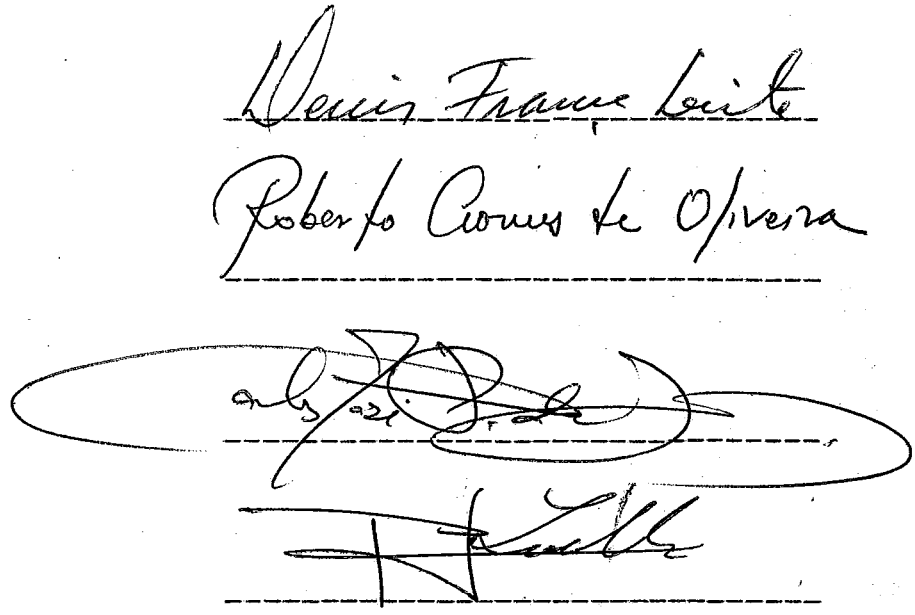
"ESTUDO E IMPLANTAÇÃO DE
UM COMPILADOR SNOBOL"

GUILHERME CHAGAS RODRIGUES

"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 OBTENCAO DO GRAU DE MESTRE EM CIENCIA (M.SC.)"

APROVADA POR :



RIO DE JANEIRO
ESTADO DA GUANABARA - BRASIL
JANEIRO DE 1971

À VERA.

ÍNDICE

AGRADECIMENTOS

AO PROFESSOR DENIS FRANCA LEITE PELA VALIOSA ORIENTAÇÃO. AOS COLEGAS DE TRABALHO LUCIANO PEREIRA, JOSÉ PAULO FAVILLA LOBO, GUY DAMM, PEDRO SALENBAUCH, MIGUEL ARANHA BORGES, LUIZ ANTONIO COUCEIRO, PAULO BIANCHI E JAYME LUIZ SZWARCFITER PELAS CRÍTICAS CONSTRUTIVAS E SUGESTÕES VALIOSAS APRESENTADAS NO DECORRER DESTE TRABALHO. A TODA A EQUIPE DO DEPARTAMENTO DE CÁLCULO CIENTÍFICO QUE ME AJUDOU NA REALIZAÇÃO DESTE TRABALHO. MEU AGRADECIMENTO ESPECIAL AO COLEGA YSMAR VIANNA E SILVA PELA VALIOSA COLABORACAO NA DETERMINAÇÃO DA ESTRUTURA DO COMPILADOR.

Guilherme Chagas Rodrigues
1971

RESUMO

O PRESENTE TRABALHO;DESCREVE AS DIRETRIZES SEGUIDAS NA IMPLANTAÇÃO DE UM COMPILADOR SNOBOL PARA O COMPUTADOR 1130.

É DISCUTIDA TÔDA A ESTRUTURA E ORGANIZAÇÃO DO SISTEMA DO PONTO DE VISTA PRÁTICO, VISANDO A SUA IMPLANTAÇÃO A CURTO PRAZO.

SÃO TAMBÉM APRESENTADAS A ESTRUTURA E A LÓGICA DO SISTEMA, TANTO NA FASE COMPILAÇÃO COMO NA FASE EXECUÇÃO, COM UMA DESCRIÇÃO SUCINTA DAS ROTINAS PRINCIPAIS.

ALGUMAS FACILIDADES QUE NÃO FORAM IMPLANTADAS NO SISTEMA SNOBOL ORIGINAL SÃO DISCUTIDAS DE MANEIRA SUPERFICIAL.

Guillermo Chaga, Rodrigo
1971

ABSTRACT

THE PRESENT WORK DESCRIBES THE DESIGN OF A SNOBOL COMPILER FOR THE IBM 1130.

THE STRUCTURE AND ORGANIZATION OF THE SYSTEM IS DISCUSSED FROM A PRACTICAL VIEW POINT WITH THE OBJECTIVE OF MAKING IT READY IN A SHORT TIME.

THE STRUCTURE AND LOGIC OF THE COMPILATION PLASE AND THE RUN-TIME ENVIROMENT IS PRESENTED WITH A CONCISE DESCRIPTION OF THE MAIN ROUTINES.

SOME FACILITIES OF SNOBOL 3 NOT IMPLEMENTED ARE ALSO DISCUSSED.

INTRODUÇÃO

I- HISTÓRICO DA LINGUAGEM:

ORIGINALMENTE, O COMPUTADOR DIGITAL FOI CONSTRUIDO E USADO APENAS PARA PROGRAMAÇÃO NUMÉRICA.

COM O TEMPO, AS TÉCNICAS DE PROGRAMAÇÃO SE DESENVOLVERAM E O COMPUTADOR EVOLUIU DE UM PROCESSADOR NUMÉRICO PARA UM PROCESSADOR DE SÍMBOLOS.

ESTA APLICAÇÃO OPERA BASICAMENTE SOBRE LISTAS E SURGIU DO DESENVOLVIMENTO DE TRABALHOS SOBRE TEMAS COMO DIFERENCIAÇÃO E INTEGRAÇÃO ANALÍTICA (KENNENY,1969), JOGO DE XADREZ (BERNSTEIN,1958), PROVA DE TEOREMAS (DUNHAN, GILMORE E WANG) E OUTROS PROBLEMAS QUE ENVOLVIAM A MANIPULAÇÃO DE CADEIAS.

COMEÇARAM ENTÃO A SURGIR VÁRIAS LINGUAGENS QUE FACILITASSEM A PROGRAMAÇÃO DESTES TIPOS DE PROBLEMAS, DENTRE AS QUAIS ENCONTRAM-SE IPL-V, LISP, COMIT E OUTRAS.

APÓS ALGUNS ANOS DE EXPERIÊNCIA COM COMIT, ALGUMAS DE SUAS DEFICIÊNCIAS TORNARAM-SE BEM CLARAS. DENTRE ELAS AS MAIS SIGNIFICANTES SÃO A IMPOSSIBILIDADE DE DAR NOME A UMA CADEIA OU DE REALIZAR OPERAÇÕES ARITMÉTICAS CONVENIENTEMENTE. ESTE FOI O MOTIVO QUE LEVOU D.J.FARBER, R.E.GRISWOLD E J.P.POLONSKY, EM 1962, A DESENVOLVEREM A LINGUAGEM SNOBOL (STRING ORIENTED SIMBOLIC LANGUAGE) NO BELL TELEPHONE LABORATORIES.

A LINGUAGEM TORNOU-SE POPULAR ENTRE CERTOS GRUPOS, PRINCIPALMENTE EM UNIVERSIDADES, O QUE LEVOU OS AUTORES DA MESMA A REALIZAR CERTAS MELHORIAS QUE RESULTARAM NA CRIAÇÃO DO SNOBOL 3, POR VOLTA DE 1966.

NOVAS MELHORIAS FORAM ADICIONADAS AO SNOBOL 3, RESULTANDO, POR VOLTA DE 1967, O SNOBOL 4.

SENDO SNOBOL UMA LINGUAGEM DE ALTO NÍVEL E DE MUITOS RECURSOS, ELA E INERENTEMENTE LENTA DEVIDO AO FATO DE OPERAR SOBRE CADEIAS. ISTO A TORNA POUCO ACEITAVEL PARA PROCESSAMENTO DE GRANDES CADEIAS COMO POR EXEMPLO PARA A ANÁLISE DE TEXTOS LITERÁRIOS PARA DETERMINAÇÃO DE ESTILO. ESTE TIPO DE PROCESSAMENTO E POUCO RECOMENDÁVEL PARA A LINGUAGEM DEVIDO AO TEMPO DE COMPUTAÇÃO.

POR OUTRO LADO, DEVIDO AOS RECURSOS QUE APRESENTA, SIMPLIFICA UMA PROGRAMAÇÃO COMPLEXA COMO COMPILAÇÃO, INTELIGÊNCIA ARTIFICIAL E AS APLICAÇÕES CITADAS ACIMA.

EM SUMA, SNOBOL É UMA LINGUAGEM QUE SE PRESTA A UM PROCESSAMENTO COMPLEXO SOBRE PEQUENAS CADEIAS, SENDO MUITO ÚTIL AOS QUE SE DEDICAM A CIÊNCIA DA COMPUTAÇÃO.

II- HISTÓRICO DA TESE:

A NOSSA IDEIA ORIGINAL FOI DESENVOLVER NESTE TRABALHO, UM ESTUDO TEÓRICO SOBRE TECNICAS DE COMPILAÇÃO APLICADAS A UM COMPILADOR SNOBOL.

MAS, APÓS POSTERIORES ENTENDIMENTOS COM O NOSSO

ORIENTADOR, DECIDIMOS REALIZAR TAMBÉM A IMPLANTAÇÃO DO SISTEMA, AINDA QUE RESTRITO EM SUA POTENCIALIDADE, NO COMPUTADOR DA COPPE QUE É UM IBM 1130.

ESTA DECISAO FOI TOMADA LEVANDO-SE EM CONSIDERAÇÃO QUE A REALIZACAO DO SISTEMA NOS DARIA UMA EXPERIÊNCIA MAIOR NO ASSUNTO, JÁ QUE CERTOS ASPECTOS DA COMPILAÇÃO SÃO DE ORDEM EXTRITAMENTE PRÁTICA E CERTOS PROBLEMAS SÓ APARECEM NA HORA DE SE ESCREVER O PROGRAMA.

OUTRO INCENTIVO QUE TIVEMOS PARA REALIZAR A IMPLANTAÇÃO DO SISTEMA, FOI O FATO DO COMPUTADOR 1130 SER UMA MÁQUINA DE USO BASTANTE DIFUNDIDO NAS UNIVERSIDADES NO BRASIL, DEVIDO A DISPONIBILIDADE ECONOMICA BRASILEIRA. COMO SABEMOS, O SISTEMA 1130 FOI PROJETADO PARA CÁLCULOS DE ENGENHARIA, NÃO POSSUINDO PORTANTO FACILIDADES QUE INCENTIVEM A PESQUISA NO CAMPO DA CIÊNCIA DA COMPUTAÇÃO.

MAS O PRINCIPAL MOTIVO QUE NOS LEVOU A EXECUÇÃO DO PROJETO FOI O FATO DE ESTAR PREVISTA A IMPLANTACAO DE UM CURSO DE CIENCIA DA COMPUTAÇÃO NA COPPE. NESTA EPOCA, O UNICO COMPUTADOR QUE POSSUÍAMOS ERA O IBM 1130 E A REALIZACAO DO COMPILADOR DARIA UMA ÓTIMA FERRAMENTA DE TRABALHO AOS ALUNOS DO CURSO.

NOSSA FINALIDADE AO IMPLANTAR O SISTEMA, FOI A DE REALIZAR O TRABALHO A CURTO PRAZO. DESTA FORMA FORAM IMPOSTAS CERTAS RESTRICOES AO SISTEMA, COMO O USO DE FUNÇÕES QUE NÃO FOI IMPLANTADO.

POR ESTE MOTIVO, MUITAS VEZES, A OPÇÃO SEGUIDA FOI A DE FACILIDADE DE PROGRAMAÇÃO EM DETRIMENTO DE UMA ECONOMIA DE MEMÓRIA OU VELOCIDADE DE COMPUTAÇÃO.

AO NOS REFERIRMOS A LINGUAGEM SNOBOL, ESTAMOS NOS REFERINDO IMPLICITAMENTE AO SNOBOL 3 QUE FOI A LINGUAGEM QUE TOMAMOS POR BASE NA REALIZACAO DESTE TRABALHO.

COMO A ABNT É AINDA OMISSA NA PARTE DE TERMINOLOGIA EM CIÊNCIA DA COMPUTAÇÃO, APRESENTAMOS UM PEQUENO GLOSÁRIO NO APÊNDICE III, AO QUAL RECOMENDAMOS A CONSULTA ANTES DA LEITURA DESTE TRABALHO.

PARTE I

FILOSOFIA E ESTRUTURA GERAL

1.1- FILOSOFIA:

A DIRETRIZ PRINCIPAL QUE REGE ESTE TRABALHO PODE SER EXPRESSA EM UMA SÓ FRASE: FACILIDADE DE PROGRAMAÇÃO.

ISTO PORQUE O TRABALHO DE PROGRAMAÇÃO QUE SE TEM COM UM SISTEMA DESTE PORTE E CONSIDERÁVEL E NOSSA INTENÇÃO FOI TERMINA-LO EM CURTO ESPAÇO DE TEMPO.

DESTE MODO, ALGUMAS OTIMIZAÇÕES SÃO LEGADAS A SEGUNDO PLANO EM FAVOR DE UMA MAIOR FACILIDADE DE PROGRAMAÇÃO.

ALÉM DISTO, É DESEJÁVEL TAMBÉM UMA CERTA MODULARIDADE NO SISTEMA, DE MODO A QUE NÃO FIQUE RÍGIDO E DE DIFÍCIL MODIFICAÇÃO. ISTO PERMITE TAMBÉM QUE SE ESCREVA AS PARTES MAIS DIFÍCEIS EM UMA LINGUAGEM DE ALTO NÍVEL PARA UMA PRIMEIRA IMPLANTAÇÃO, TENDO-SE ASSIM, EM CURTO ESPAÇO DE TEMPO, O COMPILADOR JÁ FUNCIONANDO PARA UM ESTUDO MAIS DETALHADO DO MESMO.

UMA VEZ IMPLANTADO O SISTEMA EM SUA PRIMEIRA CONFIGURAÇÃO, PODE-SE TER UMA NOÇÃO MELHOR DE QUAIS AS PARTES QUE DEVEM SER APRIMORADAS E, DEPENDENDO DO INTERESSE DO USUÁRIO, QUAIS OS RECURSOS QUE PODEM SER ADICIONADOS.

1.2- ESTRUTURA GERAL:

O SISTEMA É DIVIDIDO BASICAMENTE EM DUAS FASES, FASE COMPILAÇÃO E FASE EXECUÇÃO.

NA REALIDADE, EXISTEM AINDA MAIS DUAS FASES MENORES. UMA QUE INICIALIZA A FASE COMPILAÇÃO E UMA INTERMEDIÁRIA ENTRE A COMPILAÇÃO E A EXECUÇÃO.

NA FASE COMPILAÇÃO, É LIDO O PROGRAMA FONTE, LISTADO SE NECESSÁRIO, GERADO O PROGRAMA OBJETO, MONTADA A TABELA DE SÍMBOLOS E ALOÇADA AS CONSTANTES E VARIÁVEIS.

NA FASE EXECUÇÃO O CONTRÔLE DA MÁQUINA É REALIZADO PELO PROGRAMA OBJETO. FICAM TAMBEM NA MEMÓRIA, NESTA FASE, AS SUBROTINAS QUE AJUDAM A PERFAZER AS OPERAÇÕES DESEJADAS. ESTAS SUBROTINAS SÃO CHAMADAS PELO PROGRAMA OBJETO.

1.2.1- PROGRAMA OBJETO:

1.2.1.1 - LINGUAGEM OBJETO:

A FINALIDADE DE UM COMPILADOR, COMO SABEMOS, É TRADUZIR UM PROGRAMA DE UMA LINGUAGEM FONTE PARA UMA LINGUAGEM OBJETO. ESTA ÚLTIMA, PRESA A SEMÂNTICA DA LINGUAGEM FONTE, É GERALMENTE LINGUAGEM DE MÁQUINA, COMO OCORRE EM NOSSO CASO.

AS PRINCIPAIS OPERAÇÕES QUE SE DESEJA REALIZAR COM UMA LINGUAGEM DE MANIPULAÇÃO DE CADEIAS SÃO: COMPARAÇÃO COM UM PADRÃO E SUBSTITUIÇÃO DE PARTE DE UMA CADEIA POR OUTRA.

COMO SABEMOS, OS COMPUTADORES CONVENCIONAIS NÃO POSSUEM INSTRUÇÕES DE MÁQUINA PRÓPRIAS PARA ESTES TIPOS DE OPERAÇÕES. HÁ PORTANTO, NA FASE EXECUÇÃO, SUBROTINAS QUE REALIZAM AS OPERAÇÕES DESEJADAS.

DESTE MODO, O PROGRAMA OBJETO SERÁ UMA SÉRIE DE CHAMADAS A ESTAS SUBROTINAS, SENDO DADOS OS PARÂMETROS CONVENIENTES.

EM RESUMO, O PROGRAMA OBJETO CONSTA DE INSTRUÇÕES DE MÁQUINA QUE SÃO BASICAMENTE CHAMADAS A SUBROTINAS E DESVIOS QUE CONTROLAM A LÓGICA DO PROGRAMA.

1.2.1.2- ALOCAÇÃO DO PROGRAMA OBJETO:

O PRIMEIRO PONTO QUE ABORDAMOS É A MONTAGEM DO PROGRAMA OBJETO: PODEMOS MONTA-LO NA MEMÓRIA OU NO DISCO. ESTA ESCOLHA É UMA DAS QUE MAIS INFLUENCIA O DESENVOLVIMENTO DESTE TRABALHO UMA VEZ QUE ATINGE ATÉ O SISTEMA DE ENTRADA E SAIDA (VER PARAGRAFO 1.2.10).

CADA UMA DAS MANEIRAS APRESENTA SUAS VANTAGENS E INCONVENIÊNCIAS.

A PRIMEIRA SOLUÇÃO, PROGRAMA MONTADO NA MEMÓRIA, APRESENTA AS SEGUINTES VANTAGENS: FACILIDADE DE PROGRAMAÇÃO E VELOCIDADE DE COMPILAÇÃO. MAS, O QUE SE GANHA EM SIMPLICIDADE, SE PERDE EM GENERALIDADE POIS OS PROGRAMAS OBJETOS GERADOS NÃO SÃO COMPATÍVEIS COM O SISTEMA OPERACIONAL E NÃO PODEM SER GUARDADOS EM MEMÓRIA AUXILIAR PARA UM USO FUTURO.

OUTRO INCONVENIENTE É O TAMANHO MÁXIMO DO PROGRAMA OBJETO QUE, NA PRIMEIRA SOLUÇÃO, É MENOR DEVIDO AO FATO DO COMPILADOR COMPARTILHAR COM ELE A MEMORIA DISPONIVEL.

APESAR DE TODOS OS INCONVENIENTES APONTADOS, ADOTAMOS A MONTAGEM DO PROGRAMA NA MEMÓRIA, PRINCIPALMENTE PELA FACILIDADE DE PROGRAMAÇÃO. ESTE PROBLEMA VOLTARÁ A SER ABORDADO NA QUARTA PARTE DESTE TRABALHO.

PARA NAO RESTRINGIRMOS MAIS AINDA O TAMANHO DO PROGRAMA OBJETO, NEM FICARMOS PRESOS AO USO DO DISCO, O QUE ABAIXARIA O RENDIMENTO DO SISTEMA, FIZEMOS O COMPILADOR DE UMA SÓ PASSAGEM DE MODO QUE O PROGRAMA FONTE NÃO FICA TODO, DE UMA SO VEZ NA MEMÓRIA.

EM RESUMO, É UM COMPILADOR DE UMA SÓ PASSAGEM QUE MONTA O PROGRAMA OBJETO NA MEMÓRIA (TIPO "LOAD & GO").

1.2.2- REPRESENTAÇÃO INTERNA DO VALOR DE UMA VARIÁVEL:

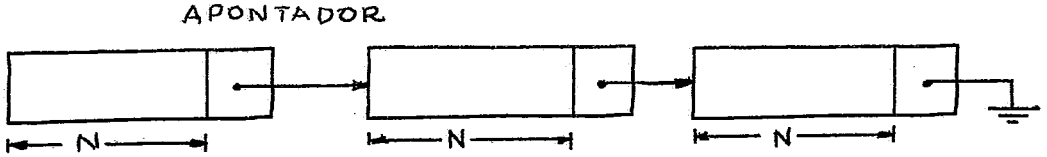
COMO SABEMOS, EM SNOBOL, AS VARIÁVEIS PODEM ASSUMIR COMO VALORES, CADEIAS DE SÍMBOLOS DE TAMANHO ARBITRÁRIO. A MANIPULACAO DE VALORES IMPLICA NUM REARRANJO DESTAS CADEIAS.

PARA ISTO NECESSITAMOS MOVIMENTAR FREQUENTEMENTE EXTENSAS CADEIAS DE UMA PARA OUTRA REGIAO DA MEMÓRIA.

COMO SABEMOS, O 1130 NÃO APRESENTA INSTRUÇÕES PRÓPRIAS PARA ESTE TIPO DE PROCESSAMENTO. MAS UMA MANEIRA DE CONTORNARMOS, PELO MENOS EM PARTE, ESTE PROBLEMA E REPRESENTARMOS UMA CADEIA EM LISTA DE APONTADORES E PARA FACILITARMOS A PROGRAMAÇÃO, FAREMOS QUE ESTA LISTA TENHA SEMPRE NÓS DO MESMO TAMANHO.

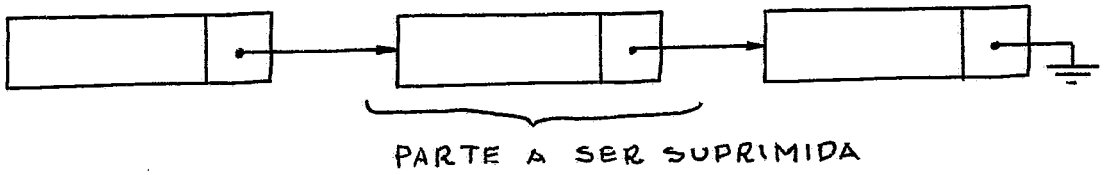
EM RESUMO, A CADEIA É COMPOSTA DE VÁRIOS TRECHOS OU NÓS DE MESMO TAMANHO, NÃO NECESSARIAMENTE CONTÍGUOS, SENDO QUE CADA NÓ CONTÉM A INFORMAÇÃO NECESSÁRIA PARA ACHARMOS O PROXIMO NÓ DA CADEIA.

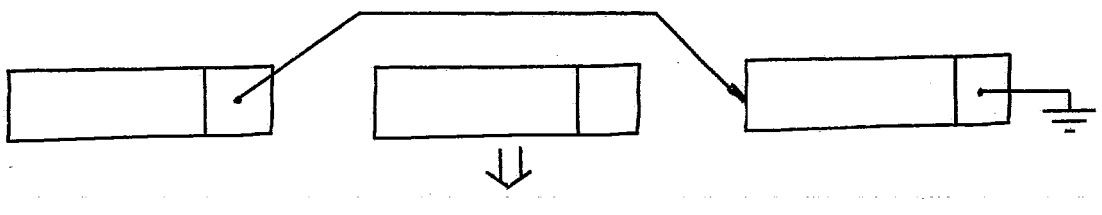
EXEMPLO



ASSIM CADA NÓ CONTÉM N CARACTERES E UM APONTADOR PARA O PROXIMO NÓ.

DESTA MANEIRA TORNA-SE FÁCIL A INSERÇÃO OU SUPRESSÃO DE UMA PARTE DA CADEIA SEM QUE HAJA A NECESSIDADE DE FAZER A RELOCAÇÃO DE TODA A CADEIA, E OS NÓS LIBERADOS SÃO DEVOLVIDOS À MEMÓRIA DISPONÍVEL (VER PARAGRAFO 1.2.4).





PARTE DO NÓ CONTÉM A INFORMAÇÃO PARA ACHARMOS O PRÓXIMO NÓ E ESTA INFORMAÇÃO NÃO É ÚTIL DO PONTO DE VISTA DA LINGUAGEM APENAS SERVINDO PARA LIGAR A LISTA DE APONTADORES. DEVE PORTANTO HAVER UM COMPROMISSO ENTRE O TAMANHO DO NÓ E A MEMÓRIA UTILIZÁVEL.

SE USARMOS UM NÓ COM MUITOS CARACTERES, A MEMÓRIA UTILIZÁVEL AUMENTA, MAS FICAMOS COM O SISTEMA UM POUCO MAIS RÍGIDO. ISTO PORQUE É MAIS FÁCIL ESVAZIARMOS NÓS MENORES (E CONSEQUENTEMENTE DEVOLVE-LOS A MEMÓRIA DISPONÍVEL) DO QUE NÓS COM MUITOS CARACTERES.

CONSIDERANDO-SE QUE A REPRESENTAÇÃO DOS CARACTERES ALFANUMÉRICOS É, PARA MÁXIMO APROVEITAMENTO, DE 2 CARACTERES POR PALAVRA E CONSIDERANDO-SE QUE O APONTADOR OCUPA UMA PALAVRA, MONTAMOS A TABELA:

NO.CARAC.POR NÓ	% UTILIZÁVEL
2	50
4	66
6	75
8	80
10	83
12	85
14	87
16	88
18	90

PELA TABELA ACIMA, DECIDIMOS USAR 6 CARACTERES POR NÓ, RESULTANDO UM NÓ DE 4 PALAVRAS.

ESTE VALOR FOI ESCOLHIDO TENDO-SE EM CONTA APENAS A PERCENTAGEM DA MEMÓRIA UTILIZAVEL. UMA ANÁLISE MAIS DETALHADA EXIGIRIA UMA SIMULAÇÃO DO SISTEMA, O QUE POR SI SÓ É BEM EXTENSA, FUGINDO A FILOSOFIA ORIGINAL DE SE TER O

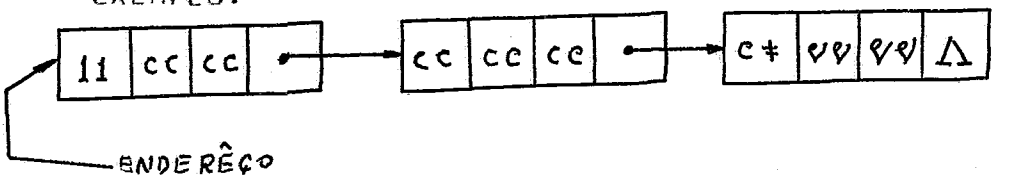
SISTEMA PRONTO A CURTO PRAZO.

PARA FACILITAR A MANIPULAÇÃO DAS CADEIAS, A PRIMEIRA PALAVRA DO PRIMEIRO NÓ DE UMA CADEIA CONTÉM O TAMANHO DA CADEIA. ISTO EVITA QUE TENHA QUE SE CONTAR O NÚMERO DE CARACTERES QUANDO NECESSÁRIO.

NA REPRESENTAÇÃO DE UMA CADEIA TEMOS OS SEGUINTES CARACTERES ESPECIAIS

ǂ- MARCA O FIM DA CADEIA.
ν- VAZIO (NÃO DEVE SER CONFUNDIDO COM BRANCO).
Λ- FIM DA LISTA. ULTIMO APONTADOR.

EXEMPLO:



ONDE C REPRESENTA CARACTERES QUAISQUER.

1.2.3- TABELA DE SÍMBOLOS- (TS):

UM PROGRAMA EM SNOBOL DEVE TER ACESSO A TABELA DE SÍMBOLOS DURANTE A EXECUÇÃO, DEVIDO AO FATO DE PODERMOS USAR ENDEREÇAMENTO INDIRETO. DESTE MODO A TABELA DE SÍMBOLOS PODE CRESCER DURANTE A EXECUÇÃO DO PROGRAMA. POR ESTE MOTIVO NÃO É INTERESSANTE TERMOS UMA TS DE TAMANHO FIXO.

POR EXEMPLO

```
A = 'J'
B = 'J'
$A =.............$B
```

NO PROGRAMA ACIMA, QUANDO É CALCULADA A VARIÁVEL $A, A VARIAVEL J É INSERIDA NA TS SE ELA AINDA NÃO FOI DEFINIDA. MAS QUANDO FOR CALCULADO $B, NÃO É MAIS NECESSÁRIO INSERIR J NA TS.

OS ELEMENTOS DA TS TAMBÉM SAO DE TAMANHO VARIÁVEL POIS NÃO HÁ LIMITE QUANTO AO NÚMERO DE CARACTERES NO NOME DE UMA VARIÁVEL.

NOVAMENTE ACHAMOS QUE A MELHOR SOLUÇÃO É FAZE-LA EM LISTA DE APONTADORES. MAS UMA VEZ QUE OS ELEMENTOS DA TS NÃO PODEM CRESCER OU DIMINUIR, FIZEMOS COM NÓS DE TAMANHO VARIÁVEL.

UMA VEZ QUE A TS SOMENTE É CONSULTADA DURANTE A EXECUÇÃO QUANDO FAZEMOS ENDEREÇAMENTO INDIRETO, É VIÁVEL A

POSSIBILIDADE DE COLOCA-LA NO DISCO. COM ISTO, TERÍAMOS UM GANHO DE MEMÓRIA EM DETRIMENTO DO TEMPO DE EXECUÇÃO E DE COMPILAÇÃO DO PROGRAMA.

HÁ UM COMPROMISSO ENTRE VELOCIDADE DO SISTEMA E GASTO DE MEMÓRIA. SE COLOCARMOS A TS NO DISCO, O RENDIMENTO DO SISTEMA ABAIXA EM FAVOR DE UM GANHO DE MEMÓRIA. UMA ANÁLISE DETALHADA DO ASSUNTO, EXIGIRIA UMA SIMULAGÃO, QUE COMO DISSEMOS ANTERIORMENTE, FOGE A FILOSOFIA ORIGINAL.

UMA VEZ QUE NOSSO OBJETIVO É A REALIZAÇAO DO SISTEMA EM CURTO ESPAÇO DE TEMPO, OPTAMOS PELA SOLUÇÃO QUE APRESENTA MAIOR FACILIDADE DE PROGRAMAÇÃO, OU SEJA A ALOCAÇÃO DA TS NA MEMÓRIA.

PARA INCREMENTARMOS A VELOCIDADE DE PROCURA NA TS, ELA É DIVIDIDA EM 16 SETORES.

