" corresponde ao número de CINV's existentes na CA. O "sequence set" possui ponteiros horizontais para permitir o processamento sequencial.

Cada elemento do "sequence set" está associado a

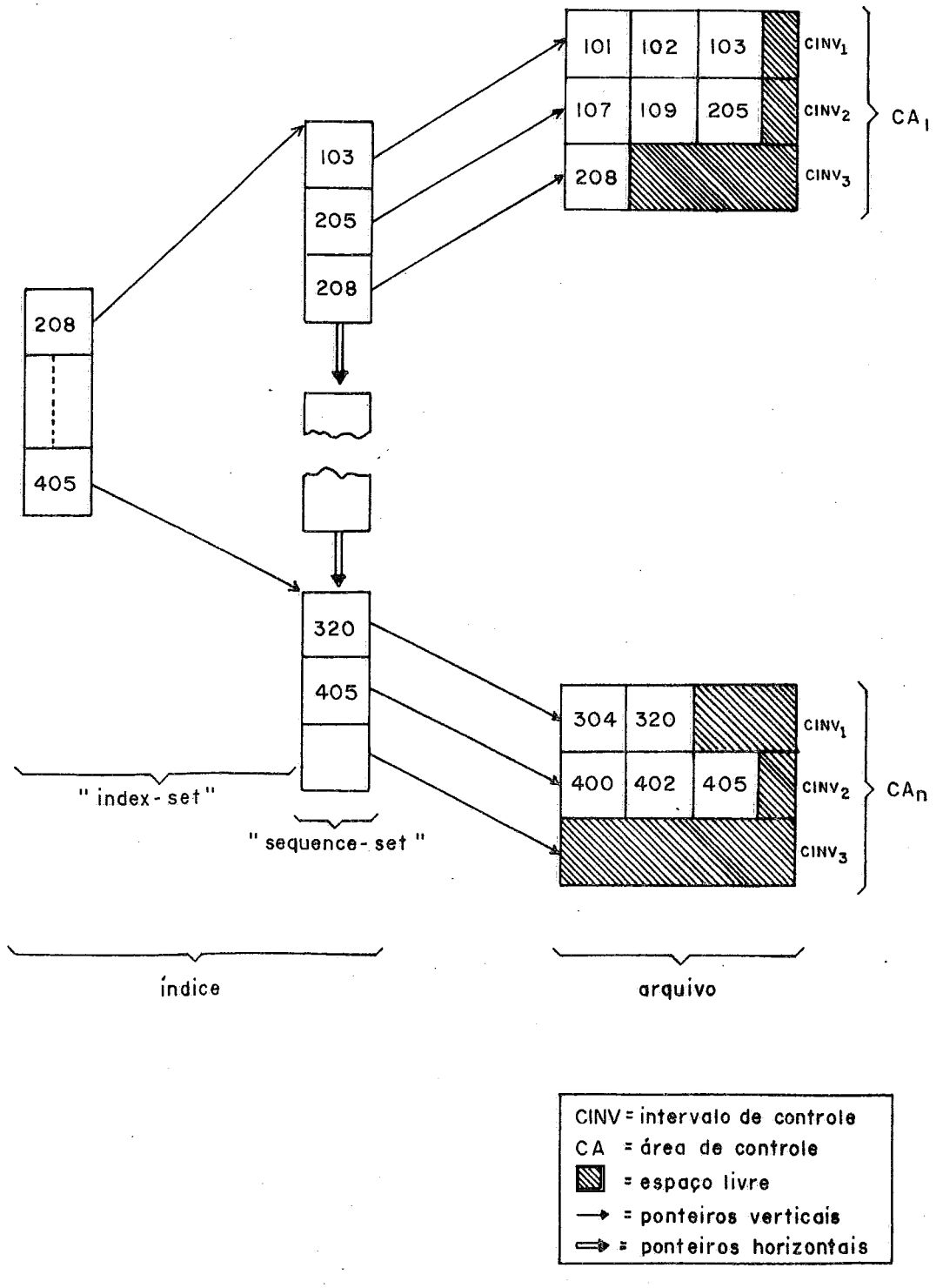


FIGURA 2.2 - ESTRUTURA DO ÍNDICE DE UM ARQUIVO **VSAM**, ORGANIZADO SEGUNDO A SEQUÊNCIA DAS CHAVES.

uma área de controle, que é geralmente armazenada em um cilindro. Desta forma, se os nós do "sequence set" forem alocados junto aos próprios cilindros, os dados podem ser acessados sem acessos extras ao disco.

Outra característica do VSAM que aumenta o seu desempenho é o uso de técnicas de compressão. Tanto os ponteiros quanto as chaves são compactados. Ressalte-se que a compressão implica em maior tempo de UCP.

... 3 - Operações

.... 1 - Busca

Um registro é localizado através do índice usando-se comparações do tipo menor ou igual em cada nível, até encontrar o CINV correspondente. Opcionalmente o "Sequence set" e até níveis mais altos do índice podem ser armazenados junto às trilhas visando diminuir o tempo de acesso.

A busca ao próximo é simples devido aos ponteiros horizontais do "sequence set".

.... 2 - Inserção

Quando o arquivo tipo VSAM é criado são deixados alguns espaços livres para futuras adições dentro dos próprios CINV's e também CINV's são deixados livres. É definida uma percentagem de ocupações para as CA's e os CINV's. Quando as inserções começam a ser feitas, obedecem ao seguinte procedimento:

1 - Localiza-se o CINV no qual o registro deve ser inserido.

2 - Se há espaço no CINV a inserção é feita, mantendo-se a ordem entre os registros. Não há ne

cessidade de atualização dos níveis superiores do índice.

3 - Se não há espaço no CINV verifica-se se há algum CINV livre dentro daquela CA.

4 - Se há CINV livre então:

a) Os registros do CINV no qual deveria ser feita a inserção são distribuídos meio a meio com o CINV livre, inclusive o novo registro, mantendo-se a ordem.

b) Atualiza-se o "sequence set" dessa CA inserindo-se uma nova entrada e atualizando a correspondente ao CINV assinalado no início do procedimento.

5 - Se não há CINV livre na CA então:

a) Aloca-se uma nova CA e um novo nó para o "Sequence set".

b) Distribui-se meio a meio os intervalos de controle com a nova CA. Atualiza-se o "se - quence set".

c) Insere-se uma nova entrada no nível do índice superior ao "sequence set" para o novo nó deste.

d) Insere-se o registro segundo o passo 4.

.... 3 - Remoção

Quando registros de um arquivo VSAM, organizado segundo a sequência das chaves, são removidos ou reduzidos, o espaço liberado passa a ser considerado como disponível. Para que os espaços livres sejam contíguos alguns registros do intervalo de controle sofrem deslocamentos, podendo inclusive

um CINV ficar totalmente livre.

Melhores detalhes podem ser encontrados na referência onze (11) às páginas 190 a 234.

II - 4 - 2 - ÁRVORE B

... 1 - Definição Original

Em 1970 Rudolf BAYER e Edward McCREIGHT desenvolveram uma estrutura para organização e manutenção de índices externos , para arquivos com registros de chave única, com custos de operação relativamente baixos, batizando a estrutura usada como Árvore B. A estrutura desenvolvida permite processamento sequencial (seguindo a sequência entre as chaves) e direto (pela procura do registro correspondente à chave dada).

O conceito de árvore-B foi posteriormente trabalhado por outros autores, surgindo algumas variações e extensões da idéia original. Não houve um consenso entre os autores quanto aos nomes dados às variações e extensões do conceito. Neste trabalho vamos adotar a nomenclatura usada por COMER[06] , que também constatou a multiplicidade de nomes usados na literatura, por ser a mais completa e recente, com algumas modificações para aumentar a clareza.

Def.: Seja $h \geq 0$ um inteiro, k um número natural.

Uma árvore T está na classe $\tau(k,h)$ de árvores B se T é vazia (h=0) ou tem as seguintes propriedades:

i) Cada caminho da raiz até qualquer folha tem o mesmo comprimento h, chamado altura da árvore, isto é, h = número de nós do caminho.

ii) Cada nó exceto as folhas e a raiz tem no mínimo k+1 filhos. A raiz ou é uma folha ou têm no mínimo 2 filhos.

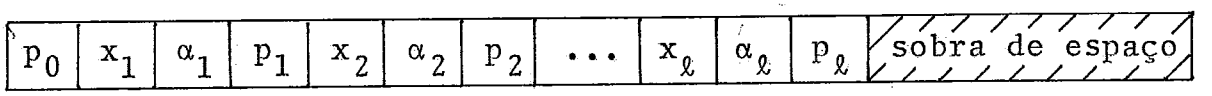
iii) Cada nó tem no máximo 2k+1 filhos.

Esta é a definição original apresentada por BAYER e McCREIGHT[02]. Posteriormente, KNUTH[14] apresentou outra definição (1) aceitando que o fator de ocupação máximo dos nós seja um número ímpar m-1.

Os nós são usualmente chamados de páginas.

Dois parâmetros determinam a árvore B, isto é, h, a altura da árvore e 2k, o tamanho do nó, onde k é dependente do dispositivo usado para armazenamento do índice (características físicas do dispositivo). A ordem da árvore é dada por k.

Lógicamente um nó é organizado da forma como mostra a figura (2.3).



p_j = ponteiros $\qquad k \leq \ell \leq 2k$

x_i, α_i = dupla de informação associada (chave, apontador de informação)

Figura 2.3 - Organização de um nó

Cada nó, exceto a raiz, tem sempre ocupação mínima de 50%, isto é, contém k duplas (x_i, α_i).

Uma árvore B com k=1, h=3 é mostrada na figura (2.4). Os números naturais representam as chaves. A parte de dados foi omitida.

---

(1) Uma árvore B de ordem m é uma árvore que satisfaz às seguintes propriedades:
- i) Cada nó tem no máximo m filhos.
- ii) Cada nó, exceto a raiz e as folhas, tem no mínimo m/2 filhos.
- iii) A raiz tem no mínimo 2 filhos (salvo quando é folha)
- iv) Todas as folhas aparecem no mesmo nível e não contém informação.
- v) Um nó não folha com k filhos contém k-1 chaves.

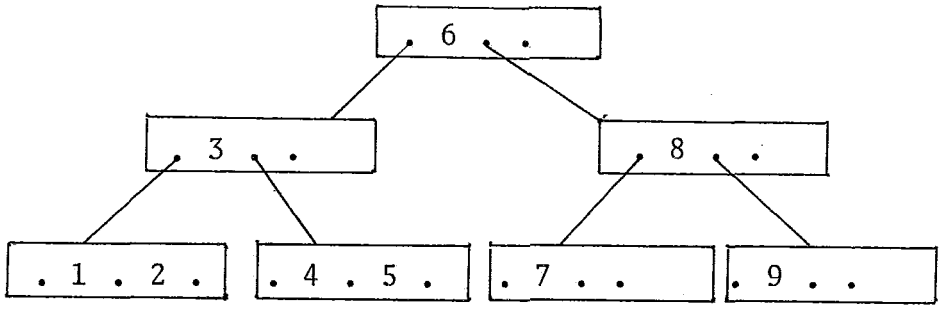


Figura 2.4

Os ponteiros nos nós folhas são dispensáveis e foram introduzidos para dar um formato comum a todos os nós.

A organização de índices tipo árvore B permite a busca , inserção e remoção de chaves em tempo proporcional a $\log_{k+1} I$, onde I é o tamanho do índice (número total de registros) e k um número natural, dependente do dispositivo, tal que o desempenho aproxime-se do ótimo.

II - 4 - 2 - 2 - OPERAÇÕES PERMITIDAS

A estrutura da árvore B é tal que os métodos de inserção e remoção de chaves sempre a mantêm balanceada.

.... 1 - Busca

B1 - Começa-se lendo a página raiz.

B2 - A chave procurada está na página.

Fim do processo com sucesso.

B3 - A chave não está na página.

Se a página é uma folha, então o processo termina com o insucesso.

B4 - A página é um nó interno e a chave não está presente nela:

a) A chave é menor que a primeira chave da página. O processo continua na página indicada pelo ponteiro esquerdo à primeira chave da página.

b) A chave é maior que a última chave da página. O processo continua na página indicada pelo ponteiro direito à última chave da página.

c) A chave tem valor intermediário a duas chaves contíguas da página. O processo continua na página indicada pelo ponteiro que se encontra entre as duas chaves consideradas.

Tornar a B2.

.... 2 - Busca o próximo

Para acessar-se a chave logicamente seguinte a uma dada chave é necessário que o algoritmo de busca assinale a página em que houver o sucesso da operação. Pode haver necessidade de trabalhar com as páginas dos níveis superiores e a melhor solução é ter registrado o caminho percorrido na etapa da busca; este pode estar armazenado, por exemplo, numa pilha.

O algoritmo de "busca o próximo" começa então a partir da página assinalada, havendo três possibilidades:

a) O algoritmo de busca indica um valor não extremo direito de uma folha. Então o próximo é o elemento que o sucede na própria folha.

b) O algoritmo de busca indica um valor extremo direito de uma folha. Se na página mãe há um valor à direita do ponteiro que leva à página assinalada este é o próximo, caso contrário assinala-se a página mãe e repete-se o processo até ser encontrado um valor à

direita ou encontrar-se a raiz e não haver valores à direita.

c) O algoritmo de busca indica um valor de uma página não folha. Então o próximo é o primeiro elemento da folha extrema esquerda da subárvore gerada pelo ponteiro direito da chave.

Podemos adotar procedimento semelhante para "obter o anterior".

... 3 - Inserção

I_1 - Busca-se a chave que se deseja inserir. Esse procedimento vai indicar qual a folha em que deverá ser feita a inserção.

I_2 - Há vaga na folha indicada para a inserção. A chave é inserida de forma a manter a ordem existente. Os níveis superiores da árvore não são afetados.

I_3 - Não há vaga na folha indicada (já existem 2k chaves). Procede-se a uma divisão de página que obedece ao seguinte procedimento:

a) Aloca-se uma nova página e distribui-se o conteúdo da folha assinalada meio a meio com a nova página. A chave fica na página adequada.

b) A chave de valor intermediário entre as páginas consideradas é deslocada para o nível superior.

c) O processo de inserção repete-se agora no nível superior com a chave que ascendeu.

I_4 - A chave que ascendeu deve ser inserida na raiz

e esta já está cheia. A divisão da raiz ocorre e aloca-se uma nova página para receber essa chave. Esta nova página passa a ser a raiz. Outra maneira de se fazer inserção é usando a técnica de OVERFLOW, que será vista na seção II-4-2-4-1. A figura (2.5) ilustra a inserção com divisão de página.

.... 4 - Remoção

R1 - Busca-se a chave a ser removida.

R2 - A chave a ser removida encontra-se numa página não folha. Troca-se a chave com o seu sucessor imediato, com isso a chave vai parar numa folha e o processo de remoção continua.

R3 - A chave a ser removida encontra-se numa folha. Então remove-se a chave.

R4 - Se a página está com mais de k chaves o processo está concluído, senão verifica-se o número de chaves da página vizinha e prossegue-se com a técnica de concatenação de páginas ou redistribuição de chaves ("underflow").

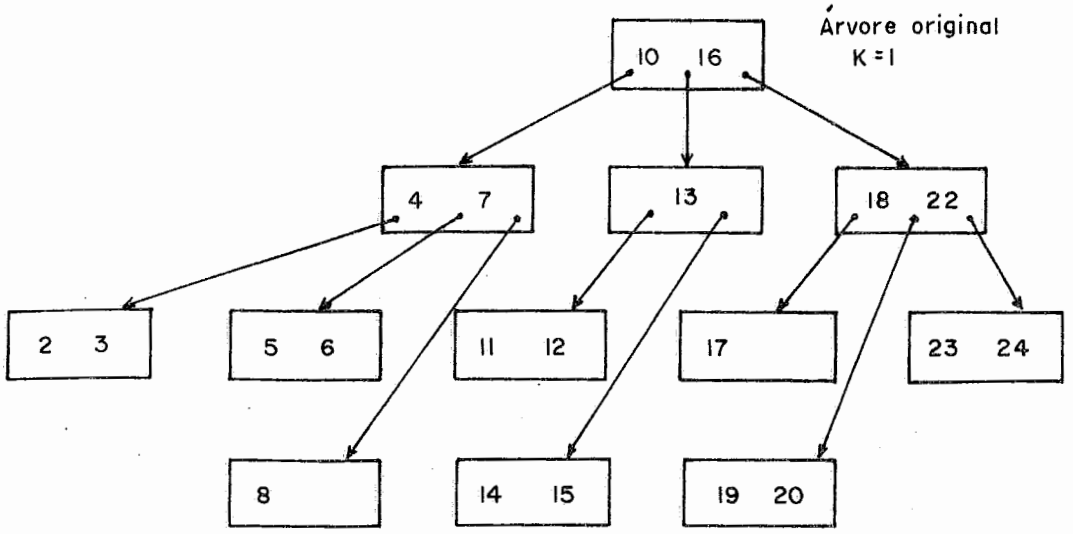
Concatenação:

Empregada quando a soma com o número de chaves da página irmã vizinha é menor que 2k.

C_1 - Junta-se as chaves das duas páginas, e a chave separadora entre ambas na página mãe, numa única página. Libera-se a outra página.

C_2 - Verifica-se quantas chaves restaram na página mãe e torna-se a R_4.

C_3 - A raiz tem duas filhas e essas foram con-



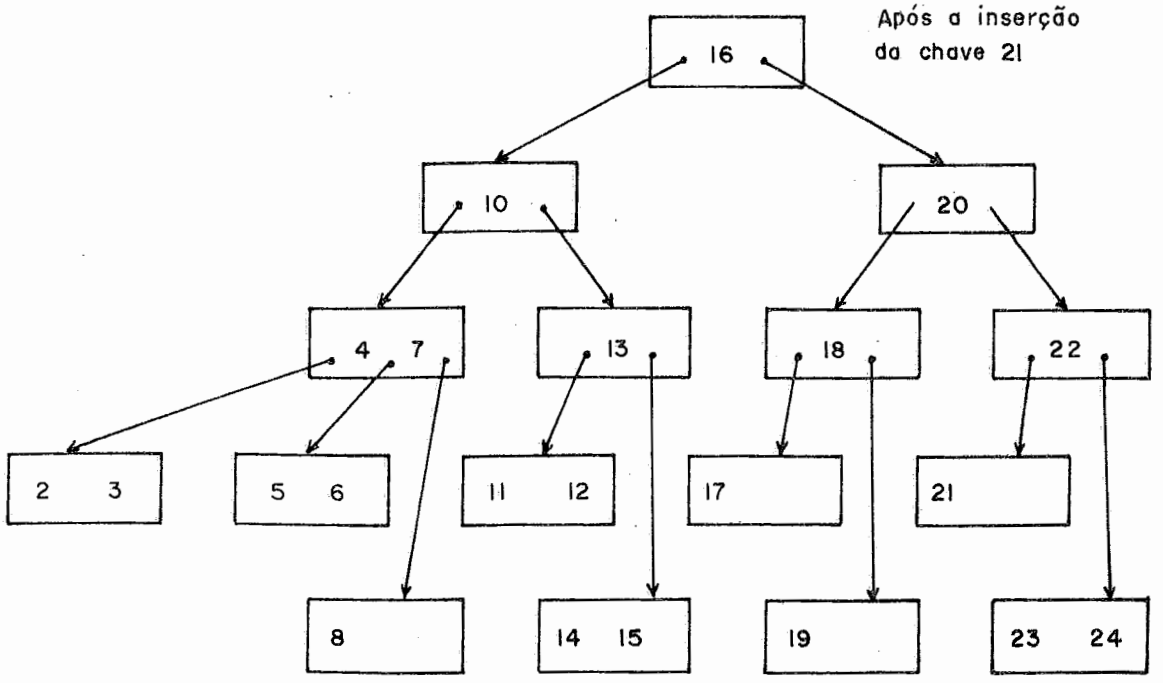


FIGURA 2.5 - INSERÇÃO COM DIVISÃO DE PÁGINA

catenadas, então a nova raiz é a página que recebeu as chaves no processo de concatenação. Libera-se a página que era raiz.

Redistribuição de chaves (Underflow):

Empregada quando a soma com o número de chaves da página irmã vizinha é maior ou igual a 2k.

Consiste em redistribuir as chaves (da página considerada e da irmã vizinha) o mais equitativamente possível entre as duas páginas. Nesta técnica apenas a chave separadora na página mãe é modificada não havendo propagação para outros níveis da árvore.

O algoritmo de remoção pode ser construído de forma a considerar o "underflow" pelos dois lados (direito e esquerdo) e até mesmo com a escolha de qual técnica deverá ser usada, conforme a situação das páginas irmãs adjacentes. A implementação da opção de escolha permite que haja uma sintonização dinâmica do sistema.

A figura (2.6) ilustra a remoção feita com as técnicas de concatenação e "underflow".

II - 4 - 2 - 3 - CUSTOS DAS OPERAÇÕES

Para visitar um nó numa árvore B é necessário fazer um acesso ao dispositivo secundário de armazenamento, de forma que o número de nós visitados durante uma determinada opera-

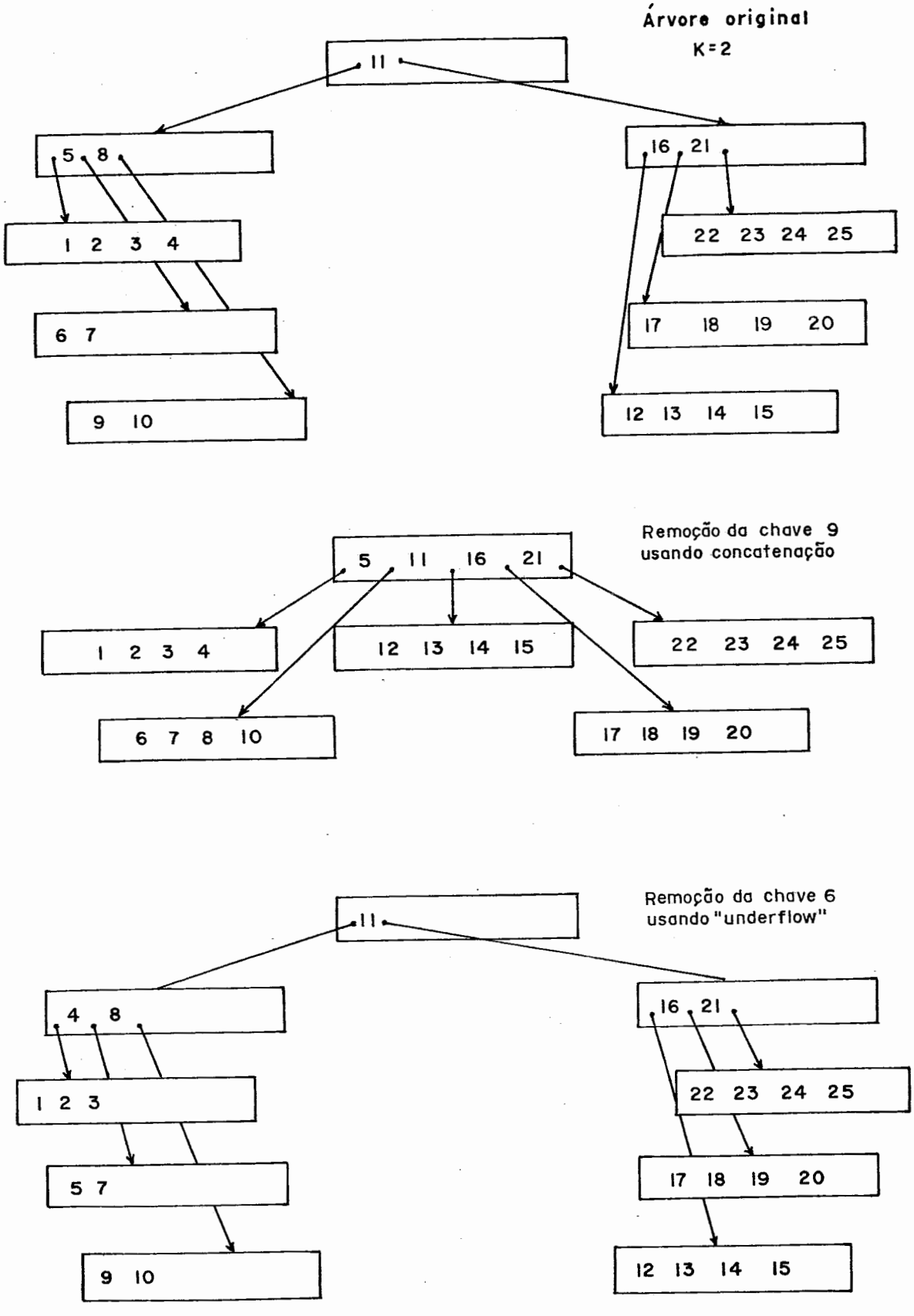


FIGURA 2.6 - REMOÇÃO USANDO CONCATENAÇÃO E "UNDERFLOW"

ção fornece uma medida do seu custo.

Antes de entrar na análise do custo de cada operação vamos ver qual é o número de nós de uma árvore B. Seja N_{min} e N_{max} o número mínimo e máximo de nós de uma árvore T, pertencente a família $\tau(k,h)$ de árvores B. Então:

$$N_{min} = 1 + ((k+1)^0 + (k+1)^1 + \ldots + (k+1)^{h-2})$$

$$= 1 + \frac{2}{k}((k+1)^{h-1} - 1)$$

II-1

$$N_{max} = \sum_{i=0}^{h-1} (2k+1)^i = \frac{1}{2k}((2k+1)^h - 1)$$

Então o número N(T) de nós de $T \in \tau(k,h)$ é dado por:

$N(T) = 0$ se $T \in \tau(k,0)$

$1+\frac{2}{k}((k+1)^{h-1} -1) \leq N(T) \leq \frac{1}{2k}((2k+1)^h - 1)$ caso contrário.

.... 1 - Custo da busca

Sendo h a altura da árvore B, no pior dos casos h páginas precisarão ser consultadas para determinar o êxito ou fracasso da busca. O valor de h é facilmente determinado em função do número I de elementos do índice. Se considerarmos I_{min} e I_{max} o número mínimo e máximo de chaves da árvore $T \in \tau(k,h)$ então podemos facilmente tirar de II-1 que:

$$I_{min} = 1 + k\left(\frac{2}{k}((k+1)^{h-1} -1)\right) = 2(k+1)^{h-1} -1$$

II-2

$$I_{max} = 2k\left(\frac{(2k+1)^h -1}{2k}\right) = (2k+1)^h - 1$$

A altura h da árvore B fica então delimitada por:

$$\log_{2k+1}(I+1) \leq h \leq 1+\log_{k+1}\left(\frac{I+1}{2}\right), \quad \text{para } I \geq 1,$$

$$h = 0 \quad \text{para } I = 0$$

.... 2 - Custo da busca ao próximo

Vamos denotar por f_{min} (f_{max}) o número mínimo (máximo) de páginas acessadas na memória auxiliar.

No melhor dos casos o algoritmo de busca assinala uma folha e o próximo nela mesmo se encontra, quando então teremos:

$$f_{min} = 0$$

No pior dos casos h-1 páginas precisarão ser consultadas, considerando-se que a página assinalada pelo algoritmo de busca já esteja numa área de armazenamento na memória principal, para se encontrar o próximo ou constatar que ele não faz parte do índice, caso o último elemento deste não esteja assinalado como tal.

No caso de haver h áreas de armazenamento podemos garantir que nenhuma página seja acessada mais de uma vez quando então teremos:

$$f_{médio} = \frac{\text{número de páginas}}{\text{número de chaves}} = \frac{N}{I}$$

.... 3 - Custo da inserção

A operação de inserção requer mais acessos à memória auxiliar que as operações de busca, mesmo porque o processo poderá se propagar para os níveis superiores da árvore. Vamos denotar por w_{min} (w_{max}) o número mínimo (máximo) de páginas escritas.

No melhor dos casos, quando houver vaga para a inserção da chave na página, teremos:

$f_{min} = h$, $w_{min} = 1$

No pior dos casos, quando houver propagação da inserção até a raiz, quando nova raiz precisará ser criada, teremos:

$f_{max} = h$, $w_{max} = 2h+1$

.... 4 - Custo da remoção

Como a operação de inserção, esta também poderá se propagar para os níveis superiores da árvore, mantendo estreita dependência à sua altura.

O melhor dos casos ocorre quando não há necessidade de trabalhar com as páginas vizinhas, isto é, a remoção da chave não afeta a estrutura da árvore.

$f_{min} = h$, $w_{min} = 1$

O pior dos casos ocorrerá quando o processo de concatenação propagar-se por h-2 níveis da árvore, o filho da raiz sofrer um "underflow" e a própria raiz for reescrita.

$f_{max} = 2h-1$, $w_{max} = h+1$

.... 5 - Conclusão

Como podemos observar todos os custos são estreitamente dependentes da altura h, ficando clara a importância da ordem k da árvore para a redução desses custos. Contudo, esse fator de ramificação k é limitado por características físicas do sistema. BAYER e McCREIGHT[02] fazem um estudo para a determinação desse parâmetro, que é abordado por nós no anexo 2.

II - 4 - 2 - 4 - LIBERDADES DE IMPLEMENTAÇÃO

.... 1 - "Overflow"

É empregado quando após a operação de inserção resultam mais de 2k elementos na página mas uma das páginas irmãs vizinhas não está totalmente cheia. Neste caso faz-se uma re - distribuição das chaves com a página irmã vizinha considerada. A chave separadora na página mãe é atualizada mas não há propagação para outros níveis.

A técnica de "overflow" possibilita uma melhor utilização do espaço e em arquivos não sujeitos a remoção consegue garantir uma utilização do espaço de armazenamento de aproximadamente 66%.

A figura (2.7) ilustra o uso dessa técnica.

.... 2 -"Pré-paginação".

Quando da criação do índice a partir de um conjunto já conhecido de chaves pode-se fazer uma pré-paginação e se as operações posteriores de atualização forem feitas em lote mais vale ordená-las por suas chaves, o que tornará o processamento mais eficiente.

II - 4 - 3 - VARIAÇÕES DE ÁRVORE B.

.... 1 - Tamanho de página variável com o nível

Como podemos observar na definição original de árvore B, todos os nós terminais (folhas) apresentam os ponteiros indicadores de páginas descendentes com valor λ, o que não ocorre nos demais níveis da árvore. Isto, no total, representa uma significativa quantidade de espaço perdido. A eliminação desses λ's supérfluos é de grande interesse por resultar em ganho de espaço considerável, visto que podemos reduzir o número de páginas do nível inferior, justamente o nível com maior número de páginas da árvore. Para conseguir esse efeito KNUTH[14] sugere

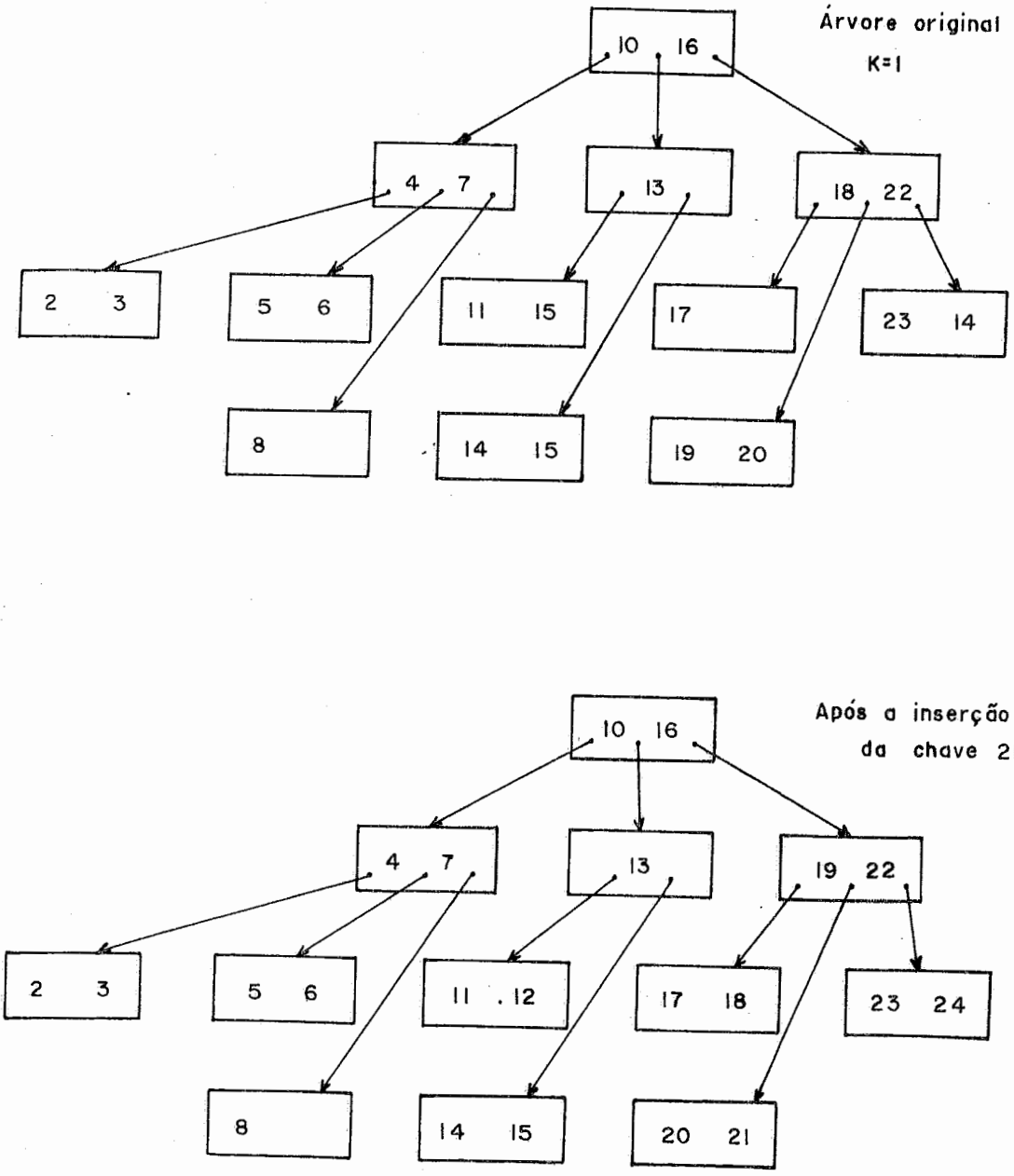


FIGURA 2.7 - INSERÇÃO COM "OVERFLOW"

que se use um diferente valor de k para os nós terminais, de forma que o espaço liberado pelos ponteiros seja usado para os registros.

Uma generalização da idéia é criar uma árvore que tenha os k_i variáveis conforme o nível i em que a página se encontra na árvore. Numa árvore dessa forma é fácil aumentar a sua abertura, bastando para isso atribuir k_i's alto para os primeiros níveis, resultando em mais rapidez para as operações de atualização do arquivo. Uma árvore desse tipo terá os nós da forma mostrada na figura (2.8). Observe que há necessidade de definir o k_i para cada página.

k_i	p_0	x_1	α_1	p_1	...	α_ℓ	p_ℓ

$k_i \leq \ell \leq 2k_i$

nó não folha

k_i'	x_i	α_1	x_2	α_2	...	x_ℓ	α_ℓ

$k_i' \leq 1 \leq 2k_i'$

nó folha

onde: k_i = ordem do nó

p_i = ponteiros

x_i, α_i = dupla de informação associada

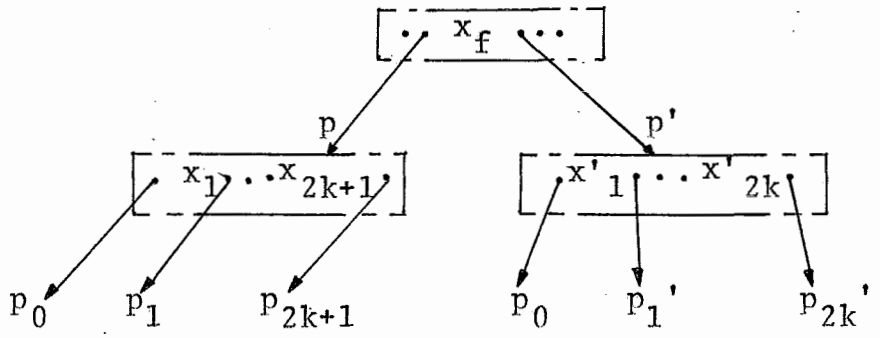
Figura 2.8

.... 2 - Aumento da ocupação mínima.

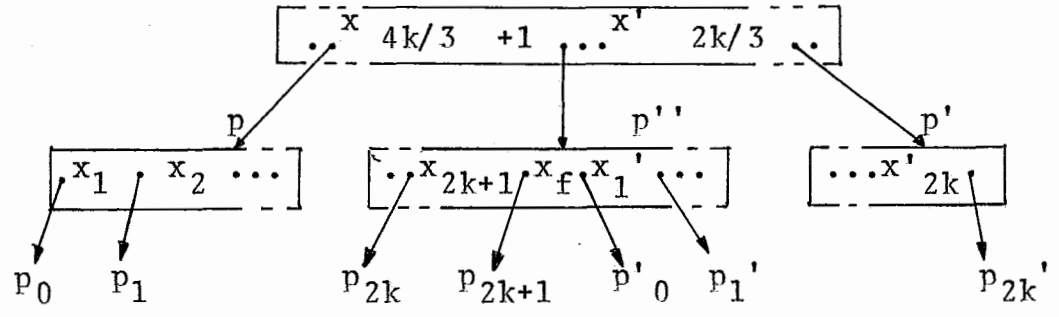
Como já vimos, na operação de inserção numa árvore B, se a operação foi feita numa página cheia há necessidade de reestruturar a árvore após a inserção. Se alguma das páginas irmãs vizinhas não estiver lotada pode-se distribuir as chaves com a página irmã escolhida, na técnica conhecida como "overflow", caso contrário, a página que so-

freu a inserção será dividida em duas. Assim, teremos duas páginas semi-ocupadas entre duas páginas com capacidade esgotada. Com o objetivo de melhor distribuir as chaves entre as páginas KNUTH[14] sugere a divisão de duas páginas em três.

