"ESTUDO DE UM INTERPRETADOR COBOL PARA UM MINICOMPUTADOR"

José Nilton Ribeiro de Oliveira

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

Aprovada por:

Martin Diamond
Presidente

RIO DE JANEIRO

ESTADO DA GUANABARA - BRASIL

MAIO DE 1974

À Glaucenete ,

Camila e Calina.

AGRADECIMENTOS

- Aos Professores da COPPE/UFRJ de um modo geral, e em particular ao Prof. Martin Allen Diamond pela paciente orientação e colaboração dada a este trabalho.

- Aos funcionários da COPPE e da Biblioteca da UFRJ.

- Às Instituições que me proporcionaram condições financeiras para a realização do Curso de Engenharia de Sistemas e Computação da COPPE.

- Aos colegas do Banco do Nordeste do Brasil S.A., Flávio Remo Correia Barbosa e Francisco de Paula Sales Pinto , pela valiosa contribuição dada na parte gráfica.

SUMÁRIO

Este trabalho consta do estudo de um interpretador COBOL que procura usar pouco espaço de memória principal.

O trabalho engloba toda parte de análise, onde são construídas tabelas usando pouco espaço que contem todas as informações necessárias. É feita uma representação muito compacta das "pictures". Também é feito um estudo das sub-rotinas que executam as instruções.

Para considerações de programação foi tomada por base a estrutura do computador MITRA 15, da CII - Compagnie Internationale pour l'Informatique.

ABSTRACT

This work consists of a study for a COBOL "interpreter" which has as a goal the use of small ammounts of main storage.

The design and construction of the tables to store the data structures used by the interpreter is presented. In particular, the representation of "PICTURES" is made on a very compact way. A study of the routines that execute the language instructions (verbs) is also presented.

Programming considerations have been based on the structure of the Mini-computer MITRA 15 made by CII - Compagnie Internationale pour l'Informatique.

INDICE

CAPÍTULO I

I N T R O D U Ç Ã O

1 - INTRODUÇÃO

O objetivo inicial deste trabalho era o projeto de um interpretador COBOL para o MITRA 15, um minicomputador produzido pela CII - Compagnie Internationale pour L'Informatique, mas devido tanto à mudança de filosofia do fabricante, no que concerne ao seu uso, como a não disponibilidade da unidade de disco do equipamento acima, instalado na COPPE, mudou-se as diretrizes, aproveitando no entanto a idéia.

Os minicomputadores tornaram-se comercialmente disponíveis no meado dos anos 60 e devido à sua grande flexibilidade e custo relativamente baixo,tendo uma larga aceitação, principalmente na indústria onde era aplicado, em grande parte, em controle de processos.

A boa receptividade no mercado estimulou a que os fabricantes aumentassem grandemente a produção, ao tempo em que novas indústrias do setor passaram a fabricar novos modelos, existindo cerca de cinquenta empresas que produzem minicomputadores. Esta produção em massa fez com que os preços baixassem em torno de 50% do preço inicial, encorajando ainda mais o seu uso. As previsões atuais são de um aumento de aproximadamente 40% por ano nas instalações de minicomputadores, durante a década de 70; espera-se que no fim de 1975 já existam mais de 200.000 minicomputadores em operação [1].

Progressivamente os minicomputadores estão sendo aplicados também na área comercial, suprindo as necessidades das pequenas e médias empresas, no processamento de suas informações, tais

como: controle de estoques, contas a pagar, etc ; o que antes era considerado anti-econômico utilizando-se computadores de maior porte. Também em grandes organizações os minicomputadores vêm sendo usados como concentradores ou pré-processadores.

Tendo em vista esta grande aplicação dos minicomputadores, aumentando agora na área comercial, pensou-se em fazer o estudo de um interpretador de um linguagem de uso mais difundido nesta área. A linguagem não poderia ser outra que não COBOL, já que é utilizada em cerca de 80% das instalações de computadores existentes hoje [2] . Deste modo decidiu-se fazer o ESTUDO DE UM INTERPRETADOR COBOL PARA UM MINICOMPUTADOR.

Como um minicomputador, por sua própria concepção de projeto, dispõe de pouco espaço de memória principal, o estudo levará em conta principalmente a economia de memória, em detrimento do tempo de execução e dificuldades de programação.

Visando a redução do uso de memória principal, se fará uma constante utilização de disco, como memória secundária. Entretanto procura-se, sempre que possível, criar uma estrutura do interpretador que minimize o número de usos da memória secundária, objetivando não torná-lo muito lento.

O interpretador constará de duas fases, que serão chamadas: fase de análise e fase de interpretação.

Na fase de análise será feita toda a análise, tanto léxica como sintática, construída a forma interna dos dados e das instruções e tabelas necessárias. O programa fonte, em sua forma interna, será alocado no disco nesta fase, colocando-se um parágrafo

por setor.

Na fase de interpretação um programa supervisor alocará na memória principal as instruções que se encontram no disco , transferindo um setor de cada vez, numa área de memória apropriada. Os dados serão mantidos também no disco e levados para a memória principal, pelo supervisor, somente aqueles necessários para executar determinada instrução, economizando desta maneira bastante espaço na memória. De posse da instrução e dados, este mesmo programa supervisor chamará uma sub-rotina residente no disco para executar a instrução citada. O supervisor fará a gestão de memória de modo que apenas poucas instruções e sub-rotinas e poucos dados permaneçam na memória principal, fazendo a troca disco-memória sempre que necessário. Como observado, o supervisor utilizará a forma interna das instruções e dados e as tabelas construídas na fase de análise, além das sub-rotinas de execução das instruções.

Já que o supervisor constará de outro trabalho de tese do Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ, sob o título ESTUDO DE UM SUPERVISOR PARA UM INTERPRETADOR COBOL , será estudado aqui apenas a fase de análise e as sub-rotinas que executam as instruções. Tabelas usadas na análise que usam pouco espaço mas com todas as informações necessárias, serão projetadas. Uma representação compacta das "pictures" será dada.

Embora este estudo seja de um interpretador COBOL para um minicomputador de um modo geral, para considerações de programação foi tomado por base a estrutura do MITRA-15 , que é um minicomputador com palavra de 16 bits e capacidade de memória de 4K a 32 K palavras. Como periférico foi considerado uma unidade de dis

co do mesmo computador, cuja configuração tem 256 trilhas e cada trilha possue 16 setores. Cada setor contém 128 palavras.

CAPÍTULO II

F A S E D E A N Á L I S E

2 - FASE DE ANÁLISE

Esta fase constará de toda a análise léxica e sintática, construção das tabelas de variáveis, literais e arquivos e forma interna das instruções do programa fonte.

Como COBOL é formado de quatro divisões, com funções quase que totalmente independentes, esta fase será estudada para cada divisão, na ordem em que elas aparecem no programa fonte.

2.1 - IDENTIFICATION DIVISION

Nesta divisão, a análise verifica as entradas permitidas, que são:

PROGRAM-ID . nome do programa

AUTHOR . nome do autor

DATE-WRITTEN . data

SECURITY . comentário

REMARKS. observações

Todas as entradas acima serão tomadas apenas como comentário, sendo lidas e enviadas para impressão e somente a primeira, PROGRAM-ID, é obrigatória.

2.2 - ENVIRONMENT DIVISION

2.2.1 - ANÁLISE LÉXICA E SINTÁTICA

Nesta divisão a análise verifica os caracteres válidos na formação dos nomes e a existência das seções nela permitidas, ou seja: CONFIGURATION SECTION e INPUT-OUTPUT SECTION, sendo

que somente a última é obrigatória. Dentro de cada seção analisa se as entradas existentes têm construção sintática válida ou não.

Na CONFIGURATION SECTION as entradas serão tomadas como comentário. Estas entradas são:

SOURCE-COMPUTER . modelo do computador

OBJECT-COMPUTER . modelo do computador

Na INPUT-OUTPUT SECTION a primeira entrada deverá ser FILE CONTROL e as demais serem do tipo:

SELECT nome-arquivo ASSIGN TO unidade-tipo acesso.

2.2.2 - TABELA AUXILIAR I

Esta tabela será construída à medida que são encontradas as entradas SELECT e servirá para montar a TABELA DE ARQUIVOS, que será estudada em 2.2.3. A tabela será usada somente durante a fase de análise e terá o seguinte formato:

NOME ARQUIVO	NOME REGISTRO	APONTADOR

O campo "NOME ARQUIVO" será formado de seis bytes contendo o nome do arquivo, permitindo o endereçamento da descrição das características do arquivo, na TABELA DE ARQUIVOS, quando ocorre uma instrução READ.

O campo "NOME REGISTRO" também será formado de seis bytes e conterá o nome do registro. O campo somente será preenchi-

do quando do estudo da "DATA DIVISION", pois é lá que é encontrado o nome do registro. NOME REGISTRO tem funções semelhantes daque - las de NOME ARQUIVO, mas será usado quando ocorre uma instrução WRITE.

"APONTADOR" é um campo formado de uma palavra, 16 bits, e indicará o endereço, na TABELA DE ARQUIVOS, onde o arquivo é descrito. Este endereço representará o deslocamento em relação ao início da referida tabela.

2.2.3 - TABELA DE ARQUIVOS

Esta tabela será construída na análise e mantida na memória principal para ser usada durante a fase de execução pelas instruções READ e WRITE. Seus campos serão preenchidos com informações contidas nas entradas SELECT, completados com informações da "DATA DIVISION" e "PROCEDURE DIVISION". A tabela terá o seguinte formato:

TAMANHO REGISTRO	UNIDADE	E/S	ENDEREÇO DISCO	TIPO DE ORGANIZAÇÃO
10	4	1	12	2

O campo "TAMANHO REGISTRO" será formado de dez bits que, obviamente, informará o tamanho do registro, dado indispensável nas operações READ e WRITE para indicar a quantidade de caracteres a transferir, bem como para atualizar o próximo endereço em um arquivo, quando se trata de organização sequencial. O campo será preenchido com informação obtida da entrada FD da " FILE

SECTION" , seção esta que faz parte da "DATA DIVISION".

O campo "UNIDADE", contendo quatro bits, identificará o periférico, de onde o arquivo será lido ou onde o mesmo será gravado. Assim,tem-se:

0000 - Leitora de cartões

0001 - Impressora

0010 - Disco

0011 - Fita magnética

0100 - Fita de papel perfurada

0101 - Console

0110 - Perfuradora de papel

0111 - Traçador de gráfico

1000 - Display gráfico

Existindo opções para outros periféricos que possam ser acoplados ao sistema.

O campo "ENTRADA/SAIDA(E/S)" contém um bit para informar se o arquivo é de entrada ou saída, dando o sentido do fluxo da informação, será mais um campo de segurança. O valor do campo será preenchido quando for encontrada uma instrução OPEN , na PROCEDURE DIVISION.

O campo"ENDEREÇO DISCO" será formado de doze bits , onde os oito primeiros fornecem o número da trilha e os quatro seguintes o número do setor. Este campo informa onde o arquivo é armazenado no disco.

TIPO DE ORGANIZAÇÃO será um campo de dois bits para informar o tipo de organização do arquivo, podendo o conteúdo destes dois bits ter o segnificado seguinte:

00 - sequencial

01 - acesso direto

10 - indexado sequencial

Quando a organização é sequencial, após cada leitura ou impressão o endereço para a próxima leitura ou impressão é atualizado. Esta atualização é feita no campo "ENDEREÇO DISCO",este campo fornecendo sempre o endereço do próximo regstro a ser lido ou impresso. Somente quando a operação de entrada/saída é executada de ou para um periférico que permite organização tanto sequencial como direta é que se necessita atualizar o endereço.

Quando o periférico aceita somente acesso sequencial, o posicionamento do próximo registro é automático, isto é, o equipamento já para na posição do acesso seguinte.

Toda entrada da tabela de arquivos constará de duas palavras, existindo em cada entrada três bits que não serão utilizados.

Considerando uma quantidade máxima de cinco arquivos para serem manipulados em um programa fonte, esta tabela ocupa uma área de memória máxima de dez palavras, vinte bytes.

2.3 - DATA DIVISION

Neste parágrafo será estudada a "DATA DIVISION",on

de define-se todos os arquivos e variáveis a serem utilizados pela "PROCEDURE DIVISION".

Será estudado inicialmente a construção das entradas, quanto à sua validade, feito em 2.3.1 .

Nos parágrafos 2.3.2 e 2.3.3 será estudada a construção de duas tabelas que objetivam a representação interna dos dados. Estas tabelas serão montadas visando essencialmente economia de memória principal.

Em 2.3.4 será vista uma tabela que fornece o endereço no disco de cada variável de nível 01.

Será estudada posteriormente uma tabela, montada no disco, contendo informações adicionais sobre as variáveis, como: PICTURE e OCCURS. Esta tabela será vista em 2.3.5.

Os valores das variáveis serão armazenados no disco também visando reduzir o uso de memória principal,e a disposição destes valores no disco será analisada também no parágrafo 2.3.4.

2.3.1 - ANÁLISE LÉXICA E SINTÁTICA

Na "DATA DIVISION" a análise estudará a obediência às regras para a construção de nomes definidos pelo programador, o uso de palavras reservadas, a construção das "PICTURES" e os nomes das seções permitidas nesta divisão, que são: FILE SECTION e WORKING-STORAGE SECTION, estudando ainda a sintaxe das entradas válidas em cada das seções anteriores.

2.3.1.1 - FILE SECTION

No COBOL, esta seção deve sempre começar por uma entrada FD, entrada de definição de arquivos, que tem o formato:

FD nome-arquivos [BLOCK inteiro RECORDS]
DATA RECORDS nome-registro-1 nome-registro-2
RECORDS inteiro [CHARACTERS]

As demais entradas desta seção devem ser iniciadas por FD ou por um número de nível de 01 a 49, sendo que toda entrada com número de nível 01 deve ser precedida por uma entrada FD e vice-versa. As entradas com números de nível de 01 a 49 têm o seguinte formato:

número-nível identificador [OCCURS inteiro TIMES]
PICTURE cadeia-de-caracteres.

A análise estudará a sintaxe de cada entrada da FILE SECTION, a fim de determinar se sua construção está correta ou não. Estudará também se os nomes, número de nível e cadeia de caracteres da "PICTURE" estão construídos corretamente.

As informações desta seção, daquelas entradas com número de nível de 01 a 49, serão utilizadas para construir a TABELA AUXILIAR II e a TABELA DE VARIÁVEIS, tabelas estas que serão descritas nas seções 2.3.2 e 2.3.3 respectivamente.

As entradas FD fornecem nome do registro, que será usado na construção da TABELA AUXILIAR I, descrita no parágrafo 2.2.2 e tamanho do registro, utilizado para dar o valor do campo "TAMANHO REGISTRO" da TABELA DE ARQUIVOS, estudada em 2.2.3 .

2.3.1.2 - WORKING-STORAGE SECTION

Toda entrada na WORKING-STORAGE SECTION deve iniciar por um número de nível 77 ou 01 a 49, tendo o formato:

número-nível identificador [OCCURS inteiro TIMES]
PICTURE cadeia-de-caracteres [VALUE literal] .

A análise fará um estudo desta seção semelhante àquele da FILE-SECTION e as informações obtidas serão utilizadas na montagem da TABELA AUXILIAR II e TABELA DE VARIÁVEIS, que serão estudadas nos dois parágrafos que seguem.

Quando existe a cláusula "VALUE", o literal que lhe é associado será colocado no disco, no endereço correspondente ao da variável descrita nesta entrada.

2.3.2 - TABELA AUXILIAR II

À medida que se pesquisa a "DATA DIVISION" constroe-se a TABELA AUXILIAR II, que tem por objetivo endereçar as variáveis na TABELA DE VARIÁVEIS, que será estudada no parágrafo 2.3.3 , durante a análise da "PROCEDURE DIVISION".

Esta tabela conterá uma entrada para cada variável definida na "DATA DIVISION" e será mantida na memória somente durante a fase de análise, sendo seu espaço liberado para outros fins na fase de execução. A tabela terá o seguinte formato:

NOME VARIÁVEL	NÚMERO NIVEL	APONTADOR

O campo "NOME VARIAVEL" será formado de seis bytes que conterão o nome da variável. Este nome da variável pode ser formado de até trinta caracteres, mas serão tomados apenas os seis primeiros. Quando o nome é formado de menos de seis caracteres,as posições mais à direita do campo serão completadas com brancos.

O campo "NÚMERO DE NÍVEL" será constituído de seis bits que informarão o número de nível de cada variável. Os números de nível de 01 a 49 serão representados pelo próprio conteúdo do campo. Quando este conteúdo é 50, o campo representará um número de nível 77.

Esta informação é necessária para determinar os comprimentos das variáveis que contêm "OCCURS" e para fornecer as dimensões dos índices, quando da formação de quádruplas para variáveis subscritas, que será estudado em 2.4.5 .

O campo "APONTADOR" será formado de dez bits que indicarão o deslocamento, em relação ao início da TABELA DE VARIÁVEIS, onde a variável é descrita.

Na fase de análise, quando se encontra uma variável em uma instrução da "PROCEDURE DIVISION", a instrução será traduzida para quádrupla, que é sua forma interna, e o campo desta variável na quádrupla conterá o valor do apontador, obtido da TABELA AUXILIAR II .

Deste modo, nas quádruplas, que serão explicadas em 2.4.5 , cada operando será um apontador para uma entrada na TABELA DE VARIÁVEIS.

2.3.3 - TABELA DE VARIÁVEIS

Paralelamente à construção da TABELA AUXILIAR II , monta-se a TABELA DE VARIÁVEIS, que será mantida na memória principal a fim de ser usada durante a fase de execução. Esta tabela objetiva, tanto endereçar as variáveis no disco ou na memória, como fornecer o endereço em outra tabela, no disco, que descreve as características das variáveis, como "PICTURE" e "OCCURS".

O comprimento desta tabela será limitado a 512 palavras.

As entradas da TABELA DE VARIÁVEIS terão o formato seguinte:

ENDEREÇO MEMÓRIA	TIPO-V	MUD	ORDENA TABELA

Cada entrada usará uma palavra de memória. Deste modo podem ser definidas até 512 variáveis no programa fonte.

