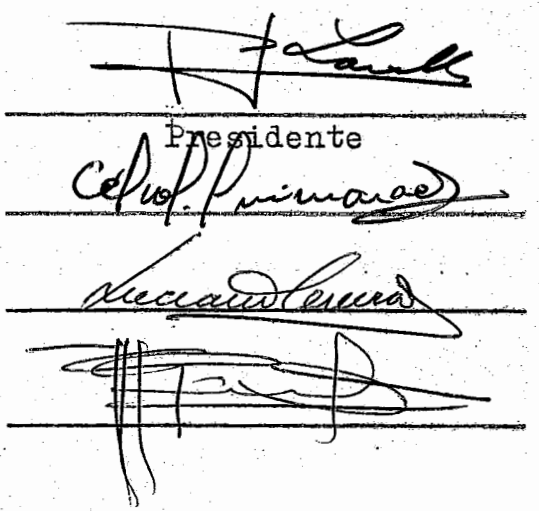
C O M P R E S S Ã O D E D A D O S

Luciano Pietracci

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

Aprovada por:



RIO DE JANEIRO

ESTADO DA GUANABARA - BRASIL

OUTUBRO DE 1973

À

SOLANGE

AGRADECEMOS:

Ao Professor PIERRE JEAN LAVELLE
pela orientação proporcionada na elaboração deste trabalho.

Aos Professôres da COPPE
sem a ajuda dos quais a realização deste trabalho não teria sido possível.

Aos Amigos ANTONIO SERGIO SECCO FERREIRA
ARY BERNARDO HANDLER
CARLOS FLORES CUNHA
GARRONE VIOT PINHEIRO
e aos outros que ajudaram com suas idéias no desenvolvimento do presente trabalho.

Luciano Pietracci

RESUMO

R E S U M O

A Compressão de Dados é de grande interêsse no tratamento da informação pois oferece meios para reduzir os custos de processamento e o potencial para aumentar a capacidade dos dispositivos de armazenamento, dos canais e das linhas de comunicação.

Este trabalho apresenta algumas técnicas de compressão e como estas se aplicam no tratamento da informação.

Descrevemos a implantação de algumas destas técnicas e fazemos considerações no que se refere aos rendimentos obtidos e aos tempos de execução de cada método.

A conclusão do trabalho apresenta um estudo para a escolha da melhor técnica em função da aplicação desejada e uma comparação dos métodos.

A B S T R A C T

Data compression is of great interest in information processing as it furnish means to reduce processing costs and offers the potential to enlarge the capacity of mass storage devices, channels and communication lines.

This work presents some compression techniques and how they are applied in information processing.

It describes the implementation of some compression techniques and makes considerations about the attained profits and execution time of each method.

The conclusion of this work presents a study for the choice of the best technique in connection to the desired aplication and a comparison of the methods.

ÍNDICE

INTRODUÇÃO

1. IMPORTÂNCIA DA COMPRESSÃO DE DADOS

O aperfeiçoamento das características dos computadores (aumento quase ilimitado de memórias, acesso direto e randômico, elevadíssimas velocidades de processamento, interconecção de computadores a grande distância, possibilidade de colóquio homem-máquina com pontos de centralização de dados) levou à criação de sistemas de informações sempre mais sofisticados.

Com o consequente desenvolvimento dos sistemas de elaboração de dados, o tratamento das informações está evoluindo sempre mais ao encontro de uma automação total.

Podemos prever que qualquer funcionário de uma empresa poderá, com relação ao seu setor de responsabilidade, pedir ao sistema, previamente elaborado, informações dos arquivos, tabelas, normas etc. ..., inserir dados da sua atividade retirando respostas imediatas consequentes de suas operações, sugestões etc. ...

O conceito de intercomunicação homem-máquina tende a se extender a computadores de empresas diversas (em base mundial) para todos os intercâmbios de informações e relativos processamentos.

De quanto foi sinteticamente exposto vem a necessidade e a importância da "compressão de dados", não somente no que se refere à transmissão, como também, no que tange ao armazenamento dos mesmos.

De fato a compressão de dados permite não somente otimizar o uso das linhas de comunicação, permitindo transmitir um

{mesmo / maior} conjunto de informações numa {menor / mesma} quantidade de tempo, como também, reduzir consideravelmente o volume dos arquivos, possibilitando a coexistência de um {mesmo / maior} número de informações num {menor / mesmo} suporte de armazenamento.

Como sabemos, o teleprocessamento e a transmissão de dados exigem a manutenção de grandes arquivos em dispositivos de acesso direto, e de onerosas linhas de comunicação.

Na prática, frequentemente, somos limitados pela insuficiência de espaço em arquivos com acesso direto ou pelo alto custo das linhas de transmissão.

A compressão de dados é uma solução para todo esse condicionamento que envolve o planejamento da distribuição dos arquivos e da utilização e escolha das linhas.

O uso de arquivos compactados permite, também, reduzir o tempo de execução dos programas quando não é necessário descomprimir cada registro (uma consulta ou uma atualização, por exemplo).

Na transmissão de dados a compressão visa abaixar os custos de transmissão diminuindo o tempo de uso das linhas de comunicação que podem ser aproveitadas por outras funções ou planejadas para uma menor velocidade de transmissão.

2. VANTAGENS OFERECIDAS PELA COMPRESSÃO DE DADOS

2.1. NO RENDIMENTO

* MAIOR RENDIMENTO PARA "JOB" QUE ATUALIZAM ARQUIVOS

Estes tipos de "JOBS" são caracterizados por um arquivo cadastro e outro de transações a serem processadas. O arquivo de transações é normalmente muito menor do arquivo cadastro. A finalidade do "JOB" é comparar o arquivo de transações com o cadastro para retirar, inserir ou modificar os registros neste gravados.

Ora, se estes arquivos são sequenciais, o "JOB", consiste basicamente na leitura e gravação dos registros do arquivo cadastro até que seja encontrado, um registro que corresponda a uma transação. Se este arquivo cadastro for compactado, então estas funções de ler e gravar, serão muito rápidas pois o número de bytes a ser processado nas duas operações será menor.

Quando for encontrado um registro para uma transação, este será descompresso e após completada a transação, compresso e gravado.

Para todos os "JOBS" que apresentam estas características podemos obter reduções significativas no tempo de execução.

* MAIOR EFICIÊNCIA PARA A CLASSIFICAÇÃO

Há aqui dois aspectos a serem considerados:

Primeiro: quando se classifica um arquivo em disco,

o número de registros que podemos processar é limitado pela capacidade do disco, se o tamanho dos registros for menor podemos reduzir o volume utilizado, possibilitando até, que o espaço economizado possa ser usado por outras aplicações.

Segundo: quando usamos arquivos compressos a quantidade de dados a ser lida ou gravado é menor permitindo aumentar a velocidade da classificação.

* MAIOR EFICIÊNCIA PARA APLICAÇÕES ON-LINE

A compressão de dados tem grande significado em aplicações "on-line", especialmente quando os arquivos são de acesso randômico, pois, o tempo da compressão e/ou descompressão é desprezível comparado ao tempo de acesso ao registro.

2.2. NA PROGRAMAÇÃO

* TRANSPARÊNCIA DA COMPRESSÃO E DA DESCOMPRESSÃO

O programador trabalha com registros de tamanho fixo não se preocupando como estes são armazenados. A descompressão é feita logo após a leitura e a compressão antes da gravação ficando transparente para o programador o fato que o registro no suporte de armazenamento, é variável.

* SIMPLIFICAÇÃO DO DESENHO DOS REGISTROS E DOS ARQUIVOS

Normalmente, torna-se necessário desenhar arquivos com complexos registros variáveis. Isto dificulta não somente, o desenho dos arquivos como a programação.

A compressão de dados é uma solução para as duas coisas bastando para isto acrescentar algumas linhas de codificação no programa.

2.3. NOS CUSTOS

* REDUÇÃO DO TEMPO DE PROCESSAMENTO

A redução de tempo proporcionada pela compressão de dados se reflete diretamente numa redução de custos se pensarmos que a maioria dos computadores são alugados e pagos por hora de processamento. Além disso, podemos pensar em um aumento do rendimento do computador, pois sobra tempo para o processamento de outros "JOBS".

* REDUÇÃO DO CUSTO DAS LINHAS

O custo das linhas de transmissão, em certos casos, supera o custo do equipamento. Quando isto ocorre, trânsmitir dados na forma compactada pode ser de grande significado.

* REDUÇÃO DO CUSTO DA MEMÓRIA AUXILIAR

Se considerarmos um sistema "ON-LINE" e o grande volume de informações que devem ser armazenados, podemos verificar como um método de compressão pode reduzir o

volume e o custo da memória auxiliar utilizada.

Muitos centros de processamento de dados, especialmente os que controlam os históricos de uma empresa, aqueles militares ou governamentais, tem a necessidade de reservar grande número de volumes, às vezes, por períodos que duram anos.

A compressão de dados pode reduzir o número dos suportes de armazenamento que são necessários na configuração normal de um sistema.

2.4. NA SEGURANÇA DOS ARQUIVOS

* UM ARQUIVO COMPRESSO NÃO PODE SER LIDO POR ESTRANHOS

Normalmente, quando é efetuada a compressão, o arquivo torna-se muito diferente do que era na sua forma original. Para que estranhos possam ler o arquivo compresso, estes, devem conhecer as técnicas utilizadas, isto permite que as informações não sejam indevidamente divulgadas.

3. OBJETIVOS DO TRABALHO

Pelos motivos que citamos a compressão de dados é de grande interesse no processamento de dados diminuindo os custos e oferecendo o potencial para aumentar a capacidade de armazenamento, a velocidade das linhas de comunicação e dos canais.

Estas vantagens, evidentemente, têm um preço, que consiste num acréscimo no tempo de uso da CPU, porém, normalmente este tempo é desprezível, poucas sendo as Unidades Centrais de

de Processamento aproveitadas ao 100%.

Nos Centros de Processamento de Dados, geralmente, os pontos de estrangulamento são os canais, que permitem o acesso aos periféricos (discos e/ou fitas e/ou leitoras etc. ...). A compressão de dados oferece a possibilidade de ler e escrever maior número de registros, na mesma unidade de tempo, portanto de melhorar o aproveitamento da CPU, ainda sobra o tempo necessário à compressão e descompressão. As técnicas de compressão escolhidas não devem, entretanto fazer com que este acréscimo de rendimento exceda a capacidade da CPU que, desta maneira, se tornaria o ponto de estrangulamento do Sistema, o que seria altamente prejudicial.

Essas técnicas que permitem obter melhores rendimentos com menos "HARDWARE" não são divulgadas, por razões óbvias, pelos fabricantes de computadores.

Este trabalho têm por finalidade apresentar algumas técnicas de compressão, descrever como estas se aplicam no tratamento dos dados e implementar tipos de compressão que permitem reduzir consideravelmente o volume dos arquivos, o tempo de execução e os custos de transmissão.

São feitas comparações entre os métodos no que se refere a tempos de execução e quantidade de memória utilizada.

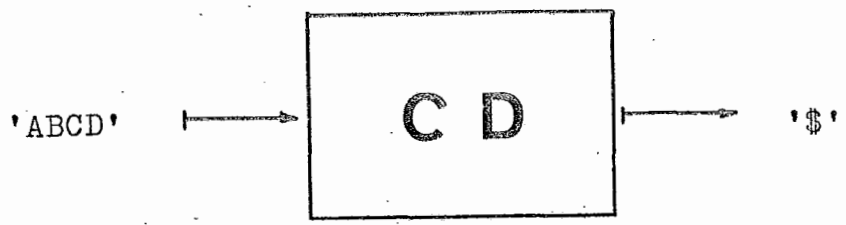
São apresentados, também, alguns critérios para a escolha do melhor método em função da aplicação desejada.

CAPÍTULO-1

1. DEFINIÇÕES:

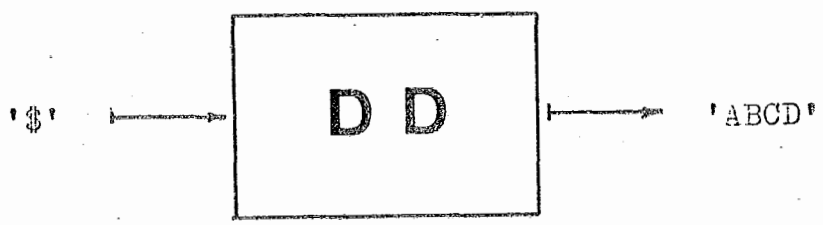
1.1. COMPRESSOR DE DADOS

É uma rotina de "software" ou um dispositivo de "hardware" que recebe como entrada uma sequência de bits ou caracteres e a transforma em uma sequência, geralmente, menor de bits ou caracteres.



1.2. DESCOMPRESSOR DE DADOS

É uma rotina de "software" ou um dispositivo de "hardware" que transforma de volta ao seu estado original uma sequência de bits ou caracteres.



1.3. COMPRESSÃO ABSOLUTA

Se o comprimento inicial de uma sequência é CI (expresso em bits ou caracteres) e se o seu comprimento final é CF (também, expresso em bits ou caracteres) diremos que:

COMPRESSÃO ABSOLUTA = COMPRIM. INICIAL - COMPRIM. FINAL

e escrevemos:

$CA = CI - CF$ onde $CA \geq 0$ (bits ou caracteres).

1.4. COMPRESSÃO RELATIVA E PERCENTUAL

Se obtivemos uma compressão absoluta de 10 caracteres ao comprimir uma sequência de 20 caracteres, esta será bem mais significativa do que uma compressão de 10 caracteres numa sequência de 100.000 caracteres, no entanto nos dois casos a compressão absoluta é de 10 caracteres.

Para termos uma idéia mais real de compressão devemos introduzir o conceito de:

* COMPRESSÃO RELATIVA E COMPRESSÃO PERCENTUAL

Utilizando as definições dadas no parágrafo 1.3. teremos:

$$\text{COMPRESSÃO RELATIVA} = \frac{\text{COMPRIM. INICIAL} - \text{COMPRIM. FINAL}}{\text{COMPRIM. INICIAL}}$$

e escrevemos:

$$CR = \frac{CI - CF}{CI} \quad \text{onde} \quad 0 \leq CR < 1 \qquad CF > 0$$

COMPRESSÃO PERCENTUAL = COMPRESSÃO RELATIVA * 100

e escrevemos:

$CP = CR * 100$ onde $0\,\% \leq CP < 100\%$

1.5. EXEMPLOS

Se ao comprimirmos uma sequência de 100 caracteres obtivermos uma nova sequência de 25 caracteres diremos que:

a) $CA = CI - CF$

$CA = 75$ caracteres

b) $CR = \frac{CI - CF}{CI}$

$CR = \frac{75}{100} = 0.75$

c) $CP = CR \times 100\%$

$CP = 75\%$

OBSERVAÇÃO:

No nosso trabalho chamaremos a Compressão Percentual simplesmente de "compressão".

CAPÍTULO-2

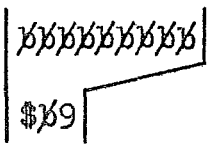
1. COMPRESSÃO DE SEQUENCIAS DE CARACTERES IDÊNTICOS.

Estes tipos de "compressores" englobam uma grande variedade de rotinas que reduzem ou eliminam sequencias de zeros, de brancos ou de outros caracteres que podem se repetir sequencialmente num determinado arquivo ou texto.

A compressão pode ser obtida de várias maneiras.

A primeira consiste em substituir as sequencias de caracteres idênticos por um conjunto de três caracteres em que:
o primeiro indica que houve compressão,
o segundo que tipo de carater foi retirado e
o terceiro o número de caracteres retirados. [7]

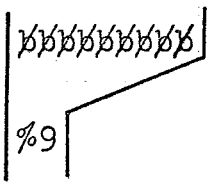
Por exemplo, uma sequencia de nove caracteres brancos ficaria assim reduzida:



onde: '$' indica que houve compressão
'b' indica que tipo de caráter foi retirado
'9' indica o número de caracteres retirados

No entanto, o fato de uma sequencia de caracteres brancos ter sido retirada pode ser expresso por um conjunto de apenas dois caracteres. Um "caráter sentinela" indicando que houve compressão de brancos e outro indicando o número de brancos retirados. [3]

A sequencia do exemplo anterior ficaria assim reduzida:



onde: '%' indica que houve compressão de brancos

'9' indica o número de brancos retirados.

O primeiro método permite comprimir qualquer tipo de sequencia de mais de três caracteres idênticos.

O segundo método é mais eficiente, pois permite uma maior compressão, mas implica na escolha, a priori, de um "caráter sentinela" para cada tipo de sequencia.

Existem ainda, casos em que podemos eliminar (ao invés de reduzir) sequencias de caracteres idênticos sem alterar o significado da informação.

Estes processos de compressão trazem grandes vantagens no processamento comercial, pelo fato que, uma das características dos arquivos comerciais é a grande frequencia de brancos e de zeros.

Exemplo:

1 - 35	nome	LUCIANObPIETRACCIbbbbbbbbbbbbbbbbbb
35 - 70	endereço	AVbNSbCOPACABANAb11Ø2b8Ø2bbbbbbbbbb
71 - 75	codigo	ØØØ35
76 - 80	quant.ordenada	ØØØ3Ø
81 - 85	quant.enviada	ØØØØØ
86 - 95	quant.paga	ØØØØ2ØØØØØ
96 - 105	quant.devida	ØØØØØØØØØØ
106 - 106	codigo de classificação	Ø

Neste tipo de registro são evidentes as vantagens que obter-se-iam aplicando uma rotina de compressão de brancos e de zeros.

A compressão por substituição de caracteres idênticos é fácil no seu conceito, assim como na sua implantação e em geral, permite uma alta compressão e um custo muito baixo em termos de tempo de CPU.

O compressor e descompressor podem ser programados como rotinas gerais que permitam elaborar os mais diversos tipos de registros.

Alguns computadores já possuem rotinas de supressão de brancos, especialmente, para a transmissão de dados da memória para os periféricos. (Para impressoras, por exemplo, são su-

primidos os brancos finais da linha).

Em alguns casos, porém, este tipo de compressão não dá resultados tão bons quanto outras técnicas de compressão que estudaremos mais adiante.

2. IMPLEMENTAÇÃO DE UMA ROTINA DE SUPRESSÃO DE BRANCOS:

2.1. GENERALIDADES

Como exemplo, desenvolvemos em nosso trabalho a implementação de uma rotina de COMPRESSÃO de sequencias de até 255 brancos e uma rotina de DESCOMPRESSÃO que permita transformar de volta ao seu estado original as sequencias compressas.

A compressão é obtida retirando as sequencias de três ou mais brancos e substituindo-as por um "caráter sentinela" seguido do número de brancos eliminados. Este "caráter sentinela" é consultado na descompressão para se poder restaurar o registro.

Exemplo:

a)	×Ø×Ø×ØØØØCOMPRESSÃOØØDEØØDADOSØØØØ×Ø×Ø×	original
b)	545454444CDDDDCEECD44CC44CCCDE444454545 CØCØCØØØØ3647952216ØØ45ØØ41462ØØØØCØCØC	representação interna
c)	54545BØCDDDDCEECD44CC44CCCDEBØ54545 CØCØCD43647952216ØØ45ØØ41462D4CØCØC	representação interna compressa

Obs.: O caráter sentinela usado é a configuração de um byte contendo |B|D|

É necessário observar que nesta rotina a configuração X'BD' não poderá aparecer no registro a ser compresso, assim como uma sequencia maior de 256 brancos. Veremos mais adiante como resolver este problema.

As rotinas de compressão "COMP" e as de descompressão "DCOMP" são escritas em ASSEMBLER IBM/360; estas não possuem instruções de entrada ou saída e são independentes do SISTEMA OPERACIONAL. Foram estudadas para poderem ser chamadas por programas escritos em PL/1.

2.2. COMO O PL/1 PASSA OS PARÂMETROS PARA UMA SUB-ROTINA

Quando em PL/1 declaramos uma área definida como "CHAR":

Exemplo:

```
DCL ENTRADA CHAR (133);
DCL SAIDA   CHAR (133);
```

É criado um "DOPE VECTOR" (D.V.) que contem as informações desta área. (Vide FIGURA-1)

Caso a área não seja variável CM = CA, isto é,
COMPRIMENTO MÁXIMO = COMPRIMENTO ATUAL.

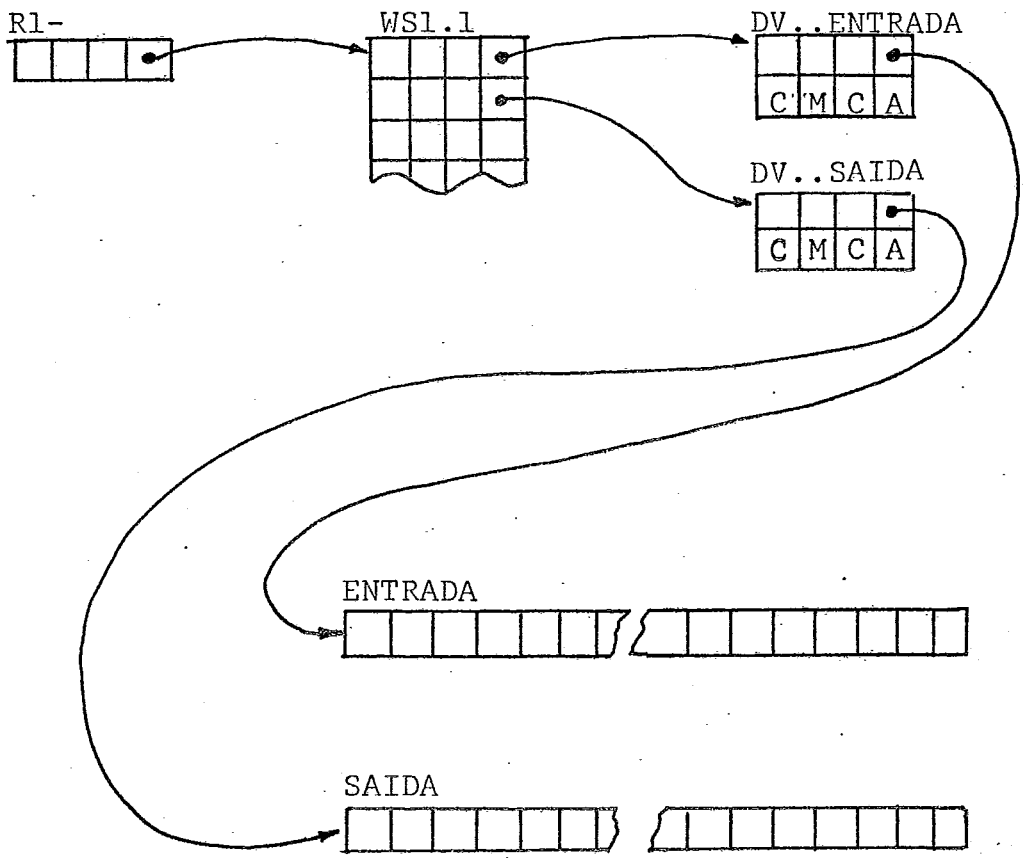
Quando é dado um "CALL" em PL/1:

Exemplo:

```
CALL COMP (ENTRADA , SAIDA) .
```

O registrador "1" aponta para uma lista de endereços, cada um dos endereços apontando para o D.V. respectivo de cada parâmetro. Após ter montado a lista de apontadores o controle é transferido para a rotina "COMP".

COMO O PL/1 PASSA OS PARAMETROS
PARA UMA SUB-ROTINA



CM: COMPRIMENTO MÁXIMO
CA: COMPRIMENTO ATUAL

EXPANSÃO DA INSTRUÇÃO: CALL COMP (ENTRADA,SAIDA);

```
LA    8,DV..ENTRADA
ST    8,WS1.1
LA    8,DV..SAIDA
ST    8,WS1.1+4
OI    WS1.1+4,X'80'
LA    1,WS1.1
L     15,A..COMP
BALR  14,15
```

FIGURA - 1

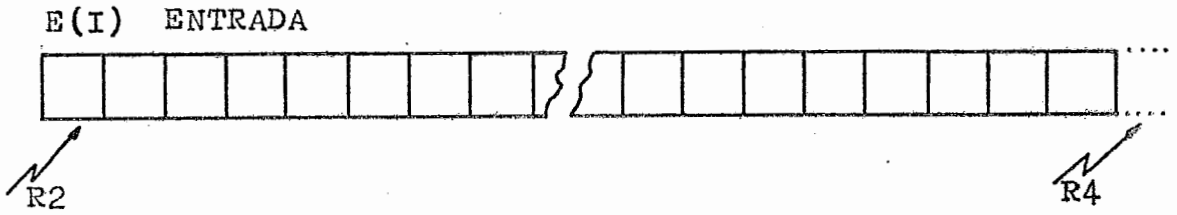
2.3. ROTINA DE COMPRESSÃO

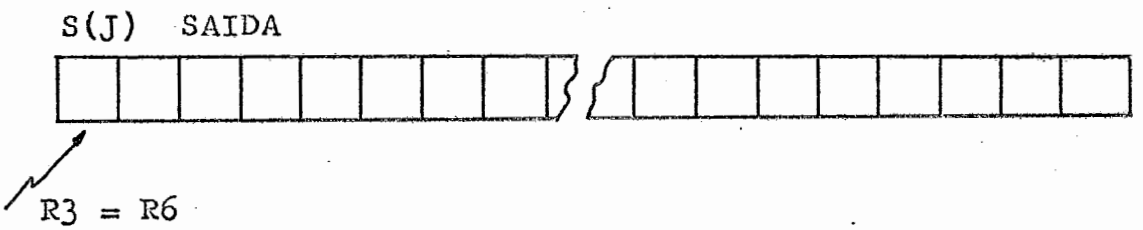
A rotina de compressão de brancos que desenvolvemos é chamada através de dois parâmetros da seguinte forma:

CALL COMP (ENTRADA,SAIDA);

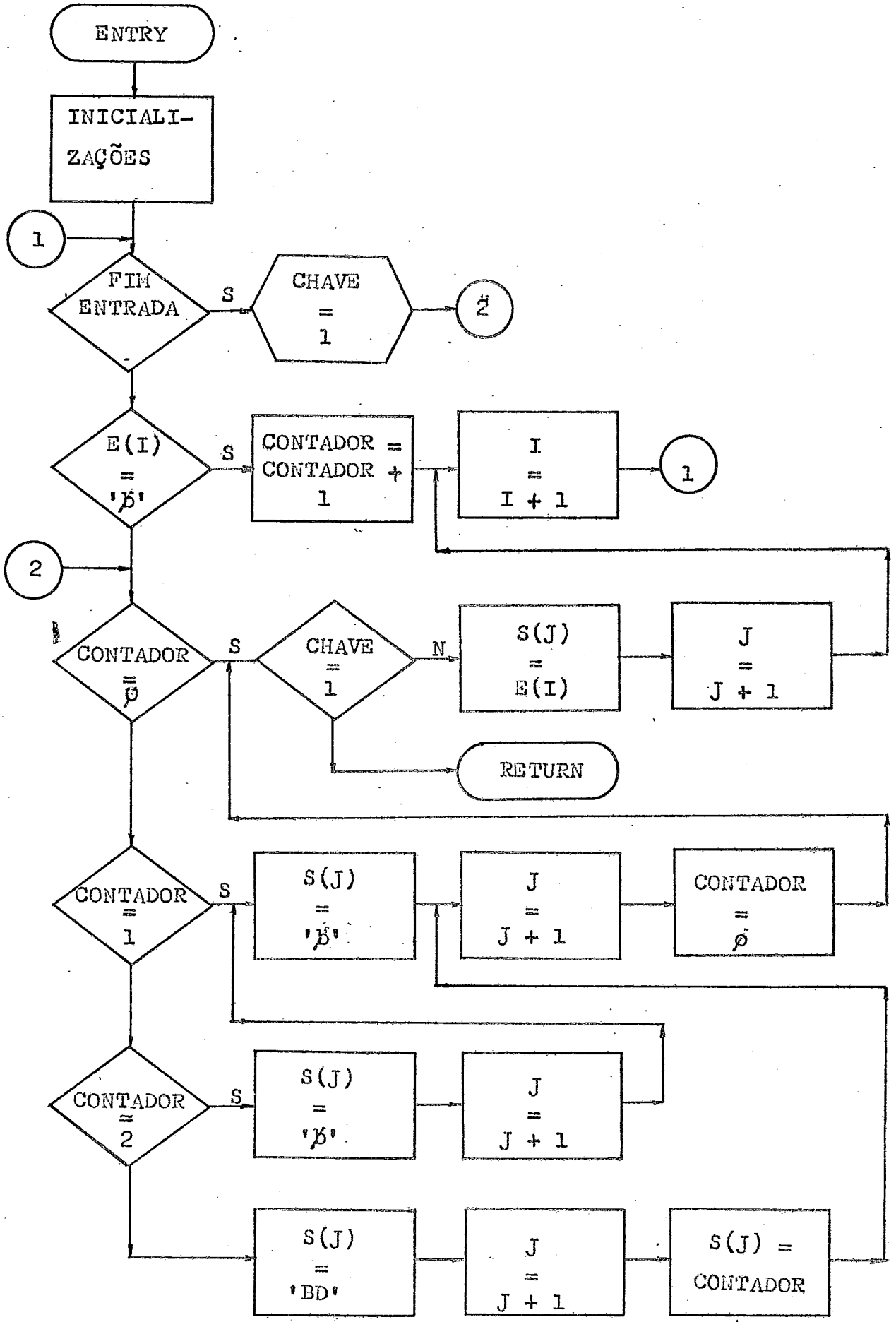
ENTRADA: É o endereço da área a ser compressa

SAIDA: É o endereço de uma área para a devolução da informação compressa.