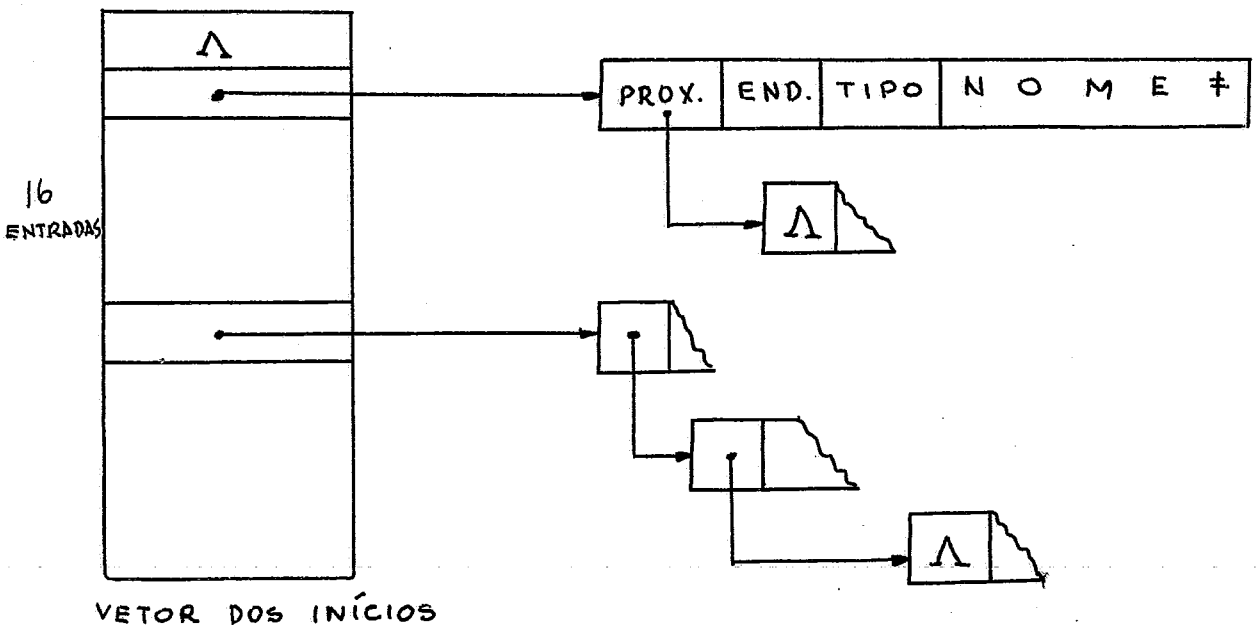
O ENDEREÇO DO INICIO DE CADA SETOR ESTÁ EM UM VETOR, QUE É IGUAL A UM CARACTER ESPECIAL (Λ) SE ESTE SETOR ESTIVER VAZIO.

QUANDO QUISERMOS PROCURAR OU INSERIR UM DETERMINADO NOME NA TS, APLICAMOS A ESTE NOME UM ALGORÍTMO QUE NOS FORNECE UM NÚMERO ALEATÓRIO ENTRE ZERO E 15. COM ESTE NÚMERO, ACHAMOS NO VETOR ACIMA MENCIONADO O ÍNICIO DO SETOR DA TS QUE NOS INTERESSA. DESTA MANEIRA, O TEMPO DE PROCURA NA TS FICA DIVIDIDO POR 16.

A TABELA DE SÍMBOLOS DEVE CONTER A SEGUINTE INFORMAÇÃO:

1) ENDEREGO DO PRÓXIMO ELEMENTO DA TS.
2) ENDEREÇO DA VARIÁVEL OU CONSTANTE OU RÓTULO ETC.
3) TIPO DO ELEMENTO (VARIÁVEL, CONSTANTE, FUNÇÃO OU NOME ESPECIAL).
4) NOME DO ELEMENTO COM CARACTER ESPECIAL MARCANDO O SEU FIM (≠).

ESQUEMATICAMENTE



EM CASO DE RÓTULOS, DEVE-SE GUARDAR AINDA A INFORMAÇÃO DE SE O RÓTULO FOI REFERENCIADO OU DEFINIDO.

NÃO E NECESSÁRIO GUARDARMOS INFORMAÇÃO DE SE UMA VARIÁVEL FOI DEFINIDA OU NÃO, POIS EM SNOBOL AS VARIÁVEIS SÃO ZERADAS NO INÍCIO DO PROGRAMA E PORTANTO AUTOMATICAMENTE DEFINIDAS.

NO CASO DO TIPO SER FUNÇÃO, NO LUGAR DO ENDEREÇO COLOCAMOS O ENDEREÇO DO DESCRITOR DE FUNÇÃO, QUE SERÁ EXPLICADO POSTERIORMENTE (VER PARAGRAFO 1.2.7).

1.2.3.1- REFERÊNCIA FUTURA:

DIZEMOS QUE TEMOS UMA REFERÊNCIA FUTURA QUANDO UM RÓTULO É REFERENCIADO ANTES DE SER DEFINIDO.

SENDO A COMPILAÇÃO DE UMA SÓ PASSAGEM, É NECESSÁRIO QUE SE GUARDE INFORMAÇÃO DE ONDE OCORREU A REFERÊNCIA FUTURA PARA QUE POSTERIORMENTE QUANDO FOR FEITA A DEFINIÇÃO, ESSA REFERÊNCIA SEJA PREENCHIDA.

TODA VEZ QUE A SUBROTINA DE PROCURA NA TS É CONSULTADA PARA UM RÓTULO É PORQUE HOUVE UMA TRANSFERÊNCIA PARA ESTE RÓTULO E SERÁ GERADA UMA INSTRUÇÃO DE DESVIO PARA ELE. SE A SUBROTINA NÃO ENCONTRA O RÓTULO NA TS, ELA ZERA O ENDEREÇO DA INSTRUÇÃO DE DESVIO E COLOCA COMO ENDEREÇO NA TS UM APONTADOR PARA A PALAVRA ZERADA NA INSTRUÇÃO DE DESVIO, LIGANDO O BIT ZERO PARA QUE POSSA SER RECONHECIDO MAIS TARDE COMO UM INDICADOR DE REFERÊNCIA FUTURA. SE NOVA REFERÊNCIA FOR FEITA, É FORMADA UMA LISTA DE APONTADORES ENCABEÇADA PELO ENDEREÇO DA TS E TERMINANDO, PELA PALAVRA ZERADA. ASSIM QUANDO FOR CONHECIDO O ENDERECO DO ROTULO, ESTA LISTA E TODA PREENCHIDA COM O VALOR DO RÓTULO, SENDO ENTÃO DESLIGADO O BIT ZERO DO ENDERECO NA TS.

POR EXEMPLO

ANTES DO APARECIMENTO DE ROT

```
                ---   --
                ---   --
                ---   --
        C       BSC   L   0         (DESVIO P/ ROT)
                ---   --
                ---   --
                ---   --
                ---   --
                ---   --
                ---   --
        B       BSC   L   C+1       (DESVIO P/ ROT)
                ---   --
                ---   --
```

```
                    ---   --
              A     BSC   L   B+1        (DESVIO P/ ROT)
```

```
TS ===>|PROX.| END=A+1 C/BIT 0 LIG.| TIPO=ROTULO | R O T  |
       ---------------------------------------------------
```

APÓS O APARECIMENTO DE ROT

```
                    ---   --
                    ---   --
                    ---   --
                    BSC   L   XXX
                    ---   --
                    ---   --
                    ---   --
                    BSC   L   XXX
                    ---   --
                    ---   --
                    ---   --
                    BSC   L   XXX
                    ---   --
                    ---   --
                    ---   --
              XXX       ( VALOR DE ROT)
```

```
TS ===>|PROX.| END= XXX  | TIPO=ROTULO | R O T  |
       ------------------------------------------
```

1.2.4- CONTROLE DA MEMÓRIA DISPONÍVEL:

A ALOCAÇÃO DA MEMÓRIA É DIFERENTE NAS FASES COMPILAÇÃO E EXECUÇÃO.

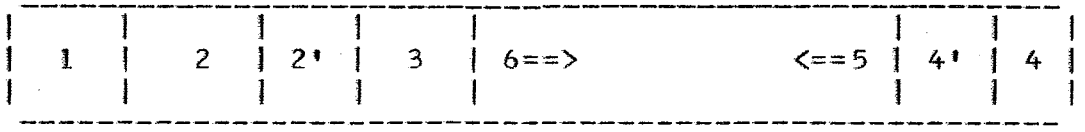
DISPOSIÇÃO DA MEMÓRIA NA FASE COMPILAÇÃO

```
 ---------------------------------------------------------
|     |           |                             |       |
|  1  |     2     | 3==>                  <==5  |   4   |
|     |           |                             |       |
 ---------------------------------------------------------
```

SENDO: 1- ÁREA DO SISTEMA MONITOR.

2- COMPILADOR.
3- PROGRAMA OBJETO CRESCENDO NO SENTIDO DA SETA.
4- ÁREA DE COMUNICACAO.
5- TS, ALOCAÇÃO DE VARIÁVEIS, CONSTANTES, ETC, CRESCENDO NO SENTIDO DA SETA.

DISPOSIÇÃO DA MEMÓRIA NA FASE EXECUÇÃO



SENDO:1 - ÁREA DO SISTEMA MONITOR.
2 - SUBROTINAS DA FASE EXECUÇÃO.
2'- VARIÁVEIS.
3 - PROGRAMA OBJETO.
4 - ÁREA DE COMUNICAÇÃO DA FASE EXECUCAO (QUE É MENOR QUE A DA FASE COMPILACAO).
4'- VARIÁVEIS.
5 - TS, ALOCAÇÃO DE NOVAS VARIÁVEIS, CONSTANTES, ETC, CRESCENDO NO SENTIDO DA SETA.
6 - PILHA DE RECURSIVIDADE CRESCENDO NO SENTIDO DA SETA.

SENDO PERMITIDA RECURSIVIDADE NAS FUNÇÕES EM SNOBOL, É NECESSÁRIO MANTER UMA PILHA PARA PROVER ESTA RECURSIVIDADE. ESTA PILHA CONTÉM OS VALORES DE VARIÁVEIS LOCAIS, ENDEREÇO DE RETORNO ETC. ESTA PILHA DE RECURSIVIDADE CRESCE NO SENTIDO CONTRÁRIO AO DAS CADEIAS PARA QUE TODA MEMORIA ÚTIL SEJA UTILIZADA (VER PARAGRAFO 1.2.7).

SÃO 3 AS SUBROTINAS QUE CONTROLAM A MEMÓRIA DISPONÍVEL. UMA DELAS FORNECE, SEMPRE QUE REQUISITADA, UM NÓ DE 4 PALAVRAS. A OUTRA DA UM NÓ DE TAMANHO VARIÁVEL, CUJO TAMANHO É FORNECIDO COMO PARÂMETRO, E A TERCEIRA DEVOLVE UM NÓ DE 4 PALAVRAS A MEMÓRIA DISPONÍVEL.

NA FASE COMPILAÇÃO, APENAS AS 2 PRIMEIRAS SÃO NECESSÁRIAS, JÁ QUE NENHUM NÓ É DEVOLVIDO. A PRIMEIRA SERVE PARA ALOCAÇÃO DE VARIÁVEIS E CONSTANTES E A SEGUNDA PARA ALOCAÇÃO DA TS.

NA FASE EXECUÇÃO, A TERCEIRA SUBROTINA TAMBÉM É NECESSÁRIA. AS 2 PRIMEIRAS FARÃO O MESMO PAPEL QUE NA FASE COMPILAÇÃO E A TERCEIRA MANTÉM UMA LISTA DE APONTADORES COM OS NÓS QUE SÃO DEVOLVIDOS. SEMPRE QUE UM NÓ É REQUISITADO, A LISTA DE APONTADORES COM OS NÓS VAZIOS (LISTA DE VAZIOS) É CONSULTADA. SE ELA ESTIVER VAZIA, ENTÃO É FORNECIDO UM NÓ DO ESPAÇO CONTÍGUO.

DESTA MANEIRA, QUANDO DISSERMOS QUE UM NÓ FOI DEVOLVIDO A MEMÓRIA DISPONÍVEL, SIGNIFICA QUE ELE FOI INSERIDO NA LISTA DE VAZIOS.

1.2.5- PILHA DE PARÂMETROS E VARIÁVEIS TEMPORÁRIAS:

COMO EM SNOBOL O VALOR DAS VARIÁVEIS NÃO TEM TAMANHO FIXO, O PROBLEMA DAS VARIÁVEIS TEMPORÁRIAS TORNA-SE CRÍTICO. ISTO PORQUE, PODEMOS TER VARIÁVEIS TEMPORÁRIAS DE QUALQUER TAMANHO O QUE TORNA INEFICIENTE O USO DA MEMÓRIA, A NÃO SER QUE ELAS SEJAM ENTREGUES A MEMÓRIA DISPONÍVEL LOGO QUE NÃO SEJAM MAIS NECESSÁRIAS. PORTANTO FICA A CARGO DAS SUBROTINAS DA FASE EXECUÇÃO A DEVOLUÇÃO DESTAS VARIÁVEIS À MEMÓRIA DISPONÍVEL. DESTA MANEIRA, ESTAS SUBROTINAS DEVEM ESTAR APTAS A RECONHECER VARIÁVEIS TEMPORÁRIAS PARA PODEREM TER O PROCEDIMENTO ADEQUADO.

EM SNOBOL, DURANTE A EXECUÇÃO DE UMA DECLARAÇÃO, HÁ A POSSIBILIDADE DE OCORRER FALHA EM UMA FUNÇÃO, OPERAÇÃO ARITMÉTICA OU PADRÃO.

NESTES CASOS, O RESTO DA DECLARAÇÃO É PULADO SENDO EXECUTADA A PRÓXIMA DECLARAÇÃO OU O CAMPO DE TRANSFERÊNCIA.

ISTO TORNA UM POUCO COMPLICADO O PROBLEMA DAS VARIÁVEIS TEMPORÁRIAS, POIS AS SUBROTINAS QUE AS USARIAM, E PORTANTO AS DEVOLVERIAM A MEMÓRIA DISPONÍVEL, PODEM SER PULADAS, DEVIDO A FALHA EM UMA PARTE ANTERIOR DA DECLARAÇÃO. TORNA-SE NECESSÁRIO ENTÃO TERMOS UMA LISTA DE VARIÁVEIS TEMPORÁRIAS EM USO, DURANTE A EXECUÇÃO DO PROGRAMA OBJETO, PARA QUE ESTAS VARIÁVEIS POSSAM SER DEVOLVIDAS A MEMÓRIA DISPONÍVEL EM CASO DE FALHA NA DECLARACAO.

ESTE PROBLEMA FOI SOLUCIONADO COM O QUE CONVENCIONAMOS CHAMAR "PILHA DE PARÂMETROS", QUE É UMA PILHA QUE FICA NA MEMÓRIA DURANTE A EXECUÇÃO DO PROGRAMA OBJETO.

TODAS AS SUBROTINAS, AO INVÉS DE TEREM OS PARÂMETROS TRANSMITIDOS APÓS A CHAMADA, BUSCAM SEUS PARÂMETROS NESTA PILHA E DEIXAM OS ENDEREÇOS DAS CADEIAS RESULTANTES NO TOPO DA MESMA. SE ALGUMA CADEIA FOR INTERMEDIÁRIA, ELA É MARCADA PARA QUE POSSA SER IDENTIFICADA PELA SUBROTINA QUE A USA.

A REALIZAÇÃO DESTA PILHA APRESENTA AS SEGUINTES VANTAGENS:

1) TODAS AS SUBROTINAS BUSCAM OS PARÂMETROS EM UM MESMO LOCAL: A PILHA DE PARAMETROS. DESTE MODO, A BUSCA DOS PARÂMETROS PODE SER FEITA POR UMA SUBROTINA PADRÃO.

2) NÃO HÁ A NECESSIDADE DE SE MANTER UMA LISTA DE VARIÁVEIS TEMPORÁRIAS, POIS TODAS AS VARIÁVEIS EM USO ESTÃO NA PILHA DE PARÂMETROS. DESTA FORMA AS VARIÁVEIS TEMPORÁRIAS SÃO FORMADAS SOMENTE QUANDO NECESSÁRIO E O SÃO PELAS SUBROTINAS

DA FASE EXECUÇÃO.
3) A PILHA DE PARÂMETROS FACILITA A SUBROTINA QUE REALIZA A ANALISE SINTÁTICA DA DECLARAÇÃO, NÃO SENDO NECESSÁRIO MANTER-SE UMA PILHA DE OPERANDOS NA FASE DE COMPILAÇÃO.

MAIS ADIANTE QUANDO ABORDARMOS O ASSUNTO RECURSIVIDADE, VOLTAREMOS A PILHA DE PARÂMETROS, SOB O ASPECTO DE LOCALIZAÇÃO E TAMANHO.

1.2.6- PADRÃO PARA CASAMENTO:

QUANDO É IDENTIFICADO NA DECLARAÇÃO SNOBOL UM PADRÃO PARA CASAMENTO, O COMPILADOR DEVE GUARDAR A INFORMAÇÃO NECESSÁRIA PARA QUE A SUBROTINA DA FASE EXECUÇÃO POSSA FAZER A COMPARAÇÃO NECESSÁRIA.

ESTA ÁREA ONDE SERÁ MONTADA A INFORMAÇÃO CHAMAREMOS DE DESCRITOR DE PADRÃO OU SIMPLESMENTE DP.

COMO NÃO SABEMOS A PRIORI O NÚMERO DE ELEMENTOS DO PADRÃO, TORNA-SE NECESSÁRIO UMA AREA INTERMEDIÁRIA PARA MONTAGEM DO DP, OU ENTÃO MONTAMOS O DP EM UMA LISTA DE APONTADORES.

OPTAMOS PELA SEGUNDA SOLUÇÃO, POIS A OUTRA IMPORIA RESTRICÕES QUANTO AO TAMANHO DO PADRÃO.

CADA ELEMENTO DO PADRÃO OCUPA ENTÃO UM NÓ DA LISTA QUE DEVE CONTER AS SEGUINTES INFORMAÇÕES: TIPO DO ELEMENTO (VARIÁVEL ARBITRÁRIA, BALANCEADA, CONSTANTE, ETC...), TAMANHO DO ELEMENTO NO CASO DE VARIÁVEL DE TAMANHO FIXO E ENDEREÇO DO ELEMENTO.

UMA VEZ MONTADO O DP, QUANDO FOR CHAMADA A SUBROTINA PARA CASAMENTO DO PADRÃO, O SEU ENDEREÇO É DADO COMO PARÂMETRO.

COMO É POSSÍVEL FALHA EM UM CASAMENTO DE PADRÃO, LOGO APÓS A CHAMADA DA SUBROTINA É GERADA UMA INSTRUÇÃO DE DESVIO PARA PARTE DA DECLARAÇÃO QUE PROCESSA A FALHA DA DECLARAÇÃO. O RETORNO DA SUBROTINA DE CASAMENTO É FEITO PARA UMA INSTRUÇÃO ADIANTE DA CHAMADA EM CASO DE SUCESSO, PULANDO DESTA FORMA A TRANSFERÊNCIA PARA A FALHA.

RESTA APENAS RESSALTAR O FATO DE NÃO TERMOS NA FASE EXECUÇÃO INFORMAÇÃO SUFICIENTE PARA DETETAR CERTOS ELEMENTOS DO TIPO REFERÊNCIA PASSADA, FICANDO ESTA PARTE PARA A FASE EXECUÇÃO.

EXEMPLO

```
X = 'AB'
Y  *AB*  VAR  $X
```

O TIPO DO TERCEIRO ELEMENTO DO PADRÃO ACIMA, $X, NÃO PODE SER DETETADO NA FASE COMPILAÇÃO, POIS O

ENDEREÇAMENTO INDIRETO DEVE SER REALIZADO NA HORA DA EXECUÇÃO DA DECLARAÇÃO.

1.2.7- FUNÇÕES E RECURSIVIDADE:

DA MESMA MANEIRA QUE MONTAMOS O DESCRITOR DE PADRÃO, PARA FUNÇÕES É NECESSÁRIO TAMBÉM GUARDARMOS AS INFORMAÇÕES NECESSÁRIAS PARA A EXECUÇÃO DE UMA FUNÇÃO. GUARDAMOS ESTAS INFORMAÇÕES EM UMA ÁREA À QUAL CHAMAREMOS DESCRITOR DE FUNÇÃO OU SIMPLESMENTE DF.

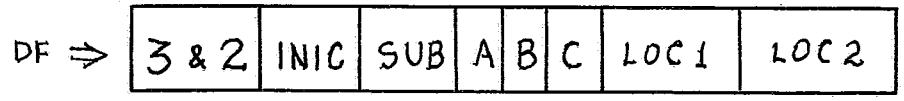
O DF É MONTADO QUANDO A FUNÇÃO FOR DEFINIDA ATRAVÉS DE UMA DECLARAÇÃO DEFINE E CONTÉM AS SEGUINTES INFORMAÇÕES:

1- NO. DE PARÂMETROS E VARIÁVEIS LOCAIS.
2- ENDERECO DE ENTRADA DA FUNÇÃO.
3- ENDERECO DO VALOR DA FUNÇÃO.
4- ENDERECO DOS PARÂMETROS FORMAIS.
5- ENDERECO DAS VARIÁVEIS LOCAIS.

O FORMATO DO DF SERA O SEGUINTE:

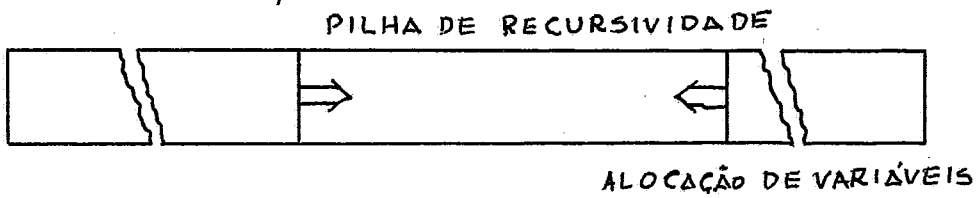
POR EXEMPLO:

```
DEFINE('SUB(A,B,C)','INIC','LOC1,LOC2')
```



SENDO PERMITIDA RECURSIVIDADE DE FUNÇÃO EM SNOBOL, TORNA-SE NECESSÁRIO MANTER-SE UMA PILHA PARA GUARDAR OS VALORES GLOBAIS DOS PARÂMETROS, DAS VARIAVEIS LOCAIS E DOS ENDEREÇOS DE RETORNO.

PARA TOTAL APROVEITAMENTO DA MEMÓRIA, ESTA PILHA CRESCE EM SENTIDO CONTRÁRIO AO DA ALOCAÇÃO DE NOVOS VALORES PARA VARIÁVEIS. ASSIM A PILHA DE RECURSIVIDADE CRESCE NO SENTIDO DE ENDEREÇOS CRESCENTES E A ALOCAÇÃO CRESCE NO SENTIDO DE ENDEREÇOS DECRESCENTES.



DESTA MANEIRA A MEMÓRIA SOMENTE É ESGOTADA QUANDO A PILHA DE RECURSIVIDADE ATINGIR A ÁREA DAS VARIÁVEIS OU VICE-VERSA.

QUANDO FOR IDENTIFICADA UMA CHAMADA PARA UMA FUNÇÃO, É CHAMADA UMA SUBROTINA QUE SALVA NA PILHA DE RECURSIVIDADE OS VALORES NECESSÁRIOS. ESTA FUNÇAO TERA COMO PARÂMETRO APENAS O NÚMERO DE PARÂMETROS DA CHAMADA E O ENDEREÇO DO DF. OS PARÂMETROS PROPRIAMENTE DITOS ESTÃO NA PILHA DE PARÂMETROS.

PARA O RETÔRNO DA FUNÇÃO, É CHAMADA OUTRA SUBROTINA QUE PROCURA NO TOPO DA PILHA DE RECURSIVIDADE QUAL A ÚLTIMA FUNÇÃO QUE FOI CHAMADA, RESTAURANDO OS VALORES DAS VARIÁVEIS LOCAIS E RETORNANDO PARA O ENDEREÇO INDICADO NA PILHA.

DA MESMA FORMA QUE O PADRÃO DE CASAMENTO, AS FUNÇÕES PODEM FALHAR E NESTE CASO, TODO O RESTO DA DECLARAÇÃO DEVE SER PULADO.

ISTO É FEITO, COMO NO PADRÃO DE CASAMENTO, RETORNANDO PARA A INSTRUCAO SEGUINTE EM CASO DE FALHA E PARA UMA INSTRUCAO ADIANTE EM CASO DE SUCESSO. PARA ISTO, LOGO APÓS A CHAMADA DA SUBROTINA QUE PROVE A RECURSIVIDADE É GERADA UMA INSTRUÇÃO DE DESVIO PARA A PARTE DA DECLARAÇÃO QUE PROCESSA FALHA. MAIS TARDE ABORDAREMOS EM QUE CONSISTE A PARTE DA DECLARAÇÃO QUE PROCESSA A FALHA.

QUANDO UMA FUNÇÃO É CHAMADA, A PILHA DE PARÁMETROS TAMBÉM DEVE SER SALVA, ISTO É, DEVEMOS SALVAR OS VALORES DO TOPO E DA BASE DA PILHA.

SE FIZERMOS A PILHA DE PARÂMETROS DE TAMANHO FIXO, PODE OCORRER O CASO DE UMA FUNÇÃO SER CHAMADA RECURSIVAMENTE UM NÚMERO MUITO GRANDE DE VEZES E ENCHER A PILHA DE PARÂMETROS ANTES DE ESGOTAR A MEMÓRIA. ISTO PORQUE, OS PARÂMETROS QUE JÁ FORAM CALCULADOS NO MEIO DE UMA DECLARAÇÃO ATÉ A CHAMADA DA FUNÇÃO, SÃO GUARDADOS.

POR EXEMPLO

```
DEFINE ('SUB(A)', 'INIC')
INIC ---   ---   --
     ---   ---   --
     ---   ---   --
     ---   ---   --
         J = A...'BC'   SUB(A)
```

ESTES VALORES JÁ ESTARÃO NA PILHA DE PARÂMETROS QUANDO SUB(A) FOR CHAMADA.

SE SUB(A) FOR CHAMADA UM NÚMERO GRANDE DE VEZES E SE A PILHA DE PARÂMETROS TIVER TAMANHO FIXO, ELA PODE ESGOTAR ANTES DA PILHA DE RECURSIVIDADE.

POR ESTE MOTIVO ACHAMOS QUE O MELHOR LOCAL PARA A PILHA DE PARÂMETROS É APÓS O ÚLTIMO ELEMENTO DA PILHA DE RECURSIVIDADE.

A SUBROTINA QUE MONTA A PILHA DE RECURSIVIDADE (A QUAL CHAMAREMOS FUNC) TEM AS SEGUINTES FUNÇÕES:

1- SALVAR EM VARIÁVEIS TEMPORÁRIAS OS VALORES DA FUNÇÃO, OS VALORES GLOBAIS DAS VARIÁVEIS LOCAIS E OS VALORES DOS PARÂMETROS FORMAIS.

2- COPIAR OS PARÂMETROS REAIS EM CIMA DOS PARÂMETROS QUE FORAM SALVADOS.

3- ZERAR O VALOR DA FUNÇÃO E DE QUALQUER PARÂMETRO FORMAL QUE NÃO TENHA CORRESPONDENTE REAL, TAMBÉM COMO AS VARIÁVEIS LOCAIS.

NOTA- CONVÉM LEMBRARMOS QUE EM SNOBOL, UMA FUNÇÃO PODE SER DEFINIDA COM MAIS PARÂMETROS DO QUE É CHAMADA, SENDO ZERADOS OS PARÂMETROS SEM CORRESPONDÊNCIA.

4- MONTAR A PILHA DE RECURSIVIDADE COM AS SEGUINTES INFORMAÇÕES:

A- ENDEREÇO DE RETÔRNO.
B- ENDEREÇO DO DF.
C- ENDEREÇO DA VARIÁVEL TEMPORÁRIA QUE CONTÉM O VALOR DA FUNÇÃO.
D- IDEM DOS PARÂMETROS FORMAIS.
E- IDEM DAS VARIÁVEIS LOCAIS.
F- TOPO E BASE DA PILHA DE PARÂMETROS.
G- MODIFICAR O ENDEREÇO DO TOPO E DA BASE DA PILHA DE PARÂMETROS.
H- TRANSFERIR PARA ENTRADA DA FUNÇÃO.

POR EXEMPLO:

UMA FUNÇÃO DEFINIDA

```
DEFINE ('SUB(A,B,C,)','INICIO','LOC1,LOC2')
```

E FOI CHAMADA- SUB (D,E)

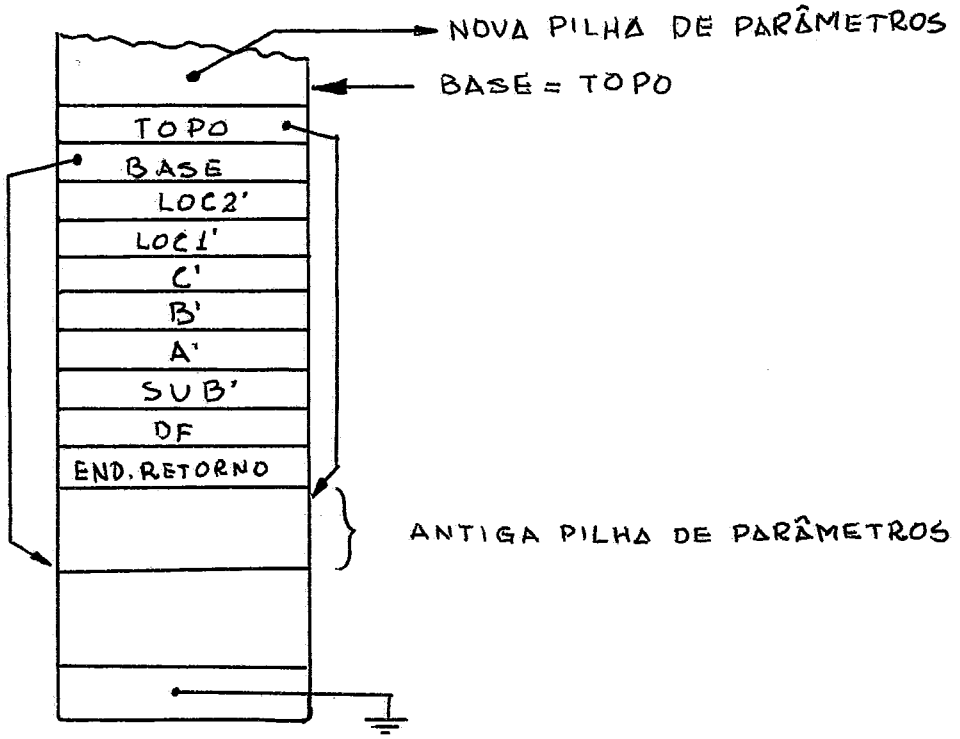
A SUBROTINA FUNC FAZ:

1)COPIA EM VARIÁVEIS TEMPORÁRIAS OS VALORES DE SUB, A, B, C, LOC1 E LOC2, QUE ESTARÃO EM SUB', A', B', C', LOC1' E LOC2' RESPECTIVAMENTE.

2)FAZ A = D, B = E E C = VAZIO.

3)SUB = VAZIO, LOC1 = VAZIO E LOC2 = VAZIO.

4)MONTA A PILHA DE RECURSIVIDADE COM O SEGUINTE FORMATO:



5)TRANSFERE CONTRÔLE DO PROGRAMA PARA INICIO.

O RETÔRNO DA FUNÇÃO TAMBÉM É FEITO ATRAVÉS DE UMA SUBROTINA QUE DESFAZ A PILHA DE RECURSIVIDADE.

ESTA SUBROTINA TEM AS SEGUINTES TAREFAS:

1- MODIFICAR O ENDEREÇO DO TOPO E BASE DA PILHA DE PARÂMETROS PARA OS VALORES RECALCADOS (VER PARAGRAFO 1.2.9).

2- RESTAURAR OS ANTIGOS VALORES DAS VARIÁVEIS LOCAIS E PARÂMETROS FORMAIS.