Nos parágrafos seguintes serão explicados os campos da TABELA DE VARIÁVEIS.

2.3.3.1 - ENDEREÇO MEMÓRIA

Objetivando a economia de espaço na memória principal, as variáveis serão armazenadas no disco e alocadas na memória somente quando solicitadas. Também as variáveis serão liberadas da memória se não mais estiverem sendo utilizadas. Deste modo haverá uma constante permuta de variáveis do disco para a memória.

O campo "ENDEREÇO MEMÓRIA", composto de nove bits,

informará o endereço de uma variável na memória principal, quando a mesma se encontra ali.

Quando se trata de variáveis simples, isto é, sem subscritos, este endereço representa o deslocamento em relação ao início de uma área da memória onde se alocam as variáveis em uso, chamada TABELA DE VARIÁVEIS EM USO.

A TABELA DE VARIÁVEIS EM USO será formada de dois campos, o primeiro contendo a "PICTURE" tendo tamanho de um, dois ou quatro bytes, dependendo do tipo da "PICTURE", o segundo campo terá comprimento de acordo com o tamanho da variável e será usado para armazenar o próprio valor desta variável. A quantidade de bytes para cada variável será determinado pela "PICTURE" . Como as variáveis têm comprimento diferentes, reserva-se uma área na memória, uma palavra, para informar o próximo endereço disponível na tabela.

Quando a variável é subscrita, o campo de ENDEREÇO DE MEMÓRIA aponta para uma lista. A lista conterá como primeira entrada o valor da "PICTURE", ocupando um espaço de um, dois ou quatro octetos dependendo do tipo da "PICTURE", do seguinte modo:

TIPO DA PICTURE	ESPAÇO OCUPADO NA LISTA
Alfabética ou alfanumérica sem caracteres de edição	1 byte
Numérica sem edição	1 palavra
Quando há edição	2 palavras

As demais entradas na lista conterão dois campos, um dando o deslocamento da variável em relação àquela que tem nú-

mero de nível 01 e outro para o valor. Quando o valor de uma variável subscrita é solicitado, primeiro calcula-se o seu deslocamento em relação à variável de número de nível 01, através da expressão dada em 2.3.4 . Com este valor pesquisa-se a lista sequencialmente. Caso o deslocamento seja encontrado, a variável está na memória, caso contrário, procura-se a variável no disco e a coloca na lista . A fim de não tornar estas listas muito grandes, e com isto gastar muito espaço de memória, limita-se a dez o número máximo de entradas em cada lista. Também objetivando economia de memória, será limitado a três o número destas listas, para os três operandos envolvidos numa instrução.

2.3.3.2 - TIPO-V

O campo "TIPO-V", formado de 1 bit, tem por função informar o tipo de variável, isto é, se seu número de nível é 01 ou não, permitindo assim identificar o significado do campo ORDEM TABELA que será visto em 2.3.3.4 .

Uma variável de nível 01 com todas suas sub-variáveis, ou seja, aquelas variáveis que lhe seguem e tem número de nível entre 02 e 49, será chamado um registro.

2.3.3.3 - MUD

MUD é um campo formado de um bit e será usado para informar se determinada variável foi alterada ou não.

2.3.3.4 - ORDEM TABELA

O campo "ORDEM TABELA" será formado de cinco bits e tem dois significados, de acordo com o tipo de variável.

Quando a variável tem número de nível 01, o campo informará o número de ordem da definição da variável, na "DATA DIVISION", em relação às variáveis de número de nível 01 anteriores, isto é, se o conteúdo do campo é 8 indica que se trata da oitava variável de nível 01 da divisão "DATA". Este campo será utilizado para calcular o endereço onde as especificações da variável se encontram no disco, a ser descrito em 2.3.5 . O campo também será usado para determinar o endereço do registro no disco, o que será estudado em 2.3.4 .

Limitando-se a 32 o número máximo de variáveis definidas em um registro e armazenando as especificações de cada variável em quatro palavras, no disco, as especificações de todas as variáveis de um registro serão alocadas em um mesmo setor do disco, 128 palavras. Deste modo, se o campo "ORDEM TABELA" de uma variável com número de nível 01 contém um valor n, isto indicará que as especificações da variável se encontrarão nas quatro primeiras palavras do n-ésimo setor, a partir do primeiro setor reservado para este fim.

Quando a variável tem número de nível maior do que 01, o campo será usado para informar a ordem da variável em relação àquela variável de número de nível 01 que é raiz da estrutura da qual a variável considerada faz parte. Esta informação será usada para endereçar, dentro de um setor, as especificações da variável. Assim, quando se quer as especificações de uma determinada

variável, primeiro aloca-se na memória o setor que as contêm e a seguir calcula-se o seu endereço dentro do setor, da seguinte maneira:

END-SETOR = 4*ORDEM-TABELA

onde END-SETOR é o deslocamento em relação ao início do setor, onde são descritas as especificações da variável e 4 representa a quantidade de palavras utilizadas para descrever cada variável. Na memória tem-se o setor:

2.3.4 - ENDEREÇO DISCO

ENDEREÇO DISCO é uma tabela cujas entradas, constituídas de uma palavra , informam o endereço de cada registro no disco.

O acesso à tabela é feito usando-se o valor do campo ORDEM TABELA da tabela de variáveis, quando a variável é de nível 01 . Este campo fornece o deslocamento, em número de palavras, em relação ao início da tabela, onde o endereço do registro é definido.

Em cada entrada, os 8 primeiros bits fornecem o número da trilha e os 4 bits restantes informam o número do setor. Es

te campo informará onde o registro está alocado no disco. Pode-se, desta maneira, endereçar 256 trilhas e, em cada uma, 16 setores, sendo esta a constituição de uma das configurações de disco do MITRA 15 .

No disco as variáveis serão armazenadas de modo que cada registro se inicie em um novo setor e todas as sub-variáveis de um registro fiquem armazenadas sequencialmente no disco, a partir do setor onde se inicia a colocação deste registro.

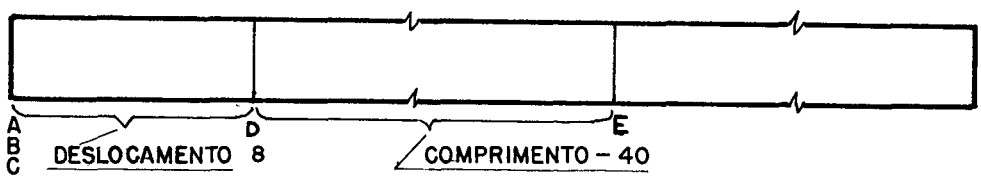
Quando uma variável é solicitada em uma instrução , primeiro são lidas as suas especificações, do disco, aonde se obtem a "PICTURE" e um deslocamento. Este deslocamento informa a distância da variável considerada em relação àquela com número de nível 01 do mesmo registro, isto é, informa o deslocamento da variável em relação ao início do setor onde ela se encontra. Assim se é requerida a variável D, do registro A abaixo

```
01      A.
        02      B.
                03      C       PIC     9(8).
                03      D       PIC     X(40).
                03      E       PIC     9(5)V99.
                03      F       PIC     99.
        02      G
```

tem-se o seguinte:

Em primeiro lugar são lidas as especificações de D do disco, fornecendo PICTURE X(40) e DESLOCAMENTO 8 . Posteriormen

te o registro é lido do disco, para uma área de memória reservada para este fim, aparecendo da seguinte maneira:



por meio de indexação obtem-se a variável desejada.

Quando a variável é subscrita, isto é, quando ela tem a cláusula "OCCURS" ou quando o ítem grupo, do qual a variável faz parte, contém "OCCURS", o conteúdo do endereço no setor do disco, determinado através do deslocamento, como visto no caso anterior, representa aquela variável cujos índices são iguais a 1. Deste modo, o deslocamento de uma variável subscrita, com índices quaisquer, em um setor, será determinado pela seguinte expressão

```
DESLOC(VAR(I,J,K))=DESLOC(VAR(1,1,1))+(I-1)*COMP1+
                   (J-1)*COMP2+(K-1)*COMP3
```

Os termos da expressão serão explicados a seguir, considerando n o número de nível da variável VAR .

DESLOC(VAR(I,J,K)) - Deslocamento para determinar o endereço da variável VAR(I,J,K), em relação ao início do setor onde o registro, do qual a variável VAR faz parte, está armazenado.

DESLOC(VAR(1,1,1)) - Deslocamento para determinar o endereço da variável com índices iguais a 1, também com relação ao início do setor onde o registro está armazenado.

COMP1 - Tamanho dado pelo campo "PICTURE" da própria variável VAR , se ela possuir cláusula "OCCURS", ou da variável de número de nível imediatamente menor que n que contenha aquela cláusula.

COMP2 - Tamanho fornecido pelo campo "PICTURE" da variável com número de nível imediatamente menor que aquele da variável usada em COMP1 e contenha cláusula "OCCURS".

COMP3 - Tamanho fornecido pelo campo "PICTURE" da variável com número de nível imediatamente menor que o número de nível da variável usada em COMP2 e contenha cláusula "OCCURS".

O cálculo deste deslocamento será preparado na fase de análise, tranformando a expressão em quádruplas, e será visto no parágrafo 2.4.5 .

O seguinte exemplo mostra a maneira que um registro contendo variáveis subscritas será armazenado no disco.

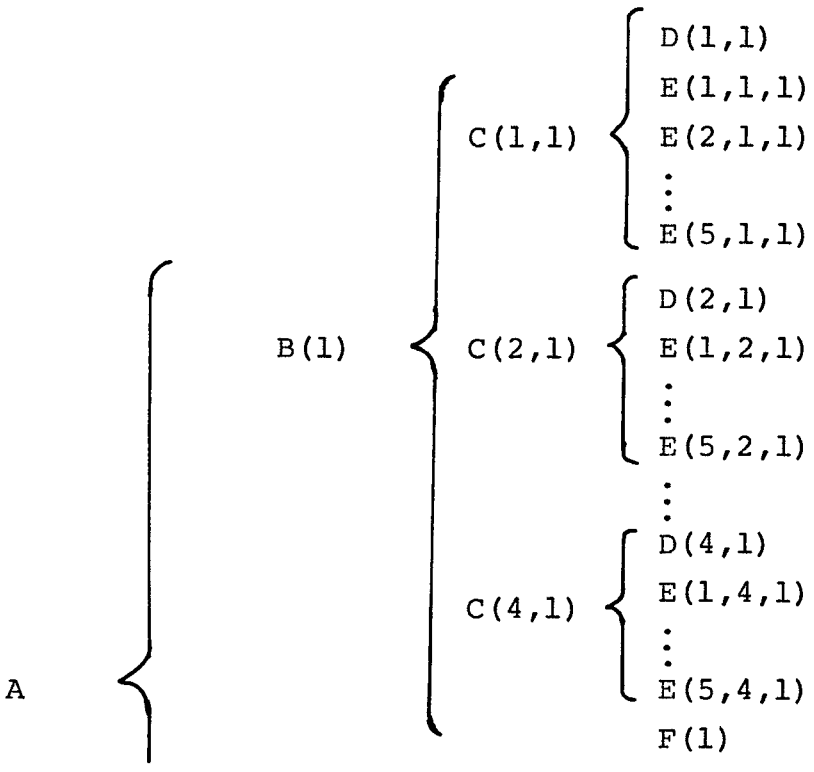
Seja o registro:

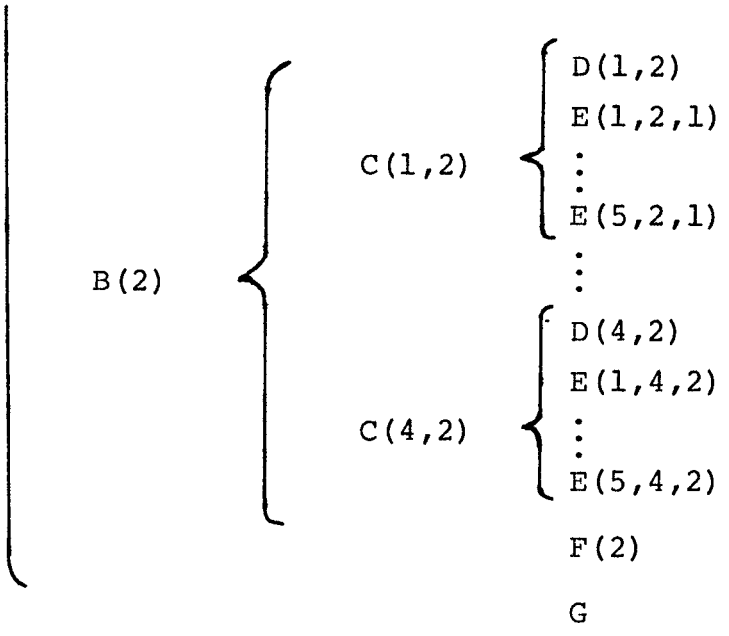
```
01   A.
     02     B    OCCURS  2  TIMES.
            03   C  OCCURS  4  TIMES.
                 04   D   PIC  XXX.
                 04   E   OCCURS  5  TIMES  PIC 99.
            03   F       PIC X(8).
     02     G           PIC X(10).
```

Os comprimentos das variáveis são:

VARIÁVEL	COMPRIMENTO (BYTES)
A	130
B	60
C	13
D	3
E	2
F	8
G	10

No setor do disco as variáveis serão alocadas na ordem seguinte:





As dimensões dos índices serão:

COMP1	2
COMP2	13
COMP3	60

Quando o campo DESLOC ou DESLOC(VAR(I,J,K)) é maior do que 256, significa que a variável desejada não está no primeiro setor do disco a partir de onde o registro é armazenado. Para se obter o número do setor onde se encontra a variável, divide-se o valor de DESLOC por 256 obtendo-se o valor que se deve adicionar ao endereço do setor inicial, a fim de encontrar o setor onde se encontra realmente a variável. O novo deslocamento será o resto da divisão.

2.3.5 - TABELA DE ESPECIFICAÇÕES

Objetivando economizar memória principal, as especificações de cada variável serão alocadas no disco, em uma área que será chamada TABELA DE ESPECIFICAÇÕES. Para descrever as especificações de uma variável serão usadas quatro palavras e, limitando a 32 o número máximo de variáveis em um registro, as especificações de todas as variáveis de um registro serão armazenadas em um setor. Do mesmo modo, um setor conterá especificações de apenas um registro.

Esta tabela será formada dos seguintes campos:

PICTURE	OCCURS	DESLOC

Estes campos serão discutidos nos próximos parágrafos.

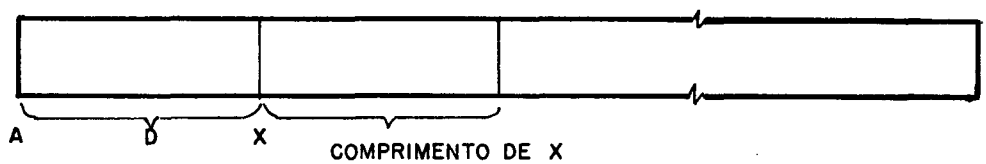
2.3.5.1 - OCCURS

O campo "OCCURS" é formado dos dez primeiros bits da terceira palavra e informará o número de vezes que determinada variável, ou estrutura, será repetida, fornecendo assim as dimensões dos índices quando a variável é subscrita. Cada variável poderá, portanto, ser repetida no máximo 1024 vezes.

2.3.5.2 - DESLOC

DESLOC é um campo formado dos dez bits seguintes àqueles do "OCCURS" de uma entrada na tabela e informará o deslocamento de determinada variável em relação ao início do setor onde a mesma está alocada, quando a variável não tem subscritos. Quando a variável é subscrita, este deslocamento se refere à posição da variável com índices iguais a 1, isto é, DESLOC(VAR(1,1,1)) da expressão dada em 2.3.4 e o deslocamento da variável com os subscritos desejados é calculado por meio dessa expressão.

Quando uma variável é solicitada por uma operação , o setor endereçado pela variável de nível 01, associado a ela, é lido numa área de memória especificada para isto, e o endereço da variável requerida é obtido por indexação, usando este deslocamento . Assim, se no registro A, nível 01 , deseja-se obter a variável X cujo deslocamento, em relação a A , é D, tem-se :



2.3.5.3 - PICTURE

O campo "PICTURE" informará o comprimento da variável, bem como detalhes de seu formato, como posição do ponto decimal, existência de caracteres de edição, etc.

Como a "picture" será alocada na memória principal,

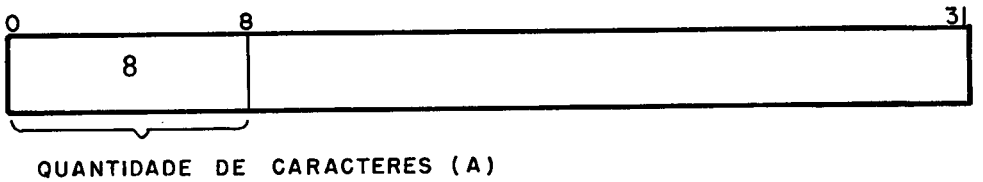
durante a fase de execução, na tabela de variáveis em uso, junto com o valor de sua variável, procurou-se representar estas "pictures" de modo bastante compactado, tirando-se vantagem quer do tipo de "picture" quer das propriedades dos caracteres de edição, como ca - racteres mutuamente exclusivos, caracteres com posições fixas, etc. Deste modo conseguiu-se descrever qualquer tipo de "picture" em no máximo duas palavras, 32 bits, o que representa uma boa compactação, dada a grande quantidade de caracteres de edição.

No disco as "pictures" sempre ocupam duas palavras para sua descrição, mas quando alocadas na memória, elas ocupam um byte, se a "picture" é alfabética, uma palavra se é numérica ou alfa numérica ou duas palavras nos demais casos.