Apresentamos a seguir o diagrama de blocos e a codificação da rotina de compressão 'COMP'


ENTRY
INICIALI-
ZAÇÕES
1
FIM
ENTRADA
S
CHAVE
=
1
2
E(I)
=
'b'
S
CONTADOR =
CONTADOR +
1
I
=
I + 1
1
2
CONTADOR
=
0
S
CHAVE
=
1
N
S(J)
=
E(I)
J
=
J + 1
RETURN
CONTADOR
=
1
S
S(J)
=
'b'
J
=
J + 1
CONTADOR
=
0
CONTADOR
=
2
S
S(J)
=
'b'
J
=
J + 1
S(J)
=
'BD'
J
=
J + 1
S(J) =
CONTADOR

```
STMT   SOURCE STATEMENT                                          F01MAY7
   1 *
   2 ********************************************************************
   3 *         IMPLEMENTACAO DA ROTINA DE COMPRESSAO DE BRANCOS         *
   4 *         LUCIANO PIETRACCI                                        *
   5 ********************************************************************
   6 *
   7           PRINT NOGEN
   8 COMP      CSECT
   9           SAVE  (14,12)
  12           LR    12,15
  13           USING COMP,12
  14           ST    13,SALVA+4
  15           LR    11,13
  16           LA    13,SALVA
  17           ST    13,8(11)
```

```
STMT   SOURCE STATEMENT                                          F01MAY

  19 *
  20 ***************************************************************
  21 *        INICIALIZACOES                                       *
  22 ***************************************************************
  23 *
  24           L      4,0(0,1)       CARREGA O END. DO DV..ENTRADA
  25           L      2,0(0,4)       CARREGA EM R2 O END. DE ENTRADA
  26           L      5,4(0,1)       CARREGA O END. DO DV..SAIDA
  27           L      3,0(0,5)       CARREGA EM R3 O END. DE SAIDA
  28           LH     4,4(0,4)       CARREGA EM R4 COMPR. DE ENTRADA
  29           AR     4,2            R4 APONTA PARA O FIM DA ENTRADA
  30           LR     6,3            R6 APONTA PARA O INICIO DA SAIDA
  31           SR     7,7            ZERA R7, CONTADOR DE BRANCOS
  32           MVI    CHAVE,C'0'     FAZ CHAVE = '0'
```

```
STMT   SOURCE STATEMENT                                          F01MAY7

  34 *
  35 ************************************************************************
  36 *         PROCESSAMENTO E COMPRESSAO                                   *
  37 ************************************************************************
  38 *
  39           B      INICIO
  40 MOVE3     MVI    0(6),C' '       MOVE BRANCO PARA SAIDA
  41           LA     6,1(0,6)        INCREMENTA O APONTADOR DA SAIDA
  42 MOVE2     MVI    0(6),C' '       MOVE BRANCO PARA SAIDA
  43           B      MOVE0
  44 MOVE4     MVC    0(1,6),0(2)     MOVE A LETRA PARA SAIDA
  45           LA     6,1(0,6)        INCREMENTA O APONTADOR DA SAIDA
  46           B      INCRE1
  47 INCRE0    LA     7,1(0,7)        INCREMENTA CONTADOR DE BRANCOS
  48 INCRE1    LA     2,1(0,2)        INCREMENTA APONTADOR DE ENTRADA
  49 INICIO    CR     2,4             COMPARA FIM DE ENTRADA
  50           BNE    TESTE
  51           MVI    CHAVE,C'1'      FAZ CHAVE = '1'
  52           B      TECON
  53 TESTE     CLI    0(2),C' '       TESTA SE A POS. DE ENTRADA E BRANCO
  54           BE     INCRE0
  55 TECON     C      7,ZERO          TESTA SE CONTADOR = 0
  56           BE     MOVE1
  57           C      7,UM            TESTA SE CONTADOR = 1
  58           BE     MOVE2
  59           C      7,DOIS          TESTA SE CONTADOR = 2
  60           BE     MOVE3
  61           MVI    0(6),X'BD'      MOVE 'BD' PARA SAIDA
  62           LA     6,1(0,6)        INCREMENTA O APONTADOR DA SAIDA
  63           STC    7,0(0,6)        MOVE NO. DE BRANCOS PARA SAIDA
  64 MOVE0     LA     6,1(0,6)        INCREMENTA O APONTADOR DA SAIDA
  65           SR     7,7             CONTADOR = 0
  66 MOVE1     CLI    CHAVE,C'1'      TESTA SE CHAVE = '1'
  67           BNE    MOVE4
```

```
STMT   SOURCE STATEMENT                                          F01MAY

   69 *
   70 ********************************************************************
   71 *         FIM DA ROTINA, DEFINICAO DE AREAS E CONSTANTES           *
   72 ********************************************************************
   73 *
   74           SR     6,3              R6 CONTEM O COMPR. DA SAIDA
   75           STH    6,6(0,5)         RESTAURA O DV..SAIDA
   76           L      13,SALVA+4       RESTAURA REGISTRADERES E
   77           RETURN (14,12),RC=0 DEVOLVE O CONTROLE
   81 SALVA     DS     18F
   82 ZERO      DC     X'00000000'
   83 UM        DC     X'00000001'
   84 DOIS      DC     X'00000002'
   85 CHAVE     DC     C'0'
   86           END    COMP
```

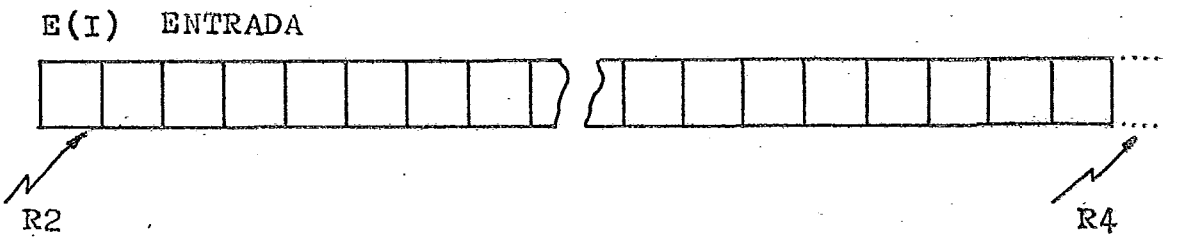
2.4. ROTINA DE DESCOMPRESSÃO

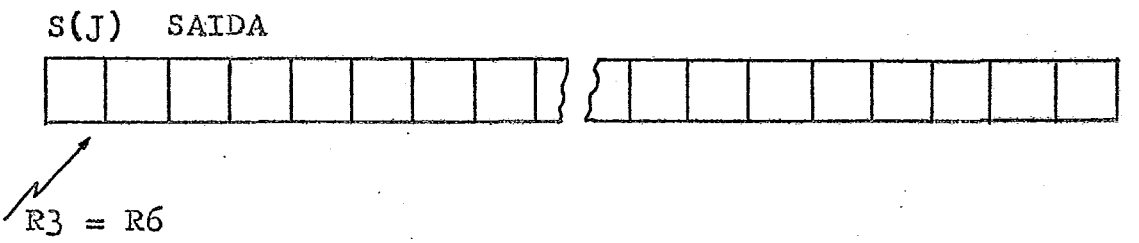
A rotina de descompressão de brancos que desenvolvemos é chamada através de dois parâmetros da seguinte forma:

CALL DCOMP (ENTRADA,SAIDA);

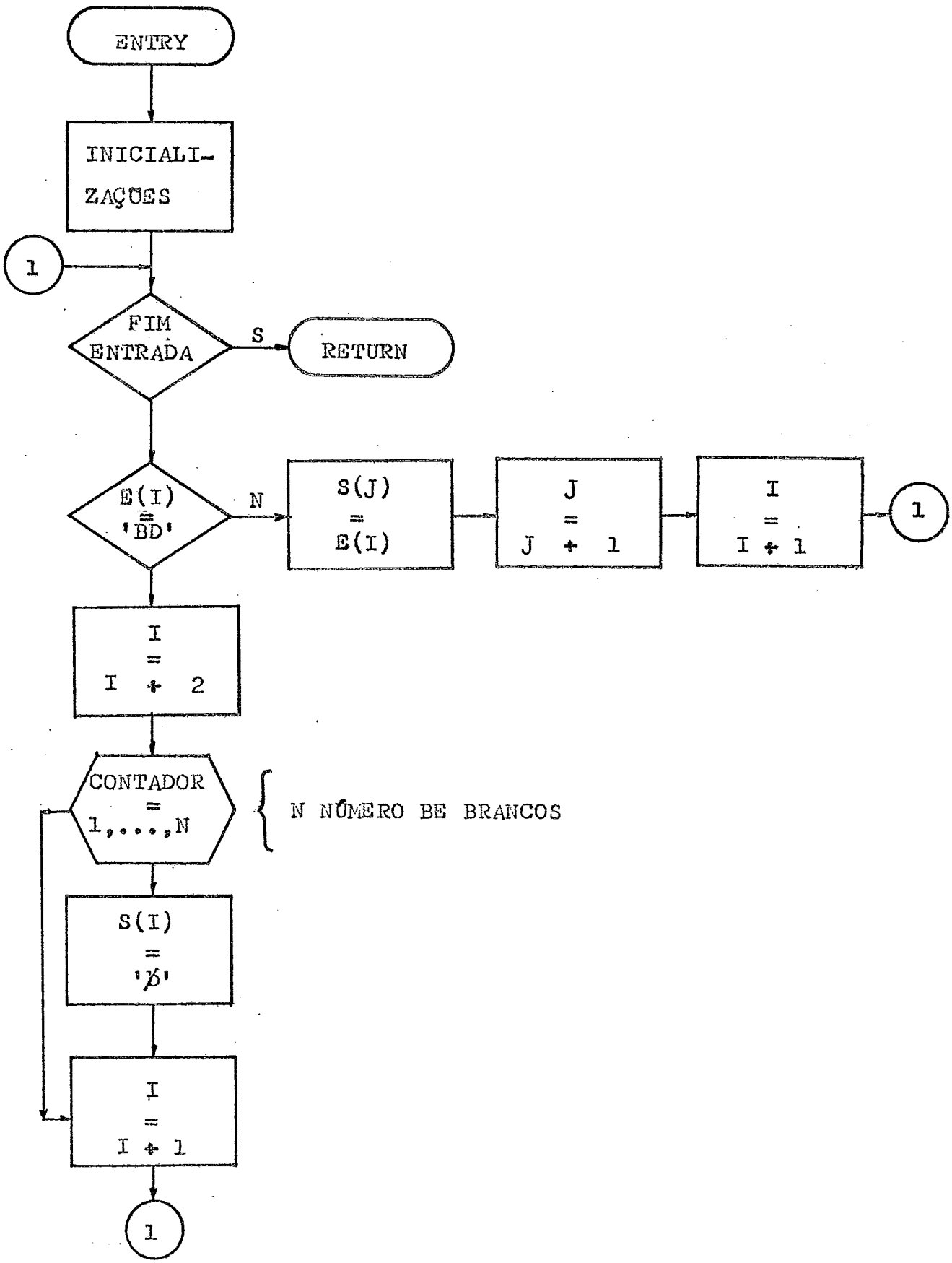
ENTRADA: É o endereço da área a ser descompressa

SAIDA: É o endereço de uma área para a devolução da informação na sua forma original ou descompressa





Apresentamos a seguir o diagrama de blocos e a codificação da rotina de descompressão 'DCOMP'


ENTRY
INICIALI-
ZAÇÕES
1
FIM
ENTRADA
S
RETURN
E(I)
=
'BD'
N
S(J)
=
E(I)
J
=
J + 1
I
=
I + 1
1
I
=
I + 2
CONTADOR
=
1,...,N
N NÚMERO BE BRANCOS
S(I)
=
'b'
I
=
I + 1
1

```
STMT   SOURCE STATEMENT                                          F01MAY7
   1 *
   2 ********************************************************************
   3 *         IMPLEMENTACAO DA ROTINA DE DESCOMPRESSAO DE BRANCOS      *
   4 *         LUCIANO PIETRACCI                                        *
   5 ********************************************************************
   6 *
   7           PRINT NOGEN
   8 DCOMP     CSECT
   9           SAVE  (14,12)
  12           LR    12,15
  13           USING DCOMP,12
  14           ST    13,SALVA+4
  15           LR    11,13
  16           LA    13,SALVA
  17           ST    13,8(11)
```

```
ADDR2  STMT   SOURCE STATEMENT                                    F01MAY7
        19 *
        20 ********************************************************************
        21 *          INICIALIZACOES                                          *
        22 ********************************************************************
        23 *
00000   24            L     4,0(0,1)        CARREGA O END. DO DV..ENTRADA
00000   25            L     2,0(0,4)        CARREGA EM R2 O END. DE ENTRADA
00004   26            L     5,4(0,1)        CARREGA O END. DO DV..SAIDA
00000   27            L     3,0(0,5)        CARREGA EM R3 O END. DE SAIDA
00006   28            LH    4,6(0,4)        CARREGA EM R4 O COMPR. DE ENTRADA
        29            AR    4,2             R4 APONTA PARA O FIM DA ENTRADA
        30            LR    6,3             R6 APONTA PARA O INICIO DA SAIDA
```

```
STMT   SOURCE STATEMENT                                          F01MAY72
   32 *
   33 ************************************************************************
   34 *         PROCESSAMENTO E DESCOMPRESSAO                                *
   35 ************************************************************************
   36 *
   37 INICIO   CR    2,4             TESTA FIM DE ENTRADA
   38          BE    FIMDCOMP
   39 TESTBD   CLI   0(2),X'BD'      TESTA SF POS. DE ENTRADA = 'BD'
   40          BE    MOVEB
   41          MVC   0(1,6),0(2)     MOVE A LETRA PARA SAIDA
   42          LA    6,1(0,6)        INCREMENTA O APONTADOR DA SAIDA
   43          LA    2,1(0,2)        INCREMENTA O APONTADOR DA ENTRADA
   44          B     INICIO
   45 MOVEB    LH    7,0(0,2)        CARREGA EM R7 'BDXX'
   46          LA    2,2(0,2)        INCREMENTA DE 2 O APONTADOR ENTRADA
   47          N     7,MASC          R7 CONTEM O NO. DE BRANCOS 'XX'
   48 MOVERB   MVI   0(6),C' '       MOVE BRANCO PARA SAIDA
   49          LA    6,1(0,6)        INCREMENTA O APONTADOR DE SAIDA
   50          BCT   7,MOVERB        REPETE A OPERACAO 'XX-1' VEZES
   51          B     INICIO
```

```
STMT   SOURCE STATEMENT                                          F01MAY72
   53 *
   54 ********************************************************************
   55 *         FIM DA ROTINA, DEFINICAO DE AREAS E CONSTANTES           *
   56 ********************************************************************
   57 *
   58 FIMDCOMP SR    6,3             R6 CONTEM O COMPR. DA SAIDA
   59          STH   6,6(0,5)        RESTAURA O DV..SAIDA
   60          L     13,SALVA+4      RESTAURA REGISTRADORES E
   61          RETURN (14,12),RC=0   DEVOLVE O CONTROLE
   65 *
   66 SALVA    DS    18F
   67 MASC     DC    X'000000FF'
   68          END   DCOMP
```

2.5. APLICAÇÃO E RENDIMENTO DAS ROTINAS

Foram desenvolvidos dois programas em PL/1 para testar as rotinas e seu rendimento.

O primeiro programa lê 1000 registros de uma fita 'SPOOL' e os grava na forma compressa num disco.

O segundo lê os registros gravados no disco na forma compressa e os imprime na forma normal ou descompressa.

No teste que fizemos foram lidos 133.000 bytes, isto é, 1.000 registros de 133 bytes, e o número de bytes gravados no disco, após a compressão feita pela rotina 'COMP', foi 31.341. Podemos observar que o rendimento foi bastante elevado tendo em vista que:

COMPRESSÃO ABSOLUTA = 133.000 - 31.341 = 101.695

COMPRESSÃO RELATIVA = 101.695 / 133.000 = 0,764353

COMPRESSÃO PERCENTUAL = 76,4353%

Isto significa que se fossem necessários 10 volumes de fita para armazenar todo o arquivo, este na sua forma compressa ocuparia apenas 3 volumes de fita.

Em apêndice apresentamos um exemplo de como podem ser utilizadas as rotinas 'COMP' e 'DCOMP'.

3. DESCRIÇÃO DE UMA ROTINA MAIS GERAL:

Usando as idéias expostas, nos parágrafos anteriores, passa-

mos a descrever agora uma rotina de compressão mais geral e sofisticada. [3]

Podemos propor, em primeiro lugar, que as rotinas de compressão e a de descompressão formem um único módulo.

Através de um parâmetro podemos indicar se o registro a ser processado deve ser compresso ou descompresso.

A rotina pode ter vários 'ENTRY POINTS' de tal maneira que possa ser chamada por programas escritos em linguagens, diferentes:

EXEMPLO:	COMPL1	ENTRY POINT para o PL/1
	COMASS	" " " o ASSEMBLER
	COMCOB	" " " o COBOL

os 'ENTRY POINTS' para o COBOL e o ASSEMBLER podem ser os mesmos, tendo em vista que os parâmetros são passados da mesma maneira nas duas linguagens.

Os parâmetros para chamar a rotina deverão ser:

1. ENTRADA - endereço do registro a ser processado
2. SAIDA - endereço de uma área para a devolução do registro compresso ou descompresso
3. COMPENT - comprimento do registro de entrada (este parâmetro não é necessário quando o programa principal é em PL/1 pois o D.V. contem esta informação).

4. COMPSAI - comprimento do registro de saída (não necessário se o programa principal for em PL/1).

5. COD - código para indicar se o registro deve ser compresso ou descompresso.

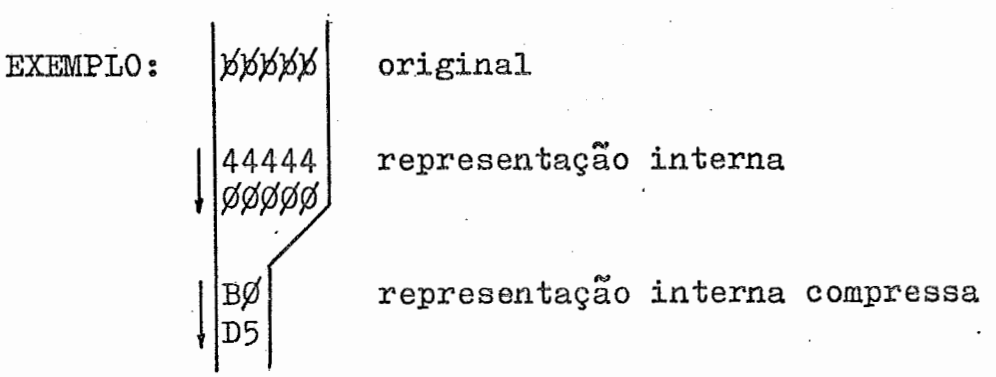
6. KEY - comprimento da chave do registro (que deverá ocupar as primeiras posições do registro) neste caso o número de bytes indicados por este parâmetro não sofrerá compressão, isto para permitir uma consulta mais rápida ao arquivo.

   Este parâmetro será opcional.

A rotina pode compactar não somente brancos, mas também zeros compactados, da seguinte forma:

1. sequencias de mais de dois brancos (caráter sentinela X'BD')

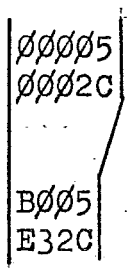
   BDXX onde XX indica o número de brancos retirados



2. sequencias de mais de cinco "zeros compactados" sem sinal (caráter sentinela X'BE')

   BEXX onde XX indica o número de bytes retirados

EXEMPLO:



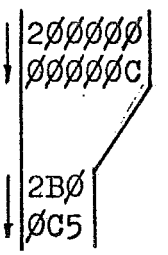
representação interna

representação interna compressa

3. sequencias de mais de quatro "zeros compactados com sinal positivo" (caráter sentinela X'BC')

BCXX onde XX indica o número de bytes retirados

EXEMPLO:



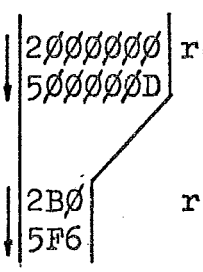
representação interna

representação interna compressa

4. sequencias de mais de quatro "zeros compactados com sinal negativo" (caráter sentinela X'BF')

BFXX onde XX indica o número de bytes retirados

EXEMPLO:

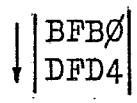


representação interna

representação interna compressa

Se as sequencias forem maiores de 255 bytes serão feitas compressões sucessivas; por exemplo se quiservos comprimir uma sequencia de 257 brancos teríamos:

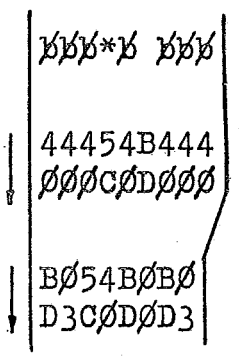
mas se a sequencia fosse de 259 brancos teríamos:



Um outro detalhe importante a ser considerado é o seguinte: suponhamos que no registro a ser compresso apareçam as configurações X'BD', X'BC', X'BE' ou X'BF' neste caso devemos ter o cuidado de acrescentar um byte contendo zeros, para evitar erros na descompressão.

EXEMPLO:

sequencia a ser compressa:



OBSERVAÇÕES:

a) Pode acontecer, como caso extremo, que o registro compresso seja maior do que o registro na forma normal ou descompressa.

b) Alguns tipos de terminais ou concentradores usam certas configurações (bytes) como caracteres de controle (sincronização, endereçamento de linha etc.), neste caso, devemos prever a impossibilidade de usar tais configurações para representar o número de BYTES retirados. [7]

4. CONCLUSÕES:

É comum, especialmente quando desenhamos arquivos para sistemas on-line, desenvolver registros sofisticados. Em muitos casos o analista ou o programador é obrigado a desenhar um complexo registro de tamanho variável com campos variáveis, ou campos sem informações.

Estes registros são complexos não somente no seu desenho, mas necessitam também, de uma programação complexa para poderem ser processados.

As rotinas descritas, neste capítulo, permitem simplificar as duas coisas. Ao mesmo tempo oferecem a vantagem de aumentar a eficiencia da memória auxiliar usando registros de tamanho variável.

Para o programador a compressão é transparente, este não se preocupa com o fato de que os registros são compressos antes de serem gravados e que a descompressão é feita logo após a leitura dos mesmos.

CAPÍTULO – 3

1. COMPRESSÃO DE DADOS USANDO CÓDIGOS VARIÁVEIS

Este tipo de compressão permite diminuir o volume das informações a serem armazenadas ou transmitidas, aproveitando a probabilidade estatística de ocorrencia das unidades das informações (caracteres no caso), atribuindo menores representações de bits para os caracteres que se repetem mais frequentemente e representações maiores para os que ocorrem pouco frequentemente.

O Morse, por exemplo, usa menores códigos para as letras comuns e códigos maiores para as outras menos frequentes. Devemos observar, porém, que o Morse possue três símbolos para a representação dos seus caracteres:

* o ponto .
* a linha —
* e o espaço para a separação dos caracteres.

A compressão de dados obtida usando códigos variáveis, perde obviamente a sua eficiencia se as propriedades estatísticas das informações mudam com o passar do tempo.

Este tipo de compressão, portanto, deve ser aproveitado para arquivos ou mensagens cujo conteúdo é estatísticamente estável.

2. ALGUMAS PROPRIEDADES IMPORTANTES DOS CÓDIGOS VARIÁVEIS

A FIGURA-2 dá dois códigos diferentes para representar as

letras do alfabeto e o caráter branco. Estes códigos possuem propriedades que são de certo interesse.

Em primeiro lugar definiremos como sendo "GRUPO" de um código cada uma das configurações de bits atribuída a um caráter do alfabeto.

Os códigos apresentados na FIGURA-2 são formados de grupos de comprimento variável, isto é, o grupo atribuído a cada caráter é uma sequencia de dígitos binários, mas os grupos não são formados pelas mesmas sequencias de dígitos binários e o número destes dígitos é variável.

Existe uma relação entre a probabilidade de ocorrencia de um caráter (Pi) e o número de bits (Ci) usado para representar este caráter; esta relação pode ser empressa da seguinte forma:

se $P_1 < P_2 < \ldots < P_{N-1} < P_N$ então

$C_1 > C_2 > \ldots > C_{N-1} > C_N$ [15] .

onde N é o número de caracteres do alfabeto.

Veremos adiante uma relação mais precisa que permite calcular o valor Ci em função de Pi.

Os códigos representados na FIGURA-2 possuem a propriedade do "PREFIXO", isto é, nenhum grupo de um código é o início de outro grupo do mesmo código. Esta propriedade, como veremos, é importante para permitir a decodificação de uma mensagem. [4] [9]

ALFABETO	*PROB.	CÓDIGO DE HUFFMAN	CÓDIGO ALFABÉTICO
∅	0.1859	000	00
A	0.0612	0100	0100
B	0.0127	011111	010100
C	0.0128	11111	010101
D	0.0317	01011	01011
E	0.1031	101	0110
F	0.0208	001100	011100
G	0.0152	011101	011101
H	0.0467	1110	01111
I	0.0575	1000	1000
J	0.0008	0111001110	1001000
K	0.0049	01110010	1001001
L	0.0321	01010	100101
M	0.0198	001101	10011
N	0.0574	1001	1010
O	0.0632	0110	1011
P	0.0152	011110	110000
Q	0.0008	0111001101	110001
R	0.0484	1101	11001
S	0.0514	1100	1101
T	0.0796	0010	1110
U	0.0228	11110	111100
V	0.0083	0111000	111101
W	0.0175	001110	111110
X	0.0013	0111001100	1111110
Y	0.0164	001111	11111110
Z	0.0005	0111001111	11111111
		CUSTO 4.1195	CUSTO 4.1978

* As probabilidades desta tabela foram obtidas da pesqisa de textos na lingua Inglesa.

FIGURA - 2

Definimos "CUSTO" de um código como sendo o número médio de dígitos binários necessário para representar um caráter desse código. [4]

Se P_i á a probabilidade de ocorrencia do caráter L_i; se C_i é o número de bits usado para representar o caráter L_i e se N é o número de caracteres do alfabeto, diremos que o custo médio é expresso por

$$\sum_{i=1}^{N} Pi * Ci$$

O primeiro desses códigos chamado CÓDIGO DE HUFFMAN, foi construído pelo método dado por Huffman e têm a vantagem de possuir o menor "custo" possível gozando da propriedade do prefixo.

O segundo desses códigos, chamado código alfabético, têm a seguinte propriedade: a ordem alfabética dos caracteres corresponde à ordem numérica dos grupos.

Este último código, também, foi construído de maneira a possuir o menor "custo" possível.

Podemos notar que a imposição do código ser alfabético aumenta um pouco o custo médio deste código.

Nos parágrafos que seguem desenvolveremos alguns métodos para construir códigos variáveis que satisfaçam às propriedades anteriormente citadas.