3- EM CASO DE FALHA (DA FUNÇÃO), RESTAURAR O VALOR ANTIGO DA FUNÇÃO.

4- RETORNAR PARA INSTRUCAO SEGUINTE À CHAMADA EM CASO DE FALHA, OU PARA UMA INSTRUÇÃO ADIANTE EM CASO DE SUCESSO.

POR EXEMPLO

NO CASO CITADO ANTERIORMENTE, FAZ O SEGUINTE:

1) A = A', B = B', C = C', LOC1 = LOC1', LOC2 = LOC2'.

2) MODIFICA OS VALORES DA BASE E TOPO DA PILHA DE RECURSIVIDADE.

3) EM CASO DE FALHA FAZ SUB = SUB'.

4) EM CASO DE FALHA DEVOLVE PARA INSTRUÇÃO SEGUINTE À CHAMADA OU UMA INSTRUÇÃO A FRENTE EM CASO DE SUCESSO.

CONVÉM LEMBRAR QUE É REALIZADO UM CONTROLE PARA QUE NÃO HAJA MAIS RETORNOS QUE CHAMADAS PARA UMA FUNÇÃO.

1.2.8- FALHA DE UMA DECLARAÇÃO:

COMO FOI DITO ANTERIORMENTE, EM SNOBOL, UMA DECLARAÇÃO PODE FALHAR. NESTE CASO, O RESTANTE DA DECLARAÇÃO É PULADO, SENDO O CONTRÔLE TRANSFERIDO PARA UMA PARTE QUE PROCESSA A FALHA.

SOMENTE HÁ POSSIBILIDADE DE FALHA QUANDO HOUVER UM PADRÃO DE CASAMENTO, UMA OPERAÇÃO ARITMÉTICA OU UMA CHAMADA DE UMA FUNCÃO. EM TODOS OS CASOS, LOGO APÓS A CHAMADA DA SUBROTINA QUE REALIZA A TAREFA DESEJADA, É GERADA UMA INSTRUCAO DE DESVIO PARA A FALHA DA DECLARAÇÃO. SE A TAREFA TIVER SUCESSO, O CONTRÔLE E RETORNADO PARA UMA INSTRUÇÃO ADIANTE E SE FALHAR O CONTRÔLE E RETORNADO PARA A INSTRUÇÃO QUE DESVIA PARA A FALHA.

A PARTE QUE PROCESSA FALHA TEM AS SEGUINTES FUNÇÕES:

1- ESVAZIAR A PILHA DE PARÂMETROS, DEVOLVENDO A MEMÓRIA DISPONÍVEL TODAS AS VARIAVEIS TEMPORÁRIAS.

2- CALCULAR UM CAMPO DE TRANSFERÊNCIA INCONDICIONAL OU DE TRANSFERÊNCIA COM FALHA.

3- TRANSFERIR O CONTRÔLE DO PROGRAMA PARA ESTE CAMPO.

1.2.9- ÁREA DE COMUNICAÇÃO:

ÁREA DE COMUNICAÇÃO É UMA ÁREA RESERVADA, NO FIM DA MEMÓRIA, ONDE FICAM AS CONTANTES, VARIÁVEIS E INDICADORES QUE SÃO REFERENCIADOS POR VÁRIAS SUBROTINAS, TANTO NA FASE COMPILAÇÃO COMO NA FASE DE EXECUÇÃO.

CONTÉM INDICADORES ESPECIFICOS COMO BASE E TOPO DA PILHA DE PARÂMETROS, CONTADOR DE INSTRUÇÕES, TOPO DA PILHA DE OPERADORES, ETC...

A DESCRIÇÃO DETALHADA DOS INDICADORES E SEUS SIGNIFICADOS SERÁ REALIZADA EM UMA PUBLICAÇÃO À PARTE QUE CONTERÁ A LÓGICA DO SISTEMA. APENAS DEIXAMOS AQUI QUE ESTA ÁREA EXISTE E ESTÁ LOCALIZADA NO FINAL DA MEMÓRIA (VER PARÁGRAFO 1.2.4).

1.2.10- ENTRADA E SAIDA:

UMA VEZ QUE O PROGRAMA OBJETO JÁ FICA NA MEMÓRIA, AO INICIAR A EXECUÇÃO DO MESMO, TODAS AS SUBROTINAS QUE SÃO USADAS JÁ DEVEM ESTAR NA MEMÓRIA.

UMA OUTRA SOLUÇÃO QUE NOS OCORRE, É DE CARREGARMOS NA HORA DA EXECUÇÃO, APENAS AS SUBROTINAS QUE SÃO USADAS. MAS ISTO IMPLICARIA TERMOS DE RELOCAR O PROGRAMA OBJETO OU RECOLOCAR AS SUBROTINAS DE ENTRADA E SAIDA, O QUE EXIGIRIA MAIS UMA FASE NA COMPILAÇÃO, ALÉM DE PERDERMOS EM SIMPLICIDADE E TEMPO DE EXECUÇÃO.

SE O PROGRAMA FOSSE ARMAZENADO NO DISCO, O PRÓPRIO SISTEMA SE ENCARREGARIA DE SÓ CARREGAR AS SUBROTINAS QUE FOSSEM CHAMADAS. NA QUARTA PARTE DESTE TRABALHO, TECEREMOS ALGUMAS CONSIDERAÇÕES SOBRE ESTE ASPECTO.

SENDO O COMPUTADOR 1130 UM COMPUTADOR DE POUCOS RECURSOS QUANTO A PERIFÉRICOS (GERALMENTE APRESENTA APENAS IMPRESSORA, LEITORA E PERFURADORA DE CARTOES E DISCO), NÃO ESTARIAMOS RESTRINGINDO SEU POTENCIAL SE LIMITASSEMOS A ENTRADA E SAIDA APENAS A ALGUNS PERIFERICOS.

PELA RAZÕES ACIMA, EXISTEM APENAS 3 APARELHOS DE ENTRADA E SAIDA, A IMPRESSORA, A LEITORA E A PERFURADORA DE CARTÕES, QUE CORRESPONDEM AS VARIÁVEIS SYSPOT, SYSPIT E SYSPPT. DESTA MANEIRA AS FUNÇÕES READ, PRINT E OUTRAS FUNÇÕES SEMELHANTES NÃO PODEM SER IMPANTADAS IMEDIATAMENTE.

PARTE II

FASE COMPILAÇÃO

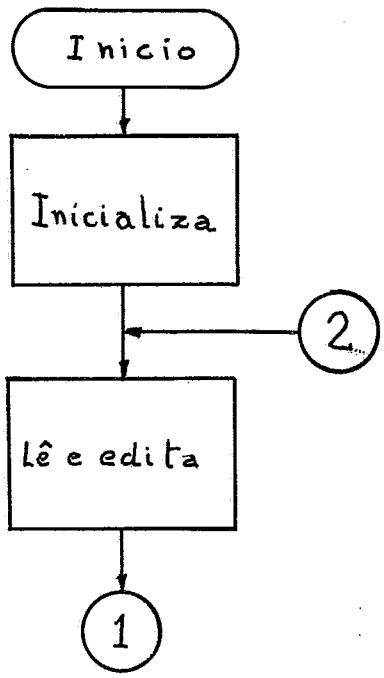
2.1- ESTRUTURA GERAL:

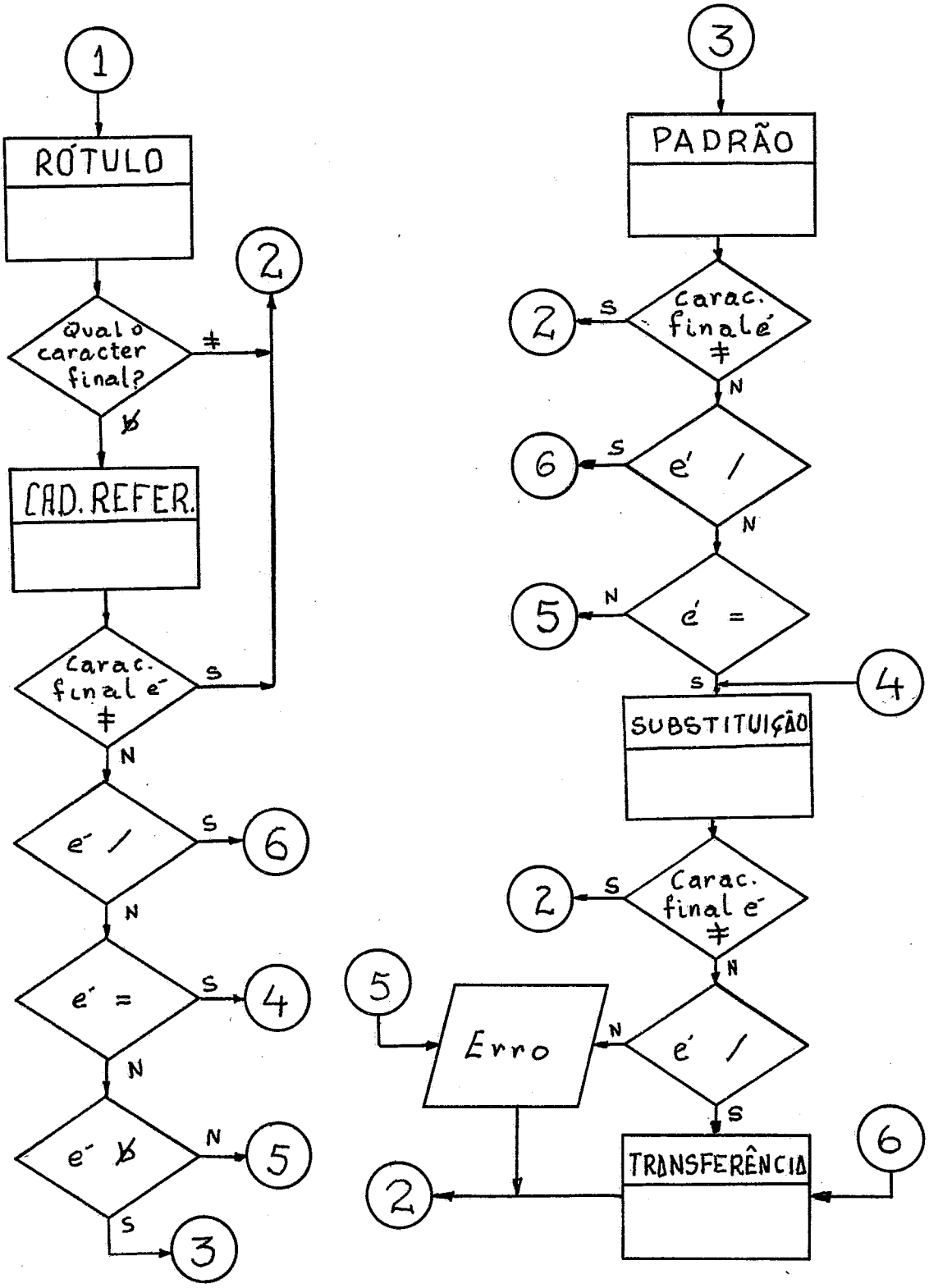
A LINGUAGEM SNOBOL É COMPOSTA DE APENAS UM TIPO DE DECLARAÇÃO. ESTA DECLARAÇÃO É DIVIDIDA EM CINCO CAMPOS DISTINTOS; RÓTULO, CADEIA DE REFERÊNCIA, PADRÃO, SUBSTITUIÇÃO E TRANSFERÊNCIA. A SINTAXE DA LINGUAGEM PERMITE QUE ESTES CAMPOS SEJAM OPCIONAIS, EMBORA ALGUMAS VEZES SUA EXISTENCIA IMPLIQUE NA EXISTENCIA DE UM CAMPO ANTERIOR. DEVIDO A SINTAXE DA LINGUAGEM ESTES CAMPOS SÃO FACILMENTE RECONHECIDOS. ESTE FOI O MOTIVO QUE NOS LEVOU A SEPARAR A ROTINA DE ANÁLISE EM CINCO PARTES PRINCIPAIS, UMA PARA CADA CAMPO.

POR EXEMPLO, A PARTE QUE AVALIA A CADEIA DE REFERÊNCIA TERÁ TRES SAIDAS (DEPENDENDO DO CAMPO QUE ENCONTRAR A SEGUIR, POIS SÃO OPCIONAIS) UMA PARA A PARTE QUE AVALIA SUBSTITUIÇÃO, A OUTRA PARA TRANSFERÊNCIA E A ÚLTIMA PARA A PRÓXIMA DECLARAÇÃO.

PARA FACILITAR A ANÁLISE DA DECLARAÇÃO, FAZEMOS UMA EDIÇÃO INICIAL ONDE SÃO RETIRADOS OS BRANCOS SUPÉRFLUOS E VERIFICADOS O BALANCEAMENTO DE ASPAS E PARÊNTESES. A VERIFICAÇÃO DE ASPAS E PARÊNTESES TORNA-SE NECESSÁRIA PARA IMPEDIR QUE SE DIVIDA UMA DECLARAÇÃO NO MEIO DE UM TERMO (MAIS ADIANTE, NO PARAGRAFO 2.2.1, DAREMOS UMA DEFINIÇÃO RIGOROSA DE TERMO).

DESTE MODO TEREMOS COMO ESTRUTURA SIMPLIFICADA:





NOTA - O CARACTER '≠' MARCA O FINAL DO CARTÃO.

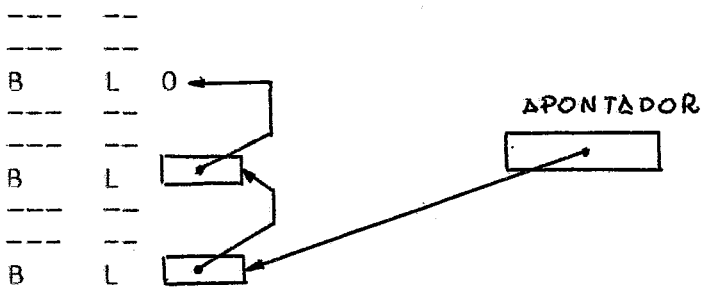
MUITAS VEZES AS ROTINAS DE ANÁLISE NÃO TERMINAM A COMPILAÇÃO DA PARTE ESPECIFICADA AO ENCONTRAR O FINAL DO REGISTRO, POIS PODE HAVER INDICAÇÃO DE CONTINUAÇÃO. ELAS APENAS LIGAM UM INDICADOR PARA QUE O CONTRÔLE LHES SEJA DEVOLVIDO CASO SEJA DETETADO UM CARTÃO DE CONTINUAÇÃO. DESTE MODO, HÁ UMA ROTINA QUE TERMINA A DECLARACAO ANTERIOR SE O REGISTRO ATUAL NÃO FOR CONTINUAÇÃO. ESTA ROTINA SABE ONDE A COMPILAÇÃO FOI SUSPENSA PELO INDICADOR QUE FOI LIGADO.

ALEM DESTAS ROTINAS PRINCIPAIS, HÁ AINDA UMA OUTRA QUE FAZ A IDENTIFICAÇÃO DOS CARTÕES DE CONTRÔLE E CONTROLA A LISTAGEM.

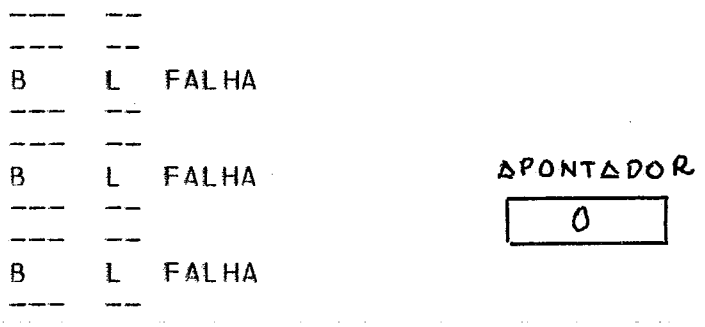
JÁ NOS REFERIMOS ANTERIORMENTE AO FATO DE APÓS A CHAMADA DE UMA SUBROTINA EM QUE POSSA HAVER FALHA, SER GERADA UMA TRANSFERÊNCIA INCONDICIONAL PARA A PARTE DA DECLARAÇÃO QUE PROCESSA O CAMPO DE FALHA. MAS NESTE PONTO, AINDA NÃO SABEMOS O ENDEREÇO DE TAL PARTE, NÃO PODENDO ASSIM GERAR A INSTRUÇÃO COMPLETA. A SOLUÇÃO DADA FOI SEMELHANTE A EXPLICADA PARA REFERÊNCIA FUTURA. A SEGUNDA PALAVRA DA INSTRUÇÃO DE DESVIO É ZERADA E UM APONTADOR ESPECIAL APONTA PARA ELA. AO SER GERADA OUTRA TRANSFERÊNCIA PARA FALHA, MONTAMOS UMA LISTA DE APONTADORES, ENCABECADA POR UM APONTADOR E TERMINANDO NA PALAVRA ZERADA. AO SER COMPILADA A PARTE DE FALHA, ESTA LISTA E DESFEITA E TODOS OS DESVIOS SÃO APONTADOS PARA A PARTE DE FALHA.

EXEMPLO:

ANTES DO APARECIMENTO DA PARTE DE FALHA:



APÓS O APARECIMENTO DA PARTE DE FALHA:



AO SER DESFEITA A LISTA, O APONTADOR É ZERADO PARA INDICAR QUE NÃO HÁ DESVIOS A SEREM FEITOS.

RESTA-NOS AGORA DAR UMA BREVE DESCRIÇÃO DO QUE CONSTA A PARTE DE FALHA.

ELA SURGE QUANDO TEMOS DE CALCULAR UM CAMPO DE TRANSFÊRENCIA PARA FALHA (OU INCONDICIONAL) OU SIMPLESMENTE NO FINAL DE UMA DECLARAÇÃO. CONSTA DE UMA CHAMADA PARA A SUBROTINA QUE ESVAZIA A PILHA DE PARÂMETROS E, SE NECESSÁRIO, DO CÁLCULO E TRANSFERÊNCIA PARA RÓTULO ESPECIFICADO (VER PARAGRAFO 2.4.8).

ALÉM DO CORPO PRINCIPAL DO COMPILADOR, EXISTE AINDA SUBROTINAS AUXILIARES. AS MAIS IMPORTANTES SERÃO DESENVOLVIDAS EM DETALHES ADIANTE.

2.2- COMPILAÇÃO DE EXPRESSÃO:

2.2.1- GENERALIDADES:

DE TODAS AS SUBROTINAS AUXILIARES, A MAIS IMPORTANTE E A QUE AVALIA UMA EXPRESSÃO SNOBOL. ISTO PORQUE SEMPRE ONDE É PERMITIDA UMA VARIÁVEL, É TAMBÉM PERMITIDO ENDEREÇAMENTO INDIRETO EM UM GRUPAMENTO O QUE NÃO DEIXA DE SER UMA EXPRESSÃO.

DESTE MODO PODEMOS TER EXPRESSÕES EM TODOS OS CAMPOS (EXCETO RÓTULO) DE UMA DECLARAÇÃO SNOBOL.

EXEMPLO DE DECLARAÇÃO QUE CONTÉM EXPRESSÃO EM TODOS OS CAMPOS:

```
ROT $(A B + '2') *$(A '7')/(J * '2')* = A B + '3'
.       /($(ROT '.' J + '3'))
```

ALÉM DISTO, OS PARÂMETROS DE CHAMADA DE UMA FUNÇÃO TAMBÉM PODEM SER EXPRESSÕES.

É CONVENIENTE PORTANTO QUE FAÇAMOS UMA SUBROTINA GENÉRICA QUE COMPILE UMA EXPRESSÃO EM QUALQUER CAMPO. ESTA SUBROTINA DEVE SABER DE ONDE FOI CHAMADA, PARA TERMINAR A COMPILAÇÃO DA EXPRESSÃO PELO CARACTER CERTO. POR EXEMPLO SE CHAMADA NA PARTE DE SUBSTITUIÇÃO, SÓ TERMINA A COMPILAÇÃO DA EXPRESSÃO NO FINAL DO REGISTRO OU QUANDO DETETADO UM CAMPO

DE TRANSFERÊNCIA. POR OUTRO LADO SE CHAMADA NA COMPILAÇÃO DA CADEIA DE REFERÊNCIA DEVE COMPILAR A MENOR EXPRESSÃO POSSIVEL.

UMA EXPRESSAO EM SNOBOL, PODE SER DEFINIDA COM A AJUDA DA BNF, DA SEGUINTE MANEIRA:

<VAZIO> ::=

 ::= BRANCO
<LETRA> ::= A|B|...|Z
<DÍGITO> ::= 1|2|...|9
<CARACTER ESPECIAL> ::= (|)|,|.|=|*|-|/|$|
<CARACTER> ::= <LETRA>|<DÍGITO>|<CARAC.ESP.>
<U> ::=
|<U>
<CARACTER DE NOME> ::= <LETRA>|<DÍGITO>|.
<NOME EXPLÍCITO> ::= <CARAC.NOME>|<NOME EXPL.><CARAC.NOME>
<NOME IMPLÍCITO> ::= $<NOME EXPL.>|$<FUNCAO>|$<GRUPAMENTO>
<NOME> ::= <NOME EXP.>|<NOME IMPL.>
<CADEIA> ::= <VAZIO>|<CARACTER>|<CADEIA><CARACTER>
<CONSTANTE> ::= '<CADEIA>'
<OPERADOR> ::= +|-|*|/|**
<OPERADOR ARITMÉTICO> ::= <U><OPERADOR><U>
<FUNÇÃO> ::= <NOME EXPL.>(<LISTA DE PARÂMETROS>)
<LISTA NÃO VAZIA> ::= <EXPRESSÃO>|<LISTA NÃO VAZIA>,<EXPR.>
<LISTA DE PARÂMETROS> ::= <VAZIO>|<LISTA NÃO VAZIA>
<OPERANDO> :: = <NOME>|<CONSTANTE>|<GRUPAMENTO>|<FUNÇÃO>
<EXPRESSÃO ARITMETICA> :: = <OPERANDO><OP.ARITM.><OPERANDO>
<GRUPAMENTO> ::= (<EXPRESSÃO>)
<TERMO> ::= <EXPRESSÃO ARITMÉTICA>|<OPERANDO>
<EXPRESSÃO> ::= <TERMO>|<EXPRESSAO><U><TERMO>

DESTA DEFINIÇÃO, VEMOS QUE É CONVENIENTE QUE FAÇAMOS COM QUE A SUBROTINA QUE COMPILA UMA EXPRESSÃO SEJA RECURSIVA COM A QUE COMPILA FUNÇÃO. ISTO PORQUE UM TERMO DE UMA EXPRESSÃO PODE SER UMA FUNÇÃO E ESTA PODE TER EXPRESSÕES NA LISTA DE PARÂMETROS. MAIS TARDE, SERA DETALHADA ESTA RECURSIVIDADE E QUAIS OS PARÂMETROS LOCAIS DESEJÁVEIS.

2.2.2- CÓDIGO OBJETO GERADO NA COMPILAÇÃO DE UMA EXPRESSÃO:

UM DOS PRIMEIROS PASSOS NA REALIZAÇÃO DE UM COMPILADOR É SEM DUVIDA A ELABORAÇÃO DO CÓDIGO OBJETO A SER GERADO. PARA ISTO PRECISAMOS SABER DE ANTE-MÃO QUAIS SÃO E O QUE FAZEM AS SUBROTINAS DE EXECUÇÃO. A SEGUIR VAI UMA BREVE DESCRIÇÃO DAS SUBROTINAS QUE IMPORTAM NA AVALIAÇÃO DE UMA EXPRESSÃO E SUAS FUNÇÕES.

1) STACK(END) - COLOCA O ENDEREÇO END NO TOPO DA PILHA DE PARÂMETROS.

2) CONC(N) - CONCATENA AS N PRIMEIRAS CADEIAS DA PILHA DE PARÂMETROS E COLOCA O RESULTADO DA CONCATENAÇÃO NO TOPO DA MESMA.

3) SOMA, SUBTR, MULT, DIVIZ E EXPO - REALIZAM SOMA, SUBTRAÇÃO, MULTIPLICAÇÃO, DIVISÃO E EXPONENCIAÇÃO RESPECTIVAMENTE, COM AS DUAS ÚLTIMAS CADEIAS DA PILHA, DEIXANDO O RESULTADO NO TOPO DA MESMA.

4) IND - FAZ ENDEREÇAMENTO INDIRETO NA ÚLTIMA CADEIA DA PILHA DEIXANDO O ENDEREÇO DA VARIÁVEL RESULTANTE NO TOPO DA MESMA.

5) FUNC (DF,N) - FAZ A RECURSIVIDADE DE FUNÇÕES. SEUS PARAMETROS SÃO:

DF- DESCRITOR DE FUNÇÃO.

N - NUMERO DE PARÂMETROS REAIS. OS N PARAMETROS REAIS SÃO ENCONTRADOS NO TOPO DA PILHA EM ORDEM INVERSA.

NOTA - QUANDO UMA FUNÇÃO RETORNA CONTROLE AO PROGRAMA PRINCIPAL, O VALOR DA MESMA JÁ ESTÁ NO TOPO DA PILHA DE PARÂMETROS.

A MELHOR MANEIRA DE SE EXPLICAR O CODIGO OBJETO GERADO E DAR UM EXEMPLO BEM GENÉRICO. É O QUE FAREMOS A SEGUIR, CABENDO AINDA EXPLICAR AS SEGUINTES CONVENÇÕES USADAS:

1) O CÓDIGO OBJETO É REPRESENTADO EM TRÊS CAMPOS: RÓTULO, OPERAÇÃO E OPERANDOS.

2) PARA FACILITAR, O NOME DA SUBROTINA FICA NO CAMPO DA OPERAÇÃO E OS PARÂMETROS NO DO OPERANDO, SEPARADOS ENTRE SI POR VÍRGULA.

3) O NOME DE UMA VARIÁVEL OU VALOR DE UMA CONSTANTE ENTRE ASPAS, SIGNIFICA SEU ENDEREÇO NA MEMÓRIA.

EXPRESSÃO:

```
 $(P $T $(N + ('5' * $J)) N + ('7' * $J) $(I N) 'F' F(A,$B))
```

CÓDIGO GERADO:

```
        STACK   P
        STACK   T
        IND
        STACK   N
        STACK   '5'
        STACK   J
        IND
        MULT
        B    L  FALHA
        SOMA
        B    L  FALHA
        IND
        STACK   N
        STACK   '7'
        STACK   J
```

```
        IND
        MULT
        B     L   FALHA
        SOMA
        B     L   FALHA
        STACK     I
        STACK     N
        CONC      2
        IND
        STACK     'F'
        STACK     A
        STACK     B
        IND
        FUNC      DF,2
        B     L   FALHA
        CONC      7
        IND
```

NO FINAL DA EXECUÇÃO DESTE CÓDIGO, O VALOR DA EXPRESSÃO FICA NO TOPO DA PILHA DE PARÂMETROS.

NOTA - FALHA É O ENDEREÇO DA PARTE DA DECLARAÇÃO QUE PROCESSA A FALHA.

2.2.3 - SUBROTINAS AUXILIARES:

EXISTEM ALGUMAS SUBROTINAS QUE SÃO USADAS PELA SUBROTINA QUE COMPILA UMA EXPRESSÃO ARITMÉTICA. DAREMOS A SEGUIR AS MAIS IMPORTANTES COM SUAS FUNÇÕES.

2.2.3.1 - AVALIA UM NOME VÁLIDO (NOME):

ESTA SUBROTINA APENAS VERIFICA SE OS CARACTERES A SEGUIR SÃO VALIDOS EM UM NOME DE VARIÁVEL, SEPARANDO-OS E MONTANDO-OS EM UM VETOR SEPARADO.

SE O PRIMEIRO CARACTER A SER EXAMINADO FOR UMA ASPA, ELA TRANSFERE PARA ESTE VETOR TODOS OS CARACTERES A SEGUIR, ATÉ A PROXIMA ASPA, NÃO VERIFICANDO A VALIDADE DOS CARACTERES.

ALÉM DISTO, CONTA O NÚMERO DE CARACTERES DO NOME OU CONSTANTE E LIGA UM INDICADOR PARA AVISAR SE FOI DETETADA UMA CONSTANTE OU UM NOME.

2.2.3.2 - OPERAÇÕES ARITMÉTICAS E END. INDIRETO (AVARS):

COMO SERÁ EXPLICADO MAIS ADIANTE, HÁ NA FASE COMPILAÇÃO, UMA PILHA DE OPERADORES PARA A ANÁLISE SINTÁTICA.

ESTA SUBROTINA, APENAS REALIZA UMA OPERAÇÃO QUE SERIA REPETIDA VÁRIAS VEZES EM DIVERSOS PONTOS DO COMPILADOR.

ELA EXAMINA O ÚLTIMO OPERADOR DA PILHA DE OPERADORES. SE ESTE FOR UM OPERADOR ARITMETICO OU INDICADOR DE ENDEREÇAMENTO INDIRETO, ELA GERA O CÓDIGO OBJETO RESPECTIVO. SE NÃO FOR NENHUM DOS ANTERIORES, ELA LIGA UM INDICADOR PARA AVISAR QUE NÃO GEROU CÓDIGO E DEVOLVE.

2.2.3.3 - RECURSIVIDADE (SAVE E VOLTA):

ESTAS SUBROTINAS FAZEM A RECURSIVIDADE ENTRE A SUBROTINA QUE COMPILA UMA EXPRESSÃO E A QUE COMPILA FUNÇÕES. ELAS RECALCAM O ENDEREÇO DE RETORNO E AS VARIÁVEIS LOCAIS, RESTAURANDO-OS QUANDO O CONTROLE FOR DEVOLVIDO.

DEVIDO AO FATO DE NÃO IMPLANTARMOS FUNÇÕES, ESTAS SUBROTINAS NÃO FORAM IMPLANTADAS.

2.2.4 - COMPILAÇÃO DE UMA EXPRESSÃO:

COMO SABEMOS, SNOBOL É UMA UMA LINGUAGEM COM UMA SINTAXE SIMPLES E FÁCIL DE SER ANALISADA.

NÃO ADMITE, POR EXEMPLO, PRIORIDADES EM OPERAÇÕES ARITMÉTICAS, SENDO QUE A ÚNICA PRIORIDADE EXISTENTE, ALÉM DA DADA POR PARÊNTESES, É DA OPERAÇÃO ARITMÉTICA SOBRE A CONCATENAÇÃO.

ESTE FATO, TORNA DESNECESSÁRIO UMA ASSOCIAÇÃO DE PRIORIDADES DOS OPERADORES, JÁ QUE COM APENAS UMA PRIORIDADE, BASTA UM SIMPLES TESTE PARA IDENTIFICARMOS QUAL A OPERAÇÃO A SER REALIZADA EM PRIMEIRO LUGAR.

OS SÍMBOLOS QUE CONTROLAM A GERAÇÃO DE CÓDIGO OBJETO NA COMPILACAO SAO OS SEGUINTES:

( ) $ + - / * ** E BRANCO(B)

PARA FACILITAR A EXPLANAÇÃO, CHAMAREMOS OS OPERADORES ARITMÉTICOS DE AR, E O CARACTER BRANCO DE B.

O PROCEDIMENTO USADO PARA A COMPILAÇÃO É O SEGUINTE:

TODA VEZ QUE É DETETADO UM OPERANDO, GERAMOS O CODIGO OBJETO PARA RECALCA-LO NA PILHA DE PARÂMETROS.