Vamos considerar para efeito de esquematização, que a divisão seja feita com a página irmã à direita p' que está com 2k chaves x_i, i = 1, ... 2k. A inserção foi feita na página p que está com sua capacidade estourada por um. Na página mãe existe uma chave x_f que é o separador entre p e p'.



A divisão de p e p' em três páginas resultará em páginas com quase 2/3 de ocupação, contendo respectivamente $\lfloor 4K/3 \rfloor$, $\lfloor (4K + 1)/3 \rfloor$ e $\lfloor (4K + 2)/3 \rfloor$.



Como consequência desse processo de divisão de duas páginas em três resultou a árvore B*.

... 2 - 1 - Árvore B*

Uma árvore B* difere da árvore B somente quanto ao fator de ocupação das páginas simples e da raiz. As diferenças são estas:

a) Cada nó, exceto a raiz, tem no máximo 2k+1 filhos.

b) Cada nó, exceto a raiz, tem no mínimo (4k+1)/3 filhos.

c) A raiz tem no mínimo 2 e no máximo 2(4k/3)+1 filhos.

A condição mais característica da árvore B* é a b, que garante que são utilizados no mínimo 2/3 do espaço disponível por nó. A condição c, limitadora do número máximo de elementos da raiz garante que quando a raiz for dividida teremos duas páginas com 2/3 de seu espaço ocupado.

... 2 - 2 Árvore B**

O conceito de árvore B* foi desenvolvido um pouco mais por McCREIGHT[17], resultando numa árvore que apresenta ocupação mínima dos nós, salvo a raiz, de 75%. A rotina de paginação, no caso, trataria de escolher as três páginas irmãs vizinhas que seriam redistribuidas em quatro páginas.

... 3 - Chaves de tamanhos variáveis

As chaves podem também ser de tamanho variável. Isto ocasiona algumas modificações na estrutura da árvore que, no entanto, lhe dão maior flexibilidade.

A primeira modificação é na definição do fator de ocupação, que passa a ser considerado em função da percentagem de espaço físico ocupado na página e não mais pelo número de chaves presentes nesta. Assim uma árvore B é definida como aquela cujos nós tem comprimento de p palavras e pode conter registros tais que a soma de seus

tamanhos (incluindo os ponteiros) não ultrapasse p palavras e não seja inferior a p/2 e, a raiz contenha no mínimo um registro. Os demais conceitos de variações de árvore B podem ser adaptados de maneira semelhante.

Observa-se que quando preenchemos a raiz com chaves pequenas vamos conseguir uma grande ramificação, que implica na redução da altura da árvore, como mostra a comparação entre as figuras (2.9) e (2.10) e onde consideramos seis (6) caracteres como a ocupação máxima sem considerar o espaço ocupado pelos apontadores. Isso nos leva a pensar na possibilidade de modificar a estrutura da árvore de maneira a colocar as menores chaves nos níveis superiores, procurando aumentar ao máximo a ramificação. Uma estratégia para tal, a migração de chaves pequenas para o topo, é apresentada por McCREGHT[17] e baseia-se na procura da distribuição que minimize a soma dos caminhos até a raiz.

Outra consequência do uso de chaves de tamanhos variáveis é a possibilidade de usar pseudo-chaves, que podem ser separadores, prefixos ou chaves compactadas.

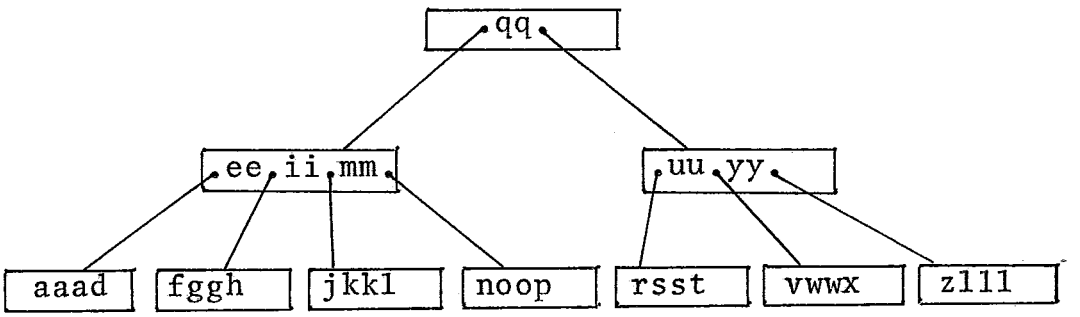


Figura 2.9

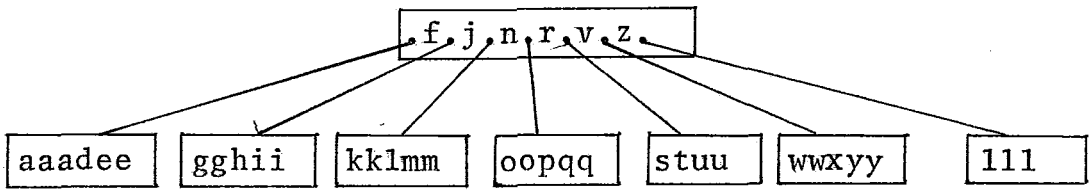


Figura 2.10

Para a utilização dessas técnicas, os nós internos (ou páginas brancas) são preenchidos apenas com pseudo - chaves e ponteiros como será visto com detalhes na seção II-4-3-4.

.... 4 - Localização da informação dentro da estrutura da árvore.

Algumas vezes as chaves constituem pequena parte do registro e a informação associada, quando disposta junto à chave, não contribui para as operações feitas na árvore, apenas ocupa o espaço implicando num maior custo de busca.

Uma idéia apresentada por WEDEKIND[25] e KNUTH[14] consiste em dispor todos os dados nas fo - lhas, deixando os nós internos para as chaves e apontadores. Já BAYER e UNTERAUER[04] avançam mais na idéia e propõem que as chaves e seus registros associados sejam colocados apenas nas folhas, sendo os nós internos preenchidos com separadores apropria - dos.

... 4 - 1 Árvore B^+

É uma árvore B na qual os nós internos são preenchidos apenas por chaves e apontadores para filhos. Estes nós são conhecidos como nós brancos. As informações ou apontadores para estas são colocados apenas nas folhas.

A árvore proposta por WEDEKIND[25] tem os nós terminais (folhas) e não terminais (internos) como mostrado na figura (2.11).

nó interno numa árvore B^+

p_0	x_1	p_1	x_2	p_2	....	x_{2k}^*	p_{2k}^*

nó folha numa árvore B^+

M	x_1	α_1	x_2	α_2	....	x_{2k}	α_{2k}

onde:

p_j = ponteiros para nós filhos.

x_i = chaves

α_i = informação associada a x_i

M = marca para distinguir os nós folhas dos não folhas.

Figura 2.11

Ambos os formatos de nós apresentam a mesma capacidade ℓ em bytes. As árvores desse tipo são descritas por três parâmetros, k, k* e h. Todas as chaves aparecem nos nós folhas com suas informações associadas mas nem todas as chaves aparecem nos nós internos. Observa-se que para conseguir o processamento sequencial do índice basta colocar ponteiros encadeando os nós terminais.

A parte da árvore B^+ formada pelos nós internos é conhecida como índice B^+ e a formada pelas folhas como arquivo B^+.

No caso em que α for apenas um ponteiro para as in

formações poderemos ter k=k*.

... 4 - 2 - Árvore B^+ com prefixo simples.

É uma árvore B^+ onde o índice não é formado necessariamente por chaves mas por separadores de tamanho variável.

As chaves armazenadas no índice B^+ têm função puramente diretiva para o algoritmo de busca. Neste caso, em que não há necessidade de se trabalhar realmente com as chaves, poderemos usar como separadores partes dessas chaves ou mesmo qualquer cadeia de caracteres que direcione o algoritmo de busca.

No caso de estarmos com uma folha da forma:

ALAGOAS, BAHIA, CEARA, PARANA, SERGIPE

a inserção de PARA causaria a divisão da página em:

ALAGOAS, BAHIA, CEARA

PARA, PARANA, SERGIPE

onde em vez de armazenarmos a chave PARA no índice podemos usar um separador, por exemplo P.

Quando as chaves são formadas com as letras de um alfabeto e obedecem à ordem alfabética é possível escolher um separador tal que obedeça à propriedade dos prefixos, assim definida:

Seja x e y duas chaves tais que $x < y$.

Então existe um único prefixo $\bar{y}$ de y tal que

a) $\bar{y}$ é um separador entre x e y.

b) nenhum outro separador entre x e y é menor que $\bar{y}$.

A técnica de escolher o separador que deve ascender à página mãe, quando ocorre a divisão, é usada somen

te na divisão de folhas, na divisão dos nós internos é o próprio separador que é deslocado.

A utilização de uma estrutura de árvore é melhorada à medida que ela se achata, o que é conseguido aumentando o grau de ramificação, especialmente nas partes superiores, isto é, próximo à raiz. O grau de ramificação é determinado pelo número de pares (separadores, apontadores) que é possível armazenar na página o que nos leva a procurar usar os menores separadores possíveis (por isso não se usa as chaves). Uma maneira de conseguir isso sem um custo demasiado grande é definir um intervalo de divisão de página, assim, quando houver necessidade de dividir uma página, pode-se escolher dentro do intervalo o menor dos separadores. A mesma idéia de BAYER e UNTERAUER[04], pode ser aplicada para nós internos, onde define-se um intervalo no qual procura-se o menor separador, que irá ascender para a página mãe.

Os efeitos do uso do intervalo de divisão são úteis no que diz respeito ao aumento do grau de ramificação da árvore, mas causam uma menor ocupação das páginas (na divisão das páginas uma delas pode ficar com fator de ocupação inferior a 50%) o que exige um maior número de páginas.

Embora não tenha feito um estudo sobre a relação entre o tamanho do intervalo de divisão e a estruturação da árvore, BAYER e UNTERAUER[04] indicam como favorável o uso de pequenos intervalos de divisão.

Na figura (2.12) temos um exemplo de árvore B^+ com prefixo simples.

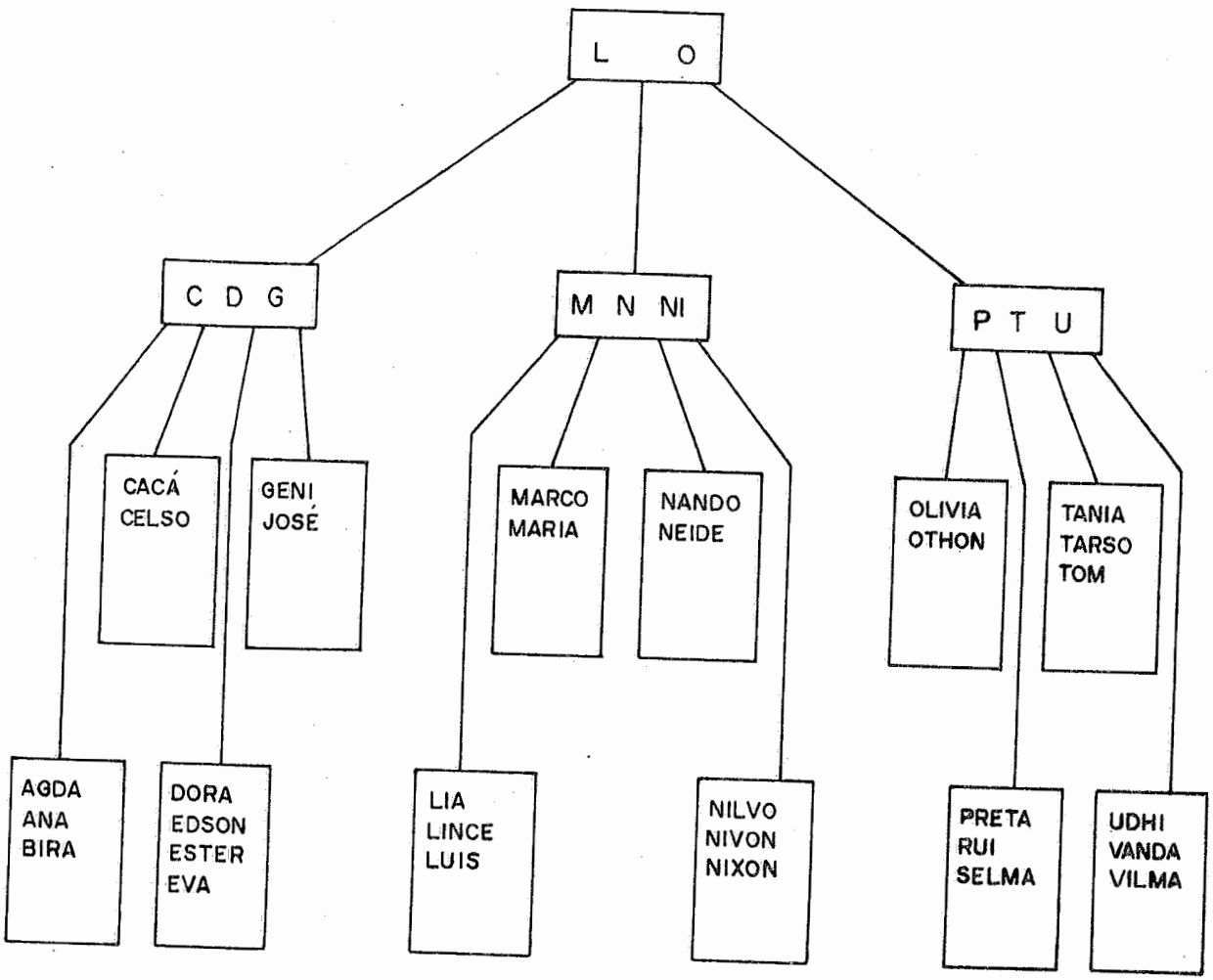


FIGURA 2.12 - ÁRVORE B^+ COM PREFIXO SIMPLES (K=2)

... 4 - 3 - Árvore B^+ com prefixo.

É uma árvore B^+ na qual o índice é constituído com separadores compactados. A compactação de prefixos visa eliminar redundâncias, o que diminui o tamanho dos separadores, trazendo os resultados já discutidos.

A parte do índice de uma árvore B^+ com prefixo é tal que os prefixos são construídos à medida que o algoritmo de busca é realizado.

Para melhor compreensão dessa estrutura assuma que as chaves estejam ordenadas lexicograficamente, segundo o alfabeto com que foram definidas.

Para uma página arbitrária P, seja T(P) a subárvore de nós internos e nós folhas que tem P como raiz. Se considerarmos a parte do índice da árvore, podemos observar que é possível determinar um delimitador inferior $\lambda(P)$ (o maior dos delimitadores inferiores) e um delimitador superior $\mu(P)$ (o menor dos limitadores superiores) tal que, para toda chave x e para todo separador s de T(P), tenha-se:

$$\lambda(P) \leqslant x < \mu(P),$$

$$\lambda(P) \leqslant s < \mu(P).$$

Para definir λ e μ para outros nós considere P um nó interno com delimitadores inferiores e superiores $\lambda(P)$ e $\mu(P)$ respectivamente, e com a estrutura mostrada abaixo:

P $\boxed{P_0, s_1, P_1, \ldots, s_i, P_i, s_{i+1}, \ldots, s_m, P_m}$

onde $P_0, \ldots, P_m$ são apontadores para os filhos de P os quais são nós internos ou folhas; $s_1, \ldots, s_m$ são apontadores, sendo s_m o último em P. Então $\lambda(P(p_i))$ e $\mu(P(p_i))$,

também denotados por $\lambda(p_i)$ e $\mu(p_i)$ são definidos como:

$$\lambda(p_i) = \begin{cases} s_i, & i=1,2,\ldots,m. \\ \lambda(P) & \text{se } i=0 \end{cases}$$

$$\mu(p_i) = \begin{cases} s_{i+1}, & i=0,1,\ldots,m-1. \\ \mu(P) & \text{se } i=m \end{cases}$$

Deduz-se então que todos os separadores ou chaves em $T(p_i)$ devem ter no mínimo um prefixo comum $k(p_i)$, o qual pode ser definido como segue: seja $\bar{k}_i$ o mais extenso dos prefixos comuns (possivelmente a cadeia de caracteres vazia) entre $\lambda(p_i)$ e $\mu(p_i)$.

Então:
$$k(p_i) \begin{cases} k_i.1_j & \text{se } \lambda(p_i) = \bar{k}_i 1_j z \text{ e } \mu(p_i) = \bar{k}_i 1_{j+1} \text{ onde } z \text{ é uma cadeia de caracteres arbitrária e a letra } 1_j \text{ prescede imediatamente a } 1_{j+1}. \\ \bar{k}_i & \text{caso contrário.} \end{cases}$$

$\lambda(p_i)$ e $\mu(p_i)$, e consequentemente $k(p_i)$, podem ser derivadas caminhando-se através da árvore da raiz até o nó $P(p_i)$. Assim, não há necessidade de se repetir a armazenagem do prefixo comum $k(p_i)$ em $P(p_i)$, é suficiente armazená-lo uma vez e armazenar o restante $\hat{s}$ dos separadores em $P(p_i)$. Assim quando se chega ao nível das folhas a chave está completa embora seja interessante armazenar as chaves completas nas folhas ou então repetir nas folhas o prefixo comum para o processamento sequencial do arquivo. Um separador completo s de $P(p_i)$ é facilmente reconstruído pela concatenação $s=k(p_i)/\hat{s}$.

III. DEFINIÇÃO DO SISTEMA

Pág.

1. Apresentação

O sistema foi desenvolvido visando a permitir que usuários da linguagem FORTRAN pudessem construir e acessar arquivos residentes em discos magnéticos, de maneira sequencial e aleatória, utilizando os recursos oferecidos por uma estrutura de índices.

É permitido o uso de chaves numéricas e alfa numéricas, de tamanho constante ou variável, e pode-se construir mais de um índice para um mesmo arquivo, não havendo uma relação hierárquica entre esses índices.

A implementação foi feita num computador IBM 1130 usando-se para a estruturação do índice uma variação de árvore B, a árvore B^+ com prefixo simples.

Neste capítulo, vamos apresentar na seção III-2 a forma de interação do usuário com o sistema; na seção III-3 apresentamos a definição da estrutura escolhida para a implementação e, na seção III-4 apresentamos todos os subprogramas do sistema com seus correspondentes algoritmos.

Na definição dos subprogramas explicamos sua função, apresentamos os parâmetros, os códigos de erro e algumas observações explicativas. Os algoritmos são apresentados escritos numa pseudo-linguagem algoritmica em português.

2. Interação do Usuário com o Sistema

O usuário interage com o sistema mediante a chamada às subrotinas de Inserção, Remoção e Busca por Chave. Está também a sua disposição outras subrotinas auxiliares de Busca, tais como a Busca ao Anterior, a Busca ao Próximo e a Busca por Posição. Outra facilidade colocada a sua disposição é a subrotina de Carga, capaz de fazer a carga inicial do índice de forma maciça, evitando o trabalho de inserção um a um. Todas as subrotinas apresentam código de erro, que deve ser analisado pelo usuário.

O sistema permite que mais de um índice seja construído para um arquivo, mas compete ao usuário coordenar a inserção e remoção nos índices associados.

No caso da inserção, deverá certificar-se de que um registro pertence ou não ao arquivo antes de inseri-lo em um índice, para evitar multiplicidade de registros. Se o registro já existe no arquivo é necessário fornecer à rotina de inserção a posição em que se encontra o registro no arquivo, o que pode ser obtido seja buscando-se num dos índices já existentes (uma vez que se conheça a correspondência entre as chaves do índice consultado e as do índice a ser construído) ou buscando-se diretamente no arquivo de registros.

Quanto à remoção, ela é feita geralmente só no índice indicado e só será feita também no arquivo de registros se existir apenas esse índice associado ao arquivo.

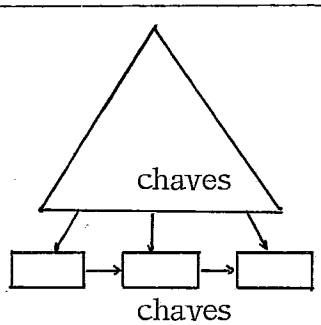
O sistema básico é composto por três arquivos residentes em disco: o arquivo contendo os registros de informações, o arquivo índice e um pequeno arquivo usado como cabeçalho. Compete ao usuário fazer a criação desses arquivos em disco e definir, em seu programa de utilização, a área de uso comum ("COMMON") necessária ao sistema.

Na seção III-4-3 e no Anexo 04 - Manual do Usuário, constam outros esclarecimentos quanto ao uso dos arquivos e do sistema.

3. Definição da Estrutura Usada

Para a escolha da árvore a ser usada nesta implementação, fizemos uma comparação entre as estruturas vistas na Revisão da Literatura, esquematizada a seguir.

Árvore B

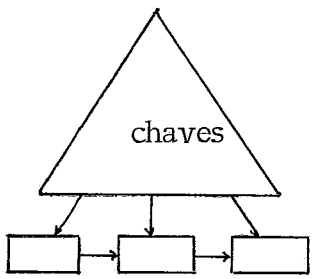


Características:

a) Tem as chaves distribuídas pelas páginas internas e folhas.
b) As páginas tem ocupação mínima de 50%.
c) Em arquivos não sujeitos a remoção, usando-se a técnica de "overflow" tem-se uma ocupação média de 66%.

Vantagens:
Algoritmos simples de serem implementados.
Nível de programação: simples.

Árvore B^+



Características:

a) Todas as chaves estão presentes no nível das folhas.
b) As páginas tem ocupação mínima de 50%.

Vantagens:

a) Facilidade para o processamento sequencial.
b) Economia de ponteiros (os apontadores de registros aparecem apenas no nível das folhas).

Nível de programação: simples.

Árvore B^*

Características:

a) No processo de divisão de páginas é feita a divisão de duas páginas em três.
b) As páginas tem ocupação mínima de 66%.

Vantagens:

A ocupação média é mais elevada.

Nível de programação: menos simples que B e B^+.

Árvore B^{**}

Características:

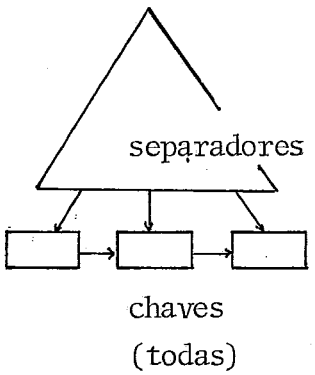
a) No processo de divisão de páginas é feita a divisão de três páginas em quatro.
b) As páginas tem ocupação mínima de 75%.

Vantagens:

a) Melhora o aproveitamento do espaço de armazenamento.
b) Pode-se trabalhar com um "buffer" menor tendo-se um bom desempenho quanto ao número de acessos ao disco.

Nível de programação: menos simples que B^*.

Árvore B^+ com PREFIXO SIMPLES



Características:

a) Todas as chaves estão presentes no nível das folhas.

b) As páginas internas são formadas com prefixos das páginas filhas.

c) As páginas tem ocupação mínima de 50%.

Vantagens:

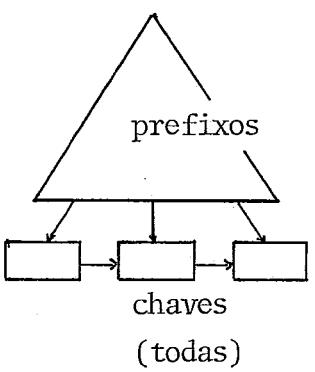
a) Facilidade para o processamento sequencial.

b) A busca no nível das páginas internas é mais rápida.

c) Exige um número de acessos ao disco sempre menor ou igual ao exigido pela Árvore B^+.

Nível de programação: menos simples que B^+.

Árvore B^+ com PREFIXO



Características:

a) Todas as chaves estão presentes no nível das folhas .

b) As páginas internas são formadas com partes de prefixos das chaves. O prefixo completo é formado percorrendo-se a árvore da raiz até a folha.

c) As páginas tem ocupação mínima de 50%.

Vantagens:

a) Facilidade para o processamento sequencial.

b) Exige um número de acessos ao disco menor ou igual ao exigido pela árvore B^+ com prefixo simples.

Nível de programação: muito menos simples que B^+ com prefixo simples.

Utilizamos o trabalho do BAYER e UNTERAUER[04] para a estimativa do tamanho dos separadores, na árvore B^+ com prefixo simples, e, dos prefixos comuns, na árvore B^+ com prefixo.

Para a construção do Gráfico 3.1 e Gráfico 3.2 utilizamos a Tabela 3.1 onde apresenta-se a relação entre a altura da árvore e o tamanho do índice considerando-se diferentes valores para o tamanho das sub-chaves (que podem ser separadores ou prefixos).

Os gráficos foram feitos considerando-se um alfabeto de cardinalidade α=13. No Gráfico 3.1 os tamanhos dos separadores foram obtidos interpolando-se entre os alfabetos de cardinalidade 10 e 16; já no Gráfico 3.2 os tamanhos dos separadores e prefixos foram obtidos interpolando-se entre 2 e 26. Isto explica a diferença existente entre a Árvore B^+ com prefixo simples do Gráfico 3.1 e a do Gráfico 3.2.

A análise do Gráfico 3.1 nos mostra que a Árvore B^+ com prefixo simples exige um menor número de acessos ao disco que a B^+, no caso de índices pequenos (cerca de 1000 registros) e menor ou igual à medida que o índice aumenta. Sabemos também que a busca interna nas páginas de uma árvore B^+ com prefixo simples é mais rápida que a feita nas páginas de uma árvore B^+.

O gráfico 3.2 nos mostra que a árvore B^+ com prefixo exige um menor número de acessos ao disco que a B^+ com prefixo simples. Contudo o uso dos prefixos exige um esforço computacional maior que o necessário quando se usa separadores.

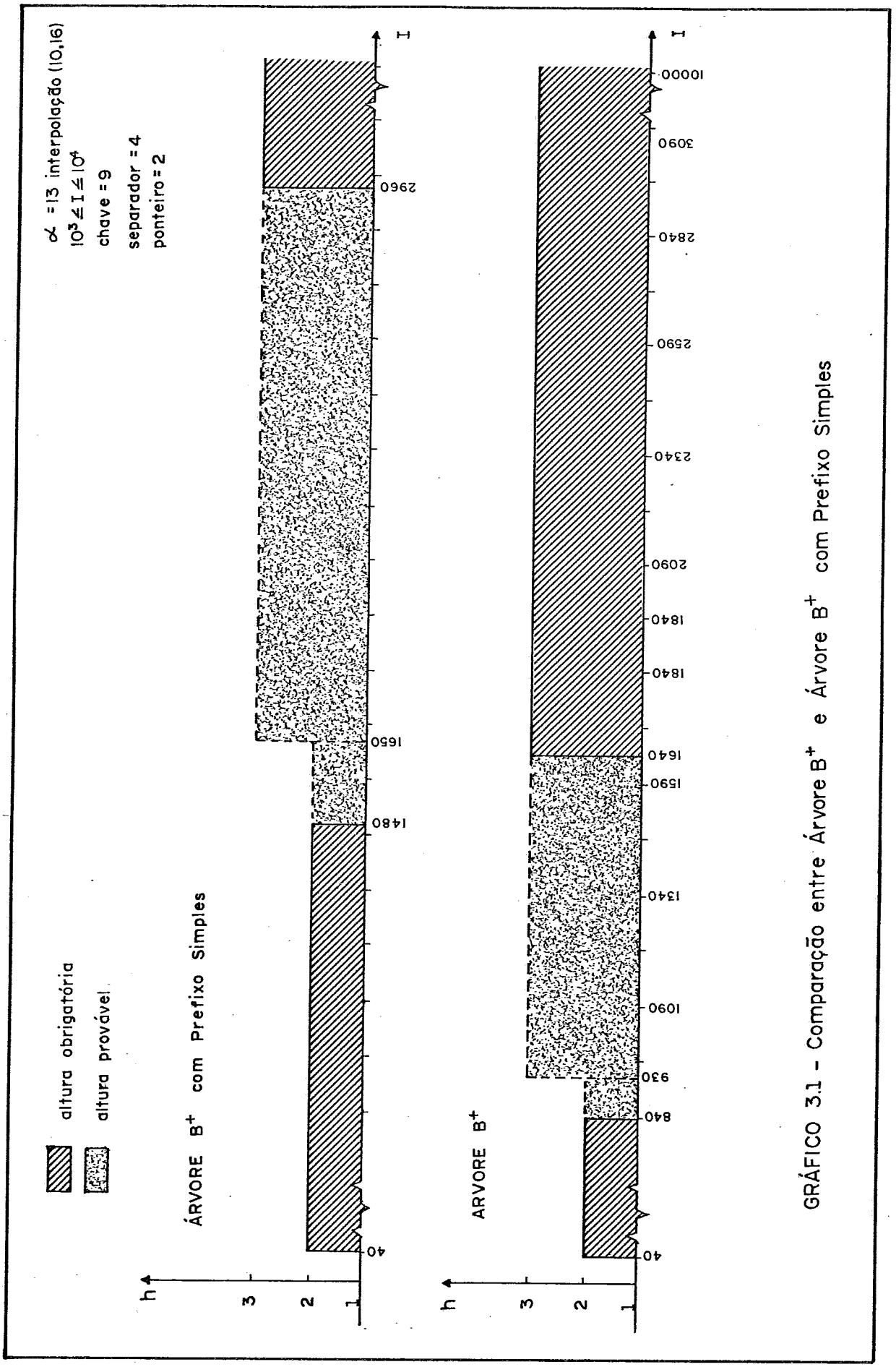


GRÁFICO 3.1 - Comparação entre Árvore B^+ e Árvore B^+ com Prefixo Simples

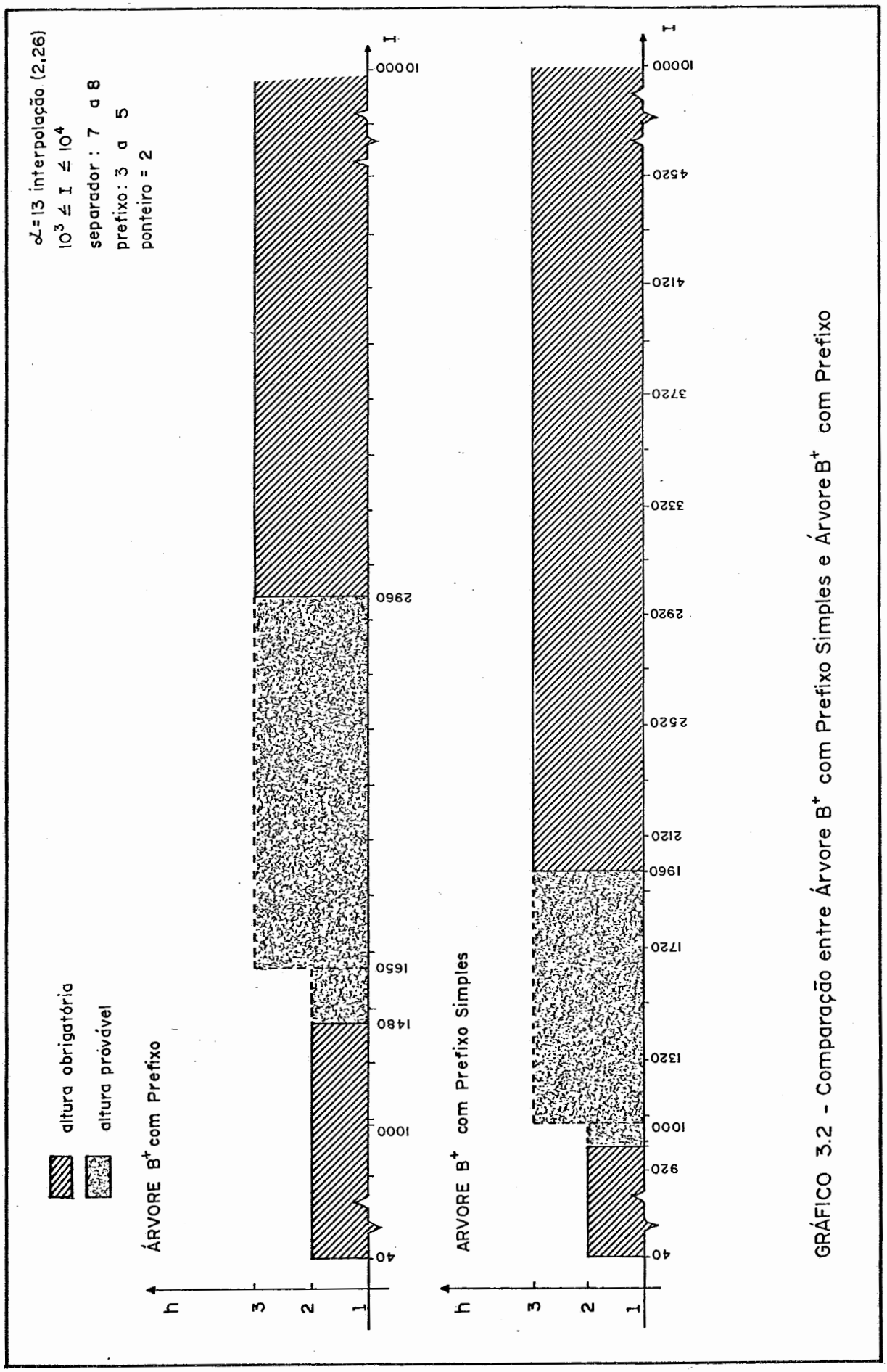


GRÁFICO 3.2 - Comparação entre Árvore B^+ com Prefixo Simples e Árvore B^+ com Prefixo

Não usamos as conclusões tiradas pelos autores[04] de seus resultados experimentais por encontrarmos algumas falhas na própria apresentação dos dados feita por eles, tais como: apresentam em colunas por número de páginas e k os resultados obtidos em termos de tempo computacional e número de acessos ao disco para os três modelos de árvore (B^+, B^+ com prefixo simples e B^+ com prefixo).

α=13			ponteiro = 2 bytes			página = 440 bytes	
h	I_{min}	$I_{méd}$	$I_{máx}$	h	I_{min}	$I_{méd}$	$I_{máx}$
1	1	30	40	1	1	30	40
2	40	930	1640	2	40	1650	2960
3	840	28830	67240	3	1480	90750	219040
4	17640			4	57760		
Chave = 9 bytes				Sub-chave = 4 bytes			
h	I_{min}	$I_{méd}$	$I_{máx}$	h	I_{min}	$I_{méd}$	$I_{máx}$
1	1	30	40	1	1	30	40
2	40	1410	2520	2	40	1230	2240
3	1280	66270	158760	3	1120	50430	125440
4	40960			4	31360		
Sub-chave = 5 bytes				Sub-chave = 6 bytes			
h	I_{min}	$I_{méd}$	$I_{máx}$	h	I_{min}	$I_{méd}$	$I_{máx}$
1	1	30	40	1	1	30	40
2	40	1110	1960	2	40	1020	1800
3	1000	41070	96040	3	920	34680	81000
4	25000			4	21160		
Sub-chave = 7 bytes				Sub-chave = 8 bytes			

Tabela 3.1 - Relação entre Altura e Tamanho do Índice.

Ora, se considerarmos o k e o número de páginas da árvore fixos, para os três modelos, deveremos então ter o tamanho das páginas internas variável de um modelo para outro mas, a altura das árvores seria sempre correspondente entre si, o que traria um igual desempenho, quanto ao número de acessos ao disco, para os três modelos, o que não corresponde aos resultados experimentais apresentados pelos autores citados.

Se considerarmos todas as árvores com o mesmo tamanho de página, não temos como justificar o mesmo valor do k e do número de páginas, entre os três modelos, visto que as chaves, os separadores e os prefixos diferem muito em tamanho. Também não há como justificar um desempenho igual entre as três, quanto ao número de acessos ao disco, para índices pequenos.

Como o sistema a ser projetado deveria atender tanto a chaves de tamanho constante como variável e ser eficaz para o processamento sequencial, uma estrutura tipo Árvore B^+ para chaves de tamanho variável era o mínimo desejável.

Considerando que o desempenho da B^+ com prefixo simples é sempre melhor ou igual ao da B^+, com relação ao número de acessos ao disco; a busca interna nas páginas é mais rápida na B^+ com prefixo simples do que na B^+.

Considerando que a B^+ com prefixo, embora exija um menor número de acessos ao disco que a B^+ com prefixo simples necessita de muito mais tempo computacional que os outros modelos de árvore B e a sua programação é a mais complexa.

Considerando que a divisão de duas páginas em três causa um melhor aproveitamento do espaço ocupado pelas páginas.