Tem-se os seguintes tipos de "picture":

a) ALFABÉTICO

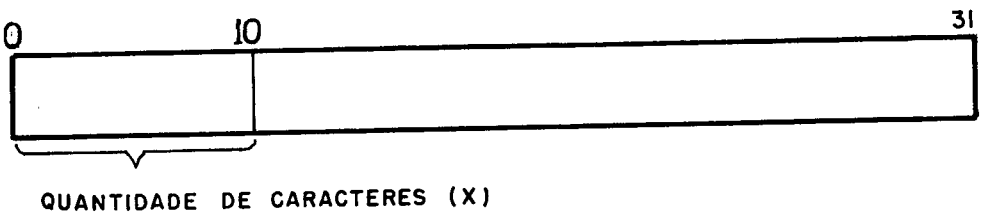
Uma variável tem picture alfabética se a cadeia de caracteres usada em sua definição é formada somente de A . Seu conteúdo pode ser qualquer letra ou branco. A quantidade de caracteres será determinada pelo conteúdo dos oito primeiros bits da primeira palavra da entrada da tabela para a variável considerada. Assim tem-se:



b) ALFANUMÉRICO

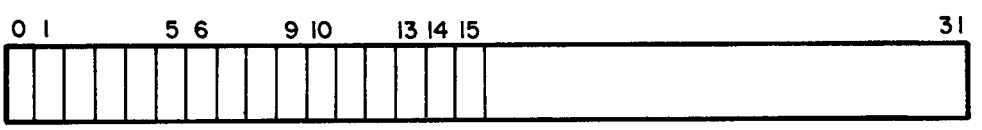
A picture de uma variável é alfanumérica se sua de

finição é formada somente de caracteres X . O seu valor pode ser constituído de qualquer caracter EBCDIC . O comprimento da variável será determinado também pelo conteúdo dos primeiros dez bits da primeira palavra da entrada, da TABELA DE ESPECIFICAÇÕES, para aquela variável. Deste modo, tem-se:



c) NUMÉRICO

Uma picture é considerada numérica quando na entrada para definição da variável o campo da referida picture é constituído de uma combinação dos caracteres 9,V,P e S , desde que tenha pelo menos um 9 . A quantidade de caracteres da variável será determinada pela primeira palavra da entrada na tabela, para a variável desejada, da seguinte maneira:



Os bits nesta palavra tem o segnificado descrito a seguir.

BITS	SIGNIFICADO
0	informará se a variável tem sinal ou não.
1-5	fornecerá a quantidade de algarismos decimais da parte inteira da variável.

6-9 fornecerá o número de algarismos decimais da parte fracionária

10-13 informará a quantidade de P's existentes na definição da picture . Cada P indica que o número será multiplicado ou dividido por 10, dependendo da posição de P , se à direita ou à esquerda do número, respectivamente.

14 determinará se os P's existentes estão à direita ou à esquerda do número.

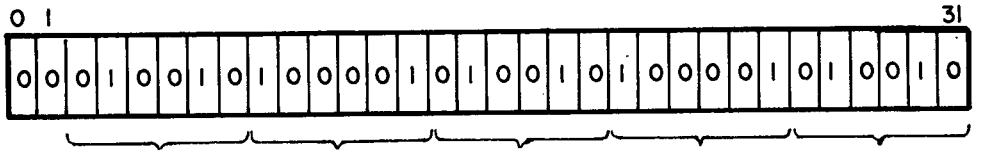
15 informará se a variável será armazenada em binário ou decimal, isto é, indicará a "USAGE".

O código P em uma picture indica o posicionamento de um suposto ponto decimal à esquerda ou à direita da picture , conforme P esteja à esquerda ou à direita. Assim, tem-se:

PICTURE	VALOR NA MEMÓRIA	VALOR REAL
PPP999	520	0.000520
999PPP	608	608000

d) ALFANUMÉRICO EDITADO

Quando na definição de uma variável o campo de picture é constituído de uma combinação dos caracteres A,X,B e Ø de modo que contenha pelo menos um B ou um Ø além de um dos outros caracteres, a variável é dita alfanumérica de edição. Os B's ou Ø's que apareçam significa que haverá inserção de brancos ou zeros, respectivamente, nas posições correspondentes a estes caracteres. A distribuição dos tipos de caracteres numa variável ocupará duas palavras, e será feita como segue:



BITS Ø e 1

Os bits Ø e 1 informam como será a distribuição dos demais bits que compõem as duas palavras. De acordo com os componentes destes dois bits, o restante da palavra dupla terá a seguinte composição:

CONTEÚDO DOS BITS 0 e 1	COMPOSIÇÃO
00	Quando os bits Ø e 1 contêm ØØ , o restante da palavra dupla será formado de cinco campos de seis bits, onde os dois primeiros bits de cada campo informam o tipo de caracter e os quatro seguintes a quantidade destes caracteres.
01	Este valor para os bits Ø e 1 informa que o restante da palavra dupla será formado de seis campos de cinco bits, sendo que os dois primeiros bits de cada campo indicam o tipo de caracter e os três seguintes a quantidade dos mesmos.
10	Este valor indica que os demais bits da palavra dupla formam três campos de dez bits cada, com as mesmas características dos anteriores.
11	Os bits restantes das duas palavras formam três cam

pos de seis bits e um de doze. Em cada um destes campos os dois primeiros bits, como sempre, indi - cam o tipo de caracter e os demais bits a quantida de.

Os tipos de caracteres permitidos, associados com o valor dos dois primeiros bits de cada campo são:

00 - A

01 - X

10 - B

11 - Ø

Assim, se a picture fôsse definida pela cadeia de caracteres: XXBXXBXX , a palavra dupla de definição da "PICTURE" teria a composição dada na figura vista anteriormente neste ítem.

Decidiu-se usar várias configurações na palavra dupla que define este tipo de PICTURE devido a probabilidade de surgir PICTURES com muitos campos, de caracteres diferentes, de poucos caracteres, como o já foi visto antes XXBXXBXX que pode ser descrito em cinco campos de seis bits, mas também há possibilidade de aparecer PICTURES com poucos campos, de muitos caracteres, que seria definido com três campos de dez caracteres, além das outras possibilidades estudadas.

e) NUMÉRICO DE EDIÇÃO I

Enquadram-se neste tipo as variáveis em cujas "PICTURES" aparecem os caracteres 9 e pelo menos um B ou um Ø.

Onde aparece B ou Ø na "PICTURE", será inserido um caracter branco ou zero respectivamente.

A configuração das palavras de PICTURE da tabela de especificações será semelhante àquela do tipo alfanumérico editado. Assim, os dois primeiros bits informarão a distribuição dos demais em campos, que variam de acordo com o conteúdo destes dois bits. A única diferença é que aqui são permitidos caracteres 9 e não serão admitidos caracteres X e A .

f) NUMÉRICO DE EDIÇÃO II

São classificados neste grupo as variáveis que em cujas PICTURES aparece pelo menos um tipo dos seguintes caracteres: Z , . * + - $ CR DB além de poder aparecer qualquer dos outros caracteres permitidos numa picture numérica.

Alguns dos caracteres citados são mutuamente exclusivos, são eles: + e - , CR e DB, Z e * . Quando a picture contém +, - ou $ múltiplos, estes caracteres, juntamente com Z e * também são mutuamente exclusivos.

O campo PICTURE da tabela de especificações, uma palavra dupla, representa a distribuição dos caracteres na variável, para isto os bits neste campo significam:

BIT(S)	SIGNIFICADO
0	Informará se a variável tem ponto decimal editado ou não. Informação posterior, nesta palavra dupla, indicará a posição do ponto.
1	O conteúdo do bit 1 indicará se a variável tem inserção de vírgula ou não. Apenas um bit é usado porque a vírgula sempre é inserida de 3 em 3 dígitos,a partir do ponto decimal, da direita para a esquerda, portanto é suficiente informar a existência de inserção ou não da vírgula.
2-4	Os bits 2 a 4 informarão se haverá inserção fixa de um dos caracteres: +,-,CR ou DB; os sinais podem ser colocados à direita ou à esquerda do valor da variável. Os sinais + e - , quando indicados neste campo, não podem ser repetidos. Assim, tem-se: 000 - não há inserção de nenhum dos caracteres, 001 - inserção de CR , 010 - inserção de DB, 011 - inserção de + à direita, 100 - inserção de + à esquerda, 101 - inserção de - à direita, 111 - inserção de - à esquerda.

De acordo com a picture, a inserção se processa da seguinte maneira:

PICTURE	VALOR	RESULTADO EDITADO
999.99+	+6555.556	555.55+
+9999.99	-5685.505	-5685.50
9999.99CR	+615.12	0615.12
9999.99DB	-308.28	0308.28DB

5 O bit 5 informará se o caracter $ aparece apenas uma vez na picture. Neste caso, haverá uma inserção fixa, isto é, o caracter será inserido na posição indicada por ele na picture. A inserção se verifica do seguinte modo:

PICTURE	VALOR	RESULTADO EDITADO
$999.99	-245.61	$245.61
-$999.99	-185.222	-$185.22
$99999.99CR	+108.36	$00108.36

6-7 Estes bits indicarão a quantidade de brancos a serem inseridos após o primeiro ou segundo caracter, dependendo da picture ser iniciada por um único sinal +, - ou $ ou por um par de caracteres +$ ou -$. A inserção se processa assim:

PICTURE	VALOR	RESULTADO EDITADO
$BB999.99	+805.40	$ 805.40
-$B9999.99	-128.85	-$ 0128.85

8-9 Os bits 8 e 9 informarão a quantidade de brancos a serem inseridos antes do caracter mais à direita. A inserção se verifica como segue:

PICTURE	VALOR	RESULTADO EDITADO
$9999.99BBDB	-123.45	$0123.45 DB

10-12 Os bits 10 a 12 informarão a existência de um dos caracteres de inserção flutuante e supressão e substituição de zero, ou seja, a existência na picture de um dos seguintes tipos de caracteres:

001 - existência de um ou mais "Z"

010 - existência de um ou mais "*"

011 - a picture possue dois ou mais "$"

100 - existência de dois ou mais "+"

101 - existência de dois ou mais "-"

A edição para este caso se verifica da seguinte maneira:

PICTURE	VALOR	RESULTADO EDITADO
$$$9.99	.123	$0.12
$,$$$,999.99	-1234.56	$1,234.56
+,+++,999.99	-123456.789	-123,456.78
$Z,ZZZ,ZZZ.ZZCR	+12345.67	$ 12,345.67
$B*,****,***.**BBDB	-12345.67	$ ***12,345.67 DB

13-17 Estes bits informarão a quantidade dos caracteres determinados pelo bits 10-12 .

18-22 O conteúdo dos bits 18 a 22 fornecerão a quantidade de 9's existentes no campo de picture da variável. Estes caracteres sempre estarão à direita dos caracteres determinados pelos bits 10 a 12.

23-26 Os bits 23 a 26 serão usados para informar a quantidade de algarismos da parte fracionária.

27-30 O conteúdo dos bits 27 a 30 informarão a quantidade de zeros que devem ser inseridos à direita do valor da variável. A edição, neste caso, será do seguinte modo:

PICTURE	VALOR	RESULTADO EDITADO
9,999,000	12345	2,345,000
9B999B000	6789	6 789 000

2.4 - PROCEDURE DIVISION

Na "PROCEDURE DIVISION" será completada a TABELA DE ARQUIVOS, vista na seção 2.2.3 , preenchendo-se o campo ENTRADA/SAÍDA, quando for encontrada uma instrução OPEN. Na seção 2.4.1 será estudada a validade das entradas e dos nomes encontrados nesta divisão. Na seção 2.4.2 será estudada a tabela de endereçamento de parágrafos. Em 2.4.3 estuda-se a construção da tabela de literais .

Em 2.4.4 será vista a constituição da tabela de temporárias. A forma interna das instruções será estudada em 2.4.5 .

2.4.1 - ANÁLISE LÉXICA E SINTÁTICA

Nesta divisão a análise verificará os caracteres válidos na formação dos elementos do COBOL, a quantidade de caracteres utilizados nos identificadores e a construção das entradas de acordo com a sintaxe da linguagem.

Segundo a sintaxe, toda entrada que inicie por um nome definido pelo programador, este será um nome de parágrafo. Quando a entrada é em cartões, este nome deve ser iniciado na margem A, isto é, entre as colunas 8 e 12 do cartão. Quando gravado em fita ou disco, deve-se colocar um caracter especial para indicar que é um rótulo.

Quando a entrada não é um nome de parágrafo, deve ser iniciada por uma palavra reservada, no caso: um IF, GO TO ou um verbo como: MOVE, ADD, SUBTRACT , READ, etc.

Para todo nome definido pelo programador, que não seja rótulo, verifica-se se ele foi definido na "DATA DIVISION", isto é, toda variável usada na "PROCEDURE DIVISION" deve ter sido definida na "DATA DIVISION".

Nesta seção verifica-se também a compatibilidade das "PICTURES" usadas numa operação. Assim em operações aritméticas , as "PICTURES" das variáveis têm que ser numéricas sem edição. Quando existe a cláusula "GIVING", a variável que lhe segue pode ter "PICTURE" numérica de edição.

Na análise verifica-se ainda, através das "PICTURES", as operações de movimento que são válidas. A seguir,tem-se os tipos de movimento permitidos.

TABELA DE MOVIMENTOS PERMITIDOS

ITEM RECEPTOR / ITEM EMISSÃO	GR	AL	AN	NM	NE	ANE
GRUPO (GR)	P	P	P	P	P	P
ALFABÉTICO (AL)	P	P	P	N	N	P
ALFANUMÉRICO (AN)	P	P	P	P	P	P
NUMÉRICO (NM)	P	N	P	P	P	P
NUMÉRICO DE EDIÇÃO (NE)	P	N	P	N	N	P
ALFANUMÉRICO DE EDIÇÃO (ANE)	P	P	P	N	N	P
SPACES (AL)	P	P	P	N	N	P
LITERAL NUMÉRICO (NM)	P	N	P	P	P	P
LITERAL NÃO-NUMÉRICO (AN)	P	P	P	P^I	P^I	P

onde:

P - movimento permitido

N - movimento não permitido

P^I- permitido desde que o literal contenha somente caracteres numéricos

2.4.2 - TABELA DE PARÁGRAFOS

Esta tabela será construída na fase de análise, à medida que os rótulos são encontrados no programa fonte. A "TABELA DE PARÁGRAFOS" objetiva fornecer os endereços dos parágrafos nas instruções de desvio, bem como permitirá a solução do problema de referências futuras. Esta tabela será usada somente durante a análise, sendo seu espaço liberado após esta análise, ficando assim disponível para outros fins na fase de execução.

O programa fonte, na sua forma interna, será armazenado no disco, sendo cada parágrafo gravado em um novo setor. Assim na tabela de parágrafos, o endereço fornecido é o do setor do disco onde o parágrafo é armazenado. A tabela tem o seguinte formato:

RÓTULO	DEF	ENDEREÇO DISCO

2.4.2.1 - RÓTULO

O campo "RÓTULO" será formado de seis octetos onde ficará armazenado o nome do parágrafo.

2.4.2.2 - ENDEREÇO-DISCO

ENDEREÇO-DISCO é um campo formado de quinze bits, sendo usado para informar onde o parágrafo é mantido. Os dez pri-

meiros bits fornecem o número da trilha e os cinco seguintes informam o número do setor, que compõem o endereço.

2.4.2.3 - DEF

Este campo formado de um bit, informará se o rótulo já foi definido ou não,pela ocorrência como nome de parágrafo.

Deste modo, quando se encontra uma instrução de desvio, pesquisa-se sequencialmente a tabela de parágrafos até encontrar o rótulo, especificado nesta instrução de desvio e, se for definido, o substitue na instrução, pelo valor do endereço-disco da tabela. Isto é, constroe-se a forma interna da instrução colocando no segundo campo o conteúdo do endereço-disco. A forma interna da instrução de desvio seria:

CÓDIGO DA OPERAÇÃO	ENDEREÇO DE DESVIO

cada campo sendo formado de uma palavra.

Quando são usadas referências futuras, os nomes de parágrafos usados nas declarações de desvio são adicionados na tabela de parágrafos e o campo DEF indicará que se trata de rótulo ainda não definido. O campo ENDEREÇO-DISCO, neste caso, apontará para a última instrução que desvia para o rótulo citado.

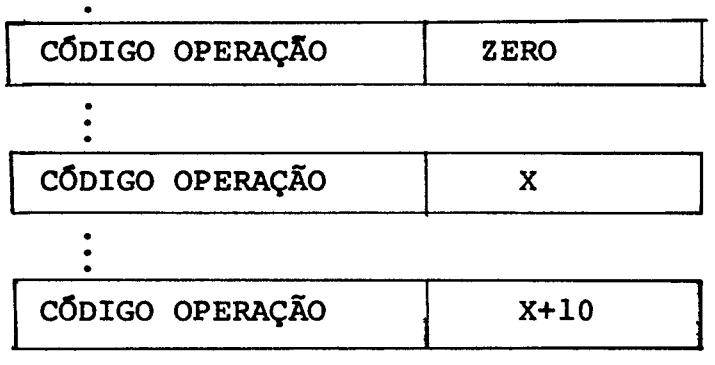
Sempre que surge uma nova instrução de desvio, para rótulo ainda não definido, pesquisa-se sequencialmente a tabela de parágrafos. Quando o rótulo é encontrado na tabela significa que ele já apareceu em outra instrução de desvio anterior e seu campo EN

DEREÇO-DISCO aponta para esta instrução. Assim, após o rótulo ser encontrado, o conteúdo do campo ENDEREÇO-DISCO da tabela será colocado no campo ENDEREÇO DE DESVIO da nova instrução e o ENDEREÇO-DISCO da tabela de parágrafos passará a apontar para esta nova instrução.

Quando é a primeira instrução fazendo referência futura, no campo ENDEREÇO DE DESVIO da instrução será colocado zero . Desta maneira, quando existem várias instruções desviando para um mesmo endereço que vem depois, estas instruções aparecem como uma lista com apontadores, como segue:

ENDEREÇO	INSTRUÇÃO
:	:
X	GO TO X+30
:	:
X+10	GO TO X+30
:	:
X+15	GO TO X+30
:	
X+30	

na forma interna ficaria:



no campo de endereço da tabela de parágrafos ficaria X+15, endereço da última instrução de desvio para aquele rótulo.