3. ALGORÍTMOS PARA DETERMINAR CÓDIGOS VARIÁVEIS:

3.1. CÓDIGOS DE SHANNON-FANO:

O processo de codificação deve ser reversível, isto significa que deve ser possível decifrar onde um grupo do código inicia e onde termina.

Nós dispomos apenas de zeros e uns, portanto, a divisão entre grupos de um código deve ser feita através de um grupo reservado para esta finalidade, em outras palavras, o código deve ter a propriedade que nenhum dos seus grupos seja o início de um outro grupo (definimos esta propriedade como sendo a propriedade do prefixo).

Um dos códigos que satisfaz a esta propriedade é o código de Shannon-Fano. [9]

Este código é determinado da seguinte maneira:

a) Constrói-se uma tabela, com todos os caracteres do alfabeto escolhido, em ordem decrescente de probabilidade de ocorrencia.

L_i	P_i	CÓDIGO	PASSO
A	0.25000	00	2
F	0.25000	01	1
B	0.12500	100	3
C	0.12500	101	2
D	0.06250	1100	4
N	0.06250	1101	3
O	0.03125	1110	
.	.	.	.

b) A tabela deve ser dividida em duas partes, de maneira que a diferença entre a soma das probabilidades dos caracteres que formarão a parte superior e a soma das probabilidades dos caracteres que formarão a parte inferior, seja a menor possível (Passo-1).

c) Atribui-se '0' (zero) como primeiro dígito dos grupos dos caracteres da parte superior da tabela.

Atribui-se '1' (um) como primeiro dígito dos grupos dos caracteres da parte inferior da tabela.

d) Cada uma das duas partes deve ser dividida em duas sub-partes em que a soma das probabilidades dos caracteres que formarão a sub-parte superior seja tão próxima quanto possível da soma das probabilidades dos caracteres que formarão a sub-parte inferior. (Vide definição - b) (Passo-2).

e) Atribui-se '0' (zero) como segundo dígito dos grupos dos caracteres da sub-parte superior.

Atribui-se '1' (um) como segundo dígito dos grupos dos caracteres da sub-parte inferior.

f) O processo é repetido até existir apenas um caráter em cada sub-parte (Passo-i).

Na figura abaixo são representados alguns códigos obtidos para alfabetos em que os caracteres tem diferentes probabilidades.

L_i	P_i	CÓDIGO	PASSO
A	1/2	0	1
B	1/4	10	2
C	1/8	110	3
D	1/16	1110	4
E	1/32	11110	5
F	1/64	11111	.
G	.		.
H	.		.
I	.		

L_i	P_i	CÓDIGO	PASSO
A	1/4	00	2
B	1/4	01	1
C	1/8	100	3
D	1/8	101	2
E	1/16	1100	4
F	1/16	1101	3
G	1/32	11100	.
H	.		.
I	.		

L_i	P_i	CÓDIGO	PASSO
A	1/8	000	3
B	1/8	001	2
C	1/8	010	3
D	1/8	011	1
E	1/8	100	3
F	1/8	101	2
G	1/8	110	3
H	1/8	111	

Existe, como vimos, uma relação entre o comprimento de cada "grupo" do código (C_i) e a propriedade de ocorrencia do caráter correspondente (P_i); esta relação é dada por:

$$C_i = \log_2 \frac{1}{P_i} \qquad [12]$$

Exemplo:

Se o caráter 'A' têm probabilidade $P_i = 0,25$ num certo alfabeto podemos afirmar que o comprimento do grupo que representará o carater 'A' será:

$$\log_2 \frac{1}{0,25} = 2$$

O comprimento do código (C_i) será igual a $\log_2 \frac{1}{P_i}$ se for possível dividir a tabela em sub-partes que tenham somas de probabilidades exatamente iguais: quando isto não for possível, deve acrescentar-se um dígito a alguns grupos do código.

O número real de dígitos por grupo é dado, portanto, pela relação: [12]

$$\left[\log_2 \frac{1}{P_i}\right] \leq C_i \leq \left[\log_2 \frac{1}{P_i}\right] + 1 \qquad \text{(FIGURA-3)}$$

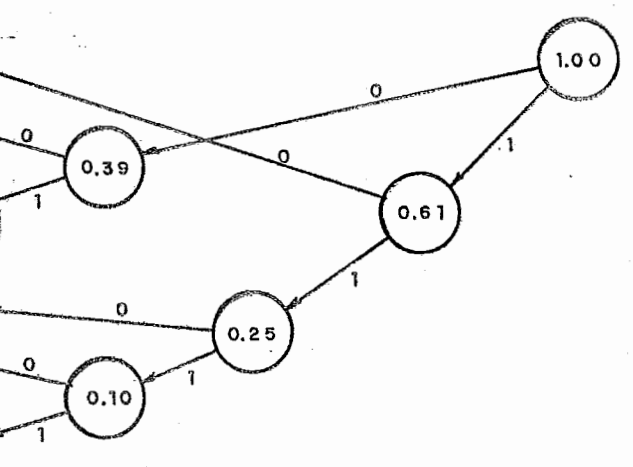
3.2. CÓDIGO DE HUFFMAN:

Descreveremos neste parágrafo outro processo para construção de um outro tipo de código variável chamado código de Huffman. [5] [15]

Este código é construído da seguinte maneira:

a) Constrói-se uma lista com todos os caracteres do alfabeto escolhidos, em ordem decrescente de probabilidade de ocorrencia.

L_i	P_i
A	0.36
B	0.24
C	0.15
D	0.15
E	0.05
F	0.05



b) Escolhem-se os dois caracteres com a menor probabilidade de ocorrencia; ao grupo de um, atribui-se '0' (zero) ao grupo do outro, atribui-se '1' (um).

c) A probabilidade de ocorrencia dos dois caracteres é somada e o resultado entra a fazer parte da lista que fica diminuída de um elemento.

d) O processo é repetido a partir do passo-b até que fique um único elemento na lista.

NO EXEMPLO DADO TEMOS OS SEGUINTES GRUPOS:

L_i	P_i	CÓDIGO	$P_i . C_i$
A	0.36	10	0.72
B	0.24	00	0.48
C	0.15	01	0.30
D	0.15	110	0.45
E	0.05	1110	0.20
F	0.05	1111	0.20
	$\Sigma P_i = 1$		$\Sigma P_i . C_i = 2.35$

Se fosse usado um código binário de comprimento fixo seriam necessários três bits para representar cada caráter da tabela ($2^2 < 6 < 2^3$). Usando o Código de HUFFMAN, construido acima. Podemos usar em média 2.35 bits por caráter.

OBSERVAÇÃO: Os resultados obtidos com os dois métodos o de SHANNON FANO e o de HUFFMAN são os mesmos em termos de custo.

Implementaremos no próximo parágrafo um algorítmo para construir o código de Huffman dado um conjunto de carác teres e suas respectivas probabilidades de ocorrencia.

3.3. IMPLEMENTAÇÃO DO ALGORITMO PARA A DETERMINAÇÃO DO CÓDIGO DE HUFFMAN:

3.3.1. DETERMINAÇÃO DO ALFABETO E DAS PROBABILIDADES DE OCORRENCIA:

Antes de podermos implementar o algoritmo de HUFFMAN devemos determinar o conjunto de caracte res, dos quais necessitamos para representar as informações desejadas, e suas respectivas probabilidades de ocorrencia. Como exemplo desenvolve mos uma pesquisa num conjunto de 192.366 nomes armazenados num campo fixo de 35 posições.

Definimos o conjunto de caracteres como sendo o de 28 elementos representado abaixo:

{*, ∅, A, B, C, D, E, F, G, H, I, J, K, L,
M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}

onde "*" representa qualquer caráter especial, co mo por exemplo apóstrofe " ' ", ponto " . ", traço " - " etc. ..., e " ∅ " representa o caráter branco.

A pesquisa foi feita considerando apenas a parte

útil dos nomes, isto é, desprezando os brancos finais.

Deste modo o número total de caracteres lidos não corresponde ao produto do número de registros lidos, pelo comprimento do registro:

192.366 x 35 = 6.732.810 bytes; e sim a

4.007.515 que é o número de caracteres úteis dos nomes.

Podemos observar que se considerarmos o campo do nome como sendo variável, a simples supressão dos brancos finais proporcionaria uma compressão de 40,47 %.

$$CP = \frac{Cl - CF}{Cl} * 100 = \frac{6.732.810 - 4.007.515}{6.732.810} * 100$$

$$CP = \frac{2.725.295}{6.732.810} * 100 = 40,47$$

Os resultados obtidos na nossa pesquisa estão representados na FIGURA-3.

Os valores de P_i foram obtidos dividindo F_i (número de ocorrencias do caráter L_i) por 4.007.515 e foram feitas aproximações para podermos ter

$$\sum_{i=1}^{N} P_i = 1$$

LETRA	P_i	C_i	$-\log_2 P_i$	CÓDIGO
A	0.123778	3	3.0141	000
∅	0.102294	3	3.2891	101
O	0.088888	4	3.4918	0111
E	0.088202	4	3.5030	0110
I	0.078269	4	3.6754	0101
R	0.075123	4	3.7345	0100
S	0.055359	4	4.1750	1110
N	0.051280	4	4.2854	1100
L	0.051203	4	4.2876	1001
D	0.041137	4	4.6034	1000
T	0.035258	5	4.8258	00110
C	0.032247	5	4.9546	00101
M	0.029643	5	5.0761	11111
U	0.025580	5	5.2888	11010
G	0.016022	6	5.9638	001001
B	0.014258	6	6.1321	001000
V	0.014247	6	6.1332	111101
H	0.013834	6	6.1756	110111
P	0.012613	6	6.3088	110110
F	0.010679	7	6.5491	0011111
J	0.009339	7	6.7424	0011110
Z	0.008891	7	6.8133	0011101
※	0.007823	7	6.9981	1111001
Y	0.004393	8	7.8304	00111001
K	0.003293	8	8.2461	11110001
W	0.002742	8	8.5108	11110000
Q	0.002369	9	8.7218	001110001
X	0.001236	9	9.6606	001110000
	1.000000			CUSTO = 4.208534

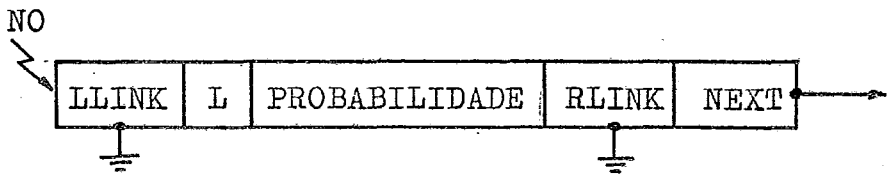
FIGURA - 3

3.3.2. DETERMINAÇÃO DO CÓDIGO DE HUFFMAN:

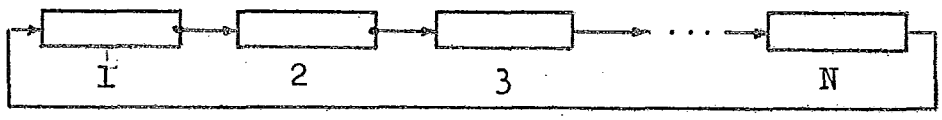
Para a determinação do código de HUFFMAN foi usado o algoritmo descrito no parágrafo 3.2.

A codificação deste algorítmo em PL/1 encontra-se no apêndice.

1) Para cada caráter do alfabeto (L_i) é criado um nó contendo o caráter, sua probabilidade e três apontadores.



2) É criada uma fila circular de N nós, sendo N o número de caracteres do alfabeto.



3) Percorre-se a fila procurando os dois caracteres com as menores probabilidades; cria-se um novo nó em que RLINK aponta para um dos caracteres e LLINK para o outro. A probabilidade dos dois caracteres é somada dando origem a probabilidade do novo nó.

4) Retiram-se da fila os dois nós dos caracteres de menores probabilidades, insere-se o novo nó.

5) O processo é repetido até que na fila exista um único nó cuja probabilidade seja igual a 1.0

6) Percorre-se a árvore binária obtida em "POS-TORDER" obtendo-se os grupos do código para cada folha.

Os resultados obtidos neste algorítmo estão representados na tabela da FIGURA-3.

Podemos observar como existe a correspondencia entre o comprimento de cada grupo (C_i) e o logarítmo do inverso da probabilidade de ocorrencia do caráter correspondente.

$$(\log_2 \frac{1}{P_i} = -\log_2 P_i)$$

O custo médio obtido com este código é de 4,208.534 bits/caráter.

4. IMPLEMENTAÇÃO DAS ROTINAS DE COMPRESSÃO E DESCOMPRESSÃO PARA CÓDIGOS VARIÁVEIS.

4.1. ROTINA DE COMPRESSÃO

Desenvolvemos neste parágrafo uma rotina de compressão usando códigos variáveis. A rotina foi desenvolvida para ser aplicada em um arquivo contendo apenas nomes (visto que o nosso código foi obtido desse arquivo) evidentemente, esta não é uma restrição, pois a rotina poderá

ser generalizada para outros tipos de arquivos.

A rotina é constituída dos seguintes passos.

1) Retiram-se os brancos finais do registro, por exemplo:

ROSSI␣MARIO␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣

2) Cada letra, da esquerda para a direita, é codificada com o código variável obtido pelo algoritmo de HUFFMAN

GRUPO

BYTE 0100011111110111001011011111100001000101011110000

3) Como a unidade de gravação, no nosso caso, é o byte devemos completar os bits restantes com um grupo que permita identificar onde a mensagem acabou.

Os grupos utilizados são os descritos na tabela abaixo:

GRUPOS	
1	quando sobra 1 bit
10	quando sobram 2 bits
100	" " 3 "
1000	" " 4 "
10000	" " 5 "
100000	" " 6 "
1000000	" " 7 "
10000000	" " 0 "

Como podemos observar quando a mensagem compressa ocupa um número inteiro de bytes acrescenta-se um grupo de um byte, isto torna-se necessário para podermos restaurar a mensagem.

Para maiores detalhes sobre a rotina consultar o apêndice.

4.2. ROTINA DE DESCOMPRESSÃO

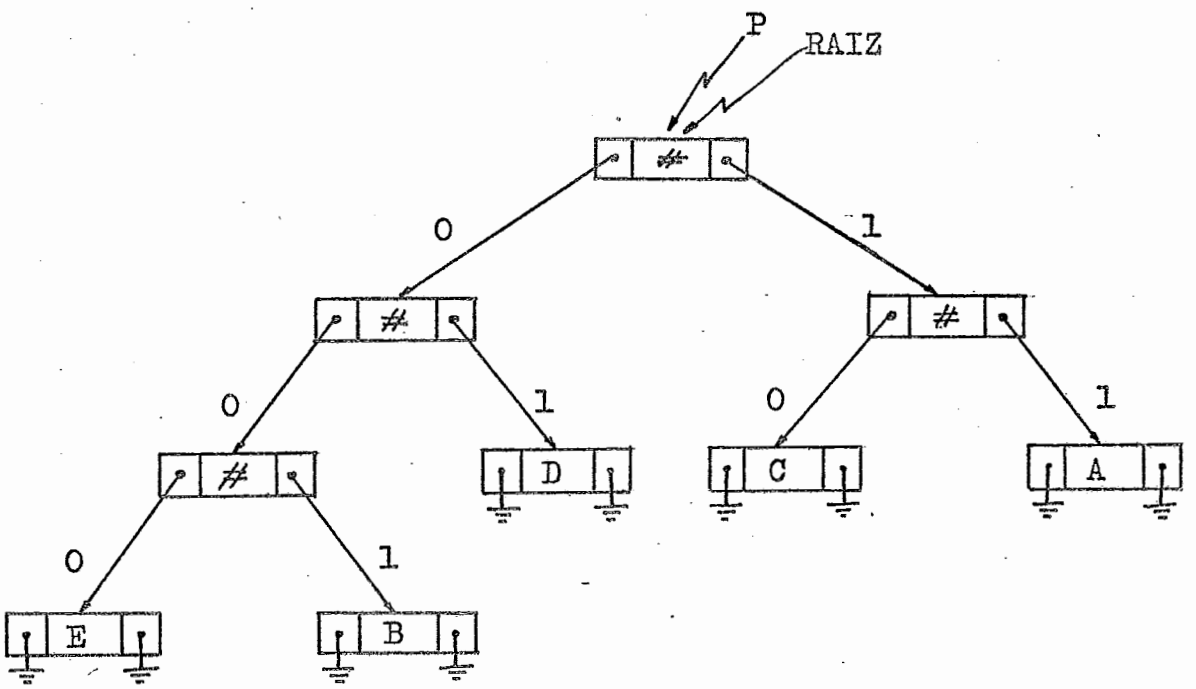
Esta rotina foi desenvolvida para descomprimir o arquivo de nomes compresso pela rotina descrita no parágrafo anterior. Daremos exemplos de como é efetuada a descompressão.

Para facilitar o nosso estudo vamos admitir que o conjunto de caracteres do nosso alfabeto seja formado apenas de 5 elementos ao invés de 28.

LETRA	P_i	CÓDIGO
A	0,30	11
C	0,20	10
D	0,20	01
B	0,15	001
E	0,15	000
	$\Sigma P_i =$ 1,00	CUSTO 2,3

A rotina é constituida dos seguintes passos:

1) Constrói-se uma árvore binária em que cada folha corresponde a uma letra. (Isto é sempre possível considerando a propriedade do prefixo)



2) Determina-se o fim da sequencia de entrada detetando o primeiro 1, da direita para esquerda.

Por exemplo, se a sequencia de entrada fosse:

'1110110010001000'

fim da sequencia

3) Lê-se a sequencia de bits, da esquerda para direita, percorrendo a árvore até encontrar uma folha.

4) A letra contida na folha é movida para a área de saída.

5) O processo é repetido, a partir do passo 3, até o fim da sequencia de entrada.

Em nosso exemplo teríamos como sequencia de saída:

Para maiores detalhes sobre a rotina consultar o apêndice.

4.3. APLICAÇÃO E RENDIMENTOS DAS ROTINAS:

Como vimos nos parágrafos anteriores, retirando os brancos finais dos registros de nomes, podemos obter uma compressão percentual de aproximadamente 40%, isto significa que a parte útil dos nomes ocupa apenas 60% do volume total do arquivo.

Considerando que estes 60% do volume total são codificados com códigos de 8 bits por caráter e que com o código variável desenvolvido, podemos codificar cada caráter usando em média 4,2 bits, chegamos à conclusão de que podemos reduzir o volume útil do arquivo de aproximadamente 50%, obtendo assim uma compressão percentual de aproximadamente 70% em relação ao volume

inicial do arquivo.

Na aplicação que desenvolvemos foram lidos 10.000 nomes do arquivo portanto 350.000 bytes. Com o sistema descrito no parágrafo 4.1. gravamos apenas 110.921 bytes obtendo, portanto, uma compressão percentual de 68,308%

$$\frac{350.000 - 110.921}{350.000} \; 100 = \frac{239.079}{350.000} \; 100 = 68,308\%$$

5. COMPRESSÃO POR CODIFICAÇÃO CONDICIONAL

5.1. DESCRIÇÃO DO MÉTODO

Este método consiste em codificar cada caráter com grupos de tamanho variável de bits em função de sua probabilidade de ocorrencia que é condicionada aos caracteres que o precedem.

Como vimos nos parágrafos anteriores cada caráter é codificado com um grupo de bits, variável em função da sua probabilidade de ocorrencia. Construiremos agora alguns códigos variáveis considerando porém, a probabilidade de ocorrencia de um caráter condiciona ao caráter que o precede.

Por exemplo, se considerarmos a letra "U" sua probabilidade de ocorrencia, se considerada isoladamente, é de 0.025580 e será codificada com um grupo de 5 (cinco) bits '11010' (vide FIGURA-3). Se considerarmos porém,

a mesma letra condicionada a 'Q' (isto é o caráter Q precede U) veremos que sua probabilidade de ocorrencia sobe a 0,978529 e poderá ser codificado com um grupo de 1 (um) bit (vide tabela-20 em apêndice).

Isto significa que depois de um Q a probabilidade de ocorrencia de um U sobe a 0,978529 e que esta letra pode ser codificada com apenas 1 bit se memorizarmos a letra anterior.

5.2. DETERMINAÇÃO DO ALFABETO DAS PROBABILIDADES DE OCORRENCIAS E CONSTRUÇÃO DOS CÓDIGOS.

O arquivo que escolhemos para a nossa aplicação foi o arquivo, já conhecido, contendo 192.366 nomes.

O conjunto de caracteres escolhido foi de 28 elementos representados abaixo:

$$\{*,/,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z\}$$

A pesquisa foi feita considerando apenas a parte útil dos nomes.

Construímos um código para a primeira letra, em função de sua probabilidade de ocorrencia isoladamente, obtendo assim, os resultados da tabela-1 em apêndice.

Para estudarmos a probabilidade de ocorrencia dos demais caracteres condicionando-a ao caráter anterior pesquisamos a probabilidade de ocorrencia de cada grupo de

2 caracteres, obtendo assim as tabelas de 2 a 29 em apêndice.

Se por exemplo quisermos saber como deve ser codificada a letra O quando precedida da letra J bastará consultar a tabela 13 e veremos que o grupo correspondente a "O" é de 1 (um) bit.

Como podemos observar a quantidade de informação necessária para transmitir um caráter é menor se pudermos memorizar o caráter que o precede.

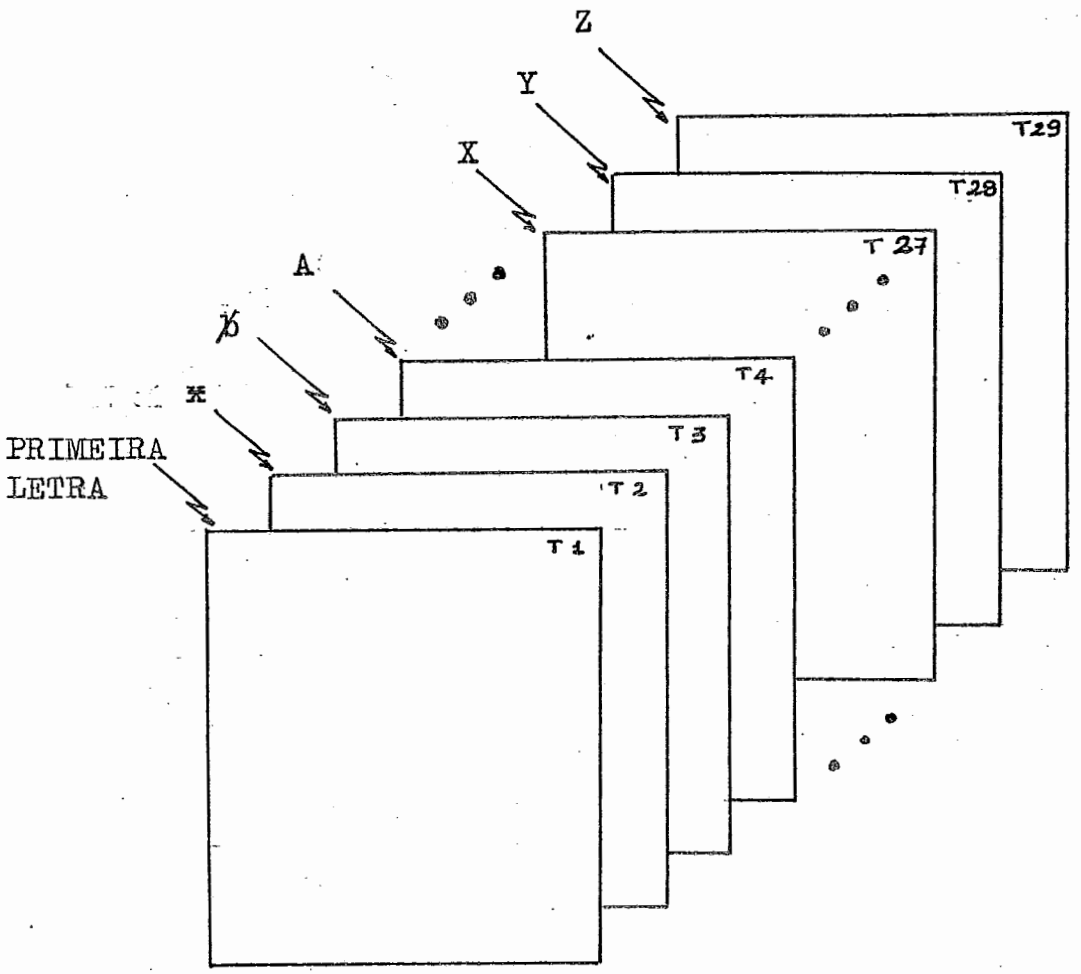
Para obtermos os códigos aplicamos sucessivamente o algorítmo de HUFFMAN descrito nos parágrafos anteriores, calculando para cada código o respectivo custo.

5.3. IDÉIAS PARA IMPLEMENTAR A ROTINA DE COMPRESSÃO

Para podermos implementar a rotina de compressão devemos estar de posse dos dados contidos nas tabelas de 1 a 29.

Estes dados deverão fazer parte da rotina de compressão.

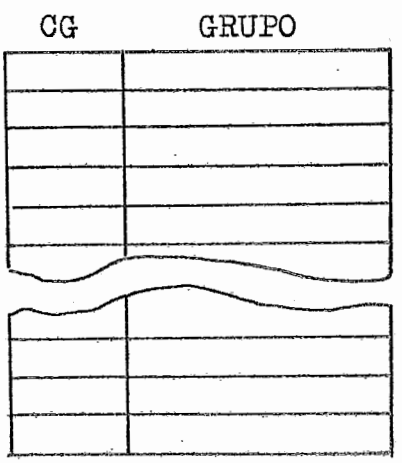
A rotina deverá conter portanto uma tabela para podermos codificar o primeiro caráter de cada nome e mais 28 tabelas para podermos codificar cada caráter em função do anterior.



Cada tabela possuirá as seguintes informações:

CG - Comprimento do Grupo - 1 byte

GRUPO - Grupo do Código - 2 bytes



Cada tabela ocupará 84 bytes. Sendo 29 tabelas teremos 2.436 bytes.

Para cada nome lido a primeira letra será codificada pela tabela-1 e as demais letras serão codificadas em função da anterior.

Por exemplo o nome 'JOSÉ' será codificado da seguinte maneira:

	CÓDIGO	C_i
J - TABELA-1	'111'	- 3
O - TABELA-13	'1'	- 1
S - TABELA-18	'010'	- 3
E - TABELA-22	'110'	- 3

11110101101000000

O último byte será completado com um grupo (como visto nos parágrafos anteriores) para podermos informar o fim da informação.

A pesquisa às tabelas será direta bastando para isto, converter a letra no endereço da tabela correspondente.

5.4. IDÉIAS PARA IMPLEMENTAR A ROTINA DE DESCOMPRESSÃO

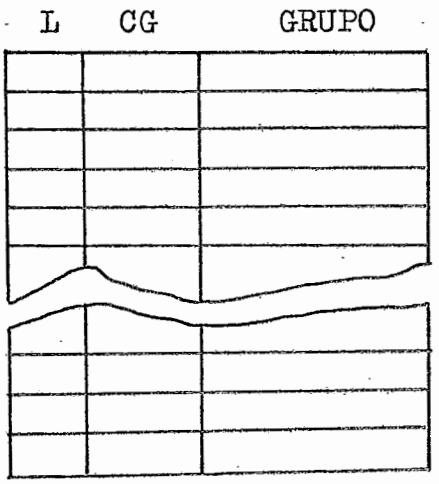
A rotina de descompressão deverá conter, também, 29 tabelas, uma para decodificar a primeira letra e as demais para decodificar cada caráter em função do anterior.

As tabelas para a rotina de descompressão deverão ser construídas da seguinte maneira:

L - LETRA - 1 byte contendo a letra correspondente a um código

CG - COMPRIMENTO DO GRUPO - 1 byte - este comprimento deverá ser em ordem crescente.

GRUPO - GRUPO DO CÓDIGO - 2 bytes.



Cada tabela ocupará 112 bytes sendo 29 tabelas teremos 3.248 bytes.

O método de pesquisa para as tabelas será o seguinte:

a) Seleciona-se a tabela em função do caráter (seja este o primeiro ou não)

b) Lê-se um bit, consultam-se na tabela os grupos de um bit, se não existir nenhum grupo igual ao lido lê-se

outro bit, consultam-se na tabela os grupos de dois bits; o processo é repetido até encontrarmos o grupo desejado.