Decidimos desenvolver uma estrutura que poderia ser chamada de "árvore B^+ com prefixo simples com características de B^*", pois implementamos a divisão de duas páginas em três e cuidamos para que na carga inicial do índice (carga maciça) todas as páginas estivessem ocupadas em dois terços de sua ca

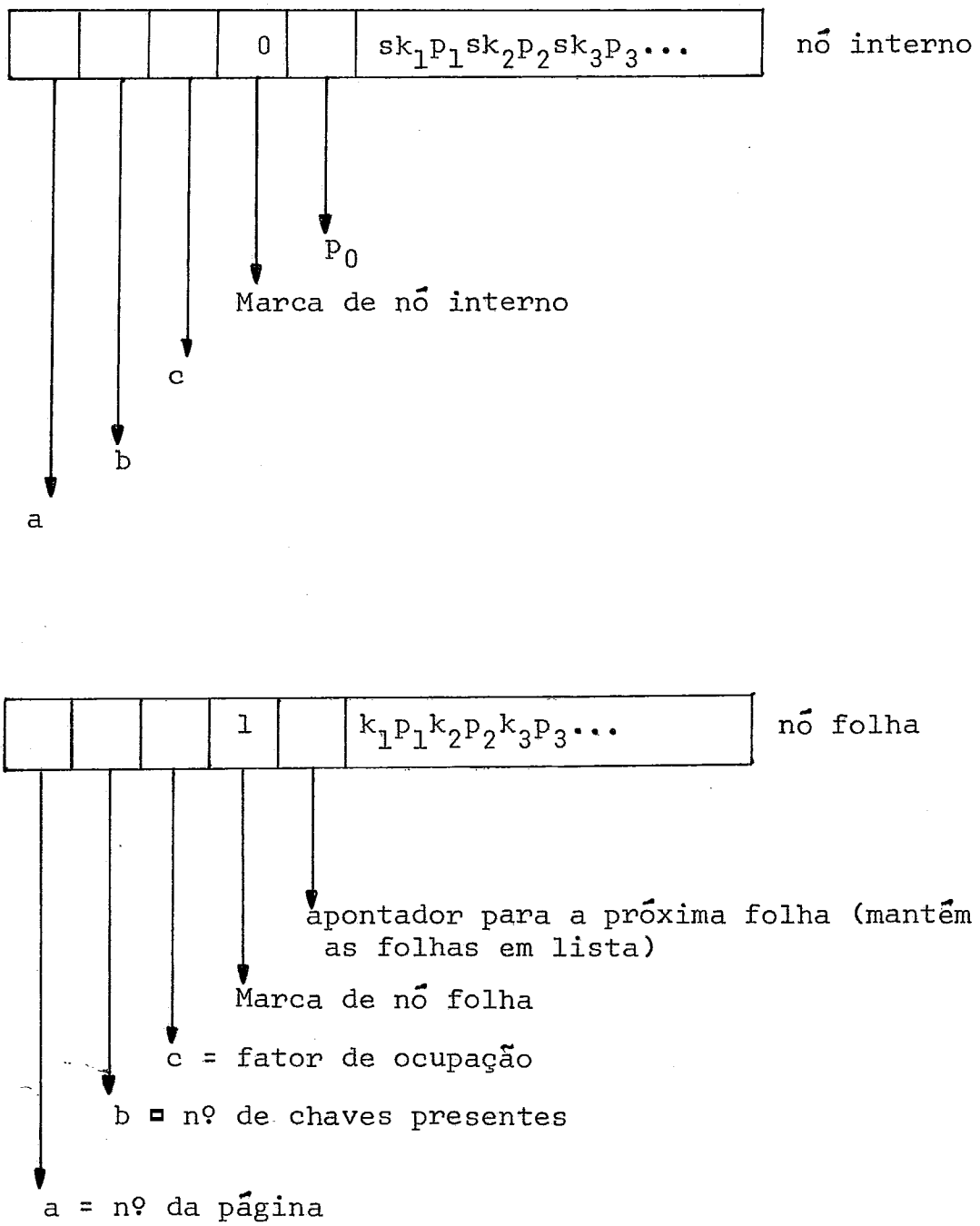
pacidade.

Introduzimos uma pequena modificação na estrutura da árvore no sentido de permitir que sempre fosse possível fazer, de imediato, uma inserção ou remoção, ocorrendo só depois um balanceamento, que envolve apenas a página afetada, suas irmãs e a página mãe, o que oferece facilidades para a implementação do sistema com acesso concorrente.

As páginas usadas nessa estrutura tem o formato como apresentado na Figura 3.1. Observa-se que a quinta palavra de controle tem função diferente nas páginas internas (onde é usada para guardar o apontador da primeira página descendente) e nas folhas (onde é usada para encadeá-las no sentido da próxima, o que torna muito simples o processamento sequencial).

O estudo do tamanho ideal do parâmetro k, conforme indicado por BAYER e McCREIGHT[02], e apresentado por nós no Anexo 02, indicou ser onze setores o tamanho ideal para as páginas do índice mas, como o computador usado para a implementação dispunha de pouca memória (um IBM 1130 com 16K), o tamanho da página foi reduzido a um setor (320 palavras), e isto significa que estamos trabalhando com um valor 32% pior que o ótimo.

O sistema foi projetado para o uso de chaves numéricas e alfanuméricas de tamanho constante ou variável. No caso das chaves apresentarem brancos intermediários, o tamanho da chave deverá ser definido como constante. As chaves numéricas devem ser inteiros positivos não maiores que 32767 e são transforma



sendo:

k_i = chave i

sk_i = separador i

p_i = ponteiro para página folha ou para o registro i

Figura 3.1 - DEFINIÇÃO DE PÁGINAS

das pelo sistema para o código EBCDIC, sendo armazenados dois caracteres dígitos por palavra, ou seja, é interpretada como sendo alfanumérica de tamanho variável (o código de tamanho da chave deverá ser definido como variável). O tamanho máximo das chaves alfanuméricas está restrito a 16 caracteres (8 palavras).

O índice é construído com chaves e separadores alfanuméricos e por isso as chaves numéricas precisam ser transformadas, o que é feito dentro do procedimento de Busca. A chave numérica deverá ser sempre armazenada pelo usuário na primeira posição do vetor chave antes do uso de uma função que o tenha como argumento de entrada para o procedimento (tais como Busca, Inserção ou Remoção).

Damos a seguir destaque a algumas partes do sistema:

a) Preparação do Índice

A preparação do índice é uma das funções da subrotina de Carga, isto é, é com o seu uso que se escreve o cabeçalho do índice e prepara-se a pilha de páginas disponíveis. Quando da preparação do primeiro índice associado ao arquivo é construída também a pilha dos registros disponíveis.

São necessários três arquivos em disco, dimensionados como consta do Anexo 04, o Manual do Usuário, para preparar o índice: o primeiro arquivo é construído pelo usuário e contém informações necessárias para a identificação do índice, não tendo utilização após a Carga; o segundo arquivo é o dos regis-

tros e o terceiro é o arquivo índice.

Como pode-se criar mais de um índice para um mesmo arquivo, o cabeçalho do arquivo de registros é atualizado sempre que se faz uso da função de Carga.

Na Figura 3.2 apresentamos o desenho dos cabeçalhos dos arquivos e logo a seguir a definição de todas as variáveis empregadas nos cabeçalhos. A maior parte das informações constantes dos cabeçalhos é mantida numa área de uso comum às rotinas do sistema e ao programa de utilização.

b) Carga Inicial do Índice (Carga Maciça)

Outra função da subrotina de carga é fazer a carga inicial do índice de forma maciça, evitando a inserção das chaves uma a uma. Para que se possa fazer uma carga maciça é necessário que o arquivo de registros esteja ordenado crescentemente segundo a chave a ser usada no índice. A idéia usada para essa carga é simples: constrói-se cada nível da árvore por vez, começando pelo das folhas, preenchendo as páginas até 2/3 de sua ocupação, salvo a raiz e a última página do nível (que será balanceada ao final da carga). A opção para o emprego ou não da carga maciça é um dos parâmetros da subrotina de Carga.

Quando não se pode usar o recurso da carga maciça o índice é construído pela inserção das chaves e registros um a um.

Caso o programa de utilização do sistema não

DESENHO DOS CABEÇALHOS

```
1-2-3-4-5-7-8-9-10-11-12-13-22.
NOARI, NOARR, MAXRG, MAXPG, NOREG, TMCHV, TMPAG,
TMREG, NOCMP, TABCP, TMCMP, CHV, ALFA, DOCUM.
```

arquivo CARTÃO (22 palavras)

```
5-16-20-21-2-3-9-10
11-12-13-22-19.
NOREG, DISPR, NOIND,
INDCS, NOARR, MAXRG,
TMREG, NOCMP, TABCP,
TMCMP, CHV* , ALFA*,
DOCUM, TBFRM.
```

arquivo REGISTRO (151 palavras)

```
6-5-16-15-14-17-18-20
1-2-4-7-8-9-10-11-12
13-22.
NOCHI, NOREG, DISPR,
DISPI, RAIZ, PRIM,
ULTMA, NOIND, NOARI,
NOARR, MAXPG, TMCHV,
TMPAG, TMREG, NOCMP,
TABCP, TMCMP, CHV,
ALFA, DOCUM.
```

arquivo ÍNDICE (69 palavras)

Obs.: (*) Refere-se à chave do índice principal, segundo o qual o arquivo está ordenado.

O arquivo cartão cabeçalho deve ser transformado num arquivo em disco.

Os números correspondem às variáveis e auxiliam a encontrar a sua definição na folha seguinte.

Figura 3.2

ELEMENTOS CONSTITUINTES DOS CABEÇALHOS

1. NOARI - nº/nome do arquivo índice
2. NOARR - nº/nome do arquivo de informações
3. MAXRG - nº máximo de registros permitidos (no arquivo)
4. MAXPG - nº máximo de páginas permitidas
5. NOREG - nº atual de registros no arquivo
6. NOCHI - nº atual de chaves no índice
7. TMCHV - tamanho da chave { = 0 tamanho fixo; ≠ 0 tamanho variável }
   (em palavras - é armazenada como alfanumérica no índice)
8. TMPAG - tamanho da página (em palavras)
9. TMREG - tamanho do registro (em palavras)
10. NOCMP - nº de campos no registro
11. TABCP(20) - tabela de campos - contêm os números das palavras
    TMCMP(20) onde tem início os campos e os seus tamanhos.
12. CHV - nº do campo chave no registro
13. ALFA - tipo da chave { ≠ 1 numérica; = 1 alfanumérica }
14. RAIZ - número da página raiz do índice
15. DISPI - apontador topo da pilha de páginas disponíveis (no arquivo índice).
16. DISPR - apontador topo da pilha de registros disponíveis (no arquivo registros)
17. PRIM - apontador primeira folha do índice
18. ULTMA - apontador última folha do índice
19. TBFRM - tabela de formatos
20. NOIND - número de índices associados ao arquivo
21. INDCS (10) - número dos índices associados ao arquivo.
22. DOCUM (12) - informações documentacionais.

OBS.: Parte desses elementos são fornecidos pelo usuário quando da criação do índice, outros são atualizados quando do seu uso.

tenha usado anteriormente a subrotina de Carga, o usuário deve providenciar a leitura do cabeçalho do arquivo índice, pois aí estão grande parte das variáveis da área de uso comum ao sistema.

c) Inserção no Índice (ou nos índices)

A inserção poderá ser feita no índice e no arquivo ou só no índice (quando já houver sido feita a inserção num outro índice associado ao arquivo). O vetor com a chave a ser inserida é argumento de entrada da subrotina de Inserção enquanto que o registro completo passa ao subprograma através da área de uso comum ao sistema.

Compete ao usuário coordenar a inserção nos índices associados ao arquivo pois, caso contrário, poderá haver multiplicidade de registros no arquivo desses. A partir da inserção no segundo índice associado a um arquivo de registros, o apontador de registro passa a ser argumento de entrada para a subrotina de Inserção. Não existe uma relação hierárquica entre os índices mas é aconselhável que se eleja um dos índices como principal, para que se possa saber com facilidade se um determinado registro já foi inserido no arquivo.

O sistema, da maneira como está implementado, considera todas as chaves como sendo um campo de registro, podendo o usuário contudo criar um índice sobre a concatenação de diversos campos, o que poderá ser feito com o artifício de se definir um campo fictício que englobe os outros.

Os números identificadores dos arquivos (de registros e de índice) são mantidos em área de uso comum ao sistema devendo-se cuidar da sua atualização quando se está trabalhando com mais de um índice num mesmo programa.

d) Remoção no Índice (ou nos índices)

A remoção geralmente só é feita no índice indicado. Para ser feita também no arquivo é necessário que a variável que contém o número de índices associados ao arquivo seja um. Esta variável consta do cabeçalho do arquivo de registros e é mantida em área de uso comum, podendo ser modificada, pelo usuário, como um artifício, para fazer a remoção também no arquivo, quando ele estiver coordenando a remoção em todos os índices associados e, tratar-se realmente do último índice a ser considerado.

O espaço ocupado pelo registro removido é colocado como disponível, podendo ser reutilizado.

e) Balanceamento

Como tanto a inserção como a remoção podem afetar a ocupação das páginas, deixando-a fora do intervalo desejado, foi desenvolvida uma subrotina de balanceamento que usa as técnicas de Redistribuição entre Páginas (escolhendo aleatoriamente uma das páginas irmãs vizinhas), Divisão de Página (é feita a divisão da página em 2/3 e o 1/3 restante é redistribuído com uma página irmã vizinha) e União de Páginas (também considerando aleatoriamente uma das irmãs vizinhas). O balanceamento ocorre após

o uso de uma inserção ou remoção ou após a última inserção feita, quando da carga inicial do índice.

O balanceamento, como foi implementado, envolve apenas a página, suas irmãs vizinhas e a página mãe e ocorre sempre que, no caminho percorrido da raiz até a página folha, com que se está trabalhando, encontra-se uma página cujo fator de ocupação atingiu um dos extremos, máximo ou mínimo, previstos. É possível que, entre as páginas do caminho percorrido, mais de uma precise ser balanceada.

f) Busca por Chave

A função de Busca por Chave consiste em pesquisar se uma determinada chave pertence, ou não, ao índice e ao arquivo e fornecer o registro ao usuário, caso pertença, através da área de uso comum ao sistema. Dentro das páginas a busca é feita usando-se a pesquisa binária.

Durante a Busca por Chave constrói-se uma pilha do caminho percorrido, o que é de muita utilidade para a execução de outras funções (o caminho percorrido permanece na área de uso comum ao sistema).

É durante a Busca por Chave que se verifica se uma chave é numérica, providenciando-se então sua conversão para alfanumérica.

g) Buscas Auxiliares

Devido ao encadeamento existente entre as fo

lhas, no sentido do próximo, a função de Busca ao Próximo é facilmente executável. Seu uso deve ser posterior ao de uma Busca por Chave ou a si própria.

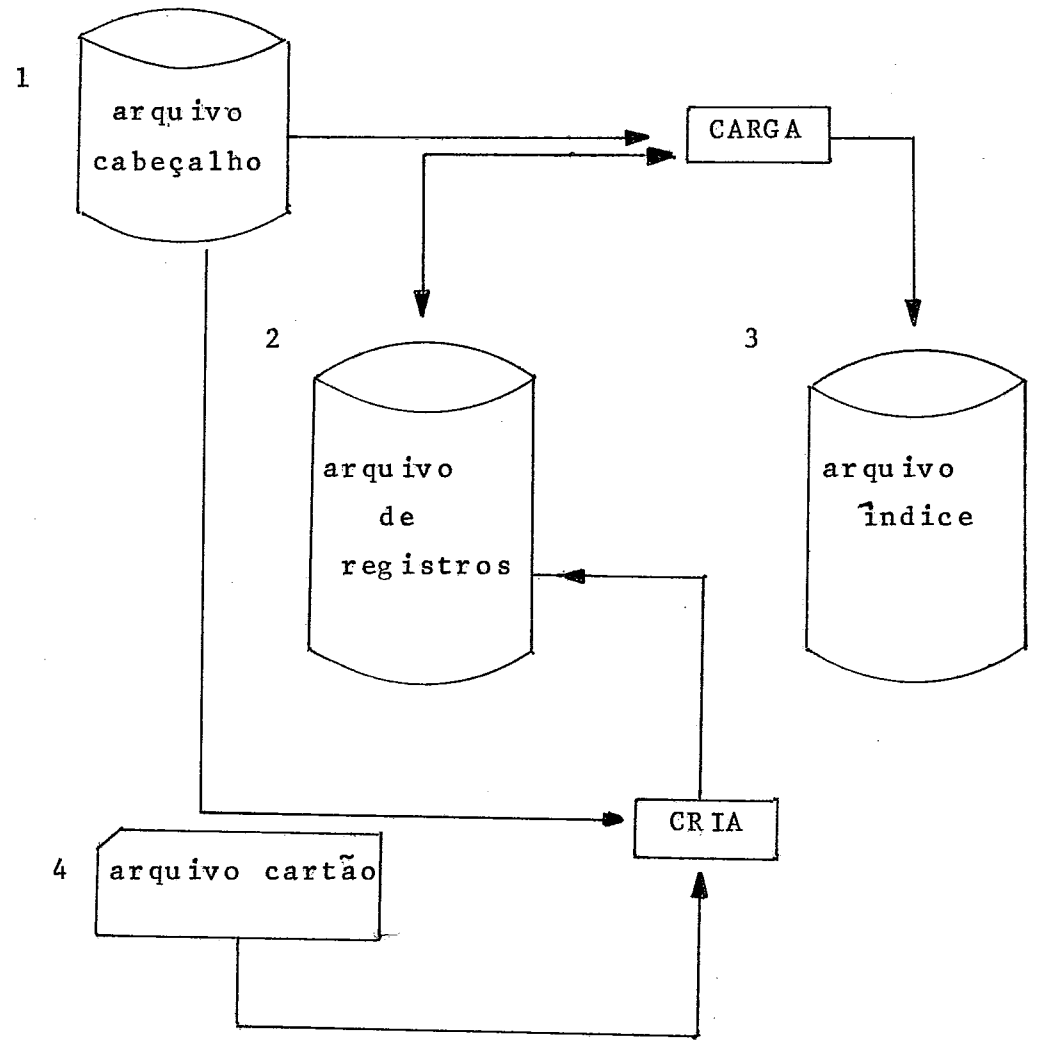
A Busca ao Anterior pode ser um pouco menos facilmente executável devido à existência da pilha com o caminho percorrido na fase de busca (ao invés de um ponteiro para o encadeamento no sentido do anterior, considerando-se o pouco uso que se espera dessa função). Seu uso é posterior ao de uma busca por Chave ou a si própria e mantém atualizada a pilha com o caminho percorrido.

A outra função de busca auxiliar é a Busca por Posição. Sua execução é facilitada pela existência de apontadores para a primeira e última folha.

Todas as funções auxiliares de busca devolvem ao usuário um apontador para a posição ocupada pelo registro no arquivo.

h) Extensões Previstas

Previu-se também a utilização de uma subrotina que tivesse a função de interpretar os cartões de dados, segundo uma tabela de formatos, e armazenar os registros num arquivo em disco, conferindo a ordem existente entre eles. Foi definida como sendo a subrotina Cria mas, como seu uso seria opcional no sistema, não foi incorporada a ele nessa fase.



Uso dos Arquivos pela CARGA e CRIA

Figura 3.3

Na figura 3.3 temos um esquema apresentando a opção entre o uso das funções de Carga e/ou Cria, que também demonstra a relação existente entre os arquivos necessários ao sistema.

4. Definição dos Subprogramas do Sistema

Apresentamos os algoritmos das subrotinas e observações referentes ao seu uso. No Anexo III encontram-se as listagens.

.. 01 - Procedure CARGA

parâmetros: tipo da carga, erro.

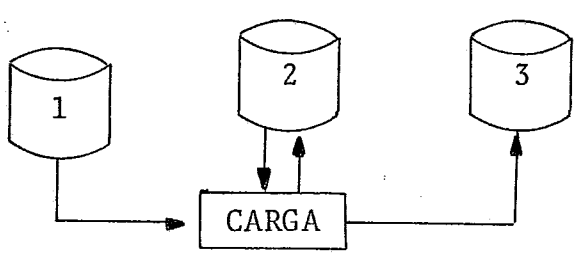
tipo da carga { maciça = 0
só cabeçalhos ≠ 0

função: Lê o arquivo cabeçalho (1).
Se a carga é maciça
então lê o arquivo de registros (2), constrói o cabeçalho para esse arquivo, constrói um índice para o arquivo indicado.
Constrói o cabeçalho para o arquivo índice.
Arruma o espaço disponível no arquivo índice.

idéia: Coloca-se as chaves nas páginas folhas enchendo-as até 2/3 de sua capacidade. As folhas são encadeadas em fila através do ponteiro de próxima folha.
Depois constrói-se os níveis superiores retirando um separador pequeno entre a página e sua vizinha à direita , do nível anterior. Estas páginas também são preenchidas até 2/3 de sua capacidade e os números de suas páginas são colocados numa fila (a ser usada na construção do nível imediatamente superior).

Obs. 1 - Se a chave do índice não corresponde à chave segundo a qual o arquivo está ordenado não existe carga maciça e sim inserção um a um.

Obs. 2 - O usuário deve construir um arquivo cabeçalho com informações necessárias à construção do índice.



Códigos de erro
0 - tudo bem
1 - o arquivo índice está mal dimensionado.

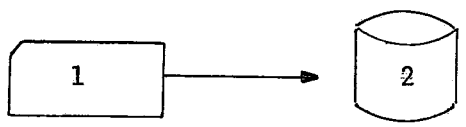
Algoritmo da CARGA
(Carga maciça propriamente dita)

```
início
  i := 1;
  Pegapágina (P1);
  enquanto  i ≤ nº de chaves existentes no arquivo
    faça % nível das folhas
      enquanto P1 não ocupada em 2/3 e  i ≤ nº de chaves.
        faça Leia registro (i) no arquivo);
             se a chave é numérica então transformá-la em alfa;
             Escreva a chave do registro em P;
             i := i+1;
      Pegapágina (P2);
      Encadeie P2 em P1;  % formará uma fila
      Grave P1 no arquivo índice;
      P1 := P2;
  enquanto tamanho da fila do nível anterior > 1
    faça % nível dos nós internos
      Pegapágina (P);
      Filha := início da fila do nível anterior;
      enquanto P não ocupada em 2/3 e Filha ≠ λ
        faça Pegapágina (Filha);
          se existe irmã à direita da Filha
              então
                  se Filha é folha
                    então
                      Construa um separador entre Filha e sua
                        irmã;
                      Escreva o separador em P;
                    senão
                      Escolha um bom separador entre Filha e sua
                        irmã;
                      Escreva o separador em P;
                      Retire o separador de sua página original;
          Filha := página próxima a Filha na fila do nível
                   anterior;
      Coloque a página P na fila deste nível;
      Grave P no arquivo índice;
  RAIZ := página começo da fila;
  Call BUSCA (última chave, não livre, erro); % para balancear
    o extremo.
  BALANCEAMENTO;
fim.
```

.. 02 - Procedure CRIA

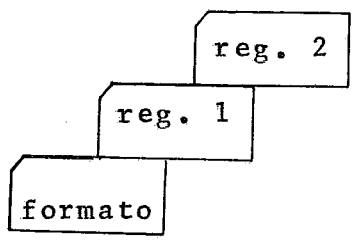
parâmetros: erro.

função: construir um arquivo em disco, a partir de um arquivo já existente em cartão (ordenado pela chave)



1 - arquivo cartão
2 - arquivo de registros

detalhe de 1:



Ações: Analisa o cartão formato, construindo uma tabela de formatos de modo a que o arquivo cartão possa ser "lido" pela subrotina;
Lê a massa de cartões, segundo a tabela formato, verificando se os mesmos estão em ordem, segundo o campo chave;
Interpreta os cartões, usando os formatos (procedure RETIRAVALOR) e prepara um "buffer" para o registro;
Grava os registros;

Obs.: A tabela de formatos é constituída por quatro linhas (índice, a, w, d) e o número de colunas é igual ao de formatos:

Exemplo: (I3,I5,2I2,F4.3,F5.1,E94,5X,A2,I3,F10.5) cartão formato

linha índice	1	1	1	2	2	3	4	5	1	2	códigos de formato: I=1
linha a	1	1	2	1	1	1	5	1	1	1	F=2
linha w	3	5	2	4	5	9	0	2	3	10	E=3
linha d	0	0	0	3	1	4	0	0	0	5	X=4
											A=5

Códigos de erro
0 - tudo bem
1 - erro de formato (caracter inválido)
2 - os cartões estão em ordem errada.

Procedure RETIRAVALOR

função: interpreta os cartões de dados segundo a tabela de formatos, armazena-os no "buffer" de impressão, e grava o "buffer" no disco, quando pronto para tal.

ação: Verifica qual o código que se encontra na coluna *i* da linha *índice* na tabela de formatos. O código pode levá-la a construir um número inteiro (código 1), um número real na forma exponencial (código 3) ou simples (código 2), armazenar um (ou dois) caracteres (código 5) ou simplesmente ignorar um determinado número de colunas do cartão (código 4). Quando ocorre a formação de um número (ou de uma parte de uma cadeia alfanumérica), ele é armazenado no "buffer" a ser gravado no disco, quando completo.

Procedure RETIRAVALOR

```
início
   para cada cartão do registro
     faça  ler cartão em formato 80A1
           PC := 1; FLAG := desligado;
           para I:=1 até TAM
           faça GO TO (10, 20, 30, 40, 50), INDICE (i)
        10 % inteiro
           L := A(i);
           repita   VALORINT := 0; SINAL := "+";
                    PC := "correr b"
                    SINAL := CARTAO (PC) ; PC := PC+1
                   para J:=PC até  W(i)
                     faça   VALORINT := NCARTAO(PC) + VALORINT*10;
                            PC := PC+1;
                     se sinal = "-" então VALORINT := - VALORINT;
                     se FLAG ligado
                        então Completo TEMP com b;
                              BUFFD(PI) := TEMP;
                              PI := PI+1;
                              desliga FLAG;
                     BUFFD(PI) := VALORINT;
                     PI := PI + 1;
                     L := L-1;
           até       L = 0;  % considerar a repetição do formato
           GO TO 60;

        40 % espaços
           PC := PC+W(i);
           GO TO 60;

        50 % cadeia
           L := A(i);
           repita   ALFA := CARTAO(PC);
                    se FLAG desligado
                         então  % 1º caracter lido
                                TEMP := ALFA;
                                liga FLAG;
                         senão  % 2º caracter lido
                                TEMP := TEMP, ALFA;  (em A2)
                                BUFFD := TEMP;
                                PI := PI+1;
                                desliga FLAG;
                    L := L-1;
           até L = 0;
           GO TO 60;
```

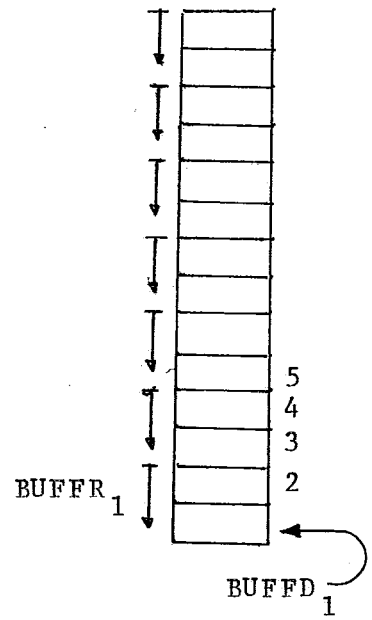
```
30  % exponencial
    L := A(i);
    repita  VALOREXP := 0; PONTO:0; DVSOR:=10; SINAL:= "+";
            VALOREAL := 0;
            PC := "correr b"; atualiza LMTE;
            SINAL := CARTAO (PC); PC:= PC+1;
            para  J:=PC até LMTE
                faça
                  se CARTAO(PC) = "."
                     então PONTO:=TRUE
                     senão se ¬ PONTO
                              então VALOREAL:=NCARTAO(PC)+
                                    10*VALOREAL;
                              senão VALOREAL:=NCARTAO(PC)+
                                    DVSOR*VALOREAL;
                                    VALOREAL:=VALOREAL/DVSOR;
                                    DVSOR := DVSOR 10;

                    PC := PC+1;
              se    SINAL = "-"
                    então  VALOREAL := - VALOREAL;
              se    CARTAO(PC) ≠ "E"
                    então  "erro"
                    senão  SINAL := CARTAO (PC+1);
                           PC := PC+2;
                           EXP:=10*NCARTAO(PC)+NCARTAO(PC+1);
                           PC := PC+2;
              se    SINAL = "+"
                    então  VALOREXP:=VALOREAL * 10**EXP
                    senão VALOREXP := VALOREAL/(10**EXP)
              se    FLAG ligado
                    então  Completo TEMP com b;
                           BUFFD(PI):=TEMP;
                           PI:=PI+1;
                           desliga FLAG;
              se    PI é par
                    então BUFFD(PI):=b;
                          PR:=PI/2+1;
                          PI:=PI+3;
                    senão PR := (PI+1)/2;
                          PI := PI+2;
            BUFFR(PR):=VALOREAL;
            L:=L-1;
    até L=0;
    GO TO 60;
```

```
20 % real
   L:=A(i);
   repita VALOREAL:=0; PONTO:=0; DVSOR:=10; SINAL:= "+";
          PC:= "correr b"; atualiza LMTE;
          SINAL:=CARTAO(PC); PC:=PC+1;
          para J:=PC até LMTE
           faça
            se CARTAO(PC) = "."
               então PONTO:=TRUE
               senão se ¬ PONTO
                        então VALOREAL:=NCARTAO(PC)+10*VALOREAL;
                        senão VALOREAL:=NCARTAO(PC)+DVSOR*
                                VALOREAL;
                              VALOREAL:=VALOREAL/DVSOR;
                              DVSOR:=DVSOR*10;
            PC:=PC+1;
          se SINAL = "-"
             então VALOREAL:= -VALOREAL;
          se FLAG ligado
             então completo TEMP com b;
                   BUFFD(PI):=TEMP;
                   PI:=PI+1; desliga FLAG;
          se PI é par
             então BUFFD(PI):=b
                   PR:=PI/2+1;
                   PI:=PI+3;
             senão PR:=(PI+1)/2;
                   PI:=PI+2;
           BUFFR(PR):=VALOREAL;
           L:=L-1;
   até L=0;
60 CONTINUE;
fim.
```

OBS.: TAM - nº de campos no cartão formato
NCARTAO(PC) - é o valor do CARTAO(PC) convertido em número.

Como preparar o buffer a ser gravado no disco?

```
                Real BUFFR ( )
                Integer BUFFD ( )
                Equivalence (BUFFR(1), BUFFD(2))
                PI := 1;
                PR := 1;
* se o valor é inteiro
    então    se FLAG ligado
                 então  completo TEMP com ♭;
                        BUFFD(PI) := TEMP;
                        PI := PI+1;
                        desliga FLAG,
        BUFFD(PI) := VALORINT;
        PI := PI+1;
* se o valor é real (formato real ou exponencial)
    então    se FLAG ligado
                 então  completo TEMP com ♭;
                        BUFFD(PI)+1;
                        desliga FLAG;
             se PI é par
                 então  BUFFD(PI):=♭;
                        PR:=PI/2+1;
                        PI:=PI+3;
                 senão  PR:=(PI+1)/2;
                        PI:=PI+2;
        BUFFR(PR):=VALOREAL
* se cadeia
    então    se FLAG desligado
                 então  % 1º caracter lido
                        coloca cadeia em TEMP;
                        liga FLAG;
                 senão  % 2º caracter lido
                        TEMP:=TEMP, V alfa; (em A2)
                        BUFFD:=TEMP;
                        PI:=PI+1;
                        desliga FLAG;
```



OBS.: No caso de se querer reservar posições no registro estas deverão ser lidas como alfa.

.. 03 - Opção entre o uso da CRIA e/ou CARGA

CRIA - Usa o arquivo cabeçalho (1) para saber em qual arquivo do disco deverá armazenar seus registros em cartão (arquivo 4)

CARGA - Usa o arquivo cabeçalho (1) para construir os cabeçalhos dos arquivos de registros (2) e de índices (3). Como já encontra os registros armazenados em ordem, segundo a chave principal, é a responsável pela construção do índice.

Se o usuário optar por usar unicamente a CARGA é responsável por fornecer os registros ordenados segundo a chave principal. Nesse caso a tabela de formatos não tem utilidade.

.. 04 - Procedure BUSCACHAVE

parâmetros: vetor chave, apontador para o registro, erro.

função: Pesquisa se determinada chave pertence ao índice e fornece ao usuário a posição do registro a ser lido ou indica erro.

Obs.1 - Dentro da página a busca da chave será feita usando-se a pesquisa binária.

Obs.2 - A página do índice em que estiver a chave é colocada na fila dos 3 "buffers".

Obs.3 - O caminho percorrido é armazenado numa pilha.

Obs.4 - Se a chave for numérica deverá estar armazenada na primeira posição do vetor chave, e será então transformada, nesta subrotina, para o código EBCDIC, sendo armazenados dois caracteres dígitos por palavra, ou seja, é interpretada como sendo uma alfanumérica de tamanho variável.

Códigos de erro
0 - tudo bem
1 - chave não encontrada no índice
2 - chave não encontrada no arquivo.

Algoritmo da BUSCACHAVE

```
início
     "condições iniciais";
     P := raiz;
     repita
        ler do arquivo de índices a página P;
        colocar P na pilha de páginas (do caminho percorrido)
        COMEÇO := 1;
        FIM := fator de ocupação da página P;
        repita
           "busca binária na página";
        até "Busca ◘ True" ou "descer de página";
        se pesquisa binária foi feita em página folha
            então
            se "Busca ◘ True"
                então ler registro do arquivo de informações;
                      se chave do registro ≠ chave procurada
                          então erro := 2;
                senão erro := 1 (chave não encontrada);
     até último nível da árvore (P ◘ λ)
fim.
```

.. 05 - Procedure INSERÇÃO

parâmetros: vetor chave, apontador para registro, condição da inserção, erro.

função:

Busca pela chave a ser inserida.

Se ela já consta da árvore então seta parâmetro de erro, senão localiza a posição adequada à inserção.

Localizada a posição em que deve ser feita a inserção, insere-se a chave no índice e a informação associada no arquivo de registros,

Obs. 1 : Já é garantido que haverá lugar para a inserção, e após cada inserção a árvore sofre balanceamento.

Obs. 2 : A inserção poderá ser feita só no índice (condição ≠ 1) ou no índice e no arquivo (condição = 1)

Obs. 3 : Compete ao usuário providenciar a inserção nos demais índices associados, chamando outras vezes a subrotina.

Obs. 4 : Se a chave for numérica deverá estar armazenada na primeira posição do vetor chave.

Código de erro
0 - tudo bem
1 - a chave já consta do índice
3 - falta espaço no índice
4 - falta espaço no arquivo.

Algoritmo da INSERÇÃO

```
início
      "condições iniciais"
      Call BUSCACHAVE (...,...., erro)
      Se erro = 1
         então  % chave não encontrada no índice
                Abrir espaço para o encaixe, afastando as demais
                 chaves e ponteiros para a direita;
                Atualizar o fator de ocupação e o número de cha-
                 ves na página;
                Se  condição da inserção = 1
                  então inserir o registro no arquivo;
                Atualizar o cabeçalho do índice e do arquivo de
                 registros quanto ao número de chaves e registros;
                BALANCEAMENTO da estrutura;
fim.
```

.. 06 - Procedure REMOÇÃO

parâmetros: vetor chave, erro.

função: Busca pela chave a ser removida.
Se ela não consta da árvore, então seta o parâmetro de erro, senão ela é removida do índice e o registro associado do arquivo*.

Obs. 1 - Após a remoção a árvore sofre balanceamento.

Obs. 2(*) - No caso de haver n > 1 índices associados ao arquivo, a remoção é feita apenas no índice.

Obs. 3 - Se a chave for numérica deverá estar armazenada na primeira posição do vetor chave.

Códigos de erro
0 - tudo bem
1 - a chave não consta do índice.
2 - a chave não consta do arquivo.

Algoritmo da REMOÇÃO

```
início
    "Condições iniciais"
    Call BUSCACHAVE (..., ..., erro);

    Se erro = 0
       então % tudo bem na busca
           Retirar a chave encontrada deslocando as demais à di-
            reita sobre a sua posição;
           se o número de índices do arquivo = 1
             então
               Acessa o campo no registro;
               Se  as chaves conferem
                 então
                    Retirar o registro do arquivo, deixando sua
                      posição disponível;
                 senão
                    erro := 2
           Atualizar o número de chaves e o fator de ocupação na
             página;
           Atualizar o cabeçalho (índice e arquivo de registros)
             quanto ao número de chaves.
           BALANCEAMENTO da estrutura;
    senão
       erro := 1;
fim.
```

.. 07 - Procedure BUSCAPRÓXIMO

parâmetros: apontador para o registro, erro.

função: Acessa a primeira chave imediatamente seguinte à acessada por último e fornece ao usuário a posição em que se encontra o registro associado à chave desejada.

Obs. 1 - Seu uso é posterior à procedure BUSCACHAVE ou a si própria.

Obs. 2 - As folhas do índice estão encadeadas no sentido do próximo.

Códigos de erro

0 - tudo bem

1 - não existe próximo

Algoritmo da BUSCAPRÓXIMO

```
início
  % APREG - apontador para o registro
  % CHAVE - chave acessada por último
  % P - folha acessada por último
  % PROX - chave próxima procurada
  "condições iniciais".
  se CHAVE não é a última em P
     então  PROX := chave posterior a CHAVE
     senão
       se P não é a última folha
          então
            P := folha próxima a P;
            PROX := primeira chave de P;
          senão
            ERRO := 1; % não existe próximo
  APREG := apontador para o registro;
fim.
```

.. 08 - Procedure BUSCANTERIOR

parâmetros: apontador para o registro, erro.

função: Acessa a primeira chave imediatamente anterior à acessada por último e fornece ao usuário a posição em que se encontra o registro associado à chave desejada.

Obs. 1 : Seu uso é posterior à Procedure BUSCACHAVE, ou a si própria.

Obs. 2 : Não existe ponteiro encadeando as folhas no sentido do anterior. O algoritmo trabalha conhecendo o caminho percorrido na fase de busca.

Obs. 3 : A pilha com registro do caminho percorrido é mantida atualizada.

Códigos de erro

0 - tudo bem

1 - não existe anterior

Algoritmo da BUSCANTERIOR

```
início
   % APREG - apontador para o registro
   % CHAVE - chave acessada por último
   % P - folha acessada por último
   % ANTER - chave anterior à acessada por último
   "condições iniciais"
   se CHAVE não é a primeira em P
     então ANTER := chave anterior a CHAVE
     senão
       se P não é a primeira folha
         então  "P" := folha anterior a P";
                ANTER := última chave de P;
         senão  ERRO := 1; % não existe anterior
   APREG := apontador para o registro
fim.
```

```
"P := folha anterior a P"
início
   enquanto  o apontador associado à chave (ou separador) pela
      pilha for o primeiro e página não for raiz
        faça  desempilhe a página e o apontador;
      se página é RAIZ e CHAVE é a primeira na página
        então ERRO := 1  % não existe anterior
        senão leia  página topo da pilha;
              encontre o apontador anterior;
              atualize a pilha de apontadores;
              enquanto página não folha
                  faça
                    atualize caminho percorrido;
                    desça pela última chave da página;
fim.
```

.. 09 - Procedure BUSCAPOSIÇÃO

parâmetros: posição desejada, apontador para o registro, erro.

função: Acessa a chave correspondente à posição desejada e fornece ao usuário a posição em que se encontra o registro associado à chave desejada.

idéia: É verificado se a posição desejada encontra-se na última folha, nesse caso a procura é feita diretamente nela, caso contrário começa-se a procura a partir da primeira folha, usando lógica semelhante à da subrotina BUSCAPRÓXIMO.

Obs. 1 - É possível o acesso nos dois sentidos,

1a. chave	e	última chave
2a. chave		penúltima chave
3a. chave		.
.		.
.		.
.		.