Quando o rótulo é encontrado, seu endereço será colocado na tabela de parágrafos e no campo de endereço de desvio de cada instrução onde este rótulo apareceu. Muda-se também o bit indicando que o rótulo foi definido. Assim se no caso anterior o rótulo foi encontrado no endereço X+30 , os campos de endereço de desvio das instruções e da tabela conteriam o endereço do setor do disco onde este parágrafo for alocado.

A tabela de parágrafos será utilizada, também, para detetar rótulos duplicados ou indefinidos.

2.4.3 - TABELA DE LITERAIS

A "TABELA DE LITERAIS" será construída à medida que são encontrados literais, quer numéricos ou não numéricos, no programa fonte. Os literais numéricos, somente serão especificados pela tabela, se não forem inteiros, valores inteiros serão descritos diretamente nas quádruplas e será visto em 2.4.5 .

O valor dos literais serão alocados no disco, visando economia de memória principal, e a tabela será mantida na memória, tendo o seguinte formato:

ENDEREÇO DISCO	TIPO	PICTURE

2.4.3.1 - ENDEREÇO DISCO

O campo "ENDEREÇO DISCO" será formado de 20 bits com o seguinte significado:

- Os oito primeiros bits fornecerão o número da trilha onde se encontra o literal.
- Os quatro bits seguintes informarão o número do setor na trilha.
- Os demais oito fornecerão o deslocamento, em relação ao início do setor, informando onde começa o literal.

2.4.3.2 - TIPO

"TIPO" é um campo formado por um bit informando o tipo de literal, se numérico ou não numérico.

2.4.3.3 - PICTURE

O campo "PICTURE" será formado de onze bits, sendo usado para informar as características do literal, da maneira seguinte:

a) Tipo numérico:
   - O primeiro bit do campo informará o sinal.
   - Os cinco bits seguintes informarão a quantidade de algarismos inteiros do literal
   - Os cinco bits que seguem fornecerão o número de algarismos da parte fracionária.

b) Tipo não numérico

   Os onze bits informarão a quantidade de caracteres alfanuméricos que formam o literal.

2.4.4 - TABELA DE TEMPORÁRIAS

A "TABELA DE TEMPORÁRIAS" será uma tabela objetivando informar o valor e formato das variáveis temporárias. Ela permanecerá na memória principal, sendo usada somente na fase de execução e será formada de campos.

PICTURE	VALOR

2.4.4.1 - PICTURE

"PICTURE" é um campo formado de uma palavra, 16 bits, que informará o comprimento e formação da variável, onde:

- o primeiro bit indicará o sinal.
- os oito bits seguintes fornecerão a quantidade de algarismos da parte inteira da variável.
- os sete últimos bits da palavra informarão o número de algarismos da parte decimal.

2.4.4.2 - VALOR

O campo "VALOR" tem comprimento de 18 bytes, para caber variável de qualquer tamanho permitido, isto é, entre 1 e 18 bytes. A variável ocupará sempre os primeiros bytes do campo, de acordo com o comprimento fornecido pela picture. Assim se a picture informa que uma variável temporária possue dez dígitos deve-se armazenar o valor nos dez primeiros bytes do campo "VALOR".

Na forma interna das instruções, quando um operando se refere a uma variável temporária, ele informará que se trata de uma variável temporária e fornecerá o deslocamento em relação ao início da tabela, onde ela se encontra.

Cada entrada na tabela terá um comprimento de 20 octetos , 2 octetos para o campo "PICTURE" e 18 octetos para o campo "VALOR". Limitando a cinco o número máximo de variáveis temporárias, para uma instrução do programa fonte, a TABELA DE TEMPORÁRIAS terá um tamanho de 100 octetos, 50 palavras.

2.4.5 - FORMA INTERNA DAS INSTRUÇÕES

A forma interna das instruções, como consta de outro trabalho de tese na COPPE intitulado ESTUDO DE UM SUPERVISOR PARA UM INTERPRETADOR COBOL, ela será apenas citada aqui.

Para forma interna foi escolhido quádruplas, que fornecem as operações na ordem em que as mesmas devem ser executadas, o que facilita em muito a sub-programação. Deste modo mantem-se no disco uma sub-rotina para cada operação, deixando-se na memória principal apenas aquela sub-rotina da operação a executar ou esta e as sub-rotinas mais solicitadas, caso haja espaço disponível. Assim, limita-se o espaço de memória principal usado durante a execução de um programa. Um programa supervisor, que também faz parte do trabalho citado no início deste parágrafo, fará a gestão de memória, alocando as sub-rotinas, e as liberando de acordo com as necessidades.

As quádruplas terão a seguinte forma:

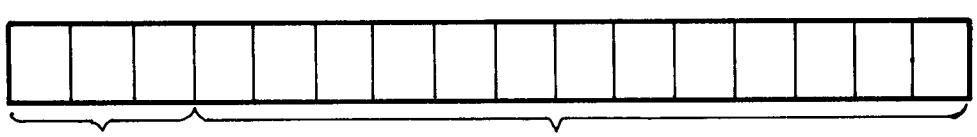
OPERAÇÃO, OPERANDO1, OPERANDO2;RESULTADO

Cada um dos campos de uma quádrupla terá o compri - mento de uma palavra.

O campo OPERAÇÃO conterá, evidentemente, o código da operação a ser executada.

Os demais campos terão o formato que segue, podendo ocorrer duas situações: na primeira, o conteúdo do campo representa um número inteiro, isto é, o próprio operando de uma instrução; na segunda este conteúdo representa um apontador para o operando. Tem-se a descrição destes dois casos:

a) Quando o campo é o próprio valor de um operando, número inteiro.



Este campo tem dois sub-campos:

- Os três primeiros bits informarão que o conteúdo dos bits seguintes se refere ao valor de um operando.
- Os bits seguintes fornecerão o próprio valor do operando, um número inteiro , desde que esteja entre zero e 8.191.

b) Quando o campo representa um apontador tem-se três sub - campos:

- Os três primeiros bits indicarão a tabela onde se encontra o operando, isto é, TABELA DE VARIÁVEIS, TABELA DE LI

TERAIS, TABELA DE PARÁGRAFOS, TABELA DE TEMPORÁRIAS ou TABELA DE ARQUIVOS.

- O bit seguinte informará se a variável é subscrita ou não.
- Os três bits seguintes informarão o tipo de picture, objetivando facilitar tanto as decisões a serem tomadas na fase de execução, como analisar o próprio campo "PICTURE" . Os tipos de "PICTURE" são os seguintes:

  000 - item grupo
  001 - item elementar alfabético
  010 - item elementar alfanumérico
  011 - item elementar numérico
  100 - item elementar alfanumérico de edição
  101 - item elementar numérico de edição I
  110 - item elementar numérico de edição II

  Estes tipos de "PICTURE" foram descritos em 2.3.4.3 .

- Os demais bits, nove, fornecerão o deslocamento, em relação ao início da tabela, onde se encontra o operando, ou onde ele é descrito. Deste modo pode-se endereçar no máximo 512 entradas em uma tabela.

Quando da formação das quádruplas, deve-se analisar a existência de variáveis subscritas e criar quádruplas que verifiquem, na fase de execução, se os índices estão coerentes com as dimensões destas variáveis, definidas na "DATA DIVISION", através da cláusula "OCCURS". Assim o valor de um índice não deve exceder ao valor do número de vezes dado na cláusula "OCCURS" que lhe é associado. Deste modo, se for considerada a segunda estrutura dada no pa

rágrafo 2.3.3 4 , os índices da variável E(I,J,K) , podem assumir no máximo os valores:

ÍNDICE	VALOR MÁXIMO
I	5
J	4
K	2

Para verificar estes limites, formam-se quádruplas à medida que os índices são calculados, na análise, que na fase de execução determinarão a validade destes índices, da maneira que segue. Logo após todas as quádruplas para o cálculo de um índice terem sido construídas, formam-se as quádruplas de teste de valor limite deste índice do seguinte modo:

```
-I,VL,T1
BMZ  X
```

onde I é o valor do índice calculado, o operando VL é um valor imediato fornecendo o limite deste índice, BMZ é a operação de desvio se T1 maior que zero e o operando X apontará para uma instrução que imprimirá uma mensagem de erro.

Nas operações com variáveis subscritas, deve-se formar as quádruplas de modo que se determine, em primeiro lugar, o valor dos índices. Para fazer isto, coloca-se as variáveis numa pilha e os operadores em outra, pilhas estas do tipo "LIFO". As operações são executadas segundo prioridades pré-estabelecidas, como notação polonesa por exemplo. Assim,tem-se:

DELIMITADOR	PRIORIDADE
( )	0
,	1
+ -	2
* /	3

Desta maneira, sempre que se encontra uma vírgula , ou parêntese de fechamento, todas as operações relacionadas à determinação de um índice são transformadas em quádruplas na fase de análise. Durante a execução o resultado será mantido em uma área especial da memória, reservada especialmente para este fim.

A área para os índices de uma variável será formada de três palavras, uma para cada índice, que serão chamadas IND1 , IND2 e IND3 . Portanto serão usados no máximo três subscritos e,evidentemente, não mais do que três níveis de "OCCURS". Será usado um contador para decidir em qual das palavras alocar o subscrito.

Calculados os índices, dos mesmos serão subtraídos 1 e posteriormente multiplicados pelos seus comprimentos, COMP1 , COMP2 e COMP3 vistas no parágrafo 2.3.3.4 , e depois somados os resultados, a fim de fornecer o deslocamento da variável subscrita desejada, em relação àquela de índices iguais a 1 . Para obter o deslocamento em relação à variável de número de nível 01, início do setor, adiciona-se ao deslocamento calculado até agora o valor do campo "DESLOC" da TABELA DE ESPECIFICAÇÕES correspondente à variável em consideração.

Também serão reservadas três palavras, de nomes DESL1, DESL2 e DESL3 , onde ficarão salvos os deslocamentos até sua utilização.

As palavras IND1, IND2 e IND3 são usadas para cada variável subscrita, isto é, após obter o valor do deslocamento(DESL1, DESL2 ou DESL3) de variável, IND1, IND2 e IND3 serão liberados para serem usados pela variável seguinte.

Como exemplo de geração de quádruplas para variáveis subscritas, considera-se a instrução fonte que se segue:

```
ADD   A(I/3-2,J),B(I+J/2,M-1)  TO  C(K) .
```

As palavras IND1 e IND2 serão usadas no cálculo de DESL1 que é usado com A . As palavras IND1 e IND2 serão usadas no cálculo de DESL2 que é usado com B . Usar-se-á IND1 para cálculo de DESL3.

As quádruplas para a instrução acima serão escritas a seguir, onde o número de índices será determinado pelo valor do contador utilizado para decidir em qual das palavras, IND1, IND2 ou IND3 colocar o subscrito.

```
QUÁDRUPLAS

/  I,3,T1
-  T1,2,T1
-  T1,VL ,T2
BMZ  Y
-  T1,1,IND1
-  J,VL,T1
BMZ  Y
-  J,1,IND2
*  IND1,COMP1 ,T1
*  IND2,COMP2,T2
```

```
+  T1,T2,T2
+  T2,DESLOC ,DESL1
/  J,2,T1
+  T1,I,T1
-  T1,VL,T2
BMZ  Y
-  T1,1,IND1
-  M,1,T1
-  T1,VL,T2
BMZ  Y
-  T1,1,IND2
*  IND1,COMP1,T1
*  IND2,COMP2,T2
+  T1,T2,T2
+  T2,DESLOC,DESL2
+  A,B,T1
-  K,VL,T2
BMZ  Y
-  K,1,IND1
*  IND1,COMP1,T2
+  T2,DESLOC,DESL3
+  T1,C,C
```

Significado de alguns operandos utilizados nas quádruplas:

- VL será um valor imediato, obtido na fase de análise da cláusula "OCCURS", informando o valor limite do subscrito utilizado.

- Y será um endereço de desvio para uma instrução que imprimirá uma mensagem de erro, comunicando que o índice é maior que o seu valor máximo permitido. O desvio se processará se o conteúdo do acumulador for maior do que zero (BMZ).

- COMP1 será um valor imediato obtido na fase de análise,como visto em 2.3.3.4 e informará o comprimento (dimensão) do índice.

- DESLOC será o valor obtido no campo de mesmo nome da tabela de especificação, visto em 2.3.4.2 , para a variável em consideração. Este campo fornece o deslocamento, em relação ao início do setor do disco, da variável de subscritos iguais a 1.

Quando são usadas as variáveis A,B e C nas palavras que as definem, nas quádruplas, emprega-se um bit para informar que se trata de variáveis indexadas. Quando um operando se refere a uma variável subscrita, a localização de seu valor no disco será obtida tomando-se o endereço do setor onde a variável se encontra e adiciona-se o valor de DESL1, DESL2 ou DESL3 dependendo do operando ser OPERANDO1, OPERANDO2 ou OPERANDO3 (RESULTADO) respectivamente.

2.4.6 - ORGANIZAÇÃO DO PROGRAMA FONTE NA MEMÓRIA

O programa fonte, em sua forma interna, será mantido no disco, sendo armazenado um parágrafo por setor. Na execução, o programa supervisor alocará na memória principal um setor de cada vez, em um "BUFFER". Os setores serão lidos do disco sequencialmen

te, a não ser que haja uma instrução desviando para um setor qual - quer.

Como é pouco provável que um parágrafo seja do tamanho de um setor do disco, 128 palavras, coloca-se uma marca de fim de setor, para informar que o setor seguinte deve ser lido ou um desvio deve ser efetuado, dependendo se a última instrução do parágrafo proporciona um desvio ou não.

Quando um parágrafo é maior que um setor do disco, ele será armazenado em setores seguidos e será colocado em cada setor uma marca de que o parágrafo continua no próximo setor, com exceção evidentemente do último setor que terá uma marca de fim de parágrafo.

Sempre que o supervisor lê um setor no "BUFFER", ele salvará o número deste setor lido, a fim de manter pista do próximo setor para ler ou fornecer endereço de desvio quando ocorre uma instrução desviando para o início do próprio parágrafo em execução.

CAPÍTULO III

FASE DE INTERPRETAÇÃO

3 - FASE DE INTERPRETAÇÃO

A fase dd interpretação, ou execução, constará de um programa supervisor e de várias sub-rotinas, uma para cada instrução.

O programa supervisor tomará as instruções do programa fonte, em sua forma interna, do disco, alocará as variáveis e sub-rotinas necessárias para executar cada instrução e fará a gestão de memória. Como já foi citado, este supervisor consta de outro trabalho de tese do Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ e não será estudado aqui.

As sub-rotinas serão armazenadas no disco, a fim de diminuir o uso de memória principal, sendo colocadas na memória central somente quando solicitadas pela operação correspondente. Caso haja espaço disponível, pode-se deixar mais de uma destas sub-rotinas na memória, procurando-se manter residentes aquelas mais utilizadas. Existirá uma sub-rotina para cada instrução.

Neste capítulo serão estudadas apenas as sub-rotinas, já que, como foi citado, o programa supervisor fará parte de outro trabalho.

Deste modo, no parágrafo 3.1 estudam-se as sub-rotinas que executam as operações aritméticas onde as variáveis serão representadas em decimal.

No parágrafo 3.2 serão vistas as operações de ENTRADA e SAÍDA , que usam sub-rotinas do próprio Monitor de Base para fazer as leituras e impressões.

Em 3.3 estuda-se a sub-rotina de PERFORM que tem por função fazer com que um grupo de comandos seja executado uma ou mais vezes como será visto posteriormente.

A operação IF , que na análise será substituída por uma instrução de desvio condicional, na fase de execução deixa de existir e estuda-se apenas as instruções de desvio criadas na análise, o que será feito em 3.4 .

A instrução de desvio incondicional, GO TO , será estudada no parágrafo 3.5 .

Nos parágrafos 3.6 e 3.7 serão vistas as sub-rotinas para as instruções OPEN e CLOSE , usadas para abrir e fechar arquivos, respectivamente.

A instrução STOP que apenas informa o fim do processamento do programa fonte será estudada em 3.8 .

EXIT , que é uma instrução que indica o fim de um grupo de comando executado por uma instrução PERFORM, estuda-se no parágrafo 3.9 .

A sub-rotina MOVE , que é a maior de todas, tem por função transferir uma cadeia de caracteres de uma área para outra da memória e será vista em 3.10 .

3.1 - OPERAÇÕES ARITMÉTICAS

Desde que COBOL é uma linguagem orientada para aplicações comerciais, é reduzido o número de operações aritméticas efetuadas com cada variável, muitas vezes efetua-se uma única operação

com quase todos os itens de dados. Deste modo, a conversão de decimal para binário para efetuar as operações aritméticas com as variáveis na base 2, e depois uma nova conversão, desta vez de binário a decimal para enviar para impressão, não se justifica. Diante disso as operações aritméticas, neste interpretador, serão processadas com as variáveis representadas em decimal. Outra razão para executar as operações com as variáveis em decimal é que em instruções "MOVE" , com item grupo que contém um ou mais itens elementares numéricos,estes itens numéricos devem estar representados em decimal (EBCDIC) antes de executarem a operação. Também em operações "MOVE" de um item numérico sem edição para um item numérico com edição, novamente os números devem estar representados em EBCDIC.

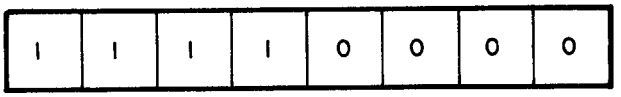
A seguir estuda-se inicialmente as sub-rotinas das operações aritméticas propriamente ditas, parágrafo 3.1.1, e depois a formação das pictures das variáveis temporárias, em 3.1.2 .