O caráter correspondente ao grupo é movido para a sequencia de saída.

c) Volta-se ao passo a até terminar a sequencia de entrada.

Este processo de decodificação é menos eficiente que o da árvore binária descrito nos parágrafos anteriores para decodificar códigos variáveis, mas torna-se mais econômico em tamanho de memória. Neste caso devemos escolher entre tempo de CPU e tamanho de memória disponível.

Se as duas rotinas formarem um único módulo as tabelas podem ser comuns às duas rotinas.

5.5. RESULTADOS DA CODIFICAÇÃO CONDICIONAL

Como podemos observar as rotinas de compressão e descompressão, para este método, necessitam de uma quantidade maior de memória para o armazenamento das tabelas, no entanto, são bastante eficientes. Os resultados que obtivemos são os indicados na tabela-30 em apêndice.

Cada uma das tabelas de 2 a 29 representa um código, e o custo deste código. Na FIGURA-3 temos a probabilidade de ocorrencia de cada letra isoladamente. Multiplicando a probabilidade de ocorrencia de cada letra (P_i) da FI-

GURA-3 pelo custo do código que é necessário para representar todas as letras que vem após esta ($CUSTO_i$) e somando os resultados teremos o custo global médio por caráter no método de codificação condicional, que é:

$$\sum_{i=1}^{28} P_i \times CUSTO_i = 3.455589 \text{ bits / caráter}$$

isto significa que serão necessários em média 3,4 bits para representar um caráter.

Este método proporciona, portanto, uma compressão de 56,80% sôbre a parte útil dos nomes.

Os resultados que obtivemos não são os melhores por não ter sido feita uma consistencia dos nomes que apresentam certas irregularidades. Resultados melhores poderiam ser obtidos fazendo-se algumas padronizações.

O conceito de codificação condicional pode ser extendido a mais de 1 caráter, neste caso, porém, seria necessário um grande número de tabelas presentes na memória em tempo de compressão, o que tornaria estas funções pouco eficientes. (Vide conclusão).

Estes códigos são baseados nas propriedades estatísticas das informações que queremos processar. Estas propriedades quando variáveis, podem reduzir, enormemente, a capacidade de compressão dos códigos.

No entanto, é possível criar rotinas que permitam tornar os códigos adaptáveis às novas propriedades das in-

formações alterando as tabelas em função das novas distribuições.

CAPÍTULO - 4

1. COMPRESSÃO POR AGRUPAMENTO DE CARACTERES

Muitas vezes os arquivos comerciais contem conjuntos de caracteres que se apresentam repetidos com grande número de ocorrencias nos seus registros.

Esses conjuntos de caracteres podem ser formados de informações alfabéticas ou numéricas.

Com este método a compressão é obtida substituindo os conjuntos de caracteres mais frequentes por um único carater.[14]

A tabela dos grupos (conjuntos de caracteres) escolhidos, para serem substituidos, pode ser armazenada ou transmitida com as informações compressas ou pode ser parte integrante da rotina de compressão e de descompressão.

Como vimos no segundo capítulo, é conveniente, para representar esses conjuntos, usar configurações que não sejam utilizadas no texto a ser compresso.

A compressão por substituição de grupos proporciona bons rendimentos para dados densos de informações em que a redução de caracteres idênticos (visto no segundo capítulo) não traria grandes vantagens.

A eficiencia, no caso da compressão por agrupamento de caracteres apresenta alguns problemas pois devemos fazer várias comparações entre os dados e a tabela dos conjuntos de caracteres escolhidos. A pesquisa à tabela deverá portanto ser estudada de maneira a reduzir ao máximo o tempo necessário pa

ra identificar se um conjunto pertence ou não à tabela.

A rotina de descompressão como veremos não apresenta grandes problemas e é bastante eficiente.

Devemos observar que este método de compressão é baseado na probabilidade estatística de ocorrencia dos grupos (assim como nos códigos de tamanho variável), este tipo de compressão portanto é apropriado para arquivos ou dados cujo conteúdo é estatísticamente estável.

Os nomes de pessoas e endereços se prestam particularmente a este tipo de compressão.

Já foi constatado que usando um campo de tamanho fixo, para o armazenamento dos mesmos, estamos introduzindo em nossos arquivos um grande volume de espaço morto, este espaço não será considerado na aplicação que vamos desenvolver.

2. DESCRIÇÃO DO MÉTODO

Apresentamos neste capítulo um método para compactar nomes baseado na utilização de um código de 256 elementos (EBCDIC) dos quais 27 (vinte e sete) representam as letras e o caráter branco, 1 (um) representa todos os caracteres inválidos porventura encontrados nos nomes e 228 (256 - 28) representarão conjuntos de duas e três letras escolhidos como sendo os que proporcionam maior compressão em função de suas probabilidades de ocorrencia. [10]

Este método, baseado na teoria da informação, é o resultado

de uma longa pesquisa da probabilidade de ocorrencia de cada carater, seja isoladamente, seja em associação com outros caracteres, formando grupos.

Para representar nomes costuma-se usar o código EBCDIC, este código representa letras, números e caracteres especiais atribuindo a cada um uma determinada configuração de um byte. Um byte permite representar 256 configurações diferentes ficando muitas delas sem uso para esse fim. [13]

Para os nomes que usam, como vimos, apenas 28 configurações 228 restam inaproveitadas. Podemos ver portanto que codificar cada nome em tantos bytes quantos são os seus caracteres torna-se pouco econômico.

Os 28 símbolos necessários para representar nomes podem ser codificados com um código de cinco bits (32 configurações). Com isso podemos economizar três bits por carater obtendo uma compressão percentual de 3/8*100 = 37,5% sem considerar os 3,5 bits, em média, necessários para completar o último byte.

Como podemos observar esta compressão é menor do que a obtida usando códigos variáveis, ou uma, codificação condicional.

Os resultados da compressão, usando códigos variáveis, foram obtidos levando em consideração a frequencia com que cada letra é usada.

A probabilidade de ocorrencia de uma letra em uma determinada posição é influenciada pelas letras que a cercam (codificação condicional).

Por exemplo depois de um 'A' a probabilidade de ocorrencia de outro A cai a menos de 0,1%; depois de um Q a probabilidade de ocorrencia de um U é de 98%. Assim se codificarmos também os grupos mais frequentes, estamos nos aproveitando dessas particularidades.

3. CRITÉRIOS UTILIZADOS PARA A ESCOLHA DOS GRUPOS:

3.1. DEFINIÇÕES IMPORTANTES

a) Tendo em vista que cada grupo será codificado em um byte definimos como sendo o Poder de Compressão de um Grupo (PC) como sendo:

$$PC = \emptyset [G_i] * (C [G_i] - 1)$$

onde $\emptyset [G_i]$ é o número de ocorrencias do grupo G_i e $C [G_i]$ é o comprimento do grupo G_i expresso em bytes.

Se por exemplo o grupo 'ABC' ocorreu 300 vezes num determinado texto, ao substituirmos o grupo por um byte teremos um poder de compressão $PC_{ABC} = 600$ bytes.

Isto significa que o poder de compressão de um grupo é a economia de bytes obtida substituindo aquele grupo por um byte.

b) A escolha de um grupo de duas letras L1, L2 altera o número de ocorrencias de outros grupos de duas letras no mesmo conjunto.

Designando por $\emptyset$ (Lm, Ln) o número de ocorrencias do grupo Lm, Ln e por $\emptyset$ (Lm, Ln, Lo) o do grupo Lm, Ln, Lo; a escolha do grupo L1, L2 altera o número de ocorrencias de outros grupos de duas letras da seguinte forma:

$$\emptyset(Ln,L1) = \emptyset(Ln,L1) - \emptyset(Ln,L1,L2)$$

$$\emptyset(L2,Ln) = \emptyset(L2,Ln) - \emptyset(L1,L2,Ln)$$

para Ln = *, $\not{b}$, A, B, ..., Z [10]

Por exemplo, se escolhermos o grupo "JO" (po ser um grupo muito frequente), para ser compresso, e sabemos que o grupo "JOS" ocorreu 300 vezes e o grupo "OS" 4000, subtrairemos 300 do número de ocorrencias de "OS" pois sabemos que a letra "O" de "OS" será codificada 300 vezes juntamente a "J". Este conceito é muito importante para efeito de comparação dos grupos.

EXEMPLO:

$\emptyset$(JO) = 5000
$\emptyset$(JOS) = 300
$\emptyset$(OS) = 4000

se escolhermos "JO"
$\emptyset$(OS) = 4000 - 300

Uma vez feitas estas considerações podemos passar a escolher o número de grupos de três e de dois caracteres que proporcionem a maior compressão.

3.2. DETERMINAÇÃO DOS CONJUNTOS DE CARACTERES E DAS FREQUENCIAS.

O processo que vamos propor neste parágrafo para a determinação dos conjuntos de caracteres e de seu número de ocorrencias, é um processo iterativo, que necessita de um razoável tamanho de memória e que se torna demorado, mas que proporciona os resultados desejados [10]

O processo é constituido das seguintes etapas:

a) Contagem do número de ocorrencias de cada grupo de três letras e escolha do grupo mais frequente.

b) Exclusão do grupo mais frequente de tres caracteres nos nomes da amostra. Um carater especial que funciona como indicador é gravado no lugar em que foi retirado o grupo.

c) Contagem do número de ocorrencias de cada grupo de duas e tres letras (na amostra alterada no passo - b). Os grupos e frequencias são armazenados em tabelas.

d) Escolha dos grupos de duas letras, com o maior número de ocorrencias, que faltam para completar os 228 códigos.
Esta escolha deverá ser feita levando em consideração o item (b) do parágrafo 3.1.

e) Cálculo do poder de compressão dos grupos escolhidos por meio das fórmulas do item (a) do parágrafo 3.1.

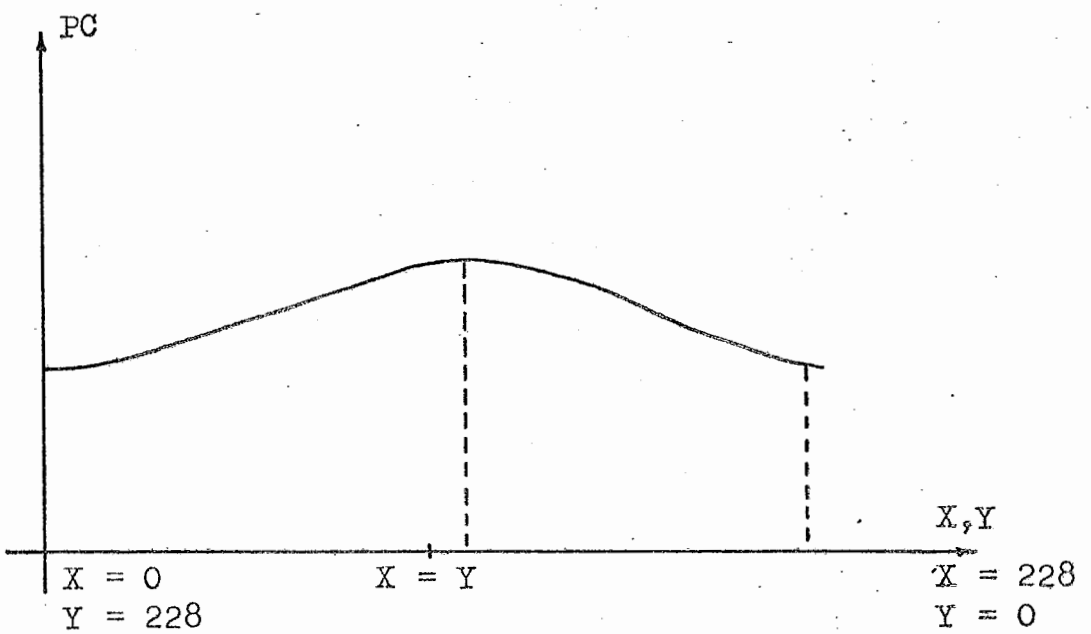
f) A amostra é novamente submetida ao passo - b.

O processo é repetido até se verificar uma diminuição considerável no poder de compressão.

Em Função dos poderes de compressão obtidos podemos escolher os X grupos de tres letras e os Y grupos de duas letras (tal que X + Y = 228) que correspondem ao maior poder de compressão.

Na figura que segue representamos uma curva teórica do poder de compressão em função da variação de X e Y.

Propomos desenvolver um método que permita selecionar conjuntos de caracteres maiores, fonemas por exemplo. Este método não foi desenvolvido por envolver um aspecto linquístico que não faz parte dos objetivos deste trabalho. [11]



4. ROTINAS DE COMPRESSÃO E DE DESCOMPRESSÃO POR CONJUNTO DE CARACTERES.

4.1. ROTINA DE COMPRESSÃO:

Uma vez determinado o conjunto de 228 grupos formados de dois e tres caracteres podemos desenvolver a rotina de compressão.

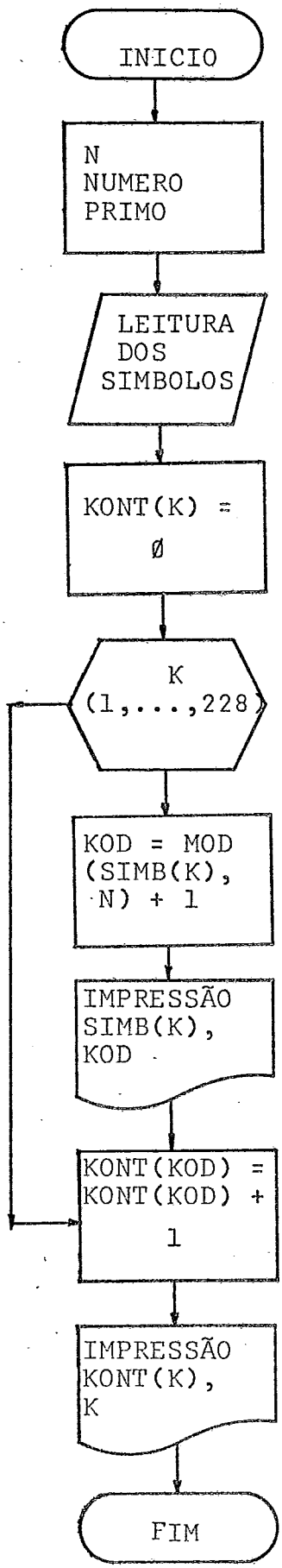
Como vimos um dos problemas da compressão por agrupamento de caracteres é a comparação entre os dados e as tabelas dos grupos escolhidos. Para reduzir as máximo o tempo de pesquisa desenvolveremos uma tabela HASH.

Para podermos construir a tabela devemos calcular o número de colisões entre os grupos, isto é o número de grupos que geram o mesmo código, para isto usaremos o algorítmo descrito a seguir:

H1 - Escolhe-se um número N que seja o número primo mais próximo a 228 e a este superior, no caso 229.
L é o número de grupos no caso 228
KONT é o contador de colisões
SIMB(K) representa um grupo K
MOD(M,N) = M - (M/N) N, isto é o resto da divisão de M por N

H2 - Leitura dos símbolos em uma tabela
LER SIMB(L)
para L = 1,2,...,228

FUNÇÃO: MOD (M,N) ≡ M - (M/N) * N
ponto fixo



N TAMANHO DA TABELA

NESTE CASO N = 229

SIMB(L)

L = 1,2,...,228

K = 1,2,...,N

CONTADORES DE COLISÕES ZERADOS

GERAÇÃO

DOS

CODIGOS

K = 1,2,...,N

IMPRESSÃO DOS ENDEREÇOS DA TABELA

E Nº DE COLISÕES PARA CADA ENDEREÇO

FIGURA - 4

H3 - Geração do código para cada grupo

para K = 1,2,...,228

CÓDIGO = MOD(SIMB(K),N) + 1

KONT(CÓDIGO) = KONT(CÓDIGO) + 1

IMPRIMIR SIMB(K),CÓDIGO

H4 - IMPRIMIR K,KONT(K)

para K = 1,2,3,...,229

Uma vez determinado o número de colisões e os grupos que os geraram devemos construir a tabela da seguinte maneira:

Sabendo que S é o número de colisões escolhemos os L - S grupos com maior número de ocorrencias e os colocamos nas respectivas entradas da tabela em função de seus códigos.

Sobrarão assim N - (L - S) entradas vazias que servirão para resolver as S colisões. Quando há colisão a primeira entrada será ocupada pelo grupo mais frequente dos que colidiram.

A tabela terá as seguintes características:

Cada entrada será dividida em tres campos;

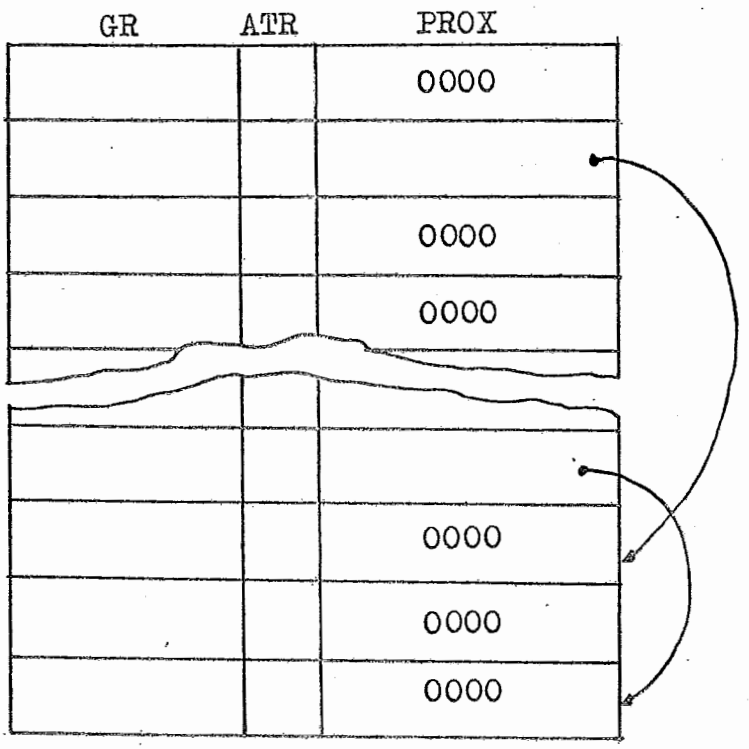
GR - conterá o grupo de duas ou tres letras (alinhado para direita quando for de duas letras)

ATR - conterá o atributo do grupo correspondente, no

caso a configuração de um byte correspondente ao grupo

PROX - indicará qual a próxima entrada a ser consultada no caso de colisão.

Se PROX é igual a 'Ø' não há colisão para aquele símbolo.



Daremos a seguir um algorítmo para consultar a tabela.

C1 - Escolha do grupo na esquerda de entrada
M = GRUPO, K = MOD (M/N)

C2 - Se GR(K) = GRUPO compactar e ir para C1,
Se PROX(K) = ∅ o grupo escolhido não têm compressão, mover o grupo para a saída e ir para C1.
K = PROX(K) ir para C2;

OBS.: O grupo é alinhado a direita para calcularmos o módulo.

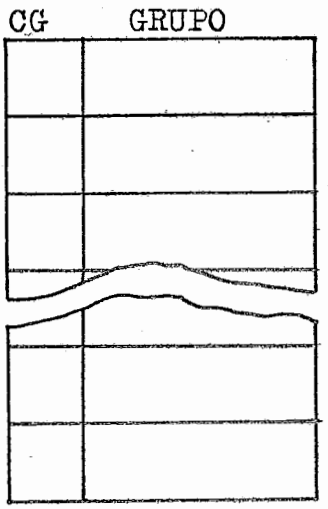
4.2. A ROTINA DE DESCOMPRESSÃO

A rotina de descompressão não apresenta problemas e é bastante rápida e eficiente.

A rotina consulta uma tabela de 256 entradas cada entrada formada dos seguintes campos:

CG - comprimento do grupo ou letra
GR - grupo que pode ser de uma, duas ou três partes.

Para cada byte lido na sequencia compressa obtem-se o endereço do grupo correspondente, que é substituído na sequência de saída; o comprimento do grupo é necessário para mover para a sequência de saída o número certo de bytes e para endereçar a posição em que será movido o próximo grupo.



5. RENDIMENTOS DO MÉTODO:

Pela complexidade do método e pela grande utilização de computador e de memória que seriam necessárias para a escolha dos grupos que proporcionam a maior compressão não implantei o método. No entanto podemos citar resultados obtidos com este método numa pesquisa desenvolvida no "SERPRO" tomando como amostra um conjunto de 18.000 nomes do Estado de São Paulo, estado em que nomes das mais diversas origem podem ser encontrados.

Os grupos escolhidos foram os apresentados na tabela que segue. A tabela consta de 113 grupos de tres letras e 115 de duas letras que com as 26 letras, o branco e o indicador de caracteres especiais completam as 256 configurações dos códigos.

A compressão obtida com este método foi de aproximadamente 51% sobre a parte útil dos nomes, considerando que a parte útil corresponde em média a 20 caracteres.

OS GRUPOS DE LETRAS ESCOLHIDAS FORAM:

bAL	AM	CAR	EIR	GO	IX	MA	ONE	RE	TO
bB	AND	CA	ELb	GUI	Jb	MEN	ONI	RIB	TTI
bCA	ANG	CE	ELI	GUS	JA	ME	ON	RI	Ub
bCO	ANI	CHA	ELL	GU	JOA	MIL	ORG	ROb	UL
bF	ANO	CHI	ELS	Hb	JOS	MIN	OR	ROS	UNI
bGO	ANT	CH	ENT	HA	JO	MI	OSb	RO	UN
bLE	AN	CIS	EN	HER	KA	MO	OS	RU	UR
bLO	AOb	CI	ERb	HE	KI	MP	Pb	Sb	US
bMA	ARA	CK	ERI	HI	KO	NAb	PAU	SCH	VAL
bMO	ARD	COb	ER	HO	Lb	NAL	PA	SER	VA
bNE	ARI	CO	ESb	HU	LAU	NA	PED	SE	VE
bPA	ART	CU	ES	Ib	LA	NE	PER	SI	VIC
bRO	AR	Db	ET	IAb	LE	NHA	PE	SOU	VI
bSA	ASb	DA	FER	IAN	LI	NI	PIN	SO	WAL
bSI	ASS	DE	FIL	IDA	LOd	NOb	PI	STA	WA
bVI	AST	DI	FI	IGU	LO	NO	PO	STE	Yb
Ab	AU	DOb	FON	IL	LUI	NT	QUE	Tb	Zb
AB	BAR	DO	FRE	IMA	LU	ObA	QUI	TA	ZA
AC	BA	DR	Gb	INI	LVA	ObB	Rb	TE	ZE
ADE	BEN	DU	GA	INO	LV	Ob	RAb	TIN	ZI
ALE	BER	Eb	GER	IN	Mb	OLI	RAN	TI	ZZ
ALV	BO	EDI	GE	IOb	MAN	OL	RA	TOb	
AL	Cb	ED	GI	IS	MAR	ONb	REN	TOR	

"Ø" Indica Espaço.

6. COMPRESSÃO POR SUBSTITUIÇÃO DE PALAVRAS

6.1. DESCRIÇÃO DO MÉTODO

Estendendo a idéia de compressão por substituição de conjuntos de dois e três caracteres fizemos uma pesquisa para compressão usando como conjunto, palavras (no nosso caso nomes).

Desenvolvemos um programa que lesse o arquivo de nomes gerando para cada nome tantos registros quantos fossem os nomes elementares que o compusessem, colocando na frente de cada registro o tamanho correspondente.

Por exemplo: o nome

JOSÉ DA SILVA OLIVEIRA

Daria origem aos seguintes registros:

04	JOSE

02	DA

05	SILVA

08	OLIVEIRA

Os registros assim gerados são classificados em ordem alfabética e de tamanho de maneira que os nomes iguais fiquem juntos. Um outro programa lê os registros classificados e calcula o número de ocorrencias de cada nome.

Nº	CN	NOME	PC(I)
1	07	ANTONIO	2578
2	02	DE	5840
3	07	ALBERTO	5789
4	07	ARMANDO	5096
5	07	ALFREDO	4389
6	08	OLIVEIRA	4144
7	05	SILVA	4120
8	08	FERREIRA	3632
9	07	PEREIRA	3073
10	09	RODRIGUES	2979
11	06	SANTOS	2892
12	07	ARNALDO	2709
13	06	ALVARO	2664
14	09	ALEXANDRE	2610
15	09	FERNANDES	2340
16	06	ANGELO	2292
17	05	SOUZA	2275
18	02	DA	2272
19	08	CARVALHO	2224
216	06	TOLEDO	0228
217	06	CABRAL	0228
218	04	JOÃO	0228
219	12	VASCONCELLOS	0228
220	08	LOURENCO	0224
221	08	LOUREIRO	0224
222	08	FERNANDO	0224
223	06	ALTINO	0222
224	05	ANITA	0220
225	05	BRAGA	0220
226	05	MOTTA	0220
227	07	ABIGAIL	0217
228	07	ACCACIO	0217
			$\sum$ PC(I) = 217322

CN - Comprimento do nome.
PC(I) - Poder de compressão do nome.

Considerando que cada nome dos 228 mais frequentes seguido de um carater branco poderá ser codificado com um byte, multiplicando o número de ocorrencias pelo comprimento de cada nome, podemos obter o poder de compressão de cada um deles, esta informação passa a fazer parte do registro.

$(PC_{PALAVRA} = \emptyset$ (PALAVRA) * COMPRIMENTO DA PALAVRA)

Classificando finalmente os nomes por ordem decrescente de poder de compressão obtivemos uma lista dos nomes com o maior poder de compressão. A pesquisa foi feita numa amostra de 20.000 nomes e obtivemos os resultados da tabela anexa.

Como podemos observar o nº de ocorrencias cai muito rapidamente. Por exemplo: o 1º nome da lista "ANTONIO" ocorreu 4.654 vezes na amostra enquanto o 6º nome "SILVA" ocorreu apenas 824 vezes.

Escolhemos para nosso estudo os 228 nomes com maior poder de compressão para podermos substituí-los por 1 byte, deixando as 28 configurações restantes para representar, sem alterações, os demais nomes obtendo assim um código de 256 configurações.

Os 228 nomes escolhidos proporcionaram um poder de compressão de 217.322 bytes, considerando que 20.000 nomes ocupam 35 * 20.000 = 700.000 bytes e que a parte útil é de aproximadamente 60% isto é, 420.000 bytes a

compressão obtida será de $\frac{217.568}{420.000} = 51\%$

6.2. ROTINAS DE COMPRESSÃO E DESCOMPRESSÃO PARA PALAVRAS

Para implementar as rotinas de compressão e descompressão por substituição de nomes podemos usar os conceitos expostos na descrição de compressão por substituição de grupos.

As tabelas das rotinas ficarão maiores das usadas para grupos.

Considerando que o maior nome tem 12 posições teremos uma tabela de aproximadamente 3.200 bytes para a rotina de compressão e uma tabela de aproximadamente 2.400 bytes para a rotina de descompressão.

Se as duas rotinas formarem um único módulo pode ser utilizada uma única tabela.

COMPARAÇÃO

Comparando os dois métodos podemos observar que a compressão de palavras proporciona os mesmos resultados da compressão por grupos sendo muito mais fácil a escolha dos nomes que a dos grupos.

A rotina de compressão e descompressão de nomes no entanto necessitam de tabelas maiores.

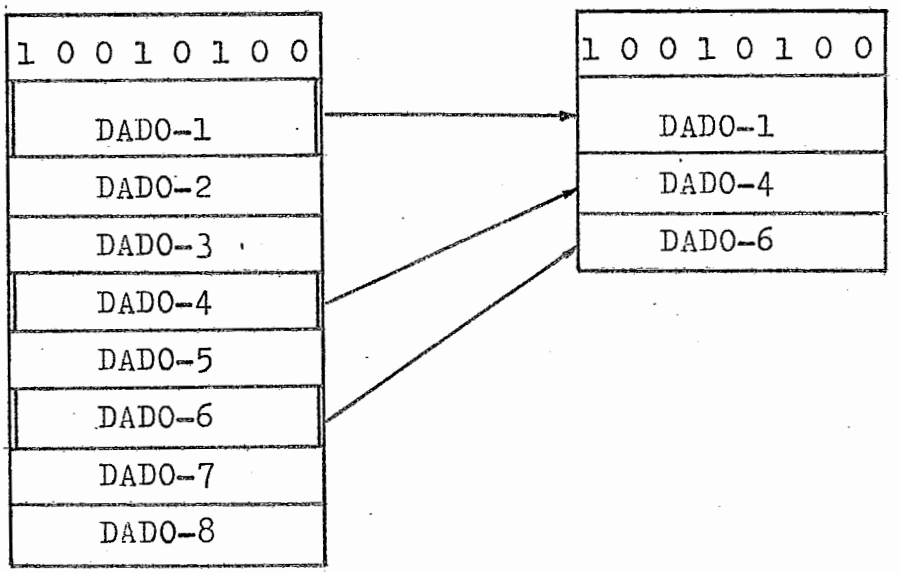
CAPÍTULO – 5

1. COMPRESSÃO USANDO MAPAS DE BITS

Este método consiste em fazer corresponder um bit a cada unidade da informação a ser processada, formando um mapa de bits.