Obs. 2 - Seu uso é livre

Obs. 3 - A primeira e a última folha do índice são assinaladas por ponteiros específicos.

Códigos de erro:

0 - tudo bem

1 - não existe a posição no índice.

Algoritmo da BUSCAPOSIÇÃO

```
início
  % APREG - apontador para o registro
  % CHVPG - número de chaves na página
  % LUGAR - posição da chave que se deseja acessar
  % NOCHI - número atual de chaves no índice
  % PRIM  - apontador para a primeira folha
  % ULTMA - apontador para a última folha
"condições iniciais"
AUX := NOCHI - CHVPG(ULTMA) - LUGAR
se  AUX < 0
  então % a chave desejada está na última folha
     P := ULTMA;
     percorre P até alcançar LUGAR;
  senão  SOMACHV := 0
         P := PRIM;
         enquanto  SOMACHV<LUGAR
            faça  SOMACHV := SOMACHV + CHVPG(P);
                  P := folha próxima a P;
         percorre P até alcançar LUGAR;
  APREG := apontador para o registro;
fim.
```

.. 10 - Procedure BALANCEAMENTO

parâmetro: erro

função: Equilibra o fator de ocupação da página com uma sua irmã vizinha, cuidando para que sempre haja a possibilidade de inserção ou remoção de uma chave na página.
É usada após uma inserção ou remoção, e no final da carga maciça.

Código de erro
0 - tudo bem
3 - falta de página.

Algoritmo de BALANCEAMENTO

```
início
   Se não há condições para inserção e remoção
      (o espaço de folga não satisfaz)
      então % tentar redistribuição
         escolha uma página vizinha aleatoriamente;
         repita
            Se esta página vizinha existe
               então
                  Encontre na página mãe o separador entre as páginas
                     (filha e irmã);
                  Leia a página vizinha do arquivo de índices;
                  Se há condições de balanceamento entre as páginas
                     (filha e irmã)
                     então
                        REDISTRIBUIÇÃO
                     senão
                        troque de lado
                  senão
                     troque de lado;
               até ter tentado o balanceamento dos 2 lados da página
                     (filha)
                  ou  término do processo de redistribuição;
               Se não houve redistribuição e não há condições para
                  inserção.
                  então
                     DIVISÃO DE PÁGINA (em 2/3) e REDISTRIBUIÇÃO;
               Se não houve redistribuição e não há condições para
                  remoção
                  então repita
                     Se há condições para união das páginas
                        (filha e irmã)
                        então
                           UNIÃO DE PÁGINAS
                        senão
                           Troque de lado;
                     até ter tentado o balanceamento dos 2 lados da
                           página filha ou término do processo de
                           união de páginas;
fim.
```

REDISTRIBUIÇÃO

```
início
   Coloque no vetor de armazenamento a página à esquerda;
   se as páginas (filha e irmã) são folhas
      então
         Guarde o ponteiro de próxima página da página à direita;
      senão
         Coloque no vetor após a última posição ocupada, o separa
          dor da página mãe;
   Coloque no vetor, após a última posição ocupada, a página à di
    reita;
   Ache a chave que ocupa a posição central do vetor;
   Procure à esquerda e à direita da chave central o menor separa
    dor entre as chaves, restringindo-se ao limite de balanceamen
    to entre as páginas;
   Atualize o número de chaves e o fator de ocupação de cada uma
    das páginas;
   Grave as páginas balanceadas;
   Atualize na página mãe o separador entre as páginas balancea-
    das e o fator de ocupação;
   Grave a página mãe,
fim.
```

DIVISÃO DE PÁGINAS

```
início
  P := página filha;
  se P é RAIZ
     então
       Divida a página ao meio para criar uma segunda página;
       Pegapágina (P1);
       Distribua as metades entre as páginas P e P1;
       Pegapágina (P2);
       Coloque o separador em P2;
       RAIZ := P2;
     senão
       Pegapágina (P1);
       Divida a página P em 2/3 com P1 de modo a possibilitar a
        redistribuição de P1 com uma irmã vizinha de P;
       Grave a página maior (P ou P1);
       Redistribua a página menor com a irmã vizinha escolhida;
fim.
```

UNIÃO DE PÁGINAS

```
início
  Coloque no vetor de armazenamento a página à esquerda;
  se as páginas (filha e irmã vizinha) são folhas
     então
       Guarde o ponteiro de próxima página da página à direita
     senão
       Coloque no vetor, após a última posição ocupada, o separa
        dor entre elas na página mãe;
  Coloque no vetor, após a última posição ocupada, o conteúdo da
   página à direita;
  Remova da página mãe o separador das páginas unidas atualizan-
   do o seu fator de ocupação e seu número de chaves;
  Grave a página resultante da união (o nº é o da esquerda) e li
   bere a página direita;
  se página mãe é RAIZ e esvaziou com a retirada do separador.
     então
       Libere a página RAIZ;
       RAIZ := página filha;
     senão
       Grave RAIZ;
fim.
```

Obs.: Esta subrotina atualiza o apontador da última folha.

.. 11 - Subprogramas Auxiliares Definidos

Subrotina BYTE (palavra, byte1, byte2)
  função: separa uma palavra em seus bytes, armazenados em duas palavras distintas (como primeiro byte).
  uso : na subrotina BUSCACHAVE
Subrotina WORD (byte1, byte2, palavra)
  função: junta dois bytes (os primeiros) de palavras distintas numa única palavra.
  uso : na subrotina CRIA.
Subrotina NALFA (número, alfa)
  função: transforma um número (inteiro positivo menor ou igual a 32767) na sua cadeia de caracteres, armazenada no vetor alfa, com dois caracteres por palavras, ajustados à esquerda.
  uso : nas subrotinas CARGA e BUSCACHAVE
Subrotina BYT1 (A)
  função: transforma um dígito decimal no caracter EBCDIC correspondente e o armazena no primeiro byte de A.
  uso : na subrotina NALFA.
Função NUMER (A)
  função: transforma um caracter número em número.
  uso : na subrotina CRIA.
Subrotina DVPAG
          PGPAG (erro)
  função: controla o uso da pilha de páginas disponíveis.
  uso : nas subrotinas CARGA, BALANCEAMENTO, INSERÇÃO e REMOÇÃO.
Subrotina DVREG
          PVREG (erro)
  função: controla o uso da pilha de registros disponíveis.
  uso : nas subrotinas CARGA, INSER e REMOV.
Função RANDO(K)
  função: gera um número aleatório.
  uso : na subrotina BALANCEAMENTO.

IV - CONCLUSÕES

Foi possível desenvolver e colocar à disposição dos usuários da linguagem FORTRAN uma estrutura moderna de índice que lhes permite acessar arquivos residentes em discos magnéticos de maneira aleatória ou sequencial. A estrutura escolhida, uma variação da árvore B, pode ser implementada numa máquina pequena, um IBM 1130 com 16 K de memória, ocupando a maior de suas subrotinas cerca de 6 K.

As subrotinas básicas de um sistema de acesso a índice são as de busca, inserção e remoção. Subrotinas especiais foram desenvolvidas nessa implementação como as de busca ao próximo, busca ao anterior e busca por posição, além de uma subrotina capaz de fazer a carga inicial do índice, evitando a inserção um a um.

O sistema apresenta uma boa portabilidade tendo sido desenvolvido em linguagem FORTRAN do IBM 1130, salvo três subrotinas auxiliares de manipulação de bytes que estão em assembler, o que aumenta as possibilidades de sua utilização prática.

Quando da definição do projeto algumas idéias surgidas foram abandonadas, por outras mais práticas, e vemos agora, a possibilidade de desenvolver outros trabalhos sobre ele, tais como:

a) O desenvolvimento de pré-processadores, o que facilitaria o uso do índice pelo usuário;
b) A incorporação da subrotina CRIA ao sistema que, conforme foi definida, assumiria o trabalho de construir o arquivo ordenado em disco;
c) Avaliação melhor do sistema para diferentes tamanhos de arquivo.

VI - REFERÊNCIAS BIBLIOGRÁFICAS

01 - BAYER, R. - "Symetric binary B-trees: Data structure and maintenance algorithms". Acta Informatica, vol.: 290-306, 1972.

02 - BAYER, R. & McCREIGHT, E. - "Organization and Maintenance of large ordered indexes". Acta Informatica, 1:290-306, 1972.

03 - BAYER, R. & SCHKOLNICK, M. - "Concurrency of operations on B-trees". Acta Informatica, 9:1-21, jan. 1977.

04 - BAYER, R. & UNTERAUER, K. - "Prefixe B-trees". ACM Transaction Database System, 2(1):11-26, mar. 1977.

05 - BROWN, M. - "A Storage Scheme for heigt-balanced trees". Information Processing Letters, 7(5):231-232, ago. 1978.

06 - COMER, D. - "The ubiquitous B-tree". ACM Computing Surveys, 11(2):121-137, jun. 1979.

07 - HELD, G. & STONEBRAKER, M. - "B-trees Re-examined". Comm. ACM, 21(2):139-143, fev. 1978.

08 - "IBM OS/VS Virtual Storage Access Method (VSAM) Planning Guide", Form. nº GC26-3799. IBM Corporation, Data Processing Division. White Plains, New York.

09 - "IBM SYSTEM/360 Operating System Indexed Sequential Access Method", Form. nº Y28-6618. IBM Corporation, Data Processing Division. White Plains, New York.

10 - IBM. "Introduction to IBM Direct - Access Storage Devices and Organization Methods". IBM Corporation, Data Processing division. White Plains, New York.

11 - IBM. "DOS/VS Data Management Guide", Form. GC33-5372-3. IBM Corporation, Data Processing Division. White Plains, New York.

12 - IBM "1130 Disk Monitor System, Version 2, Programmer's and Operator's Guide", Form. nº GC26-3717-9. IBM Corporation, Data Processing Division. White Plains, New York.

13 - KEEHN, D.G. & LACY, J.O. - "VSAM data set design parameters". IBM System Journal, 13(3):186-213, 1974.

14 - KNUTH, D.E. - "The Art of Computer Programming, V.3 - Sorting and Searching". Addison Wesley, Publ. Co. Reading, Mass., 1973.

15 - MALY, K. - "A Note on Virtual Memory Indexes". Comm. ACM, 21/9:786-787, set. 1978.

16 - MARUYAMA, K. & SMITH, S.E. - "Analysis of design alternatives for virtual memory indexes". Comm. ACM. 20(4):245-254, abr. 1977.

17 - McCREIGH, E.M. - "Pagination of B* Trees with variable lenght records". Comm. ACM, 20(9):670, set. 1977.

18 - MILLER, R. & PIPPENGER, N. & ROSENBERG, A. & SNYDER, L. "Optimal 2-3 trees". SIAN J. Of Computing, 8(1):42-59. fev. 1979.

19 - ROSEMBERG, A. & SNYDER, L. - "Minimal Comparison 2-3 trees". SIAN J. COMPUT., 7(4):465-480, nov. 1978.

20 - RUBIN, F. "Experiments in Text File Compression". Comm. ACM, 19(11):617-623, nov. 1976.

21 - SAMADI, B. - "B-trees in a system with multiple users". Information Processing Letters, 5(4):107-112, out. 1976.

22 - SEVERENCE, D. - "Identifier Search Mechanisms: a survey and generalized model", Computing Surveys, 6(3): 175-194, set. 1974.

23 - SOUZA, J. M. & TORRES, E.F. - "Um estudo sobre árvores-B e suas variações". Anais do I Simpósio de Computação da Universidade Federal de São Carlos, 1980.

24 - WAGNER, R.E. - "Indexed Design Considerations". IBM Systems Journal, 12(4):351-367, 1973.

25 - WEDEKIND, H. - On the selection of access paths in a Data Base System. Data Base Management, North-Holland Publishing Company, Klimbie and Koffeman, :385-397, 1974.

26 - YAO, A. - "On randon 3-2 trees". Acta Informatica; 2:159-170, 1978.

ANEXOS

1 - Características do Computador Hospedeiro

2 - Determinação do k

3 - Listagens

4 - Manual do usuário

ANEXO 01

CARACTERÍSTICAS DO COMPUTADOR HOSPEDEIRO

Sistema - IBM 1130 com 16 k

UCP 1131 com ciclo de 3,6 μs

unidade de disco - 2310

modelo do disco - 2315

200 cilindros

400 trilhas

4 setores por trilha

320 palavras por setor

16 bits por palavra

1500 rpm

retardo rotacional médio - 20 ms (1/2 rotação)

estabilização média - 22,5 ms

deslocamento entre cilindros - 7,5 ms

ler/escrever 1 palavra - 27,8 μs

$$\text{NS (nº médio de seeks por acesso)} = \frac{\sum_{i=1}^{200} \sum_{j=1}^{200} |i-j|}{40.000} \simeq 66,6$$

TM (tempo médio por acesso) = NS * 7,5 ms $\simeq$ 500 ms

ANEXO 02

DETERMINAÇÃO DO k

Como já foi visto anteriormente (seção II-4-2-1) as classes de árvores B são caracterizadas pelos dois parâmetros k e h, onde a altura h é determinada em função do tamanho do índice (I) e do grau de ramificação da árvore, associado a k.

Para a determinação desse último parâmetro (k) vamos fazer algumas considerações:

a) O tempo gasto com cada página a ser lida ou escrita pode ser expresso como:

$$\alpha+\beta(2k+1)+\gamma\ln(\nu k+1)$$

onde:

α - o tempo fixo gasto por página (tempo médio de "seek" mais CPU "overhead", etc.)

β - tempo de transferência por entrada da página

γ - constante para a parte logarítmica do tempo (busca binária)

ν - fator de ocupação média $1<\nu<2$

b) O número médio de páginas a serem lidas ou escritas, por transação individual, num processamento envolvendo buscas, inserções ou remoções é aproximadamente proporcional, conforme BAYER e McCREIGHT[02], a h, digamos δh. O tempo total gasto por transação pode ser então aproximado para:

$T \simeq \delta h(\alpha+\beta(2k+1)+\gamma\ln(\nu k+1))$

Fazendo para h a aproximação

$h \simeq \log_{k+1}(I+1)$, onde I é o tamanho do índice, nós teremos:

$$T \simeq T_a = \delta\log_{\nu k+1}(I+1)(\alpha+\beta(2k+1)+\gamma\ln(\nu k+1))$$

derivando-a, com relação a k:

$$y=\log_{\nu k+1}(I+1) \Rightarrow y' = - y \frac{(\nu k+1)'}{(\nu k+1)\ln(\nu k+1)} = \frac{-y.\nu}{(\nu k+1)\ln(\nu k+1)}$$

$$T_a' = \delta\left[y'(\alpha+\beta(2k+1)+\gamma\ln(\nu k+1)) + y\ (2\beta+\gamma\frac{\nu}{\nu k+1})\right]$$

Teremos um mínimo para k se:

$$\frac{\nu\ (\alpha+\beta(2k+1) + \gamma\ln(\nu k+1))}{(\nu k+1)\ \ln(\nu k+1)} = \frac{2\beta(\nu k+1) + \gamma\nu}{(\nu k+1)}$$

$$\frac{\nu(\alpha+\beta(2k+1))}{\ln(\nu k+1)} + \gamma\nu = 2\beta(\nu k+1) + \gamma\nu$$

$$\alpha+\beta\ (2k+1) = 2\beta\frac{(\nu k+1)}{\nu}\ln(\nu k+1)$$

$$\frac{\alpha}{\beta} = 2\frac{(\nu k+1)}{\nu}\ln(\nu k+1) - (2k+1) = f(k,\nu)$$

Vemos abaixo uma tabulação para a função f(k,ν), extraída de BAYER e McCREIGHT[02].

k	f(k,1)	f(k,(1,5))	f(k,2)
2.00000 E+00	1.59167 E+00	2.39353 E+00	3.04718 E+00
4.00000 E+00	7.09437 E+00	9.16182 E+00	1.07750 E+01
8.00000 E+00	2.25500 E+01	2.74591 E+01	3.11646 E+01
1.60000 E+01	6.33292 E+01	7.42958 E+01	8.23847 E+01
3.20000 E+01	1.65769 E+02	1.89265 E+02	2.06334 E+02
6.40000 E+01	4.13670 E+02	4.62662 E+02	4.97915 E+02
1.28000 E+02	9.96831 E+02	1.09726 E+03	1.16911 E+03
2.56000 E+02	2.33922 E+03	2.54299 E+03	2.68826 E+03
5.12000 E+02	5.37752 E+03	5.78842 E+03	6.08075 E+03
1.02400 E+03	1.21625 E+04	1.29881 E+04	1.35748 E+04
2.04800 E+03	2.71506 E+04	2.88062 E+04	2.99818 E+04
4.09600 E+03	5.99647 E+04	6.32806 E+04	6.56343 E+04
8.19200 E+03	1.31269 E+05	1.37906 E+05	1.42617 E+05
1.63840 E+04	2.85235 E+05	2.98514 E+05	3.07938 E+05
3.27680 E+04	6.15877 E+05	6.42442 E+05	6.61292 E+05
6.55360 E+04	1.32258 E+06	1.37572 E+06	1.41342 E+06

Como a árvore usada é uma B^+ com prefixo simples precisamos definir o alfabeto. Vamos considerar um alfabeto de cardinalidade $\alpha=36$ (letras e números) e um arquivo de tamanho $n=10^4$. Segundo BAYER e UNTERAUER[04] o tamanho médio dos separadores será 2,884 bytes.

Quando consideramos as folhas internas da árvore (onde estão os separadores) temos cada entrada da página com tamanho igual a três (uma palavra para ponteiro e duas para o separador). Teremos então:

α = tempo fixo gasto por página

= TM (tempo médio por acesso) + (estabilização) + (latência)

CPU overhead

- (500+22,5+20) ms
- = 542,5 ms

β = tempo de transferência por entrada da página

- 3 x 27,8 μs
- = 0,0834 ms

portanto

$\alpha/\beta = 6{,}50480 \times 10^3$

Com o uso da tabela apresentada anteriormente podemos interpolar, obtendo os seguintes valores para K, dado ν

ν	K	2 x K
1,0	$6{,}19330 \times 10^2$	$1{,}23866 \times 10^3$
1,5	$5{,}75370 \times 10^2$	$1{,}15074 \times 10^3$
2,0	$5{,}47710 \times 10^2$	$1{,}09542 \times 10^3$

Como cada entrada ocupa 3 palavras, e o disco utilizado tem setores com 320 palavras, o tamanho ótimo da página exigiria um número de setores igual a:

ν	nº de setores necessários
1,0	11,612
1,5	10,788
2,0	10,26

Já vimos que o tempo total gasto por transação numa árvore B com chaves de tamanho constante, pode ser representado por:

$$T \simeq T_a = \delta \log_{\nu k+1} (I+1) (\alpha+\beta (2k+1) + \gamma \ln (\nu K+1))$$

onde I representa o tamanho do índice. Neste caso, como estamos considerando a árvore B^+ com prefixo simples, interessa-nos apenas as folhas internas (que constitui uma árvore B com chaves de tamanho variável).

Para traçarmos um gráfico do tempo gasto em função do K, vamos considerar $I=10^4$, $\nu=1,5$ e as constantes γ e δ igual a um. Observamos então que o valor real considerado para k (53) é 32% pior que o valor ótimo (575).

A tabela A2-1 demonstra a relação existente entre o número de setores, o K e o tempo (T) total gasto por transação.

Nº de Setores	1	3	5	10	11	20	39	77
K	53	128	256	512	575	1024	2048	4096
T	1162	996	916	882	878	908	1026	1309

↑ real

↑ ótimo

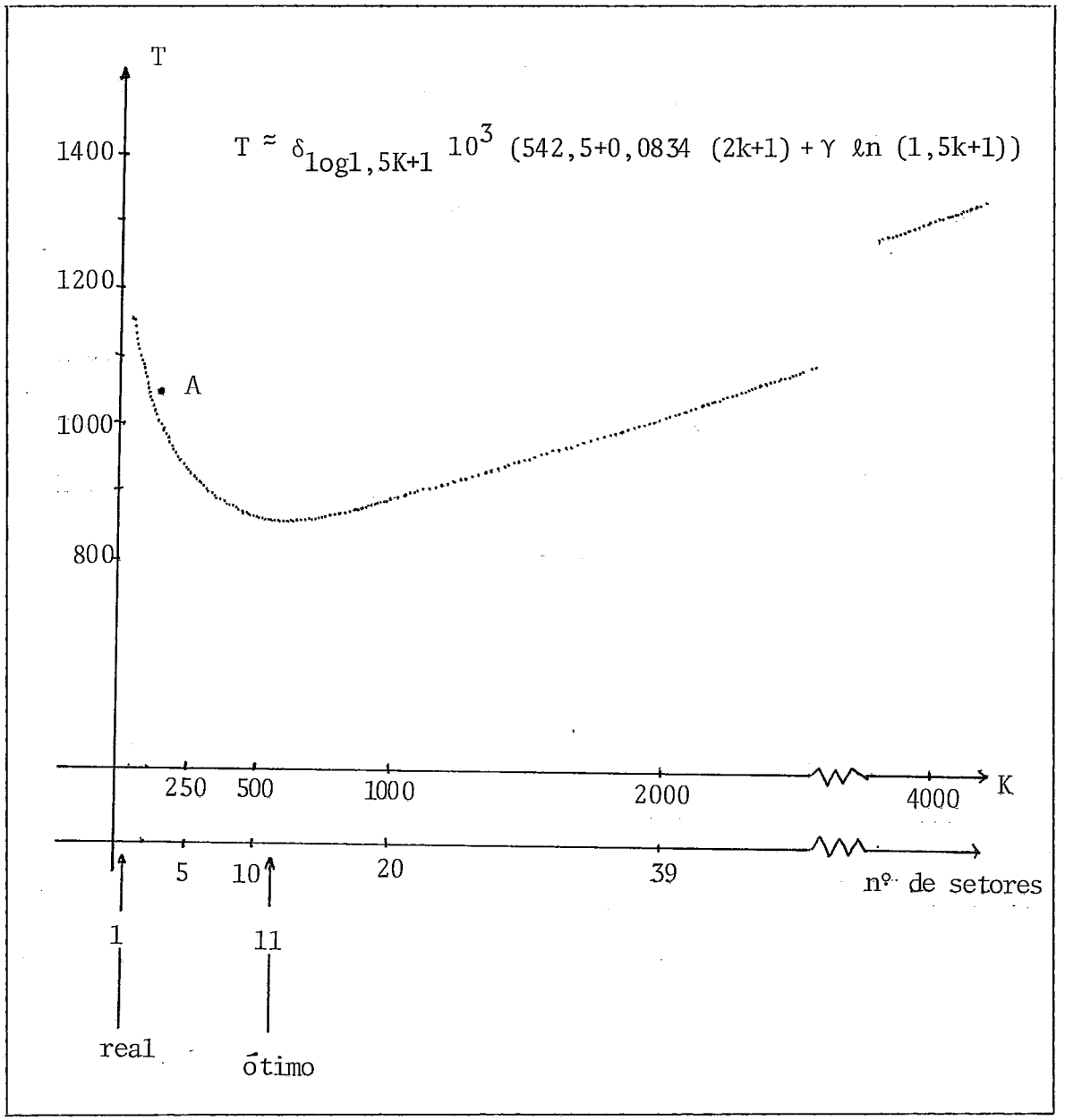


Gráfico A2-1 - Relação entre Tempo, K e Número de Setores.

Obs.: O ponto A representa o tempo gasto se o tamanho da página ocupasse dois setores.