HAVERÁ TAMBÉM, NA FASE COMPILAÇÃO, UMA PILHA NA QUAL SÃO RECALCADOS OS OPERADORES À MEDIDA QUE FOREM APARECENDO. QUANDO DETETAMOS UM OPERADOR DE CONCATENAÇÃO (B), É FEITO O TESTE SE O TOPO DA PILHA CONTÉM UM OPERADOR ARITMÉTICO, EM CASO AFIRMATIVO, É GERADO O CÓDIGO PARA PERFAZER A OPERAÇÃO ARITMÉTICA COM OS OPERANDOS QUE JÁ FORAM RECALCADOS NA PILHA DE PARÂMETROS E O OPERADOR AR É RETIRADO DA PILHA, SENDO RECALCADO O DE CONCATENAÇÃO. DESTE MODO, DEIXAMOS NA PILHA OS OPERADORES DA CONCATENACAO, QUE NO FINAL SAO CONTADOS E ENTÃO É GERADO O CODIGO PARA CONCATENAR O NÚMERO DE CADEIAS DADO PELO CONTADOR MAIS UM.

O OPERADOR DE ENDEREÇAMENTO INDIRETO É UM OPERADOR UNITÁRIO, ISTO É, APLICA-SE A APENAS UM OPERANDO. NÃO RECEBE TRATAMENTO ESPECIAL, SENDO RECALCADO QUANDO APARECER E RETIRADO QUANDO O OPERADOR A SER INSERIDO FOR B OU AR.

O SIMBOLO '(' NÃO É PROPRIAMENTE UM OPERADOR, MAS TAMBÉM É RECALCADO PARA SERVIR DE DELIMITADOR. QUANDO ENCONTRAMOS O SIMBOLO ')', RETIRAMOS DA PILHA (E LOGICAMENTE GERAMOS O CÓDIGO OBJETO RESPECTIVO) TODOS OS OPERADORES ATÉ ENCONTRARMOS O DELIMITADOR '(' QUE NESTE PONTO TAMBÉM É RETIRADO DA PILHA.

HAVERÁ TAMBÉM UM CONTADOR DE PARÊNTESES, QUE CHAMAREMOS DE $PAR, PARA A VERIFICAÇÃO DE PARÊNTESES MAL INTERCALADOS.

A TABELA ABAIXO RESUME O PROCEDIMENTO DA SUBROTINA DE COMPILAÇÃO DE EXPRESSÃO:

SÍMBOLO	PROCEDIMENTO
$	RECALCA NA PILHA DE OPERADORES.
(	IDEM E INCREMENTA $PAR DE 1.
AR	SE ANTERIOR FOR AR: ERRO; CASO CONTRÁRIO RECALCA AR.
B	VOLTA NA PILHA RETIRANDO '$' OU AR ATE ENCONTRAR '(', B OU ESVAZIAR A PILHA. RECALCA B NA PILHA.
)	VOLTA RETIRANDO TODOS OS SIMBOLOS E AO INVÉS DE GERAR CODIGO AO ENCONTRAR B, CONTA-OS. ASSIM AO ENCONTRAR O SIMBOLO '(', QUE É RETIRADO DA PILHA, É GERADO O CÓDIGO PARA CONCATENAÇÃO. DECREMENTA $PAR DE 1 TESTANDO SE PASSOU A NEGATIVO; EM CASO AFIRMATIVO: ERRO.
⊣	ESTE CARACTER REPRESENTA O FINAL DA EXPRESSÃO E

DEPENDE DO CAMPO DE ONDE A SUBROTINA FOI CHAMADA. O PROCEDIMENTO TOMADO E ESVAZIAR A PILHA E TERMINAR A COMPILAÇÃO. SE $PAR FOR DIFERENTE DE ZERO: ERRO.

PARA FACILITAR A COMPREENSAO DO METODO DESCRITO ACIMA, DAMOS A SEGUIR, A COMPILAÇÃO DE UMA EXPRESSÃO DANDO A CADA ELEMENTO EXAMINADO, A SITUACAO DA PILHA DE OPERADORES, O VALOR DA EXPRESSAO RESTANTE E O CONTADOR DE PARÊNTESES. NOTE QUE QUANDO NOS REFERIMOS A OPERADOR ARITMÉTICO, É CONSIDERADO COMO PARTE DO OPERADOR OS CARACTERES BRANCOS ANTES E APÓS O MESMO.

A EXPRESSÃO ESCOLHIDA PARA EXEMPLO E:

```
$(A $J $(C + ('5' * $D)) N + ('57' / $D) 'FIM') |
```

COD. GERADO	PILHA OPERADORES	$PAR	DECL. RESTANTE
		0	$(A $J...
	$	0	(A $J ...
	$(	1	A $J $...
STACK A	$(	1	$J $(...
	$(B	1	$J $(C...
	$(B$	1	J $(C ...
STACK J	$(B$	1	$(C +...
IND	$(BB	1	$(C + ...
	$(BB$	1	(C + (...
	$(BB$(	2	C + N'...
STACK C	$(BB$(	2	+ ('5...
	$(BB$(+	2	('5' *...
	$(BB$(+(	3	'5' * ...
STACK '5'	$(BB$(+(	3	* $D)...
	$(BB$(+(*	3	$D)) N...
	$(BB$(+(*$	3	D)) N ...
STACK D	$(BB$(+(*$	3	)) N +...
IND	$(BB$(+(*	3	)) N +...
MULT	$(BB$(+(	3	)) N +...
B L FALHA	$(BB$(+(	3	)) N +...
SOMA	$(BB$(	2	) N + ...
B L FALHA	$(BB$(	2	) N + ...
IND	$(BBB	1	N + ('...
STACK N	$(BBB	1	+ ('5...
	$(BBB+	1	('57' ...
	$(BBB+(	2	'57' /...
STACK '57'	$(BBB+(	2	/ $D)...
	$(BBB+(/	2	$D) 'F...
	$(BBB+(/$	2	D) 'FI...
STACK D	$(BBB+(/$	2	) 'FIM...
IND	$(BBB+(/	2	) 'FIM...
DIVIZ	$(BBB+(	2	) 'FIM...
B L FALHA	$(BBB+(	2	) 'FIM...
SOMA	$(BBBB	1	'FIM')...

```
B  L  FALHA   - $(BBBB          -  1   - 'FIM')...
STACK 'FIM'   - $(BBBB          -  1   - )⊣
CONC 5        - $               -  0   - ⊣
IND           -                 -  0   -
```

NA VERDADE (COMO PODEMOS VERIFICAR PELO DIAGRAMA DE BLOCOS NO APÊNDICE I), O OPERADOR ARITMÉTICO SÓ É IDENTIFICADO APÓS RECALCARMOS O CARACTER BRANCO QUE VEM ANTES DO MESMO. ASSIM, AO IDENTIFICA-LO, RETIRAMOS (SEM GERAR CODIGO NENHUM) O BRANCO ANTERIOR E ENTÃO PROCEDEMOS NORMALMENTE.

COMO DISSEMOS ANTERIORMENTE, A SUBROTINA QUE COMPILA UMA EXPRESSÃO É BEM GENÉRICA, PODENDO SER CHAMADA DE QUALQUER CAMPO ONDE UMA EXPRESSÃO SEJA VÁLIDA. CADA CAMPO TERMINA POR UM CARACTER ESPECIAL QUE DEVE SER RECONHECIDO PELA SUBROTINA COMO INDICADOR DE TÉRMINO DA COMPILAÇÃO. QUANDO NÃO AVISAMOS NADA A SUBROTINA, ELA COMPILA A MENOR EXPRESSÃO POSSÍVEL. MAS EXISTEM TRES INDICADORES PARA AVISA-LA QUE CHAMAMOS NO CAMPO DE SUBSTITUIÇÃO, DE TRANSFERÊNCIA OU AO AVALIAR OS PARÂMETROS DE UMA FUNÇÃO.

A SUBROTINA PODE SER CHAMADA DOS SEGUINTES CAMPOS:

1) CADEIA DE REFERÊNCIA.
2) AVALIAÇÃO DO NOME DE UM ELEMENTO DE PADRÃO ARBITRÁRIO.
3) AVALIAÇÃO DO TAMANHO DE UM ELEMENTO DE TAMANHO FIXO.
4) AVALIAÇÃO DO NOME DE UM ELEMENTO BALANCEADO.
5) CAMPO DE SUBSTITUIÇÃO.
6) CAMPO DE TRANSFERÊNCIA.

NOS TRES PRIMEIROS CASOS, A SUBROTINA TERMINA A COMPILAÇÃO NA MENOR EXPRESSÃO POSSÍVEL, ISTO É, QUANDO ENCONTRA UM DOS SEGUINTES CARACTERES: BRANCO, '*' OU '/' COM O CONTADOR DE PARÊNTESES NULO. ADIANTAMOS QUE É DE RESPONSABILIDADE DA ROTINA QUE CHAMA, O TESTE PARA VERIFICAR SE A EXPRESSÃO TERMINOU EM UM CARACTER VÁLIDO.

AO SER CHAMADA PARA AVALIAR UMA LISTA DE PARÂMETROS, A SUBROTINA SO TERMINA A COMPILAÇÃO AO ENCONTRAR UMA ',' OU ')' COM O CONTADOR DE PARÊNTESES NULO. SE ENCONTRAR UM ')' ELA LIGA UM INDICADOR PARA AVISAR A ROTINA QUE A CHAMOU QUE AQUELE E O ÚLTIMO PARÂMETRO.

SENDO CHAMADA PARA AVALIAR UM CAMPO DE SUBSTITUIÇÃO, A COMPILAÇÃO É TERMINADA QUANDO FOR DETETADA UMA '/' DO CAMPO DE TRANSFERÊNCIA OU O FINAL DO REGISTRO, SEMPRE COM O CONTADOR DE PARÊNTESES NULO.

A COMPILAÇÃO DO CAMPO DE TRANSFERÊNCIA É SEMELHANTE A DO NOME DE UM ELEMENTO BALANCEADO, POIS AMBAS TERMINAM COM ')' E O CONTADOR DE PARÊNTESES NULO.

PARA COMPLETAR, FALTA-NOS APENAS FALAR SOBRE A VERIFICAÇÃO DA SINTAXE DA EXPRESSÃO. A SEGUIR MOSTRAMOS A TABELA DE SINTAXE DE UMA EXPRESSAO SNOBOL. NESTA TABELA, SE DETERMINADA POSIÇÃO FOR IGUAL A 1, SIGNIFICA QUE UM ELEMENTO

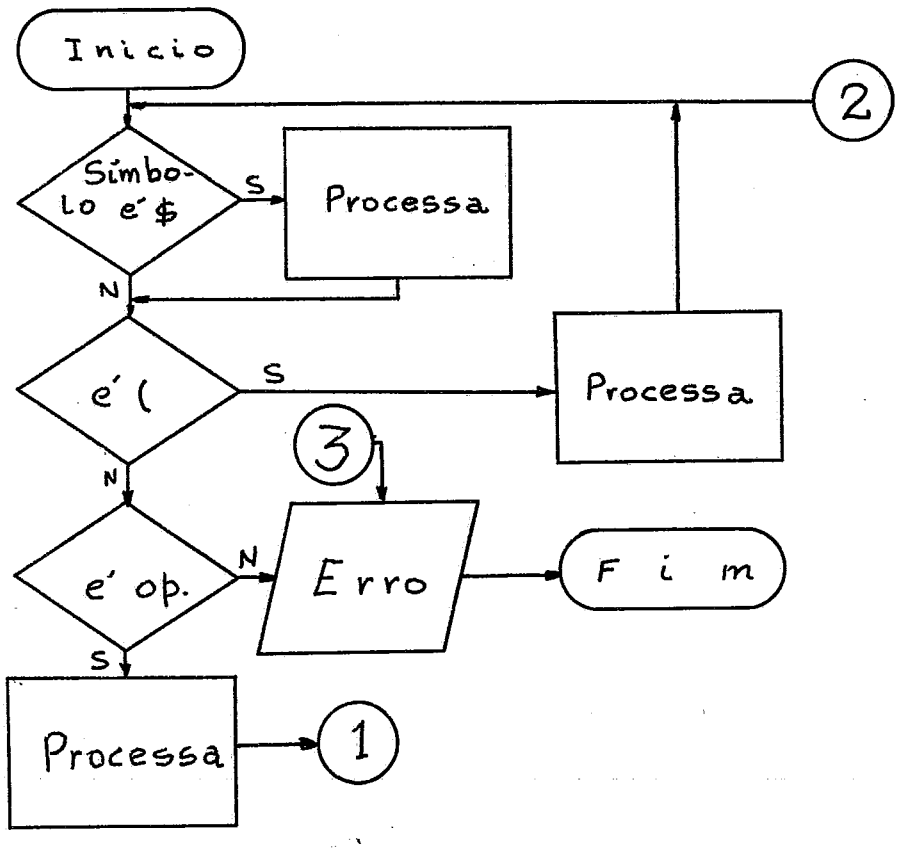
DA ENTRADA VERTICAL PODE VIR EM SEGUIDA DO ELEMENTO DA ENTRADA HORIZONTAL.

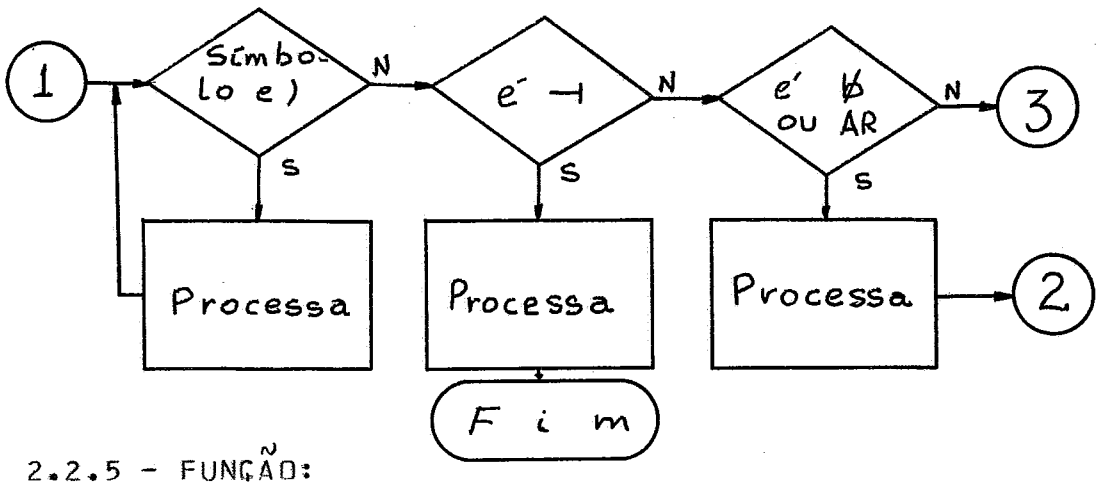
	$	(	op.	)	⊬	AR	⊣
⊢	1	1	1				
$		1	1				
(	1	1	1				
op.				1	1	1	1
)				1	1	1	1
⊬	1	1	1				
AR	1	1	1				

op. = OPERANDO

NOTA - O SIMBOLO ⊢ REPRESENTA O INICIO DA EXPRESSÃO.

PELA TABELA ACIMA VEMOS QUE OS ELEMENTOS ESTÃO AGRUPADOS, O QUE TORNA DESNECESSÁRIA A MANUTENÇÃO DA MESMA NA MEMÓRIA, BASTANDO PARA ISTO QUE SE TESTE OS CARACTERES EM UMA SEQUÊNCIA LÓGICA. DAMOS ABAIXO UMA ESTRUTURA SIMPLIFICADA DA SUBROTINA COM A FINALIDADE DE MOSTRAR A SEQUÊNCIA DOS TESTES.





2.2.5 - FUNÇÃO:

A SUBROTINA QUE AVALIA UMA FUNÇÃO É CHAMADA PELA SUBROTINA QUE AVALIA UMA EXPRESSÃO AO SER DETETADO O CARACTER '(' APÓS UM NOME VALIDO. UMA FUNÇÃO É TRATADA COMO SE FOSSE UM OPERANDO PELA SUBROTINA QUE AVALIA A EXPRESSÃO.

ESTA SUBROTINA, APENAS PROCURA O DESCRITOR DE FUNÇÃO NA TABELA DE SÍMBOLOS E CHAMA (RECURSIVAMENTE) A SUBROTINA QUE COMPILA UMA EXPRESSÃO PARA AVALIAR OS PARÂMETROS. HÁ UM CONTADOR DO NÚMERO DE PARÂMETROS PARA PODER SER GERADO O CÓDIGO OBJETO NO FINAL DA AVALIAÇÃO. O ÚLTIMO PARÂMETRO É DETETADO PELO CARACTER ')' (COMO FOI DITO ANTERIORMENTE) NO FINAL DA EXPRESSÃO.

ESTA SUBROTINA NÃO FOI IMPLANTADA NESTA VERSÃO DO SISTEMA.

2.2.6 - VARIÁVEIS LOCAIS:

PARA FINALIZAR ESTA PARTE DE COMPILAÇÃO DE EXPRESSÕES CITAMOS A SEGUIR QUAIS AS VARIÁVEIS LOCAIS DAS SUBROTINAS QUE AVALIAM UMA EXPRESSÃO E UMA FUNÇÃO.

EXPRESSÃO: A SUBROTINA QUE AVALIA EXPRESSÃO TEM COMO VARIÁVEIS LOCAIS O INICIO E O TOPO DA PILHA DE OPERANDOS E O CONTADOR DE PARÊNTESES.

FUNÇÃO: A SUBROTINA QUE AVALIA UMA FUNÇÃO TEM COMO VARIÁVEIS LOCAIS, O CONTADOR DE PARÂMETROS E O ENDEREÇO DO DF.

2.3 - PROCURA NA TABELA DE SÍMBOLOS:

2.3.1 - CODIGO ALEATÓRIO:

COMO DISSEMOS ANTERIORMENTE, A TABELA DE SÍMBOLOS É DIVIDIDA EM 16 PARTES. AO INSERIRMOS UM NOME, APLICA-SE SOBRE O MESMO UM ALGORÍTMO GERANDO O CÓDIGO ALEATÓRIO QUE NOS DARÁ A PARTE ONDE INSERIR O NOME.

O ALGORÍTMO USADO E O SEGUINTE: MULTIPLICA-SE OS CODIGOS ALFABÉTICOS DOS CARACTERES IGNORANDO QUANDO A CAPACIDADE ARITMÉTICA DA MAQUINA FOR ULTRAPASSADA E TOMA-SE OS 4 BITS CENTRAIS DANDO-SE ASSIM O CÓDIGO ALEATÓRIO MÓDULO 16.

2.3.2- INICIALIZAÇÃO DA TS:

A TABELA DE SÍMBOLOS É INICIALIZADA PELO PROGRAMA INICIALIZADOR. ESTA INICIALIZAÇÃO CONSISTE EM DAR O VALOR INICIAL (Λ) AO VETOR DOS INÍCIOS DAS 16 PARTES E INSERIR OS RÓTULOS E VARIÁVEIS RESERVADAS NA TS.

NESTA PARTE SÃO ALOCADAS E INSERIDAS NA TS AS VARIÁVEIS QUOTE, SYSPIT, SYSPOT E SYSPPT.

OS RÓTULOS RETURN E FRETURN, SÃO TAMBÉM INSERIDOS, MAS EM SEU ENDEREÇO COLOCAMOS O ENDEREÇO DA SUBROTINA QUE CONTROLA A RECURSIVIDADE.

SAO INSERIDOS TAMBÉM OS NOMES DAS FUNÇÕES DEFINIDAS E ALOCADOS OS RESPECTIVOS DF'S.

2.3.3- FUNÇÕES DA SUBROTINA DE PROCURA NA TS:

2.3.3.1- ALOCAÇÃO DE VARIÁVEIS E CONSTANTES:

QUANDO A SUBROTINA DE PROCURA NA TS (QUE DE AGORA EM DIANTE CHAMAREMOS DE PROC) É CHAMADA PARA UMA CONSTANTE OU UMA VARIÁVEL, PRIMEIRAMENTE É FEITA UMA PESQUISA PARA SABER SE A VARIÁVEL OU CONSTANTE JÁ ESTÁ NA TS. SE ESTA PESQUISA FALHAR, A SUBROTINA SE ENCARREGA DE ALOCAR A CONSTANTE OU A VARIÁVEL COM VALOR NULO ANTES DE INSERI-LA NA TS.

2.3.3.2- REFERÊNCIA FUTURA:

JÁ NOS REFERIMOS ANTERIORMENTE A REFERÊNCIA FUTURA (VER PARAG. 1.2.3.1) E COMO É REALIZADA.

RESTA-NOS DIZER QUE ELA É CONTROLADA PELA SUBROTINA PROC.

EXISTEM 2 INDICADORES NA AREA DE COMUNICAÇÃO PARA INDICARMOS SE QUEREMOS INSERIR OU CONSULTAR A TS.

QUANDO QUEREMOS CONSULTAR UM RÓTULO E ELE AINDA NÃO FOI DEFINIDO, ELE É INSERIDO NA TS E É INICIADA UMA LISTA DE APONTADORES COMO FOI DITO ANTERIORMENTE. O ENDEREÇO DO INÍCIO DA LISTA É COLOCADO NO LUGAR DO RÓTULO COM VALOR NEGATIVO PARA QUE POSSA SER RECONHECIDO POSTERIORMENTE.

AO CHAMARMOS PROC PARA INSERIR UM RÓTULO E ESTE JÁ FOI INSERIDO NA TS, É TESTADO O ENDERECO DA TS PARA SABER SE O RÓTULO JÁ FOI DEFINIDO OU NÃO. SE O ENDEREÇO FOR POSITIVO, É ERRO, POIS TEREMOS RÓTULO DUPLICADO. SE FOR NEGATIVO, É ENTÃO DESFEITA A LISTA APONTANDO TODOS OS ELEMENTOS PARA O CONTADOR DE INSTRUÇÕES (QUE APONTA PARA A PRÓXIMA INSTRUCAO A SER GERADA).

AO FINAL DA COMPILAÇÃO, É CHAMADO UM PROGRAMA QUE VERIFICA SE TODOS OS ENDEREÇOS DA TS SÃO POSITIVOS. CASO HAJA ALGUM NEGATIVO, É IMPRESSO UMA MENSAGEM DE ERRO E LIGADO O INDICADOR PARA NÃO EXECUÇAO DO PROGRAMA OBJETO.

2.3.3.3- ENTRADA E SAIDA:

QUANDO PROCURAMOS NA TS UMA VARIÁVEL DO TIPO QUE ESTEJA VINCULADA A UMA OPERACAO E/S, É A SUBROTINA PROC QUEM IDENTIFICA E CONTROLA O CÓDIGO GERADO PARA PERFAZER A OPERAÇÃO DESEJADA.

SE A VARIÁVEL ESTA VINCULADA A UMA OPERAÇÃO DE ENTRADA, A PRÓPRIA SUBROTINA PROC GERA O CÓDIGO OBJETO NECESSÁRIO PARA PERFAZER A OPERAÇÃO DE LEITURA.

SE POR OUTRO LADO, A OPERAÇÃO FOR DE SAIDA, ELA APENAS LIGA UM INDICADOR NA ÁREA DE COMUNICAÇÃO PARA QUE, AO PROCESSARMOS A PARTE DE SUCESSO DA DECLARAÇÃO, SEJA GERADO O CÓDIGO NECESSÁRIO PARA A SAIDA DESEJADA.

POR EXEMPLO, AO CHAMARMOS PROC PARA SABER O ENDEREÇO DA VARIÁVEL SYSPIT ELA IMEDIATAMENTE GERA O CÓDIGO CALL SPIT (QUE É A SUBROTINA ASSOCIADA A VARIÁVEL SYSPIT).

SE CHAMARMOS PARA SYSPOT, ELA APENAS LIGA O INDICADOR DOUT5 PARA QUE AO PROCESSARMOS A PARTE DE SUCESSO,

SEJA GERADO O CODIGO CALL SPOT.

2.3.3.4- IDENTIFICAÇÃO DA FUNÇÃO DEFINE:

A FUNÇÃO DEFINE É IDENTIFICADA PELA SUBROTINA PROC E É TRATADA DE MANEIRA ESPECIAL.

AO SER DETETADA, A PRÓPRIA SUBROTINA FAZ A ANÁLISE SINTÁTICA DO COMANDO MONTA O DF E INSERE A FUNÇÃO NA TS.

2.3.4- SUBROTINA PROC:

A SUBROTINA PROC, TAMBÉM FOI FEITA O MAIS GENÉRICA POSSÍVEL, ISTO É, ACEITA INSERIR QUALQUER SÍMBOLO NA TS.

PARA QUE NÃO CONFUNDA UM RÓTULO COM UMA FUNÇÃO, VARIÁVEL OU CONSTANTE QUE TENHAM O MESMO NOME OU VALOR, AO CHAMA-LA DEVEMOS LIGAR UM INDICADOR NA ÁREA DE COMUNICAÇÃO PARA INDICARMOS SE QUEREMOS REFERENCIAR UMA CONSTANTE, VARIÁVEL, RÓTULO OU FUNÇÃO.

AO IDENTIFICARMOS O NOME PROCURADO COM O NOME DA TS, VERIFICAMOS SE O TIPO DE ELEMENTO É IGUAL AO QUE FOI REFERENCIADO. CASO NÃO SEJA, CONTINUAMOS A PESQUISA.

AO ENCONTRARMOS UM ELEMENTO TIPO FUNÇÃO, HÁ DOIS CASOS A CONSIDERAR: 1) QUEREMOS PROCURAR UMA FUNÇÃO - NESTE CASO ESTAMOS INTERESSADOS NO DF QUE É DEVOLVIDO. 2) QUEREMOS PROCURAR UMA VARIÁVEL - NESTE CASO ESTAMOS INTERESSADOS NO ENDEREÇO ONDE FOI COLOCADO O VALOR DA FUNÇÃO. ASSIM, É NECESSARIO QUE O ENDEREÇO A SER DEVOLVIDO SEJA O ENDERECO DO VALOR DA FUNÇÃO QUE É PROCURADO NO DF E DEVOLVIDO. NOTE-SE QUE QUANDO FALAMOS DEVOLVIDO, QUEREMOS DIZER QUE FOI O ENDEREÇO QUE RETORNOU COMO RESPOSTA.

RESTA-NOS RESSALTAR, QUE POR MOTIVO DE FACILIDADE DE PROGRAMAÇÃO, OS NOMES SÃO GUARDADOS NA TS SEM COMPACTAÇÃO, ISTO É, UM CARACTER POR PALAVRA.

PARA MAIORES DETALHES, VER DIAGRAMA DE BLOCOS DO APENDICE I.

2.4- CORPO DO COMPILADOR:

CHAMAMOS DE CORPO DO COMPILADOR O PROGRAMA QUE

VARRE A DECLARAÇÃO IDENTIFICANDO OS CAMPOS, CONTROLANDO A CHAMADA DE SUBROTINAS AUXILIARES E CONTROLANDO O CÓDIGO OBJETO GERADO.

COMO DISSEMOS ANTERIORMENTE, É CONSTITUIDO DE VÁRIAS PARTES DISTINTAS QUE SERÃO DETALHADAS A SEGUIR.

2.4.1- EDICAO E LISTAGEM:

ESTA PARTE SUPRIME OS BRANCOS SUPÉRFLUOS E VERIFICA SE CADA CARTÃO CONTÉM PARÊNTESES E ASPAS BALANCEADOS. ESTA RESTRIÇÃO É FEITA PARA QUE NÃO PAREMOS A COMPILAÇÃO NO MEIO DA AVALIAÇÃO DE UM TERMO.

ESTA PARTE, DEPENDENDO DOS INDICADORES DE LISTAGEM, LISTA O CARTÃO QUE CONTÉM A DECLARAÇÃO. DETETA TAMBÉM O RÓTULO END E LIGA UM INDICADOR PARA QUE A COMPILAÇÃO SEJA TERMINADA NA PARTE QUE FINALIZA A DECLARACAO ANTERIOR.

SE O PRIMEIRO CARACTER DO CARTAO FOR '*', ELE NÃO É COMPILADO, PASSANDO PARA A LEITURA DO PRÓXIMO CARTÃO.

SE FOR DETETADO UM CARTÃO DE CONTRÔLE É TRANSFERIDO O CONTRÔLE PARA A ROTINA QUE PROCESSA ESTE TIPO DE CARTÃO. ESTA ROTINA MODIFICA OS INDICADORES NECESSÁRIOS OU PERFAZ A OPERAÇÃO DESEJADA (SALTA PÁGINA, SALTA ESPAÇO, ETC...).

NESTE PONTO, É TESTADO SE O CARTÃO É CARTÃO DE CONTINUAÇÃO E EM CASO AFIRMATIVO, É TRANSFERIDO O CONTRÔLE PARA O PROCESSAMENTO DE CONTINUAÇÃO QUE POR SUA VEZ DARÁ O CONTRÔLE PARA A PARTE ONDE A COMPILAÇÃO FOI PARADA NO ÚLTIMO CARTÃO.

SE NENHUMA DAS CONDIÇÕES ACIMA OCORRER, É DADO O CONTRÔLE PARA A ROTINA QUE TERMINA A DECLARAÇÃO ANTERIOR.

2.4.2- PROCESSAMENTO DE CONTINUAÇÃO:

QUANDO O PRIMEIRO CARACTER DO CARTÃO FOR UM PONTO, O CONTRÔLE É TRANFERIDO PARA UMA ROTINA DE CONTINUAÇÃO.

ESTA PARTE TESTA, ATRAVÉS DE INDICADORES DA ÁREA DE COMUNICAÇÃO, O LOCAL ONDE A COMPILAÇÃO DO CARTÃO ANTERIOR FOI SUSPENSA E TRANSFERE O CONTRÔLE PARA LÁ.

2.4.3- TERMINA DECLARAÇÃO ANTERIOR:

HÁ APENAS 3 PARTES ONDE A COMPILAÇÃO PODE SER INTERROMPIDA: APÓS A CADEIA DE REFERÊNCIA, NO PADRÃO OU NA SUBSTITUIÇÃO.

COMO VEREMOS MAIS ADIANTE, A ROTINA DE PADRÃO TEM 2 ENTRADAS: UMA PARA QUANDO A COMPILAÇÃO FOR SUSPENSA NO FINAL DA CADEIA DE REFERÊNCIA E OUTRA QUANDO FOR SUSPENSA NO MEIO DO PADRÃO. NO PRIMEIRO CASO A ROTINA QUE TERMINA A DECLARAÇÃO ANTERIOR NÃO GERA CODIGO ALGUM, INDO PARA A PARTE QUE PROCESSA A PARTE DE SUCESSO. NO SEGUNDO CASO, E GERADO O CÓDIGO OBJETO PARA A CHAMADA DA SUBROTINA PATT QUE FARÁ A VARREDURA DO PADRÃO. ALÉM DISTO, O ÚLTIMO NO DO DP É FECHADO (VER PARAGRAFO 2.4.6).