Um bit "ligado", no mapa de bits, indica que o dado correspondente está presente, na sua posição relativa, dentro do registro e um bit "zero" indica a ausencia do dado.

Os campos que não contem informação são eliminados durante a compressão e reinseridos durante a descompressão. [6]



Este método pode ser aplicado utilizando como unidades de informação bytes, meias palavras, palavras inteiras ou outras unidades.

Entretanto, é conveniente usar este método utilizando como unidades de informação uma palavra; isto permite uma grande eficiencia do método, unidades menores como meias palavras ou caracteres, tendem a melhorar um pouco a percentagem de compressão, mas aumentam o custo em tempo de CPU, pois o número de iterações, necessárias para as funções de compressão e descompressão, aumenta consideravelmente. [14]

É possível, também, usar o método de compressão por mapa de bits, em registros em que os bits representam a presença ou ausencia de um campo, ou de unidades lógicas de tamanho diferente.

Neste caso a rotina de compressão e de descompressão devem ter acesso a uma tabela que identifique os campos dentro do registro.

Desenvolveremos este último tipo de utilização que nos parece mais geral e que permite maior elasticidade nas aplicações normais. [1]

2. REGISTROS SEGMENTADOS

2.1. CONCEITOS BÁSICOS

Um registro segmentado é um registro em que os componentes são identificados (simbolicamente) e agrupados

de acordo com certas relações lógicas. Os grupos identificados são chamados segmentos.

Um segmento é um conjunto de um ou mais campos adjacentes em um registro.

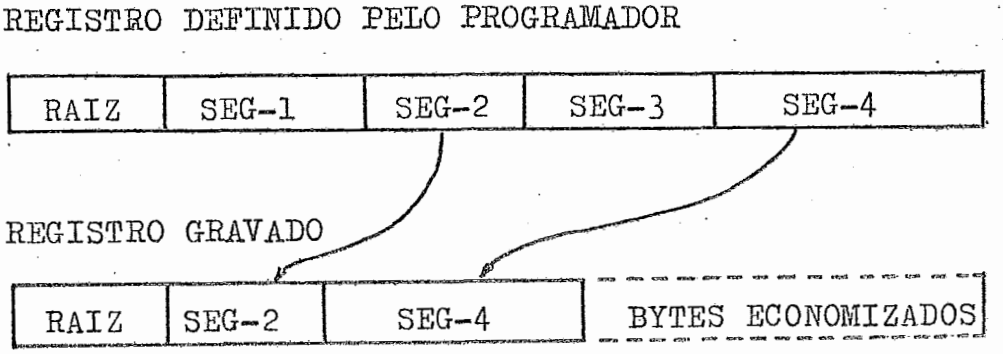
Alguns segmentos aparecem em todos os registros (um campo de controle ou de identificação por exemplo), enquanto outros segmentos aparecem apenas em alguns registros.

Como vimos, antes de usar este método de registros segmentados devemos definir a estrutura de cada segmento e de todo o registro, isto, como veremos, será feito através de uma tabela.

Usando o conceito de registros segmentados podemos reduzir o volume dos arquivos se usarmos registros variáveis.

O registro é compresso antes de ser gravado, assim o espaço necessário aos segmentos que não são utilizados pode ser economizado.

Este fato é expresso pela figura que segue.



2.2. FORMATOS DE REGISTROS SEGMENTADOS

Um registro segmentado deve conter um campo chamado raiz, que aparece no início do registro e deve conter:

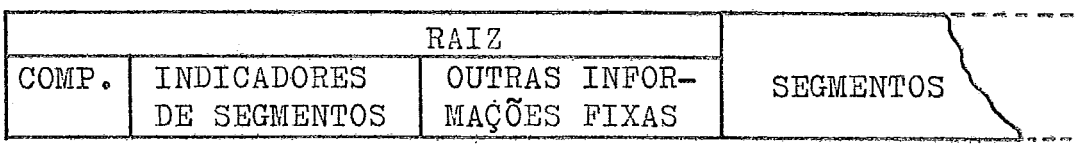
a) Comprimento do registro, é um campo que indica o comprimento do registro.

b) Indicadores de segmentos, que indicam a presença ou ausencia de cada segmento no registro.

Mais adiante veremos quais podem ser os tipos de indicadores de segmentos.

Além disso a raiz pode conter outras informações que possam ajudar no processamento dos registros (um campo de controle ou certas informações fixas por exemplo).

REGISTRO SEGMENTADO



A sequencia dos segmentos dentro de um registro lógico deve ser fixa, isto é, um segmento não pode mudar de posição em relação aos outros.

Cada segmento por sua vez pode ser fixo ou variável (a idéia de segmento variável permite uma maior compres-

são).

Se o segmento é variável então os primeiros bytes deverão conter o comprimento do segmento.

Cada segmento pode possuir suas características e estas podem ser diferentes das de outro segmento; cada segmento pode ter um comprimento.

2.3. INDICADORES DE SEGMENTOS

Os indicadores de segmentos são os meios pelos quais podemos determinar a presença ou ausencia de determinado segmento em um registro.

a) Os indicadores de segmentos devem estar na raiz do registro.

b) Deve existir um indicador para cada segmento e os indicadores devem de preferencia estar na mesma sequencia dos segmentos que representam.

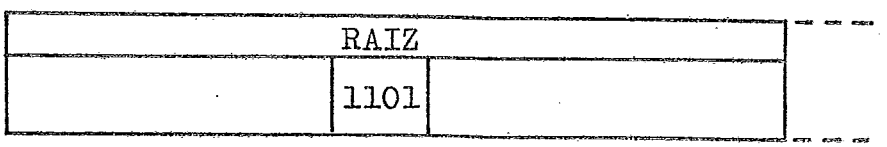
c) O usuário deve ter acesso aos indicadores e é sua responsabilidade controlá-los para retirar, modificar ou inserir segmentos.

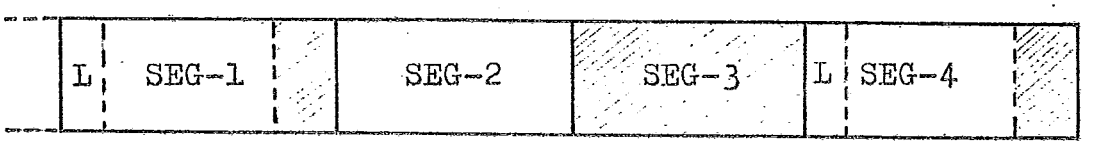
2.3.1. INDICADORES TIPO BIT

Neste caso cada segmento é representado por um bit.

Um byte de indicadores deve ser colocado na raiz para cada oito segmentos.

Se um bit é '1' (um), o segmento correspondente está presente, caso contrário, o segmento não está presente no registro.



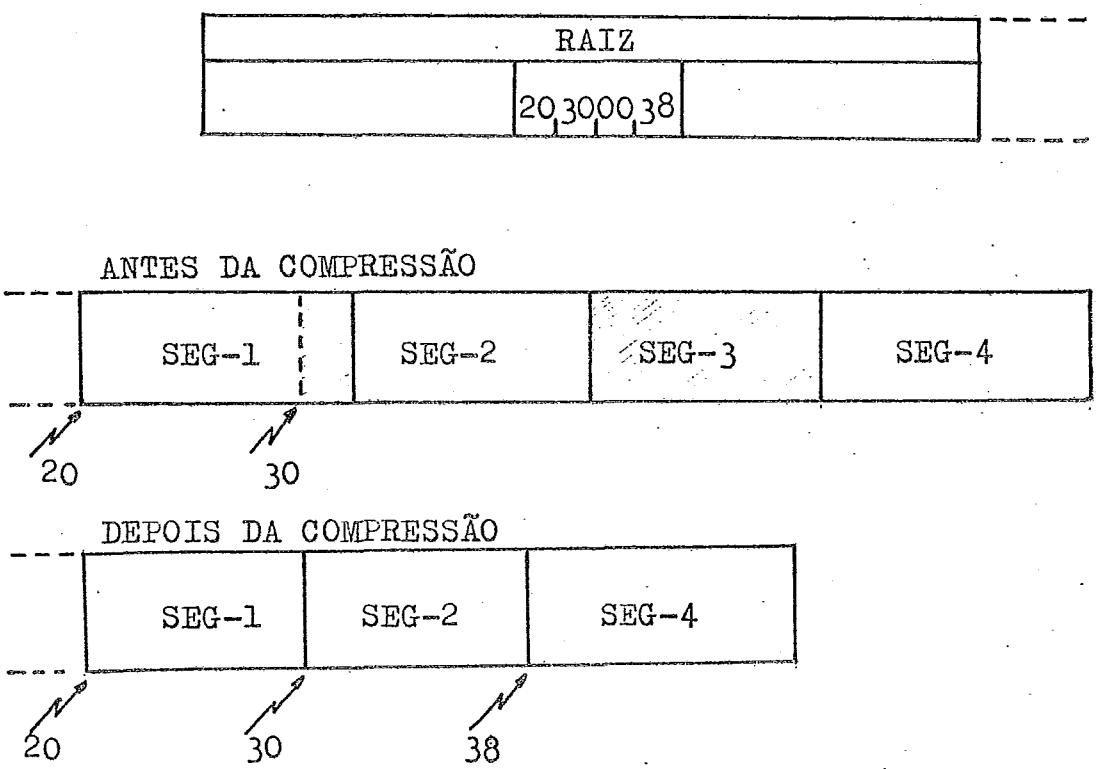


L - Indica o tamanho do segmento

2.3.2. INDICADORES TIPO DESLOCAMENTO

Com este tipo de indicadores, cada segmento é representado por um conjunto de bits.(Dois bytes normalmente).

Um valor zero deste conjunto indica que o segmento correspondente está ausente. Um valor diferente de zero indica que o segmento está presente e representa a posição relativa do segmento dentro do registro. Este valor pode representar, também, apenas o tamanho atual do segmento.



3. IMPLEMENTAÇÃO DAS ROTINAS DE COMPRESSÃO E DESCOMPRESSÃO PARA REGISTROS SEGMENTADOS

3.1. ROTINA DE COMPRESSÃO

Usando os conceitos expostos nos parágrafos anteriores desenvolvemos uma rotina de compressão para registros segmentados.

A rotina "COMPS" foi escrita em ASSEMBLER IBM/360 para ser utilizada por programas escritos em PL/1.

A rotina é chamada por três parâmetros

CALL COMPS (ENTRADA,SAIDA,TDR);

1) ENTRADA - Endereço do registro a ser compresso

2) SAIDA - Endereço de uma área de tamanho variável para a devolução do registro compresso.

3) TDR - Tabela de Descrição do Registro que contem todas as informações necessárias ao processamento do registro.

A TDR é definida (assim como a área de entrada e de saida) no programa principal e contem as seguintes informações:

1) COMPRIMENTO DA RAIZ - 2 bytes (binário

2) INÍCIO DO MAPA DE BITS - 2 bytes (binário), indica

a posição, em relação ao início do registro, em que inicia o mapa de bits, o mapa poderá conter apenas um byte, isto significa que o registro poderá ser formado de, no máximo, 8 segmentos (esta restrição pode facilmente ser modificada)

3) INFORMAÇÕES RELATIVAS AOS SEGMENTOS - para cada segmento serão dados dois bytes de informações, um byte para indicar se o segmento é variável ou fixo e outro byte indicando seu tamanho máximo. Mesmo que o segmento não exista seu campo de comprimento deve ser preenchido com zeros.

Exemplo de TDR:

1º byte	2º byte	
	200	Comprimento da Raiz
	73	Posição inicial do mapa
V	58	Info seg 1
F	73	" " 2
V	75	" " 3
F	255	" " 4
F	254	" " 5
	0	" " 6
	0	" " 7
	0	" " 8

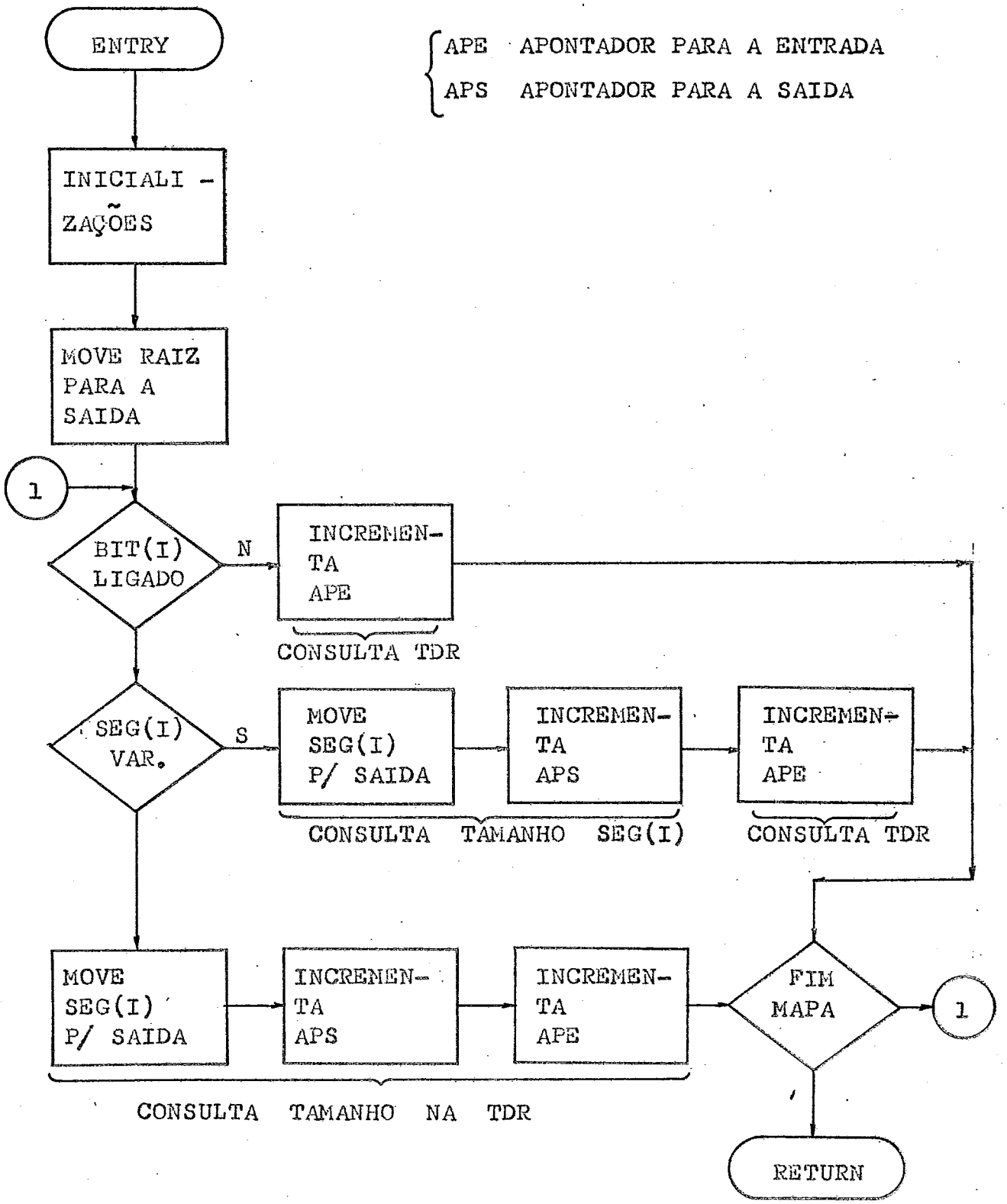
Os segmentos 6, 7 e 8 não existem, no entanto, o campo de comprimento foi preenchido com zeros.

Se o segmento é variável então o primeiro byte indica o seu tamanho atual, este tamanho é o real, isto é, não se inclui no tamanho o byte de controle, isto significa que um segmento variável, assim como o fixo, pode conter, no máximo, 255 bytes de informações.

A rotina de compressão recebe a área de entrada, move a raiz inalterada para a área de saida (sabendo o seu tamanho pela TDR), pesquisa o MAPA DE BITS contido na raiz, se o segmento estiver presente consulta a tabela para saber se este é variável ou fixo e move o segmento para a saída com o seu tamanho atual, caso o segmento não exista passa-se ao processamento de outro segmento.

Após ter terminado o processamento do registro e tê lo compresso na área de saída o comprimento desta é atualizada e o controle retorna ao programa principal.

Apresentamos a seguir o diagrama de blocos e a codificação em ASSEMBLER da rotina de compressão 'COMPS'.


ENTRY
APE APONTADOR PARA A ENTRADA
APS APONTADOR PARA A SAIDA
INICIALI - ZAÇÕES
MOVE RAIZ PARA A SAIDA
1
BIT(I) LIGADO
N
INCREMEN- TA APE
CONSULTA TDR
SEG(I) VAR.
S
MOVE SEG(I) P/ SAIDA
INCREMEN- TA APS
CONSULTA TAMANHO SEG(I)
INCREMEN- TA APE
CONSULTA TDR
MOVE SEG(I) P/ SAIDA
INCRENEN- TA APS
INCREMEN- TA APE
CONSULTA TAMANHO NA TDR
FIM MAPA
1
RETURN

```
ADDR2  STMT   SOURCE STATEMENT                                    FOIMAY7
          1 *
          2 ********************************************************************
          3 *         IMPLEMENTACAO DA ROTINA DE COMPRESSAO                    *
          4 *         PARA REGISTROS SEGMENTADOS                               *
          5 *         LUCIANO PIETRACCI                                        *
          6 ********************************************************************
          7 *
          8           PRINT NOGEN
          9 COMPS     CSECT
         10           SAVE  (14,12)
         13           LR    12,15
         14           USING COMPS,12
000B8    15           ST    13,SALVA+4
         16           LR    11,13
000B4    17           LA    13,SALVA
00008    18           ST    13,8(11)
```

```
ADDR2  STMT   SOURCE STATEMENT                                        FOLMA
         20 *
         21 ************************************************************
         22 *         INICIALIZACOES                                   *
         23 ************************************************************
         24 *
00000    25            L      4,0(0,1)        CARREGA O END. DO DV..ENTRADA
00000    26            L      2,0(0,4)        CARREGA EM R2 O END. DE ENTRADA
00004    27            L      5,4(0,1)        CARREGA O END. DO DV..SAIDA
00000    28            L      3,0(0,5)        CARREGA EM R3 O END. DE SAIDA
00008    29            L      4,8(0,1)        CARREGA O END. DO DV..TDR
00000    30            L      4,0(0,4)        CARREGA EM R4 O END. DA TDR
00000    31            LH     1,0(0,4)        CARREGA EM R1 O TAMANHO DA RAIZ
         32            BCTR   1,0             R1 = R1 - 1
00104    33            EX     1,MOVE          MOVE A RAIZ PARA SAIDA
00002    34            LH     15,2(0,4)       R15 CONTEM DESLOC. DO MAPA DE BITS
         35            BCTR   15,0            R15 = R15 - 1
         36            AR     15,2            R15 APONTA PARA O MAPA DE BITS
00080    37            LA     14,B'10000000'      MASCARA
00004    38            LA     4,4(0,4)        R4 APONTA P/ INFORMACOES DO SEG(1)
00001    39            LA     1,1(0,1)        R1 = R1 + 1
         40            AR     2,1             INCREMENTA APONTADOR DA ENTRADA
         41            LR     8,3             R8 APONTA P/ INICIO DA SAIDA
         42            AR     3,1             INCREMENTA APONTADOR DA SAIDA
         43            SR     7,7             R7 = 0
```

```
ADDR2  STMT   SOURCE STATEMENT                                        F01MAY7
         45 *
         46 ************************************************************************
         47 *          PROCESSAMENTO E COMPRESSAO                                  *
         48 ************************************************************************
         49 *
00100    50 TESTA     EX    14,TESTE       TESTA SE O SEG(I) ESTA PRESENTE
00076    51           BZ    NAOEX          NAO EXISTE SEG(I)
         52           CLI   0(4),C'V'      TESTA SE O SEG(I) E VARIAVEL
0007C    53           BE    MOVEVAR        PROCESSAR SEGMENTO VARIAVEL
00001    54 MOVEFIX   IC    1,1(0,4)       R1 CONTEM COMPRIMENTO DO SEG(I)
         55           BCTR  1,0            R1 = R1 - 1
00104    56           EX    1,MOVE         MOVE SEG(I) P/ SAIDA
00001    57           LA    1,1(0,1)       R1 = R1 + 1
         58           AR    2,1            INCREMENTA APONTADOR DA ENTRADA
0008E    59           B     INCRE1
         60 NAOEX     SR    1,1            ZERA R1
00088    61           B     INCREO
00000    62 MOVEVAR   IC    1,0(0,2)       R1 CONTEM COMPRIMENTO DO SEG(I)
00104    63           EX    1,MOVE         MOVE SEG(I) P/ SAIDA
00001    64           LA    1,1(0,1)       R1 = R1 + 1
00001    65 INCREO    IC    7,1(0,4)       R7 CONTEM COMPR. MAX. DO SEGI
         66           AR    2,7            INCREMENTA APONTADOR DA ENTRADA
         67 INCRE1    AR    3,1            INCREMENTA APONTADOR DA SAIDA
00002    68           LA    4,2(0,4)       R4 APONTA P/ INFORMACOES DO SEG(I)
00001    69           SRA   14,1           MUDA A MASCARA
00FC     70           CH    14,ZERO        SE MASCARA = 0 FIM
00052    71           BNE   TESTA
```

```
ADDR2  STMT   SOURCE STATEMENT                                    F01MAY

        73 *
        74 ***************************************************************
        75 *        FIM DA ROTINA, DEFINICAO DE AREAS E CONSTANTES       *
        76 ***************************************************************
        77 *
        78           SR    3,8             R3 CONTEM COMPRIMENTO DA SAIDA
00006   79           STH   3,6(0,5)        RESTAURA O DV..SAIDA
00088   80           L     13,SALVA+4      RESTAURA REGISTRADORES E
        81           RETURN (14,12),RC=0   DEVOLVE O CONTROLE
        85 *
        86 SALVA     DS    18F
        87 ZERO      DC    X'00000000'
        88 *
        89 TESTE     TM    0(15),0         TESTA SE O SEG(I) ESTA PRESENTE
        90 *
00000   91 MOVE      MVC   0(0,3),0(2)     MOVE P/ SAIDA
        92           END   COMPS
```

3.2. ROTINA DE DESCOMPRESSÃO

A rotina de descompressão de registros segmentados 'DCOMPS' foi escrita em ASSEMBLER IBM/360 para ser utilizada por programas escritos em PL/1.

Esta rotina usa a mesma filosofia utilizada na rotina de compressão, é chamada por três parâmetros

CALL DCOMPS (ENTRADA,SAIDA,TDR);

1) ENTRADA - Endereço do registro a ser descompresso (área variável)

2) SAIDA - Endereço de uma área fixa para devolução do registro descompresso

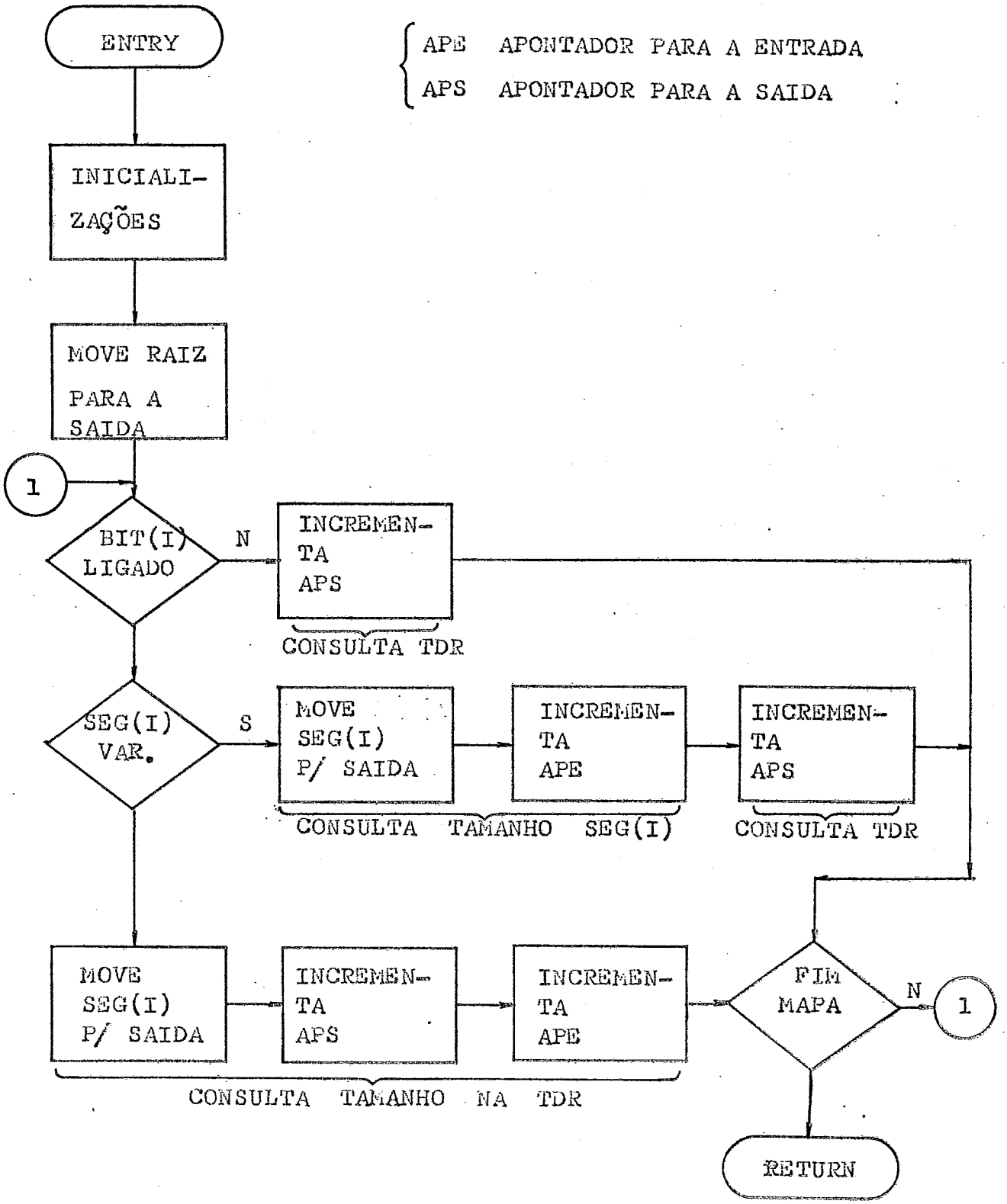
3) TDR - Endereço da tabela de descrição do registro.

A rotina de descompressão recebe a área de ENTRADA move a raíz inalterada para a área de saída (sabendo seu comprimento pela TDR; pesquisa o mapa de bits contido na raíz, se o segmento estiver presente consulta a tabela para saber se este é variável ou fixo e move o segmento para a saída com seu tamanho máximo, caso o segmento não exista a área de saída correspondente à quele segmento é deixada em branco e passa-se ao processamento de outro segmento.

Após ter terminado o processamento do registro e tê-lo

descompresso na área de saída o controle retorna ao programa principal.

Apresentamos a seguir o diagrama de blocos e a codificação em ASSEMBLER da rotina de descompressão 'DCOMPS'.