ANEXO 03

LISTAGENS DO SISTEMA

```
PAGE   1

// JOB    0005

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      SUBROUTINE BUSCA(CHAVE,APREG,ERRO)
C     ALFA  = INDICA O TIPO DA CHAVE (1 SE ALFA, SENAO NUMERICA )
C     APBUF = APONTADOR DO BUFF
C     APCHV = APONTADOR DA CHAVE
C     APREG = APONTADOR DO REGISTRO
C     BYTB1 = PRIMEIRO BYTE DA POSICAO INDICADA DENTRO DO BUFF
C     BYTB2 = SEGUNDO  BYTE DA POSICAO INDICADA DENTRO DO BUFF
C     BYTC1 = PRIMEIRO BYTE DA POSICAO INDICADA DENTRO DA CHAVE
C     BYTC2 = SEGUNDO  BYTE DA POSICAO INDICADA DENTRO DA CHAVE
C     BRANC = BRANCO
C     BUFF  = CONTEM AS CHAVES/SEPARADORES E PONTEIROS DE UMA DETERMINADA PAGINA
C     CHAUX = VETOR PARAMETRO DE SAIDA DA SUBROTINA NALFA
C     CHAVE = VETOR CHAVE PROCURADA
C     CHV   = INDICADOR NA POSICAO DO CAMPO CHAVE
C     COMEC = INDICA A POSICAO INICIAL DENTRO DO BUFF NA PESQUISA BINARIA
C     DISPI = APONTA P/ O TOPO DA PILHA DE PAGINAS DISPONIVEIS
C     DISPR = APONTA P/ O TOPO DA PILHA DE REGISTROS DISPONIVEIS
C     FIM   = INDICA A POSICAO FINAL DENTRO DO BUFF NA PESQUISA BINARIA
C     FTOCP = FATOR DE OCUPACAO
C     M     = MOSTRA SE A PAGINA E FILHA (M=1) OU NO INTERNO (M=0)
C     MEIO  = APONTA A POSICAO CENTRAL ENTRE O COMEC E FIM
C     N     = PARAMETRO DE ENTRADA DA SUBROTINA NALFA
C     NOARC = NUMERO DO ARQUIVO CABECALHO
C     NOARI = NUMERO DO ARQUIVO DE INDICES
C     NOARR = NUMERO DO ARQUIVO DE REGISTROS
C     NOCHV = NUMERO DE CHAVES NA PAGINA
C     NOME  = PARAMETRO DE ENTRADA DA SUBROTINA BYTE
C     NOPAG = NUMERO DA PAGINA
C     NOREG = NUMERO DE REGISTROS
C     PAGIN = NUMERO DA PAGINA
C     PLAPT = PILHA DE APONTADORES (PELO QUAL 'DESCEU' O ALGORITMO DE BUSCA)
C     PLPAG = PILHA DE PAGINAS (CAMINHO PERCORRIDO)
C     PLPDB = PILHA PARA BALANCEAMENTO ( SE POSITIVO, SERA' BALANCEADO)
C     RAIZ  = APONTADOR P/ A RAIZ DA ARVORE-INDICE
C     REGIS = REGISTRO DO ARQUIVO DE INFORMACOES
C     TABCP = TABELA DE POSICOES INICIAIS DOS CAMPOS
C     TMCMP = VETOR CONTENDO O TAMANHO DOS CAMPOS
C     TMPAG = TAMANHO DA PAGINA
C     TMREG = TAMANO DO REGISTRO
C     TOPO  = INDICA O TOPO DA PLAPT E PLPAG
C
      INTEGER APBUF,APCHV,APREG,BRANC,BYTB1,BYTB2,BYTC1,BYTC2,COMEC,
     *        ERRO,FATOR,FILHA,FIM,NOME
      INTEGER CHAUX,CHAVE(8)
C
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *         NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *         TMREG,TOPO,ULTMA
      INTEGER BUFF(3,215),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *         PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
```

```
PAGE    2
        COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
       *         NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
       *          TMCHV,TMPAG,TMREG,TOPO,ULTMA
        COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
       *         REGIS,TABCP,TMCMP
C
        DATA BRANC/' '/,FILHA/1/

        ERRO = 0
        TOPO = 0
        IF(ALFA-1) 3,4,4
    3   CHAUX = CHAVE(1)
        CALL NALFA(CHAUX,CHAVE)
    4   IF(NOCHI) 570,570,5
C . . P = RAIZ
    5   PAGIN=RAIZ
C
C . . REPITA
C . . . LER A PAGINA INDICADA PELO PONTEIRO DE PAGINAS
    6   READ(NOARI'PAGIN)NOPAG(FILHA),NOCHV(FILHA),FATOR,M(FILHA),
       *                    (BUFF(FILHA,I),I=1,FATOR)
        FTOCP(FILHA) = FATOR
        COMEC=1
        FIM=FTOCP(FILHA)
   10   MEIO=(COMEC+FIM)/2
   20   IF(BUFF(FILHA,MEIO))30,40,40
   30      MEIO=MEIO-1
           GO TO 20
   40   IF(BUFF(FILHA,MEIO)-BRANC)43,30,30
   43   APBUF=MEIO+1
        APCHV=1
C . . . ENQUANTO CHAVE PROCURADA NAO MENOR QUE UMA DETERMINADA CHAVE DA PAGINA
C          LIDA   E   MAIOR QUE A SUA MAIS PROXIMA VIZINHA ANTERIOR   OU
C                  CHAVE PROCURADA MAIOR QUE UMA DETERMINADA CHAVE DA PAGINA
C          LIDA   E   MENOR QUE A SUA MAIS PROXIMA VIZINHA POSTERIOR FACA
C . . . . . 'BUSCA BINARIA' BYTE A BYTE
   45   NOME=BUFF(FILHA,APBUF)
        CALL BYTE(NOME,BYTB1,BYTB2)
        NOME=CHAVE(APCHV)
        CALL BYTE(NOME,BYTC1,BYTC2)
        IF(BYTC1-BYTB1)50,200,350
   50      PAGIN=MEIO
   60      IF(MEIO - 1) 500,500,65
   65      MEIO=MEIO-1
           IF(BUFF(FILHA,MEIO))60,60,70
   70   IF(BUFF(FILHA,MEIO)-BRANC)73,60,60
   73      APCHV=1
           APBUF=MEIO+1
   75      NOME=BUFF(FILHA,APBUF)
           CALL BYTE(NOME,BYTB1,BYTB2)
           NOME=CHAVE(APCHV)
           CALL BYTE(NOME,BYTC1,BYTC2)
           IF(BYTC1-BYTB1)80,110,500
   80         FIM=MEIO
              IF(FIM-COMEC)100,100,10
  100            PAGIN=MEIO
                 GO TO 500
  110         IF(BYTC2-BYTB2)120,130,160
  120            IF(BYTB2-BRANC)80,500,500
  130            IF(BYTC2-BRANC)140,500,500
```

```
PAGE    3
  140               APCHV=APCHV+1
                    APBUF=APBUF+1
                    IF(BUFF(FILHA,APBUF)) 150,150,500
  150               IF(CHAVE(APCHV)-BRANC)075,100,100
  160             IF(BYTC2-BRANC)500,80,80
C
  200       IF(BYTC2-BYTB2)50,210,240
  210          IF(BYTC2-BRANC)220,350,350
  220             APCHV=APCHV+1
                  APBUF=APBUF+1
                  IF(BUFF(FILHA,APBUF))230,230,350
  230             IF(CHAVE(APCHV)-BRANC)045,50,50
  240          IF(BYTC2-BRANC)350,50,50
  350       MEIO=MEIO+1
            IF(BUFF(FILHA,MEIO))350,350,355
  355       IF(BUFF(FILHA,MEIO)-BRANC) 360,350,350
  360       IF(MEIO-FIM)370,365,365
  365          PAGIN=MEIO
               GO TO 500
  370       PAGIN=MEIO
            APCHV = 1
            APBUF=MEIO+1
  375       NOME=BUFF(FILHA,APBUF)
            CALL BYTE(NOME,BYTB1,BYTB2)
            NOME=CHAVE(APCHV)
            CALL BYTE(NOME,BYTC1,BYTC2)
            IF(BYTC1-BYTB1)500,380,450
  380          IF(BYTC2-BYTB2)390,400,430
  390             IF(BYTB2-BRANC)500,450,450
  400             IF(BYTC2-BRANC)410,405,405
  405                PAGIN = PAGIN + 1
                     IF(BUFF(FILHA,PAGIN))405,405,500
  410                APCHV=APCHV+1
                     APBUF=APBUF+1
                     IF(BUFF(FILHA,APBUF))420,420,450
  420                   IF(CHAVE(APCHV)-BRANC)375,500,500
  430             IF(BYTC2-BRANC)450,500,500
  450          COMEC=MEIO
               GO TO 10
C
C . . .  COLOCAR P NA PILHA DE PAGINAS (DO CAMINHO PERCORRIDO)
  500  TOPO=TOPO+1
       PLPAG(TOPO)=NOPAG(FILHA)
       PLAPT(TOPO)=PAGIN

       IF(FTOCP(FILHA)+2*(TMCMP(CHV)+1)-(TMPAG-5))501,501,506
  501  IF(FTOCP(FILHA)-(TMCMP(CHV)+1)-(TMPAG-5)/2)504,507,507
  504  IF(NOPAG(FILHA)-RAIZ)506,507,506
  506  PLPDR(TOPO)=1
       GO TO 502
  507  PLPDR(TOPO)=0
C . . .  SE A PAGINA E FOLHA
  502  IF(M(FILHA))503,503,505
  503  PAGIN=BUFF(FILHA,PAGIN)
       GO TO 6
  505     IF(PAGIN - 1) 570,570,510
  510     APCHV=1
          APBUF = PAGIN
          IF(TMCHV)512,511,512
  511        APBUF = APBUF - TMCMP(CHV)
```

```
PAGE    4
                GO TO 514
   512          APBUF = APBUF - 1
                IF(BUFF(FILHA,APBUF - 1)) 512,512,514
   514       PAGIN = BUFF(FILHA,PAGIN)
             MEIO = APBUF - 1
C . . . .  ENTAO SE CHAVE PROCURADA = CHAVE ENCONTRADA PELA PESQUISA BINARIA
   515       IF(BUFF(FILHA,APBUF)-CHAVE(APCHV))570,520,570
   520          APCHV=APCHV+1
                APBUF=APBUF+1
                IF(APCHV-TMCMP(CHV))525,525,530
   525             IF(TMCHV)528,515,528
   528                   IF(BUFF(FILHA,APBUF))515,515,529
   529                      IF(CHAVE(APCHV)-BRANC)570,530,570
C . . . . . .  ENTAO LER REGISTRO DO ARQUIVO DE INFORMACOES INDICADO
   530             READ(NOARR'PAGIN)(REGIS(I),I=1,TMREG)
                   APCHV=1
                   APREG=TABCP(CHV)
         K = APREG + TMCMP(CHV) - 1

C . . . . . SE A CHAVE DO REGISTRO E' NUMERICA
                IF(ALFA-1) 531,535,531
   531             N = REGIS(APREG)
                   IF(N-CHAUX) 532,533,532
   532                ERRO = 2
   533                RETURN
   535             IF(CHAVE(APCHV)-REGIS(APREG))560,540,560
   540                APCHV=APCHV+1
                      APREG=APREG+1
                      IF(APCHV-TMCMP(CHV))535,535,550
C . . . . . FIM DA BUSCA
   550                APREG = PAGIN
         RETURN
C . . . . . SENAO ERRO = 2 (CHAVE NAO ENCONTRADA NO ARQUIVO)
   560             ERRO = 2
         RETURN
C . . . . . SENAO ERRO = 1 (CHAVE NAO ENCONTRADA NO INDICE)
   570       ERRO=1
             MEIO = PLAPT(TOPO)
         RETURN
C . . ATE ULTIMO NIVEL DA ARVORE
         END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR BUSCA
 COMMON   1130  VARIABLES     22  PROGRAM   1040

RELATIVE ENTRY POINT ADDRESS IS 001A (HEX)

END OF COMPILATION

// DUP

*DELETE             BUSCA
CART ID 0005   DB ADDR  27CC   DB CNT   0040

*STORE      WS  UA  BUSCA
CART ID 0005   DB ADDR  287A   DB CNT   0040
```

```
PAGE   1
// JOB    0005

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
C
      SUBROUTINE CARGA(TIPO,ERRO)
C
C   ALFA  = INDICA O TIPO DA CHAVE (1 SE ALFA, SENAO NUMERICA )
C   APAUX = APONT. AUXILIAR P/ A POS. DO SEPARADOR ANTERIOR AO INDICADO POR APSEP
C   APBUF = APONTADOR DO BUFF
C   APMEN = APONTADOR PARA O MENOR SEPARADOR
C   APSEP = APONTADOR PARA O SEPARADOR
C   AUX   = VARIAVEL AUXILIAR
C   AUX1  = VARIAVEL AUXILIAR
C   BRANC = POSICAO EM BRANCO
C   BUFF  = CONTEM AS CHAVES/SEPARADORES E PONTEIROS DE UMA DETERMINADA PAGINA
C   CHV   = NUMERO DO CAMPO DA CHAVE
C   COMEC = APONTADOR DE COMECO DA FILA CIRCULAR
C   CONT  = CONTADOR DE CHAVES
C   DISPI = APONTADOR DA PILHA DE PAGINAS DISPONIVEIS
C   DISPR = APONTA P/ O TOPO DA PILHA DE REGISTROS DISPONIVEIS
C   DOCUM = PALAVRAS RESERVADAS PARA INFORMACOES DOCUMENTACIONAIS
C   ERRO  = INDICADOR DE ERRO
C   FILHA = NIVEL DAS PAGINAS FOLHAS = FILHAS
C   FTOCP = FATOR DE OCUPACAO DA PAGINA
C   FIM   = APONTADOR DE FIM DA FILA CIRCULAR
C   I     = INDICA A PAGINA FILHA COM QUE SE ESTA TRABALHANDO
C   INDCS = VETOR COM OS NUMEROS DOS INDICES ASSOCIADOS
C   IRMA  = INDICA A LINHA QUE CONTEM AS INFORMACOES DA PAGINA VIZINHA (IRMA)
C   LADO  = INDICA QUAL A PAGINA (FILHA OU IRMA) QUE CONTEM O MENOR SEPARADOR
C   M     = MOSTRA SE A PAGINA INDICADA (FILHA,IRMA,MAE) E' FILHA OU NO' INTERNO
C   MAE   = NUMERO DAS PAGINAS MAES
C   MAXPG = NUMERO MAXIMO DE PAGINAS DO INDICE
C   N     = PARAMETRO (CHAVE NUMERICA) DA SUBROTINA NALFA
C   NOARI = NUMERO DO ARQUIVO DE INDICES
C   NOARC = NUMERO DO ARQUIVO CABECALHO
C   NOARR = NUMERO DO ARQUIVO DE REGISTROS(INFORMACAO)
C   NOCHA = NUMERO DE CHAVES PRESENTES NAS 2 PAGINAS (IRMA E FILHA)
C   NOCHI = NUMERO ATUAL DE CHAVES NO INDICE
C   NOCHV = NUMERO DE CHAVES PRESENTES NA PAGINA
C   NOCMP = NUMERO DE CAMPOS NO REGISTRO
C   NOIND = NUMERO DE INDICES ASSOCIADOS AO ARQUIVO
C   NOPAG = NUMERO DA PAGINA INDICADA
C   NOREG = NUMERO DE REGISTROS
C   PAGIN = APONTA PARA PAGINA DISPONIVEL
C   PGPAG = SUBROTINA DE PEGA PAGINA DISPONIVEL
C   PLAPT = PILHA COM OS PONTEIROS PELOS QUAIS 'DESCEU' O ALGORITMO DE BUSCA
C   PRIM  = APONTADOR DA PRIMEIRA FOLHA DO INDICE
C   PROX  = PONTEIRO PARA A PROXIMA PAGINA
C   RAIZ  = APONTA PARA A RAIZ DA ARVORE
C   REGIS = REGISTRO DO ARQUIVO DE INFORMACOES
C   TAAUX = APONTADOR DENTRO DO VETOR AUXILIAR DA VARIAVEL APAUX
C   TABCP = TABELA CONTENDO POSICOES INICIAIS DOS CAMPOS DO REGISTRO
C   TAMEN = TAMANHO DO MENOR SEPARADOR
```

```
PAGE   2
C   TASEP = TAMANHO DO SEPARADOR
C   TBFRM = TABELA DE FORMATOS
C   TIPO  = TIPO DA CARGA (0=MACICA, SENAO SO' CABECALHO)
C   TMCHV = TAMANHO DA CHAVE ( 0=FIXO, SENAO VARIAVEL )
C   TMCMP = VETOR CONTENDO OS TAMANHOS DOS CAMPOS DO REGISTRO
C   TMPAG = TAMANHO DA PAGINA DO ARQUIVO DE INDICES
C   TMREG = TAMANHO DO REGISTRO DO ARQUIVO DE INFORMACOES
C   TOPO  = APONTADOR DA PLAPT
C   ULTMA = APONTADOR DA ULTIMA FOLHA DO INDICE
C
C
      INTEGER APAUX,APBUF,APREG,APSEP,APMEN,AUX,AUX1,BRANC,COMEC,CONT,
     *        ERRO,FATOR,FILHA,FIM,IRMA,LADO,MAE,MAXPG,NOCHA,PROX,TAAUX,
     *        TAMEN,TASEP,TIPO,ZERO
      INTEGER CHAVE(8),INDCS(10)
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *         NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *          TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *         PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *         NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *          TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *        REGIS,TABCP,TMCMP
      DATA BRANC/' '/,MAE/2/,IRMA/3/,FILHA/1/,ZERO/0/
C
      APREG = 0
      M(1) = 1
      M(2) = 0
      M(3) = 1
      ERRO=0
CNIVEL DAS PAGINAS FOLHAS
      READ(NOARC'1) NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,TMREG,
     *  NOCMP,(TABCP(I),I=1,20),(TMCMP(I),I=1,20),CHV,ALFA,DOCUM
      DISPI = 80/TMPAG+2
      IF(TIPO) 1,4,1
    1    RAIZ = 0
         ULTMA = 0
         PRIM = 0
         IF(NOREG) 3,2,3
    2 NOIND = 1
      DISPR = 160/TMREG+2
      INDCS(1) = NOARI
         GO TO 605
    3 READ(NOARR'1) NOREG,DISPR,NOIND,INDCS,NOARR,MAXRG,TMREG,NOCMP,
     *TABCP,TMCMP,CHV,ALFA,DOCUM
      NOIND = NOIND + 1
      INDCS(NOIND) = NOARI
      GO TO 615
C ROTINA PARA CRIACAO DAS PAGINAS FOLHAS
    4 NOIND = 0
      DISPR = 160/TMREG+2
      I = DISPR
      ITAM = DISPR+NOREG
      PAGIN = 80/TMPAG+1
      PRIM= PAGIN+1
    5 FTOCP(FILHA) = 1
      NOCHV(FILHA) = 0
```

```
PAGE   3
C . . ENQUANTO I NAO ATINGIR O REGISTRO EXTREMO (ITAM) FACA
   10 IF(I-ITAM) 20,70,20
C . . .  ENQUANTO PAGINA NAO OCUPADA EM PELO MENOS DOIS-TERCOS (2/3) FACA
   20    IF(FTOCP(FILHA)-2*(TMPAG-5)/3)30,50,50
C . . . . . LER CHAVE NO ARQUIVO DE INFORMACOES
   30     READ(NOARQ'I) (REGIS(J),J=1,TMREG)
C . . . . . SE CHAVE NUMERICA

            IF(ALFA-1)33,38,33
C . . . . .    ENTAO TRANSFORMA-LA EM ALFA
   33           AUX=TABCP(CHV)
                N=REGIS(AUX)
                CALL NALFA(N,CHAVE)
C . . . . .  COLOCAR A CHAVE NA PAGINA
                DO 35 J=1,5
                   IF(CHAVE(J) - BRANC) 34,36,34
   34              AUX=FTOCP(FILHA)+J
   35              BUFF(FILHA,AUX) = CHAVE(J)
                GO TO 45
   36           J = J-1
                GO TO 45
   38         TAAUX = TMCMP(CHV)
              DO 40 J=1,TAAUX
                 AUX = FTOCP(FILHA)+J
                 AUX1 = TABCP(CHV)+J-1
                 IF(REGIS(AUX1)-BRANC) 40,39,40
C . . . . . TESTA SE A CHAVE TEM TAMANHO FIXO OU VARIAVEL
   39               IF(TMCHV) 45,40,45
   40            BUFF(FILHA,AUX) = REGIS(AUX1)
              GO TO 45
C . . . . . ATUALIZE FATOR DE OCUPACAO E NUMERO DE CHAVES DA PAGINA
   45         NOCHV(FILHA)=NOCHV(FILHA)+1
              FTOCP(FILHA)=FTOCP(FILHA)+J
              APBUF = FTOCP(FILHA)
              BUFF(FILHA,APBUF)=I
C . . . . . INCREMENTAR I
              I = I+1
              GO TO 10
   50      PAGIN = PAGIN+1

           IF(PAGIN-MAXPG) 55,55,1000
   55      NOPAG(FILHA) = PAGIN
C . . . . . GRAVE A PAGINA QUE APONTARA' PARA A PROXIMA PAGINA
              PROX = PAGIN+1
              WRITE(NOARI'PAGIN) NOPAG(FILHA),NOCHV(FILHA),FTOCP(FILHA),
     *         M(FILHA),PROX,(BUFF(FILHA,J), J=2,APBUF)
           GO TO 5
C . . . . . .   GRAVE A PAGINA OMITINDO O PONTEIRO DE PROXIMA PAGINA
   70         PAGIN=PAGIN+1
              NOPAG(FILHA) = PAGIN
              IF(PAGIN-MAXPG) 80,80,1000
   80         WRITE(NOARI'PAGIN) NOPAG(FILHA),NOCHV(FILHA),FTOCP(FILHA),
     *         M(FILHA),ZERO,(BUFF(FILHA,J),J=2,APBUF)
              ULTMA=PAGIN
C
CNIVEL DAS PAGINAS INTERNAS
           FIM=PAGIN
           COMEC=PRIM
           I=COMEC
  103      FTOCP(MAE)=1
```

```
PAGE   4
            NOCHV(MAE)=0
C  .  .  ENQUANTO HOUVER MAIS DE UMA PAGINA A SER TRABALHADA FACA
C  .  .  .  LER PAGINA NO ARQUIVO INDICE
   105      READ(NOARI'I) NOPAG(FILHA),NOCHV(FILHA),AUX,M(FILHA),
      *      (BUFF(FILHA,J),J=1,AUX)
            FTOCP(FILHA) = AUX
            APBUF = FTOCP(MAE)
C  .  .  .  COLOCAR O NUMERO DA PAGINA NA PAGINA MAE
            BUFF(MAE,APBUF) = I
C  .  .  .  SE EXISTE IRMA A DIREITA DA PAGINA FILHA
            IF(I-FIM) 120,460,460
C  .  .  .  .  ENTAO SE PAGINA MAE NAO OCUPADA EM DOIS TERCOS (2/3)
   120        IF(FTOCP(MAE)-2*(TMPAG-5)/3) 130,460,460
C  .  .  .  .  .  .  .  .  ENTAO
C  .  .  .  .  .  .  .  .  .  .  .  .  LER A PAGINA IRMA NO ARQUIVO INDICE
   130           I = I + 1
                 READ(NOARI'I) NOPAG(IRMA),NOCHV(IRMA),AUX1,
      *           M(IRMA),(BUFF(IRMA,J),J=1,AUX1)
C  .  .  .  .  .  .  .  .  .  .  .  SE PAGINA FILHA E' FOLHA
C  .  .  .  .  .  .  .  .  .  .  .  .  ENTAO CONSTRUIR UM SEPARADOR ENTRE AS PAGINAS
C                                          (FILHA E IRMA)
C  .  .  .  .  .  .  .  .  .  .  .  .  SENAO ACHAR O SEPARADOR MENOR ENTRE AS PAGINAS
C  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  RETIRAR O SEPARADOR DA RESPECTIVA PAGINA
              FTOCP(IRMA) = AUX1
              LADO = IRMA
              TAMEN = 0
              AUX1 = 1
              APMEN = AUX1
              IF(M(IRMA)) 121,121,131
   121           AUX1 = AUX1+1
                 IF(BUFF(IRMA,AUX1)) 125,125,123
   123           IF(BUFF(IRMA,AUX1)-BRANC) 138,125,125
   125           TAMEN = TAMEN + 1
                 GO TO 121
   131        AUX = AUX-1
              IF(BUFF(FILHA,AUX)) 131,131,133
   133        IF(BUFF(FILHA,AUX)-BRANC)135,131,131
   135        AUX = AUX+1
              AUX1 = AUX1+1
              TAMEN = TAMEN+1
              IF(BUFF(FILHA,AUX)-BUFF(IRMA,AUX1))138,135,138
   138        IF(TAMEN - 1) 310,310,139
   139        APAUX = FTOCP(FILHA)
   140        APAUX = APAUX-1
              IF(BUFF(FILHA,APAUX)) 140,140,150
   150        IF(BUFF(FILHA,APAUX)-BRANC) 153,140,140
   153        APSEP = APAUX
   155        APAUX = APAUX-1
              IF(APAUX-(7*(TMPAG-5)/12+(TMCMP(CHV)+1)))210,160,160
   160           IF(BUFF(FILHA,APAUX)) 155,155,165
   165           IF(BUFF(FILHA,APAUX)-BRANC) 170,155,155
   170           IF(M(FILHA))171,171,174
   171              TASEP = APAUX
                    GO TO 180
   174              TASEP = APSEP
                    TAAUX = APAUX
   175              TASEP = TASEP+1
                    TAAUX = TAAUX+1
                    IF(BUFF(FILHA,TASEP)-BUFF(FILHA,TAAUX)) 180,175,180
   180                 IF(TAMEN-ABS(TASEP-APSEP)) 200,200,190
```

```
PAGE    5
  190                     TAMEN = ABS(TASEP - APSEP)
                          APMEN = APAUX
                          LADO = FILHA
  200                     GO TO 153
  210         APAUX = 1
  253        APSEP = APAUX
  255         APAUX = APAUX+1
              IF(FTOCP(IRMA)-APAUX-(7*(TMPAG-5)/12)-(TMCMP(CHV)+1))
     *         310,260,260
  260           IF(BUFF(IRMA,APAUX)) 255,255,265
  265           IF(BUFF(IRMA,APAUX)-BRANC)270,255,255
  270           IF(M(IRMA)) 271,271,274
  271              TASEP = APAUX
                   GO TO 280
  274              TASEP = APSEP
                   TAAUX = APAUX
  275              TASEP = TASEP+1
                   TAAUX = TAAUX+1
                   IF(BUFF(IRMA,TASEP)-BUFF(IRMA,TAAUX)) 280,275,280
  280                 IF(TAMEN-ABS(TASEP-APSEP)) 300,300,290
  290                    TAMEN = ABS(TASEP-APSEP)
                         APMEN = APSEP
                         LADO = IRMA
  300                    GO TO 253
C
C . . . . . COLOCAR O SEPARADOR NA PAGINA MAE
  310         J = 0
  315         J = J+1
              AUX = APMEN+J
              AUX1 = FTOCP(MAE)+J
              BUFF(MAE,AUX1)=BUFF(LADO,AUX)
              IF(M(LADO))316,316,317
  316         IF(BUFF(LADO,AUX+1)) 315,315,318
  318         IF(BUFF(LADO,AUX+1)-BRANC)320,315,315
  317         IF(J-TAMEN)315,320,320
  320       NOCHV(MAE) = NOCHV(MAE)+1
            FTOCP(MAE) = FTOCP(MAE)+J+1
            IF(M(LADO))323,323,328
  323         AUX = APMEN
              AUX1 = APMEN + TAMEN + 1
  325         BUFF(LADO,AUX) = BUFF(LADO,AUX1)
              AUX = AUX + 1
              AUX1 = AUX1 + 1
              IF(AUX1 - FTOCP(LADO)) 325,325,326
  326            FTOCP(LADO) = FTOCP(LADO) - TAMEN - 1
                 NOCHV(LADO) = NOCHV(LADO) - 1
C
  328       IF(LADO-FILHA) 380,330,380
  330         AUX = FTOCP(LADO)
              CONT = 0
              APBUF = 2
  340         APBUF = APBUF+1
              IF(BUFF(LADO,APBUF)) 340,340,345
  345         IF(BUFF(LADO,APBUF)-BRANC)350,340,340
  350            CONT = CONT+1
                 IF(APBUF-APMEN) 340,360,360
  360         NOCHA = NOCHV(IRMA)+NOCHV(FILHA)
              NOCHV(LADO) = CONT
              NOCHV(IRMA) = NOCHA-CONT
              WRITE(NOARI'NOPAG(FILHA)) NOPAG(FILHA),NOCHV(FILHA),APMEN,
```

```
PAGE    6

     *        M(FILHA),(BUFF(FILHA,J),J=1,APMEN)
              FTOCP(FILHA) = APMEN
              AUX1 = FTOCP(IRMA)+(AUX-FTOCP(FILHA))
              APBUF = FTOCP(IRMA)
              WRITE(NOARI'NOPAG(IRMA))NOPAG(IRMA),NOCHV(IRMA),AUX1,M(IRMA)
     *         ,(BUFF(FILHA,J),J=APMEN,AUX),(BUFF(IRMA,J),J=2,APBUF)
              GO TO 105
  380         AUX = FTOCP(IRMA)
              CONT = 0
              APBUF = APMEN+1
  390         APBUF = APBUF+1
              IF(BUFF(IRMA,APBUF)) 390,390,395
  395         IF(BUFF(IRMA,APBUF)-BRANC) 400,390,390
  400            CONT = CONT+1
                 IF(APBUF-AUX) 390,410,410
  410         NOCHA = NOCHV(IRMA)+NOCHV(FILHA)
              NOCHV(IRMA) = CONT
              NOCHV(FILHA) = NOCHA-CONT
              FTOCP(IRMA) = FTOCP(IRMA)-APMEN+1
              AUX1 = FTOCP(FILHA)+(AUX-FTOCP(IRMA))
              WRITE(NOARI'NOPAG(IRMA)) NOPAG(IRMA),NOCHV(IRMA),FTOCP(IRMA)
     *         ,M(IRMA),(BUFF(IRMA,J),J=APMEN,AUX)
              APBUF = FTOCP(FILHA)
              WRITE(NOARI'NOPAG(FILHA)) NOPAG(FILHA),NOCHV(FILHA),AUX1,
     *         M(FILHA),(BUFF(FILHA,J),J=1,APBUF),(BUFF(IRMA,J),J=2,APMEN)
              GO TO 105
C . . . . . SENAO GRAVAR PAGINA MAE
  460         PAGIN = PAGIN+1
              IF(PAGIN-MAXPG) 470,470,1000
  470         APBUF = FTOCP(MAE)
              NOPAG(MAE) = PAGIN
              WRITE(NOARI'PAGIN) NOPAG(MAE),NOCHV(MAE),FTOCP(MAE),
     *         M(MAE),(BUFF(MAE,J),J=1,APBUF)
              IF(I - FIM) 103,500,500
C
  500      IF(PAGIN-1-FIM) 600,600,510
  510           COMEC = FIM +1
                FIM = PAGIN
                I= COMEC
                GO TO 103
C      RAIZ = PAGINA MAE
  600      RAIZ = PAGIN
           DISPR = NOREG + DISPR
           DISPI = RAIZ + 1
           NOIND = NOIND + 1
           INDCS(NOIND) = NOARI
           NOCHI= NOREG
C . . CONSTRUCAO DAS PILHAS
  605      AUX1  = MAXRG - 1
           DO 610 I = DISPR,AUX1
              AUX = I + 1
  610         WRITE(NOARR'I) I,AUX
           AUX = 0
           WRITE(NOARR'I) I,AUX
  615      AUX1 = MAXPG - 1
           DO 620 I = DISPI,AUX1
              AUX = I + 1
  620         WRITE(NOARI'I) I,AUX
           AUX = 0
           WRITE(NOARI'I) I,AUX
```

```
PAGE    7
C  .  . GRAVACAO DO CABECALHO DO ARQUIVO
        WRITE(NOARR'1) NOREG,DISPR,NOIND,INDCS,NOARR,MAXRG,TMREG,NOCMP,
       *            TABCP,TMCMP,CHV,ALFA,DOCUM
C  .  . GRAVACAO DO CABECALHO DO INDICE
        WRITE(NOARI'1)NOCHI,NOREG,DISPR,DISPI,RAIZ,PRIM,ULTMA,NOIND,NOARI,
       *            NOARR,MAXPG,TMCHV,TMPAG,TMREG,NOCMP,TABCP,TMCMP,CHV,
       *            ALFA,DOCUM
C
C
        IF(TIPO) 2000,1000,2000
C  .  . LER O ULTIMO REGISTRO
 1000 K = NOREG + 160/TMREG + 1
        READ(NOARR'K)  (REGIS(J),J=1,TMREG)
C  .  . SE CHAVE DO REGISTRO E' NUMERICA .
        IF(ALFA-1)733,738,733
  733     AUX = TABCP(CHV)
          N   = REGIS(AUX)
          CHAVE(1) = REGIS(AUX)
          GO TO 745
  738     TAAUX = TMCMP(CHV)
C  .  .  . SENAO JOGAR A CHAVE DO REGISTRO NO VETOR CHAVE
          DO 740 J=1,TAAUX
            AUX1 = TABCP(CHV) + J - 1
            IF(REGIS(AUX1) - BRANC) 740,739,740
  739           IF(TMCHV) 745,740,745
  740           CHAVE(J) = REGIS(AUX1)
  745     CONTINUE
C  .  . CHAMAR A SUBROTINA DE BUSCA PARA ESSA CHAVE
          CALL BUSCA(CHAVE,APREG,ERRO)
          ERRO = 0
          I = TOPO
C  .  . BALANCEAMENTO  .  .
          DO 750 TOPO = 1,I
            IF(PLPDB(TOPO))750,750,748
  748       PAGIN = PLPAG(TOPO)
            READ(NOARI'PAGIN) NOPAG(FILHA),NOCHV(FILHA),FATOR,M(FILHA),
       *        (BUFF(FILHA,J),J=1,FATOR)
            FTOCP(FILHA) = FATOR.
            CALL BALAN(ERRO)
            IF(ERRO) 2000,750,2000
  750     CONTINUE
          TOPO = I
C
 2000 RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR CARGA
 COMMON   1130  VARIABLES     60  PROGRAM   2558

RELATIVE ENTRY POINT ADDRESS IS 0046 (HEX)

END OF COMPILATION

// DUP

*DELETE              CARGA
```

```
PAGE   1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      SUBROUTINE INSER(CHAVE,APREG,INDC1,ERRO)
C
C     APREG = APONTADOR PARA O REGISTRO
C     BUFF  = CONTEM AS CHAVES/SEPARADORES E PONTEIROS DE UMA PAGINA
C     CHAVE = CHAVE A SER INSERIDA ( VETOR ALFANUMERICO )
C     CHV   = INDICA O NUMERO DO CAMPO CHAVE DENTRO DO REGISTRO
C     DISPR = APONTA P/ O TOPO DA PILHA DE REGISTROS DISPONIVEIS
C     ERRO  = PARAMETRO DE ERRO
C     FILHA = INDICA A LINHA QUE CONTEM AS INFORMACOES DA PAGINA FILHA
C     FTOCP = FATOR DE OCUPACAO
C     INDC1 = CONDICAO DA INSERCAO ( = 1 - INSERIR TAMBEM NO ARQUIVO)
C     MEIO  = INDICA A POSICAO DENTRO DO BUFF ONDE DEVE SER INSERIDA A CHAVE
C     NOARC = NUMERO DO ARQUIVO CABECALHO
C     NOARI = NUMERO DO ARQUIVO INDICE
C     NOARR = NUMERO DO ARQUIVO DE INFORMACOES
C     NOCHV = NUMERO DE CHAVES
C     PAGIN = APONTADOR DE PAGINAS
C     POS1  = APONTADOR AUXILIAR DENTRO DO BUFF
C     POS2  = APONTADOR AUXILIAR DENTRO DO BUFF
C     TMCMP = VETOR CONTENDO O TAMANHO DOS REGISTROS DE INFORMACAO
C
      INTEGER APREG,AUX,ERRO,FATOR,FILHA,PRIM,POS1,POS2
      INTEGER CHAVE(8)
C
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *         NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *          TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *         PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *         NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *          TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *        REGIS,TABCP,TMCMP
C
      DATA FILHA/1/,BRANC/' '/
      ERRO=0
C
CSE NUMERO DE CHAVES NO ARQUIVO E' ZERO
      IF(NOCHI) 5,5,15
C   ENTAO PEGUE NOVA PAGINA PARA A RAIZ
    5   CALL PGPAG(ERRO)
C . . .  SE ERRO
         IF(ERRO) 8,9,8
C . . . .  ENTAO ERRO = 1
    8      ERRO = 1
C . . . . . .  PARE
      RETURN
C . . . .  SENAO INICIALIZE M,NUMERO DE CHAVES,FATOR DE OCUPACAO,MEIO E
C . . . . . APONTADOR DE PROXIMA PAGINA (BUFF(FILHA,1))
```

```
PAGE    2

     9       M(FILHA) = 1
             RAIZ = PAGIN
             NOPAG(FILHA) = PAGIN
             NOCHV(FILHA) = 0
             FTOCP(FILHA) = 1
             MEIO = 1
             BUFF(FILHA,1) = 0
             PRIM = PAGIN
             ULTMA = PAGIN
             AUX = 0
             IF(TMCHV) 14,13,14
    13          AUX = TMCMP(CHV) + 1
                POS2 = AUX + 1
                GO TO 60
    14       AUX = AUX + 1
             POS2 = AUX + 1
                IF(CHAVE(AUX)) 14,14,60
C   SENAO
C         ENCONTRE A PAGINA E A POSICAO DENTRO DESTA ONDE SERA INSERIDA A CHAVE
    15 CALL BUSCA(CHAVE,APREG,ERRO)
C         SE CHAVE JA EXISTE NO ARQUIVO DE INDICES
       IF(ERRO-1)100,20,100
    20    POS1=FTOCP(FILHA)
          MEIO = PLAPT(TOPO)
C   ENTAO
C        ERRO
C   SENAO
C         INSIRA NA POSICAO CORRESPONDENTE DA PAGINA A NOVA CHAVE
C         ATUALIZE O FATOR DE OCUPACAO E O NUMERO DE CHAVES DA PAGINA ALTERADA
C         SE O REGISTRO E PARA SER GRAVADO NO ARQUIVO
C            ENTAO  GRAVE O NOVO REGISTRO
C                   ATUALIZE A PILHA DE REGISTROS DISPONIVEIS
C
       IF(TMCHV)25,35,25
    25    AUX = TABCP(CHV)
    28    AUX = AUX + 1
          IF(AUX - TMCMP(CHV))29,29,38
    29    IF(REGIS(AUX)-BRANC) 28,30,28
    30          AUX = AUX - TABCP(CHV) + 1
                GO TO 38
    35          AUX = TMCMP(CHV)+1
    38    POS2=FTOCP(FILHA)+AUX
    40    IF(POS1-MEIO)60,60,50
    50       BUFF(FILHA,POS2)=BUFF(FILHA,POS1)
             POS1=POS1-1
             POS2=POS2-1
             GO TO 40
    60    FTOCP(FILHA)=FTOCP(FILHA)+AUX
          NOCHV(FILHA)=NOCHV(FILHA)+1
          PAGIN = APREG
          IF(INDC1-1) 83,75,83
    75    ERRO = 0
          CALL PGREG(ERRO)
          IF(ERRO)70,80,70
    70       ERRO=4
       RETURN
    80 WRITE(NOARR'PAGIN) (REGIS(I),I=1,TMREG)
          NOREG = NOREG+1
          APREG = PAGIN
          WRITE(NOARR'1) NOREG,DISPR
```

```
PAGE    3

    83      AUX = TABCP(CHV) - 1
    85      MEIO = MEIO + 1
            AUX = AUX + 1
            BUFF(FILHA,MEIO) = REGIS(AUX)
            IF(MEIO-POS2)85,90,90
    90      BUFF(FILHA,POS2) = PAGIN
            FATOR = FTOCP(FILHA)
            NOCHI = NOCHI+1
            WRITE(NOARI'NOPAG(FILHA))NOPAG(FILHA),NOCHV(FILHA),FTOCP(FILHA)
       *     ,M(FILHA),(BUFF(FILHA,I),I=1,FATOR)
            WRITE(NOARI'1) NOCHI,NOREG,DISPR,DISPI,RAIZ,PRIM,ULTMA
            IF(NOCHI - 1)  98,98,93
    93      ERRO = 0
            I = TOPO

C   .   BALANCEAMENTO   .   .
            DO 95 TOPO = 1,I
              IF(PLPOB(TOPO))95,95,94
    94        PAGIN = PLPAG(TOPO)
              READ(NOARI'PAGIN)NOPAG(FILHA),NOCHV(FILHA),FATOR,M(FILHA),
       *         (BUFF(FILHA,J),J=1,FATOR)
              FTOCP(FILHA) = FATOR
              CALL BALAN(ERRO)
              IF(ERRO) 98,95,98
    95      CONTINUE
            TOPO = I
    98  RETURN
   100  ERRO = 1
        RETURN
        END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR INSER
 COMMON    1130   VARIABLES      12   PROGRAM     630

RELATIVE ENTRY POINT ADDRESS IS 000F (HEX)

END OF COMPILATION

// DUP

*DELETE              INSER
CART ID 0005   DB ADDR  27D9   DB CNT   002A

*STORE      WS  UA  INSER
CART ID 0005   DB ADDR  2A8B   DB CNT   0029
```

```
PAGE   1
// JOB    0005
LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000
V2 M12   ACTUAL 16K  CONFIG 16K
// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
      SUBROUTINE REMOV(CHAVE,ERRO)
C
C     AUX    = VARIAVEL AUXILIAR
C     CHAVE  = CHAVE A SER REMOVIDA
C     DISPR  = APONTA P/ O TOPO DA PILHA DE REGISTROS DISPONIVEIS
C     ERRO   = PARAMETRO DE ERRO
C     FILHA  = INDICA A LINHA QUE CONTEM AS INFORMACOES DA PAGINA FILHA
C     FTOCP  = FATOR DE OCUPACAO
C     MEIO   = INDICA A POSICAO DENTRO DO BUFF ONDE DEVE SER REMOVIDA A CHAVE
C     NOARC  = NUMERO DO ARQUIVO CABECALHO
C     NOARI  = NUMERO DO ARQUIVO INDICE
C     NOARR  = NUMERO DO ARQUIVO DE INFORMACAO
C     NOCHV  = NUMERO DE CHAVES
C     NOIND  = NUMERO DE ARQUIVOS INDICES P/ O ARQUIVO DE INFORMACOES
C     TMCMP  = VETOR CONTENDO O TAMANHO DOS REGISTROS DE INFORMACAO
C
      INTEGER APREG,AUX,ERRO,FATOR,FILHA
      INTEGER CHAVE(8),INDCS(10)
C
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *         NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *         TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *         PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *         NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *         TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *        REGIS,TABCP,TMCMP
C
      DATA FILHA/1/
      ERRO=0
CENCONTRE A CHAVE E REGISTRO  A SEREM REMOVIDAS DOS RESPECTIVOS ARQUIVOS
      CALL BUSCA(CHAVE,APREG,ERRO)
      IF(ERRO) 100,3,100
CSE ENCONTRADAS (A CHAVE E REGISTRO  CORRESPONDENTES)
C  ENTAO
    3 IF(TMCHV)5,4,5
    4    FATOR = TMCMP(CHV) + 1
         GO TO 8
    5    FATOR = 0
    6    FATOR = FATOR + 1
         MEIO = MEIO + 1
         IF(BUFF(FILHA,MEIO))6,6,7
    7    MEIO = MEIO - FATOR
C      SE O ARQUIVO POSSUI UM UNICO INDICE
    8 IF(NOIND-1) 15,10,15
C        ENTAO REMOVA O REGISTRO DO ARQUIVO DE INFORMACAO
C         DEIXANDO DISPONIVEL A SUA POSICAO
   10    CALL DVREG
```

```
PAGE   2
           NOREG = NOREG - 1
C         REMOVA A CHAVE DA PAGINA DESLOCANDO AS DEMAIS A DIREITA SOBRE SUA POSICA
    15     IF(MEIO+FATOR-FTOCP(FILHA))20,30,30
    20     MEIO=MEIO+1
           AUX=FATOR + MEIO
           BUFF(FILHA,MEIO)=BUFF(FILHA,AUX)
           GO TO 15
C         ATUALIZE O FATOR DE OCUPACAO E O NUMERO DE CHAVES DA PAGINA ALTERADA
    30         FTOCP(FILHA)=MEIO
               NOCHV(FILHA)=NOCHV(FILHA)-1
               NOCHI = NOCHI - 1
               IF(NOCHI)35,35,49
    35           READ(NOARR'1) NOREG,DISPR,NOIND,INDCS
               NOIND = NOIND - 1
                  DO 40 I = 1,NOIND
                     IF(INDCS(I)-NOARI)40,43,40
    40            CONTINUE
    43            IF(I-NOIND)45,45,48
    45               INDCS(I) = INDCS(I+1)
                     GO TO 43
    48            NOIND = NOIND - 1
    49     WRITE(NOARI'I) NOCHI,NOREG,DISPR,DISPI,RAIZ,PRIM,ULTMA,NOIND
           WRITE(NOARR'1) NOREG,DISPR,NOIND,INDCS
           FATOR = FTOCP(FILHA)
           WRITE(NOARI'NOPAG(FILHA))NOPAG(FILHA),NOCHV(FILHA),FTOCP(FILHA)
       *     ,M(FILHA),(BUFF(FILHA,I),I=1,FATOR)
           ERRO = 0
           I = TOPO

C  . . BALANCEAMENTO . .
           DO 60 TOPO = 1,I
             IF(PLPDB(TOPO))60,60,50
    50       PAGIN = PLPAG(TOPO)
             READ(NOARI'PAGIN)NOPAG(FILHA),NOCHV(FILHA),FATOR,M(FILHA),
       *          (BUFF(FILHA,J),J=1,FATOR)
             FTOCP(FILHA) = FATOR
             CALL BALAN(ERRO)
             IF(ERRO) 100,60,100
    60     CONTINUE
           TOPO = I
C   SENAO
C         ERRO
   100 RETURN
       END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR REMOV
 COMMON    1130  VARIABLES     20  PROGRAM     420

RELATIVE ENTRY POINT ADDRESS IS 0016 (HEX)

END OF COMPILATION

// DUP

*DELETE              REMOV
CART ID 0005   DB ADDR  268F   DB CNT   001E
```

```
PAGE   1

// JOB    0005

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS

      SUBROUTINE BALAN(ERRO)


C   APBUF = APONTADOR PARA AS POSICOES DENTRO DO BUFFER
C   APMEN = APONTADOR PARA O MENOR SEPARADOR(A ESQUERDA E A DIREITA DA CHAVE
C           CENTRAL) DENTRO DO VETOR
C   APFIX = APONTADOR FIXO PARA O SEPARADOR DENTRO DA PAGINA MAE
C   APSEP = APONTADOR PARA O SEPARADOR DENTRO DO VETOR
C   APVAR = APONTADOR VARIAVEL P/ O SEPARADOR DENTRO DA PAGINA MAE
C   APVET = APONTADOR PARA AS POSICOES DENTRO DO VETOR
C   APAUX = APONTADOR P/ A POSICAO DO SEPARADOR ANTERIOR AO INDICADO PELO APSEP
C   AUX   = VARIAVEL AUXILIAR P/ TROCA DE VALORES ENTRE 2 VARIAVEIS
C   AUXI  = VARIAVEL AUXILIAR USADA COMO INDICE
C   BUFF  = CONTEM AS CHAVES/SEPARADORES E PONTEIROS DE UMA DETERMINADA PAGINA
C   CHV   = INDICA O NUMERO DO CAMPO CHAVE DENTRO DO REGISTRO
C   CONT  = CONTADOR DE CHAVES DE UMA PAGINA, DENTRO DO VETOR
C   DISPI = APONTA P/ O TOPO DA PILHA DE PAGINAS DISPONIVEIS
C   DISPR = APONTA P/ O TOPO DA PILHA DE REGISTROS DISPONIVEIS
C   FILHA = INDICA A LINHA QUE CONTEM AS INFORMACAOES DA PAGINA FILHA
C   FTOCP = FATOR DE OCUPACAO DA PAGINA INDICADA (FILHA,MAE,IRMA)
C   K     = INDICA SE A DIVISAO DE PAGINA DEVE OCORRER EM 1/3 (UM-TERCO) OU EM
C           2/3 (DOIS-TERCOS) CONFORME A SUA RELACAO DE LADO COM A PAGINA IRMA
C   LDDIR = NUMERO DA PAGINA (VIZINHA OU FILHA) MAIS A DIREITA
C   LDESQ = NUMERO DA PAGINA (VIZINHA OU FILHA) MAIS A ESQUERDA
C   IRMA  = INDICA A LINHA QUE CONTEM AS INFORMACOES DA PAGINA VIZINHA (IRMA)
C   L     = INDICA SE OS DOIS LADOS DA PAGINA FILHA FORAM ACESSADOS
C   LADO  = INDICA O LADO DO VIZINHO DA PAGINA FILHA QUE DEVE SER ACESSADO
C   LIMIT = VARIAVEL AUXILIAR QUE INDICA O LIMITE DE UMA PAGINA DENTRO DO VETOR
C   M     = MOSTRA SE E FILHA(1) OU NO INTERNO(0), A PAGINA INDICADA
C   MAE   = INDICA A LINHA QUE CONTEM AS INFORMACOES DA PAGINA MAE
C   MEIO  = POSICAO DENTRO DA PAGINA DO PONTEIRO ENCONTRADO PELA BUSCA
C   NOARC = NUMERO DO ARQUIVO CARECALHO
C   NOARI = NUMERO DO ARQUIVO DE INDICES
C   NOARR = NUMERO DO ARQUIVO DE REGISTROS DE INFORMACAO
C   NOCHA = NUMERO DE CHAVES DE UMA PAGINA, DENTRO DO VETOR
C   NOCHV = NUMERO DE CHAVES DA PAGINA INDICADA
C   NOPAG = NUMERO DA PAGINA INDICADA
C   PAGIN = APONTADOR PARA AS PAGINAS FILHAS, SENDO QUE AO TERMINO DA BUSCA
C           APONTA PARA O REGISTRO ASSOCIADO A CHAVE
C   PARAM = PARAMETRO DE ENTRADA DA FUNCAO RANDO
C   PLAPT = PILHA COM OS PONTEIROS PELOS QUAIS 'DESCEU' O ALGORITMO DE BUSCA
C   POS1  = POSICAO DA INFORMACAO DENTRO DO BUFFER MAE, QUE SERA ENVIADA P/ POS2
C   POS2  = POSICAO REF. AO DESLOCAMENTO DOS SEPARADORES E PONTEIROS DA PAGINA
C           MAE  DE MODO A POSSIBILITAR A INSERCAO DE NOVO SEPARADOR
C   PROX  = INDICA A PROXIMA PAGINA NOS NIVEIS DA FOLHA
C   SEPI  = USADA P/ INSERCAO DO SEPARADOR NO BUFFER MAE
C   TAAUX = APONTADOR DENTRO DO VETOR AUXILIAR DA VARIAVEL APAUX
C   TAMEN = TAMANHOS DOS MENORES SEPARADORES DENTRO DO LIMITE DE BALANCEAMENTO,
C           DO VETOR DE ARMAZENAMENTO
```

```
PAGE   2

C   TASEP = APONTADOR DENTRO DO VETOR, AUXILIAR DO APONTADOR APSEP
C   TMCMP = VETOR QUE INDICA O TAMANHO DOS CAMPOS DO REGISTRO DE INFORMACAO
C   TMPAG = TAMANHO DA PAGINA
C   TOPO  = INDICA O TOPO DA PILHA PLAPT E PLPAG
C
       INTEGER APAUX,APBUF,APFIX,APSEP,APVAR,APVET,AUX,AUXI,CONT,ERRO,
      *  FATOR,FILHA,IRMA,LADO,LDDIR,LDESQ,LIMIT,MAE,NOCHA,PARAM,POS1,
      *        POS2,PROX,SEPI,TAAUX,TAMAN,TASEP,TOPO
       INTEGER BRANC,ZERO
       INTEGER APMEN(2),TAMEN(2),VETOR(650)
C
       INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
      *         NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
      *          TMREG,TOPO,ULTMA
       INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
      *         PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
C
       COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
      *         NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
      *          TMCHV,TMPAG,TMREG,TOPO,ULTMA
       COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
      *        REGIS,TABCP,TMCMP
C
       DATA FILHA/1/,MAE/2/,IRMA/3/,BRANC/' '/,ZERO/0/
C
       ERRO=0
CSE NAO HA CONDICOES P/ INSERCAO (O ESPACO DE FOLGA NAO SATISFAZ)
       IF(FTOCP(FILHA)+2*(TMCMP(CHV)+1)-(TMPAG-5))25,25,10
C  ENTAO
C . . . SE PAGINA E RAIZ
   10    IF(NOPAG(FILHA)-RAIZ)35,20,35
C . . . . ENTAO
C . . . . . . FACA A DIVISAO DA PAGINA CONFORME ALGORITMO ABAIXO
   20       APBUF=FTOCP(FILHA)/2
            APSEP=APBUF
            GO TO 635
C . . . . SENAO
C . . . . . . CONTINUE COM O PROCESSO DE REDISTRIBUICAO DE PAGINAS
C  SENAO
C . . . SE NAO HA CONDICOES P/ REMOCAO (O ESPACO DE FOLGA NAO SATISFAZ)
C . . . . ENTAO
   25 IF(FTOCP(FILHA)-(TMCMP(CHV)+1)-(TMPAG-5)/2)30,1100,1100
C . . . . . SE PAGINA E RAIZ
   30    IF(NOPAG(FILHA)-RAIZ)35,1100,35
C . . . . . . ENTAO FIM DO BALANCEAMENTO
C . . . . . . SENAO CONTINUE O PROCESSO DE REDISTRIBUICAO DE PAGINAS
C . . . . SENAO
C . . . . . FIM DO BALANCEAMENTO
   35    PARAM=NOPAG(FILHA)
         L=0
C         TENTAR A REDISTRIBUICAO
C . . .  ESCOLHA UMA PAGINA VIZINHA ALEATORIAMENTE
C . . .   USANDO A SUBROTINA RANDO
         IF(RANDO(PARAM)-0.5)40,40,50
   40       LADO=-1
            GO TO 60
   50       LADO=1
C . REPITA (ATE TER TENTADO O BALANCEAMENTO DOS 2 LADOS DA PAGINA FILHA
C             OU TERMINO DO PROCESSO DE REDISTRIBUICAO )
C . . .  SE ESTA PAGINA VIZINHA EXISTE
```

```
PAGE  3

C . . . . . ENTAO
C . . . . . . . . DETERMINE O LADO DA PAGINA IRMA E DA PAGINA FILHA
C . . . . . . . . ENCONTRE NA PAGINA MAE O SEPARADOR ENTRE AS PAGINAS(IRMA,FILHA)
   60     APFIX=PLAPT(TOPO - 1)
          NOPAG(MAE) = PLPAG(TOPO - 1)
          READ(NOARI'NOPAG(MAE))NOPAG(MAE),NOCHV(MAE),AUXI,M(MAE),
     *                          (BUFF(MAE,I),I=1,AUXI)
          FTOCP(MAE) = AUXI
   63    APAUX=APFIX
   65     APAUX=APAUX+LADO
          IF(APAUX)600,600,70
   70     IF(APAUX-FTOCP(MAE))80,80,600
   80        IF(BUFF(MAE,APAUX))65,65,85
   85        IF(BUFF(MAE,APAUX)-BRANC)90,65,65
C . . . . . . . . LEIA A PAGINA VIZINHA DO ARQUIVO DE INDICE
   90           READ(NOARI'BUFF(MAE,APAUX))NOPAG(IRMA),NOCHV(IRMA),
     -                                       AUXI,M(IRMA),
     *                                       (BUFF(IRMA,I),I=1,AUXI)
          FTOCP(IRMA) = AUXI
          K = LADO
          APVAR=APAUX
          IF(LADO)93,93,95
   93        LDESQ=IRMA
             LDDIR=FILHA
             GO TO 98
   95        LDESQ=FILHA
             LDDIR=IRMA
   98     AUXI=(IABS(APFIX-APVAR)-1)*(1-M(FILHA))
C . . . . . . . . SE HA CONDICOES DE BALANCEAMENTO ENTRE A PAGINA FILHA E A IRMA
          IF(FTOCP(FILHA)-(TMPAG-5)+2*(TMCMP(CHV)+1)+AUXI-
     *       ((TMPAG-5)-FTOCP(IRMA)-2*(TMCMP(CHV)+1)))100,100,600
  100     IF(FTOCP(FILHA)-(TMPAG-5)/2-(TMCMP(CHV)+1)-(TMPAG-5)/2
     *       +FTOCP(IRMA)-(TMCMP(CHV)+1)+AUXI)600,600,110
  110        APVET=0
C . . . . . . . . ENTAO
C**REDISTRIBUICAO** *  COLOQUE NO VETOR DE ARMAZENAMENTO A PAGINA A ESQUERDA
             APBUF = 0
             LADO=LDESQ
  120           APVET=APVET+1
                APBUF=APBUF+1
                VETOR(APVET)=BUFF(LADO,APBUF)
             IF(APBUF-FTOCP(LADO))120,130,130
  130        IF(LADO-LDDIR)135,183,135
C                    *  SE AS PAGINAS (FILHA E IRMA) SAO  FOLHAS
  135           IF(M(FILHA))140,140,180
C                       *    ENTAO
C                       *          GUARDAR O PONTEIRO DE PROXIMA PAGINA DA PAGINA A
C                       *           DIREITA
C                       *    SENAO
  140             IF(APVAR-APFIX)150,150,160
  150                APSEP=APVAR
                     GO TO 170
  160                APSEP=APFIX
C                       *          PONHA NO VETOR, APOS A ULTIMA POSICAO OCUPADA O