3.1.1 - SUB-ROTINAS

Como os minicomputadores não têm, em geral, instruções para operações decimais, esta lacuna deve ser suprida pelo interpretador. Deste modo serão programadas sub-rotinas que executarão as operações aritméticas em decimal. Estas sub-rotinas efetuarão cada operação como se ela fosse executada manualmente, isto é , algarismo por algarismo. A única diferente é a divisão, que será feita por subtrações sucessivas.

Já que os números serão armazenados no código EBCDIC, precisa-se encontrar o equivalente binário de cada algarismo, antes de operar com ele. Para isto reserva-se um byte da memória, conten

do:



Assim, carrega-se o algarismo desejado no acumulador e por meio de uma operação XOR, OR exclusivo, obtem-se o valor em binário deste algarismo no acumulador.

Encontrado o resultado da operação entre dois algarismos, este se encontra no acumulador em binário e deve ser descarregado no octeto apropriado do resultado, usando-se indexação, em EBCDIC. Para se conseguir este resultado, em EBCDIC, usa-se o mesmo octeto do caso anterior, mas empregando-se agora a operação OR em vez de XOR . Assim o conteúdo do acumulador após a operação OR com o byte da figura será sempre FX , onde X é um algarismo de 0 a 9 .

3.1.2 - VARIÁVEIS TEMPORÁRIAS

As variáveis temporárias terão suas pictures formadas em função das pictures das variáveis envolvidas na operação e da própria operação, sendo da seguinte maneira:

ADIÇÃO E SUBTRAÇÃO

A quantidade de algarismos decimais, parte fracionária, será igual à maior das partes fracionárias das variáveis que participam da operação.

O número de algarismos inteiros será igual ao maior

comprimento das partes inteiras das parcelas mais 1 .

MULTIPLICAÇÃO

Na multiplicação, o comprimento das partes fracionária e inteira da variável temporária resultante será obtido pela soma das partes fracionárias e inteiras dos fatores, respectivamente.

DIVISÃO

Quando a operação é divisão, o comprimento da parte inteira da variável temporária será igual à quantidade de algarismos da parte inteira do dividendo menos o número de algarismos da parte inteira do divisor.

A parte decimal terá o comprimento do campo fracionário da variável resultante da operação mais 2 , para permitir melhores precisões.

Um diagrama de blocos para cada sub-rotina será anexado no apêndice.

3.2 - ENTRADA E SAÍDA

As sub-rotinas de entrada e saída fazem uso de uma seção do Monitor de Base - MOB que executará estas operações, isto quando se trata de MITRA 15 . A sub-rotina será chamada pelo programa supervisor que fornecerá o tamanho do registro a ler ou imprimir, o nome do registro e o tipo de operação. A operação LEITURA , que torna um registro do arquivo disponível na memória para proces-

samento, será vista no parágrafo 3.2.1 e IMPRESSÃO, que fará a transferência do conteúdo de um registro para um periférico será estudada em 3.2.2 .

3.2.1 - LEITURA

A operação de leitura tem por função tornar disponível para processamento um registro de um determinado arquivo de entrada.

O formato da instrução será:

```
READ  nome-do-arquivo
```

Na linguagem LP 15 , para chamar a seção do MOB que executará a operação de entrada faz-se:

```
LDS : ARRAY   tamanho do registro BYTE  nome do registro
WORD  CB1
BYTE  CB2 = (00) ;
BYTE  CB3 = (M%EI);
WORD  CB4 = (@ . nome-do-registro)
WORD  CB5 = (tamanho do registro)

LPS :
AC: =@CB1
M%IO;
M%WAIT ;
:
FIN
```

onde na LDS define-se o bloco de controle. Ali, no BYTE CB2 será definida a operação de entrada-saída desejada; o conteúdo, ØØ , do byte, indica que a operação é uma leitura. O octeto seguinte,BYTE CB3 contém o equivalente numérico da etiqueta operacional, que representa uma função de entrada e saída. M%EI significa entrada de elementos binário ou alfanumérico, mais especificamente entrada de dados. Na palavra CB4 contém o endereço onde alocar, na execução,os octetos da informação a transferir. A palavra seguinte contém a quantidade de caracteres a transferir.

3.2.2 - IMPRESSÃO

A instrução de saída, WRITE, libera um registro para um arquivo de saída, tendo o seguinte formato:

```
WRITE  nome-do-registro
```

A seção do MOB para executar a instrução será chamada de modo semelhante àquela para LEITURA, mudando apenas alguns campos do bloco de controle. Assim temos:

```
LDS:ARRAY  tamanho do registro BYTE nome do registro
WORD BC1
BYTE BC2 = (80);
BYTE BC3 = (M%LO)
WORD BC4 = (@. nome do registro);
WORD BC5 = (tamanho do registro);

LPS:
AC: = @ CB1
```

```
M%IO ;
M%WAIT
:
FIN
```

onde no BYTE CB3 o valor 80 indica que a operação é de saída. No octeto seguinte, a etiqueta operacional M%LO , informa que se trata de uma saída de lista, alfanumérico.

3.3 - PERFORM

Esta instrução produz um desvio para um rótulo especificado e faz com que um grupo de comandos seja executado uma ou mais vezes dependendo das cláusulas da instrução, como será visto logo a seguir. Após a execução do grupo de comandos determinado pela operação é feito um desvio de retorno ao comando imediatamente após a instrução PERFORM .

O comando PERFORM é usado para executar um conjunto de instruções como se fosse uma sub-rotina e também para controlar a execução de laços.

O formato da instrução é:

```
PERFORM  nome-de-procedimento-1
         [THRU  nome-de-procedimento-2 ]
         [{nome-1          }        TIMES ]
          {literal-numérico-1}
```

Nome-de-procedimento-1 e nome-de-procedimento-2

devem ser nomes de parágrafos da "Procedure Division". Nome-1 deve ser um item numérico elementar descrito na "Data Division", como um inteiro. Literal-numérico-1 é um número inteiro.

A expressão entre colchetes significa que é opcional. Expressões entre parênteses indicam que se deve escolher uma destas expressões.

Quando usado somente nome-de-procedimento-1 na instrução, os comandos entre este rótulo e o próximo, isto é, o parágrafo iniciado em nome-de-procedimento-1 , serão executados e após o último comando do parágrafo, um desvio é feito a instrução imediatamente após o PERFORM.

Quando existe a cláusula : THRU nome-de-procedimento-2 , todos os comandos entre nome-de-procedimento-1 e nome-de-procedimento-2 , inclusive os comandos do parágrafo iniciado pelo último rótulo, serão executados e depois retorna-se para a instrução que segue ao PERFORM.

Quando a cláusula: nome-1 (ou literal-numérico-1) TIMES, está presente na instrução, indica que o grupo de comandos a serem executados terá sua execução repetida o número de vezes indicada pelo valor de nome-1 ou literal-numérico-1 .

A quádrupla da instrução PERFORM conterá:
código da operação, endereço do nome-de-procedimento-1, endereço do nome-de-procedimento-2 e o número de vezes que será executada.

À medida que um comando PERFORM é lido para execução, constroe-se uma pilha contendo as informações necessárias ao controle de sua execução. Cada elemento da pilha, que será do ti-

po 'último elemento que entra primeiro que sai' , será formado dos campos:

N-VEZES	P-INICIAL	P-CORRENTE	P-FINAL	END. RETORNO

Os campos desta pilha serão descritos a seguir.

3.3.1 - NVEZES

"NVEZES" é um campo formado de oito bits que informará quantas vezes a execução do conjunto de instruções, especificado pelo PERFORM, será repetida.

3.3.2 - P-INICIAL

P-INICIAL é um campo de doze bits que informará o endereço, no disco, do parágrafo para onde desviar a execução objetivando repetir o conjunto de operações quando NVEZES é maior do que 1 . Este campo informará o endereço do parágrafo inicial dado pelo rótulo nome-de-procedimento-1 da instrução.

3.3.3 - P-FINAL

P-FINAL é também um campo de doze bits que informará o endereço no disco do último parágrafo a ser executado pela instrução PERFORM. Será utilizado, juntamente com P-CORRENTE, pa-

ra terminar o campo de ação da instrução, desviando para o endereço seguinte ao do comando PERFORM ou para P-INICIAL caso NVEZES seja maior do que 1 .

3.3.4 - P-CORRENTE

P-CORRENTE é ainda um campo de doze bits que informará o endereço do parágrafo sendo executado. Servirá para determinar o fim da instrução, quando o parágrafo corrente coincide com o final. O término da instrução PERFORM ocorrerá somente após a execução de todo o parágrafo corrente. Este valor será constantemente atualizado à medida que novos parágrafos vão sendo encontrados.

3.3.5 - END-RETORNO

O campo "END-RETORNO" é composto de vinte bits,com o objetivo de informar o endereço de retorno após uma instrução PERFORM ser terminada. Este endereço representará o número da trilha e setor do disco, os doze primeiros bits, e o deslocamento em relação ao início do setor, os oito bits restantes.

Como se nota, cada elemento da pilha ocupa 64 bits, 4 palavras. Limitando a cinco o número de instruções PERFORM embutidas, a pilha ocupará uma área de 40 octetos. Um fluxograma para esta instrução está no apêndice.

3.4 - IF

Formato: IF identificador [NOT] { GREATER THAN / LESS THAN / EQUAL TO } { literal / identificador }

THEN declarações-imperativas { ELSE / OTHERWISE }

declarações-imperativas.

Quando da construção da forma interna, quádruplas , desta instrução, o IF desaparece e surgirá uma das instruções:

BZ - desvie se acumulador igual a zero

BMZ- desvie se acumulador maior que zero

BN - desvie se acumulador for negativo

BME- desvie se acumulador maior ou igual a zero

BNZ- desvie se acumulador for negativo ou zero

B - desvio incondicional; este desvio será usado após a execução das declarações que seguem o THEN ,para pular a primeira instrução após as declarações que seguem o ELSE.

A forma interna para cada destas instruções será do tipo:

OPERAÇÃO	OPERANDO

onde o campo de operando aponta para o endereço de desvio, isto é , será um deslocamento em relação ao início do parágrafo, fornecendo o endereço de desvio. Assim, como as instruções de um setor do disco são lidas em um "BUFFER", quando o supervisor encontra uma destas instruções de desvio ele tomará como instrução a ser executada aquela obtida tomando o endereço do "BUFFER" como base e o conteú-

do de OPERANDO como deslocamento.

Como o tamanho do "BUFFER" é 256 bytes, igual ao tamanho de um setor do disco, quando o operando tem um valor maior que 256, o endereço de desvio não estará no "BUFFER", mas em um setor posterior àquele colocado na memória.

Quando um setor é alocado na memória, seu endereço é salvado, em uma palavra reservada para este fim, permitindo assim tanto identificar o setor em execução, como aquele que lhe segue.

Deste modo, quando o conteúdo de OPERANDO é maior que 256, o setor seguinte é lido na memória e o conteúdo do operando é substituído por seu valor original, subtraído de 256, fornecendo o deslocamento dentro do "BUFFER".

Em cada das instruções de desvio, descarrega-se o conteúdo do acumulador numa variável, compara esta variável com zero e toma-se a decisão de acordo com a instrução.

3.5 - GO TO

Formato: GO TO nome-de-parágrafo

Na forma interna tem-se:

CÓDIGO OPERAÇÃO	OPERANDO

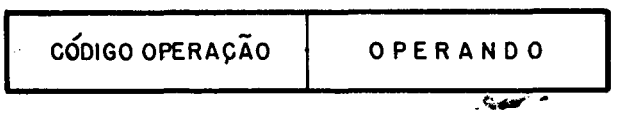
O campo de operando informará o endereço, número do setor do disco, para onde desviar, já que cada parágrafo será armazenado em um setor diferente. Desta maneira quando é lida esta instrução, o programa supervisor lerá no "BUFFER" o parágrafo que se

encontra no setor do disco endereçado pelo OPERANDO. Cuidado deve ser tomado quando o desvio for para o próprio parágrafo, já na memória. Como o endereço do setor alocado na memória é salvo , quando surge uma instrução GO TO ,primeiro testa se o parágrafo é o que já se encontra na memória; caso afirmativo, a execução é desviada para o início do "BUFFER", caso contrário será lido na memória o parágrafo endereçado pelo OPERANDO.

3.6 - OPEN

Formato: OPEN INPUT arquivo-1, arquivo-2 ...
OUTPUT arquivo-A, arquivo-B ...

Forma interna:



O campo "OPERANDO" é um apontador para a tabela de arquivos, onde este arquivo é definido.

A instrução OPEN reservará uma área de memória do tamanho do registro para cada arquivo onde serão alocados os dados quando da leitura dos arquivos.

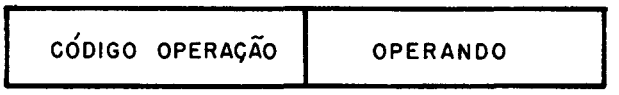
Em LP 15 reserva-se a área através de uma declaração de tabela. Assim, tem-se:

ARRAY tamanho-do-registro BYTE nome-registro

3.7 - CLOSE

Formato: CLOSE arquivo-1, arquivo-2,...

Forma interna:



Nesta instrução, o campo de operando também será um apontador para uma entrada na tabela de arquivos, vista em 2.2.3 , onde este arquivo é descrito.

A instrução CLOSE apenas liberará as áreas,"BUFFERS" de leitura e impressão, criadas pela OPEN.

3.8 - STOP

Formato: STOP RUN .

Forma interna:

A instrução "STOP" informa o fim do processamento do programa fonte, retornando o controle ao sistema. A sub-rotina para esta instrução apenas desvia para a declaração STOP do supervisor.

3.9 - EXIT

Formato: EXIT

Forma interna:

Esta instrução sempre será declarada entre dois nomes de parágrafos e tem por finalidade retornar a execução para o parágrafo inicial de uma instrução "PERFORM" ou para instrução seguinte ao "PERFORM" se a execução deste tiver sido terminada.

A sub-rotina EXIT apenas fará um desvio para P-INICIAL (I) ou END-RETORNO(I), dependendo de NVEZES(I)≠0 ou NVEZES(I)==0, respectivamente, onde:

P-INICIAL(I) - é o parágrafo de início da instrução "PERFORM" associada ao "EXIT" ;

END-RETORNO(I) - é o endereço de retorno, após a instrução "PERFORM" ter sido completada;

NVEZES(I) - é o número de vezes que o laço do "PERFORM" deve ter sua execução repetida.

Estes valores são verificados usando a pilha descrita em 3.3 .

3.10 - MOVE

Formato: MOVE { literal / identificador } TO identificador-2

Forma interna:

CÓDIGO OPERAÇÃO	OPERANDO - 1	OPERANDO - 2

A instrução "MOVE" é usada para transferir dados de uma área para outra da memória.

A sub-rotina "MOVE" é a mais extensa, dado a grande quantidade de opções.

Quando se tratar de movimentos simples, isto é, não há edições e os campos, tanto emissor como receptor, são alfabéti - cos ou alfanuméricos, que serão alinhados pela esquerda, a sub-rotina utiliza a instrução FOR do LP 15 que executa a transferência de octetos da esquerda para a direita. Quando o item receptor é maior que o emissor, os campos mais à direita serão completados com brancos, caso contrário, serão truncados.

Em movimentos de variáveis numéricas, o alinhamento se processa pela direita, se as variáveis são inteiras, ou se dará pelo ponto decimal no caso contrário.

Quando um "MOVE" utiliza variáveis numéricas inteiras, a sub-rotina lançará mão da instrução REPEAT do LP 15, que executa a transferência da direita para a esquerda.

Para variáveis numéricas, se o campo receptor for maior que o emissor, os campos mais à esquerda serão completados com zeros, caso contrário haverá truncamento.

Quando existe caracteres de edição, no campo da variável receptora, a transferência será efetuada caracter por caracter. No apêndice acompanhará um fluxograma para esta sub-rotina.

Cada campo de operando da forma interna, "OPERANDO1" e "OPERANDO2", aponta para a entrada de uma tabela, onde a variável é descrita.

Para executar a instrução "MOVE" , o primeiro ope-

rando é lido em um "BUFFER" na memória e a seguir os seus caracteres são transferidos para o endereço no disco indicado pelo OPERANDO2 .