ENTRY
APE APONTADOR PARA A ENTRADA
APS APONTADOR PARA A SAIDA
INICIALI-
ZAÇÕES
MOVE RAIZ
PARA A
SAIDA
1
BIT(I)
LIGADO
N
INCREMEN-
TA
APS
CONSULTA TDR
SEG(I)
VAR.
S
MOVE
SEG(I)
P/ SAIDA
INCREMEN-
TA
APE
INCREMEN-
TA
APS
CONSULTA TAMANHO SEG(I)
CONSULTA TDR
MOVE
SEG(I)
P/ SAIDA
INCREMEN-
TA
APS
INCREMEN-
TA
APE
CONSULTA TAMANHO NA TDR
FIM
MAPA
N
1
RETURN

```
ADDR2  STMT   SOURCE STATEMENT                                    F01MAY7
         1 *
         2 ********************************************************************
         3 *         IMPLEMENTACAO DA ROTINA DE DESCOMPRESSAO                 *
         4 *         PARA REGISTROS SEGMENTADOS                               *
         5 *         LUCIANO PIETRACCI                                        *
         6 ********************************************************************
         7 *
         8           PRINT NOGEN
         9 DCOMPS    CSECT
        10           SAVE  (14,12)
        13           LR    12,15
        14           USING DCOMPS,12
00080   15           ST    13,SALVA+4
        16           LR    11,13
000AC   17           LA    13,SALVA
00008   18           ST    13,8(11)
```

```
ADDR2  STMT   SOURCE STATEMENT                                    F01MAYT
        20 *
        21 ************************************************************
        22 *         INICIALIZACOES                                   *
        23 ************************************************************
        24 *
00000   25            L     4,0(0,1)        CARREGA O END. DO DV..ENTRADA
00000   26            L     2,0(0,4)        CARREGA EM R2 O END. DE ENTRADA
00004   27            L     5,4(0,1)        CARREGA O END. DO DV..SAIDA
00000   28            L     3,0(0,5)        CARREGA EM R3 O END. DE SAIDA
00008   29            L     4,8(0,1)        CARREGA O END. DO DV..TDR
00000   30            L     4,0(0,4)        CARREGA EM R4 O END. DA TDR
00000   31            LH    1,0(0,4)        CARREGA EM R1 O TAMANHO DA RAIZ
        32            BCTR  1,0             R1 = R1 - 1
00FFC   33            EX    1,MOVE          MOVE A RAIZ PARA SAIDA
00002   34            LH    15,2(0,4)       R15 CONTEM DESLOC. DO MAPA DE BITS
        35            BCTR  15,0            R15 = R15 - 1
        36            AR    15,2            R15 APONTA PARA O MAPA DE BITS
00080   37            LA    14,B'10000000'      MASCARA
00004   38            LA    4,4(0,4)        R4 APONTA P/ INFORMACOES DO SEG(1)
00001   39            LA    1,1(0,1)        R1 = R1 + 1
        40            AR    2,1             INCREMENTA APONTADOR DE ENTRADA
        41            AR    3,1             INCREMENTA APONTADOR DE SAIDA
        42            SR    7,7             R7 = 0
```

```
)DR2  STMT   SOURCE STATEMENT                                          FO1MA

        44 *
        45 ************************************************************************
        46 *          PROCESSAMENTO E DESCOMPRESSAO                                *
        47 ************************************************************************
        48 *
00F8    49 TESTA     EX    14,TESTE       TESTA SE O SEG(I) ESTA PRESENTE
0074    50           BZ    NAOEX          NAO EXISTE SEG(I)
        51           CLI   0(4),C'V'      TESTA SE O SEG(I) E VARIAVEL
007A    52           BE    MOVEVAR        PROCESSAR SEGMENTO VARIAVEL
0001    53 MOVEFIX   IC    1,1(0,4)       R1 CONTEM COMPRIMENTO DO SEG(I)
        54           BCTR  1,0            R1 = R1 - 1
00FC    55           EX    1,MOVE         MOVE SEG(I) P/ SAIDA
0001    56           LA    1,1(0,1)       R1 = R1 + 1
        57           AR    3,1            INCREMENTA APONTADOR DA SAIDA
008C    58           B     INCRE1
        59 NAOEX     SR    1,1            ZERA R1
0086    60           B     INCREO
0000    61 MOVEVAR   IC    1,0(0,2)       R1 CONTEM COMPRIMENTO DO SEG(I)
00FC    62           EX    1,MOVE         MOVE SEG(I) P/ SAIDA
0001    63           LA    1,1(0,1)       R1 = R1 + 1
0001    64 INCREO    IC    7,1(0,4)       R7 CONTEM COMPR. MAX. DO SEG(I)
        65           AR    3,7            INCREMENTA APONTADOR DA SAIDA
        66 INCRE1    AR    2,1            INCREMENTA APONTADOR DA ENTRADA
0002    67           LA    4,2(0,4)       R4 APONTA P/ INFORMACOES DO SEG(I)
0001    68           SRA   14,1           MUDA MASCARA
00F4    69           CH    14,ZERO        SE MASCARA = 0 FIM
0050    70           BNE   TESTA
```

```
ADDR2  STMT   SOURCE STATEMENT                                    F01MAY7
         72 *
         73 ***************************************************************
         74 *        FIM DA ROTINA, DEFINICAO DE AREAS E CONSTANTES       *
         75 ***************************************************************
         76 *
00080    77           L      13,SALVA+4     RESTAURA REGISTRADORES E
         78           RETURN (14,12),RC=0   DEVOLVE O CONTROLE
         82 *
         83 SALVA     DS     18F
         84 ZERO      DC     X'00000000'
         85 *
         86 TESTE     TM     0(15),0        TESTA SE O SEG(I) ESTA PRESENTE
         87 *
00000    88 MOVE      MVC    0(0,3),0(2)    MOVE P/ SAIDA
         89           END    DCOMPS
```

4. OUTRAS CONSIDERAÇÕES

4.1. APRESENTAÇÃO DE ALGUMAS IDÉIAS PARA A SOFISTICAÇÃO DAS ROTINAS

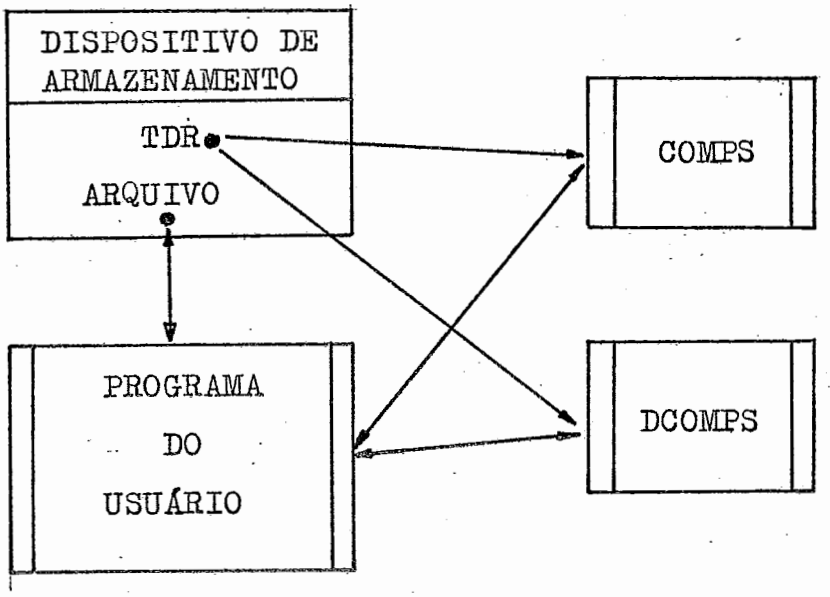
As rotinas implementadas nos parágrafos anteriores podem ser mais gerais. Em primeiro lugar poderíamos fazer com que estas pudessem ser utilizadas por outros programas que não aqueles escritos em PL/1.

Podemos aumentar o tamanho do mapa de bits para podermos aceitar qualquer número de segmentos, e ainda se o segmento for de tamanho variável podemos estender o indicador de seu comprimento a mais de um byte. Estas informações seriam dadas pela TDR e com pequenas modificações as rotinas poderiam, facilmente, manipular estas informações.

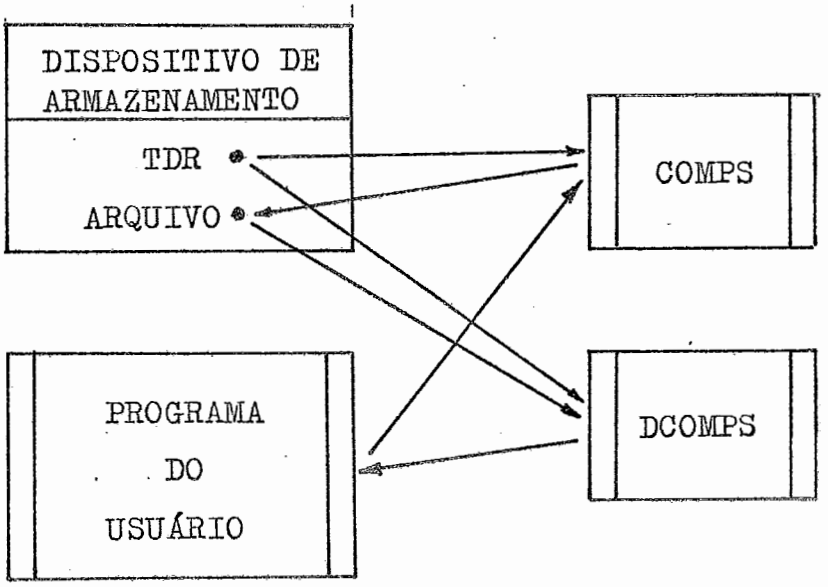
Estas modificações são fáceis e podem ser facilmente introduzidas nas rotinas.

Existem, porem, outras idéias das quais gostaríamos de falar.

As rotinas como foram desenhadas dependem das informações contidas na TDR isto faz com que todos os programas que processam um arquivo contenham uma TDR para descrever seus registros; seria portanto aconselhável que a tabela não pertencesse ao programa e sim ao arquivo ou a uma biblioteca que a rotina pudesse consultar em tempo de execução.



No caso de uso generalizado de arquivos compactados num CPD pode ser desenvolvida uma rotina mais geral que dotada de métodos de acesso permita liberar os programadores das funções de entrada e saída e do controle do mapa de bits ou do comprimento dos segmentos, permitindo assim que estes passem a manipular informações de tamanho virtualmente fixo, sem se preocuparem com a real apresentação dos dados no suporte de armazenamento ou na linha de transmissão.



No caso de multiprogramação as rotinas deverão ser reentrantes.

4.2. APLICAÇÕES E RENDIMENTOS DAS ROTINAS

O método de compressão por mapa de bits pode ser aplicado a vários tipos de informações desde que consideremos como unidade de informação uma palavra ou meia palavra ou até mesmo um byte, mas o conceito, extendido a campos maiores e a unidades lógicas de tamanhos diferentes, sugeriu uma nova filosofia no desenho dos arquivos.

O conceito de registro segmentado é um importante fator para a escolha deste método de compressão, que torna-se eficiente se os nossos arquivos possuírem uma estrutura apropriada.

Na maioria dos casos é possível e até vantajoso desenhar e desenvolver os nossos arquivos na forma segmentada.

Os rendimentos que podemos obter com esta rotina são muito variáveis dependendo da densidade do conteúdo das informações do arquivo.

As rotinas foram aplicadas a um cadastro de 'CLASSIFICAÇÃO DE MATERIAL' em que cada segmento correspondia a uma linha de descrição de 45 bytes; o número máximo era de sete segmentos.

Tendo em vista que cada item possuia em média 2,5 linhas de descrição obtivemos uma compressão de 50% con

siderando que a raiz continha 50 bytes.

Obtivemos resultados ainda melhores usando a rotina de compressão de brancos, isto é, além de retirarmos do registro os segmentos finais que não continham informação, usamos a compressão de brancos para os segmentos que continham parte da descrição, usando a técnica de segmentos variáveis obtivemos assim uma compressão de 70%.

Um exemplo de como podem ser usadas as rotinas está em apêndice.

O registro segmentado é o que mais se presta a aplicação de todas as técnicas de compressão apresentadas, se usarmos o conceito de segmentos variáveis. Além disso a arquitetura de arquivos com registros segmentados oferece outras vantagens que permitem uma grande elasticidade nas funções de inserção, de retirada e modificação de segmentos.

CAPÍTULO – 6

1. APLICAÇÕES DA COMPRESSÃO DE DADOS

Para estudarmos algumas aplicações da compressão de dados achamos interessante examinar algumas características gerais dos arquivos comerciais.

A figura abaixo apresenta um registro fictício de um arquivo

1	NÚMERO DE ESTOQUE	52253275432
2	NOME PADRONIZADO	PARAFUSO,ØSELETORØØØ
3	DESCRIÇÃO	P/CROSSBARØERICSSONØARF-102ØØØ ØØØØØØØØØØØØØØØØØØØØ
4	QUANTIDADE ORDENADA	00500
5	QUANTIDADE EM ESTOQUE	00290
6	VALOR UNITÁRIO	0000015
7	PONTO DE ENCOMENDA	00300
8	CÓDIGO DE SEGURANÇA	Ø
9	UNIDADE DE ESTOCAGEM	07
10	PEDIDOS NÃO ATENDIDOS	000

NOS CAMPOS 2, 3 deve ser reservado espaço para a informação máxima, mas a informação atual é menor que o espaço reservado.

NOS CAMPOS 8, 10 o espaço é ocupado por informações desnecessárias a este item de estoque.

NOS CAMPOS 4, 5, 6, 7, 9 grande quantidade de zeros é redundante.

Uma rotina de compressão deve explorar ao máximo estas características retirando os valores redundantes que aparecem com grande frequencia. Alem disso as informações devem ser ordenadas segundo uma certa lógica procurando agrupar dados alfabéticos de um lado e dados numéricos de outro, sem alterná-los, isto permite facilitar o uso de técnicas de compressão que podem aproveitar-se destas relações entre informações homogeneas em campos sucessivos.

2. CLASSIFICAÇÃO DAS TÉCNICAS DE COMPRESSÃO E CRITÉRIOS PARA A ESCOLHA DE UM MÉTODO DE COMPRESSÃO.

Podemos dividir as técnicas de compressão em duas classes:

- À primeira pertencem as rotinas de compressão que exploram as redundancias contidas nas informações ou a presença ou ausencia de certos campos.

  É o caso da compressão de sequencias de caracteres identicos ou por mapa de bits.

  Estes métodos são muito eficientes no que se refere aos tempos de execução e ocupam, normalmente, pequenas quantidades de memória.

- A segunda classe engloba as rotinas de compressão que se aproveitam das probabilidades estatísticas de ocorrencia das unidades que compõem as informações.

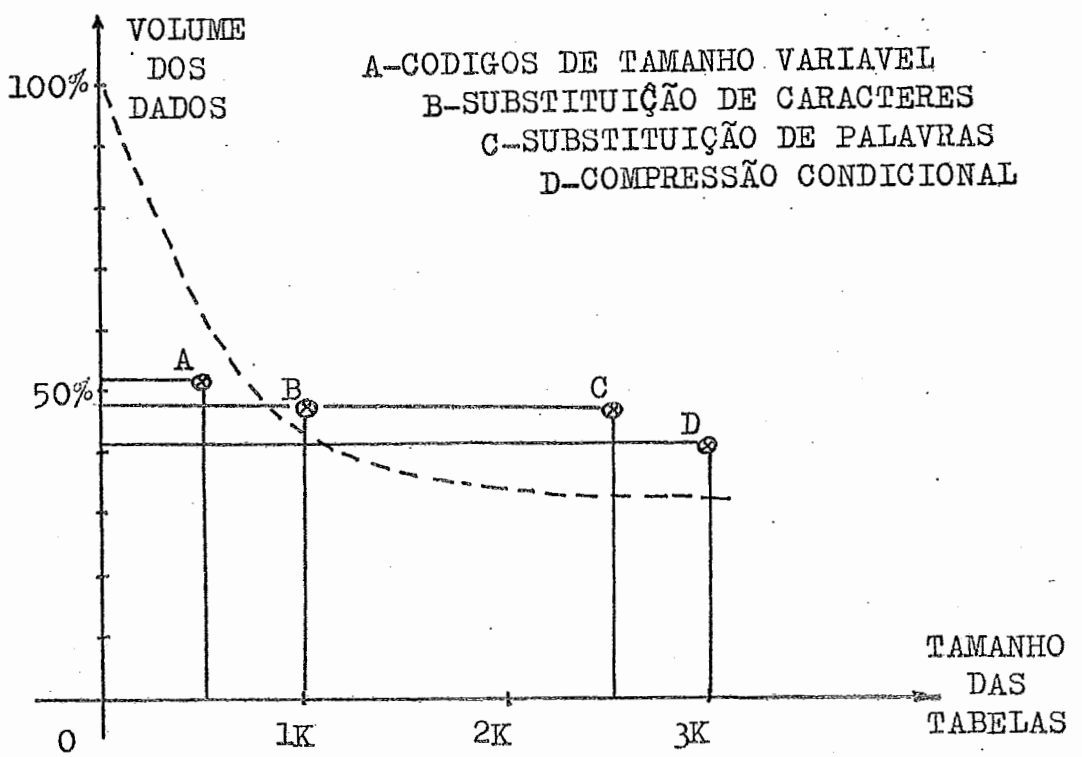
  É o caso da compressão por códigos de tamanho variável, por

agrupamento de caracteres ou de palavras.

Estes métodos, que pertencem à segunda classe, necessitam de tabelas que devem estar presentes na memória em tempo de compressão ou de descompressão, este fato pode limitar a escolha destes métodos quando a memória disponível é muito pequena (terminais por exemplo).

Como podemos observar a compressão por codificação condicional deu resultados melhores do que usando grupos variáveis para cada caráter considerado isoladamente, no entanto o volume das tabelas para o primeiro método é maior do que aquele usado no segundo método. Quando as informações não contem redundancias existe uma grande relação entre o tamanho das tabelas e a compressão obtida.

Esta relação pode ser expressa pelo gráfico abaixo.



O volume dos dados no suporte de armazenamento tende a diminuir ao aumentarmos o volume das tabelas presentes na memória em tempo de compressão existem no entanto outros fatores a serem considerados para otimizarmos a compressão.

Aumentarmos excessivamente o volume das tabelas evidentemente pode criar sérios problemas.

Para podermos analizar os critérios que devem ser adotados na escolha de um ou mais métodos de compressão, em função do tipo de informação que desejamos processar, faremos algumas considerações que podem ser importantes para a decisão.

- A compressão deve ser utilizada para arquivos em disco e para linhas de comunicação; a compressão para arquivos em fita não tem grande significado devido ao baixo custo do suporte de armazenamento, mas quando o arquivo é de múltiplos volumes traz grandes vantagens em termos de tempo relógio.

- A compressão é justificada em aplicações que envolvem grandes arquivos, quando a redução no custo e volume da memória on-line é substancial (acima de 30%).

- Nos computadores com uma CPU rápida, limitados pelos canais de entrada e saida, a compressão tende a ter um grande significado. Computadores de média e baixa velocidade de processamento limitam a escolha dos métodos de compressão. (Podemos observar que de uma geração para outra a velocidade da CPU aumenta muito mais do que a velocidade dos periféricos limitados por problemas de ordem mecânica).

- A quantidade de memória disponível também é um fator decisivo na escolha do método.

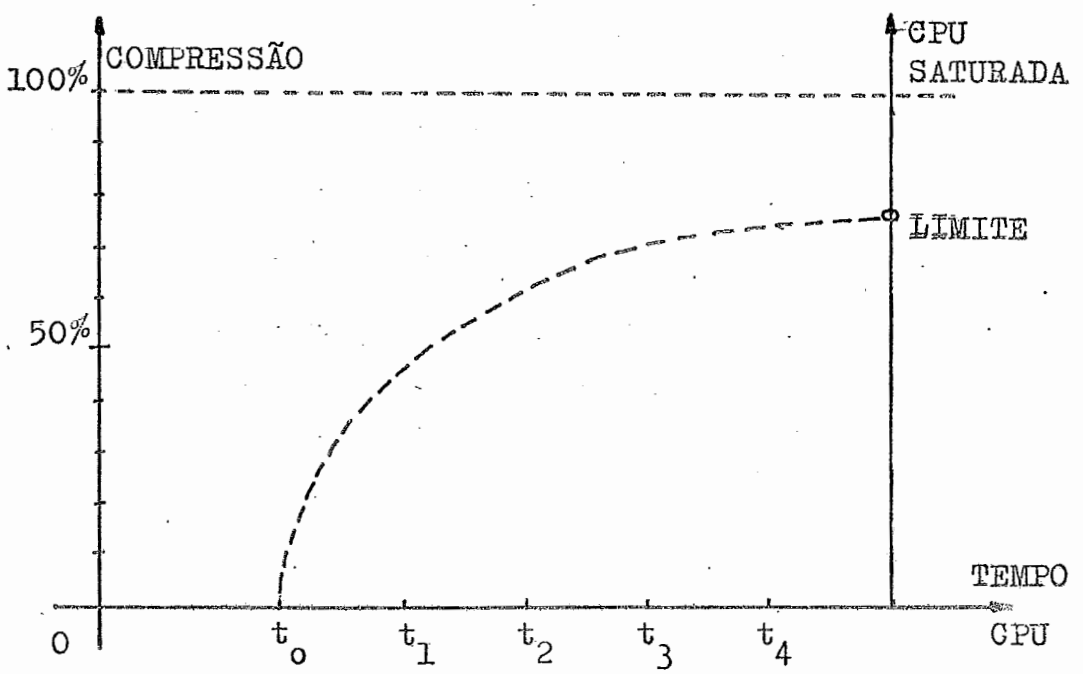
- A compressão tende a ter mais significado para arquivos de acesso randomico. O custo da compressão é mínimo comparado ao custo do método de acesso randomico.

- Para arquivos de organização sequencial quando é necessário descomprimir e comprimir todos os registros a compressão pode se tornar pouco eficiente e até anti-economica.

- A escolha de um ou mais métodos de compressão depende do nível de redundancias contidas nos arquivos e/ou de suas propriedades estatísticas.

- A relação entre a redução do volume de dados e o tempo de CPU deve ser uma das considerações preponderantes na escolha do algoritmo de compressão.

Esta relação pode ser representada pelo gráfico que segue:



Métodos para se alcançar altas percentagens de compressão podem se tornar pouco eficientes se considerarmos o custo de utilização da CPU.

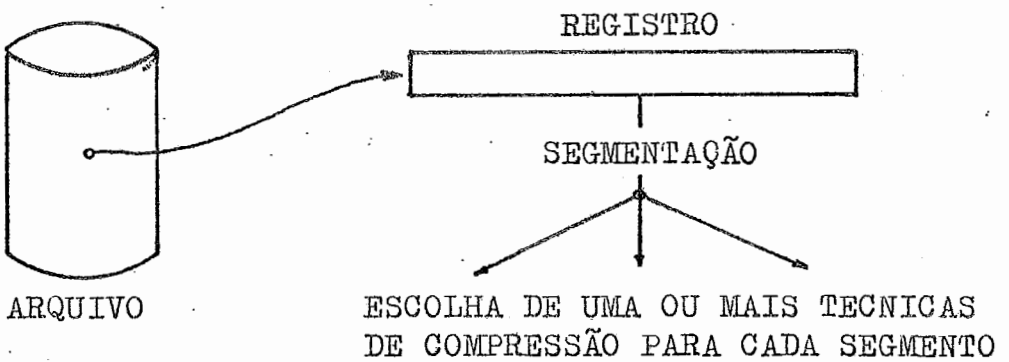
- Mais de uma técnica pode ser utilizada para um mesmo tipo de registro ou informação estes casos devem ser analizados separadamente.

Outros fatores podem ser decisivos na escolha das técnicas de compressão:

- A necessidade de arquivos facilmente legíveis assim como a utilização generalizada de arquivos (por empresas diversas por exemplo) podem se fatores que eliminam a possibilidade da compressão.

De outro lado o interesse em dificultar a divulgação indevida dos arquivos ou das informações pode ser um fator favorável para a escolha de um método de compressão.

- Como vimos os arquivos segmentados são os que mais se prestam para técnicas de compressão; o gráfico que segue representa qual o melhor enfoque que deveria ser dado ao desenhar se um arquivo para a aplicação de técnicas de compressão.



CONCLUSÃO

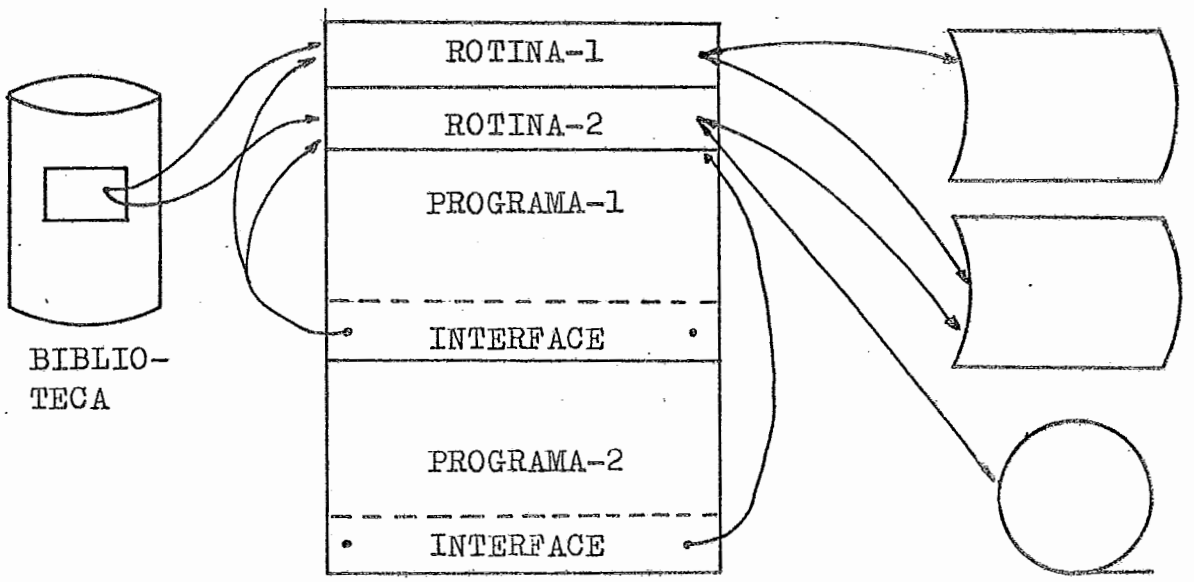
1. SUGESTÕES PARA A IMPLEMENTAÇÃO DE TÉCNICAS DE COMPRESSÃO NUM CPD

Levando em consideração as idéias e os critérios expostos no nosso trabalho, passamos a descrever algumas idéias para a implementação das técnicas de compressão num Centro de Processamento de Dados.

Duas coisas são desejáveis na implantação de técnicas de compressão: que estas sejam independentes do tipo de registro ou de informação, isto é, que permitam manipular qualquer tipo de dado; e que sejam transparentes para o programador.

As rotinas de compressão devem, de preferência, ser inseridas no Sistema Operacional, nos métodos de acesso, isto no entanto pode ser bastante complexo e nem sempre possível.