SE A DECLARAÇÃO ANTERIOR FOR TERMINADA EM UM CAMPO DE SUBSTITUIÇÃO, SE NECESSÁRIO, É GERADO O CÓDIGO DE CONCATENAÇÃO DAS PARTES QUE VIERAM EM CARTÕES SEPARADOS (ISTO SERÁ EXPLICADO NA PARTE DE SUBSTITUIÇÃO, APENAS ADIANTAMOS QUE CADA CARTÃO DE CONTINUAÇÃO DE UMA PARTE DE SUBSTITUIÇÃO É COMPILADO EM SEPARADO COMO SE FOSSE UMA EXPRESSÃO E NO FINAL SÃO CONCATENADOS) E GERADO CÓDIGO PARA CHAMADA DA SUBROTINA ASSIG QUE FARA A SUBSTITUIÇÃO.

SE TERMINARMOS A COMPILAÇÃO AO ENTRAR NA CADEIA DE REFERÊNCIA, NÃO É NECESSÁRIO TERMINAR NADA.

NO CASO DE NÃO TERMOS TERMINADO A DECLARAÇÃO EM NENHUM DOS CAMPOS ACIMA É DESVIADO O CONTRÔLE PARA A PARTE QUE PROCESSA SUCESSO (NO CASO DE TERMINARMOS UMA DECLARAÇÃO, APÓS O CODIGO GERADO, O CONTROLE TAMBÉM É TRANSFERIDO PARA ESTA PARTE). ESTA PARTE CONSISTE EM GERAR O CÓDIGO OBJETO PARA A SAIDA DE DADOS (SE NECESSÁRIO).

APOS PROCESSAR A PARTE DE SUCESSO, É PROCESSADA A PARTE DE FALHA QUE SE RESUME NA GERAÇÃO DO CÓDIGO OBJETO PARA CHAMAR A SUBROTINA QUE ESVAZIA A PILHA DE PARÂMETROS.

CASO A DECLARAÇÃO ANTERIOR TENHA CAMPOS DE TRANSFERÊNCIA E AS PARTES DE SUCESSO E FALHA JA TENHAM SIDO PROCESSADAS, ESTA PARTE DO COMPILADOR SIMPLESMENTE NÃO REALIZA NADA POIS OS INDICADORES JÁ FORAM DESLIGADOS.

NESTE PONTO É TESTADO O INDICADOR QUE CONTROLA O FINAL DA COMPILAÇÃO. SE ELE ESTIVER LIGADO, É INSERIDO O RÓTULO END NA TS PARA QUE SEJAM DESFEITAS AS REFERÊNCIAS A ESTE RÓTULO. A SEGUIR É CALCULADO O ENDEREÇO DE EXECUÇÃO DO PROGRAMA E ENTÃO CHAMAMOS UM OUTRO PROGRAMA PARA VERIFICAR A TS, PROGRAMA ESTE QUE POR SUA VEZ, SE NÃO FOI DETETADO ERRO NA COMPILAÇÃO, CHAMA AS SUBROTINAS DA FASE EXECUÇÃO.

SE O INDICADOR DE TÉRMINO DA COMPILAÇÃO ESTIVER DESLIGADO, SÃO INICIALIZADOS OS INDICADORES, SENDO O CONTRÔLE TRANSFERIDO PARA A ROTINA QUE PROCESSA O RÓTULO.

2.4.4- RÓTULO:

DE TODAS AS ROTINAS QUE COMPÕEM O CORPO PRINCIPAL, ESTA É A MAIS SIMPLES.

ELA VERIFICA A EXISTÊNCIA DE RÓTULO; EM CASO AFIRMATIVO CHAMA A SUBROTINA QUE AVALIA O NOME E A SEGUIR CHAMA A SUBROTINA PROC PARA INSERIR O RÓTULO NA TS.

AO TERMINAR O NOME DO RÓTULO, É TESTADO O PRÓXIMO CARACTER. SE FOR BRANCO, TRANSFERIMOS CONTRÔLE PARA A ROTINA QUE AVALIA A CADEIA DE REFERÊNCIA. SE FOR O CARACTER INDICADOR DE FINAL DE CARTÃO É LIGADO UM INDICADOR PARA QUE, SE HOUVER CONTINUAÇÃO, SEJA DADO O CONTROLE PARA A CADEIA DE REFERÊNCIA.

2.4.5- CADEIA DE REFERÊNCIA:

O PRIMEIRO PROCEDIMENTO DESTA ROTINA É TESTAR SE O CAMPO DA CADEIA DE REFERÊNCIA COMECA COM ASPA OU '('. EM CASO AFIRMATIVO, LIGAMOS UM INDICADOR PARA QUE NÃO POSSA HAVER SUBSTITUIÇÃO NESTA DECLARAÇÃO POIS A CADEIA DE REFERENCIA É UMA CONSTANTE OU UM GRUPAMENTO, RESPECTIVAMENTE.

É IMPORTANTE NESSE PONTO SABER COMO A CADEIA DE REFERÊNCIA É TRATADA NA FASE EXECUÇÃO.

HAVERÁ UMA PALAVRA NA ÁREA DE COMUNICAÇÃO DURANTE A FASE EXECUÇÃO, QUE CONTERÁ O VALOR DO ENDEREÇO DA CADEIA DE REFERÊNCIA. QUALQUER SUBROTINA QUE NECESSITE DESTE ENDERÊÇO, BUSCA-O ALI.

A CADEIA DE REFERÊNCIA PODE SER UM GRUPAMENTO E COMO VEREMOS ADIANTE, O RESULTADO DE UM GRUPAMENTO E COLOCADO EM UMA CADEIA INTERMEDIÁRIA.

DESTA MANEIRA, NÃO BASTA AVALIARMOS A EXPRESSÃO E COLOCAR O ENDEREÇO DO RESULTADO NA ÁREA DE COMUNICAÇÃO, É NECESSÁRIO TAMBÉM QUE SE FAÇA UM CONTRÔLE DAS CADEIAS DE REFERÊNCIAS, PARA QUE AS INTERMEDIÁRIAS SEJAM DEVOLVIDAS À MEMÓRIA.

ESTE CONTRÔLE É FEITO ATRAVÉS DA SUBROTINA STR QUE PEGA O VALOR DA CADEIA DE REFERÊNCIA ATUAL NO TOPO DA PILHA DE PARÂMETROS.

O PROCEDIMENTO DA ROTINA QUE AVALIA A CADEIA DE REFERÊNCIA É SIMPLESMENTE O DE CHAMAR A SUBROTINA QUE COMPILA UMA EXPRESSÃO PARA QUE SEJA COMPILADA A MENOR EXPRESSÃO POSSÍVEL. AO DEVOLVER O CONTRÔLE, SABEMOS QUE FOI GERADO CÓDIGO PARA QUE O RESULTADO DA EXPRESSÃO FIQUE NO TOPO DA PILHA DE PARÂMETROS. NESTE PONTO É GERADO O CÓDIGO PARA A CHAMADA DA SUBROTINA STR QUE FARÁ O CONTRÔLE (NA FASE EXECUÇÃO) DA CADEIA DE REFERÊNCIA.

APÓS A AVALIAÇÃO DA CADEIA DE REFERÊNCIA, ESTA ROTINA TESTA O PRÓXIMO CARACTER. SE FOR UM '/' DE TRANSFERÊNCIA, TRANSFERE O CONTRÔLE PARA A ROTINA QUE AVALIA ESTE CAMPO. SE FOR UM '=' TRANSFERE PARA SUBSTITUIÇÃO. SE FOR INDICADOR DE FINAL DE CARTÃO (≠), LIGA UM INDICADOR PARA QUE, SE HOUVER CONTINUAÇÃO, SEJA DADO O CONTRÔLE PARA O INÍCIO DO PADRÃO E TRANSFERE O CONTRÔLE PARA A EDIÇÃO DO PRÓXIMO CARTÃO. SE NÃO FOR NENHUM DOS CARACTERES ANTERIORES, TRANSFERE CONTRÔLE PARA A ROTINA QUE AVALIA O PADRÃO.

2.4.6- PADRÃO:

A COMPILAÇÃO DA PARTE DO PADRÃO FOI MUITO SIMPLIFICADA COM A SUBROTINA QUE AVALIA UMA EXPRESSÃO.

ESTA SUBROTINA É CHAMADA NOS SEGUINTES CAMPOS:

1) NOME DE UM ELEMENTO ARBITRÁRIO OU BALANCEADO.
2) TAMANHO DO MESMO.
3) NOME OU CÁLCULO DE UM ELEMENTO CONSTANTE.

DESTE MODO, A ROTINA DE ANALISE DO PADRÃO APENAS MONTA O DESCRITOR DO PADRÃO (DP) COM AS INFORMAÇÕES NECESSÁRIAS PARA A VARREDURA. ESTAS INFORMAÇÕES COMO FOI DITO ANTERIORMENTE, SÃO: TIPO, ENDEREÇO E TAMANHO DO ELEMENTO.

O DP E MONTADO EM LISTA DE APONTADORES. AO INICIAR A ROTINA PEDIMOS O PRIMEIRO NÓ DA LISTA E GUARDAMOS SEU VALOR PARA QUE MAIS ADIANTE SEJA GERADO O CÓDIGO OBJETO COM O MESMO.

SE UMA DECLARAÇÃO FOI INTERROMPIDA NO FINAL DA CADEIA DE REFERÊNCIA, A CONTINUAÇÃO DEVE TRANSFERIR O CONTRÔLE PARA O INÍCIO DESTA PARTE, POIS É NECESSÁRIO AINDA ALOCAR O PRIMEIRO NÓ. SE AO CONTRÁRIO, FOI INTERROMPIDA NO MEIO DO PADRÃO (ENTRE UM ELEMENTO E OUTRO), A ENTRADA DEVE SER PELA PARTE QUE ALOCA UM NOVO NÓ E LIGA-O COM O ANTERIOR.

AO SER DETETADO UM ELEMENTO SEM NOME (EX. **, */'5'*, *()* OU *()/'5'*), É ALOCADA UMA VARIAVEL QUE NÃO TERÁ ENTRADA NA TS, MAS TERÁ SEU ENDEREÇO NO RESPECTIVO CAMPO DO DP. ASSIM SE HOUVER ALGUM VALOR A SER COLOCADO NESTA VARIÁVEL, ESTE VALOR É ALOCADO SEM QUE HAJA NECESSIDADE DE TRATAMENTO ESPECIAL PELA ROTINA DE VARREDURA. CONCORDAMOS QUE DESTA MANEIRA, DESPERDICAMOS MEMÓRIA AO DAR VALOR A UMA VARIÁVEL QUE NÃO SERÁ REFERENCIADA, MAS SENDO ESTE UM TIPO DE ELEMENTO POUCO USADO, OPTAMOS PELA SOLUÇÃO DE SIMPLIFICARMOS A PROGRAMACAO E ECONOMIZAR MEMÓRIA NA SUBROTINA DE VARREDURA.

AO SER AVALIADA A EXPRESSÃO DO NOME DE UM ELEMENTO, É GERADO CÓDIGO PARA RETIRA-LO DO TOPO DA PILHA DE PARÂMETROS E COLOCA-LO EM SEU LUGAR NO DP. AO FAZERMOS O

MESMO PARA O TAMANHO DE UM ELEMENTO, É GERADO CÓDIGO, CHAMANDO A SUBROTINA NUM QUE CONVERTE A CADEIA DO TOPO DA PILHA EM NUMÉRICA E DEIXA O RESULTADO NO ACUMULADOR. ENTAO E GERADO CODIGO PARA GUARDAR ESTE VALOR NO SEU LUGAR NO DP.

EXEMPLO: ELEMENTO: *$(A B + C)/(J + '7')*

CODIGO GERADO:

```
STACK   A
STACK   B
STACK   C
SOMA
B    L  FALHA
CONC    2
IND
TIRA
STO  L  DP(END)
STACK   J
STACK   '7'
SOMA
B    L  FALHA
NUM
STO  L  DP(TAM)
```

DP(TAM) E DP(END) SÃO OS LOCAIS DO TAMANHO E DO ENDEREÇO DESTE ELEMENTO NO DP.

ADIANTAMOS QUE A SUBROTINA TIRA COLOCA NO ACUMULADOR O VALOR DO TOPO DA PILHA DE PARÂMETROS.

SE UM ELEMENTO FOR DE TAMANHO INDEFINIDO, COLOCAMOS ' Λ ' NO LUGAR DO TAMANHO. SE FOR ELEMENTO CONSTANTE, ESTE CAMPO E PREENCHIDO NA FASE EXECUÇÃO.

APOS A COMPILAÇÃO DE CADA ELEMENTO, TESTAMOS O FINAL DO CAMPO PELOS CARACTERES '≠', '=' OU '/'. SE ENCONTRARMOS '=' OU '/', TERMINAMOS A COMPILAÇÃO DO PADRÃO GERANDO O CÓDIGO PARA A CHAMADA DA SUBROTINA DE VARREDURA E FECHANDO O ÚLTIMO NO DO DP. ENTÃO O CONTRÔLE É TRANSFERIDO PARA O CAMPO DE SUBSTITUIÇÃO OU DE TRANSFERÊNCIA CONFORME O CASO. SE FOR DETETADO O FINAL DO REGISTRO, LIGAMOS UM INDICADOR PARA INDICAR ONDE FOI TERMINADA A COMPILAÇÃO E TRANSFERIMOS O CONTRÔLE PARA A EDIÇÃO DO PRÓXIMO CARTÃO.

SE O FINAL DO PADRÃO NÃO FOR ENCONTRADO NESTE PONTO, PEDIMOS UM NOVO NÓ, LIGAMO-O COM O ANTERIOR E COMPILAMOS O PRÓXIMO ELEMENTO.

2.4.7- SUBSTITUIÇÃO:

O PRIMEIRO PROCEDIMENTO DA ROTINA DE SUBSTITUIÇÃO É TESTAR UM INDICADOR PARA SABER SE É PERMITIDO SUBSTITUIÇÃO NA DECLARAÇÃO (VER PARAG.2.3.5, CADEIA DE REFERÊNCIA).

APÓS ESTE TESTE, TESTAMOS PRIMEIRO O FIM DO CAMPO DE SUBSTITUIÇÃO E SE ESTE FOR ENCONTRADO, GERAMOS CÓDIGO PARA RECALCAR NA PILHA DE PARÂMETROS UMA CADEIA NULA.

ABRO AQUI UM PARENTESES PARA DAR UMA BREVE EXPLICAÇÃO DE COMO FUNCIONA A SUBROTINA ASSIG QUE FARÁ A SUBSTITUIÇÃO NA CADEIA DE REFERÊNCIA NA FASE EXECUÇÃO.

EXISTE NA ÁREA DE COMUNICAÇÃO, 2 PONTEIROS QUE APONTAM PARA A PARTE DA CADEIA DE REFERÊNCIA QUE CASOU COM O PADRÃO. SE A DECLARAÇÃO NÃO TIVER PADRÃO ESTES PONTEIROS SÃO ZERADOS PARA INDICAR QUE A SUBSTITUIÇÃO DEVE SER FEITA EM TODA A CADEIA DE REFERÊNCIA.

AO SER CHAMADA, A SUBROTINA ASSIG BUSCA NO TOPO DA PILHA DE PARÂMETROS A CADEIA QUE VAI SUBSTITUIR NA DE REFERÊNCIA A PARTE QUE CASOU COM O PADRÃO.

VOLTANDO A COMPILAÇÃO DE SUBSTITUIÇÃO, LEMBRAMOS QUE QUANDO NÃO HÁ PARTE DE SUBSTITUIÇÃO APÓS O SINAL DE IGUAL, É DESEJADO QUE SEJA DELETADA A PARTE DA CADEIA DE REFERÊNCIA. ESTE É O MOTIVO PELO QUAL RECALCAMOS UMA CADEIA NULA.

NO FUNDO, A PARTE QUE COMPILA SUBSTITUIÇÃO, NÃO PASSA DE UMA CHAMADA À SUBROTINA QUE COMPILA UMA EXPRESSÃO, LIGANDO O INDICADOR CONVENIENTE. APENAS O CONTRÔLE DE CARTÕES DE CONTINUAÇÃO MERECE DESTAQUE ESPECIAL.

JÁ NOS REFERIMOS ANTERIORMENTE (PARAG. 2.4.3) QUE CADA CARTÃO DE CONTINUAÇÃO É COMPILADO EM SEPARADO. A EXPRESSÃO É CALCULADA, FICANDO SEU RESULTADO NO TOPO DA PILHA. AO DETETARMOS OUTRO CARTAO DE CONTINUAÇÃO, A EXPRESSÃO QUE CONTÉM TAMBÉM É CALCULADA FICANDO O RESULTADO NO TOPO. AO DETETARMOS UM CAMPO DE TRANSFERÊNCIA OU UM CARTÃO QUE NÃO SEJA DE CONTINUAÇÃO, É GERADO O CÓDIGO OBJETO PARA CONCATENAR AS EXPRESSÕES QUE JÁ FORAM RECALCADAS.

POR EXEMPLO:

```
   A  =  B  +  C  .....        1
. J  +  A  $( .....            2
. $(  .....                    3
. (  ......    ...) /(ROT)     4
```

A EXPRESSÃO 1 É CALCULADA FICANDO SEU RESULTADO NO TOPO DA PILHA. O MESMO ACONTECE COM AS EXPRESSOES 2, 3, E 4. AO DETETARMOS O FINAL DO CAMPO DE SUBSTITUIÇÃO, GERAMOS O CÓDIGO CONC 4 PARA CONCATENAR AS 4 EXPRESSÕES. SÓ ENTÃO É GERADA A CHAMADA PARA A SUBROTINA ASSIG.

QUANDO O CONTRÔLE É DEVOLVIDO DA SUBROTINA QUE COMPILA UMA EXPRESSÃO, TESTAMOS SE A COMPILAÇÃO TERMINOU NUM CAMPO DE TRANSFERÊNCIA. EM CASO AFIRMATIVO, GERAMOS CÓDIGO OBJETO PARA PERFAZER A SUBSTITUIÇÃO E TRANSFERIMOS O

CONTRÔLE PARA A PARTE QUE COMPILA TRANSFERÊNCIA.

SE A COMPILAÇÃO DA EXPRESSÃO TERMINOU NO FINAL DO CARTÃO, INCREMENTAMOS O CONTADOR DE CARTÕES DE CONTINUAÇÃO, LIGAMOS UM INDICADOR PARA ASSINALAR O PONTO ONDE FOI PARADA A COMPILAÇÃO E TRANSFERIMOS O CONTRÔLE PARA A EDIÇÃO DO PRÓXIMO CARTÃO.

2.4.8- TRANSFERÊNCIA:

ANTES DE ENTRARMOS NA COMPILAÇÃO DO CAMPO DE TRANSFERÊNCIA PROPRIAMENTE DITO, DAREMOS UMA BREVE EXPLICAÇÃO DO CONTRÔLE LÓGICO DE UMA DECLARAÇÃO.

COMO JÁ DISSEMOS, A COMPILAÇÃO DE UMA DECLARAÇÃO TEM UMA PARTE QUE PROCESSA SUCASSO E UMA QUE PROCESSA FALHA.

A PARTE QUE PROCESSA FALHA REALIZA AS SEGUINTES FUNÇÕES: SE HÁ POSSIBILIDADE DE FALHA, GERA UMA CHAMADA PARA A SUBROTINA ESVAZ E APONTA TODOS OS DESVIOS DE FALHA PARA ELA. SE HOUVER TRANSFERÊNCIA EM FALHA, CALCULA-A E GERA O DESVIO NECESSÁRIO.

A PARTE QUE COMPILA SUCESSO TEM POR FINALIDADE GERAR CÓDIGO PARA SAIDA DE DADOS (SE NECESSÁRIO) E PARA PROCESSAMENTO DE TRANSFERÊNCIA EM CASO DE SUCESSO.

PARA MELHOR VISUALIZACAO, DAMOS A SEGUIR OS CASOS POSSÍVEIS.

1) HÁ TRANSFÊRÊNCIA INCONDICIONAL:

EX: A *SYSPOT/'5'* *SYSPPT* = /($(J '2'))

PROGRAMA OBJETO:

```
        ---   --
        ---   --
        ---   --
        B     L   FALHA
        ---   --
        ---   --
        ---   --
        SPOT      }          PARTE SUCESSO
        SPPT      }
FALHA   ESVAZ     }          PARTE FALHA
        STACK     J    )
        STACK     '2'  )
        CONC      2    )     CALCULO DO
        LIND           )     CAMPO DE
        TIRA           )     TRANSFERENCIA
```

```
                    STO     *+1
                    B   L   *-*     }   TRANSFERÊNCIA
```

NOTA - A SUBROTINA LIND E SEMELHANTE A IND, DIFERINDO NO FATO DE LIND PROCURAR UM RÓTULO ENQUANTO QUE IND PROCURA UMA VARIÁVEL.

2) NÃO HÁ TRANSFERÊNCIA:

EX: A *SYSPOT/'7'* *SYSPPT* =

CÓDIGO OBJETO

```
                    ---  --
                    ---  --
                    ---  --
                    B    L  FALHA
                    ---  --
                    ---  --
                    ---  --
                    ---  --
                    SPOT      }       PARTE SUCESSO
                    SPPT      }
              FALHA ESVAZ     }       PARTE FALHA
                    ---  ---          PROX.DECLARAÇÃO
                    ---  --
                    ---  --
```

3) HÁ APENAS TRANSFERÊNCIA PARA A FALHA:

EX: A *SYSPOT* K *SYSPPT* = /F($(J + '5'))

CÓDIGO OBJETO:

```
                    ---  --
                    ---  --
                    ---  --
                    B    L  FALHA
                    ---  --
                    ---  --
                    ---  --
                    SPOT            }
                    SPPT            }   PARTE SUCESSO
                    B    L  PROX    }
              FALHA ESVAZ             }
                    STACK  J          }
                    STACK  '5'        }
                    SOMA              }
                    B    L  ERRO      }
```

```
                B        *+3      }  PARTE FALHA
ERRO            ERR      34       }
                LIND              }
                TIRA              }
                STO      *+1      }
                B    L   *-*      }
PROX            ---  ---          PRÓXIMA DECLAR.
                ---  --
                ---  --
```

NOTA - COMO É ERRO DE PROGRAMAÇÃO FALHA EM UM CAMPO DE TRANSFERÊNCIA, CHAMAMOS A SUBROTINA ERR QUE IMPRIME A MENSAGEM DE ERRO.

4) HÁ APENAS TRANSFERÊNCIA PARA SUCESSO:

EX: A *SYSPOT/'5'* *SYSPPT* = /S(ROT)

CÓDIGO OBJETO:

```
                ---  --
                ---  --
                ---  --
                B    L   FALHA
                ---  --
                ---  --
                ---  --
                SPOT             }
                SPPT             }  PARTE SUCESSO
                B    L   ROT     }
FALHA           ESVAZ            }  PARTE FALHA
                ---  ---             PRÓX.DECLARAÇÃO
                ---  --
                ---  --
```

5) HÁ 2 CONDICOES DE TRANSFERÊNCIA:

A) SUCESSO APARECE PRIMEIRO:

EX: A *SYSPOT/'5'* = /S($(E '1'))F($(E + '2'))

CÓDIGO OBJETO:

```
                ---  --
                ---  --
                ---  --
                B    L   FALHA
                ---  --
                ---  --
                ---  --
```

```
                SPOT
                STACK    E
                STACK    '1'
                CONC     2          } PARTE SUCESSO
                LIND
                TIRA
                STO      *+1
                B     L  *-*
          FALHA ESVAZ
                STACK    E
                STACK    '2'
                SOMA
                B     L  ERRO       } PARTE FALHA
                B        *+3
          ERRO  ERR      34
                LIND
                TIRA
                STO      *+1
                B     L  *-*
                ---   ---          PRÓX.DECLARAÇÃO
                ---   --
                ---   --
```

B) FALHA APARECE PRIMEIRO:

EX: A *SYSPOT/'5'* = /F($(A + B C))S(ROT)

CÓDIGO OBJETO:

```
                ---   --
                ---   --
                ---   --
                B     L  FALHA
                ---   --
                ---   --
                ---   --
                B     L  SUCSS
          FALHA ESVAZ
                STACK    A
                STACK    B
                SOMA
                B     L  ERRO
                STACK    C          } PARTE FALHA
                CONC     2
                B        *+3
          ERRO  ERR      34
                LIND
                TIRA
                STO      *+1
                B     L  *-*
          SUCSS SPOT
                B     L  ROT        } PARTE SUCESSO
```

PELOS EXEMPLOS ACIMA PODEMOS NOTAR QUE SÓ CHAMAMOS A SUBROTINA PARA AVALIAR UMA EXPRESSÃO QUANDO O PRIMEIRO CARACTER DO CAMPO, APOS '(', FOR '$'. QUANDO ISTO OCORRE, SABEMOS QUE A ÚLTIMA INSTRUÇÃO GERADA PELA REFERIDA SUBROTINA FOI UMA CHAMADA PARA IND. O PROCEDIMENTO TOMADO NESTES CASOS E SUBSTITUIR A CHAMADA DE IND PELA DE LIND.

O CONTRÔLE DO FLUXO DO PROGRAMA OBJETO É FEITO POR INDICADORES: HÁ UM PARA CADA CONDIÇÃO. HÁ TAMBÉM INDICADORES QUE VERIFICAM SE UMA CONDIÇÃO DE TRANSFERÊNCIA FOI DUPLICADA.

PARA MAIORES DETALHES, VER O DIAGRAMA DE BLOCOS NO APÊNDICE I.

2.5- ENTRADA E SAIDA:

NESTE PARAGRAFO, APENAS FAREMOS UMA COLETÂNEA DO QUE JÁ FOI DITO ATÉ AGORA SOBRE O ASSUNTO.

A ENTRADA E SAIDA SERÁ RESTRINGIDA À IMPRESSORA, LEITORA DE CARTÕES E PERFURADORA DE CARTÕES. AS SUBROTINAS QUE EXECUTAM ESTAS OPERAÇÕES SERÃO CHAMADAS PARA A MEMÓRIA DURANTE A EXECUÇÃO INDEPENDENTEDO SEU USO.

AO SER INICIALIZADA A TS, SÃO INSERIDAS AS VARIAVEIS SYSPIT, SYSPOT E SYSPPT. NO LUGAR DO TIPO DO ELEMENTO, É COLOCADO UM CÓDIGO DE TIPO ESPECIAL PARA CADA UMA DELAS PARA QUE SE POSSA RECONHECE-LAS. POR EXEMPLO: RÓTULO, CONSTANTE E VARIÁVEL SAO TIPOS 1, 2 E 3. SYSPIT, SYSPOT E SYSPPT SAO TIPOS 8, 5, E 9 RESPECTIVAMENTE.

AO SER DETETADO UM TIPO 8, A SUBROTINA PROC GERA CÓDIGO PARA CHAMAR A SUBROTINA QUE LE CARTÃO.

AO SER DETETADO UM TIPO 5 OU 9, A SUBROTINA PROC LIGA UM DOS INDICADORES DOUT5 OU DOUT9 CONFORME O TIPO DETETADO. AO SER PROCESSADA PARTE DE SUCESSO DA DECLARAÇÃO, ESTES INDICADORES SÃO TESTADOS E SE ALGUM ESTIVER ACESO, É GERADO O CÓDIGO PARA CHAMAR A SUBROTINA QUE IMPRIME SYSPOT OU PERFURA SYSPPT. LOGO A SEGUIR ESTES INDICADORES SÃO DESLIGADOS PARA QUE NA PARTE QUE TERMINA A DECLARAÇÃO ANTERIOR NÃO SEJA GERADO O CÓDIGO NOVAMENTE.

2.6- VERIFICAÇÃO DA TABELA DE SÍMBOLOS:

JA NOS REFERIMOS ANTERIORMENTE COMO É VERIFICADA A TS PARA DETETAR TRANSFERÊNCIA PARA RÓTULOS INEXISTENTES.

APÓS A COMPILAÇÃO, TOMA CONTA DA MEMÓRIA, UMA FASE INTERMEDIÁRIA ANTES DA EXECUÇÃO. ESTE PROGRAMA, COMO NÃO DEPENDE DE NENHUMA SUBROTINA NEM DADO ANTERIOR (SOMENTE DA TS COM O VETOR DOS INICIOS) FICA SOZINHO NA MEMÓRIA E TESTA SE HÁ ALGUM ENDEREÇO NEGATIVO. EM CASO AFIRMATIVO, IMPRIME MENSAGEM E NÃO CHAMA A FASE EXECUÇÃO.

APÓS A VERIFICAÇÃO, TESTA SE HOUVE ERRO QUE INIBA A EXECUÇÃO E EM CASO AFIRMATIVO DEVOLVE CONTRÔLE AO SUPERVISOR.

SE TUDO ESTIVER CERTO, CHAMA A FASE EXECUÇÃO.

PARTE III

FASE EXECUÇÃO

3.1 - ESTRUTURA GERAL:

A FASE EXECUÇÃO SE COMPÕE DE UM PROGRAMA INICIALIZADOR E AS SUBROTINAS QUE REALIZARÃO AS FUNÇÕES NECESSÁRIAS PARA EXECUÇÃO DO PROGRAMA OBJETO.

ELA É CHAMADA ATRAVÉS DO PROGRAMA QUE VERIFICA A TS E SÓ VEM PARA A MEMÓRIA SE A COMPILAÇÃO NÃO TIVER ERROS.

AS SUBROTINAS QUE FAZEM PARTE DESTA FASE PODEM SER CLASSIFICADAS EM DOIS TIPOS: SUBROTINAS AUXILIARES E SUBROTINAS CHAMADAS PELO PROGRAMA OBJETO.

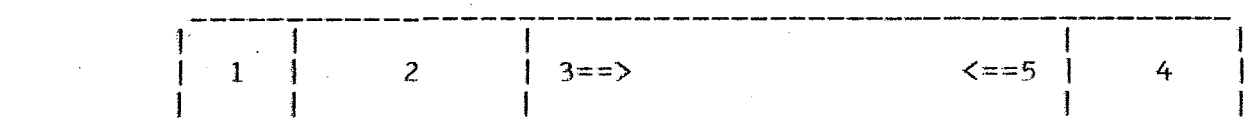
AS SUBROTINAS AUXILIARES SERVEM APENAS DE SUPORTE PARA AS OUTRAS E POR ISTO NÃO SERÃO TÃO DETALHADAS NESTE TRABALHO.

3.1.1 - ALOCAÇÃO DA MEMÓRIA:

ESTE ASSUNTO JÁ FOI TRATADO NA PRIMEIRA PARTE DESTE TRABALHO E AQUI DAMOS APENAS UM RESUMO DO QUE FOI DITO PARA MELHOR EXPLICAR O PROGRAMA INICIALIZADOR DA FASE EXECUÇÃO.

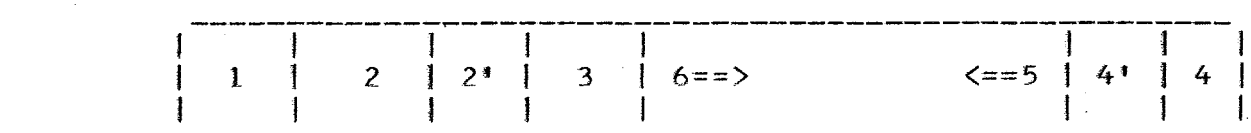
A DISPOSIÇÃO DA MEMÓRIA NAS DUAS FASES PODE SER ESQUEMATIZADA DA SEGUINTE MANEIRA:

FASE COMPILAÇÃO:



SENDO: 1 - ÁREA DO SISTEMA MONITOR.
2 - COMPILADOR PROPRIAMENTE DITO.
3 - PROGRAMA OBJETO CRESCENDO NO SENTIDO DE SETA.
4 - ÁREA DE COMUNICAÇÃO.
5 - TS, ALOCAÇÃO DE VARIÁVEIS, CONSTANTES, ETC, CRESCENDO NO SENTIDO DA SETA.