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

BIBLIOGRAFIA

1. AWAD, M.ELIAS: "Business Data Processing", Third Edition , Prentice Hall,Inc - 1971 .

2. BOUHOT, JEAN-PIERRE: "COBOL Efficace" , L'Informatique, Nov/72.

3. CII: "MITRA 15 - Manuel de Présentation" - Compagnie Internationale pour l'Informatique - 1972.

4. DONAVAN, JOHN J.:"Systems Programming" , McGraw-Hill Book Company 1972.

5. GILES, P.:"Mini COBOL" , Computer Journal , August/1969.

6. GRIES,DAVID:"Compiler Construction for Digital Computers", John Wiley & Sons, Inc. - 1971.

7. IBM : "OS Full American National Standard COBOL", GC 28-6396-3 International Business Machine Corporation.

8. KATZAN JR., HARRY: "Advanced Programming" , Van Nostrand Reinhold Company, 1970.

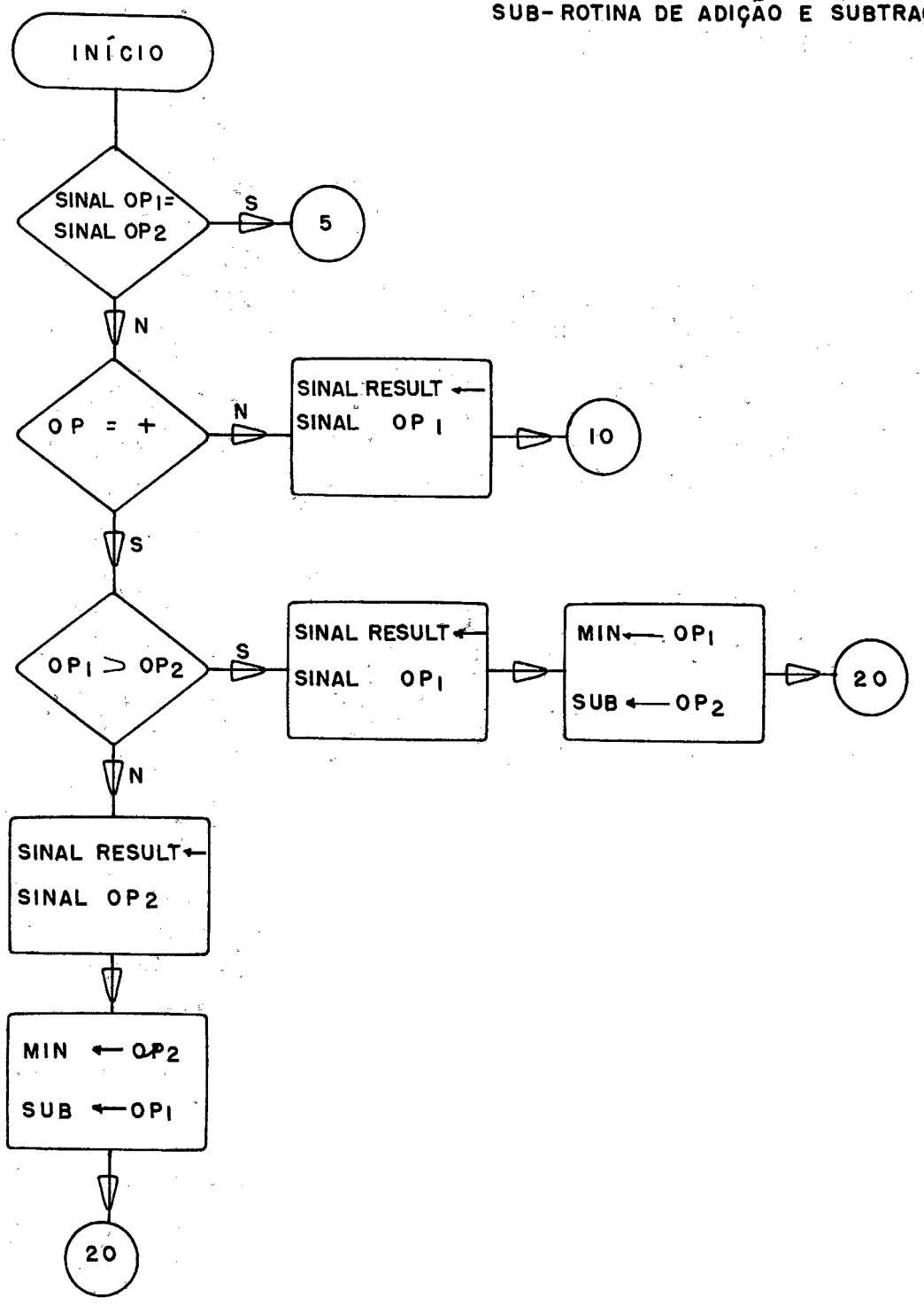
9. KNUTH,D.E. : "The Art of Computer Programming" - Vol.I, Addison-Wesley Publishing Company, 1968.

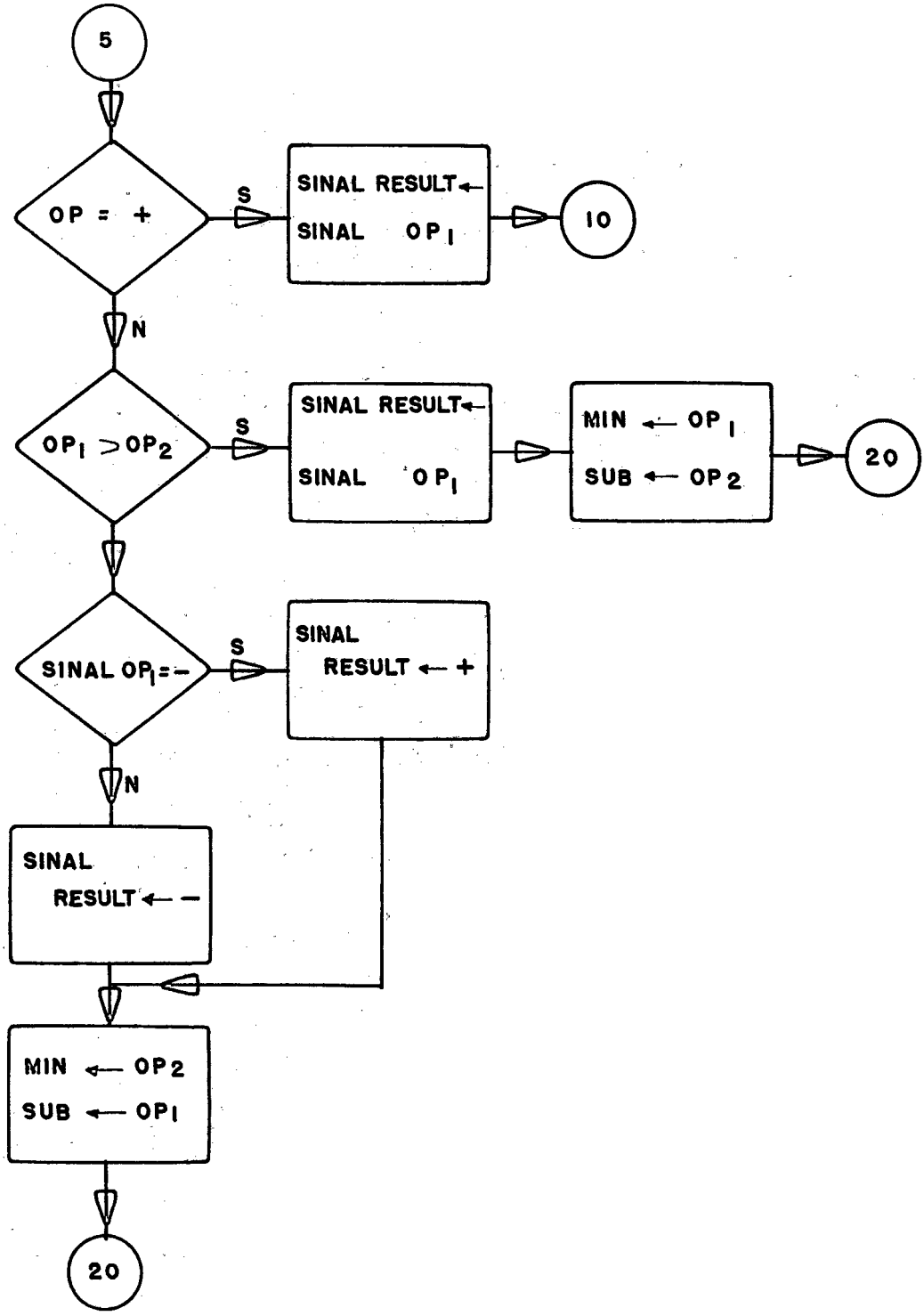
10. RANDELL,B. e RUSSEL,L.J. :"ALGOL 60 Implementation", Academic Press, 1964.

APÊNDICE

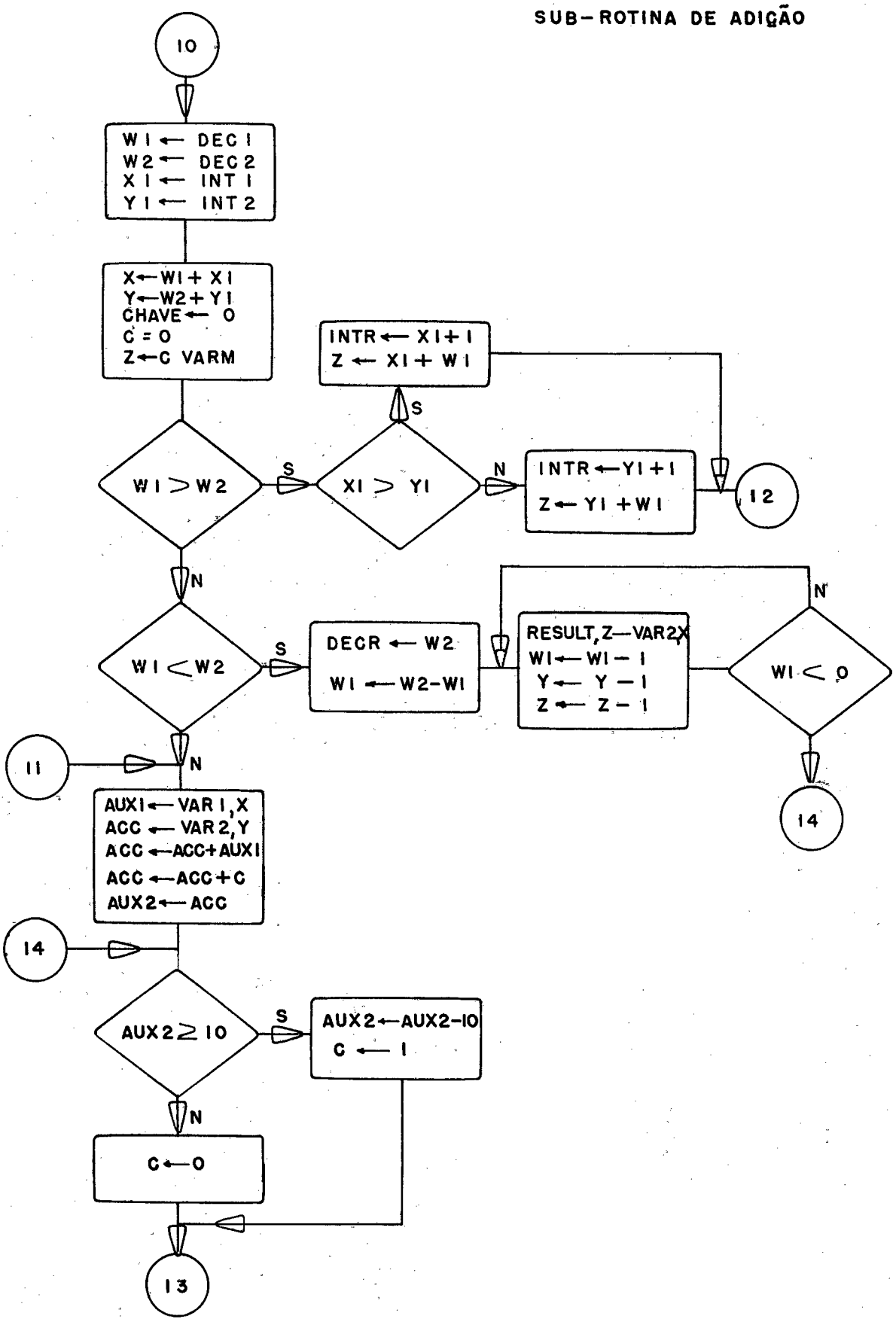
DIAGRAMAS DE BLOCOS DAS SUB-ROTINAS

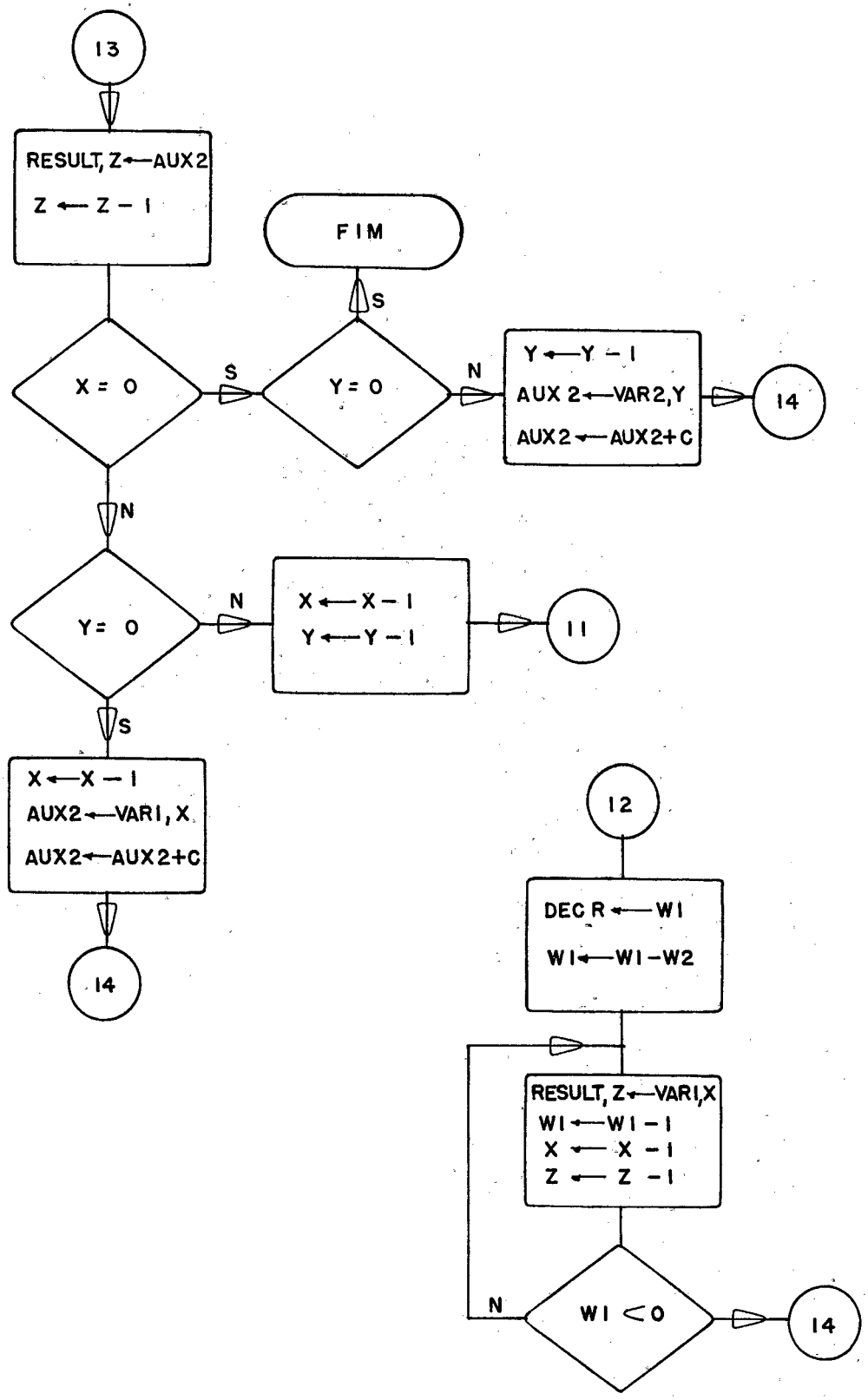
SUB-ROTINA DE ADIÇÃO E SUBTRAÇÃO



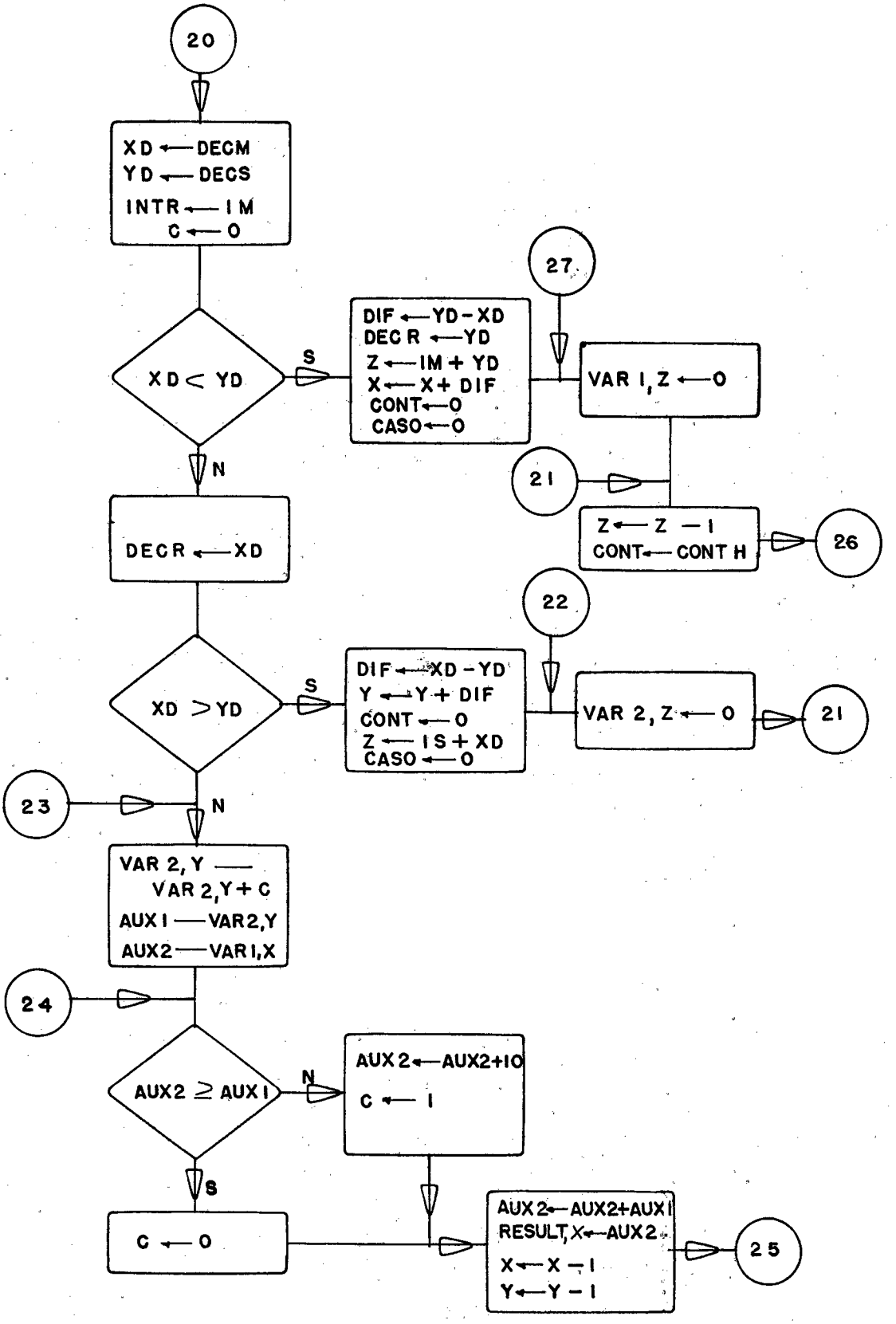

5
OP = +
S
SINAL RESULT ←
SINAL OP1
10
N
OP1 > OP2
S
SINAL RESULT ←
SINAL OP1
MIN ← OP1
SUB ← OP2
20
SINAL OP1 = −
S
SINAL
RESULT ← +
N
SINAL
RESULT ← −
MIN ← OP2
SUB ← OP1
20