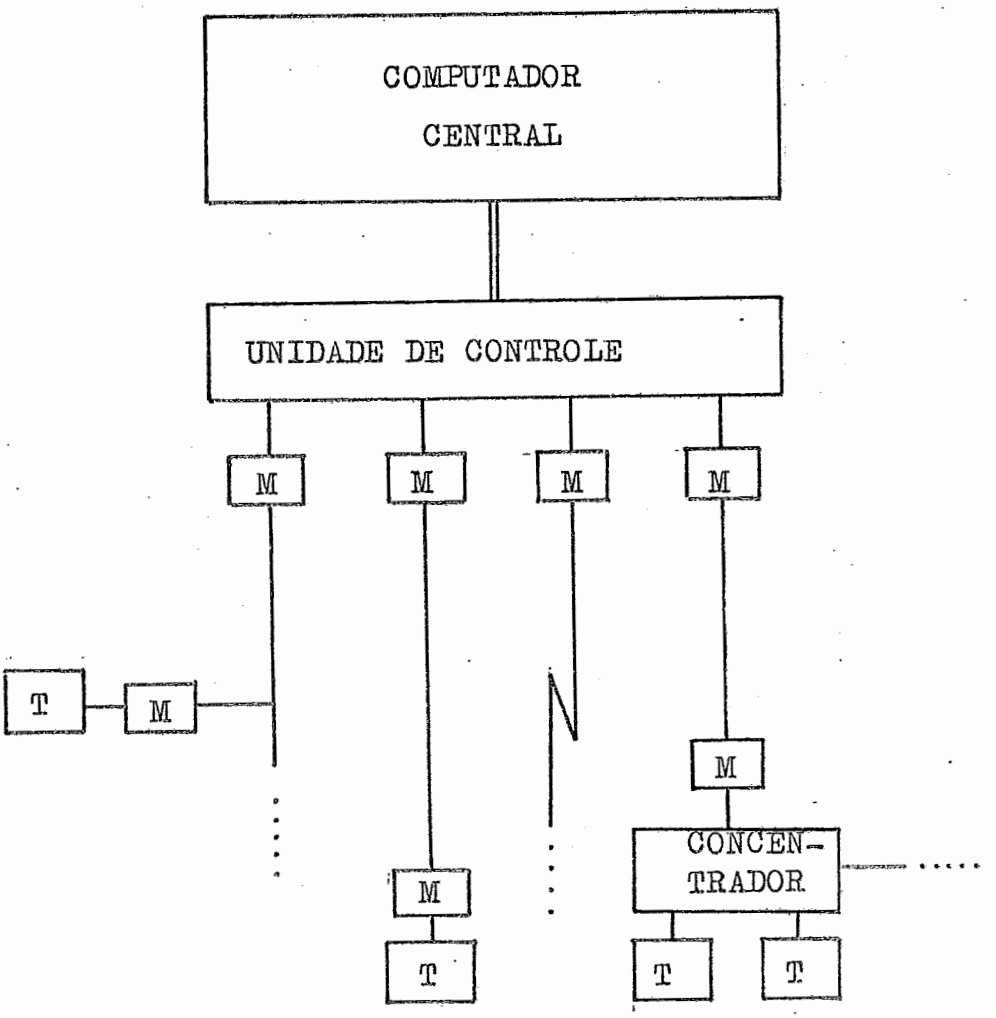
A solução mais simples, e portanto mais viável, é aquela de criar uma biblioteca de rotinas reentrantes que possam estar sempre disponíveis para serem facilmente utilizadas.



É importante também que exista uma documentação (que pode pertencer ao próprio arquivo) especificando as técnicas de compressão utilizadas para cada tipo de arquivo.

No teleprocessamento existem duas maneiras de utilizarmos a compressão de dados.

A primeira consiste em implantar o método de compressão no terminal quando este é programável; a segunda, e mais eficiente, consiste em implantar um ou mais métodos de compressão no concentrador de terminais (quando este existe).



2. AVALIAÇÃO DOS MÉTODOS

Concluindo o nosso trabalho apresentamos uma avaliação dos critérios que foram desenvolvidos.

Os valôres A, B, C (A > B > C) atribuidos como avaliação dos TEMPOS DE EXECUÇÃO dos métodos foram obtidos levando-se em consideração o número de instruções e de iterações necessárias para processar um registro.

MEMÓRIA	COMPRESSÂO	TEMPO DE EXECUÇÃO	MÉTODO
0,3 K	Até 76%	A	1
0,5 K	48%	B	2
3,0 K	56%	C	3
1,0 K	51%	B	4
2,5 K	51%	B	5
0,3 K	Até 70%	A	6

1) COMPRESSÃO DE SEQUÊNCIAS DE CARACTERES IGUAIS
2) COMPRESSÃO COM CÓDIGOS DE TAMANHO VARIÁVEL
3) COMPRESSÃO CONDICIONAL
4) COMPRESSÃO POR AGRUPAMENTO DE CARACTERES
5) COMPRESSÃO POR SUBSTITUIÇÃO DE PALAVRAS
6) COMPRESSÃO POR REGISTROS SEGMENTADOS.

o O o

No trabalho apresentamos algumas técnicas de compressão e como estas se aplicam no tratamento da informação; outras técnicas podem ser desenvolvidas ou algumas das apresentadas podem ser aperfeiçoadas.

Propomos assim que este trabalho seja continuado e que novas técnicas e novos métodos sempre mais eficientes sejam desenvolvidos.

BIBLIOGRAFIA

1. ANTONIO SERGIO SECCO FERREIRA.

UM INTERFACE DE ACESSO A REGISTROS SEGMENTADOS.
CTB-ACS - BIBLIOTECA TÉCNICA - DEPARTAMENTO DE SISTEMAS.
Este trabalho descreve a utilização de um módulo que permite o acesso a registros segmentados em processamento "BATCH".

2. BREMER, R.W.

DO IT BY THE NUMBERS - DIGITAL SHORTHAND.
COMMUNICATIONS OF THE ACM V.3 Nº 10 OCT 1960 pp 530.36
Apresenta algumas técnicas para representar numericamente palavras ou frases muito usadas; a decodificação é obtida utilizando um dicionário.

3. DYLAKOR COMPUTER SYSTEM, INC.

DYL - 255 COMPRESS.
11914 WASHINGTON BLVD. LOS ANGELES, CALIFORNIA 90066
Esta Companhia publicou um manual de usuário para a utilização do seu programa que é uma rotina de compressão e de descompressão de brancos e de zeros e que oferece bons resultados para arquivos tipo impressão.

4. GILBERT, E.N.; MOORE, E.F.

VARIABLE-LENGTH BINARY ENCODINGS.
THE BELL SYSTEM TECHNICAL JOURNAL V.38 JULY 1959 pp 933-67
Este artigo apresenta um tratamento teórico das propriedades de certos códigos variáveis. Descreve a propriedade do prefixo, da auto-sincronização, da ordem alfabética, etc. ...

5. HUFFMAN, D.A.

METHOD FOR CONSTRUCTION OF MINIMUN-REDUNDANCY CODES.
IRE. PROC. V.40 Nº 9 SEPT 1952 pp 1098-101
Desenvolve um método de construção de um código mínimo para um conjunto de mensagens composto de um número finito de unidades. Com este método o número de bits por mensagem e reduzido consideravelmente.

6. IBM.

SEGMENTED RECORDS.
CICS APPLICATION PROGRAMMER's REFERENS - SH 201047-0/1971.
Este manual descreve a utilização do registro segmentado e mostra como o "CICS" processa este tipo de registro.

7. IBM.

INTRODUCTION TO PROGRAMMING THE IBM/3270 - GC 27-26999-0.
Este manual descreve a programação para utilização do terminal IBM-3270.
São de certo interesse as técnicas de compressão utilizadas para a redução de carga nas linhas de comunicação dos terminais.

8. KORTMAN, C.M.

DATA COMPRESSION BY REDUNDANCY REDUCTION.
IEEE - SPECTRUM V.4 Nº 3 MAR 1967 pp 133-39.
Descreve técnicas para eliminar grande quantidade de redundancias nas cadeias de informações a serem transmitidas.

9. OLIVER, B.M.

EFFICIENT CODING
THE BELL SYSTEM TECHNICAL JOURNAL V.21 JULY 1952 pp 724-50
Alem de alguns aspectos da teoria da comunicação (especialmente no que se refere à relação entre transmissão e canais); são apresentados métodos para a construção de códigos variáveis e algumas das propriedades destes códigos.

10. PAULO MOTTA PIRES.

COMPACTAÇÃO DE NOMES.
SERPRO: ESTUDO -RN03. JULHO 1971 (Não publicado).
Neste estudo foi desenvolvida uma técnica de compressão para nomes usando as configurações do EBCDIC, não utilizadas, para representar grupos de dois e tres caracteres.
A compressão obtida com este método é de 50% sobre a parte útil dos nomes.

11. SCHWARTZ, E.S.; KLEIBOEMER, A.J.

LINGUAGE ELEMENT FOR COMPRESSION CODING.
INFORMATION AND CONTROL V.10 MAR 1967 pp 315-33.
São feitas considerações sobre elementos de m-palavras e restrições gramaticais; são apresentados códigos para representar conjuntos de palavras da linguagem fonte.

12. SHANNON, C.E.

A MATHEMATICAL THEORY OF COMMUNICATION.
THE BELL SYSTEM TECHNICAL JOURNAL V.37 JULY 1948 pp 379-423
Este artigo apresenta um rigoroso tratamento matemático da teoria da comunicação. São de grande interesse as fórmulas que relacionam a entropia com a probabilidade de ocorrencia das mensagens de uma fonte. São apresentados estudos sobre ruidos e canais.

13. SNYDERMAN, MARTIN; HUNT, BERNARD.

THE MYRIAD VIRTUES OF TEXT COMPACTION
DATAMATION 1-DEC 1970 pp 36-40
Este artigo descreve um sistema de compressão baseado no conjunto de caracteres do EBCDIC que são normalmente utilizados; 168 configurações, não utilizadas no EBCDIC, são reservadas para representar grupos de dois caracteres. É apresentada a rotina, em ASSEMBLER, que permite efetuar a compressão.

14. STEPHEN, S.H.; PAUL, J. KREUTZER.

DATA COMPRESSION FOR LARGE BUSINESS FILES.
DATAMATION 1-SEPT 1972 pp 62-66.
São apresentadas várias técnicas de compressão, descreve como estas se aplicam a arquivos comerciais, e os rendimentos de uma técnica para um arquivo em disco. São feitas comparações entre as técnicas.

15. WELLS, M.

FILE COMPRESSION USING VARIABLE LENGTH ENCODINGS.
THE COMPUTER JOURNAL V.15 Nº 4 pp 308-13.
Este artigo apresenta algumas técnicas de HARDWARE e de SOFTWARE para a implementação da compressão e descompressão usando códigos variáveis. É descrito o método de HUFFMAN para a construção de códigos variáveis.

APÊNDICE

```
VEL NEST
        /*****************************************************************/
        /*      APLICACOES DAS ROTINAS DE COMPRESSAO E                   */
        /*      DE DESCOMPRESSAO DE BRANCOS                              */
        /*****************************************************************/


        CDCOMP:   PROC OPTIONS (MAIN);

1                 DCL ENTRADAC CHAR(55),      /* ENTRADA COMPRESSAO    */
                      SAIDAC   CHAR(55) VAR; /* SAIDA   COMPRESSAO    */

1                 DCL ENTRADAD CHAR(55) VAR, /* ENTRADA DESCOMPRESSAO*/
                      SAIDAD   CHAR(55);     /* SAIDA   DESCOMPRESSAO*/

1                 ON ENDFILE (SYSIN) GOTO FIM;


1       LER:      GET EDIT (ENTRADAC) (A(55));
1                 GET SKIP;

1       /**/      CALL COMP (ENTRADAC,SAIDAC);                          /**/

1                 PUT SKIP DATA (ENTRADAC);
1                 PUT SKIP DATA (SAIDAC);


1                 ENTRADAD = SAIDAC;

1       /**/      CALL DCOMP (ENTRADAD,SAIDAD);                         /**/

1                 PUT SKIP DATA (ENTRADAD);
1                 PUT SKIP DATA (SAIDAD);


1                 GOTO LER;


1       FIM:      END CDCOMP;
```

VEL NEST

```
/*****************************************************************/
/*      IMPLEMENTACAO DO ALGORITMO DE HUFFMAN                    */
/*      ESTE ALGORITMO DETERMINA O CODIGO DE MENOR CUSTO         */
/*      PARA UM ALFABETO DO QUAL SAO CONHECIDAS AS PROBABI-      */
/*      LIDADES DE OCORRENCIA DE SEUS CARACTERES                 */
/*****************************************************************/

CODIFI:   PROC OPTIONS (MAIN);

1             DCL 1 NO BASED(P),
                    2 LLINK PTR,
                    2 LETRA CHAR(1),
                    2 PROB  PIC'(7)9',
                    2 RLINK PTR,
                    2 NEXT  PTR;

1             DCL 1 PILHA(28),
                    2 APOPI PTR,
                    2 CODPI CHAR(28),
                    2 KOMPI ;

1             DCL CODI CHAR(28) INIT(' '),
                  COD (28) CHAR(1) DEF CODI;

1             DCL 1 MINIMO,
                    2 MO  PTR,
                    2 M1  PTR,
                    2 MIN PIC'(7)9';

1             DCL (AP,TOPO,N,M) PIC'99';

1             DCL (Q0,Q1) PTR;
```

```
/*****************************************************************/
VEL NEST

        /*****************************************************************/
        /*       CONSTRUCAO DA FILA                                      */
        /*****************************************************************/

1                   GET LIST (N);
1                   M = N;

1                   DO I = 1 TO N;
1     1                ALLOCATE NO;
1     1                GET LIST (LETRA,PROB);
1     1                IF I = 1 THEN DO;
1     2                              Q0,Q1 = P;
1     2                              LLINK,RLINK = NULL;
1     2                              END;
1     1                         ELSE DO;
1     2                              Q1 -> NEXT = P;
1     2                              Q1 = P;
1     2                              LLINK,RLINK = NULL;
1     2                              END;
1     1     /**/           PUT SKIP LIST (LETRA,PROB);
1     1                END;
1                   Q1 -> NEXT = Q0;
```

```
VEL NEST

          /*****************************************************************/
          /*      CONSTRUCAO DA ARVORE BINARIA                             */
          /*****************************************************************/

1                   DO I = 1 TO N-2;
1     1                CALL MENOR;
1     1                ALLOCATE NO;
1     1                LLINK = MO;                 /* LLINK APONTA P/MENOR*/
1     1                PROB = MIN;

1     1                M1 -> NEXT = MO -> NEXT; /************************/
1     1                QO = MO -> NEXT;          /* RETIRAR O MENOR NO  */
1     1                Q1 = M1;                  /************************/

1     1                M = M - 1;
1     1                CALL MENOR;
1     1                RLINK = MO;                 /* RLINK APONTA P/MENOR*/
1     1                PROB = PROB + MIN;

1     1                M1 -> NEXT = MO -> NEXT; /************************/
1     1                QO = P;                   /* RETIRAR O MENOR NO  */
1     1                Q1 = M1;                  /* E INSERIR O NO RE-  */
1     1                NEXT = M1 -> NEXT;        /* SULTANTE            */
1     1                M1 -> NEXT = P;           /************************/

1     1             END;


1                   ALLOCATE NO;
1                   LLINK = QO;
1                   RLINK = Q1;
1                   PROB = QO -> PROB + Q1 -> PROB;
1          /**/     PUT SKIP LIST(PROB);
```

```
/*****************************************************************/
EVEL NEST

        /*****************************************************************/
        /*      PERCORRER A ARVORE EM 'POSTORDER'                        */
        /*****************************************************************/

1                   K,AP,TOPO = 0;
1       /**/        PUT PAGE;

1       T1:         IF P = NULL THEN GOTO T2;
1                      TOPO = TOPO + 1;
1                      APOPI(TOPO) = P;
1                      CODPI(TOPO) = CODI;
1                      KOMPI(TOPO) = K;
1                      P = LLINK;
1                      K = K + 1;
1                      COD(K) = '0';
1                      GOTO T1;

1       T2:         IF TOPO = AP THEN GOTO FIM;
1                      P = APOPI(TOPO);
1                      CODI = CODPI(TOPO);
1                      K = KOMPI(TOPO);
1                      TOPO = TOPO - 1;
1                        IF LLINK = NULL  &  RLINK = NULL THEN
1                           PUT SKIP LIST (LETRA,PROB,CODI);
1                      P = RLINK;
1                      K = K + 1;
1                      COD(K) = '1';
1                      GOTO T1;
```

```
/*************************************************************/

VEL NEST

          /*************************************************************/
          /*      ROTINA DE CALCULO DA MENOR PROBABILIDADE             */
          /*      ESTA ROTINA PERCORRE A FILA PROCURANDO O NO DE       */
          /*      MENOR PROBABILIDADE                                  */
          /*      AO RETORNAR O CONTROLE; MO APONTA PARA O NO DE       */
          /*      MENOR PROBABILIDADE E M1 PARA O NO ANTERIOR          */
          /*************************************************************/

1         MENOR:    PROC;
2                   MIN = 9999999;
2                   DO K = 1 TO M;
2    1                 IF Q0 -> PROB  < MIN THEN DO;
2    2                                          M1 = Q1;
2    2                                          M0 = Q0;
2    2                                          MIN = Q0 -> PROB;
2    2                                          END;
2    1                      Q1 = Q0;
2    1                      Q0 = Q0 -> NEXT;
2    1                 END;

2         END MENOR;


1         FIM:      END CODIFI;
```

```
EVEL NEST
          /*****************************************************************/
          /*      IMPLEMENTACAO DA ROTINA DE COMPRESSAO                    */
          /*      USANDO CODIGOS VARIAVEIS PARA NOMES                      */
          /*      LUCIANO PIETRACCI                                        */
          /*****************************************************************/

          COMPN:    PROC (NOME,NOMS);

  1                 DCL NOME CHAR(35),                  /* NOME ENTRADA    */
                        NOMA(35) CHAR(1) DEF NOME;

  1                 DCL NOMS CHAR(40) VAR;              /* NOME SAIDA      */

  1                 DCL MENSA CHAR(40) INIT(' '),       /* AREA TRABALHO   */
                        MENS BIT(320) DEF MENSA;

  1                 DCL MAP(8) BIT(8) VAR               /* GRUPOS PARA COM-*/
                        INIT('10000000'B,'1000000'B,/* PLETAR O ULTIMO */
                             '100000'B,'10000'B,    /* BYTE            */
                             '1000'B,'100'B,'10'B,
                             '1'B);


  1       PROCESS:  DO N = 35 TO 1 BY - 1;              /* RETIRA OS BRAN- */
  1     1             IF NOMA(N) ¬= ' ' THEN            /* COS FINAIS DOS  */
  1     1                GOTO COMPRESS;                 /*,NOMES          */
  1     1           END;                                /*                 */


  1       COMPRESS: K = 0;                              /* CONTADOR = 0    */
  1                 DO J = 1 TO N;
  1     1             L = TRAD(UNSPEC(NOMA(J))+1);      /* CONVERTE A LETRA*/
                                                        /* NO END. CORRES- */
                                                        /* PONDENTE DA TA- */
                                                        /* BELA QUE CONTEM */
                                                        /* O GRUPO E O SEU */
                                                        /* COMPRIMENTO     */
  1     1             SUBSTR(MENS,K+1,KOMP(L)) =
                            GRUP(L);
  1     1             K = K + KOMP(L);
  1     1           END;

  1                 L = MOD(K,8);                       /* RESTO DA DIVISAO*/
  1                 SUBSTR(MENS,K+1,8-L) =              /* DE K POR 8      */
                          MAP(L + 1);
  1                 K = K + 8 - L;

  1                 NOMS = SUBSTR(MENSA,1,K/8);

  1       END COMPN;
```

```
EVEL NEST
          /*****************************************************************/
          /*       IMPLEMENTACAO DA ROTINA DE DESCOMPRESSAO                */
          /*       USANDO CODIGOS VARIAVEIS PARA NOMES                     */
          /*       LUCIANO PIETRACCI                                       */
          /*****************************************************************/

          DCOMPN:   PROC (NOME,NOMS);

1                   DCL NOME CHAR(40) VAR,          /* NOME ENTRADA     */
                        NOM  CHAR(40) INIT(' '),    /* AREA TRABALHO    */
                        NOMA(320) BIT(1) DEF NOM;

1                   DCL NOMS CHAR(35),              /* NOME SAIDA       */
                        NOMB(35) CHAR(1) DEF NOMS;


1         PROCESS:  NOM = NOME;
1                   DO M = LENGTH(NOME)*8 TO 1 BY-1;/* DETETA O FIM DA */
1    1                 IF NOMA(M) = '1'B THEN       /* SEQUENCIA DE EN-*/
1    1                    GOTO DESPACTA;            /* TRADA           */
1    1                 END;                         /*                 */


1         DESPACTA: P = QO;                         /* QO APONTA PARA  */
                                                    /* A RAIZ DA ARVORE*/
1                   K = 0; NOMS = ' ';
1                   DO J = 1 TO M-1;                /* PERCORRE A ARVO-*/
1    1                 IF NOMA(J) = '1'B THEN       /* RE ATE ENCONTRAR*/
1    1                    P = P -> RLINK;           /* UMA FOLHA       */
1    1                                 ELSE
1    1                    P = P -> LLINK;
1    1                 IF RLINK = NULL &
1    1                    LLINK = NULL THEN DO;
1    2                                      K = K + 1;
1    2                                      NOMB(K) = LETRA;
1    2                                      P = QO;
1    2                                      END;
1    1                 END;

1         END DCOMPN;
```

TABELA - 1

CODIFICAÇÃO DO PRIMEIRO CARÁTER				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	10	0.000052	12	011011000001
/	9	0.000047	12	011011000000
A	28837	0.149907	3	010
B	3601	0.018720	6	011010
C	11858	0.061643	4	0000
D	6701	0.034835	5	01100
E	10568	0.054937	4	1100
F	7684	0.039945	5	01110
G	6057	0.031487	5	00010
H	6647	0.034554	5	00111
I	6119	0.031809	5	00011
J	23163	0.120411	3	111
K	1064	0.005531	8	01101110
L	9873	0.051324	4	1001
M	19894	0.103417	3	101
N	6318	0.032844	5	00100
O	6477	0.033670	5	00110
P	6019	0.031289	5	11011
Q	79	0.000411	10	0110110001
R	9336	0.048532	4	1000
S	8045	0.041821	5	01111
T	3272	0.017009	6	001011
U	652	0.003389	9	011011001
V	3096	0.016094	6	001010
W	4848	0.025202	5	11010
X	21	0.000109	11	01101100001
Y	1042	0.005417	8	01101101
Z	1072	0.005594	8	01101111
	192366	1.000000		CUSTO = 4.154497

TABELA - 2

CODIFICAÇÃO DOS SUCESSORES DE '*'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	87	0.003980	8	11111110
/	1677	0.076726	4	0110
A	2650	0.121243	3	000
B	841	0.038477	5	00111
C	2144	0.098092	3	100
D	2174	0.099465	3	101
E	1671	0.076451	4	0101
F	735	0.033628	5	00110
G	516	0.023608	6	011101
H	161	0.007366	7	1111110
I	1062	0.048589	5	01111
J	237	0.010843	7	0111001
K	69	0.003157	8	11001010
L	1607	0.073523	4	0100
M	1241	0.056778	4	1101
N	305	0.013954	6	110011
O	227	0.010386	7	0111000
P	1247	0.057053	4	1110
Q	96	0.004392	8	11111111
R	616	0.028183	5	11110
S	1399	0.064007	4	0010
T	564	0.025804	5	11000
U	57	0.002608	8	11001000
V	335	0.015327	6	111110
W	70	0.003203	8	11001011
X	7	0.000320	10	1100100100
Y	12	0.000549	10	1100100101
Z	50	0.002288	9	110010011
	21857	1.000000		CUSTO = 4.133459

TABELA - 3

CODIFICAÇÃO DOS SUCESSORES DE '/'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	1285	0.003135	9	011111110
/	568	0.001386	10	1101111101
A	31492	0.076822	4	0100
B	21985	0.053630	4	1110
C	38604	0.094171	3	100
D	59273	0.144591	3	001
E	8556	0.020872	6	010101
F	23930	0.058375	4	0001
G	17606	0.042948	5	01011
H	4933	0.012034	6	110110
I	4597	0.011214	7	0111110
J	9408	0.022950	6	011110
K	3215	0.007843	7	1101110
L	23418	0.057126	4	1111
M	35234	0.085950	4	0110
N	9384	0.022891	6	011101
O	7695	0.018771	6	010100
P	23849	0.058177	4	0000
Q	1067	0.002603	9	110111111
R	19063	0.046502	4	1100
S	40050	0.097698	3	101
T	9585	0.023382	5	11010
U	659	0.001607	10	0111111110
V	9246	0.022555	6	011100
W	2183	0.005325	8	01111110
X	414	0.001010	10	1101111100
Y	761	0.001856	10	0111111111
Z	1876	0.004576	8	11011110
	409936	1.000000		CUSTO = 4.130635

TABELA - 4

CODIFICAÇÃO DOS SUCESSORES DE 'A'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	7698	0.018431	6	010101
/	91211	0.218381	2	11
A	552	0.001322	10	0101101010
B	5947	0.014239	6	000010
C	12047	0.028844	5	00000
D	11264	0.026969	5	10101
E	5219	0.012496	6	101001
F	1862	0.004458	8	01010001
G	6430	0.015395	6	010000
H	1726	0.004132	8	01010000
I	6897	0.016513	6	010001
J	792	0.001896	10	0101101011
K	1987	0.004757	8	01011000
L	41769	0.100006	3	100
M	13037	0.031214	5	00011
N	56516	0.135314	3	001
O	12196	0.029200	5	00010
P	3630	0.008691	7	0101001
Q	2218	0.005310	8	01011001
R	72538	0.173674	3	011
S	24009	0.057484	4	1011
T	10084	0.024144	6	010111
U	13773	0.032976	5	01001
V	6159	0.014746	6	000011
W	519	0.001243	10	0101101001
X	361	0.000864	10	0101101000
Y	2248	0.005382	8	01011011
Z	4978	0.011919	6	101000
	417667	1.000000		CUSTO = 3.713394

TABELA - 5

CODIFICAÇÃO DOS SUCESSORES DE 'B'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	627	0.011809	7	1011111
/	1011	0.019041	6	101110
A	12633	0.237928	2	11
B	331	0.006234	7	1011001
C	28	0.000527	11	10111101010
D	298	0.005612	7	1011000
E	16341	0.307762	2	01
F	0			
G	7	0.000132	13	1011110100110
H	34	0.000640	11	10111101011
I	4115	0.077501	3	100
J	9	0.000170	12	101111000001
K	10	0.000188	12	101111010010
L	649	0.012223	6	101101
M	21	0.000396	11	10111101000
N	42	0.000791	10	1011110001
O	6044	0.113832	3	000
P	2	0.000038	14	10111100000000
Q	0			
R	8198	0.154400	3	001
S	107	0.002015	9	101111011
T	17	0.000320	11	10111100001
U	2457	0.046275	4	1010
V	7	0.000132	13	1011110100111
W	2	0.000038	14	10111100000001
X	0			
Y	102	0.001921	9	101111001
Z	4	0.000075	13	1011110000001
	53096	1.000000		CUSTO = 2.740658

TABELA - 6

CODIFICAÇÃO DOS SUCESSORES DE 'C'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	1733	0.014810	6	010001
/	1104	0.009435	7	0101001
A	31170	0.266371	2	00
B	18	0.000154	13	0101000100101
C	2537	0.021681	6	010110
D	5	0.000043	14	01010001000010
E	9628	0.082279	3	100
F	5	0.000043	14	01010001000011
G	5	0.000043	14	01010001000100
H	13476	0.115162	3	101
I	19938	0.170384	3	011
J	5	0.000043	14	01010001000101
K	1202	0.010272	7	0101010
L	1241	0.010605	7	0101011
M	33	0.000282	12	010100010011
N	156	0.001333	9	010100000
O	25966	0.221898	2	11
P	7	0.000060	9	010100001
Q	219	0.001872	14	01010001000111
R	1938	0.016562	6	010011
S	136	0.001162	10	0101000101
T	1621	0.013853	6	010000
U	2613	0.022330	6	010111
V	9	0.000077	13	0101000100000
W	14	0.000120	13	0101000100100
X	5	0.000043	14	01010001000110
Y	1920	0.016408	6	010010
Z	313	0.002675	9	010100011
	117017	1.000000		CUSTO = 3.011425

TABELA - 7

CODIFICAÇÃO DOS SUCESSORES DE 'D'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	1902	0.012137	6	000101
/	3679	0.023476	5	00001
A	33694	0.215000	2	10
B	65	0.000415	11	00010000111
C	31	0.000198	12	000100001101
D	427	0.002725	8	00010001
E	50231	0.320522	2	01
F	25	0.000160	13	0001000011001
G	427	0.002725	8	00010010
H	147	0.000938	9	000000100
I	12276	0.078333	4	0011
J	107	0.000683	10	0001000010
K	38	0.000242	11	00000010110
L	274	0.001748	8	00000001
M	779	0.004971	7	0000010
N	340	0.002170	8	00000011
O	34639	0.221030	2	11
P	1	0.000006	15	000100001100010
Q	1	0.000006	15	000100001100011
R	11839	0.075544	4	0010
S	506	0.003229	8	00010011
T	266	0.001697	8	00000000
U	3892	0.024835	5	00011
V	82	0.000523	10	0000001010
W	192	0.001225	9	000100000
X	1	0.000006	14	00010000110000
Y	798	0.005086	7	0000011
Z	58	0.000370	11	00000010111
	156716	1.000000		CUSTO = 2.675285