C                       *           SEPARADOR DA PAGINA MAE
  170              APSEP=APSEP+1
                   APVET=APVET+1
                   VETOR(APVET)=BUFF(MAE,APSEP)
                   IF(BUFF(MAE,APSEP+1))170,170,175
  175              IF(BUFF(MAE,APSEP + 1)-BRANC)180,170,170
```

```
PAGE    4

C                    *  COLOCAR NO VETOR,APOS A ULTIMA POSICAO OCUPADA, A PAGINA
C                    *   A DIREITA
   180            LADO=LDDIR
                  APBUF=M(FILHA)
                  GO TO 120
C                    *  ACHE A CHAVE QUE OCUPA A POSICAO CENTRAL DO VETOR
   183           APSEP=APVET/2
   185           IF(VETOR(APSEP))190,190,188
   188           IF(VETOR(APSEP)-BRANC)195,190,190
   190              APSEP=APSEP-1
                    GO TO 185
C                    *  PROCURE A ESQUERDA E A DIREITA DA CHAVE CENTRAL O MENOR
C                    *   SEPARADOR ENTRE AS CHAVES,RESTRINGINDO-SE AO LIMITE DE
C                    *   BALANCEAMENTO ENTRE AS PAGINAS
C                    *   INICIALIZE A TAMEN(1) E TAMEN(2)
   195           TAMEN(1)=TMCMP(CHV)
                 TAMEN(2)=TMCMP(CHV)
C                    *   INICIE A PROCURA, PELO LADO ESQUERDO
                 LADO=-1
                 I=1
   200           APAUX=APSEP
C                    *   AVANCE O PONTEIRO DA POSICAO PROCURADA COMO SEPARADORA
C                    *    ATE O LIMITE MAXIMO OU MINIMO DE OCUPACAO DE CADA LADO
C                    *    DO CENTRO DO VETOR
   210           APSEP=APSEP+LADO
C                *  SE O TAMANHO DA PAGINA DO LADO ESQUERDO NAO EXCEDE O MAXIMO
                 IF(APSEP-((TMPAG-5)-2*(TMCMP(CHV)+1)))220,220,215
   215           IF(LADO)220,320,320
C                    *  ENCONTRE A POSICAO DO APONTADOR
C                *     ENTAO
C                *       SE O TAMANHO DESSA PAGINA NAO E MENOR QUE O MINIMO
   220           IF(APSEP-((TMPAG-5)/2+(TMCMP(CHV)+1)))223,225,225
   223           IF(LADO)320,320,225
C                *         ENTAO
C                *           SE O TAMANHO DA PAGINA DO LADO DIREITO, DENTRO DO VETOR,
C                *            NAO EXCEDE O TAMANHO MAXIMO PARA PAGINA
   225           IF(APVET+M(FILHA)-APSEP-TMPAG+5+2*TMCMP(CHV)+2) 228,228,226
   226           IF(LADO) 320,320,228
C                *             ENTAO
C                *               SE ESSA PAGINA NAO E MENOR QUE O TAMANHO MINIMO
   228           IF(APVET+M(FILHA)-APSEP-(TMPAG-5)/2-TMCMP(CHV)-1)229,230,230
   229           IF(LADO) 230,320,320
C                *                 ENTAO
C                *                   CONTINUE O AVANCO DO PONTEIRO QUE PROCURA O
C                *                    SEPARADOR DAS PAGINAS  DENTRO DO VETOR
C                *                 SENAO
C                *                   SE O LADO DO AVANCO DO PONTEIRO E' O ESQUERDO
C                *                     ENTAO CONTINUE O AVANCO
C                *                     SENAO VERIFIQUE O LADO
C                *             SENAO
C                *               SE O LADO DO AVANCO DO PONTEIRO E' O DIREITO
C                *                 ENTAO CONTINUE O AVANCO
C                *                 SENAO VERIFIQUE O LADO
C                *         SENAO VERIFIQUE O LADO
C                *     SENAO VERIFIQUE O LADO QUE SE FAZ O AVANCO DO PONTEIRO,
C                *      MUDANDO O LADO CASO SEJA NECESSARIO   ( ESQUERDO / DIREITO )
   230              IF(VETOR(APSEP))210,210,235
   235              IF(VETOR(APSEP)-BRANC)240,210,210
   240                 IF(M(FILHA))245,245,248
   245                    TASEP = APAUX
```

```
PAGE    5

                    GO TO 260
   248              TASEP=APSEP
                    TAAUX=APAUX
   250              TASEP=TASEP+1
                    TAAUX=TAAUX+1
                    IF(VETOR(TASEP)-VETOR(TAAUX))260,250,260
   260              TAMAN=IABS(TASEP - APSEP)
C                     *   VERIFIQUE SE O SEPARADOR E' O MENOR ATE AGORA ENCONTRADO
                    IF(TAMAN-TAMEN(I))270,280,280
   270                 TAMEN(I)=TAMAN
C                     *   SE FOR, ANOTE SEU TAMANHO E POSICAO INICIAL
                     IF(LADO) 273,273,275
   273                  APMEN(I) = APAUX
                        GO TO 280
   275                  APMEN(I) = APSEP
   280               IF(TAMAN-1)320,320,200
   320         IF(LADO)330,330,340
C                     *   PROCURAR, AGORA, DO LADO DIREITO
   330            LADO=1
                  I=2
                  APSEP = APVET / 2
   333            IF(VETOR(APSEP))335,335,334
   334            IF(VETOR(APSEP)-BRANC)200,335,335
   335               APSEP = APSEP - 1
                     GO TO 333
   340         IF(TAMEN(1)-TAMEN(2))380,390,400
C                     *   SE O SEPARADOR DO LADO DIREITO E' O MENOR
C                     *     ENTAO
C                     *          COLOQUE-O COMO SEPARADOR DEFINITIVO DO VETOR
C                     *     SENAO
C                     *          COLOQUE O SEPARADOR DO LADO ESQUERDO COMO
C                     *           SEPARADOR DEFINITIVO DO VETOR
   380            APAUX=APMEN(1)
                  TAMAN=TAMEN(1)
                  GO TO 410
   390            IF(IABS(APSEP-APMEN(1))-IABS(APSEP-APMEN(2)))380,380,400
   400            APAUX=APMEN(2)
                  TAMAN=TAMEN(2)
C                     *  ATUALIZE O NUMERO DE CHAVES E O FATOR DE OCUPACAO
C                     *   DE CADA UMA DAS PAGINAS
   410         CONT=0
               APBUF=2
   420         APBUF=APBUF+1
               IF(VETOR(APBUF))420,420,425
   425         IF(VETOR(APBUF)-BRANC)430,420,420
   430            CONT=CONT+1
                  IF(APBUF-APAUX)420,433,433
   433               NOCHA=NOCHV(FILHA)+NOCHV(IRMA)
                     NOCHV(LDDIR)=(NOCHA-CONT)-(1-M(FILHA))
                     NOCHV(LDESQ)=CONT
               WRITE(NOARI'NOPAG(LDESQ))NOPAG(LDESQ),NOCHV(LDESQ),APAUX,
              -      M(LDESQ),(VETOR(I),I=1,APAUX)
               FTOCP(LDDIR)=(APVET-APAUX)-(1-M(FILHA))*TAMAN+1
               LIMIT=(APAUX+1)+(1-M(FILHA))*TAMAN
C                     *  GRAVE AS PAGINAS BALANCEADAS
               WRITE(NOARI'NOPAG(LDDIR))NOPAG(LDDIR),NOCHV(LDDIR),
              -      FTOCP(LDDIR),M(LDDIR),BUFF(LDDIR,1),
              -      (VETOR(I),I=LIMIT,APVET)
               IF(APVAR-APFIX) 435,435,440
C                     *  ATUALIZE NA PAGINA MAE O SEPARADOR DAS PAGINAS
```

```
PAGE    6

C                       *   BALANCEADAS E O FATOR DE OCUPACAO
   435              APSEP=APVAR
                    GO TO 450
   440              APSEP=APFIX
   450           IF(TAMAN-(IABS(APFIX-APVAR)-1))460,490,510
   460              SEPI=APSEP+1
                    AUXI=APAUX+1
   465              BUFF(MAE,SEPI)=VETOR(AUXI)
                    SEPI=SEPI+1
                    AUXI=AUXI+1
                    IF((SEPI-APSEP)-TAMAN)465,465,470
   470              LIMIT=SEPI+(IABS(APFIX-APVAR)-1)-TAMAN
   480              BUFF(MAE,SEPI)=BUFF(MAE,LIMIT)
                    SEPI=SEPI+1
                    LIMIT = LIMIT + 1
                    IF(LIMIT-FTOCP(MAE))480,480,550
   490              SEPI=APSEP+1
                    AUXI=APAUX+1
   500              BUFF(MAE,SEPI)=VETOR(AUXI)
                    SEPI=SEPI+1
                    AUXI=AUXI+1
                    IF((SEPI-APSEP)-TAMAN)500,500,550
   510              SEPI=FTOCP(MAE)
                    AUXI=FTOCP(MAE)+(TAMAN-IABS(APFIX-APVAR) + 1)
   520              BUFF(MAE,AUXI)=BUFF(MAE,SEPI)
                    SEPI=SEPI-1
                    AUXI=AUXI-1
                    IF(AUXI-FTOCP(MAE))530,520,520
   530              SEPI=APSEP+1
                    AUXI=APAUX+1
   540              BUFF(MAE,SEPI)=VETOR(AUXI)
                    SEPI=SEPI+1
                    AUXI=AUXI+1
                    IF((SEPI-APSEP)-TAMAN)540,540,550
   550              FTOCP(MAE)=FTOCP(MAE)+TAMAN-IABS(APFIX-APVAR)+1
                    FATOR = FTOCP (MAE)
C                        * GRAVE A PAGINA MAE NO ARQUIVO DE INDICES
                    WRITE(NOARI'NOPAG(MAE))NOPAG(MAE),NOCHV(MAE),FATOR,M(MAE)
     *                                    ,(BUFF(MAE,I),I = 1,FATOR)
C                       * TERMINO DO PROCESSO DE REDISTRIBUICAO
                    GO TO 1100
   600           IF(L)610,610,620
C . . . . . SENAO (NAO HA CONDICOES DE BALANCEAMENTO COM A IRMA ESCOLHIDA)
C . . . . . TROQUE DE LADO
C . . . .  SENAO ( NAO EXISTE PAGINA VIZINHA DESTE LADO )
C . . . . . TROQUE DE LADO
C . ATE TER TENTADO O BALANCEAMENTO DOS DOIS LADOS DA PAGINA FILHA OU
C . .  TERMINO DO PROCESSO DE REDISTRIBUICAO
   610              L=1
                    LADO=LADO*(-1)
                    GO TO 63
   620           IF(FTOCP(FILHA)+2*(TMCMP(CHV)+1)-(TMPAG-5))900,900,630
   630              IF(K)  631,631,632
C  SE NAO HOUVE REDISTRIBUICAO E NAO HA CONDICOES PARA UMA FUTURA INSERCAO
C .  ENTAO
C++DIVISAO DE PAGINA++
C       + SE PAGINA E RAIZ
C       +   ENTAO
C       +      DIVIDIR A PAGINA AO MEIO P/ CRIAR UMA SEGUNDA PAGINA
C       +      PEGAR UMA NOVA PAGINA
```

```
PAGE   7

C       +       DISTRIBUIR AS METADES ENTRE AS PAGINAS
C       +       PEGAR UMA NOVA PAGINA
C       +       COLOCAR O SEPARADOR NA PAGINA NOVA, QUE PASSA A SER A NOVA RAIZ
C       +    SENAO
  631                K=1
                     GO TO 633
  632                K=2
C       + DIVIDIR A PAGINA EM 2/3,CRIANDO UMA SEGUNDA PAGINA, DE MODO A
C       +  POSSIBILITAR A REDISTRIBUICAO DESTA ULTIMA COM A VIZINHA DA PAGINA
C       +    ORIGINAL
  633           APBUF=FTOCP(FILHA)*K/3
C       +   ACHE A CHAVE QUE POSSIBILITA ESTE PROCESSO
                APSEP=APBUF
  635           IF(BUFF(FILHA,APSEP))640,640,645
  640              APSEP=APSEP-1
                   GO TO 635
  645           IF(BUFF(FILHA,APSEP)-BRANC)650,640,640
  650           TAMEN(1)=TMCMP(CHV)
                TAMEN(2)=TMCMP(CHV)
                APBUF = APSEP
C       +   PROCURE A ESQUERDA E A DIREITA DESTA CHAVE O MENOR SEPARADOR ENTRE AS
C       +    CHAVES,RESTRINGINDO-SE AO LIMITE DE BALANCEAMENTO ENTRE AS 2 PAGINAS
C       +    (UM SEXTO (1/6) DO TAMANHO DA PAGINA)
                LADO=-1
                I=1
  653           APAUX=APSEP
  655           APSEP=APSEP+LADO
                IF(APSEP-(APBUF+FTOCP(FILHA)/6))660,660,720
  660           IF(APSEP-(APBUF-FTOCP(FILHA)/6))720,670,670
  670              IF(BUFF(FILHA,APSEP))655,655,675
  675              IF(BUFF(FILHA,APSEP)-BRANC)680,655,655
  680                 IF(M(FILHA))685,685,688
  685                    SEPI = APAUX
                         GO TO 695
  688                 SEPI=APSEP
                      AUXI=APAUX
  690                 SEPI=SEPI+1
                      AUXI=AUXI+1
                      IF(BUFF(FILHA,SEPI)-BUFF(FILHA,AUXI))695,690,695
  695                    TAMAN=IABS(SEPI - APSEP)
                    IF(TAMAN-TAMEN(I))710,653,653
  710                 TAMEN(I)=TAMAN
                      IF(LADO) 713,713,715
  713                    APMEN(I) = APAUX
                         GO TO 718
  715                    APMEN(I) = APSEP
  718                 IF(TAMAN-1)720,720,653
  720               IF(LADO)730,730,740
  730                 LADO=1
                      I=2
                      APSEP=APBUF
                      GO TO 653
  740               IF(TAMEN(1)-TAMEN(2))750,760,770
  750                 APAUX=APMEN(1)
                      TAMAN=TAMEN(1)
                      GO TO 810
  760                 IF(IABS(APBUF-APMEN(1))-IABS(APBUF-APMEN(2)))
     *                   750,770,770
  770                 APAUX=APMEN(2)
                      TAMAN=TAMEN(2)
```

```
PAGE    8

C       +   ATUALIZE O NUMERO DE CHAVES E O FATOR DE OCUPACAO DE CADA UMA DELAS
  810         CONT=0
              APBUF=2
  820         APBUF=APBUF+1
              IF(BUFF(FILHA,APBUF)) 820,820,825
  825         IF(BUFF(FILHA,APBUF) - BRANC) 830,820,820
  830            CONT=CONT+1
                 IF(APBUF-APAUX)820,840,840
  840               NOCHA=NOCHV(FILHA)= CONT-(1-M(FILHA))
                    FATOR=FTOCP(FILHA)-APAUX+1-(1-M(FILHA))*TAMAN
                    IF(NOPAG(FILHA)-RAIZ)860,845,860
  845                  ERRO = 0
                       CALL PGPAG(ERRO)
                       IF(ERRO)849,849,847
  847 RETURN
  849                  SEPI=APAUX+1
                       RAIZ = PAGIN
                       AUXI=2
  853                 BUFF(MAE,AUXI)=BUFF(FILHA,SEPI)
                       SEPI=SEPI+1
                       AUXI=AUXI+1
                       IF(AUXI - 2 - TAMAN)853,855,855
  855                  NOPAG(MAE)=RAIZ
                       FTOCP(MAE)=AUXI
                       ULTMA = NOPAG(FILHA)
                       BUFF(MAE,AUXI)=NOPAG(FILHA)
                       PRIM = DISPI
                       BUFF(MAE,1)=DISPI
                       NOCHV(MAE)=1
                       M(MAE)=0
                       WRITE(NOARI'1)NOCHI,NOREG,DISPR,DISPI,RAIZ,PRIM,
     *                  ULTMA
                       WRITE(NOARI'RAIZ)RAIZ,NOCHV(MAE),FTOCP(MAE),M(MAE),
     *                  (BUFF(MAE,I),I=1,AUXI)
                       AUXI = APAUX + (1 - M(FILHA)) * TAMAN + 1
                       AUX=FTOCP(FILHA)

                       WRITE(NOARI'NOPAG(FILHA))NOPAG(FILHA),NOCHA,FATOR,
     *                  M(FILHA),ZERO,(BUFF(FILHA,I),I=AUXI,AUX)
                       NOPAG(IRMA) = DISPI
                       ERRO = 0
                       CALL PGPAG(ERRO)
                       IF(ERRO)858,858,856
  856                     ERRO = 3
      RETURN
C       +   GRAVE A PAGINA MAIOR
  858                  AUX = M(FILHA) + 1
                       WRITE(NOARI'NOPAG(IRMA))NOPAG(IRMA),CONT,APAUX,
     *                  M(FILHA),NOPAG(FILHA),(BUFF(FILHA,I),I=AUX,APAUX)
      RETURN
  860                     PROX = BUFF(FILHA,1)
                          BUFF(FILHA,1) = DISPI
                          WRITE(NOARI'NOPAG(FILHA))NOPAG(FILHA),CONT,
     *                     APAUX,M(FILHA),(BUFF(FILHA,I),I=1,APAUX)
                          CALL PGPAG(ERRO)
                          IF(ERRO)873,874,873
  873 RETURN
  874                     NOPAG(IRMA) = PAGIN
                          AUXI = APAUX + 1 + (1 - M(FILHA)) * TAMAN
                          AUX = FTOCP(FILHA)
```

```
PAGE    9
                        WRITE(NOARI'NOPAG(IRMA))NOPAG(IRMA),NOCHA,
     *                   FATOR,M(FILHA),PROX,(BUFF(FILHA,I),I=AUXI,AUX)
                        IF(NOPAG(FILHA) - ULTMA) 877,875,877
   875                     ULTMA = NOPAG(IRMA)
                      WRITE(NOARI'1)NOCHI,NOREG,DISPR,DISPI,RAIZ,PRIM,
     *                 ULTMA
   877                POS1=FTOCP(MAE)
                      POS2=FTOCP(MAE)+TAMAN+1
   878                IF(APFIX-POS1)880,885,885
   880                   BUFF(MAE,POS2)=BUFF(MAE,POS1)
                         POS1=POS1-1
                         POS2=POS2-1
                         GO TO 878
   885                AUXI=POS2-TAMAN
                      SEPI=APAUX+1
   890                BUFF(MAE,AUXI)=BUFF(FILHA,SEPI)
                      SEPI=SEPI+1
                      AUXI=AUXI+1
                      IF(AUXI-POS2)890,895,895
   895                   BUFF(MAE,AUXI) = NOPAG(IRMA)
                      FTOCP(MAE)=FTOCP(MAE)+TAMAN+1
                      NOCHV(MAE)=NOCHV(MAE)+1
                      FATOR = FTOCP(MAE)
                      WRITE(NOARI'NOPAG(MAE))NOPAG(MAE),NOCHV(MAE),FATOR,
     *                 M(MAE),(BUFF(MAE,I),I=1,FATOR)
C       +  REDISTRIBUA A PAGINA MENOR COM A PAGINA VIZINHA,CONFORME O METODO DE
C       +   REDISTRIBUICAO ACIMA
C       +  TERMINO DO PROCESSO DE DIVISAO DE PAGINA
                      IF(K-1)865,865,870
   865                   READ(NOARI'NOPAG(FILHA))
     *                     NOPAG(FILHA),NOCHV(FILHA),FATOR,M(FILHA),
     *                   (BUFF(FILHA,I),I=1,FATOR)
                         FTOCP(FILHA)=FATOR
                         L = 0
                         GO TO 40
   870                READ(NOARI'NOPAG(IRMA))
     *                     NOPAG(FILHA),NOCHV(FILHA),FATOR,M(FILHA),
     *                   (BUFF(FILHA,I),I=1,FATOR)
                      FTOCP(FILHA)=FATOR
                      L = 0
                      PLAPT(TOPO - 1) = AUXI
                      GO TO 50
   900           IF(FTOCP(FILHA)+FTOCP(IRMA)+2*(TMCMP(CHV)+1)+
     *           ((1-M(FILHA))*(IABS(APFIX-APVAR)-1))-
     *           (TMPAG-5))  910,910,1020
C  SE NAO HOUVE REDISTRIBUICAO E NAO HA CONDICOES PARA UMA FUTURA REMOCAO
C  . .  REPITA
C  . . . . SE HA CONDICOES P/ UNIAO DAS PAGINAS FILHA E IRMA
C            ENTAO
C--UNIAO DE PAGINAS--
   910              APBUF=FTOCP(LDESQ)
C                -  SE PAGINA E' NO' INTERNO
                    IF(M(FILHA))913,913,912
C                -   ENTAO COLOCAR O SEPARADOR DA PAGINA MAE NA PAGINA FILHA
C                -         A ESQUERDA .
C                -   SENAO ATUALIZAR PONTEIRO DE PROXIMA PAGINA
   912                BUFF(LDESQ,1) = BUFF(LDDIR,1)
                         GO TO 940
   913                IF(LADO)915,915,918
   915                   APSEP=APVAR
```

```
PAGE  10

                      GO TO 919
  918                 APSEP=APFIX
  919               APVET = APSEP
  920               APBUF=APBUF+1
                    APSEP=APSEP+1
                    BUFF(LDESQ,APBUF) = BUFF(MAE,APSEP)
                    IF(BUFF(MAE,APSEP + 1))920,920,930
  930               IF(BUFF(MAE,APSEP+1)-BRANC)940,920,920
C               -  COLOCAR O CONTEUDO DA PAGINA A DIREITA NA A ESQUERDA
  940               APVET = M(FILHA)
  950               APVET = APVET +1
                    APBUF = APBUF + 1
                    BUFF(LDESQ,APBUF) = BUFF(LDDIR,APVET)
                    IF(APVET-FTOCP(LDDIR))950,960,960
C               -  ATUALIZAR FATOR DE OCUPACAO E NUMERO DE CHAVES
  960               FTOCP(LDESQ) = APBUF
                    NOCHV(LDESQ) = NOCHV(LDESQ)+NOCHV(LDDIR)+(1-M(FILHA))
C               -  GRAVAR A PAGINA RESULTANTE DA UNIAO
                    WRITE(NOARI'NOPAG(LDESQ))NOPAG(LDESQ),NOCHV(LDESQ),
     *                 APBUF,M(LDESQ),(BUFF(LDESQ,I),I=1,APBUF)
C               -  DEVOLVER PAGINA A PILHA DISPONIVEL
                    PAGIN = NOPAG(LDDIR)
                    CALL DVPAG
C               -  SE A PAGINA DEVOLVIDA FOR A ULTIMA
                    IF(NOPAG(LDDIR)-ULTMA)970,965,970
C               -    ENTAO
C               -         A ULTIMA SERA A NOVA PAGINA (RESULTANTE DA UNIAO)
  965               ULTMA = NOPAG(LDESQ)

                    WRITE(NOARI'1)NOCHI,NOREG,DISPR,DISPI,RAIZ,PRIM,
     *                ULTMA
C               -  SE PAGINA MAE E' RAIZ E EXISTE UMA UNICA CHAVE NESSA PAGINA
  970                   IF(NOPAG(LDESQ)-RAIZ) 975,1100,975
  975                   IF(NOPAG(MAE)-RAIZ)990,980,990
  980                      IF(NOCHV(MAE)-1)985,985,990
C               -    ENTAO PAGINA FILHA TORNA-SE RAIZ
  985                        RAIZ=NOPAG(FILHA)
                      WRITE(NOARI'1)NOCHI,NOREG,DISPR,DISPI,RAIZ
C               -          DEVOLVER PAGINA MAE A PILHA DISPONIVEL
                           PAGIN=NOPAG(MAE)
                           CALL DVPAG
                           GO TO 1100
C               -    SENAO ATUALIZAR A PAGINA MAE (RETIRAR SEPARADOR DAS PAGINAS
C               -           UNIDAS)
  990                   IF(APVAR - APFIX) 993,993,995
  993                      I = APVAR + 1
                           GO TO 1000
  995                      I = APFIX + 1
 1000                   AUXI=I+IABS(APFIX-APVAR)
                        IF(AUXI - FTOCP(MAE))1005,1005,1010
 1005                   BUFF(MAE,I)=BUFF(MAE,AUXI)
                        I=I+1
                        GO TO 1000
C                       ATUALIZAR FATOR DE OCUPACAO E NUMERO DE CHAVES
 1010                   FTOCP(MAE)=FTOCP(MAE)- IABS(APFIX-APVAR)
                        NOCHV(MAE)=NOCHV(MAE)-1
                        FATOR = FTOCP(MAE)
C                       GRAVE A PAGINA MAE

                      WRITE(NOARI'NOPAG(MAE))NOPAG(MAE),NOCHV(MAE),FATOR,
```

```
PAGE  11

     *                          M(MAE),(BUFF(MAE,I),I=1,FATOR)
                           GO TO 1100
C                -          FIM DA UNIAO DE PAGINAS
C            SENAO
 1020              IF(L)1100,1100,1030
C                TROQUE DE LADO
 1030                 L=0
                      LADO=LADO*(-1)
                      APAUX=APFIX
 1035                 APAUX = APAUX + LADO
                      IF(APAUX)1100,1100,1040
 1040                    IF(APAUX-FTOCP(MAE))1050,1050,1100
 1050                       IF(BUFF(MAE,APAUX))1035,1060,1060
 1060                       IF(BUFF(MAE,APAUX)-BRANC)1070,1035,1035
 1070                          APVAR = APAUX
                               READ(NOARI'BUFF(MAE,APAUX))NOPAG(IRMA),
     *NOCHV(IRMA),AUXI,M(IRMA),(BUFF(IRMA,I),I=1,AUXI)
                               FTOCP(IRMA) = AUXI
                               IF(LADO) 1080,1080,1090
 1080                             LDESQ = IRMA
                                  LDDIR = FILHA
                                  GO TO 900
 1090                             LDESQ = FILHA
                                  LDDIR = IRMA
                                  GO TO 900
C  . .  ATE TER TENTADO O BALANCEAMENTO DOS 2 LADOS DA PAGINA FILHA OU
C            TERMINO DO PROCESSO DE UNIAO DE PAGINAS
C  SENAO   (HA CONDICOES PARA UMA FUTURA INSERCAO OU REMOCAO)
 1100 RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR BALAN
 COMMON   1130  VARIABLES    700  PROGRAM   3908

RELATIVE ENTRY POINT ADDRESS IS 02C4 (HEX)

END OF COMPILATION

// DUP

*DELETE              BALAN
CART ID 0005  DB ADDR  255E  DB CNT  00F2

*STORE    WS  UA  BALAN
CART ID 0005  DB ADDR  27B7  DB CNT  00F2
```

```
PAGE   1

// JOB    0005

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM

      SUBROUTINE POSIC(LUGAR,APREG,ERRO)


C   APONT = APONTADOR PARA A CHAVE
C   APREG = APONTADOR PARA O REGISTRO
C   CONT = CONTADOR DE CHAVES
C   FATOR = VARIAVEL AUXILIAR DE FTOCP
C
      INTEGER APONT,APREG,BRANC,CONT,ERRO,FATOR,FILHA
C
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *         NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *         TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *         PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *         NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *         TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *         REGIS,TABCP,TMCMP
C
      DATA BRANC /' '/
      FILHA = 1
      ERRO = 0
C . . SE A ARVORE POSSUI A POSICAO DESEJADA
      IF(LUGAR-NOCHI) 10,10,100
C . . . ENTAO LER A ULTIMA PAGINA FOLHA
   10   READ(NOARI'ULTMA) NOPAG(FILHA),NOCHV(FILHA),FATOR,M(FILHA),
     *                    (BUFF(FILHA,I),I=1,FATOR)
C . . . . . SE A POSICAO DESEJADA ENCONTRA-SE NA ULTIMA PAGINA FOLHA
        IF(NOCHI-NOCHV(FILHA)-LUGAR) 15,20,20
C . . . . . ENTAO PAGIN = ULTIMA  PAGINA
C . . . . . CONT = NUMERO DE  CHAVES ANTERIORES A ULTIMA PAGINA
   15       PAGIN = ULTMA
            CONT = NOCHI-NOCHV(ULTMA)
            GO TO 30
C . . . . . SENAO PAGIN = PRIMEIRA PAGINA
   20       PAGIN = PRIM
C . . . . . CONT = 0
            CONT = 0
C . . . . . ENQUANTO POSICAO DESEJADA NAO ALCANCADA FACA
C . . . . . LEIA A PAGINA APONTADA POR PAGIN
   25       READ(NOARI'PAGIN) NOPAG(FILHA),NOCHV(FILHA),FATOR,M(FILHA),
     *                        (BUFF(FILHA,I),I=1,FATOR)
            FTOCP(FILHA) = FATOR
            IF(CONT+NOCHV(FILHA)-LUGAR) 28,30,30
   28          CONT = CONT + NOCHV(FILHA)
C . . . . . SENAO PASSE PARA A PROXIMA PAGINA, INICIANDO PELA
C . . . . . PRIMEIRA CHAVE
```

```
PAGE    2

                    PAGIN = BUFF(FILHA,1)
                    GO TO 25
   30       APONT = 2
C . . . . . . . . . . SE APONT NAO INDICA ULTIMO APONTADOR DA PAGINA
C . . . . . . . . . . ENTAO AVANCE APONT
   60             APONT = APONT + 1
                  IF(BUFF(FILHA,APONT)) 60,60,65
   65             IF(BUFF(FILHA,APONT)-BRANC) 70,60,60
C . . . . . . . . . . INCREMENTE CONT
   70       CONT = CONT + 1
            IF(LUGAR-CONT) 80,80,60
   80          APREG = BUFF(FILHA,APONT)
       RETURN
C . . . SENAO ERRO = 1 ( NAO EXISTE A POSICAO NO INDICE)
  100       ERRO = 1
       RETURN
       END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR POSIC
 COMMON    1130  VARIABLES      8  PROGRAM     246

RELATIVE ENTRY POINT ADDRESS IS 000B (HEX)

END OF COMPILATION

// DUP

*DELETE              POSIC
CART ID 0005   DR ADDR  26F5   DR CNT    0011

*STORE      WS  UA  POSIC
CART ID 0005   DR ADDR  2896   DR CNT    0011
```

```
PAGE    1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K   CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
      SUBROUTINE PROX(APREG,ERRO)
C   APREG = APONTADOR PARA O REGISTRO
C   APONT = APONTADOR PARA A CHAVE PROXIMA
C   FATOR = VARIAVEL AUXILIAR DE FTOCP
C
      INTEGER APONT,APREG,BRANC,ERRO,FATOR,FILHA
C
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *          NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *          TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *          PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *          NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *          TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *        REGIS,TABCP,TMCMP
      DATA BRANC/' '/
C
      FILHA = 1
      ERRO = 0
C . . LER A PAGINA NA QUAL SE ENCONTRA A ULTIMA CHAVE PROCURADA
      APONT = PLAPT(TOPO)
      PAGIN = PLPAG(TOPO)
   10 READ(NOARI'PAGIN)NOPAG(FILHA),NOCHV(FILHA),FATOR,M(FILHA),
     *                  (BUFF(FILHA,I),I=1,FATOR)
      FTOCP(FILHA)=FATOR
C . . SE EXISTE NA MESMA PAGINA PELO MENOS UMA CHAVE POSTERIOR
      IF(APONT-FTOCP(FILHA))20,50,50
C . . . ENTAO A PRIMEIRA CHAVE POSTERIOR E' A CHAVE DESEJADA
   20   APONT = APONT + 1
      IF(BUFF(FILHA,APONT))20,20,25
   25 IF(BUFF(FILHA,APONT)-BRANC)30,20,20
C . . . . . ATUALIZAR PILHA DE APONTADORES
   30   APREG = BUFF(FILHA,APONT)
      RETURN
C . . . SENAO SE EXISTE PROXIMA PAGINA
   50 IF(ULTMA-NOPAG(FILHA))60,100,60
C . . . . . ENTAO A CHAVE DESEJADA E' A PRIMEIRA DESSA PAGINA
   60   PAGIN = BUFF(FILHA,1)
C . . . . . ATUALIZAR AS PILHAS DE PAGINA E APONTADORES
        APONT=0
        GO TO 10
C . . . . . SENAO ERRO = 1 (NAO EXISTE PROXIMO)
  100 ERRO = 1
      RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
```

```
PAGE   2

CORE REQUIREMENTS FOR PROX
 COMMON   1130  VARIABLES      8  PROGRAM    162

RELATIVE ENTRY POINT ADDRESS IS 000A (HEX)

END OF COMPILATION

// DUP

*DELETE              PROX
CART. ID 0005   DB ADDR  26E9   DB CNT   000C

*STORE      WS  UA  PROX
CART ID 0005   DB ADDR  289B   DB CNT   000C
```

```
PAGE   1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
      SUBROUTINE ANTER(APREG,ERRO)
C   APREG = APONTADOR PARA O REGISTRO
C   APONT = APONTADOR PARA A CHAVE ANTERIOR
C   FATOR = VARIAVEL AUXILIAR DE FTOCP
C   TPAUX = VARIAVEL AUXILIAR DE TOPO
C
       INTEGER APONT,APREG,BRANC,ERRO,FATOR,FILHA,TPAUX
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *          NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *           TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *          PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
C
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *          NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *           TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *          REGIS,TABCP,TMCMP
       DATA BRANC/' '/
C
       IF(ERRO)200,10,200
   10  FILHA = 1
       TPAUX = TOPO
C.....DESEMPILHE A PAGINA E O APONTADOR  (CAMINHO PERCORRIDO)
       APONT = PLAPT(TPAUX)
       PAGIN = PLPAG(TPAUX)
       TPAUX = TPAUX - 1
C . . . . . ENCONTRAR O APONTADOR ANTERIOR
   30  APONT = APONT - 1
       IF(APONT) 200,200,35
   35  IF(BUFF(FILHA,APONT)) 30,40,40
   40  IF(BUFF(FILHA,APONT)- BRANC) 45,30,30
   45  IF(APONT - 1)  60,60,50
   50  PLAPT(TPAUX+1) = APONT
          APREG = BUFF(FILHA,APONT)
       RETURN
C . . ENQUANTO A POSICAO DA CHAVE NA PAGINA FOR A PRIMEIRA
C.....E PAGINA NAO FOR RAIZ FACA
   60  IF(TPAUX)  200,200,63
   63     APONT = PLAPT(TPAUX)
          PAGIN = PLPAG(TPAUX)
          TPAUX = TPAUX - 1
       READ(NOARI'PAGIN)NOPAG(FILHA),NOCHV(FILHA),FATOR,M(FILHA),
     *                   (BUFF(FILHA,I),I=1,FATOR)
       FTOCP(FILHA) = FATOR
       IF(APONT - 1)  80,80,65
   65  APONT = APONT - 1
       IF(BUFF(FILHA,APONT)) 65,65,70
   70  IF(BUFF(FILHA,APONT) - BRANC) 85,65,65
   80  IF(PAGIN - RAIZ) 60,200,60
```

```
PAGE   2

C . . . . . ENQUANTO PAGINA NAO FOLHA FACA
C . . . . . . ATUALIZAR PILHA DE APONTADOR
    85 TPAUX = TPAUX + 1
       PLAPT(TPAUX) = APONT
    90 PAGIN = BUFF(FILHA,APONT)
       READ(NOARI'PAGIN)NOPAG(FILHA),NOCHV(FILHA),FATOR,M(FILHA),
      *                  (BUFF(FILHA,I),I=1,FATOR)
       FTOCP(FILHA) = FATOR
C . . . . . . . . . . ATUALIZE AS PILHAS DE PAGINAS E APONTADORES
       TPAUX = TPAUX + 1
       PLPAG(TPAUX) = PAGIN
       PLAPT(TPAUX) = FTOCP(FILHA)
C . . . . . . . . . . DESCA PELA ULTIMA CHAVE DA PAGINA
       APONT = FTOCP(FILHA)
       IF(M(FILHA)) 90,90,50
C . . SE PAGINA E' RAIZ E CHAVE E' A PRIMEIRA
C . . . ENTAO ERRO = 1 (NAO EXISTE ANTERIOR)
   200    ERRO = 1
       RETURN
       END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR ANTER
 COMMON    1130  VARIABLES      10  PROGRAM     336

RELATIVE ENTRY POINT ADDRESS IS 000B (HEX)

END OF COMPILATION

// DUP

*DELETE              ANTER
CART ID 0005   DB ADDR  26D3   DB CNT   0016

*STORE      WS  UA  ANTER
CART ID 0005   DB ADDR  2891   DB CNT   0016
```

```
PAGE   1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0005         0005         0000

V2 M12   ACTUAL 16K  CONFIG 16K

// ASM
*LIST
                    00001    * SUBROTINA QUE TRANSFORMA UM DIGITO BINARIO
                    00002    *            NO EBCDIC EQUIVALENTE ( NO BYTE 01 )
                    00003    *
0000     02A23C40   00004           ENT     BYT1
0000 0   0000       00005    BYT1   DC      *-*
0001 01  C4800000   00006           LD    I BYT1
0003 0   D00B       00007           STO     AUX1  END. PARAMETRO 1
0004 0   C00B       00008           LD      F
0005 0   1084       00009           SLT     4
0006 01  8480000F   00010           A     I AUX1
0008 0   1088       00011           SLT     8
0009 01  D480000F   00012           STO   I AUX1
000B 01  74010000   00013           MDX   L BYT1,+1
000D 01  4C800000   00014           BSC   I BYT1
000F 0   0000       00015    AUX1   DC      *-*
0010 0   000F       00016    F      DC      /000F
0012                00017           END

    000 OVERFLOW SECTORS SPECIFIED
    000 OVERFLOW SECTORS REQUIRED
    003 SYMBOLS DEFINED
     NO ERROR(S) AND  NO WARNING(S)  FLAGGED IN ABOVE ASSEMBLY

// DUP

*DELETE            BYT1
D 26 NAME NOT FOUND IN LET/FLET


*STORE    WS  UA  BYT1
CART ID 0005   DB ADDR  28B5   DB CNT   0002
```

```
PAGE   1

// JOB T

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000         0005         0005         0000

V2 M12   ACTUAL 16K  CONFIG 16K

// ASM
*LIST ALL
                    00001     * SUBROUTINE QUE ENCONTRA O BYTE 01 E O BYTE 02
0000      02A23140  00002           ENT     BYTE
0000 0    0000      00003     BYTE  DC      *-*
0001 01   C4800000  00004           LD    I BYTE
0003 0    D01B      00005           STO     AUX1        END. PARAMETRO 1(WORD)
0004 01   74010000  00006           MDX   L BYTE,+1
0006 01   C4800000  00007           LD    I BYTE
0008 0    D017      00008           STO     AUX2        END. PARAMETRO 2(BYTE 1)
0009 01   C480001F  00009           LD    I AUX1        PARAMETRO 1
000B 0    1888      00010           SRT     8
000C 0    1088      00011           SLT     8
000D 0    18C8      00012           RTE     8
000E 01   D4800020  00013           STO   I AUX2        PARAMETRO 2
0010 01   74010000  00014           MDX   L BYTE,+1
0012 01   C4800000  00015           LD    I BYTE
0014 0    D00C      00016           STO     AUX3        END. PARAMETRO 3(BYTE 2)
0015 01   C480001F  00017           LD    I AUX1        PARAMETRO 1
0017 0    1088      00018           SLT     8
0018 0    18C8      00019           RTE     8
0019 01   D4800021  00020           STO   I AUX3        PARAMETRO 3
001B 01   74010000  00021           MDX   L BYTE,+1
001D 01   4C800000  00022           BSC   I BYTE
001F 0    0000      00023     AUX1  DC      *-*
0020 0    0000      00024     AUX2  DC      *-*
0021 0    0000      00025     AUX3  DC      *-*
0022                00026           END

   000 OVERFLOW SECTORS SPECIFIED
   000 OVERFLOW SECTORS REQUIRED
   004 SYMBOLS DEFINED
    NO ERROR(S) AND   NO WARNING(S)  FLAGGED IN ABOVE ASSEMBLY
```

```
PAGE   1

// JOB T

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// ASM
*LIST
                       00001     * SUBROTINA QUE JUNTA 2 BYTES NUMA PALAVRA
0000     26599100      00002           ENT     WORD
0000 0   0000          00003     WORD  DC      *-*
0001 01  C4800000      00004           LD    I WORD
0003 0   D01A          00005           STO     AUX1
0004 01  74010000      00006           MDX   L WORD,+1
0006 01  C4800000      00007           LD    I WORD
0008 0   D016          00008           STO     AUX2
0009 01  C480001F      00009           LD    I AUX2
000B 0   1808          00010           SRA     8
000C 0   1888          00011           SRT     8
000D 01  C480001E      00012           LD    I AUX1
000F 0   1808          00013           SRA     8
0010 0   1088          00014           SLT     8
0011 0   D00C          00015           STO     AUX1
0012 01  74010000      00016           MDX   L WORD,+1
0014 01  C4800000      00017           LD    I WORD
0016 0   D008          00018           STO     AUX2
0017 0   C006          00019           LD      AUX1
0018 01  D480001F      00020           STO   I AUX2
001A 01  74010000      00021           MDX   L WORD,+1
001C 01  4C800000      00022           BSC   I WORD
001E 0   0000          00023     AUX1  DC      *-*
001F 0   0000          00024     AUX2  DC      *-*
0020                   00025           END

    000 OVERFLOW SECTORS SPECIFIED
    000 OVERFLOW SECTORS REQUIRED
    003 SYMBOLS DEFINED
     NO ERROR(S) AND   NO WARNING(S)  FLAGGED IN ABOVE ASSEMBLY
```

```
PAGE  1

// JOB T

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// ASM
*LIST
0000     15914159 00001            ENT     NUMER
0000 0   0000     00002    NUMER   DC      *-*
0001 01  C4800000 00003            LD    I NUMER
0003 0   D008     00004            STO     AUX
0004 01  C480000F 00005            LD    I AUX
0006 0   188C     00006            SRT     12
0007 0   1010     00007            SLA     16
0008 0   1084     00008            SLT     4
0009 01  D480000F 00009            STO   I AUX
000B 01  74010000 00010            MDX   L NUMER,+1
000D 01  4C800000 00011            BSC   I NUMER
000F 0   0000     00012    AUX     DC      *-*
0010             00013            END

    000 OVERFLOW SECTORS SPECIFIED
    000 OVERFLOW SECTORS REQUIRED
    002 SYMBOLS DEFINED
     NO ERROR(S) AND   NO WARNING(S)  FLAGGED IN ABOVE ASSEMBLY
```

```
PAGE   1

// JOB T

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
      INTEGER FUNCTION NUMER(A)
      INTEGER A
         N1 = A/256
         NAUX = N1/16
         N2 = N1 - 16*NAUX
         NUMER = N2+15
      RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR NUMER
 COMMON       0  VARIABLES       4  PROGRAM      44

RELATIVE ENTRY POINT ADDRESS IS 0007 (HEX)

END OF COMPILATION
```

```
PAGE   1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K   CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM

      SUBROUTINE NALFA(N,CHAVE)

      INTEGER N,CHAVE(8),CHAV(8)
      INTEGER B,BRANC,DV,I,J,NAUX,WORD1,WORD2,WORDS

      DATA BRANC/'  '/,DV,I,J/1,5,1/
      DO 3 K=1,8
        CHAVE(K) = BRANC
    3 CONTINUE
    5 DV = 10*DV
      IF(DV-N) 7,7,4
    4 DV = DV/10
      GO TO 6
    7 NAUX = (N/DV)*DV
      B = (N-NAUX)/(DV/10)
      N = NAUX
      CALL BYT1(B)
      CHAV(I) = B
      I = I-1
      IF(I-1) 5,6,5
    6 B = N/DV
      CALL BYT1(B)
      CHAV (I) = B
   10 WORD1 = CHAV (I)
      IF(I-5) 13,15,13
   13 WORD2 = CHAV (I+1)
      GO TO 17
   15 WORD2 = BRANC
   17 CALL WORD (WORD1,WORD2,WORDS)
      CHAVE(J) = WORDS
      J = J+1
      I = I+2
      IF (I-5) 10,10,20
   20 RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR NALFA
 COMMON       0  VARIABLES     22  PROGRAM     200

RELATIVE ENTRY POINT ADDRESS IS 001B (HEX)

END OF COMPILATION

// DUP

*DELETE              NALFA
```

```
PAGE   1
// JOB T
LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000
V2 M12  ACTUAL 16K  CONFIG 16K
// FOR
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      FUNCTION RANDO(K)
      IY=K*899
      IF(IY)5,6,6
    5 IY=IY+32767+1
    6 YFL=IY
      RANDO=YFL/32767
      RETURN
      END
FEATURES SUPPORTED
 ONE WORD INTEGERS
CORE REQUIREMENTS FOR RANDO
 COMMON       0  VARIABLES       6  PROGRAM      44
RELATIVE ENTRY POINT ADDRESS IS 0009 (HEX)
END OF COMPILATION
```

```
PAGE   1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0005         0005         0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
      SUBROUTINE DVREG
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *         NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *          TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *         PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *         NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *          TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *        REGIS,TABCP,TMCMP
      WRITE(NOARR'PAGIN) PAGIN,DISPR
      DISPR = PAGIN
      WRITE(NOARI'1) NOCHI,NOREG,DISPR
      WRITE(NOARR'1) NOREG,DISPR
      RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR DVREG
 COMMON   1130  VARIABLES      0  PROGRAM      34

RELATIVE ENTRY POINT ADDRESS IS 0001 (HEX)

END OF COMPILATION

// DUP

*DELETE              DVREG
CART ID 0005   DB ADDR  20A6   DB CNT   0004

*STORE      WS  UA  DVREG
CART ID 0005   DB ADDR  28A3   DB CNT   0004
```

```
PAGE   1
// JOB

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0005         0005         0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
      SUBROUTINE PGREG(ERRO)
      INTEGER ERRO
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *          NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *          TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *          PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *          NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *          TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *        REGIS,TABCP,TMCMP
      IF(DISPR) 20,20,10
   10   READ(NOARR'DISPR) PAGIN,DISPR
        WRITE(NOARI'1) NOCHI,NOREG,DISPR
        WRITE(NOARR'1) NOREG,DISPR
      RETURN
   20   ERRO = 2
      RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR PGREG
 COMMON    1130  VARIABLES      0  PROGRAM      44

RELATIVE ENTRY POINT ADDRESS IS 0002 (HEX)

END OF COMPILATION

// DUP

*DELETE              PGREG
CART ID 0005   DB ADDR  20A6   DB CNT   0005

*STORE      WS  UA  PGREG
CART ID 0005   DB ADDR  289E   DB CNT   0005
```

```
PAGE   1
// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
      SUBROUTINE DVPAG
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *          NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *          TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *          PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *          NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *          TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *       REGIS,TABCP,TMCMP
      WRITE(NOARI'PAGIN)PAGIN,DISPI
      READ(NOARI'PAGIN) PAGIN,DISPI
      DISPI = PAGIN
        WRITE(NOARI'1) NOCHI,NOREG,DISPR,DISPI
      RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR DVPAG
 COMMON   1130  VARIABLES      0  PROGRAM     36

RELATIVE ENTRY POINT ADDRESS IS 0001 (HEX)

END OF COMPILATION

// DUP

*DELETE              DVPAG
CART ID 0005  DB ADDR  24F8  DB CNT  0004

*STORE      WS  UA  DVPAG
CART ID 0005  DB ADDR  289A  DB CNT  0004
```

```
PAGE   1

// JOB    0005

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
      SUBROUTINE PGPAG(ERRO)
      INTEGER ERRO
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *         NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *        TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *        PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *         NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *         TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *        REGIS,TABCP,TMCMP
      IF(DISPI)20,20,10
   10    READ(NOARI'DISPI)PAGIN,DISPI
         WRITE(NOARI'1) NOCHI,NOREG,DISPR,DISPI
      RETURN
   20    ERRO = 3
      RETURN
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS

CORE REQUIREMENTS FOR PGPAG
 COMMON  1130  VARIABLES     0  PROGRAM     38

RELATIVE ENTRY POINT ADDRESS IS 0002 (HEX)

END OF COMPILATION

// DUP

*DELETE            PGPAG
CART ID 0005   DB ADDR  20A6   DB CNT   0004

*STORE     WS  UA  PGPAG
CART ID 0005   DB ADDR  289A   DB CNT   0004
```