FASE EXECUÇÃO:



SENDO: 1 - ÁREA DO SISTEMA MONITOR.
2 - SUBROTINAS DA FASE EXECUÇÃO (QUE SÃO MENORES

QUE O COMPILADOR).
2'- ÁREA LIVRE.
3 - PROGRAMA OBJETO.
4 - ÁREA DE COMUNICAÇÃO DA FASE EXECUÇÃO (QUE É MENOR QUE A DA COMPILAÇÃO).
4'- ÁREA LIVRE.
5 - TS, ALOCAÇÃO DE VARIÁVEIS, CONSTANTES, ETC... CRESCENDO NO SENTIDO DA SETA.
6 - PILHA DE RECURSIVIDADE CRESCENDO NO SENTIDO DA SETA

3.1.2 - PROGRAMA INICIALIZADOR:

COMO VIMOS, AO SER CARREGADA A FASE EXECUÇÃO, FICAM DUAS AREAS LIVRES DEVIDO AO FATO DAS SUBROTINAS DE EXECUÇÃO E A ÁREA DE COMUNICAÇÃO SEREM MENORES QUE NA FASE COMPILAÇÃO.

O PROGRAMA INICIALIZADOR TEM COMO UMA DE SUAS FUNÇÕES A DE DEVOLVER ESTAS ÁREAS LIVRES A MEMÓRIA DISPONÍVEL. ISTO É FEITO DIVIDINDO A ÁREA EM NÓS DE 4 PALAVRAS E DEVOLVENDO-OS UM A UM PARA A LISTA DE VAZIOS.

A OUTRA FUNÇÃO DO PROGRAMA INICIALIZADOR É DAR VALOR AO TOPO E BASE DA PILHA DE PARÂMETROS.

FEITO ISTO, O PROGRAMA DESVIA O CONTROLE PARA O INÍCIO DO PROGRAMA OBJETO. O ENDEREÇO DE EXECUÇÃO ESTÁ NA ÁREA DE COMUNICAÇÃO.

3.1.3 - VARIÁVEIS TEMPORÁRIAS:

COMO DISSEMOS ANTERIORMENTE, O CONTRÔLE DAS VARIÁVEIS TEMPORÁRIAS É REALIZADO PELAS SUBROTINAS DESTA FASE. CONVENCIONAMOS LIGAR O BIT ZERO DO ENDEREÇO DE UMA VARIÁVEL QUANDO ESTA FOR TEMPORÁRIA.

AS VARIÁVEIS TEMPORÁRIAS SÃO CRIADAS PARA ARMAZENAR O RESULTADO DE UMA CONCATENAÇÃO OU DE UMA OPERAÇÃO ARITMÉTICA.

SEMPRE QUE UMA SUBROTINA QUALQUER USA UMA VARIÁVEL INTERMEDIÁRIA, APÓS FAZE-LO, DEVOLVE-A A MEMÓRIA DISPONÍVEL. O ÚNICO CASO QUE FOGE A ESTA REGRA E A SUBROTINA DE CONCATENAÇÃO QUE, AO DETETAR UMA VARIAVEL INTERMEDIÁRIA, AO INVÉS DE DEVOLVE-LA A MEMÓRIA, MODIFICA SEUS PONTEIROS JUNTANDO-A COM AS OUTRAS CADEIAS, EVITANDO ASSIM CÓPIA DESNECESSÁRIA DE UMA CADEIA.

3.1.4 - PONTEIROS:

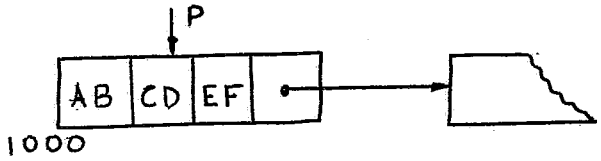
ENTENDEMOS COMO PONTEIRO O ENDEREÇO DE UM CARACTER. COMO OS CARACTERES SÃO GUARDADOS EM DOIS POR PALAVRA, TEMOS A NECESSIDADE DE INDICARMOS QUAL DOS CARACTERES DE DETERMINADA PALAVRA DESEJAMOS.

ISTO PODERIA SER FEITO LIGANDO-SE O BIT ZERO DO PONTEIRO QUE ENDEREÇA O CARACTER. PODERIAMOS CONVENCIONAR QUE, SE O BIT ZERO ESTIVESSE LIGADO, O PONTEIRO REFERIRIA-SE AO SEGUNDO CARACTER DA PALAVRA.

MAS ESTE METÓDO CRIARIA DIFICULDADES PARA SE IDENTIFICAR O FINAL DE UM NÓ, POIS AO AVANCARMOS UM PONTEIRO NÃO SERIA FÁCIL SABER SE ELE ENDEREÇARIA O APONTADOR DO NÓ.

POR ESTE MOTIVO OPTAMOS PELA REPRESENTAÇÃO DE UM PONTEIRO EM DUAS PALAVRAS. A PRIMEIRA DA O ENDEREÇO DO NÓ EM QUE O CARACTER ESTÁ CONTIDO E A SEGUNDA NOS DA A ORDEM DO CARACTER DENTRO DO NÓ.

POR EXEMPLO:



O PONTEIRO P E CONSTITUIDO DE DUAS PALAVRAS:
1 - 1000 (ENDERECO DO NÓ).
2 - 4 (ORDEM DO CARACTER NO NÓ).

3.1.5 - SUBROTINA PARA DEPURAÇÃO:

PARA FACILITAR A DEPURAÇÃO DO COMPILADOR E DOS PROGRAMAS DE PROGRAMADORES COM POUCA PRÁTICA, IMPLANTAMOS UMA SUBROTINA QUE PERMITE SEGUIR A SEQUÊNCIA DE EXECUÇÃO DE UM PROGRAMA SNOBOL. ESTA SUBROTINA IMPRIME O TAMANHO E CONTEUDO DA CADEIA DE REFERÊNCIA SEMPRE QUE REQUISITADO PELO OPERADOR. A IMPRESSÃO E CONTROLADA POR UMA CHAVE NO CONSOLE (COMO EM FORTRAN), APENAS SENDO DESNECESSÁRIO CARTÕES DE CONTRÔLE PARA A MESMA ESPECIFICAÇÃO.

3.2 - SUBROTINAS AUXILIARES:

3.2.1 - CONTROLE DE MEMÓRIA:

3.2.1.1 - AVAIR, AVAIL & SYMTB:

ESTAS 3 SUBROTINAS FAZEM O CONTRÔLE DA MEMÓRIA DISPONÍVEL PARA AS VARIÁVEIS.

A - FUNÇÃO: AVAIL SEMPRE QUE CHAMADA, FORNECE UM NÓ DE QUATRO PALAVRAS. SYMTB FAZ COM QUE ESTE NÓ SEJA DE TAMANHO QUALQUER. AVAIR SERVE PARA DEVOLVER UM NÓ À MEMÓRIA.

B - USO: A SUBROTINA SYMTB É USADA PARA ALOCARMOS ENTRADAS NA TS ONDE OS NÓS SÃO DE TAMANHO VARIÁVEL. AVAIL E USADA PARA ALOCARMOS VARIÁVEIS, DP'S, ETC.

C - FUNCIONAMENTO: A SUBROTINA AVAIL MANTEM UMA LISTA DOS NÓS QUE FORAM DEVOLVIDOS. TODA VEZ QUE PEDIMOS UM NÓ, ESTA LISTA É CONSULTADA. SE HOUVER ALGUM NÓ NELA, ESTE NÓ É DEVOLVIDO, CASO CONTRÁRIO, É FORNECIDO UM NÓ DO ESPAÇO LIVRE CONTÍGUO. NESTE ÚLTIMO CASO, TORNA-SE NECESSÁRIO TESTAR SE O ESPAÇO LIVRE CONTÍGUO NÃO ACABOU.

O PROCEDIMENTO DA SUBROTINA AVAIR É O DE INSERIR O NÓ DEVOLVIDO NA LISTA DE VAZIOS.

QUANDO CHAMAMOS SYMTB, É FORNECIDO UM NÓ DO ESPAÇO LIVRE CONTÍGUO, POIS A LISTA DE VAZIOS CONTEM NÓS DE QUATRO PALAVRAS SOMENTE.

NA VERDADE, AS 3 SUBROTINAS SAO 3 ENTRADAS DE UMA MESMA SUBROTINA.

3.2.1.2 - DEVOL:

A - FUNÇÃO: DEVOLVER A MEMÓRIA LIVRE UMA CADEIA DADA.

B - USO: E USADA PELA SUBROTINA ESVAZ E POR OUTRAS SUBROTINAS QUE APÓS USAREM VARIÁVEIS TEMPORÁRIAS, AS ENTREGAM A MEMÓRIA LIVRE.

C - FUNCIONAMENTO: A SUBROTINA CONSISTE DE UMA SERIE DE CHAMADAS A SUBROTINA AVAIR, DEVOLVENDO A CADEIA NÓ A NÓ. O PROCESSO TERMINA QUANDO E DETETADO O ÚLTIMO APONTADOR DA CADEIA (Λ).

3.2.2 - MANIPULAÇÃO DE CARACTERES:

3.2.2.1 - PUT:

A - FUNÇÃO: DADO UM PONTEIRO E UM CARACTER, ESTA SUBROTINA SE ENCARREGA DE COLOCAR O CARACTER NO LOCAL DADO PELO PONTEIRO, SEM ALTERAR OS OUTROS CARACTERES DO NÓ.

B - USO: É USADA PELAS SUBROTINAS QUE MANIPULAM CARACTERES.

C - FUNCIONAMENTO: A SUBROTINA PEGA A PALAVRA DADA PELO APONTADOR E POR MEIO DE DESLOCAMENTOS ("SHIFTS") COLOCA O CARACTER NA PRIMEIRA OU SEGUNDA METADE DA PALAVRA DEPENDENDO DE SE A SEGUNDA PALAVRA DO PONTEIRO FOR IMPAR OU PAR RESPECTIVAMENTE.

3.2.2.2 - PEGA:

A - FUNÇÃO: DADO UM PONTEIRO, A SUBROTINA RETORNA O CARACTER APONTADO POR ELE.

B - USO: É USADA PELA SUBROTINA GET E OUTRAS.

C - FUNCIONAMENTO: SEMELHANTE AO DA SUBROTINA PUT. AO INVÉS DO CARACTER SER INSERIDO, ELE E RETIRADO (SEM DESTRUIR) DA CADEIA.

3.2.2.3 - GET:

A - FUNÇÃO: DADO UM PONTEIRO, A SUBROTINA RETORNA O PROXIMO CARACTER NÃO VAZIO APONTADO PELO MESMO, E AVANCA O PONTEIRO.

B - USO: É USADA PELAS SUBROTINAS QUE MANIPULAM COM CARACTERES, PRINCIPALMENTE AS SUBROTINAS PATT, ASSIG E OUTRAS.

C - FUNCIONAMENTO: A SUBROTINA PEGA O CARACTER ATRAVÉS DA SUBROTINA PEGA. FEITO ISTO, TESTA SE O CARACTER E VAZIO (V) E EM CASO AFIRMATIVO AVANCA O PONTEIRO E REINICIA A OPERAÇÃO.

QUANDO E ACHADO UM CARACTER NÃO VAZIO, A SUBROTINA AVANÇA O PONTEIRO E DEVOLVE ESTE CARACTER.

HÁ TAMBÉM UM CÓDIGO DE ERRO FORNECIDO POR ESTA SUBROTINA QUE PODE ASSUMIR TRES VALORES PARA AS SEGUINTES CONDIÇÕES:

1) SUCESSO.

2) O CARACTER FOI ACHADO, MAS AO AVANÇARMOS O PONTEIRO, CHEGAMOS AO FINAL DA CADEIA.
3) SÓ FORAM ENCONTRADOS CARACTERES VAZIOS E O PONTEIRO CHEGOU AO FINAL DA CADEIA.

3.2.3 - MANIPULAÇÃO DE PONTEIROS:

3.2.3.1 - LINK:

A - FUNÇÃO: DADO UM PONTEIRO, ESTA SUBROTINA MODIFICA-O PARA QUE APONTE PARA O PRÓXIMO NÓ DA CADEIA.
B - USO: PELAS SUBROTINAS QUE MANIPULAM PONTEIROS COMO AVAN.
C - FUNCIONAMENTO: A SUBROTINA PEGA A QUARTA PALAVRA DO NÓ (CAMPO DO APONTADOR) E COLOCA NA PRIMEIRA PALAVRA DO PONTEIRO.
ELA TEM DOIS ENDEREÇOS DE RETORNO QUE SÃO COMANDADOS PELO FATO DE EXISTIR OU NÃO PRÓXIMO NÓ. PARA ISTO, ELA TESTA SE O APONTADOR ACHADO É O ÚLTIMO APONTADOR DA CADEIA (Λ).

3.2.3.2 - AVAN:

A - FUNÇÃO: DADO UM PONTEIRO, A SUBROTINA AVANÇA-O PARA O PRÓXIMO CARACTER.
B - USO: É USADA PELAS SUBROTINAS QUE MANIPULAM CARACTERES.
C - FUNCIONAMENTO: O PONTEIRO É AVANCADO POR SUA SEGUNDA PALAVRA. SE ELA ULTRAPASSAR O VALOR DE 6, AUTOMATICAMENTE É ACHADO O PRÓXIMO NÓ, ATRAVÉS DE LINK, E O PONTEIRO PASSA A APONTAR PARA SEU PRIMEIRO CARACTER.
DA MESMA MANEIRA QUE LINK, TEM DOIS ENDEREÇOS DE RETÔRNO REFLETINDO A CONDIÇÃO DE NÃO SER POSSÍVEL AVANÇAR O PONTEIRO, POIS A CADEIA TERMINOU.

3.2.4 - MANIPULAÇÃO DE CADEIAS:

3.2.4.1 - ALOC:

A - FUNÇÃO: DADO O ENDEREÇO DE UMA VARIÁVEL, O ENDEREÇO DE UMA ÁREA E O TAMANHO DA ÁREA, ESTA SUBROTINA SE ENCARREGA DE ALOCAR A ÁREA NUMA CADEIA INICIANDO NO ENDEREÇO DADO. O CAMPO DO TAMANHO TAMBÉM É PREENCHIDO.

B - USO: ESTA SUBROTINA É USADA PARA ALOCAR NA VARIAVEL SYSPIT O CONTEUDO DE UM CARTÃO QUE FOI LIDO EM UMA ÁREA CONTÍGUA.

C - FUNCIONAMENTO: A SUBROTINA SE ENCARREGA DE DEVOLVER A MEMÓRIA LIVRE O ANTIGO CONTEÚDO DA CADEIA ANTES DE ALOCAR O NOVO VALOR. A ALOCAÇÃO É FEITA PEDINDO A AVAIL TANTOS NÓS QUANTOS SEJAM NECESSARIOS PARA ALOCAR A CADEIA. SE O ÚLTIMO NÓ NÃO FOI TOTALMENTE PREENCHIDO, A SUBROTINA ALOC SE ENCARREGA DE PREENCHE-LO COM VAZIOS.

3.2.4.2 - MONTA:

A - FUNÇÃO: SENDO DADOS UM PONTEIRO DE DETERMINADA CADEIA, O ENDEREÇO DE UMA ÁREA CONTÍGUA E O TAMANHO DA MESMA, ESTA SUBROTINA SE ENCARREGA DE MONTAR NESTA ÁREA O CONTEUDO DA CADEIA DADA PELO PONTEIRO:

B - USO: É USADA PELAS SUBROTINAS DE SAIDA SPOT E SPPT.

C - FUNCIONAMENTO: OS CARACTERES SÃO APANHADOS DA CADEIA PELA SUBROTINA GET QUE AVANÇA TAMBÉM O PONTEIRO. SE A CADEIA FOR MENOR QUE A ÁREA, O RESTANTE DA ÁREA É PREENCHIDO COM BRANCOS. SE A CADEIA FOR MAIOR QUE A ÁREA, AO SER DEVOLVIDO O CONTRÔLE, O PONTEIRO APONTARA PARA O PRÓXIMO CARACTER QUE NÃO FOI MONTADO NA ÁREA CONTÍGUA. DESTE MODO, PARA MONTARMOS O RESTANTE DA CADEIA BASTA CHAMARMOS NOVAMENTE A SUBROTINA MONTA SEM NOS PREOCUPARMOS COM O PONTEIRO.

3.2.4.3 - COPIA:

A - FUNÇÃO: SENDO DADO DOIS PONTEIROS P1 E P2 E SENDO DADO O ENDEREÇO DE UMA OUTRA CADEIA, ESTA SUBROTINA COPIA O TRECHO ENTRE OS DOIS PONTEIROS NA NOVA CADEIA. O CAMPO DO TAMANHO DA NOVA CADEIA TAMBÉM É PREENCHIDO.

B - USO: É USADA PELAS SUBROTINAS ASSIG E PATT. SUA GRANDE UTILIDADE É DAR VALOR AOS ELEMENTOS ARBITRÁRIOS E BALANCEADOS QUE CASARAM EM UM PADRÃO.

C - FUNCIONAMENTO: A SUBROTINA SE ENCARREGA DE DEVOLVER O ANTIGO CONTEÚDO DA CADEIA DADA A MEMÓRIA DISPONÍVEL. FEITO ISTO, COPIA NA NOVA CADEIA O CONTEUDO DA CADEIA DADA A PARTIR DE P1 E PARANDO QUANDO ALCANCAR P2. SE AO AVANCAR P1, A CADEIA TERMINAR ANTES DE SE ENCONTRAR P2, E ACUSADO ERRO DO SISTEMA. SE O ULTIMO NÓ NÃO FOR TOTALMENTE PREENCHIDO, A SUBROTINA SE ENCARREGA DE PREENCHE-LO COM VAZIOS.

3.3 - SUBROTINAS CHAMADAS PELO PROGRAMA OBJETO:

3.3.1 - MANIPULAÇÃO DA PILHA DE PARÂMETROS:

SÃO TRES AS SUBROTINAS PARA ESTE FIM: STACK, TIRA E ESVAZ.

STACK E CHAMADA PARA RECALCAR UM ELEMENTO QUE É DADO COMO PARÂMETRO, ENQUANTO QUE TIRA RETIRA UM ELEMENTO DA PILHA E DEIXA-O NO ACUMULADOR.

ESVAZ E CHAMADA NA PARTE DE FALHA E ESVAZIA A PILHA DE PARÂMETROS DEVOLVENDO AS VARIÁVEIS TEMPORÁRIAS A MEMÓRIA.

3.3.2 - CADEIA DE REFERÊNCIA (STR):

A SUBROTINA STR APANHA O VALOR DO TOPO DA PILHA DE PARÂMETROS E COLOCA-O EM UMA PALAVRA NA ÁREA DE COMUNICAÇÃO.

AO RETIRAR O ELEMENTO DA PILHA O MESMO É TESTADO PARA SABERMOS SE É UMA VARIÁVEL TEMPORÁRIA. EM CASO AFIRMATIVO, O BIT ZERO É DESLIGADO E LIGADO UM INDICADOR NA PRÓPRIA SUBROTINA. AO SER CHAMADA NA PROXIMA DECLARAÇÃO, A SUBROTINA DEVOLVE A CADEIA ANTERIOR SE ESTA FOI TEMPORÁRIA.

ESTA SUBROTINA TAMBÉM ZERA OS PONTEIROS QUE MARCAM A PARTE DA CADEIA DE REFERÊNCIA QUE CASOU COM O PADRÃO (VER PARÁGRAFO 3.3.7).

3.3.3 - CONVERSÃO NUMÉRICA (NUM):

ESTA SUBROTINA CALCULA O VALOR NUMÉRICO DA CADEIA QUE ESTA NO TOPO DA PILHA DE PARÂMETROS. É CHAMADA PARA

DETERMINAR O TAMANHO DE UM ELEMENTO DO PADRÃO (POR EX: *A/(J + '7')*).

SE A CADEIA FOR INTERMEDIÁRIA ELA É DEVOLVIDA À MEMÓRIA.

SE NÃO CONTIVER SÓMENTE DÍGITOS, COM UM SINAL OPCIONAL, OU SEU VALOR ULTRAPASSAR 32.767, A SUBROTINA CHAMA A SUBROTINA DE ERRO QUE IMPRIME MENSAGEM DE ERRO E TERMINA A EXECUÇÃO DO PROGRAMA OBJETO.

3.3.4 - SUBROTINAS ARITMÉTICAS:

EM QUALQUER DAS OPERAÇÕES ARITMÉTICAS PODE HAVER FALHA. COMO JÁ DISSEMOS ANTERIORMENTE, AS SUBROTINAS ARITMÉTICAS TEM DOIS ENDEREÇOS DE RETÔRNO DEPENDENDO DO SUCESSO OU FALHA DA OPERAÇÃO.

UMA OPERAÇÃO FALHA QUANDO UM DE SEUS OPERANDOS OU O RESULTADO NÃO FOR UM INTEIRO VÁLIDO. PARA QUE SEJA UM INTEIRO VÁLIDO DEVE (1):

1) SER CONSTITUÍDA DE DÍGITOS SOMENTE, EXCETO O PRIMEIRO CARACTER QUE OPCIONALMENTE PODE SER UM SINAL.

2) TER UM VALOR ABSOLUTO MENOR QUE 10**10.

COMO A CAPACIDADE DE UMA PALAVRA NO 1130 E DE 32.767, NÃO NOS É POSSÍVEL FAZER A ARITMÉTICA USANDO AS INSTRUÇÕES DE MÁQUINA, SENDO NECESSÁRIO O USO DE SUBROTINAS QUE AUMENTEM A PRECISÃO DOS CÁLCULOS.

TEMOS ENTÃO DUAS OPCOES QUANTO AO FORMATO DOS OPERANDOS: TRABALHARMOS EM DECIMAL (UM DÍGITO DECIMAL POR PALAVRA) OU TRABALHARMOS EM INTEIRO DUPLO (BINARIO COM 32 BITS PARA REPRESENTAÇÃO).

PARA AMBOS OS MODOS DE REPRESENTACAO JÁ EXISTEM SUBROTINAS QUE REALIZAM AS OPERAÇÕES ARITMÉTICAS NECESSÁRIAS, FICANDO APENAS PARA AS SUBROTINAS DO SISTEMA SNOBOL A CONVERSÃO DAS CADEIAS.

A PRIMEIRA OPÇÃO TEM COMO VANTAGEM O FATO DE UMA CONVERSÃO UM POUCO MAIS FÁCIL, MAS EM COMPENSAÇÃO, AS SUBROTINAS QUE EXECUTAM AS OPERACOES ARITMÉTICAS SÃO MAIS EXTENSAS E MUITO MAIS LENTAS QUE AS QUE TRABALHAM EM INTEIRO DUPLO.

PELOS MOTIVOS ACIMA, DECIDIMOS REALIZAR AS OPERAÇÕES ARITMÉTICAS EM INTEIRO DUPLO.

RESTA-NOS AINDA RESSALTAR QUE O NÚMERO MÁXIMO REPRESENTÁVEL EM INTEIRO DUPLO É 1.073.741.823, QUE É MAIOR QUE 10**10, FICANDO PORTANTO ESTA RESSALVA COMO UMA EXTENSÃO DA LINGUAGEM NESTA IMPLANTAÇÃO.

DESTA FEITA, AS SUBROTINAS ARITMÉTICAS SÃO FORMADAS DE TRES FASES: CONVERSÃO DE CARACTER PARA INTEIRO DUPLO, REALIZAÇÃO DA OPERAÇÃO ARITMÉTICA EM INTEIRO DUPLO E

CONVERSÃO DO RESULTADO DE INTEIRO DUPLO PARA CARACTER.

OS OPERANDOS DAS OPERAÇÕES ARITMÉTICAS, COMO DISSEMOS ANTERIORMENTE, SÃO ACHADOS NO TOPO DA PILHA DE PARÂMETROS.

SE NA CONVERSÃO DE CARACTER PARA INTEIRO DUPLO FOR ENCONTRADO UM CARACTER INVÁLIDO, OU FOR ULTRAPASSADA A CAPACIDADE DO INTEIRO DUPLO DURANTE A OPERAÇÃO ARITMÉTICA, ENTÃO OCORRERÁ FALHA E A SUBROTINA INTERROMPE ONDE ESTÁ VOLTANDO PARA A INSTRUÇÃO SEGUINTE A CHAMADA.

SE NÃO HOUVER NENHUMA DAS CONDIÇÕES ACIMA, O RESULTADO É COLOCADO EM UMA CADEIA INTERMEDIÁRIA NO TOPO DA PILHA DE PARÂMETROS E O CONTROLE É DEVOLVIDO DUAS PALAVRAS ADIANTE DA CHAMADA PULANDO ASSIM A INSTRUÇÃO DE DESVIO QUE FOI GERADA APÓS A CHAMADA DA SUBROTINA ARITMÉTICA.

3.3.5 - SUBROTINA DE ERRO (ERR):

A SUBROTINA DE ERRO SIMPLESMENTE IMPRIME MENSAGEM PELA IMPRESSORA E TERMINA A EXECUÇÃO DO PROGRAMA OBJETO.

A GRANDE VANTAGEM DE SE USAR UMA SUBROTINA DE ERRO É A VERSATILIDADE QUE ELA PERMITE. POR EXEMPLO, DURANTE A FASE DE DEPURACAO DO SISTEMA ELA FOI SUBSTITUIDA POR UMA SUBROTINA QUE IMPRIMIA PELA IMPRESSORA A PARTE DA MEMÓRIA QUE CONTÉM O PROGRAMA OBJETO E A TS.

ATUALMENTE, AINDA NÃO IMPLANTAMOS ROTINAS QUE REARRANJAM A MEMÓRIA ("GARBAGE COLLECTION"), MAS QUANDO FOR DESEJADA TAL IMPLANTAÇÃO (VER PARÁGRAFO 4.5), ELA PODERÁ SER CHAMADA ATRAVÉS DA SUBROTINA DE ERRO, QUANDO FOR DADO COMO PARÂMETRO O CÓDIGO DE ERRO DE ESTOURO DE MEMÓRIA. ISTO FACILITA A IMPLANTAÇÃO DESTA ROTINA, EVITANDO QUE SE MODIFIQUE OUTRAS SUBROTINAS DO SISTEMA.

3.3.6 - CONCATENAÇÃO (CONC):

A SUBROTINA DE CONCATENAÇÃO TEM APENAS UM PARÂMETRO TRANSMITIDO NA CHAMADA QUE É O NUMERO DE CADEIAS A SEREM CONCATENADAS. AS CADEIAS PROPRIAMENTE DITAS SÃO BUSCADAS NO TOPO DA PILHA DE PARÂMETROS.

APÓS SER REALIZADA A CONCATENAÇÃO, O RESULTADO É COLOCADO EM UMA VARIÁVEL INTERMEDIÁRIA E RECALCADO NA PILHA DE PARÂMETROS.

COMO O RESULTADO FICARÁ EM VARIÁVEL INTERMEDIÁRIA, QUE APÓS USADA SERÁ DEVOLVIDA A MEMÓRIA, NÃO É FEITA NENHUMA

OTIMIZAÇÃO, QUANTO AO APROVEITAMENTO DE MEMÓRIA NA CADEIA RESULTANTE.

DESTA MANEIRA, AO RETIRARMOS UMA CADEIA NÃO INTERMEDIÁRIA DA PILHA DE PARÂMETROS ELA É COPIADA SEM SEREM SUPRIMIDOS OS VAZIOS QUE POR ACASO CONTENHA. O VALOR DO CAMPO DO TAMANHO DA CADEIA E SALVO E SEU LUGAR É PREENCHIDO COM VAZIOS. O MESMO E REALIZADO COM AS OUTRAS CADEIAS A CONCATENAR, SENDO QUE O TAMANHO DA CADEIA É CALCULADO PELO CAMPO DE TAMANHO DA CADEIA E A CONCATENAÇÃO É REALIZADA COM UMA MODIFICAÇÃO DE APONTADORES.

AO CONTRÁRIO DAS OUTRAS SUBROTINAS DA FASE EXECUÇÃO, AO SER DETETADA UMA CADEIA INTERMEDIÁRIA, NÃO A DEVOLVEMOS A MEMÓRIA DISPONÍVEL. O PROCEDIMENTO REALIZADO É MODIFICAR OS APONTADORES PARA CONCATENA-LA COM AS ANTERIORES (OU POSTERIORES), APROVEITANDO A PRÓPRIA CADEIA INTERMEDIÁRIA.

PARA AUMENTARMOS A VELOCIDADE DE PROCESSAMENTO AO COPIARMOS UMA CADEIA, O MOVIMENTO DE DADOS É FEITO PALAVRA POR PALAVRA, E NÃO SÃO USADAS AS SUBROTINAS GET E PUT.

NA PRIMEIRA CADEIA A SER TIRADA DA PILHA, NÃO É ALTERADO O MARCADOR DE FINAL DE CADEIA, JÁ QUE ELA SERÁ A ÚLTIMA DA CONCATENAÇÃO; MAS EM TODAS AS OUTRAS CADEIAS, ESTE CARACTER (≠) É SUBSTITUIDO POR VAZIO (V).

3.3.7 - SUBSTITUIÇÃO (ASSIG):

ESTA SUBROTINA REALIZA A SUBSTITUIÇÃO NA PARTE DA CADEIA DE REFERÊNCIA QUE CASOU COM O PADRÃO OU, SE FOR NECESSÁRIO, EM TODA A CADEIA DE REFERÊNCIA.

EXISTE, NA ÁREA DE COMUNICAÇÃO, DOIS PONTEIROS QUE INDICAM QUAL A PARTE DA CADEIA DE REFERÊNCIA QUE CASOU COM O PADRÃO. ESTES PONTEIROS SÃO ZERADOS PELA SUBROTINA STR AO SER PROCESSADA A CADEIA DE REFERÊNCIA. AO SER TERMINADA COM SUCESSO A VARREDURA, A SUBROTINA DA VARREDURA FAZ COM QUE ESTES PONTEIROS APONTEM PARA A PARTE EM QUE HOUVE SUCESSO NA CADEIA DE REFERÊNCIA.

ASSIM, A SUBROTINA DE SUBSTITUIÇÃO, PARA SABER SE HOUVE PADRÃO OU NÃO, TESTA OS PONTEIROS E SE ESTES ESTIVEREM ZERADOS INDICA QUE TODA A CADEIA DE REFERÊNCIA DEVE SER SUBSTITUIDA.

A PARTE QUE SUBSTITUIRÁ, DAQUI POR DIANTE CHAMADA DE CADEIA PARÂMETRO, É ACHADA NO TOPO DA PILHA DE PARÂMETROS. SE FOR UMA CADEIA INTERMEDIÁRIA, ELA É DEVOLVIDA A MEMÓRIA APÓS SER COPIADA. UMA CADEIA INTERMEDIÁRIA NÃO É UTILIZADA PORQUE, COMO DISSEMOS ANTERIORMENTE, ELA NÃO APRESENTA OTIMIZAÇÃO QUANTO AO APROVEITAMENTO DE MEMÓRIA. DESTE MODO, ELA É COPIADA RETIRANDO-SE OS VAZIOS SUPÉRFLUOS.