TABELA - 8

CODIFICAÇÃO DOS SUCESSORES DE 'E'

ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	310	0.000929	10	1110011011
/	67369	0.201838	2	10
A	4019	0.012041	6	111000
B	2130	0.006382	7	1110010
C	5863	0.017566	6	111111
D	11914	0.035695	5	01000
E	371	0.001112	10	0100101100
F	1151	0.003448	8	11100111
G	3269	0.009794	7	0100100
H	398	0.001192	10	0100101101
I	42307	0.126753	3	000
J	549	0.001645	9	111001100
K	528	0.001582	10	0100101111
L	34925	0.104636	3	110
M	5277	0.015810	6	111101
N	25861	0.077480	4	0101
O	3106	0.009306	7	1111101
P	1396	0.004182	8	11111001
Q	254	0.000761	10	1110011010
R	51573	0.154513	3	011
S	46535	0.139420	3	001
T	9501	0.028465	5	11101
U	3383	0.010136	7	0100110
V	3737	0.011196	7	0100111
W	514	0.001540	10	0100101110
X	1276	0.003823	8	11111000
Y	1477	0.004425	8	01001010
Z	4783	0.014330	6	111100
	333776	1.000000		CUSTO = 3.509524

TABALA - 9

CODIFICAÇÃO DOS SUCESSORES DE 'F'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
¤	692	0.020031	6	010110
/	777	0.022491	6	010111
A	3787	0.109619	3	111
B	7	0.000203	12	010010001011
C	52	0.001505	9	010010100
D	1	0.000029	14	01001000101010
E	9580	0.277303	2	00
F	1333	0.038585	5	01010
G	50	0.001447	9	010010011
H	22	0.000637	10	0100100011
I	6564	0.190002	2	10
J	2	0.000058	14	01001000101011
K	4	0.000116	12	010010001000
L	640	0.018525	6	010011
M	123	0.003560	8	01001011
N	22	0.000637	10	0100100100
O	3469	0.100414	3	110
P	2	0.000058	13	0100100010010
Q	0			
R	6447	0.186615	3	011
S	30	0.000868	9	010010000
T	60	0.001737	9	010010101
U	851	0.024633	5	01000
V	0			
W	2	0.000058	13	0100100010011
X	0			
Y	27	0.000782	10	0100100101
Z	3	0.000087	13	0100100010100
	34547	1.000000		CUSTO = 2.913654

TABELA - 10

CODIFICAÇÃO DOS SUCESSORES DE 'G'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	801	0.014017	6	001001
/	858	0.015015	6	001010
A	9902	0.173286	3	011
B	61	0.001067	9	001000000
C	3	0.000052	13	0010000010000
D	211	0.003692	8	00101100
E	8824	0.154421	3	010
F	37	0.000647	11	00100000101
G	350	0.006125	7	0010001
H	576	0.010080	7	0010111
I	5323	0.093153	4	0011
J	0			
K	3	0.000052	13	0010000010001
L	1358	0.023765	5	00000
M	140	0.002450	9	001011010
N	1369	0.023957	5	00001
O	10861	0.190068	2	10
P	1	0.000017	15	001000001001000
Q	1	0.000017	15	001000001001001
R	2795	0.048912	4	0001
S	54	0.000945	10	0010000011
T	172	0.003010	8	00100001
U	13277	0.232348	2	11
V	3	0.000052	14	00100000100101
W	6	0.000105	13	0010000010011
X	0			
Y	157	0.002747	9	001011011
Z	0			
	57143	1.000000		CUSTO = 3.053007

TABELA - 11

CODIFICAÇÃO DOS SUCESSORES DE 'H'

ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	159	0.003336	8	01000101
/	2101	0.044082	5	01011
A	12890	0.270451	2	00
B	59	0.001238	10	0101011000
C	16	0.000336	11	01010000010
D	59	0.001238	10	0101011001
E	8964	0.188078	2	10
F	28	0.000587	11	01010000111
G	5	0.000105	13	0101000001100
H	26	0.000546	11	01010000110
I	6759	0.141814	3	011
J	6	0.000126	13	0101000001110
K	51	0.001070	10	0101000010
L	325	0.006819	7	0100011
M	454	0.009526	7	0101010
N	429	0.009001	7	0101001
O	12565	0.263633	2	11
P	8	0.000168	13	0101000001111
Q	0			
R	548	0.011498	6	010000
S	132	0.002770	9	010101101
T	209	0.004385	8	01010001
U	1422	0.029836	5	01001
V	35	0.000734	10	0101000000
W	149	0.003126	8	01000100
X	0			
Y	257	0.005392	8	01010111
Z	5	0.000105	13	0101000001101
	47661	1.000000		CUSTO = 2.706112

TABELA - 12

CODIFICAÇÃO DOS SUCESSORES DE 'I'

ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	184	0.000635	11	11110011110
/	8170	0.028197	5	11101
A	34079	0.117617	3	110
B	4502	0.015538	6	111101
C	14292	0.049326	5	01111
D	10100	0.034858	5	01100
E	7744	0.026727	5	10100
F	828	0.002858	9	011100110
G	7987	0.027566	5	11100
H	275	0.000949	10	1111001101
I	114	0.000393	12	111100111111
J	213	0.000735	10	1111001100
K	1048	0.003617	8	11110010
L	29001	0.100091	3	100
M	13176	0.045474	5	01101
N	36659	0.126521	3	001
O	34679	0.119688	3	000
P	1274	0.004397	8	01110010
Q	1860	0.006419	7	1111000
R	38564	0.133096	3	010
S	15990	0.055186	4	1011
T	9695	0.033460	5	11111
U	1027	0.003544	9	011100111
V	7786	0.026872	5	10101
W	90	0.000311	12	111100111110
X	2446	0.008442	7	0111000
Y	288	0.000994	10	1111001110
Z	7675	0.026489	6	011101
	289746	1.000000		CUSTO = 3.894318

TABELA - 13

CODIFICAÇÃO DOS SUCESSORES DE 'J'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	250	0.017593	7	0111101
/	88	0.006193	8	01111000
A	2561	0.180226	2	00
B	44	0.003096	9	011110010
C	21	0.001478	10	0111010101
D	32	0.002252	9	011101000
E	1079	0.075932	4	0110
F	7	0.000493	11	01110001100
G	31	0.002182	9	011100010
H	5	0.000352	12	011101010001
I	456	0.032090	6	011111
J	16	0.001126	10	0111000111
K	12	0.000844	11	01110101001
L	26	0.001830	9	011100000
M	44	0.003096	9	011101011
N	143	0.010063	7	0111001
O	6734	0.473893	1	1
P	1	0.000070	14	01110001101000
Q	1	0.000070	14	01110001101001
R	174	0.012245	7	0111011
S	38	0.002674	9	011101001
T	28	0.001970	9	011100001
U	2364	0.166362	3	010
V	2	0.000141	13	0111000110101
W	4	0.000281	12	011100011011
X	0			
Y	4	0.000281	12	011101010000
Z	45	0.003167	9	011110011
	14210	1.000000		CUSTO = 2.396481

TABELA - 14

CODIFICAÇÃO DOS SUCESSORES DE 'K'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	31	0.002823	9	011111000
/	812	0.073946	4	0110
A	2633	0.239778	2	11
B	6	0.000546	12	011111001001
C	3	0.000273	13	0111111001110
D	12	0.001093	11	01111100101
E	1362	0.124032	3	000
F	29	0.002641	9	011101011
G	0			
H	183	0.016665	7	0111111
I	2145	0.195337	2	10
J	7	0.000637	12	011111001110
K	28	0.002550	9	011101001
L	385	0.035061	5	00111
M	81	0.007376	8	01111101
N	116	0.010564	7	0111011
O	1506	0.137146	3	010
P	1	0.000091	14	01111110011110
Q	0			
R	331	0.030143	6	011110
S	189	0.017212	6	011100
T	28	0.002550	9	011101010
U	689	0.062745	4	0010
V	12	0.001093	11	01111100110
W	27	0.002459	9	011101000
X	3	0.000273	14	01111100111111
Y	358	0.032602	5	00110
Z	4	0.000364	12	011111001000
	10981	1.000000		CUSTO = 3.241053

TABELA - 15

CODIFICAÇÃO DOS SUCESSORES DE 'L'

ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	900	0.004708	8	11011110
/	14859	0.077735	4	0011
A	17127	0.089600	3	100
B	3984	0.020842	6	011101
C	2651	0.013869	6	110110
D	11097	0.058054	4	0000
E	15799	0.082653	4	0110
F	2155	0.011274	7	0111110
G	1015	0.005310	8	11011111
H	9406	0.049208	4	1100
I	31320	0.163851	3	010
J	25	0.000131	14	11011100101101
K	301	0.001575	9	110111000
L	12219	0.063924	4	0010
M	4960	0.025948	5	11010
N	222	0.001161	10	1101110011
O	22573	0.118092	3	111
P	826	0.004321	8	11011101
Q	29	0.000152	13	1101110010111
R	58	0.000303	11	11011100100
S	3645	0.019069	6	011100
T	11810	0.061784	4	0001
U	4409	0.023066	6	011110
V	17304	0.090526	3	101
W	42	0.000220	12	110111001010
X	3	0.000016	14	11011100101100
Y	1345	0.007036	8	01111111
Z	1065	0.005572	8	01111110
	191149	1.000000		CUSTO = 3.880737

TABELA - 16

CODIFICAÇÃO DOS SUCESSORES DE 'M'

ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	1945	0.019994	5	10001
/	4730	0.048624	4	1010
A	35694	0.366928	2	01
B	1999	0.020549	5	10110
C	60	0.000617	10	1000010110
D	21	0.000216	11	10000011111
E	20544	0.211188	2	11
F	32	0.000329	11	10000101111
G	59	0.000607	10	1000010101
H	12	0.000123	12	100001011100
I	10330	0.106191	3	000
J	10	0.000103	13	1000010111011
K	23	0.000236	11	10000101000
L	156	0.001604	8	10000100
M	486	0.004996	7	1000011
N	28	0.000288	11	10000101001
O	13540	0.139188	3	001
P	4239	0.043576	4	1001
Q	6	0.000062	13	1000001111001
R	72	0.000740	9	100000110
S	142	0.001460	8	10000010
T	40	0.000411	10	1000001110
U	2855	0.029349	5	10111
V	11	0.000113	12	100000111101
W	4	0.000041	13	1000001111000
X	0			
Y	233	0.002395	7	1000000
Z	7	0.000072	13	1000010111010
	97278	1.000000		CUSTO = 2.728089

TABELA - 17

CODIFICAÇÃO DOS SUCESSORES DE 'N'

ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
¤	491	0.002546	9	011011011
/	11271	0.058439	4	1011
A	23919	0.124018	3	111
B	457	0.002369	9	011011001
C	12999	0.067398	4	0001
D	21723	0.112632	3	110
E	16361	0.084830	4	0111
F	571	0.002961	9	011011110
G	5530	0.028672	5	00000
H	6432	0.033349	5	01100
I	25169	0.130499	3	001
J	554	0.002872	9	011011100
K	563	0.002919	9	011011101
L	81	0.000420	11	01101101000
M	56	0.000290	12	011011010010
N	3953	0.020496	6	011010
O	18140	0.094054	3	100
P	33	0.000171	14	01101101001101
Q	275	0.001426	10	0110110101
R	1569	0.008135	7	0000110
S	8586	0.044517	4	1010
T	28747	0.149051	3	010
U	2530	0.013118	6	000010
V	340	0.001763	9	011011000
W	38	0.000197	13	0110110100111
X	5	0.000026	14	01101101001100
Y	747	0.003873	9	011011111
Z	1728	0.008959	7	0000111
	192868	1.000000		CUSTO = 3.684359

TABELA - 18

CODIFICAÇÃO DOS SUCESSORES DE '0'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	359	0.001165	10	0010010111
/	112311	0.364575	1	1
A	9674	0.031403	5	00010
B	4482	0.014549	6	000010
C	4026	0.013069	6	000000
D	5626	0.018263	6	011000
E	5198	0.016873	6	001000
F	885	0.002873	9	011001111
G	2082	0.006758	7	0000010
H	719	0.002334	9	011001110
I	1217	0.003951	8	00100100
J	287	0.000932	10	0010010100
K	550	0.001785	9	000001111
L	15238	0.049464	5	01101
M	10711	0.034769	5	00101
N	35172	0.114173	4	0111
O	313	0.001016	10	0010010101
P	3012	0.009777	7	0110010
Q	259	0.000841	11	00100101101
R	26041	0.084532	4	0011
S	49364	0.160242	3	010
T	4577	0.014857	6	000011
U	9957	0.032322	5	00011
V	2860	0.009284	7	0010011
W	547	0.001776	9	000001110
X	75	0.000243	11	00100101100
Y	1036	0.003363	8	00000110
Z	1482	0.004811	8	01100110
	308060	1.000000		CUSTO = 3.245277

TABELA - 19

CODIFICAÇÃO DOS SUCESSORES DE 'P'

ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	1392	0.031401	5	01010
/	378	0.008527	6	100010
A	9423	0.212564	2	11
B	14	0.000316	11	10010010001
C	209	0.004715	8	10010011
D	12	0.000271	11	10010010000
E	12478	0.281483	2	00
F	73	0.001647	9	100100101
G	4	0.000090	12	100100000010
H	1562	0.035235	5	01011
I	7364	0.166117	3	011
J	1	0.000023	15	100100000011100
K	22	0.000496	10	1001000001
L	558	0.012587	6	100011
M	10	0.000226	11	10010000000
N	40	0.000902	10	1001001001
O	5067	0.114301	3	101
P	1285	0.028986	5	10011
Q	3	0.000068	13	1001000000110
R	2698	0.060862	4	0100
S	133	0.003000	8	10010001
T	918	0.020708	5	10000
U	631	0.014234	6	100101
V	1	0.000023	15	100100000011101
W	0			
X	2	0.000045	14	10010000001111
Y	52	0.001173	9	100100001
Z	0			
	44330	1.000000		CUSTO = 2.180908

TABELA - 20

CODIFICAÇÃO DOS SUCESSORES DE 'Q'

ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	117	0.012436	2	01
/	39	0.004145	3	000
A	0			
B	0			
C	0			
D	0			
E	6	0.000638	5	00100
F	0			
G	0			
H	0			
I	2	0.000213	7	0010101
J	0			
K	0			
L	1	0.000106	8	00101100
M	0			
N	0			
O	0			
P	1	0.000106	8	00101101
Q	0			
R	0			
S	32	0.003402	4	0011
T	1	0.000106	7	0010100
U	9206	0.978529	1	1
V	3	0.000319	7	0010111
W	0			
X	0			
Y	0			
Z	0			
	9408	1.000000		CUSTO = 1.038797

TABELA - 21

CODIFICAÇÃO DOS SUCESSORES DE 'R'

ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
≠	3924	0.013803	6	011000
/	15469	0.054415	4	1010
A	62161	0.218664	2	11
B	2680	0.009427	7	0110100
C	6447	0.022679	6	011110
D	8086	0.028444	5	10110
E	37456	0.131758	3	001
F	236	0.000830	10	1000010110
G	7556	0.026580	5	10001
H	205	0.000721	10	1000010010
I	43312	0.152358	3	010
J	103	0.000362	11	10000100110
K	332	0.001168	10	1000010111
L	7378	0.025953	6	011111
M	5915	0.020807	6	011011
N	8167	0.028729	5	10111
O	32535	0.114448	3	000
P	358	0.001259	9	100001000
Q	1987	0.006990	7	1000011
R	12530	0.044077	5	01110
S	1550	0.005452	8	01101010
T	15358	0.054025	4	1001
U	4776	0.016800	6	011001
V	3616	0.012720	6	100000
W	91	0.000320	12	100001001111
X	16	0.000056	12	100001001110
Y	1594	0.005607	8	01101011
Z	440	0.001548	9	100001010
	284278	1.000000		CUSTO = 3.646853

TABELA - 22

CODIFICAÇÃO DOS SUCESSORES DE 'S'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	3001	0.016347	6	011000
/	51877	0.282581	2	00
A	21952	0.119576	3	010
B	431	0.002348	9	101010101
C	10162	0.055354	4	1011
D	95	0.000517	10	1010001010
E	21073	0.114787	3	110
F	87	0.000474	11	10100010111
G	73	0.000398	12	101000101101
H	1746	0.009511	7	0110010
I	17142	0.093375	3	100
J	7	0.000038	12	101000101100
K	1123	0.006117	7	1010010
L	483	0.002631	8	10100001
M	1190	0.006482	7	1010011
N	210	0.001144	9	101000000
O	16835	0.091703	4	0111
P	1849	0.010072	7	0110011
Q	579	0.003154	8	10100011
R	245	0.001335	9	101000001
S	6652	0.036234	5	01101
T	21216	0.115567	3	111
U	2922	0.015917	6	101011
V	266	0.001449	9	101010100
W	1397	0.007610	7	1010100
X	0			
Y	261	0.001422	9	101000100
Z	708	0.003857	8	10101011
	183582	1.000000		CUSTO = 3.298624

TABELA - 23

CODIFICAÇÃO DOS SUCESSORES DE 'T'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
⌘	2213	0.016264	6	001010
/	2483	0.018248	6	001011
A	21096	0.155040	2	10
B	25	0.000184	12	001101000000
C	192	0.001411	10	0011010111
D	6584	0.048388	5	00111
E	19043	0.139952	3	000
F	44	0.000323	12	001101010011
G	26	0.000191	12	001101000001
H	4359	0.032035	5	00100
I	15783	0.115994	3	111
J	17	0.000125	13	0011010100001
K	56	0.000412	11	00110100001
L	215	0.001580	9	001101001
M	164	0.001205	10	0011010101
N	107	0.000786	10	0011010001
O	44525	0.327226	2	01
P	8	0.000059	14	00110101000001
Q	1	0.000007	15	001101010000000
R	6856	0.050387	4	1100
S	779	0.005725	8	00110110
T	7901	0.058067	4	1101
U	2549	0.018733	6	001100
V	34	0.000250	10	001101010010
W	31	0.000228	12	001101010001
X	2	0.000015	15	001101010000001
Y	187	0.001374	10	0011010110
Z	788	0.005791	8	00110111
	136068	1.000000		CUSTO = 3.062807

TABELA - 24

CODIFICAÇÃO DOS SUCESSORES DE 'U'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	30	0.000298	12	010001101001
/	2678	0.026562	5	10111
A	4009	0.039764	5	01011
B	2252	0.022337	5	10101
C	3870	0.038385	5	01010
D	2682	0.026602	5	00000
E	15181	0.150573	3	001
F	563	0.005584	7	1011000
G	3802	0.037710	5	01001
H	244	0.002420	9	010001100
I	12177	0.120778	3	111
J	1681	0.016673	6	010000
K	641	0.006358	7	1011010
L	8684	0.086133	3	100
M	1975	0.019589	5	10100
N	8219	0.081521	4	0111
O	450	0.004463	8	01000100
P	670	0.006645	7	1011011
Q	481	0.004771	8	01000101
R	11414	0.113211	3	110
S	8044	0.079785	4	0110
T	3523	0.034943	5	00001
U	8	0.000079	12	010001101000
V	570	0.005654	7	1011001
W	45	0.000446	11	01000110101
X	173	0.001716	10	0100011011
Y	8	0.005376	8	01000111
Z	6213	0.061624	4	0001
	100821	1.000000		CUSTO = 3.968241

TABELA - 25

CODIFICAÇÃO DOS SUCESSORES DE 'V'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	432	0.008016	6	000011
/	214	0.003971	6	000000
A	22386	0.415362	1	1
B	3	0.000056	12	000010000000
C	12	0.000223	11	00001001011
D	8	0.000148	11	00001000011
E	17394	0.322738	2	01
F	1	0.000019	14	00001000100010
G	6	0.000111	11	00001000001
H	4	0.000074	12	000010001001
I	10784	0.200093	3	001
J	2	0.000037	13	0000100000011
K	11	0.000204	11	00001001010
L	21	0.000390	10	0000100100
M	2	0.000037	13	0000100010000
N	10	0.000186	11	00001000110
O	2042	0.037888	4	0001
P	1	0.000019	14	00001000100011
Q	0			
R	151	0.002802	7	0000011
S	135	0.002505	7	0000010
T	8	0.000148	11	00001000101
U	57	0.001058	9	000010011
V	7	0.000130	11	00001000010
W	1	0.000019	13	0000100000010
X	0			
Y	192	0.003562	7	0000101
Z	11	0.000204	11	00001000111
	53895	1.000000		CUSTO = 1.978273

TABELA - 26

CODIFICAÇÃO DOS SUCESSORES DE 'W'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	63	0.010422	6	000101
/	104	0.017204	6	001101
A	3082	0.509842	1	1
B	2	0.000331	12	001100101100
C	18	0.002978	8	00110000
D	6	0.000993	10	0011001000
E	802	0.132671	3	010
F	4	0.000662	11	00110010111
G	4	0.000662	11	00110010100
H	39	0.006452	7	0001101
I	899	0.148717	3	011
J	7	0.001158	10	0011001001
K	39	0.006452	7	0001110
L	29	0.004797	8	00110011
M	4	0.000662	11	00110010101
N	38	0.006286	7	0001100
O	258	0.042680	4	0000
P	0			
Q	0			
R	43	0.007113	7	0001111
S	229	0.037883	5	00111
T	288	0.047642	4	0010
U	58	0.009595	6	000100
V	2	0.000331	12	001100101101
W	0			
X	0			
Y	27	0.004467	8	00110001
Z	0			
	6045	1.000000		CUSTO = 2.461373

TABELA - 27

CODIFICAÇÃO DOS SUCESSORES DE 'X'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	34	0.007165	6	110000
/	656	0.138252	3	111
A	1242	0.261750	2	00
B	2	0.000421	11	11000110011
C	16	0.003372	8	11000111
D	0			
E	1575	0.331929	2	01
F	2	0.000421	10	1100010010
G	0			
H	0			
I	293	0.061749	4	1101
J	0			
K	1	0.000211	11	11000101010
L	3	0.000632	10	1100010011
M	4	0.000843	10	1100011010
N	4	0.000843	10	1100011011
O	425	0.089569	3	101
P	297	0.062592	3	100
Q	0			
R	3	0.000632	10	1100010100
S	2	0.000421	11	11000101011
T	171	0.036038	5	11001
U	4	0.000843	9	110001000
V	2	0.000421	11	11000110010
W	0			
X	6	0.001264	9	110001011
Y	3	0.000632	10	1100011000
Z	0			
	4745	1.000000		CUSTO = 2.630980

TABELA - 28

CODIFICAÇÃO DOS SUCESSORES DE 'Y'				
ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
*	13	0.000869	10	1001001111
/	5306	0.354845	2	01
A	1146	0.076640	3	101
B	75	0.005016	7	1001010
C	95	0.006353	7	1111010
D	839	0.056109	4	1101
E	485	0.032435	5	11111
F	27	0.001806	9	111101110
G	180	0.012038	6	100110
H	9	0.000602	11	10010011101
I	81	0.005417	7	1001011
J	16	0.001070	9	100100110
K	62	0.004146	7	1001000
L	1043	0.069752	4	0001
M	922	0.061660	4	0000
N	492	0.032903	4	1000
O	663	0.044339	4	1100
P	92	0.006153	7	1001111
Q	0			
R	2161	0.144519	3	001
S	841	0.056243	4	1110
T	90	0.006019	7	1001110
U	200	0.013375	6	111100
V	51	0.003411	8	11110110
W	32	0.002140	8	10010010
X	3	0.000201	11	10010011100
Y	0			
Z	29	0.001939	9	111101111
	14953	1.000000		CUSTO = 3.308834

TABELA - 29

CODIFICAÇÃO DOS SUCESSORES DE 'Z'

ALFABETO	F(I)	P(I)	C(I)	CÓDIGO
¤	63	0.002036	8	10010001
/	7827	0.252892	2	00
A	10553	0.340969	2	01
B	49	0.001583	9	100111100
C	36	0.001163	9	100110001
D	16	0.000517	10	1001100000
E	3851	0.124426	3	111
F	5	0.000162	12	100110100101
G	13	0.000420	11	10011010011
H	27	0.000872	9	100100000
I	3434	0.110953	3	110
J	3	0.000097	12	100110100100
K	131	0.004233	7	1001011
L	105	0.003393	8	10011111
M	122	0.003942	7	1001001
N	75	0.002423	8	10011001
O	1761	0.056898	4	1011
P	30	0.000969	9	100100001
Q	17	0.000549	10	1001100001
R	82	0.002649	8	10011011
S	51	0.001648	9	100111101
T	125	0.004039	7	1001010
U	706	0.022811	4	1000
V	18	0.000582	10	1001101000
W	40	0.001292	9	100110101
X	0			
Y	183	0.005913	7	1001110
Z	1627	0.052569	4	1010
	30950	1.000000		CUSTO = 2.725816

TABELA - 30

ALFABETO	P(I)	CUSTO(I)
*	0.007823	4.133459
/	0.102294	4.130635
A	0.123778	3.713394
B	0.014258	2.740658
C	0.032247	3.011425
D	0.041137	2.675285
E	0.088202	3.509524
F	0.010679	2.913654
G	0.016022	3.053007
H	0.013834	2.706112
I	0.078269	3.894318
J	0.009339	2.396481
K	0.003293	3.241053
L	0.051203	3.880737
M	0.029643	2.728089
N	0.051280	3.684359
O	0.088888	3.245277
P	0.012613	2.180908
Q	0.002369	1.038797
R	0.075123	3.646853
S	0.055359	3.298624
T	0.035258	3.062807
U	0.025580	3.968241
V	0.014247	1.978273
W	0.002742	2.461373
X	0.001236	2.630980
Y	0.004393	3.308834
Z	0.008891	2.725816
	1.000000	CUSTO TOTAL 3.455589

```
VEL NEST
        /*****************************************************************/
        /*      APLICACOES DAS ROTINAS DE COMPRESSAO E                   */
        /*      DE DESCOMPRESSAO PARA REGISTROS SEGMENTADOS              */
        /*****************************************************************/


        CDCOMPS:  PROC OPTIONS (MAIN);

1               DCL ENTRADAC CHAR(80),     /* ENTRADA COMPRESSAO    */
                    SAIDAC   CHAR(55) VAR; /* SAIDA   COMPRESSAO    */

1               DCL ENTRADAD CHAR(55) VAR, /* ENTRADA DESCOMPRESSAO*/
                    SAIDAD   CHAR(55)      /* SAIDA   DESCOMPRESSAO*/
                    INIT(' ');

1               DCL 1 TDR,
                      2 COMPRAIZ FIXED BIN(15) INIT(10),
                      2 INICIOMB FIXED BIN(15) INIT(10),
                      2 TIPOSEG1 CHAR(1) INIT('V'),
                      2 COMPSEG1 BIT(8)  INIT('00001010'B),
                      2 TIPOSEG2 CHAR(1) INIT('F'),
                      2 COMPSEG2 BIT(8)  INIT('00001010'B),
                      2 TIPOSEG3 CHAR(1) INIT('F'),
                      2 COMPSEG3 BIT(8)  INIT('00001010'B),
                      2 TIPOSEG4 CHAR(1) INIT('V'),
                      2 COMPSEG4 BIT(8)  INIT('00001111'B),
                      2 TIPOSEG5 CHAR(1),
                      2 COMPSEG5 BIT(8)  INIT('00000000'B),
                      2 TIPOSEG6 CHAR(1),
                      2 COMPSEG6 BIT(8)  INIT('00000000'B),
                      2 TIPOSEG7 CHAR(1),
                      2 COMPSEG7 BIT(8)  INIT('00000000'B),
                      2 TIPOSEG8 CHAR(1),
                      2 COMPSEG8 BIT(8)  INIT('00000000'B);

1               ON ENDFILE (SYSIN) GOTO FIM;
```

```
VEL NEST
1          LER:      GET EDIT (ENTRADAC)(A(80));
1          /**/      CALL COMPS (ENTRADAC,SAIDAC,TDR);                    /**/
1                    PUT SKIP DATA (ENTRADAC);
1                    PUT SKIP DATA (SAIDAC);
1                    ENTRADAD = SAIDAC;
1          /**/      CALL DCOMPS (ENTRADAD,SAIDAD,TDR);                   /**/
1                    PUT SKIP DATA (ENTRADAD);
1                    PUT SKIP DATA (SAIDAD);
1                    SAIDAD = ' ';
1                    GOTO LER;
1          FIM:      END CDCOMPS;
```