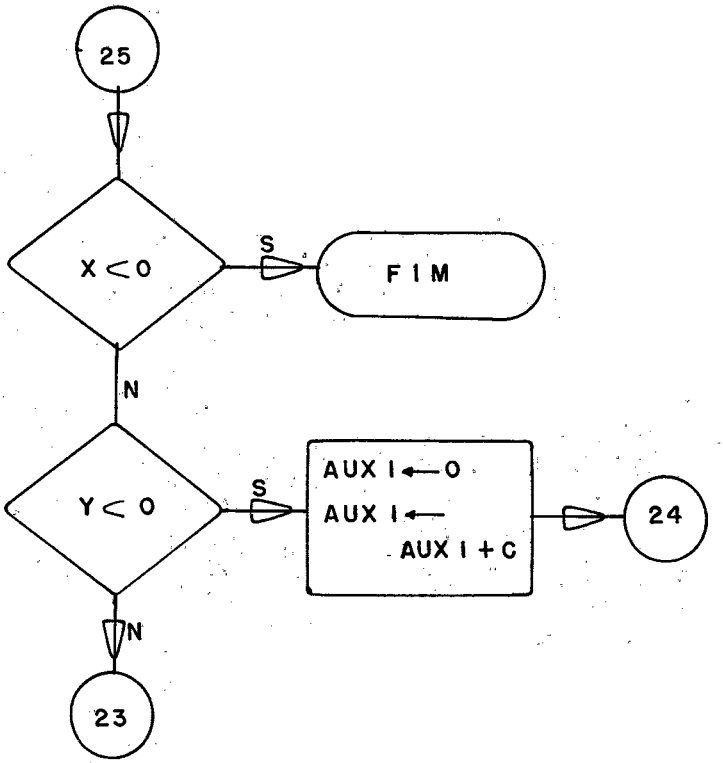
SUB-ROTINA DE ADIÇÃO

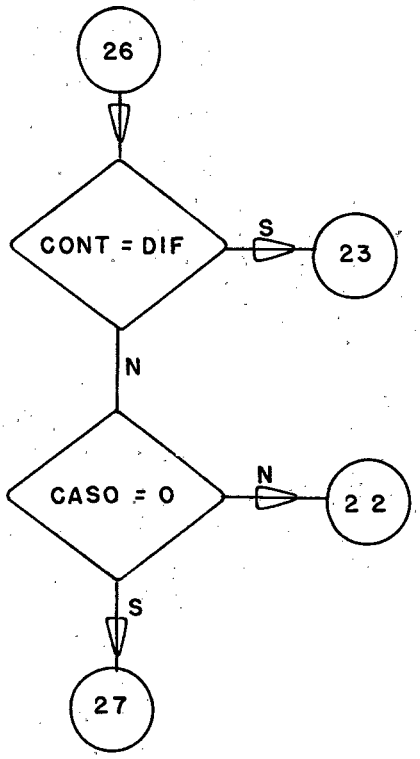


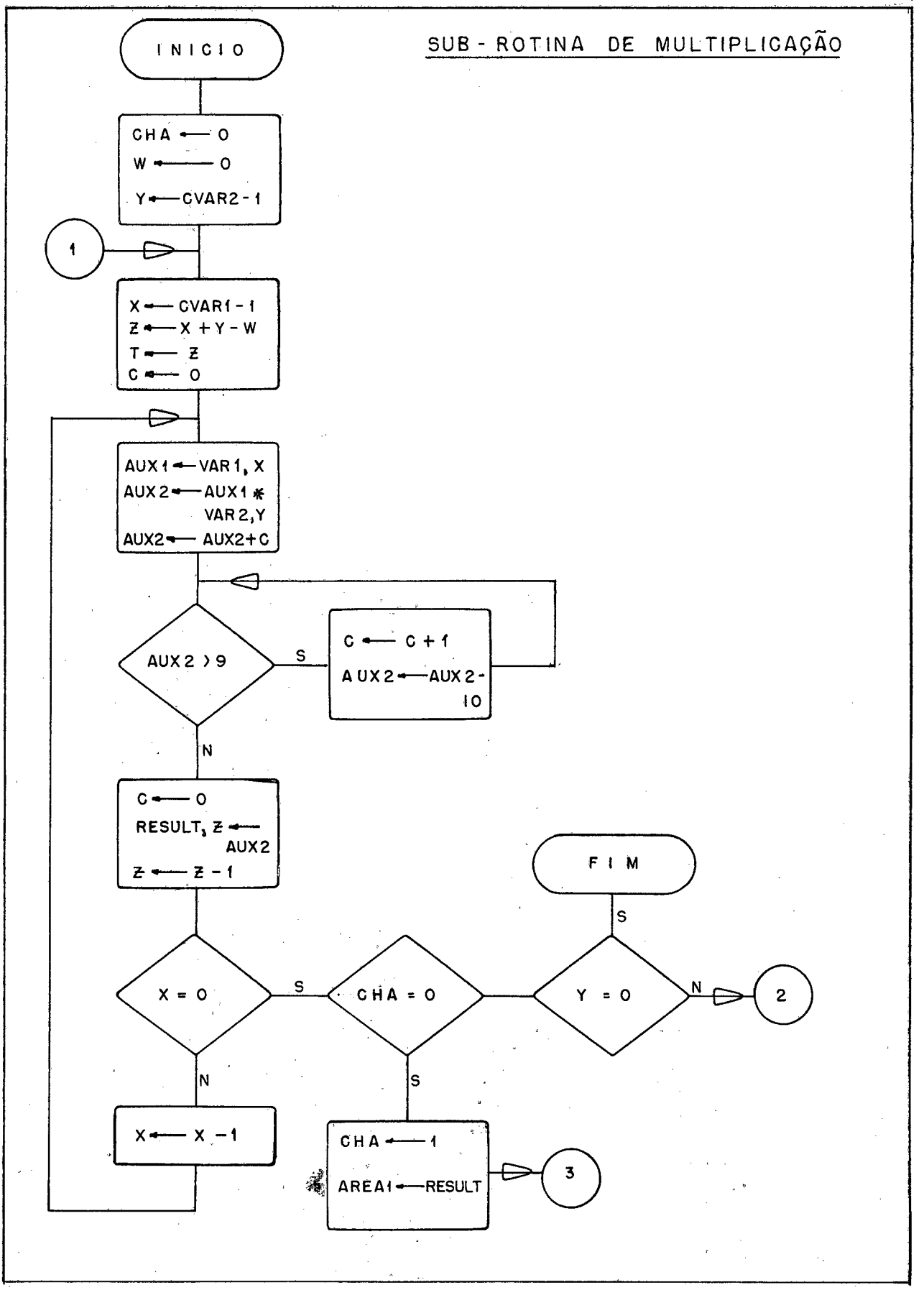

13
RESULT, Z←AUX2
Z ← Z − 1
FIM
S
X = 0
S
Y = 0
N
Y ← Y − 1
AUX 2←VAR2,Y
AUX2 ← AUX2+C
14
N
Y = 0
N
X ← X − 1
Y ← Y − 1
11
S
X ← X − 1
AUX2←VAR1, X
AUX2←AUX2+C
14
12
DEC R ← W1
W1←W1−W2
RESULT, Z←VAR1,X
W1 ← W1 − 1
X ← X − 1
Z ← Z − 1
N
W1 < 0
14

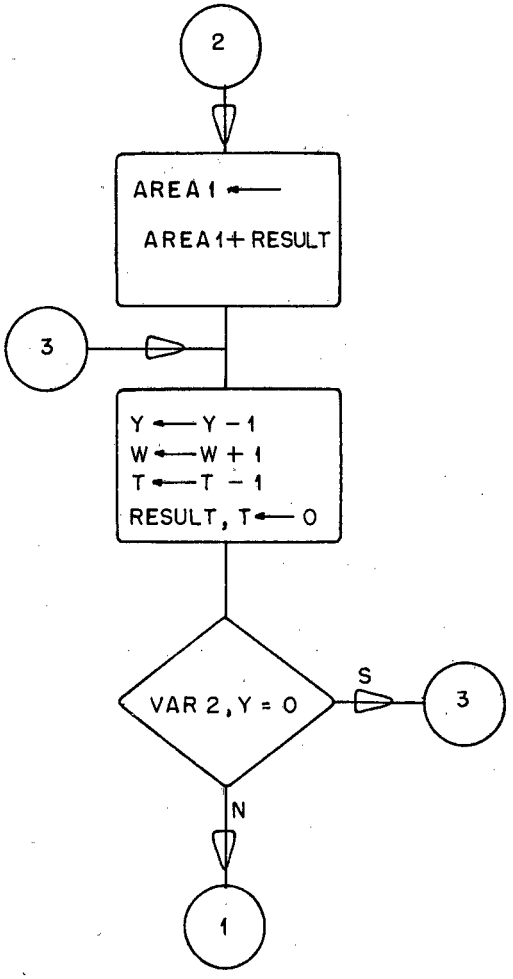
SUB-ROTINA DE SUBTRAÇÃO

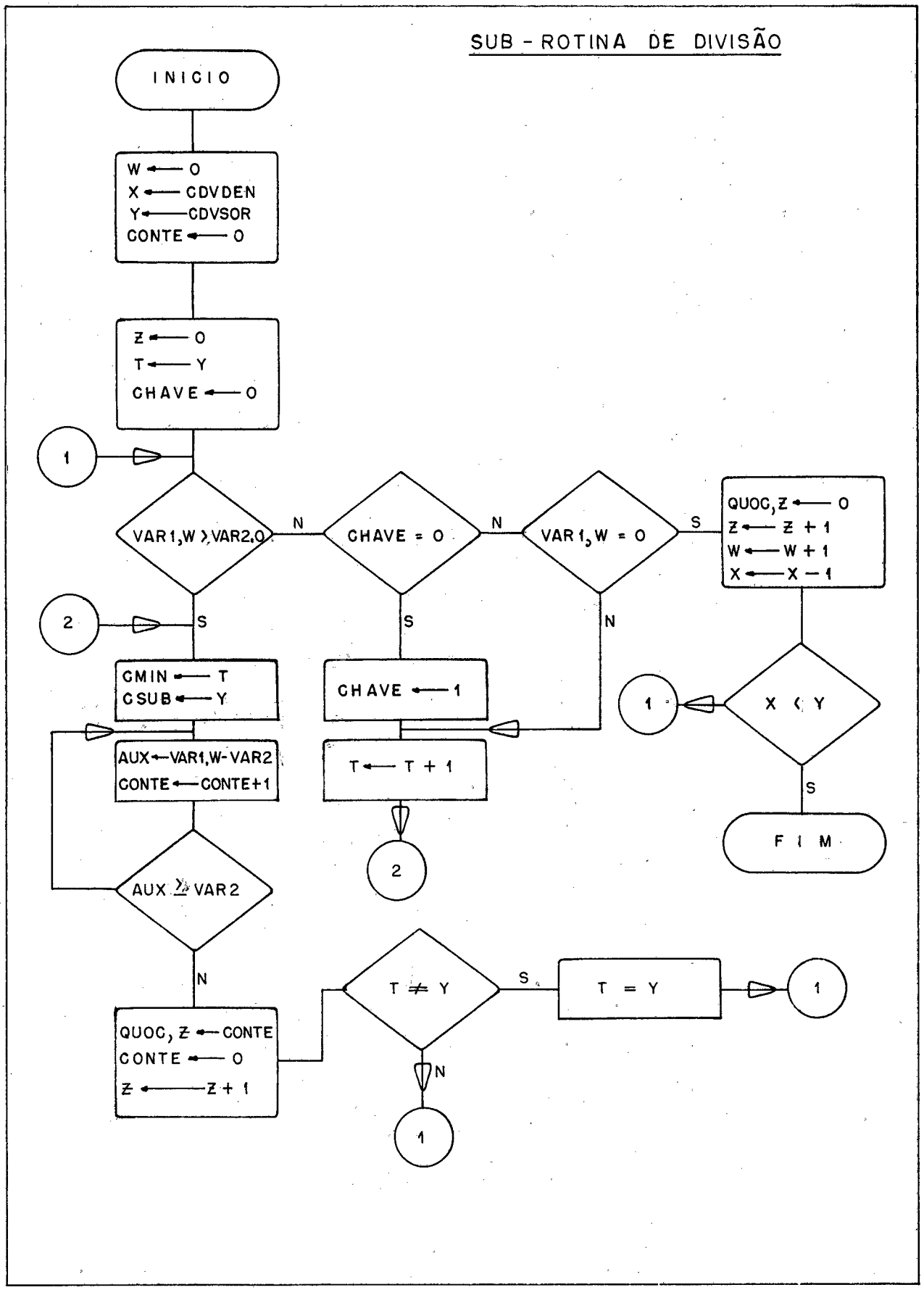



25
X < 0
S
F I M
N
Y < 0
S
AUX I ← 0
AUX I ←
AUX I + C
24
N
23

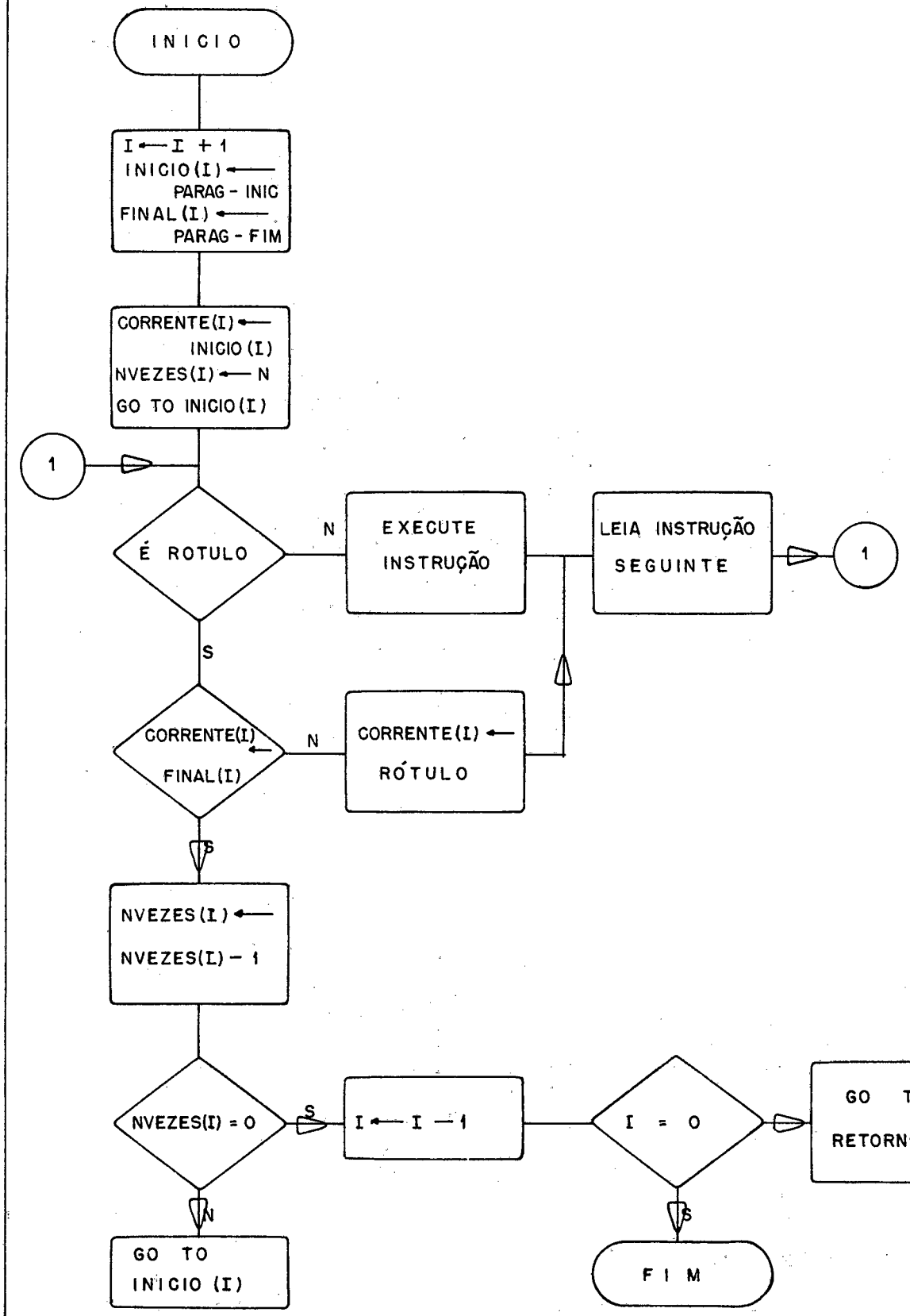

26
CONT = DIF
S
23
N
CASO = 0
N
2 2
S
27


SUB - ROTINA DE MULTIPLICAÇÃO
INICIO
CHA ← 0
W ← 0
Y ← CVAR2 - 1
1
X ← CVAR1 - 1
Z ← X + Y - W
T ← Z
C ← 0
AUX1 ← VAR1, X
AUX2 ← AUX1 * VAR2, Y
AUX2 ← AUX2 + C
AUX2 > 9
S
C ← C + 1
AUX2 ← AUX2 - 10
N
C ← 0
RESULT, Z ← AUX2
Z ← Z - 1
FIM
S
X = 0
S
CHA = 0
Y = 0
N
2
N
S
X ← X - 1
CHA ← 1
AREA1 ← RESULT
3