EMBORA NÃO PAREÇA, A SUBROTINA DE SUBSTITUIÇÃO É UMA DAS MAIS COMPLEXAS DESTA FASE, DEVIDO A MULTIPLICIDADE DE CONDIÇÕES QUE PODEM OCORRER E QUE DEVEM SER TRATADAS DE MANEIRA DIFERENTE.

POR EXEMPLO, PODEMOS TER:

1) SUBSTITUIÇÃO EM TODA A CADEIA DE REFERÊNCIA.
2) OS DOIS PONTEIROS APONTAM PARA O MESMO NÓ.
3) OS DOIS PONTEIROS APONTAM PARA NÓS DIFERENTES.
4) CADA UM DOS ITENS ANTERIORES TENDO UMA CADEIA NULA COMO CADEIA PARÂMETRO.

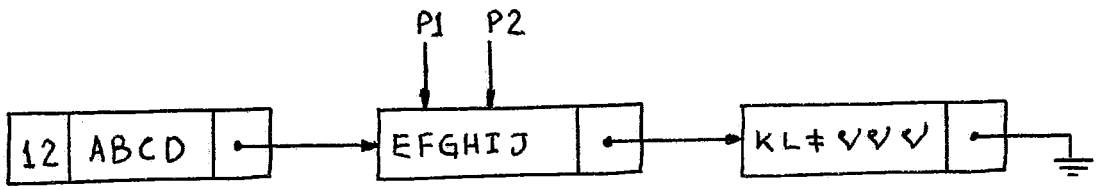
O PRIMEIRO CASO, QUANDO NÃO HÁ PADRÃO, É O MAIS SIMPLES DE TODOS, SENDO FEITO UMA CHAMADA PARA A SUBROTINA AUXILIAR COPIA QUE COPIA A CADEIA PARÂMETRO EM CIMA DA CADEIA DE REFERÊNCIA.

DE TODOS OS CASOS ACIMA CITADOS, O MAIS COMPLEXO É O SEGUNDO, QUANDO OS DOIS PONTEIROS APONTAM PARA O MESMO NÓ. NESTE CASO É NECESSARIO QUE SE SALVE O NÓ EM UM NÓ INTERMEDIARIO ANTES DE COMECAR A COPIAR A CADEIA PARÂMETRO A PARTIR DO PRIMEIRO PONTEIRO. APÓS A COPIA DA CADEIA PARAMETRO, TORNA-SE NECESSARIO COPIAR A SEGUIR A PARTE DO NÓ SALVO QUE ESTÁ DEPOIS DO SEGUNDO PONTEIRO. FEITO ISTO, O RESTANTE DO NÓ É PREENCHIDO COM VAZIOS E SEU APONTADOR LIGADO CONVENIENTEMENTE COM A CADEIA RESTANTE.

EXEMPLO:

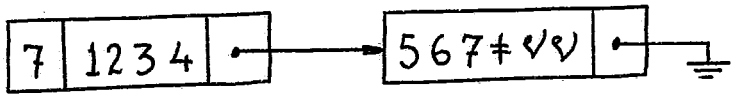
1) SITUAÇÃO ORIGINAL:

CADEIA DE REFERÊNCIA:

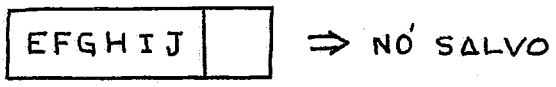


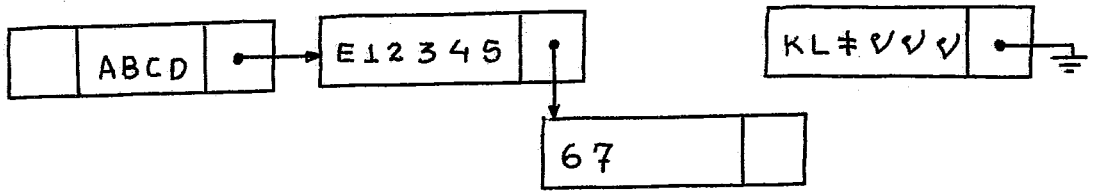
P1 E P2 SAO OS PONTEIROS QUE APONTAM PARA A PARTE QUE CASOU COM O PADRÃO.

CADEIA PARÂMETRO.

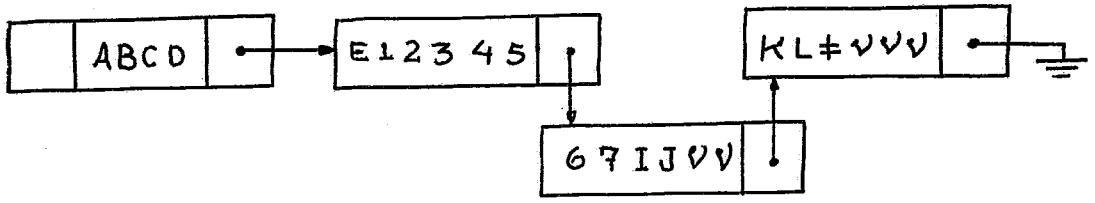


2) FOI SALVO O NÓ DOS PONTEIROS E COPIADA A CADEIA PARÂMETRO:





3) FOI COPIADA A PARTE DO NÓ SALVO E PREENCHIDO O NÓ COM VAZIOS E LIGADO O APONTADOR:



NOTA - DEIXAMOS PROPOSITALMENTE EM BRANCO O CAMPO DO TAMANHO DA CADEIA, POIS MAIS ADIANTE EXPLICAREMOS COMO E PREENCHIDO ESTE CAMPO.

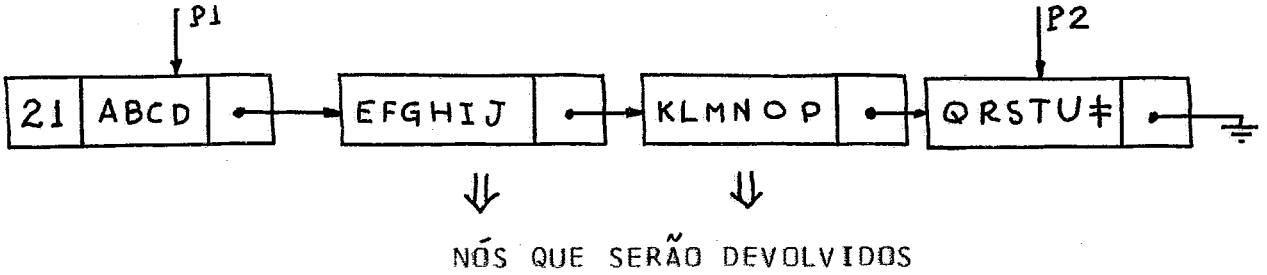
SE NESTE MESMO CASO, A CADEIA PARÂMETRO FOR NULA, APENAS PREENCHEMOS COM VAZIOS O ESPAÇO ENTRE OS PONTEIROS.

QUANDO TEMOS OS DOIS PONTEIROS APONTANDO PARA NÓS DIFERENTES, TORNA-SE NECESSÁRIO DEVOLVER À MEMÓRIA OS NÓS SITUADOS ENTRE OS DOIS. FEITO ISTO, COPIAMOS A CADEIA PARÂMETRO A PARTIR DO PRIMEIRO PONTEIRO. AO TERMINAR, PREENCHEMOS O RESTANTE DO ÚLTIMO NÓ COPIADO COM VAZIOS E O LIGAMOS AO NÓ APONTADO PELO SEGUNDO PONTEIRO QUE TERA SEU INICIO TAMBEM SUBSTITUIDO POR VAZIOS, ATÉ O REFERIDO PONTEIRO.

EXEMPLO:

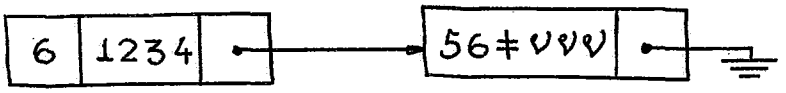
1) SITUAÇÃO ORIGINAL:

CADEIA DE REFERÊNCIA:

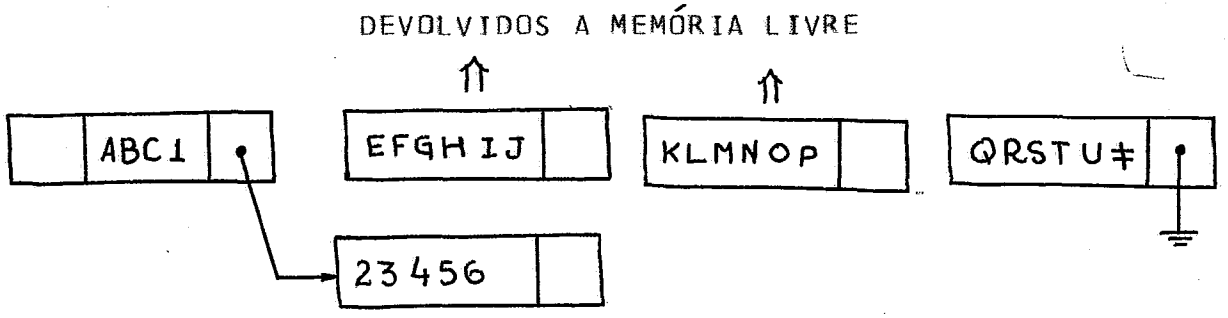


NÓS QUE SERÃO DEVOLVIDOS

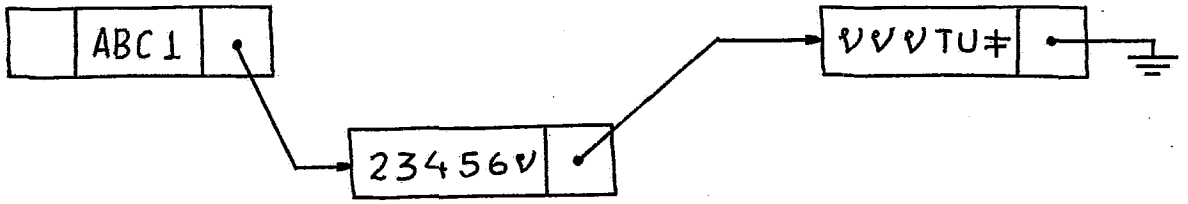
CADEIA PARÂMETRO:



2) FOI COPIADA A CADEIA PARÂMETRO:



3) FOI PREENCHIDO COM VAZIOS OS ESPAÇOS NECESSÁRIOS E FEITA A LIGAÇÃO CONVENIENTE DO ULTIMO NÓ.



SE A CADEIA PARÂMETRO FOSSE NULA, O PROCESSAMENTO SE RESUMIRIA EM DEVOLVER OS NÓS E COMPLETAR OS NÓS APOS O PRIMEIRO PONTEIRO E ANTES DO SEGUNDO COM VAZIOS.

O TAMANHO DA NOVA CADEIA É CALCULADO DA MESMA FORMA, INDEPENDENTE DA POSIÇÃO DOS PONTEIROS.

EXISTE NA ÁREA DE COMUNICAÇÃO, UMA PALAVRA QUE CONTÉM O TAMANHO DA PARTE DA CADEIA DE REFERÊNCIA QUE CASOU COM O PADRÃO. ESTE TAMANHO É CALCULADO E PREENCHIDO PELA SUBROTINA DE VARREDURA.

O TAMANHO DA NOVA CADEIA DE REFERÊNCIA SERÁ IGUAL AO TAMANHO ORIGINAL, MENOS O TAMANHO DA MESMA QUE CASOU COM O PADRÃO, ADICIONADO AO TAMANHO DA CADEIA PARÂMETRO. O TAMANHO DAS CADEIAS DE REFERÊNCIA E PARÂMETROS SÃO APANHADOS NA PRIMEIRA PALAVRA DA CADEIA.

SE NÃO HOUVE PADRÃO NESTA DECLARAÇÃO, O TAMANHO DA NOVA CADEIA SERA IGUAL AO TAMANHO DA CADEIA PARÂMETRO.

3.3.8 - ENDEREÇAMENTO INDIRETO (IND LIND):

UM ENDEREÇAMENTO INDIRETO É FEITO, RETIRANDO-SE A CADEIA DO TOPO DA PILHA DE PARÂMETROS E PESQUISANDO NA TS UM ELEMENTO CUJO NOME SEJA IDÊNTICO AO VALOR DO CONTEÚDO DESTA CADEIA.

AS DUAS SUBROTINAS IND E LIND, SÃO ENTRADAS DE UMA ÚNICA SUBROTINA QUE FAZ O ENDEREÇAMENTO INDIRETO. A ENTRADA EM LIND, FAZ COM QUE SEJA LIGADO UM INDICADOR PARA SABERMOS QUE QUEREMOS FAZER O ENDERECAMENTO INDIRETO EM UM RÓTULO.

NÃO É NECESSÁRIO MANTER, DURANTE A FASE EXECUÇÃO, UMA ROTINA TÃO GENÉRICA QUANTO A REALIZADA NA FASE COMPILAÇÃO, POR ESTE MOTIVO FOI REFEITA A SUBROTINA DE PROCURA NA TS DA FASE COMPILAÇÃO, SUPRIMINDO-SE AS PARTES DESNECESSÁRIAS E INSERINDO-SE OUTRAS.

A SUBROTINA FAZ PRIMEIRAMENTE UMA PESQUISA NA TS PARA PROCURAR A ENTRADA CUJO NOME SEJA IDÊNTICO AO CONTEÚDO DA CADEIA QUE QUEREMOS ENDEREÇAR INDIRETAMENTE. ESTA PESQUISA TEM DOIS RESULTADOS POSSIVEIS: A ENTRADA FOI ACHADA OU A ENTRADA NÃO FOI ACHADA.

SE ACHARMOS A ENTRADA DESEJADA, TESTAMOS SE O TIPO DO ELEMENTO ACHADO É IGUAL AO TIPO DO ELEMENTO PESQUISADO. SE FOR DIFERENTE, CONTINUAMOS A PESQUISA E SE FOR IGUAL O ENDEREÇO DA CADEIA OU DO RÓTULO ACHADO E RECALCADO NA PILHA DE PARÂMETROS.

SE A PESQUISA FALHAR, É TESTADO O INDICADOR DE RÓTULO. SE ESTIVER LIGADO (É DESEJADO UM RÓTULO) CHAMAMOS A SUBROTINA DE ERRO QUE IMPRIME A MENSAGEM E SUPRIME A EXECUÇÃO. SE O INDICADOR ESTIVER DESLIGADO, É CRIADA UMA NOVA ENTRADA NA TS PARA O NOME DESEJADO E É ALOCADA UMA VARIÁVEL DE VALOR NULO PARA ESTE ELEMENTO. SEU ENDEREÇO É ENTÃO RECALCADO NA PILHA DE PARÂMETROS.

COMO QUASE TODAS AS SUBROTINAS DA FASE EXECUÇÃO, É TESTADO SE A CADEIA DO TOPO DA PILHA DE PARÂMETROS É INTERMEDIÁRIA E EM CASO AFIRMATIVO ELA É DEVOLVIDA A MEMÓRIA NO FINAL DA SUBROTINA.

3.3.9 - VARREDURA DO PADRÃO (PATT):

SENDO SNOBOL UMA LINGUAGEM DE MANIPULAÇÃO DE CADEIAS DE SÍMBOLOS, A PARTE MAIS USADA DA LINGUAGEM É A COMPARAÇÃO COM PADRÕES. ESTA PARTE É TAMBÉM A MAIS CRITICA DA FASE EXECUÇÃO, EM TERMOS DE VELOCIDADE DE PROCESSAMENTO, PRINCIPALMENTE QUANDO A COMPARAÇÃO FALHA, POIS DEVEM SER EXAURIDAS TODAS AS POSSIBILIDADES DE SUCESSO.

3.3.9.1 - NOMENCLATURA USADA:

PARA FACILITAR A COMPREENSÃO DO TEXTO FAZEMOS AQUI UMA BREVE DESCRIÇÃO DA NOMENCLATURA E SIMBOLOGIA USADA PARA DESCREVER O ALGORÍTMO DE VARREDURA.

O PADRÃO É COMPOSTO DE VARIOS ELEMENTOS REPRESENTADOS PELA LETRA 'E', SÃO INDEXADOS DE 1 A N (SENDO N O NUMERO DE ELEMENTOS NO PADRÃO).

A CADEIA A SER TESTADA E COMPOSTA DE SIMBOLOS QUE SÃO REPRESENTADOS PELA LETRA 'C' E INDEXADOS DE 1 A M.

O TIPO DE CADA ELEMENTO DO PADRÃO SERÁ REPRESENTADO:

A - PARA ELEMENTO ARBITRÁRIO.
B - PARA ELEMENTO BALANCEADO.
F - IDEM DE TAMANHO FIXO.
K - IDEM CONSTANTE.
R - IDEM REFERÊNCIA PASSADA.

LEMBRAMOS QUE OS TIPOS A E B PODEM SER F AO MESMO TEMPO. DESTA MANEIRA, AO NOS REFERIRMOS A UM TIPO A OU B ESTAMOS REFERINDO A TIPOS NÃO F E AO NOS REFERIRMOS A UM TIPO F, ENGLOBAMOS OS TIPOS AF E BF.

POR EXEMPLO:

'CADEIA A TESTAR' *A* *(B)* *F/'14'* 'CONST' X A

CADEIA A SER TESTADA:
C(1) = C
C(2) = A
C(3) = D
.....
.....
C(15) = R

PADRÃO:
E(1) = A - TIPO A*
E(2) = (B) - TIPO B
E(3) = F/'14' - TIPO AF
E(4) = 'CONST'- TIPO K
E(5) = X - TIPO K
E(6) = A - TIPO R

QUANDO HÁ REFERÊNCIA PASSADA, A CADEIA QUE É REFERENCIADA E REPRESENTADA COM UM '*'.

3.3.9.2 - QUATRO REGRAS BÁSICAS:

A COMPARAÇÃO DO PADRÃO, SE PROCESSA SEGUNDO QUATRO REGRAS SIMPLES (6).

REGRA 1:
É TENTADO CASAR O PADRAO E(1) COMECANDO DO PRIMEIRO SÍMBOLO DA CADEIA. SE NÃO FOR POSSÍVEL, É TENTADO O PRÓXIMO SÍMBOLO E ASSIM POR DIANTE.

REGRA 2:
A COMPARAÇÃO SE PROCESSA DA ESQUERDA PARA A DIREITA, SUCESSIVAMENTE COMPARANDO OS ELEMENTOS DO PADRÃO. CADA ELEMENTO DO PADRÃO CASA COM A MENOR SUBCADEIA POSSÍVEL.

REGRA 3:
SE EM ALGUM PONTO UM ELEMENTO NÃO PUDER CASAR COM UMA SUBCADEIA, TENTA-SE NOVAMENTE CASAR O PADRÃO ANTERIOR. PARA ISSO, ESTENDE-SE A SUBCADEIA ATÉ FORMAR A PRÓXIMA MAIOR SUBCADEIA ACEITÁVEL. SE NÃO FOR POSSÍVEL, APLICA-SE NOVAMENTE A REGRA 3. SE NÃO HOUVER ELEMENTO ANTERIOR, TENTA-SE A REGRA 1.

REGRA 4:
SE O ÚLTIMO ELEMENTO DO PADRÃO É UM ELEMENTO ARBITRÁRIO, O CASAMENTO DA SUBCADEIA A ESTE ELEMENTO É ESTENDIDA ATÉ O FIM DA CADEIA.

PARA SE POUPAR O TEMPO, O ALGORÍTMO DE VARREDURA DA CADEIA É ACRESCIDO DE ALGUNS HEURÍSTICOS QUE TESTAM CERTAS CONDIÇÕES MÍNIMAS NECESSÁRIAS PARA QUE HAJA SUCESSO.
UM DELES É QUANTO AO TAMANHO MÍNIMO QUE A SUBCADEIA RESTANTE DEVE TER PARA PODER CASAR COM O MENOR PADRÃO POSSÍVEL.
PARA ESTE TESTE DAMOS PESOS AOS PADRÕES; PESO ESTE QUE SERÁ REPRESENTADO PELA LETRA 'R'. PESO DE UM ELEMENTO É O TAMANHO MÍNIMO QUE PODE ASSUMIR.

TIPO	PESO - R(I)
A	ZERO
B	1
F	TAMANHO ESPECIFICADO
K	COMPRIMENTO DA CONST.
R	R(I)*

NO INÍCIO DO TESTE DE CADA PADRÃO, TESTAMOS SE A SUBCADEIA RESTANTE APRESENTA UM TAMANHO MÍNIMO NECESSÁRIO

PARA HAVER SUCESSO. ESTE TAMANHO É DADO POR:

$$W(I) = \sum_{J=I}^{N} R(J)$$

CASO NÃO SEJA SATISFEITA ESTA CONDIÇÃO DIZEMOS QUE HOUVE FALHA DE TAMANHO.

3.3.9.3 - PADRÃO AUMENTADO:

AO PADRÃO PROPOSTO, ACRESCENTAMOS DOIS ELEMENTOS FANTASMAS E(0) E E(N+1) QUE SÃO TIPO ARBITRARIO (A), FORMANDO UM PADRÃO AUMENTADO.

E(0) E(1) E(2) ..... E(N) E(N+1)

ESTES ELEMENTOS NÃO CAUSAM MODIFICAÇÃO QUANTO AO ALGORÍTMO DE VARREDURA, POIS SENDO DO TIPO ARBITRÁRIO, QUALQUER CADEIA PODERÁ CASAR COM ELES.

A FINALIDADE DO ELEMENTO E(0) É DE TERMOS CERTEZA DE SEMPRE CASAR COM O PRIMEIRO ELEMENTO DO PADRÃO, PARA NÃO REQUERER TRATAMENTO ESPECIAL PARA A REGRA 1.

O ELEMENTO E(N+1) TORNA-SE NECESSÁRIO, POIS COMO DIREMOS MAIS ADIANTE, O FINAL DE UM ELEMENTO E(I) É DADO PELO INICIO DO ELEMENTO E(I+1).

DESTA FORMA O ELEMENTO E(N+1) APENAS SERVE PARA MARCAR O FINAL DE E(N).

3.3.9.4 - ESTRUTURA GERAL:

OS DELIMITADORES QUE MARCAM O INICIO E FIM DE UM ELEMENTO E(I) SÃO DOIS PONTEIROS P(I) E P(I+1). P(I) É OBTIDO NA COMPARAÇÃO DO ELEMENTO E(I-1) E P(I+1) É OBTIDO NO FIM DA COMPARAÇÃO DE E(I) SE HOUVER SUCESSO.

NO TEXTO, SEMPRE QUE NOS REFERIRMOS P(I), ESTAMOS SUPONDO QUE ELE CONTEM A ORDEM DO CARACTER DENTRO DA CADEIA A TESTAR.

POR EXEMPLO:

```
P(1)      P(2)
  ↓         ↓
C A D E I A   A   T E S T A R
```

P(1) = 1 E P(2) = 6

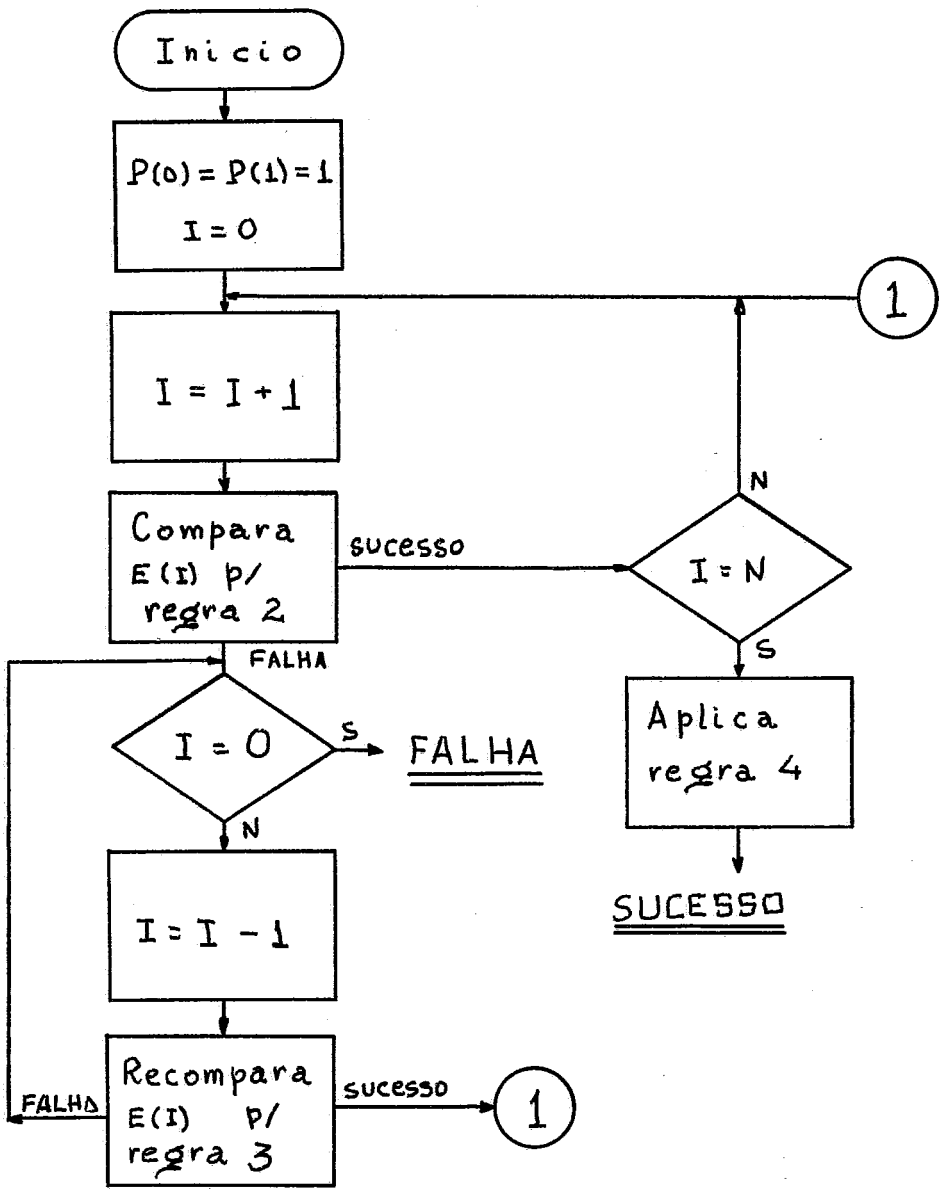
MAIS TARDE QUANDO EXPLICARMOS A PARTE INICIALIZADORA DA SUBROTINA DE VARREDURA SERA EXPLICADO COM MAIS DETALHES O FORMATO DOS PONTEIROS.

INICIALMENTE, DE ACORDO COM AS REGRAS 1 E 2, UMA CADEIA VAZIA COMECANDO NO PRIMEIRO SIMBOLO É ASSINALADA A E(0). ISTO É REALIZADO FAZENDO-SE:

P(0) = P(1) = 1

SENDO E(0) NULO, É TENTADO CASAR E(I) COMECANDO NO P(I)ÉSIMO SÍMBOLO DA CADEIA. A COMPARAÇÃO É FEITA DE ACORDO COM AS REGRAS 2, 3 E 4.

ESTRUTURA GERAL SIMPLIFICADA:



3.3.9.5 - ALGORÍTMO DA REGRA 2:

ANTES DE SER TENTADA A COMPARAÇÃO COM O ELEMENTO E(1), FAZEMOS UM TESTE DE COMPRIMENTO PARA TER CERTEZA QUE A CADEIA SATISFAZ AS CONDIÇÕES MÍNIMAS DO PADRÃO. O TESTE É SATISFEITO SE:

$$W(0) \leqslant M$$

EM GERAL, OS REQUISITOS MÍNIMOS DE E(I) .... E(N+1) SÃO SATISFEITOS SE:

$$P(I)-1+W(I) \leqslant M$$

CASO CONTRÁRIO OCORRERÁ FALHA DE TAMANHO.

APÓS O TESTE INICIAL, SÓ HÁ NECESSIDADE DE NOVOS TESTES SE O COMPRIMENTO DA SUBCADEIA QUE CASOU COM E(I) FOR MAIOR QUE R(I).

DE ACORDO COM A REGRA 2, PONTEIROS SÃO ASSINALADOS A E(I) COMO ABAIXO.

1) A - P(I+1) = P(I) (CADEIA VAZIA)

2) AF - P(I+1) = P(I) + R(I)

3) K - SE C(P(I)) .... C(P(I)+R(I)-1) FOR VALOR ACEITÁVEL DE E(I) ENTAO P(I+1) = P(I) + R(I), CASO CONTRÁRIO HÁ FALHA DE CASAMENTO.

4) R - SEJA S O COMPRIMENTO DA SUBCADEIA ASSINALADA A E(I)*. SE C(P(I)) .... C(P(I)+S-1) É O MESMO QUE A SUBCADEIA ASSINALADA A E(I)*, ENTAO P(I+1) = P(I)+S. CASO CONTRARIO HÁ FALHA DE CASAMENTO.

5) B - SE C(P(I)) NÃO É PARENTESES ENTÃO P(I+1) = P(I)+1. SE C(P(I)) FOR PARÊNTESES A DIREITA HÁ FALHA DE CASAMENTO.

SE C(P(I)) FOR PARÊNTESES A ESQUERDA É FORMADO UM CONTADOR DE PARÊNTESES PARA ACHAR A MENOR SUBCADEIA QUE CASE COM E(I). SE NÃO PUDER SER ENCONTRADA ESTA SUBCADEIA, HÁ FALHA DE CASAMENTO.

O ALGORÍTMO PARA SE ACHAR A MENOR SUBCADEIA BALANCEADA COMECANDO EM C(J) É DADO NO APENDICE I.

6) BF - SE C(P(I)) .... C(P(I)+R(I)-1) FOR UMA CADEIA BALANCEADA, ENTÃO:

$$P(I+1) = P(I) + R(I).$$

CASO CONTRÁRIO HÁ FALHA DE CASAMENTO.

NO APENDICE I É APRESENTADO O DIAGRAMA DE BLOCOS DO ALGORÍTMO DA REGRA 2.

3.3.9.6 - ALGORÍTMO DA REGRA 3:

PARA INCREMENTARMOS A EFICIÊNCIA DO ALGORÍTMO, VAMOS DAR TRATAMENTOS DISTINTOS AOS DIVERSOS TIPOS DE FALHA.

3.3.9.6.1 - FALHA DE CASAMENTO:

UMA FALHA DESTE TIPO SÓ PODE OCORRER SE E(I) FOR DO TIPO K, B, BF OU R. DE ACORDO COM A REGRA 3, DEVE-SE RECOMPARAR O PADRAO E(I-1) AUMENTANDO SUA SUBCADEIA. MAS SE E(I-1) FOR DO TIPO F, R OU K A SUBCADEIA NÃO PODERA SER AUMENTADA, O QUE É EQUIVALENTE A OUTRA FALHA DE CASAMENTO. ENTAO O INDICE I E DESCREMENTADO ATE UM ELEMENTO QUE SEJA A OU B.

SE E(I) FOR A FAZEMOS

P(I+1) = P(I+1) + 1

E VOLTAMOS A REGRA 2.

SE E(I) FOR B, A RECOMPARAÇÃO PARA E(I) É FEITA ACHANDO-SE A MENOR SUBCADEIA DE E(I). O COMPRIMENTO DESTA SUBCADEIA BALANCEADA E OBTIDO PELO MÉTODO DESCRITO E FAREMOS

P(I+1) = P(I+1) + S

SENDO S O TAMANHO DA MENOR SUBCADEIA ACHADA. VOLTA-SE ENTÃO A REGRA 2.