ANEXO 04

MANUAL DO USUÁRIO

ÍNDICE

01 - APRESENTAÇÃO

O Sistema de gerenciamento apresentado foi desenvolvido usando-se como estrutura uma variação de árvore B denominada, segundo SOUZA e TORRES, árvore B^+ com prefixo simples e possibilita tanto o acesso aleatório como o sequencial, para chaves numéricas ou alfas (sem caracteres especiais).

O usuário interage com o sistema através de chamadas a subrotinas, programando em linguagem FORTRAN. As subrotinas de denvolvidas para o usuário são sete: CARGA, BUSCACHAVE, INSERÇÃO, REMO(V)ÇÃO, BUSCAPROXIMO, BUSCANTERIOR, BUSCAPOSIÇÃO (a parte sublinhada é o nome dado a elas no sistema). As demais subrotinas são usadas pelo sistema sem interferências do usuário, por isso não constam deste manual e a subrotina CRIA não está pronta para uso.

Compete ao usuário tomar algumas providências, em seu programa, para uso do sistema, tais como: construir a área de COMMON, dimensionar matrizes, construir um arquivo cabeçalho e dimensionar e criar área para os demais arquivos (são três os arquivos: cabeçalho, índice e de registros).

No item dimensionamento (05) mostramos as dimensões usadas nas matrizes do sistema, que devem constar do programa principal, e explicamos como dimensionar os setores necessários aos arquivos.

Os ítens 11 e 12 alertam para as restrições do sistema e precauções necessárias ao seu uso.

A leitura completa do manual é indispensável a uma boa utilização do sistema e chamamos a atenção para os programas exemplos, que pretendem esclarecer o seu uso.

02 - DESCRIÇÃO E USO DAS SUBROTINAS

2.1 - Subrotina CARGA (tipo da carga, erro)

função: 1. Constrói os cabeçalhos para os arquivos de registros e índices, com parte das informações já fornecidas pelo usuário no arquivo cabeçalho.

2. Constrói um índice para o arquivo indicado, que deverá estar ordenado crescentemente segundo a chave do índice.

Obs. 1 - Se a chave do índice não corresponde à chave segundo a qual o arquivo está ordenado não existe carga maciça e sim inserção um a um, por isso, distinguimos o tipo da carga.

Tipo da carga = 0 carga maciça (função 1 e 2)

$\neq$ 0 só cabeçalhos (função 1)

Obs. 2 - As demais informações necessárias à subrotina estão na área de COMMON.

Obs. 3 - A carga maciça exige que os registros já estejam armazenados no arquivo, em ordem crescente segundo a chave considerada.

Códigos de erro:

0 - tudo bem

1 - o arquivo índice está mal dimensionado.

2.2 - Subrotina BUSCA (vetor chave, apontador para o registro, erro)

função: Pesquisa se determinada chave pertence ao índice e fornece ao usuário a posição do registro a ser lido.

Obs. 1 - As demais informações necessárias à subrotina estão na área de COMMON.

Códigos de erro:

0 - tudo bem

1 - chave não encontrada no índice

2 - chave não encontrada no arquivo.

2.3 - Subrotina PROX (apontador para o registro, erro)

função: Acessa a primeira chave imediatamente seguinte à acessada por último e fornece ao usuário a posição em que se encontra o registro associado à chave desejada.

Obs.1 - Seu uso é posterior à subrotina BUSCA ou a si própria.

Obs.2 - As demais informações estão na área de COMMON.

Códigos de erro:

0 - tudo bem

1 - não existe próximo.

2.4 - Subrotina ANTER (apontador para o registro, erro)

função: Acessa a primeira chave imediatamente anterior à acessada por último e fornece ao usuário a posição em que se encontra o registro associado à chave desejada.

Obs.1 - Seu uso é posterior à subrotina BUSCA ou a si própria.

Obs.2 - As demais informações estão na área de COMMON.

Códigos de erro:

0 - tudo bem

1 - não existe anterior.

2.5 - Subrotina POSIC (posição desejada, apontador para o registro, erro)

função: Acessa a chave correspondente à posição desejada e fornece ao usuário a posição em que se encontra o

registro associado à chave desejada.

Obs.1 - Seu uso é livre.

Obs.2 - A variável do primeiro argumento (posição desejada) deve assumir valores como 1, 2, 3, ..., etc.

Obs.3 - As demais informações necessárias estão na área de COMMON.

Códigos de erro:

0 - tudo bem

1 - não existe a posição no índice.

2.6 - Subrotina INSER (vetor chave, apontador para o registro , condição da inserção, erro)

função: Controla a inserção nos índices associados ao arquivo.

Obs.1 - A inserção poderá ser feita só no índice (condição ≠ 1) quando houver mais de um índice associado ao arquivo e já foi feita a inserção do registro no arquivo ou no índice e no arquivo (Condição ≠ 1 , quando só há um índice associado ao arquivo ou é o primeiro índice a sofrer a inserção).

Obs.2 - Compete ao usuário controlar a inserção nos demais índices associados fornecendo o valor do apontador para o registro e atribuindo à condição um valor diferente de um.

Obs.3 - As demais informações necessárias estão na área de COMMON.

Códigos de erro:

0 - tudo bem

1 - a chave já consta do índice.

3 - falta espaço no índice

4 - falta espaço no arquivo.

2.7 - Subrotina REMOV (vetor chave, erro)

função: Remover, quando encontrada, a chave do índice e a informação associada do arquivo (caso haja um único índice associado a ela).

Obs. : As demais informações necessárias estão na área de COMMON.

Códigos de erro:

0 - tudo bem

1 - a chave não consta do índice

2 - a chave não consta do arquivo.

Obs. : Em todas essas subrotinas o vetor usado como chave deve ter dimensão oito.

03 - DESCRIÇÃO DA ÁREA DE COMMON

O usuário deverá construir as seguintes áreas de COMMON, onde todas as variáveis são inteiras de uma palavra, descritas no item seis (6).

```
COMMON   ALFA, CHV, DISPI, DISPR,  I1,  J1,  K1,  MEIO,
         NOARC, NOARI, NOARR, NOCHI, NOIND, NOREG,
         PAGIN, PRIN, RAIZ, TMCHV, TMPAG,
         TMREG, TOPO, ULTMA
COMMON   BUFF, FTOCP, M, NOCHV, NOPAG, PLAPT,
         PLPAG, REGIS, TABCP, TMCMP
```

OBS.: Usamos duas declarações de COMMON para separar as variáveis simples (primeira declaração) das matrizes (segunda declaração).

04 - DETERMINAÇÃO DA ALTURA E NÚMERO DE NÓS DO ÍNDICE.

$h \simeq \log_g(\text{nfolhas}) + 2$

sendo: g - ordem da árvore

(para chaves de tamanho constante teremos g = k)

n folhas - número de folhas da árvore

(consideramos a chave como tendo tamanho constante, verificamos quantas cabem na folha e daí deduzimos quantas são necessárias para o índice).

Obs.1 - O número de nós será estimado como sendo o dobro do número de folhas.

Obs.2 - Esses parâmetros h e número de nós deverão ser dimensionado no programa do usuário com um tamanho maior, prevendo-se a expansão.

Obs.3 - A altura estimada não pode ser superior a 10, devido ao dimensionamento usado na programação.

05 - DIMENSIONAMENTOS

Compete ao usuário dimensionar as matrizes que constam do COMMON e calcular o espaço necessário para os arquivos.

a) dimensionamento das matrizes (todas inteiras de uma palavra).

BUFF(3,320),

PLAPT(10) e PLPAG(10),

REGIS(80),

TABCP(20) e TMCMP(20),

TABFRM(4,20)*

onde:

10 = altura estimada para a árvore

320 = tamanho da página (em palavras)

80 = tamanho máximo do registro (em palavras)

Obs. (*) Só é usada pela subrotina CRIA, que ainda não está implantada.

b) dimensionamento dos arquivos.

São três os arquivos a serem criados pelo usuário:

NOARC - É o arquivo cabeçalho que conterá informações necessárias aos demais arquivos. Sua construção é de total responsabilidade do usuário,

Ocupa 62 palavras podendo portanto ser definido como um único registro de 80 palavras.

NOARI - É o arquivo índice, onde estarão as páginas da árvore e o cabeçalho do arquivo. O cabeçalho desse arquivo ocupa 69 palavras. Para o 1130 o tamanho ideal das páginas é 320 palavras.

Para o cálculo do número de registros necessários a esse arquivo siga o seguinte raciocínio:

Quantas folhas serão necessárias?

- Considere N = tamanho máximo da chave + 1
  nº chaves nas folhas = 210/N

$$\text{então nº folhas} = \frac{\text{nº de chaves no índice}}{\text{nº de chaves nas folhas}}$$

Qual o número total de nós? (nº de setores do arquivo)
- Pode ser estimado como sendo o dobro do número de folhas. No caso de prever-se um grande número de inserções posteriores deve-se aumentar esse número.

NOARR - É o arquivo de registros. Seu dimensionamento vai depender do número e do tamanho definido para os registros. Como o cabeçalho desse arquivo ocupa 151 palavras precisamos verificar quantos registros serão necessários para êle e considerá-lo no total.

$$\text{nº de registros do cabeçalho} = \frac{151}{\text{tamanho do registro}} + 1$$

nº total de registros = nº de registros do cabeçalho + nº de registros de informação.

nº de registros por setor = 320 / tamanho do registro

$$\text{nº de setores do arquivo} = \frac{\text{nº total de registros}}{\text{nº de registros por setor}}$$

06 - DEFINIÇÃO DOS ELEMENTOS DOS CABEÇALHOS

1. NOARI - nº/nome do arquivo índice.
2. NOARR - nº/nome do arquivo de informações.
3. MAXRG - nº máximo de registros permitidos (no cartão).
4. MAXPG - nº máximo de páginas permitidas.
5. NOREG - nº atual de registros no arquivo.
6. NOCHI - nº atual de chaves no índice.
7. TMCHV - tamanho da chave = 0 tamanho fixo
   ≠ 0 tamanho variável
   (em palavras - é armazenada como alfanumérica no índice)
8. TMPAG - tamanho da página (em palavras)
9. TMREG - tamanho do registro (em palavras)
10. NOCMP - nº de campos no registro
11. TABCP(20) - tabela de campos - contêm os números das palavras
    TMCMP(20) onde têm início os campos e os seus tamanhos.
12. CHV - nº do campo chave no registro
13. ALFA - tipo da chave ≠ 1 numérica
    = 1 alfanumérica
14. RAIZ - número da página raiz do índice
15. DISPI - apontador topo da pilha de páginas disponíveis (no arquivo índice)
16. DISPR - apontador topo da pilha de registros disponíveis (no arquivo registros)
17. PRIM - apontador primeira folha do índice
18. ULTMA - apontador última folha do índice
19. TBFRM - tabela de formatos
20. NOIND - número de índices associados ao arquivo
21. INDCS(10) - número dos índices associados ao arquivo
22. DOCUM(12) - informações documentacionais

07 - CONSTRUÇÃO DO ARQUIVO CABEÇALHO

NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,
TMREG,NOCMP,TABCP,CHV,ALFA,DOCUM.

Compete ao usuário:

1 - Criar espaço em disco para o arquivo cabeçalho

2 - Construir o cartão cabeçalho, segundo a ordem indicada.

3 - Ler o cartão cabeçalho e gravá-lo no arquivo cabeçalho.

08 - PARÂMETROS DAS SUBROTINAS

1. BALANCEAMENTO (erro)
2. BUSCACHAVE (vetor chave, apontador para o registro, erro)
3. BUSCANTERIOR (apontador para o registro, erro)
4. BUSCAPOSIÇÃO (posição, apontador para o registro, erro)
5. BUSCAPRÓXIMO (apontador para o registro, erro)
6. CARGA (tipo da carga, erro)
7. CRIA (erro)
8. INSERÇÃO (vetor chave, apontador para o registro, condição da inserção, erro)
9. REMOÇÃO (vetor chave, erro)

09 - CÓDIGOS DE ERRO

1. BALANCEAMENTO
   - 0 - tudo bem
   - 3 - falta de página

2. BUSCACHAVE
   - 0 - tudo bem
   - 1 - chave não encontrada no índice
   - 2 - chave não encontrada no arquivo

3. BUSCANTERIOR
   - 0 - tudo bem
   - 1 - não existe a posição no índice

4. BUSCAPOSIÇÃO
   - 0 - tudo bem
   - 1 - não existe próximo

5. BUSCAPRÓXIMO
   - 0 - tudo bem
   - 1 - não existe próximo

6. CARGA
   - 0 - tudo bem
   - 1 - o arquivo índice está mal dimensionado

7. CRIA
   - 0 - tudo bem
   - 1 - erro de formato (caracter inválido)
   - 2 - os cartões estão em ordem errada

8. INSERÇÃO
   - 0 - tudo bem
   - 1 - a chave já consta do índice
   - 3 - falta de espaço no índice
   - 4 - falta de espaço no arquivo

9. REMOÇÃO
   - 0 - tudo bem
   - 1 - a chave não consta do índice
   - 2 - a chave não consta do arquivo

10 - OPÇÃO ENTRE O USO DA CRIA E/OU CARGA

CRIA - Usa o arquivo cabeçalho (1) para saber em qual arquivo do disco deverá armazenar seus registros em cartão (arquivo 4).

CARGA - Usa o arquivo cabeçalho (1) para construir os cabeçalhos dos arquivos de registros (2) e de índices (3). Como já encontra os registros armazenados em ordem, segundo a chave principal, é a responsável pela construção do índice.

Se o usuário optar por usar unicamente a CARGA é responsável por fornecer os registros ordenados segundo a chave principal. Nesse caso a tabela de formatos não tem utilidade.

11 - RESTRIÇÕES

1 - O tamanho máximo do registro é 1 cartão de 80 colunas.
2 - O número máximo de registros permitidos é 32767 (capacidade de uma palavra)
3 - As chaves numéricas devem ser inteiros positivos.
4 - O número máximo de campos para o registro é 20.
5 - Se a chave alfanumérica tiver brancos intermediários o tamanho da chave deve ser definido como fixo (1).
6 - O tamanho máximo da chave é de 8 palavras (16 caracteres)
7 - É proibido o uso de parênteses internos no cartão formato.
8 - A altura máxima permitida para a árvore é 10.

12 - PRECAUÇÕES A SEREM TOMADAS

1 - O programa do usuário deve conter o cartão
*ONE WORD INTEGERS

2 - As variáveis do COMMON são todas do tipo inteiro e devem ser dimensionadas conforme a orientação dada.

3 - O usuário deve controlar a inserção e remoção no caso de haver mais de um índice associado ao arquivo.

4 - O vetor correspondente à chave deve estar dimensionado com oito nos programas de utilização, por ser esta a dimensão usada nos subprogramas do sistema. Em se tratando de chave numérica, esta deverá ser armazenada na primeira posição do vetor usado como chave. Deve-se cuidar para não carregar valores estranhos à chave de trabalho no vetor usado como chave.

5 - Se o usuário vai usar diretamente a CARGA (carga maciça propriamente dita) deve atentar para o fato dos registros deverem estar armazenados em ordem crescente, segundo a sua chave, no arquivo de registros.

6 - O número do arquivo cabeçalho (NOARC) deve ser especificado nos programas de uso da subrotina CARGA.

7 - Quando da construção do arquivo de registros (leitura em cartão e gravação) o controle deve ser feito comparando-se o número de registros lidos com o número máximo permitido no arquivo.

8 - Caso a subrotina CARGA não tenha sido usada anteriormente no programa principal o usuário deve ler os cabeçalhos dos arquivos índice.

9 - No caso do usuário desejar criar outros índices associados ao arquivo, cuidar para que as inserções a serem feitas nos novos índices conheçam os apontadores para os registros no arquivo.

13 - EXEMPLOS DE UTILIZAÇÃO

Exemplo - Uso da Carga maciça
- 01 - (Chaves alfas de tamanho constantes e "afastadas")
- 01' - (Chaves alfas de tamanho constantes e "próximas")
- 01'' - (Chaves alfas de tamanho variável)
- 01''' - (Chaves numéricas)

Exemplo 02 - Uso das Buscas

Exemplo 03 - Uso da Remoção

Exemplo 04 - Uso da Inserção um a um.

Obs.: As páginas dos programas exemplo têm tamanho de 80 palavras, por razões didáticas.

```
PAGE   1
// JOB     0005

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
   0000        0005        0005         0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
*IOCS(CARD,1132PRINTER,DISK)

C      PROGRAMA EXEMPLO DE USO DA CARGA
C                    01     ( CHAVES ALFAS DE TAMANHO CONSTANTE E !!AFASTADAS!!)

       DEFINE FILE 12(1,80,U,I1),20(100,31,U,J1),32(15,80,U,K1)
       INTEGER DOCUM(10),ERRO
       INTEGER CODGO,SNOME(5),NOME(7),IDADE,SEXO,PROF,END(15)
       INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
      *          NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
      *          TMREG,TOPO,ULTMA
       INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
      *          PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
       COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
      *          NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
      *          TMCHV,TMPAG,TMREG,TOPO,ULTMA
       COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
      *          REGIS,TABCP,TMCMP
       NOARC = 12
       READ(2,11)NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,TMREG,NOCMP,
      *          TABCP,TMCMP,CHV,ALFA,DOCUM
   11  FORMAT(9I2,20I2/20I2,2I3,10A2)

C      DEFINE REGISTRO INICIAL DO ARQUIVO
       I = 160/TMREG + 2
       J1 = I
       I1 = 1
  100  READ(2,5) CODGO,SNOME, NOME,IDADE,SEXO,PROF,END
    5    FORMAT(I6,5A2,7A2,I3,I1,I2,15A2)
       IF(CODGO) 1,2,2
    2  NOREG = NOREG+1
       IF(NOREG+I-MAXRG) 3,3,4
    3  WRITE(NOARR'J1) CODGO,SNOME,NOME,IDADE,SEXO,PROF,END
       GO TO 100

    1  WRITE(NOARC'1) NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,TMREG,
      *          NOCMP,TABCP,TMCMP,CHV,ALFA,DOCUM
       ERRO = 0
          CALL CARGA(0,ERRO)

C      VERIFICANDO A CARGA NO NIVEL DAS FOLHAS
C
       WRITE(3,20)
   20  FORMAT ('1',T5,'NIVEL DAS FOLHAS',/,T5,
      *    'AS CHAVES SAO ALFAS DE TAMANHO CONSTANTE = 5')
       DO 200 L = PRIM, ULTMA
       READ(NOARI'L) REGIS
       WRITE(3,30) (REGIS(I),I=1,5)
   30  FORMAT(/,81('*'),/,T5,'NOPAG =',I5,'  NOCHV =',I5,'  FTOCP =',I5,
```

```
PAGE    2

     *' MARCA DE PAG. =',I3,' PROX =',I4,/)

C    AS CHAVES DO EXEMPLO TEM TAMANHO IGUAL 5
      K = REGIS(3) + 4
      WRITE(3,40) (REGIS(I),I=6,K)
   40 FORMAT(5(2X,5A2,I4))
  200  CONTINUE

C      NIVEL DA RAIZ

      WRITE(3,50)
   50 FORMAT ('1',T5,'NIVEL DA RAIZ',//)

      READ(NOARI'RAIZ) REGIS
      WRITE(3,60) (REGIS(I),I=1,5)
   60 FORMAT(/,81('*'),/,T5,'NOPAG =',I5,'  NOCHV =',I5,'  FTOCP =',I5,
     *'  MARCA DE PAG. =',I2,'  PO =',I3,/)
      K = REGIS(3) + 4
      WRITE(3,70) (REGIS(I),I=6,K)
   70 FORMAT(20(1X,A2))
      CALL EXIT

    4 WRITE(3,10)
   10 FORMAT(1X,'LOTADA A AREA DISPONIVEL PARA REGISTROS')
      CALL EXIT
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON   1130  VARIABLES     72  PROGRAM     540

END OF COMPILATION

// XEQ          1

*FILES(12,CAB1),(20,REGS1),(32,INDC1)
```

NÍVEL DAS FOLHAS

AS CHAVES SÃO ALFAS DE TAMANHO CONSTANTE = 5

```
**********************************************************************************
  NOPAG =    3  NOCHV =    9  FTOCP =   55 MARCA DE PAG. =  1 PROX =   4

ABDAS        7  ABE         8  ABECHE      9  ABUD        10  ARDUIN      11
BARBIERI    12  BOMURA     13  BONA       14  BORIN       15

**********************************************************************************
  NOPAG =    4  NOCHV =    9  FTOCP =   55 MARCA DE PAG. =  1 PROX =   5

BORGHI      16  BOTELHO    17  CAOBIANCO  18  CAPASSO     19  CARMINO     20
CARMO       21  CAVALIERI  22  CAYRES     23  CORDEIRO    24

**********************************************************************************
  NOPAG =    5  NOCHV =    9  FTOCP =   55 MARCA DE PAG. =  1 PROX =   6

CORTES      25  CORTEZ     26  COSTA      27  CRUZ        28  CUTULO      29
DERNER      30  DUARTE     31  FERRAS     32  FERRAZ      33

**********************************************************************************
  NOPAG =    6  NOCHV =    9  FTOCP =   55 MARCA DE PAG. =  1 PROX =   7

FERREIRA    34  FUMERO     35  GROSSI     36  IMAY        37  ITO         38
JESUS       39  JOSEPETTI  40  KIDO       41  KOGA        42

**********************************************************************************
  NOPAG =    7  NOCHV =    9  FTOCP =   55 MARCA DE PAG. =  1 PROX =   8

KUTNE       43  LEITE      44  LIBARDI    45  LIMA        46  LIPPI       47
LOMBARDI    48  MALUF      49  MAIMONE    50  MORAES      51

**********************************************************************************
  NOPAG =    8  NOCHV =    9  FTOCP =   55 MARCA DE PAG. =  1 PROX =   9

MORAIS      52  MORELLI    53  NAUFAL     54  NORONHA     55  ORFAO       56
OSSUCHI     57  PACE       58  PACHECO    59  PALLONE     60

**********************************************************************************
  NOPAG =    9  NOCHV =    9  FTOCP =   55 MARCA DE PAG. =  1 PROX =  10

PIGOZZO     61  PIOLLI     62  PRATIS     63  RABITO      64  RIBEIRO     65
ROCHA       66  RUI        67  SANCHES    68  SANTINI     69

**********************************************************************************
  NOPAG =   10  NOCHV =    9  FTOCP =   55 MARCA DE PAG. =  1 PROX =  11

SANTINOLI   70  SANTOS     71  SHIRAISHI  72  SOARES      73  TANAKA      74
TAVARES     75  TOZZI      76  TRALDI     77  UYEDA       78

**********************************************************************************
  NOPAG =   11  NOCHV =    9  FTOCP =   55 MARCA DE PAG. =  1 PROX =  12

VIDOTTI     79  VINHAS     80  WADA       81  WECO        82  WELZ        83
WOLFF       84  XANDER     85  XAVIER     86  YABIKU      87

**********************************************************************************
  NOPAG =   12  NOCHV =    5  FTOCP =   31 MARCA DE PAG. =  1 PROX =   0

YAMAO       88  YAMAOKA    89  ZAGATTI    90  ZAGO        91  ZIOBER      92
```