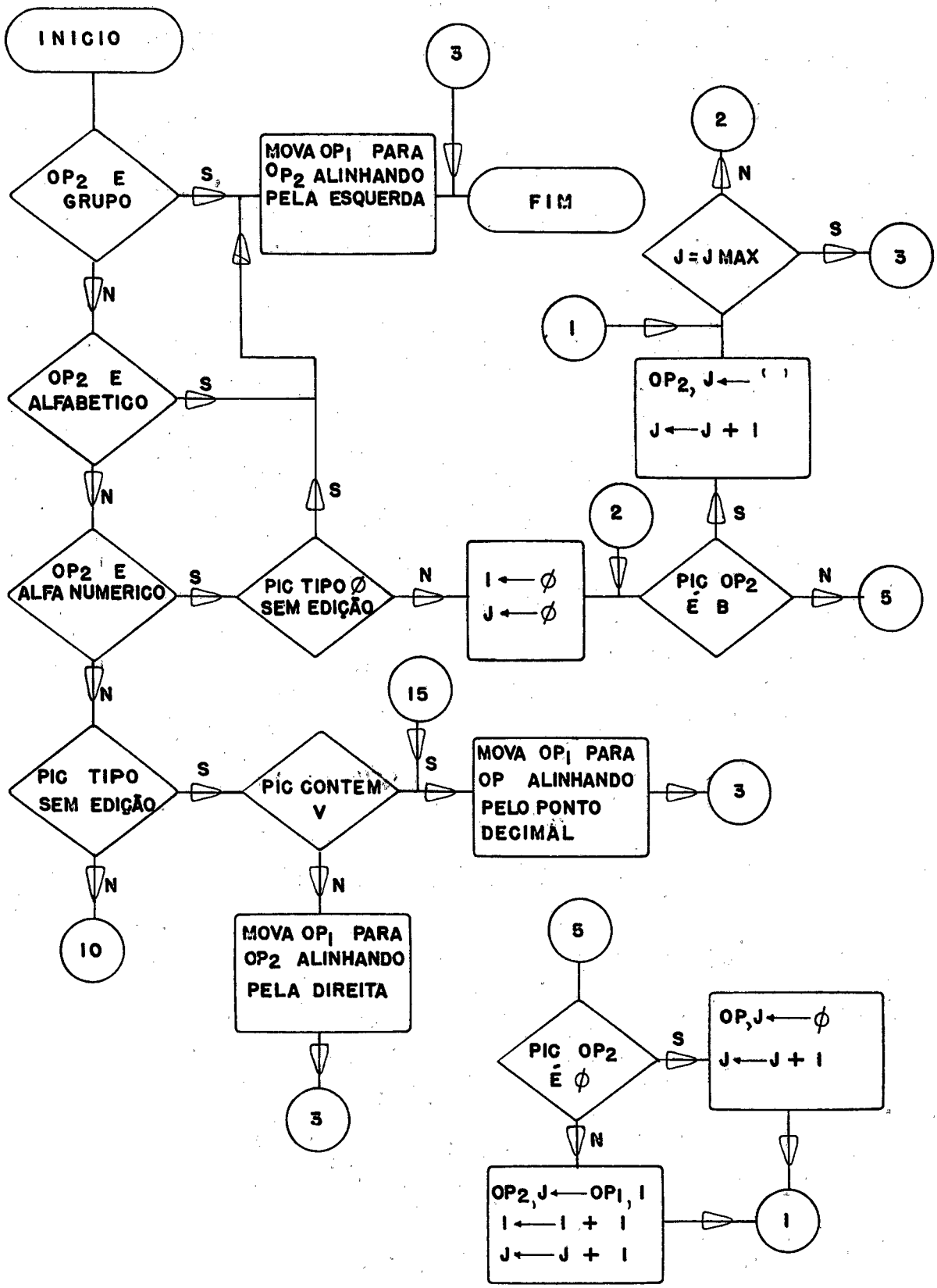

2
AREA 1 ←
AREA 1 + RESULT
3
Y ← Y − 1
W ← W + 1
T ← T − 1
RESULT, T ← 0
VAR 2, Y = 0
S
3
N
1

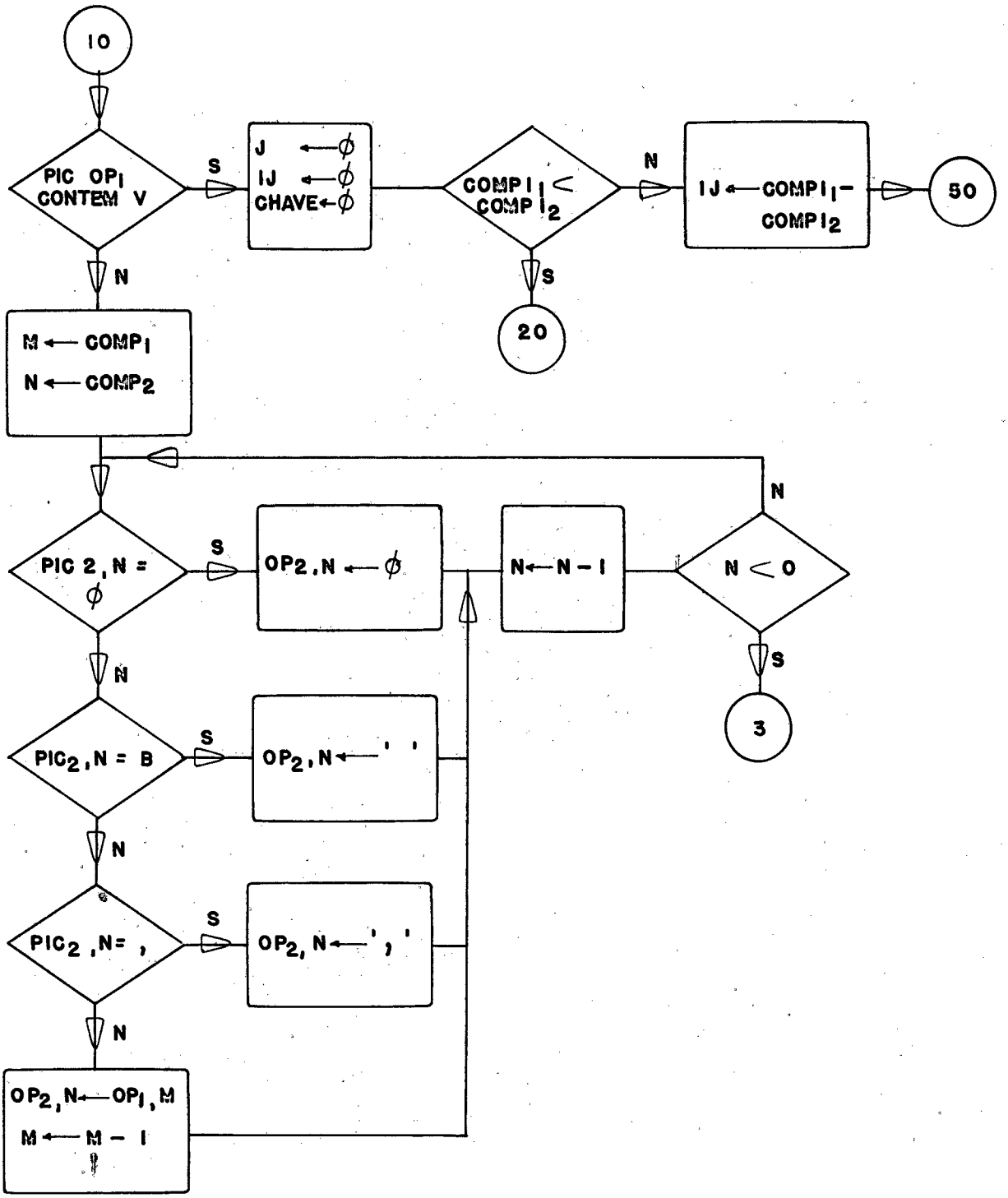

SUB - ROTINA DE DIVISÃO
INICIO
W ← 0
X ← CDVDEN
Y ← CDVSOR
CONTE ← 0
Z ← 0
T ← Y
CHAVE ← 0
1
VAR1,W ⩾ VAR2,0
N
CHAVE = 0
N
VAR1,W = 0
S
QUOC,Z ← 0
Z ← Z + 1
W ← W + 1
X ← X − 1
2
S
S
N
GMIN ← T
GSUB ← Y
CHAVE ← 1
1
X ⩽ Y
AUX ← VAR1,W-VAR2
CONTE ← CONTE+1
T ← T + 1
S
F I M
2
AUX ⩾ VAR2
N
T ≠ Y
S
T = Y
1
QUOC,Z ← CONTE
CONTE ← 0
Z ← Z + 1
N
1

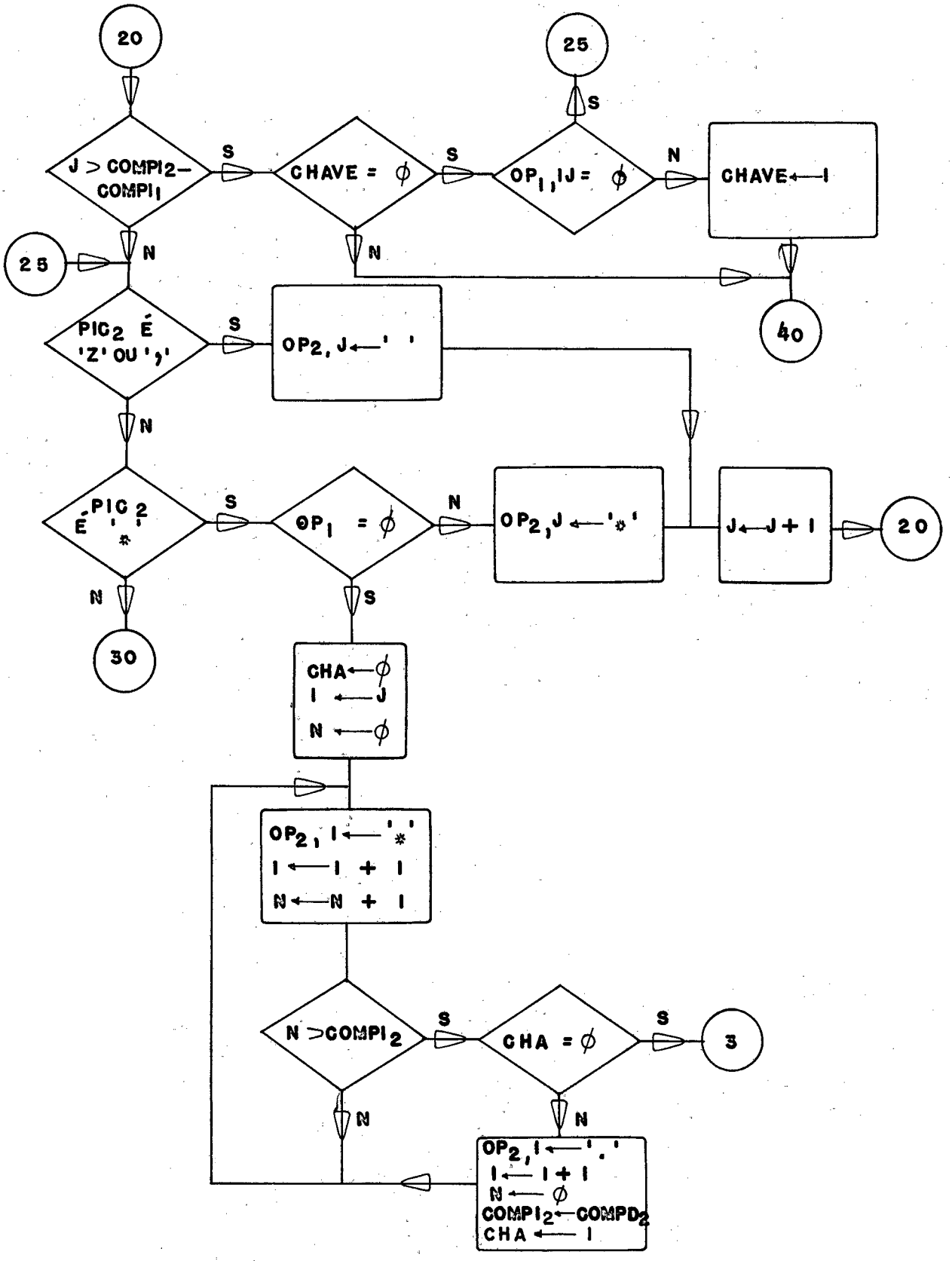
SUB - ROTINA PERFORM

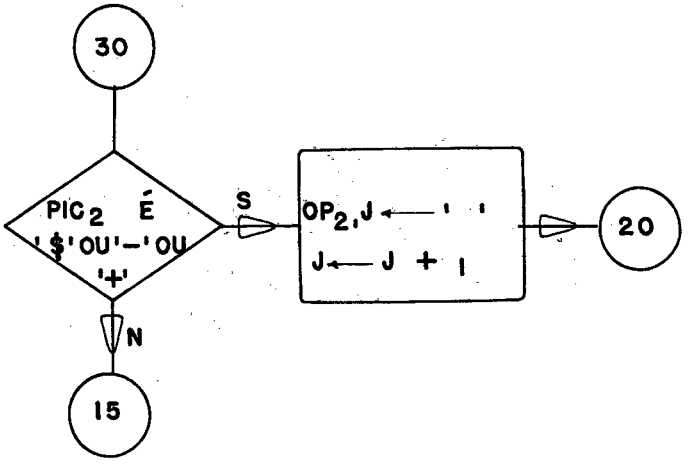


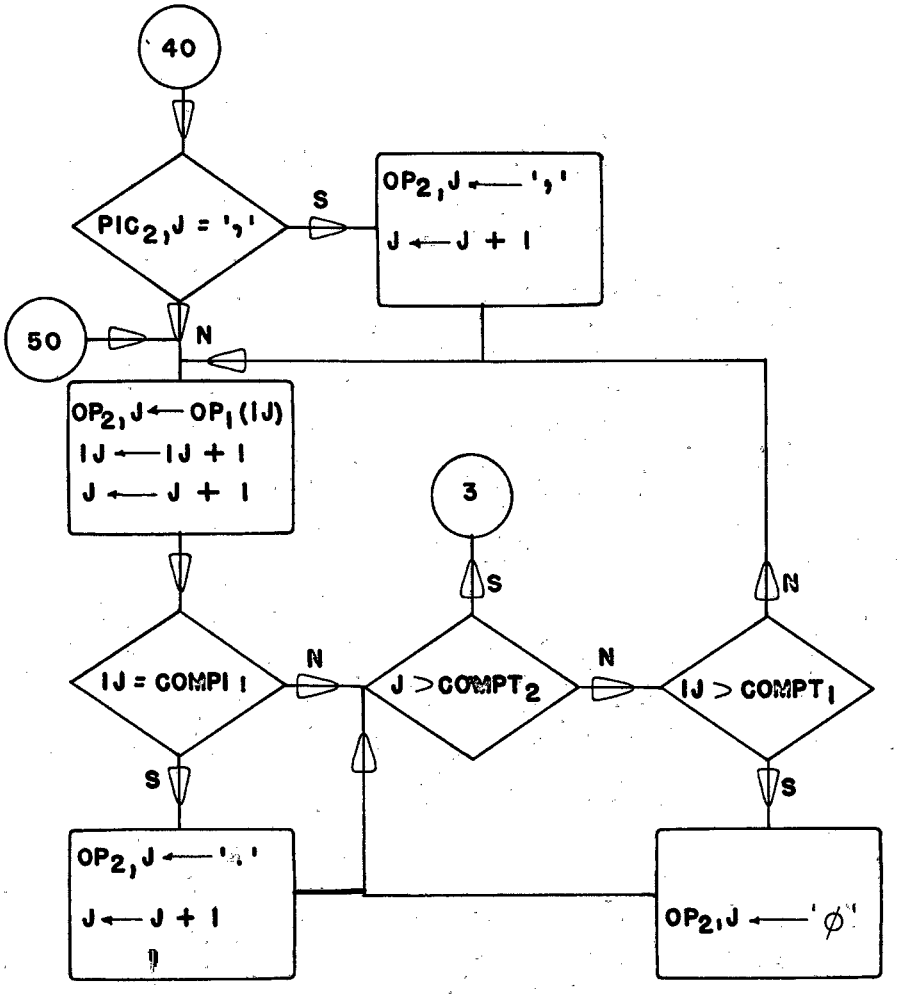
SUB – ROTINA MOVE




10
PIC OP1 CONTEM V
S
J ← ϕ
IJ ← ϕ
CHAVE ← ϕ
COMP1 1 < COMP1 2
N
IJ ← COMP1 1 − COMP1 2
50
S
20
N
M ← COMP1
N ← COMP2
PIC 2, N = ϕ
S
OP2, N ← ϕ
N ← N − 1
N < 0
N
S
3
N
PIC2, N = B
S
OP2, N ← ' '
N
PIC2, N = ,
S
OP2, N ← ', '
N
OP2, N ← OP1, M
M ← M − 1


20
J > COMPI2 − COMPI1
S
CHAVE = ϕ
S
25
S
OP1,IJ = ϕ
N
CHAVE ← I
25
N
N
40
PIC2 É 'Z' OU ','
S
OP2, J ← ' '
N
É PIC 2 '*'
S
OP1 = ϕ
N
OP2,J ← '*'
J ← J + 1
20
N
S
30
CHA ← ϕ
I ← J
N ← ϕ
OP2, I ← '*'
I ← I + 1
N ← N + 1
N > COMPI2
S
CHA = ϕ
S
3
N
N
OP2, I ← ' '
I ← I + 1
N ← ϕ
COMPI2 ← COMPD2
CHA ← I


30
PIC2 É '$' OU '-' OU '+'
S
OP2,J ← ' '
J ← J + 1
20
N
15


40
PIC2,J = ','
S
OP2,J ← ','
J ← J + 1
50
N
OP2,J ← OP1 (IJ)
IJ ← IJ + 1
J ← J + 1
3
S
N
IJ = COMPI 1
N
J > COMPT2
N
IJ > COMPT1
S
S
OP2,J ← ' '
J ← J + 1
OP2,J ← 'ϕ'