NIVEL DA RAIZ

****

NOPAG = 13 NOCHV = 9 FTOCP = 29 MARCA DE PAG. = 0 PO = 3

BO RG CO RT FE RR EI KU MO RA IS PI SA NT
IN OL VI YA MA

```
PAGE  1
// JOB    0005

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000         0005         0005         0000

V2 M12  ACTUAL 16K  CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
*IOCS(CARD,1132PRINTER,DISK)

C      PROGRAMA EXEMPLO DE USO DA CARGA
C                    01'    ( CHAVES ALFAS DE TAMANHO CONSTANTE E ''PROXIMAS'')


      DEFINE FILE 12(1,80,U,I1),20(100,31,U,J1),32(15,80,U,K1)
      INTEGER DOCUM(10),ERRO
      INTEGER CODGO,SNOME(5),NOME(7),IDADE,SEXO,PROF,END(15)
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCH
     *          NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *            TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *          PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *          NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *            TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *          REGIS,TABCP,TMCMP
      NOARC = 12
      READ(2,11)NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,TMREG,NOCMP,
     *          TABCP,TMCMP,CHV,ALFA,DOCUM
   11 FORMAT(9I3,20I2/20I2,2I3,10A2)

C     DEFINE REGISTRO INICIAL DO ARQUIVO
      I = 160/TMREG + 2
      J1 = I
      I1 = 1
  100 READ(2,5) CODGO,SNOME, NOME,IDADE,SEXO,PROF,END
    5   FORMAT(I6,5A2,7A2,I3,I1,I2,15A2)
      IF(CODGO) 1,2,2
    2 NOREG = NOREG+1
      IF(NOREG+I-MAXRG) 3,3,4
    3 WRITE(NOARR'J1) CODGO,SNOME,NOME,IDADE,SEXO,PROF,END
      GO TO 100

    1 WRITE(NOARC'1) NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,TMREG,
     *          NOCMP,TABCP,TMCMP,CHV,ALFA,DOCUM
      ERRO = 0
         CALL CARGA(0,ERRO)

C     VERIFICANDO A CARGA NO NIVEL DAS FOLHAS
C
      WRITE(3,20)
   20 FORMAT ('1',T5,'NIVEL DAS FOLHAS',/,T5,
     *   'AS CHAVES SAO ALFAS DE TAMANHO CONSTANTE = 5')
      DO 200 L = PRIM, ULTMA
      READ(NOARI'L) REGIS
      WRITE(3,30) (REGIS(I),I=1,5)
   30 FORMAT(/,81('*'),/,T5,'NOPAG =',I5,'  NOCHV =',I5,'  FTOCP =',I5,
```

```
PAGE    2
      *' MARCA DE PAG. =',I3,' PROX =',I4,/)

C     AS CHAVES DO EXEMPLO TEM TAMANHO IGUAL 5
      K = REGIS(3) + 4
      WRITE(3,40) (REGIS(I),I=6,K)
   40 FORMAT(5(2X,5A2,I4))
  200  CONTINUE

C       NIVEL DA RAIZ

      WRITE(3,50)
   50 FORMAT ('1',T5,'NIVEL DA RAIZ',//)

      READ(NOARI'RAIZ) REGIS
      WRITE(3,60) (REGIS(I),I=1,5)
   60 FORMAT(/,81('*'),/,T5,'NOPAG =',I5,'  NOCHV =',I5,'  FTOCP =',I5,
     *'  MARCA DE PAG. =',I2,'  PO =',I3,/)
      K = REGIS(3) + 4
      WRITE(3,70) (REGIS(I),I=6,K)
   70 FORMAT(20(1X,A2))
      CALL EXIT

    4 WRITE(3,10)
   10 FORMAT(1X,'LOTADA A AREA DISPONIVEL PARA REGISTROS')
      CALL EXIT
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON   1130  VARIABLES     72  PROGRAM     540

END OF COMPILATION

// XEQ          1

*FILES(12,CAB1),(20,REGS1),(32,INDC1)
```

NIVEL DAS FOLHAS
AS CHAVES SAO ALFAS DE TAMANHO CONSTANTE = 5

****

NOPAG = 3 NOCHV = 9 FTOCP = 55 MARCA DE PAG. = 1 PROX = 4

MERANCA	7	MILAN	8	MILANI	9	MINETTO	10	MIORALI	11
MIRANDA	12	MODESTO	13	MOLINA	14	MOLITOR	15		

****

NOPAG = 4 NOCHV = 9 FTOCP = 55 MARCA DE PAG. = 1 PROX = 5

MONTEIRO	16	MOREIRA	17	MORILHA	18	MORO	19	MORTARI	20
MOURA	21	MULLER	22	MUNHOZ	23	MUNIZ	24		

****

NOPAG = 5 NOCHV = 9 FTOCP = 55 MARCA DE PAG. = 1 PROX = 6

MURAKAMI	25	MURAMOTO	26	MURAN	27	MURANI	28	NAGAO	29
NAGAYA	30	NAKAMURA	31	NANTES	32	NARDI	33		

****

NOPAG = 6 NOCHV = 9 FTOCP = 55 MARCA DE PAG. = 1 PROX = 7

NARDO	34	NASCIMENTO	35	NASSER	36	NEGRO	37	NEIVA	38
NEMIR	39	NERI	40	NEVES	41	NICIO	42		

****

NOPAG = 7 NOCHV = 9 FTOCP = 55 MARCA DE PAG. = 1 PROX = 8

NIERO	43	NISHI	44	NIWA	45	NOBILE	46	NOBREGA	47
NOIVO	48	NONAKA	49	NONINO	50	NOVAES	51		

****

NOPAG = 8 NOCHV = 9 FTOCP = 55 MARCA DE PAG. = 1 PROX = 9

NUMATA	52	NUNES	53	NUVOLI	54	OBA	55	OBARA	56
OBERHAUSER	57	OBRAL	58	OBUTI	59	OCHIRO	60		

****

NOPAG = 9 NOCHV = 9 FTOCP = 55 MARCA DE PAG. = 1 PROX = 10

ODA	61	OGAMA	62	OGASAWARA	63	OGATA	64	OGAVA	65
OGINO	66	OGO	67	OGUIDO	68	OHARA	69		

****

NOPAG = 10 NOCHV = 9 FTOCP = 55 MARCA DE PAG. = 1 PROX = 11

OKADA	70	OKAMOTO	71	OKANO	72	OLAVO	73	OLIVEIRA	74
ONO	75	ORTEGA	76	OTUNHO	77	PACHER	78		

****

NOPAG = 11 NOCHV = 9 FTOCP = 55 MARCA DE PAG. = 1 PROX = 12

PACOLA	79	PADILHA	80	PADUANO	81	PEDALINO	82	PEDRA	83
PEDRAO	84	PEDRAZANI	85	PEDRINI	86	PEDROSO	87		

****

NOPAG = 12 NOCHV = 9 FTOCP = 55 MARCA DE PAG. = 1 PROX = 0

PIAI	88	PIAS	89	PICCINI	90	PICCOLI	91	PICELI	92
PIEDADE	93	PINHEIRO	94	PINHO	95	POGGIAN	96		

NIVEL DA RAIZ

```
**********************************************************************************
   NOPAG =   13  NOCHV =   9  FTOCP =   25  MARCA DE PAG. = 0  PO =  3

MO NT    MU RA    NA RD O     NI ER    NU    OD    OK    PA
CO    PI
```

```
PAGE   1
// JOB    0005
LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000         0005         0005         0000
V2 M12   ACTUAL 16K  CONFIG 16K
// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
*IOCS(CARD,1132PRINTER,DISK)
C      PROGRAMA EXEMPLO DE USO DA CARGA
C                    01''  ( CHAVES ALFAS DE TAMANHO VARIAVEL)

      DEFINE FILE 12(1,80,U,I1),20(100,31,U,J1),32(15,80,U,K1)
      INTEGER DOCUM(10),ERRO
      INTEGER CODGO,SNOME(5),NOME(7),IDADE,SEXO,PROF,END(15)
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *          NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *          TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *          PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *          NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *          TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *          REGIS,TABCP,TMCMP
      NOARC = 12
      READ(2,11)NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,TMREG,NOCMP,
     *          TABCP,TMCMP,CHV,ALFA,DOCUM
   11 FORMAT(9I3,20I2/20I2,2I3,10A2)

C     DEFINE REGISTRO INICIAL DO ARQUIVO
      I = 160/TMREG + 2
      J1 = I
      I1 = 1
  100 READ(2,5) CODGO,SNOME, NOME,IDADE,SEXO,PROF,END
    5   FORMAT(I6,5A2,7A2,I3,I1,I2,15A2)
      IF(CODGO) 1,2,2
    2 NOREG = NOREG+1
      IF(NOREG+I-MAXRG) 3,3,4
    3 WRITE(NOARR'J1) CODGO,SNOME,NOME,IDADE,SEXO,PROF,END
      GO TO 100

    1 WRITE(NOARC'1) NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,TMREG,
     *          NOCMP,TABCP,TMCMP,CHV,ALFA,DOCUM
      ERRO = 0
         CALL CARGA(0,ERRO)

C     VERIFICANDO A CARGA NO NIVEL DAS FOLHAS
C
      WRITE(3,20)
   20 FORMAT ('1',T5,'NIVEL DAS FOLHAS',/,T5,
     *   'AS CHAVES SAO ALFAS DE TAMANHO VARIAVEL')
      DO 200 L = PRIM, ULTMA
      READ(NOARI'L) REGIS
      WRITE(3,30) (REGIS(I),I=1,5)
   30 FORMAT(/,81('*'),/,T5,'NOPAG =',I5,'  NOCHV =',I5,'  FTOCP =',I5,
```

```
PAGE    2

     *' MARCA DE PAG. =',I3,' PROX =',I4,/)

C    AS CHAVES DO EXEMPLO TEM TAMANHO VARIAVEL
      K = REGIS(3) + 4
     WRITE(3,70) (REGIS(I),I=6,K)
 200 CONTINUE

C     NIVEL DA RAIZ

      WRITE(3,50)
  50 FORMAT ('1',T5,'NIVEL DA RAIZ',//)

      READ(NOARI'RAIZ) REGIS
      WRITE(3,60) (REGIS(I),I=1,5)
  60 FORMAT(/,81('*'),/,T5,'NOPAG =',I5,'  NOCHV =',I5,'  FTOCP =',I5,
     *' MARCA DE PAG. =',I2,'  PO =',I3,/)
      K = REGIS(3) + 4
      WRITE(3,70) (REGIS(I),I=6,K)
  70 FORMAT(20(1X,A2))
      CALL EXIT

   4 WRITE(3,10)
  10 FORMAT(1X,'LOTADA A AREA DISPONIVEL PARA REGISTROS')
      CALL EXIT
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON   1130  VARIABLES     72  PROGRAM    532

END OF COMPILATION

// XEQ          1

*FILES(12,CAB1),(20,REGS1),(32,INDC1)
```

```
   NIVEL DAS FOLHAS
   AS CHAVES SAO ALFAS DE TAMANHO VARIAVEL

*************************************************************************************
   NOPAG =    3  NOCHV =   12  FTOCP =   50 MARCA DE PAG. =  1 PROX =    4

AB DA S     AB E     AB EC HE     AB UD     AR DU IN     BA RB
IE RI     BO MU RA     BO NA     BO RG HI     BO RI N     BO TE
LH O     CA OB IA NC O

*************************************************************************************
   NOPAG =    4  NOCHV =   12  FTOCP =   53 MARCA DE PAG. =  1 PROX =    5

CA PA SS O     CA RM IN O     CA RM O     CA VA LI ER I
CA YR ES     CO RD EI RO     CO RT ES     CO RT EZ     CO ST A
   CR UZ     CU TU LO     DE RN ER

*************************************************************************************
   NOPAG =    5  NOCHV =   13  FTOCP =   52 MARCA DE PAG. =  1 PROX =    6

DU AR TE     FE RR AS     FE RR AZ     FE RR EI RA     FU ME RO
   GR OS SI     IM AY     IT O     JE SU S     JO SE PE TT I
   KI DO     KO GA     KU TN E

*************************************************************************************
   NOPAG =    6  NOCHV =   12  FTOCP =   53 MARCA DE PAG. =  1 PROX =    7

LE IT E     LI BA RD I     LI MA     LI PP I     LO MB AR DI
   MA LU F     MA IM ON E     MO RA ES     MO RA IS     MO RE
LL I     MA UF AL     NO RO NH A

*************************************************************************************
   NOPAG =    7  NOCHV =   12  FTOCP =   52 MARCA DE PAG. =  1 PROX =    8

OR FA O     OS SU CH I     PA CE     PA CH EC O     PA LL ON
E     PI GO ZZ O     PI OL LI     PR AT IS     RA BI TO     RI
RE IR O     RO CH A     RU I

*************************************************************************************
   NOPAG =    8  NOCHV =   11  FTOCP =   52 MARCA DE PAG. =  1 PROX =    9

SA NC HE S     SA NT IN I     SA NT IN OL I     SA NT OS
SH IR AI SH I     SO AR ES     TA NA KA     TA VA RE S     TO
ZZ I     TR AL DI     ( UY ED A     +

*************************************************************************************
   NOPAG =    9  NOCHV =   14  FTOCP =   56 MARCA DE PAG. =  1 PROX =    0

VI DO TT I     VI NH AS & WA DA     WE CO     WE LZ     WO LF
F     XA ND ER     XA VI ER     YA BI KU     YA MA O     YA MA
OK A     ZA GA TT I     ZA GO $ ZI OB ER #
```

NIVEL DA RAIZ

****

NOPAG = 11 NOCHV = 6 FTOCP = 14 MARCA DE PAG. = 0 PO = 3

CA PA DU LE OR SA VI

```
PAGE   1

// JOB    0005

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000

V2 M12   ACTUAL 16K  CONFIG 16K

// FOR
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
*IOCS(CARD,1132PRINTER,DISK)

C      PROGRAMA EXEMPLO DE USO DA CARGA
C                    01''' ( CHAVES NUMERICAS)


       DEFINE FILE 12(1,80,U,I1),20(100,31,U,J1),32(15,80,U,K1)
       INTEGER DOCUM(10),ERRO
       INTEGER CODGO,SNOME(5),NOME(7),IDADE,SEXO,PROF,END(15)
       INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
      *          NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
      *           TMREG,TOPO,ULTMA
       INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
      *          PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
       COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
      *          NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
      *           TMCHV,TMPAG,TMREG,TOPO,ULTMA
       COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
      *          REGIS,TABCP,TMCMP
       NOARC = 12
       READ(2,11)NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,TMREG,NOCMP,
      *           TABCP,TMCMP,CHV,ALFA,DOCUM
   11  FORMAT(9I3,20I2/20I2,2I3,10A2)

C      DEFINE REGISTRO INICIAL DO ARQUIVO
       I = 160/TMREG + 2
       J1 = I
       I1 = 1
  100  READ(2,5) CODGO,SNOME, NOME,IDADE,SEXO,PROF,END
    5    FORMAT(I6,5A2,7A2,I3,I1,I2,15A2)
       IF(CODGO) 1,2,2
    2  NOREG = NOREG+1
       IF(NOREG+I-MAXRG) 3,3,4
    3  WRITE(NOARR'J1) CODGO,SNOME,NOME,IDADE,SEXO,PROF,END
       GO TO 100

    1  WRITE(NOARC'1) NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,TMREG,
      *           NOCMP,TABCP,TMCMP,CHV,ALFA,DOCUM
       ERRO = 0
          CALL CARGA(0,ERRO)

C      VERIFICANDO A CARGA NO NIVEL DAS FOLHAS
C
       WRITE(3,20)
   20  FORMAT ('1',T5,'NIVEL DAS FOLHAS',/,T5,
      *    'AS CHAVES SAO NUMERICAS CONVERTIDAS EM ALFA')
       DO 200 L = PRIM, ULTMA
       READ(NOARI'L) REGIS
       WRITE(3,30) (REGIS(I),I=1,5)
   30  FORMAT(/,81('*'),/,T5,'NOPAG =',I5,'  NOCHV =',I5,'  FTOCP =',I5,
```

```
PAGE   2
     *' MARCA DE PAG. =',I3,' PROX =',I4,/)

      K = REGIS(3) + 4
      WRITE(3,70) (REGIS(I),I=6,K)
  200 CONTINUE

C     NIVEL DA RAIZ

      WRITE(3,50)
   50 FORMAT ('1',T5,'NIVEL DA RAIZ',//)

      READ(NOARI'RAIZ) REGIS
      WRITE(3,60) (REGIS(I),I=1,5)
   60 FORMAT(/,81('*'),/,T5,'NOPAG ='',I5,'  NOCHV ='',I5,'  FTOCP ='',I5,
     *' MARCA DE PAG. ='',I2,'  PO ='',I3,/)
      K = REGIS(3) + 4
      WRITE(3,70) (REGIS(I),I=6,K)
   70 FORMAT(20(1X,A2))
      CALL EXIT

    4 WRITE(3,10)
   10 FORMAT(1X,'LOTADA A AREA DISPONIVEL PARA REGISTROS')
      CALL EXIT
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON    1130  VARIABLES     72  PROGRAM     534

END OF COMPILATION

// XEQ          1

*FILES(12,CAB1),(20,REGS1),(32,INDC1)
```

```
   NIVEL DAS FOLHAS
   AS CHAVES SAO NUMERICAS CONVERTIDAS EM ALFA

*************************************************************************************
   NOPAG =    3  NOCHV =   17  FTOCP =   52 MARCA DE PAG. =  1 PROX =    4

10 0     12 0      13 0      14 0      15 0      16 0      17 0
   18 0     20 0      20 5      21 0      22 0      23 0      24
0     25 0      26 0      27 0

*************************************************************************************
   NOPAG =    4  NOCHV =   17  FTOCP =   52 MARCA DE PAG. =  1 PROX =    5

28 0     29 0      30 0      31 0      32 0      33 0      34 0
   35 0     36 7      37 0      38 0      39 0      40 0      42
0     43 0      45 0      48 0

*************************************************************************************
   NOPAG =    5  NOCHV =   17  FTOCP =   52 MARCA DE PAG. =  1 PROX =    6

49 0     50 0      51 0      52 0      53 0      55 0      60 0
   61 0     62 0      63 0      64 0      65 0      66 0      67
0     68 0      69 0      70 0

*************************************************************************************
   NOPAG =    6  NOCHV =    7  FTOCP =   22 MARCA DE PAG. =  1 PROX =    0

71 5     72 3      75 0      76 0      78 0      80 0      81 0
```

NIVEL DA RAIZ

****

NOPAG = 7 NOCHV = 3 FTOCP = 7 MARCA DE PAG. = 0 PO = 3.

28 49 71

```
PAGE  1
// JOB   0005
LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0005        0005        0000
V2 M12   ACTUAL 16K  CONFIG 16K
// FOR
*LIST SOURCE PROGRAM
*IOCS(CARD,1132 PRINTER,DISK)
*ONE WORD INTEGERS


C      PROGRAMA EXEMPLO 02 - USO DAS BUSCAS

       DEFINE FILE 12(1,80,U,I1),20(100,31,U,J1),32(15,80,U,K1)
       INTEGER APREG,ERRO
       INTEGER CHAV1(8),CHAV2(8),DOCUM(10)
       INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
      *          NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
      *          TMREG,TOPO,ULTMA
       INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
      *          PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
       COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
      *          NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
      *          TMCHV,TMPAG,TMREG,TOPO,ULTMA
       COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
      *          REGIS,TABCP,TMCMP
       DATA CHAV1/'JE','SU','S',5*' '/,CHAV2/'CA','RM','IN','O ',4*' '/
       NOARI = 32
       READ(NOARI'1) NOCHI,NOREG,DISPR,DISPI,RAIZ,PRIM,ULTMA,NOIND,
      *    NOARI,NOARR,MAXPG,TMCHV,TMPAG,TMREG,NOCMP,TABCP,TMCMP,CHV,
      *      ALFA,DOCUM

       WRITE(3,1)
    1  FORMAT('1',///,T10,'EXEMPLOS DAS BUSCAS')
       DO 30 J = 1,5
          GO TO (11,13,15,17,19),J
   11  CALL BUSCA(CHAV1,APREG,ERRO)
       WRITE(3,12)
   12  FORMAT(///,T5,'USO DA BUSCA')
       GO TO 29
   13  CALL ANTER(APREG,ERRO)
       WRITE(3,14)
   14  FORMAT(///,T5,'USO DA ANTER')
       GO TO 29
   15  CALL BUSCA(CHAV2,APREG,ERRO)
       WRITE(3,12)
       GO TO 29
   17  CALL PROX(APREG,ERRO)
       WRITE(3,18)
   18  FORMAT(///,T5,'USO DA PROX')
       GO TO 29
   19  LUGAR = 5
       CALL POSIC(LUGAR,APREG,ERRO)
       WRITE(3,22)
   22  FORMAT(///,T5,'USO DA POSIC')
   29  IF(ERRO) 6,5,6
```

```
PAGE    2

     5 READ(NOARR'APREG) (REGIS(I),I=1,TMREG)
       WRITE(3,10) (REGIS(I),I=1,TMREG)
    10 FORMAT(/,T5,'REGISTRO PROCURADO',/,T10,I6,2X,5A2,2X,7A2,2X,I3,
      *                                    2X,I1,2X,I2,2X,15A2)
     6 WRITE(3,20) ERRO
    20 FORMAT(/,T5,'*** HOUVE ERRO =',I2,' NA BUSCA')
    30 CONTINUE
       CALL EXIT
       END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON   1130  VARIABLES     56  PROGRAM     332

END OF COMPILATION

// XEQ          1

*FILES(12,CAB1),(20,REGS1),(32,INDC1)
```

EXEMPLOS DAS BUSCAS

USO DA BUSCA

REGISTRO PROCURADO
450 JESUS DANIEL 37 0 2 R BELO HORIZONTE 296

*** HOUVE ERRO = 0 NA BUSCA

USO DA ANTER

REGISTRO PROCURADO
430 ITO TIKAO 42 0 5 R MAL DEODORO 138

*** HOUVE ERRO = 0 NA BUSCA

USO DA BUSCA

REGISTRO PROCURADO
240 CARMINO VITORIO 80 0 6 AV GETULIO VARGAS 266

*** HOUVE ERRO = 0 NA BUSCA

USO DA PROX

REGISTRO PROCURADO
250 CARMO JOSE S. 36 0 5 AV LAGUNA 1041

*** HOUVE ERRO = 0 NA BUSCA

USO DA POSIC

REGISTRO PROCURADO
150 ARDUIN HERMINIO 71 0 6 R MARIO CHAMPAGNAT 1004

*** HOUVE ERRO = 0 NA BUSCA

```
PAGE  1
// JOB T
LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0005         0005         0000
V2 M12  ACTUAL 16K  CONFIG 16K
// FOR
*LIST SOURCE PROGRAM
*IOCS(CARD,1132 PRINTER,DISK)
*ONE WORD INTEGERS
C      PROGRAMA EXEMPLO 03 - USO DA REMOCAO
C         OBS:  A BUSCA SERA' USADA SO' PARA CONHECER A PAGINA
C                       PARA A IMPRESSAO DA SITUACAO ANTERIOR A REMOCAO.
      DEFINE FILE 12(1,80,U,I1),20(100,31,U,J1),32(15,80,U,K1)
      INTEGER ERRO,PGIN
      INTEGER CHAV1(8),DOCUM(10)
      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *          NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *           TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *          PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON  ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *          NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *           TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON  BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *         REGIS,TABCP,TMCMP
      NOARI = 32
      READ(NOARI'1) NOCHI,NOREG,DISPR,DISPI,RAIZ,PRIM,ULTMA,NOIND,
     *   NOARI,NOARR,MAXPG,TMCHV,TMPAG,TMREG,NOCMP,TABCP,TMCMP,CHV,
     *    ALFA,DOCUM
      DO 60 J = 1,7
      READ(2,2)CHAV1
    2 FORMAT(8A2)
      ERRO = 0
      CALL BUSCA(CHAV1,APREG,ERRO)
      IF(ERRO)50,10,50
   10 PAGIN = PLPAG(TOPO)
      PGIN = PAGIN
      READ(NOARI'PAGIN)REGIS
      IF( (J-1) - (J-1)/4*4) 12,11,12
   11 WRITE(3,1)
    1 FORMAT('1')
   12 WRITE(3,13)  CHAV1
   13 FORMAT  (/,' ANTES  DA REMOCAO DA CHAVE = ',8A2)
      WRITE(3,3)(REGIS(I),I=1,5)
    3 FORMAT(81('*'),/,T5,'NOPAG=',I5,' NOCHV=',I5,' FTOCP=',I5,
     *' MARCA DE PAG.=' ,I2,' PROX.=',I5,/)
      K=REGIS(3) + 4
      WRITE(3,4)(REGIS(I),I=6,K)
    4 FORMAT(5(2X,5A2,I4))
      ERRO = 0
      CALL REMOV(CHAV1,ERRO)
      IF(ERRO)50,20,50
   20 READ(NOARI'PGIN) REGIS
```

```
PAGE    2
      WRITE(3,23)
   23 FORMAT(//,' DEPOIS DA REMOCAO')
      WRITE(3,3)(REGIS(I),I=1,5)
      K= REGIS(3) + 4
      WRITE(3,4)(REGIS(I),I=6,K)
      GO TO 60
   50 WRITE(3,51) ERRO
   51 FORMAT(/,T5,'**** HOUVE ERRO =',I2,' NA BUSCA'
   60 CONTINUE
      CALL EXIT
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON    1130  VARIABLES      54  PROGRAM     414

END OF COMPILATION

// XEQ          1

*FILES(12,CAB1),(20,REGS1),(32,INDC1)
```

```
ANTES  DA REMOCAO DA CHAVE = ABDAS
**********************************************************************************
   NOPAG=     5 NOCHV=    10 FTOCP=    61 MARCA DE PAG.= 1 PROX=     3

                7  ABDAS         8  ABE           9  ABECHE       10  ABUD          11
 ARDUIN        12  BARBIERI      13  BOMURA       14  BONA         15  BORGHI        17

DEPOIS DA REMOCAO
**********************************************************************************
   NOPAG=     5 NOCHV=     9 FTOCP=    55 MARCA DE PAG.= 1 PROX=     3

                7  ABE           9  ABECHE       10  ABUD         11  ARDUIN        12
 BARBIERI      13  BOMURA       14  BONA         15  BORGHI       17

ANTES  DA REMOCAO DA CHAVE = YAMAO
**********************************************************************************
   NOPAG=    13 NOCHV=     8 FTOCP=    49 MARCA DE PAG.= 1 PROX=     0

 XANDER        86  XAVIER       87  YABIKU       88  YAMAO        89  YAMAOKA       90
 ZAGATTI       91  ZAGO         92  ZIOBER       93

DEPOIS DA REMOCAO
**********************************************************************************
   NOPAG=    13 NOCHV=     7 FTOCP=    43 MARCA DE PAG.= 1 PROX=     0

 XANDER        86  XAVIER       87  YABIKU       88  YAMAOKA      90  ZAGATTI       91
 ZAGO          92  ZIOBER       93

ANTES  DA REMOCAO DA CHAVE = VIDOTTI
**********************************************************************************
   NOPAG=    12 NOCHV=     9 FTOCP=    55 MARCA DE PAG.= 1 PROX=    13

 TOZZI         77  TRALDI       78  UYEDA        79  VIDOTTI      80  VINHAS        81
 WADA          82  WECO         83  WELZ         84  WOLFF        85

DEPOIS DA REMOCAO
**********************************************************************************
   NOPAG=    12 NOCHV=     8 FTOCP=    49 MARCA DE PAG.= 1 PROX=    13

 TOZZI         77  TRALDI       78  UYEDA        79  VINHAS       81  WADA          82
 WECO          83  WELZ         84  WOLFF        85

ANTES  DA REMOCAO DA CHAVE = WOLFF
**********************************************************************************
   NOPAG=    12 NOCHV=     8 FTOCP=    49 MARCA DE PAG.= 1 PROX=    13

 TOZZI         77  TRALDI       78  UYEDA        79  VINHAS       81  WADA          82
 WECO          83  WELZ         84  WOLFF        85

DEPOIS DA REMOCAO
**********************************************************************************
   NOPAG=    12 NOCHV=     7 FTOCP=    43 MARCA DE PAG.= 1 PROX=    13

 TOZZI         77  TRALDI       78  UYEDA        79  VINHAS       81  WADA          82
 WECO          83  WELZ         84
```

```
ANTES  DA REMOCAO DA CHAVE = WADA
**********************************************************************
   NOPAG=   12 NOCHV=    7 FTOCP=   43 MARCA DE PAG.= 1 PROX=   13

 TOZZI       77  TRALDI      78  UYEDA       79  VINHAS      81  WADA        82
 WECO        83  WELZ        84

DEPOIS DA REMOCAO
**********************************************************************
   NOPAG=   12 NOCHV=    8 FTOCP=   49 MARCA DE PAG.= 1 PROX=   13

 TANAKA      75  TAVARES     76  TOZZI       77  TRALDI      78  UYEDA       79
 VINHAS      81  WECO        83  WELZ        84

ANTES  DA REMOCAO DA CHAVE = WECO
**********************************************************************
   NOPAG=   12 NOCHV=    8 FTOCP=   49 MARCA DE PAG.= 1 PROX=   13

 TANAKA      75  TAVARES     76  TOZZI       77  TRALDI      78  UYEDA       79
 VINHAS      81  WECO        83  WELZ        84

DEPOIS DA REMOCAO
**********************************************************************
   NOPAG=   12 NOCHV=    7 FTOCP=   43 MARCA DE PAG.= 1 PROX=   13

 TANAKA      75  TAVARES     76  TOZZI       77  TRALDI      78  UYEDA       79
 VINHAS      81  WELZ        84

ANTES  DA REMOCAO DA CHAVE = WELZ
**********************************************************************
   NOPAG=   12 NOCHV=    7 FTOCP=   43 MARCA DE PAG.= 1 PROX=   13

 TANAKA      75  TAVARES     76  TOZZI       77  TRALDI      78  UYEDA       79
 VINHAS      81  WELZ        84

DEPOIS DA REMOCAO
**********************************************************************
   NOPAG=   12 NOCHV=    6 FTOCP=   37 MARCA DE PAG.= 1 PROX=   13

 TANAKA      75  TAVARES     76  TOZZI       77  TRALDI      78  UYEDA       79
 VINHAS      81
```

```
PAGE   1
// JOB    0005
LOG DRIVE   CART SPEC    CART AVAIL   PHY DRIVE
  0000         0005         0005         0000
V2 M12   ACTUAL 16K   CONFIG 16K
// FOR
*LIST SOURCE PROGRAM
*IOCS(CARD,1132 PRINTER,DISK)
*ONE WORD INTEGERS

C     PROGRAMA EXEMPLO 04 - USO DA INSERCAO UM A UM
C                           CHAVES ALFAS DE TAMANHO CONSTANTE )

      DEFINE FILE 12(1,80,U,I1),20(100,31,U,J1),32(15,80,U,K1)
      INTEGER APREG,ERRO,INDC1
      INTEGER CHAVE(8),DOCUM(10)

      INTEGER ALFA,CHV,DISPI,DISPR,MEIO,NOARC,NOARI,NOARR,NOCHI,
     *        NOIND,NOREG,PAGIN,PRIM,RAIZ,TMCHV,TMPAG,
     *        TMREG,TOPO,ULTMA
      INTEGER BUFF(3,315),FTOCP(3),M(3),NOCHV(3),NOPAG(3),PLAPT(10),
     *        PLPAG(10),PLPDB(10),REGIS(80),TABCP(20),TMCMP(20)
      COMMON ALFA,CHV,DISPI,DISPR,I1,J1,K1,MEIO,NOARC,NOARI,
     *        NOARR,NOCHI,NOIND,NOREG,PAGIN,PRIM,RAIZ,
     *        TMCHV,TMPAG,TMREG,TOPO,ULTMA
      COMMON BUFF,FTOCP,M,NOCHV,NOPAG,PLAPT,PLPAG,PLPDB,
     *       REGIS,TABCP,TMCMP

      DATA CHAVE/8*' '/
      READ(2,11)NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,TMREG,NOCMP,
     *          TABCP,TMCMP,CHV,ALFA,DOCUM
   11 FORMAT(9I3,20I2/20I2,2I3,10A2)
      NOARC =12
      WRITE(NOARC'1) NOARI,NOARR,MAXRG,MAXPG,NOREG,TMCHV,TMPAG,TMREG,
     *          NOCMP,TABCP,TMCMP,CHV,ALFA,DOCUM
      ERRO = 0
      CALL CARGA(1,ERRO)
  100 READ(2,5) (REGIS(I),I=1,31)
    5    FORMAT(I6,5A2,7A2,I3,I1,I2,15A2)
      DO 10 I=1,5
   10    CHAVE(I) = REGIS(I+1)
      IF(REGIS(1)) 1,2,2
    2 INDC1 = 1
      ERRO = 0
      CALL INSER(CHAVE,APREG,INDC1,ERRO)
      GO TO 100

C     VERIFICANDO A SITUACAO NO NIVEL DAS FOLHAS
    1 WRITE(3,20)
   20 FORMAT ('1',//,T5,'NIVEL DAS FOLHAS',//,T5,
     *   'AS CHAVES SAO ALFAS DE TAMANHO CONSTANTE = 5')
      L = PRIM
   25 READ(NOARI'L) REGIS
      WRITE(3,30) (REGIS(I),I=1,5)
   30 FORMAT(/,81('*'),/,T5,'NOPAG =',I5,'  NOCHV =',I5,'  FTOCP =',I5,
     *  ' MARCA DE PAG. =',I3,' PROX =',I5,/)
```

```
PAGE    2

C     AS CHAVES DO EXEMPLO TEM TAMANHO IGUAL 5
      K = REGIS(3) + 4
      WRITE(3,40) (REGIS(I),I=6,K)
   40 FORMAT(5(2X,5A2,I4))
      L = REGIS(5)
      IF(L) 45,45,25
   45 CONTINUE

C     NIVEL DA RAIZ

      WRITE(3,50)
   50 FORMAT('1',//,T5,'NIVEL DA RAIZ',//)

      READ(NOARI'RAIZ) REGIS
      WRITE(3,60) (REGIS(I),I=1,5)
   60 FORMAT(/,81('*'),/,T5,'NOPAG =',I5,'  NOCHV =',I5,'  FTOCP =',I5,
     *' MARCA DE PAG. =',I3,' PO =',I4,/)
      K = REGIS(3) + 4
      WRITE(3,70) (REGIS(I),I=6,K)
   70 FORMAT(20(1X,A2))
      CALL EXIT
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON    1130  VARIABLES      52  PROGRAM     512

END OF COMPILATION

// XFQ          1

*FILES(12,CAB1),(20,REGS1),(32,INDC1)
```

NIVEL DAS FOLHAS

AS CHAVES SAO ALFAS DE TAMANHO CONSTANTE = 5

****

NOPAG = 5 NOCHV = 9 FTOCP = .55 MARCA DE PAG. = 1 PROX = 3

ABDAS	7	ABE	8	ABECHE	9	ABUD	10	ARDUIN	11
BARBIERI	12	BOMURA	13	BONA	14	BORGHI	15		

****

NOPAG = 3 NOCHV = 9 FTOCP = 55 MARCA DE PAG. = 1 PROX = 7

BORIN	16	BOTELHO	17	CAOBIANCO	18	CAPASSO	19	CARMINO	20
CARRO	21	CAVALIERI	22	CAYRES	23	CORDEIRO	24		

****

NOPAG = 7 NOCHV = 10 FTOCP = 61 MARCA DE PAG. = 1 PROX = 6

CORTES	25	CORTEZ	26	COSTA	27	CRUZ	28	CUTULO	29
DERNER	30	DUARTE	31	FERRAS	32	FERRAZ	33	FERREIRA	34

****

NOPAG = 6 NOCHV = 10 FTOCP = 61 MARCA DE PAG. = 1 PROX = 8

FUMERO	35	GROSSI	36	IMAY	37	ITO	38	JESUS	39
JOSEPETTI	40	KIDO	41	KOGA	42	KUTNE	43	LEITE	44

****

NOPAG = 8 NOCHV = 10 FTOCP = 61 MARCA DE PAG. = 1 PROX = 9

LIBARDI	45	LIMA	46	LIPPI	47	LOMBARDI	48	MAIMONE	50
MALUF	49	MORAES	51	MORAIS	52	MORELLI	53	NAUFAL	54

****

NOPAG = 9 NOCHV = 10 FTOCP = 61 MARCA DE PAG. = 1 PROX = 10

NORONHA	55	ORFAO	56	OSSUCHI	57	PACE	58	PACHECO	59
PALLONE	60	PIGOZZO	61	PIOLLI	62	PRATIS	63	RABITO	64

****

NOPAG = 10 NOCHV = 10 FTOCP = 61 MARCA DE PAG. = 1 PROX = 11

RIBEIRO	65	ROCHA	66	RUI	67	SANCHES	68	SANTINI	69
SANTINOLI	70	SANTOS	71	SHIRAISHI	72	SOARES	73	TANAKA	74

****

NOPAG = 11 NOCHV = 8 FTOCP = 49 MARCA DE PAG. = 1 PROX = 12

TAVARES	75	TOZZI	76	TRALDI	77	UYEDA	78	VIDOTTI	79
VINHAS	80	WADA	81	WECO	82				

****

NOPAG = 12 NOCHV = 10 FTOCP = 61 MARCA DE PAG. = 1 PROX = 0

WELZ	83	WOLFF	84	XANDER	85	XAVIER	86	YABIKU	87
YAMAO	88	YAMAOKA	89	ZAGATTI	90	ZAGO	91	ZIOBER	92

NIVEL DA RAIZ

```
**************************************************************************************
   NOPAG =     4  NOCHV =     8  FTOCP =    21 MARCA DE PAG. =  0 PO =   5

BO RI    CO RT    FU   LI   NO    RI    TA VA    WE LZ
```