3.3.9.6.2 - FALHA DE TAMANHO:

UMA FALHA DE TAMANHO OCORRE SE O NUMERO DE SÍMBOLOS RESTANTES NA CADEIA FOR INSUFICIENTE PARA SATISFAZER OS REQUISITOS MÍNIMOS DOS ELEMENTOS RESTANTES.

A COMPARAÇÃO COM O PADRÃO PODE TER SUCESSO SOMENTE SE SUBCADEIAS MENORES PUDEREM SER ASSINALADAS AOS ELEMENTOS ANTERIORES. EMBORA A APLICACAO DA REGRA 3 SO POSSA AUMENTAR AS SUBCADEIAS, AS SUBCADEIAS DOS ELEMENTOS SUBSEQUENTES PODEM SER DIMINUIDOS.

HÁ UM INDICADOR $S NO ALGORÍTMO DA REGRA 2 QUE É LIGADO SE APARECER ALGUM ELEMENTO DO TIPO R OU B.

SE O INDICADOR $S NÃO FOI LIGADO, SOMENTE ELEMENTOS DO TIPO A, K OU F FORAM ENCONTRADOS ANTERIORMENTE. NENHUMA

TENTATIVA DE REDUZIR O TAMANHO DE UM ELEMENTO PODERÁ RESULTAR EM SUCESSO, POIS OS TESTES ANTERIORES EXAURIRAM TODAS AS POSSIBILIDADES DE UM CASAMENTO COM UMA SUBCADEIA MENOR. NESTE CASO A VARREDURA DO PADRÃO FALHA.

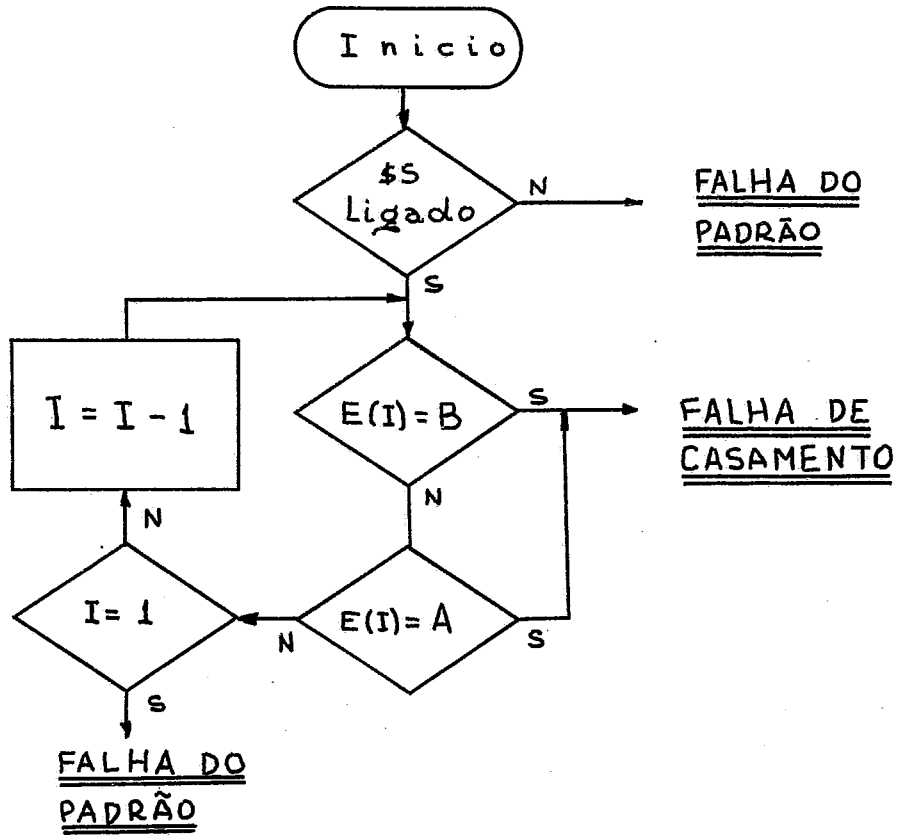
SE O INDICADOR $S FOI LIGADO, SOMENTE HAVERA A POSSIBILIDADE DE TERMOS UMA SUBCADEIA MENOR ASSINALADA AOS ELEMENTOS E(0) E(1) ... E(I) SE PUDERMOS DIMINUIR O COMPRIMENTO DE UM ELEMENTO DO TIPO B OU A*. O ELEMENTO B PODE SER DIMINUIDO SE O PONTEIRO DO SEU INICIO P(I), FOR AVANCADO ATRAVÉS DE UMA RECOMPARAÇÃO DE ELEMENTOS ANTERIORES. O MESMO ACONTECE COM A*, RESULTANDO UM VALOR MENOR PARA R.

EXEMPLOS DE PADRÕES QUE ACONTECEM OS CASOS ACIMA.

1) CAD = '(A B (C D)) EF'
CAD *A* *(B)* *C/'5'* 'D'

2) 'AABDED' *A* *B/'1'* *D* 'E' D

O ALGORÍTIMO PARA O TRATAMENTO DE FALHA DE TAMANHO E DADO A SEGUIR:



3.3.9.7 - PARTE INICIALIZADORA:

DEVIDO AO FATO DE PODERMOS FAZER ENDEREÇAMENTO INDIRETO NO NOME DA VARIÁVEL DE UM ELEMENTO DO PADRÃO, TORNA-SE NECESSÁRIO QUE O TESTE PARA IDENTIFICAÇÃO DO TIPO R (REFERÊNCIA PASSADA) SE FAÇA NA HORA DE SE FAZER A VARREDURA.

O MESMO MOTIVO NOS IMPOSSIBILITA DE, NA FASE COMPILAÇÃO, TESTAR UMA REDEFINIÇÃO DA CADEIA DE REFERÊNCIA ATRAVÉS DE UM ELEMENTO PADRÃO, O QUE É ERRO DE PROGRAMAÇÃO.

DEVEMOS TAMBÉM, NESTE PONTO PREENCHER O CAMPO DO TAMANHO DOS ELEMENTOS DO TIPO K.

PARA FACILITAR A MANIPULAÇÃO DOS ELEMENTOS NA VARREDURA, INSERIMOS MAIS DUAS INFORMAÇÕES ÀS JÁ CONSTANTES NO DP.

SÃO ELAS O TAMANHO MÍNIMO DA SUBCADEIA RESTANTE W(I) E O PONTEIRO P(I).

AINDA PARA FACILITAR A PROGRAMAÇÃO DO ALGORÍTMO DE VARREDURA, O DP É RECOPIADO EM UMA ÁREA INTERMEDIÁRIA QUE FICARÁ LOGO ACIMA DO TOPO DA PILHA DE PARÂMETROS. UMA VEZ ACABADA A VARREDURA, ESTE DP PROVISÓRIO (DPP) E DESMANCHADO. ESTE NOVO DP É MONTADO COM OS ELEMENTOS EM SEQUÊNCIA E TEM O SEGUINTE FORMATO:

SENDO:

TIPO	TIPO - TIPO DO ELEMENTO.
END	END - ENDEREÇO DA VARIÁVEL OU CTE.
PESO	PESO - R(I) - COMO JÁ EXPLICADO.
W(I)	W(I) = R(J)
P1(I)	P1(I) E P2(I) = PONTEIRO QUE APONTA PARA C(P(I)).
P2(I)	
P(I)	P(I) = ORDEM DO CARACTER EM C.

A PARTE INICIALIZADORA FAZ-SE NECESSÁRIA PELOS MOTIVOS EXPOSTOS ACIMA E REALIZA AS TAREFAS DESCRITAS A SEGUIR:

1) MONTA O ELEMENTO P(0) NO DPP.

2) MONTA CADA ELEMENTO DO DP NO DPP TESTANDO SE JÁ HÁ ALGUM ELEMENTO DO TIPO A OU B COM O MESMO ENDEREÇO. EM CASO AFIRMATIVO MODIFICA O TIPO DO ELEMENTO ATUAL PARA R E O DO ELEMENTO REFERENCIADO PARA A* OU B*. PREENCHE O CAMPO DO PESO COM SEU VALOR.

3) CONTA O NÚMERO DE ELEMENTOS DO PADRÃO.
4) MONTA O ELEMENTO P(N+1) NO DPP.
5) VERIFICA SE A CADEIA DE REFERÊNCIA É REDEFINIDA.
6) PREENCHE O CAMPO DOS W(I).
7) INICIALIZA AS CONSTANTES E INDICADORES DA PARTE QUE FAZ A VARREDURA.

3.3.9.8 - FINALIZAÇÃO:

A FINALIZAÇÃO TEM COMO FINALIDADE PRINCIPAL DAR VALOR AS VARIÁVEIS DE TIPO A, B E F QUE CASARAM COM O PADRÃO, NO CASO DE SUCESSO.

ANTES DO DPP SER DESFEITO, SEUS PONTEIROS SÃO USADOS PARA SABERMOS QUE PARTE DA CADEIA DE REFERÊNCIA CASA COM AS VARIÁVEIS.

CADA VARIÁVEL, ANTES DE SER REDEFINIDA, É ESVAZIADA, DEVOLVENDO-SE SEU CONTEÚDO A MEMÓRIA LIVRE. APÓS ISTO, CHAMAMOS A SUBROTINA COPIA QUE SE ENCARREGA DE COPIAR A PARTE DA CADEIA QUE CASOU COM O ELEMENTO RESPECTIVO.

A PARTE QUE CORRESPONDE AO ELEMENTO E(I) É DADA PELOS PONTEIROS P(I) E P(I+1).

FEITO ISTO, É PREENCHIDO NA ÁREA DE COMUNICAÇÃO OS PONTEIROS QUE MARCAM A PARTE DA CADEIA DE REFERÊNCIA QUE CASOU COM O PADRÃO. ESTES PONTEIROS SERÃO P(1) E P(N+1).

É CALCULADO TAMBÉM O TAMANHO DESTA PARTE QUE É COLOCADO EM SEU LUGAR NA ÁREA DE COMUNICAÇÃO.

SÓ ENTÃO É DESFEITO O DPP E O CONTROLE É RETORNADO PULANDO-SE A INSTRUÇÃO DE DESVIO QUE SE ENCONTRA APÓS A CHAMADA.

EM CASO DE FALHA, É DESFEITO O DPP E O CONTROLE É DEVOLVIDO PARA A INSTRUÇÃO SEGUINTE À CHAMADA.

3.3.10 - ENTRADA E SAIDA (E/S):

PARA SE EVITAR QUE AS VARIÁVEIS DE E/S FIQUEM EM POSIÇÃO FIXA E PARA EVITAR UMA PROCURA NA TS TODA VEZ QUE QUIZERMOS USA-LAS, HÁ NA AREA DE COMUNICAÇÃO UM VETOR DE TRES ENTRADAS QUE CONTÉM O ENDEREÇO DE CADA UMA DAS VARIÁVEIS. CADA ENTRADA É ASSOCIADA A UM APARELHO DE E/S (QUE NA IMPLANTACAO SÃO 3). ASSIM, QUANDO QUIZERMOS REALIZAR A E/S O ENDEREÇO DA CADEIA ASSOCIADA AO APARELHO É ACHADO NESTE VETOR.

ESTE SISTEMA DÁ FLEXIBILIDADE DE ESTENDERMOS ESTE

VETOR SE QUIZERMOS AUMENTAR A POTENCIALIDADE DE E/S DO SISTEMA.

O VETOR É PREENCHIDO COM O ENDEREÇO DAS VARIÁVEIS PELO PROGRAMA INICIALIZADOR AO SEREM ESTAS INSERIDAS NA TS.

3.3.10.1 - ENTRADA (SPIT):

ESTA SUBROTINA LÊ UM CARTAO EM UMA ÁREA FIXA DE 80 PALAVRAS, CONVERTE-O PARA EBCDIC E CHAMA A SUBROTINA ALOC QUE SE ENCARREGA DE ALOCAR ESTA ÁREA FIXA NA VARIÁVEL SYSPIT CUJO ENDEREÇO É DADO COMO PARÂMETRO. COMO DISSEMOS ANTERIORMENTE O ENDEREÇO DA VARIÁVEL SYSPIT E ACHADO NA ÁREA DE COMUNICAÇÃO.

3.3.10.2 - SAIDA (SPOT & SPPT):

ESTAS DUAS SUBROTINAS, SPOT E SPPT, FAZEM BASICAMENTE A MESMA FUNÇÃO. A ÚNICA DIFERENCA ENTRE AS DUAS ESTÁ NO TAMANHO DA ÁREA DE SAIDA E O APARELHO QUE REALIZA A OPERAÇÃO.

SPOT TEM UMA AREA DE 120 POSIÇÕES E REALIZA A SAIDA NA IMPRESSORA ENQUANTO SPPT TEM UMA AREA DE 80 POSIÇÕES E REALIZA A SAIDA EM CARTÕES PERFURADOS.

PRIMEIRAMENTE É FEITO UM TESTE PARA SABERMOS SE O CONTEÚDO DA CADEIA É NULO E EM CASO AFIRMATIVO, A SUBROTINA RETORNA SEM REALIZAR NENHUMA OPERAÇÃO.

FEITO ISTO, CALCULAMOS O NÚMERO DE LINHAS A SEREM IMPRESSAS OU O NÚMERO DE CARTÕES A SEREM PERFURADOS.

APÓS ESTE CÁLCULO, CHAMAMOS A SUBROTINA MONTA PARA MONTAR A CADEIA EM UMA ÁREA FIXA. ESTA ÁREA É ENTÃO CONVERTIDA E TRANSMITIDA AO APARELHO DE SAIDA. ESTA TAREFA É REALIZADA O NUMERO DE VEZES QUE FOR NECESSÁRIO, DEPENDENDO DO NÚMERO DE LINHAS OU DE CARTÕES CALCULADOS.

PARTE IV

EXPANSÃO DO SISTEMA

4.1- INTRODUÇÃO:

A FILOSOFIA QUE NOS BASEAMOS AO DESENVOLVER ESTE TRABALHO FOI A DE TERMOS O COMPILADOR PRONTO EM CURTO ESPAÇO DE TEMPO. POR ESTE MOTIVO, DEIXAMOS DE IMPLANTAR CERTAS FACILIDADES QUE AUMENTARIAM A POTENCIALIDADE DO SISTEMA, MAS QUE ENVOLVERIAM UM TRABALHO MAIOR DE PROGRAMAÇÃO PARA SUA IMPLANTAÇÃO. NESTA PARTE SERÃO DISCUTIDAS ALGUMAS DESTAS FACILIDADES.

4.2- TABELA DE SÍMBOLOS:

COMO DISSEMOS ANTERIORMENTE (PARÁGRAFO 1.2.3), NA FASE EXECUÇÃO, A TS SÓ SERÁ CONSULTADA AO FAZERMOS UM ENDEREÇAMENTO INDIRETO. É PORTANTO VIÁVEL A SUA ALOCAÇÃO NO DISCO.

A ALOCAÇÃO DA TS NO DISCO APRESENTA AS SEGUINTES VANTAGENS:

1) GANHO CONSIDERÁVEL DE MEMÓRIA.
2) FACILIDADE PARA UMA TÉCNICA DE PAGINAÇÃO (VER PARÁGRAFO 4.3).
3) TAMANHO PRATICAMENTE ILIMITADO.

MAS EM COMPENSAÇÃO AUMENTARIA O TEMPO DE COMPILAÇÃO E DEPENDENDO DO PROGRAMA, O TEMPO DE EXECUÇÃO.

A ORGANIZAÇÃO DA TS EM TABELA DE ACESSO ALEATÓRIO PODERIA SER MANTIDA. MAS COMO A ÚNICA PARTE DA TS QUE FICARIA NA MEMÓRIA SERIA O VETOR DOS INÍCIOS, PODERIAMOS EXTENDER SEU TAMANHO PARA AUMENTARMOS A VELOCIDADE DE PROCURA DE UM DETERMINADO ELEMENTO. NESTE CASO, SERIA NECESSÁRIO TAMBÉM MUDARMOS O MÉTODO DE GERAÇÃO DO CÓDIGO ALEATÓRIO QUE ATUALMENTE GERA UM CODIGO MÓDULO 16.

NESTA ORGANIZAÇÃO, O VETOR DOS INÍCIOS CONTERIA O ENDEREÇO DO DISCO EM QUE FOI COLOCADO O ELEMENTO RESPECTIVO. EM CASO DE SINÔNIMOS (MESMO CÓDIGO ALEATÓRIO PARA SÍMBOLOS DIFERENTES), OS ELEMENTOS SERIAM MONTADOS SEQUENCIALMENTE NO DISCO.

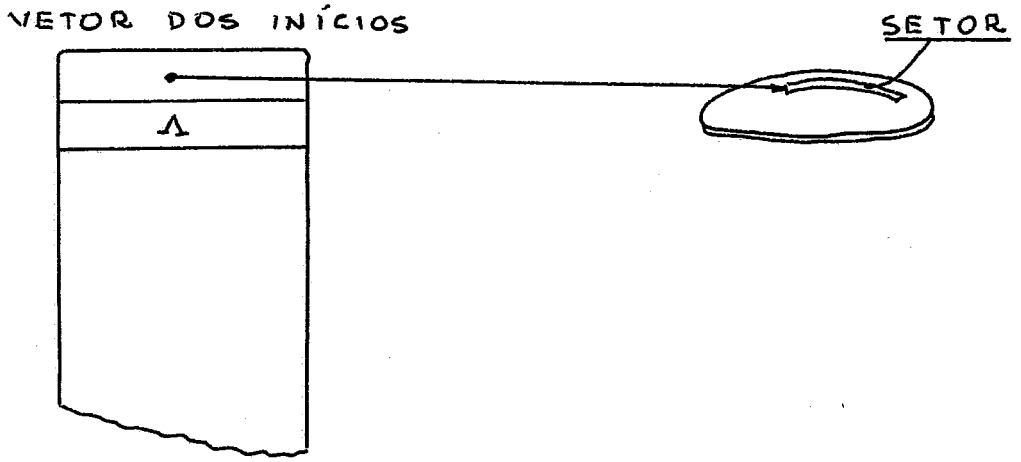
SERIA NECESSÁRIO TAMBÉM, MANTER-SE CONTRÔLE DA ÁREA UTILIZADA NO DISCO, HAVENDO UMA SUBROTINA QUE QUANDO REQUISITADA, FORNECERIA O ENDEREÇO DE UM SETOR LIVRE NO DISCO.

HAVERIA TAMBÉM A NECESSIDADE DE INFORMAÇÃO SUPLEMENTAR NA TS PARA QUE PUDESSEMOS IDENTIFICAR A CONTINUAÇÃO EM OUTRO SETOR.

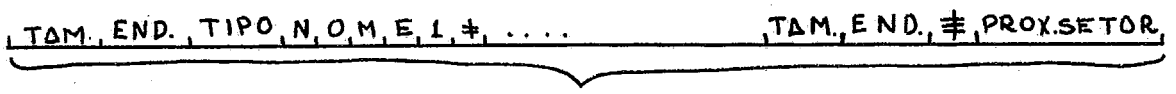
APRESENTAMOS A SEGUIR UMA SUGESTÃO PARA O FORMATO DA TS, CONSIDERANDO SUA ALOCAÇÃO NO DISCO.

CARACTERES ESPECIAIS USADOS:

Λ - INDICA PARTE DA TS AINDA VAZIA OU FINAL DA PARTE.
ǂ - MARCA O FINAL DE UMA ENTRADA.
ǂǂ - AVISA QUE A PROXIMA PALAVRA É O ENDEREÇO DO SETOR EM QUE ESTA PARTE CONTINUA.



SETOR DO DISCO:



320 PALAVRAS

PROX. SETOR:

320 PALAVRAS

SENDO:

TAM - TAMANHO EM PALAVRAS DA ENTRADA.
END - ENDEREÇO DO ELEMENTO NA MEMÓRIA.
TIPO - TIPO DO ELEMENTO: RÓTULO, FUNÇÃO, ETC...
NOME1- NOME DO ELEMENTO TERMINANDO PELO CARACTER ESPECIAL ǂ .

≢ - CARACTER ESPECIAL QUE INDICA QUE A PRÓXIMA PALAVRA CONTEM O ENDEREÇO DO SETOR ONDE CONTINUA ESTA PARTE DA TS.

PROX.SETOR - ENDEREÇO DO SETOR CONTINUAÇÃO.

SENDO AGORA A ORGANIZAÇÃO DA TS SEQUENCIAL, NÃO HÁ MAIS NECESSIDADE DE SE MANTER UM APONTADOR PARA O PRÓXIMO ELEMENTO. TROCAMOS POIS ESTE CAMPO PELO TAMANHO DA ENTRADA PARA FACILITAR A PROCURA.

4.3 - PAGINAÇÃO DE MEMÓRIA:

DE TODAS AS EXPANSÕES DESEJÁVEIS, ESTA É SEM DÚVIDA A MAIS ÚTIL. ISTO PORQUE, AO LIDARMOS COM TEXTOS, GERALMENTE NECESSITAMOS DE GRANDE CAPACIDADE DE MEMÓRIA PARA PODERMOS ARMAZENAR EXTENSAS CADEIAS. A PAGINAÇÃO AQUI DESCRITA REFERE-SE APENAS ÀS VARIAVEIS.

O MAIOR PROBLEMA PARA PAGINACAO RESIDE NO FATO DE NÃO PODERMOS ENDEREÇAR MAIS DE 64 K (SENDO K=1024) VALORES DIFERENTES USANDO APENAS UMA PALAVRA COMO ENDEREÇO. UMA VEZ QUE USAMOS O BIT ZERO PARA INDICAR VARIAVEL TEMPORARIA, ESTE VALOR BAIXA PARA 32 K, QUE É UM VALOR BEM AQUÉM DO DESEJADO.

MODIFICAR O FORMATO DO ENDERECO PARA 2 PALAVRAS, ENVOLVERIA UMA MUDANCA NA ESTRUTURA DO SISTEMA E SERIA NECESSÁRIO REESCREVER GRANDE PARTE DO MESMO.

NOTE QUE QUANDO FALAMOS EM ENDEREÇO, ESTAMOS NOS REFERINDO TANTO AO ENDEREÇO DE UMA VARIÁVEL QUANTO AO ENDEREÇO DE UM NÓ. DESTA FORMA, SE FIZESSEMOS O ENDEREÇO EM 2 PALAVRAS, CADA NÓ TERIA UM CAMPO DE 2 PALAVRAS QUE NÃO CONTERIA INFORMAÇÃO UTIL, BAIXANDO ASSIM A CAPACIDADE DA MEMORIA UTILIZAVEL.

UMA MANEIRA DE SE EXTENDER A CAPACIDADE DE ENDEREÇAMENTO, É ALOCARMOS OS NÓS SEMPRE EM ENDEREÇAMENTOS MÚLTIPLOS DE 4 (PARA NÓS COM TAMANHO DE 4 PALAVRAS). ESTE ARTÍFICIO QUADRUPLICA A CAPACIDADE DE ENDEREÇAMENTO PODENDO ASSIM ENDERECARMOS 32 K NÓS AO INVÉS DE 32 K PALAVRAS, O QUE JÁ PASSA A SER RAZOÁVEL, POIS PODERIAMOS UTILIZAR 128 K PALAVRAS PODENDO ARMAZENAR ATE 192 K CARACTERES, JÁ QUE CADA NÓ CONTEM 6 CARACTERES.

O PROBLEMA AGORA RESIDE NA ESCOLHA DO NÚMERO DE PÁGINAS A SEREM IMPLANTADAS. PODEMOS USAR POR EXEMPLO:

NO. DE PAGINAS	NÓS POR PAG.	PALAVRAS POR PAG.
8	4 K	16 K
16	2 K	8 K
32	1 K	4 K
64	512	2 K

UM VALOR RAZOÁVEL NOS PARECE SER 16 PAGINAS DE 8 K PALAVRAS, POIS 8 K E O MAIOR MÚLTIPLO DE QUALQUER TAMANHO DE MEMÓRIA. POR EXEMPLO, SE TIVERMOS UM COMPUTADOR DE 16 K DE MEMÓRIA, FICARÍAMOS COM 8 K PARA O SISTEMA E UMA PAGINA DE 8K. SE O COMPUTADOR FOSSE DE 32K DE MEMÓRIA, PODEMOS TER 3 PÁGINAS SIMULTANEAMENTE NA MEMÓRIA.

O MOTIVO DE ESCOLHERMOS O MAIOR MÚLTIPLO DE QUALQUER TAMANHO DE MEMÓRIA PARA O TAMANHO DA PÁGINA SE BASEIA EM DUAS CONSIDERACOES: 1) QUANTO MAIOR FOR A PÁGINA, MENOS VEZES NECESSITAMOS DE RECORRER AO DISCO PARA BUSCAR NOVA PÁGINA. 2) SE O TAMANHO DA PÁGINA FOR MÚLTIPLO DE QUALQUER TAMANHO DE MEMÓRIA, PODEREMOS COM MAIOR FACILIDADE REALIZAR UM PROGRAMA GENÉRICO PARA CONTROLAR A PAGINAÇÃO.

EMBORA UMA PÁGINA MAIOR EVITE ACESSOS FREQUENTES AO DISCO, TORNA TAMBÉM A LEITURA DA PÁGINA MAIS DEMORADA, POIS A QUANTIDADE DE INFORMAÇÃO É MAIOR, ACARRETANDO PORTANTO MAIOR MOVIMENTO DAS CABECAS DE LEITURA. POR OUTRO LADO, UM TAMANHO MENOR DE PÁGINA, EMBORA ACARRETE EM UM ACESSO MAIS FREQUENTE AO DISCO, TEM UM TEMPO DE LEITURA MENOR.

O TEMPO TOTAL GASTO COM O CONTRÔLE DA PAGINAÇÃO EM UM PROGRAMA, É DADO PELO PRODUTO DO NÚMERO DE ACESSOS A UMA PÁGINA DIFERENTE PELO TEMPO GASTO PARA ESCREVER A PÁGINA ANTIGA E LER UMA PÁGINA NOVA. UMA PÁGINA GRANDE TEM O PRIMEIRO FATOR PEQUENO E O SEGUNDO GRANDE. NUMA PÁGINA PEQUENA, OCORRE O INVERSO. O PONTO ÓTIMO PARA O TAMANHO DA PAGINA SÓ PODE SER ENCONTRADO ATRAVÉS DE UMA SIMULAÇÃO, POIS O NÚMERO DE ACESSOS A UMA NOVA PÁGINA E ALEATÓRIO.

CONSIDERANDO UM SISTEMA DE 16 PAGINAS DE 8 K PALAVRAS, APRESENTAMOS O SEGUINTE FORMATO PARA O ENDEREÇO QUE SERIA COMPOSTO DE 3 CAMPOS:

1- INDICADOR DE VARIÁVEL TEMPORARIA (BIT 0).
2- ENDEREÇO DO NÓ NA PAGINA (BITS 1-11).
3- NÚMERO DA PAGINA (BITS 12-15).

AO ENDEREÇARMOS UM NÓ QUALQUER, O INDICADOR DE VARIÁVEL TEMPORÁRIA FICARIA EM BRANCO, POIS NÃO SE APLICA.

O ENDEREÇO DE UM NÓ NA MEMÓRIA SERIA CALCULADO DA SEGUINTE MANEIRA:

$$EN = (EP - 1) * 4 + IP$$

SENDO: EN- ENDEREÇO DO NÓ NA MEMÓRIA.
EP- ENDEREÇO DO NÓ NA PÁGINA.
IP- ENDEREÇO DO PRIMEIRO NÓ DA PÁGINA.
4- TAMANHO DO NÓ EM PALAVRAS.

PARA AUMENTARMOS AINDA MAIS A CAPACIDADE DO SISTEMA, SERIA NECESSÁRIO AUMENTARMOS O TAMANHO DO NÓ. PODERIAMOS TER, POR EXEMPLO:

NO.PAL./NÓ	NO.CARAC./NÓ	CAPAC.EM CARAC.
5	8	256 K
6	10	320 K
7	12	352 K
10	18	448 K

O CONTRÔLE DA PAGINAÇÃO FICARIA AO ENCARGO DE UMA SUBROTINA E AO CHAMARMOS QUALQUER SUBROTINA QUE PEÇA UM APONTADOR, ENDEREÇO OU NÓ, ESTA SUBROTINA SE ENCARREGARIA DE TRAZER PARA A MEMÓRIA A PÁGINA ONDE SE ENCONTRA O NÓ DESEJADO.

NESTE ESQUEMA O CONTRÔLE DA MEMÓRIA DISPONÍVEL TAMBÉM DEVE SER MODIFICADO. CADA PÁGINA CONTERÁ SUA LISTA DE VAZIOS. O INICIO DESTA LISTA, JUNTO COM 2 APONTADORES, QUE DÃO O INÍCIO E O FINAL DO ESPAÇO LIVRE CONTÍGUO, FICARIAM EM UMA POSIÇÃO PRÉ-DEFINIDA NA PÁGINA, POR EXEMPLO, AS 3 PRIMEIRAS PALAVRAS.

ASSIM, AO DEVOLVERMOS UM NÓ A MEMÓRIA LIVRE, ELE É INSERIDO NA LISTA DE VAZIOS DA PÁGINA QUE O CONTÉM.

MAS AO PEDIRMOS UM NÓ, DEVEMOS FORNECER COMO PARÂMETRO A PÁGINA DE QUE ESTE NÓ DEVE SER RETIRADO, PARA EVITAR A PARTIÇÃO DE UMA CADEIA EM VÁRIAS PÁGINAS. ALGUMAS VEZES, ISTO NÃO PODERÁ SER POSSÍVEL DEVIDO AO FATO DA PÁGINA ESPECIFICADA ESTAR CHEIA. NESTE CASO, SERÁ DADO UM NÓ DE OUTRA PÁGINA.

A IMPLANTAÇÃO DA PAGINAÇÃO DE MEMÓRIA, REQUERERÁ UMA NOVA ALOCAÇÃO PARA A TS, PARA A PILHA DE PARÂMETROS E PARA A PILHA DE RECURSIVIDADE. NA IMPLANTAÇÃO ATUAL, A TS ESTÁ ALOCADA NA MEMÓRIA, ENTREMEADA COM AS CADEIAS. TEMOS 2 OPCOES PARA A NOVA ALOCAÇÃO. UMA É MANTERMOS A TS NA MEMÓRIA, MODIFICANDO SUA ESTRUTURA PARA TABELA SEQUENCIAL E LIMITANDO SEU TAMANHO AO DA MEMÓRIA DISPONÍVEL QUE SOBROU. OUTRA SOLUÇÃO É ALOCARMOS A TS NO DISCO COMO JÁ DISSEMOS ANTERIORMENTE. A PRIMEIRA SOLUÇÃO IMPÕE RESTRIÇÕES AO SISTEMA, JA QUE A TS TERÁ QUE DIVIDIR A MEMÓRIA DISPONÍVEL, COM AS PILHAS DE PARÂMETROS E DE RECURSIVIDADE E COM O PROGRAMA OBJETO. POR ESTE MOTIVO ACONSELHAMOS A ALOCAÇÃO DA TS NO DISCO AO SER IMPLANTADA UMA TÉCNICA DE PAGINAÇÃO DE MEMÓRIA.

CASO SEJA POSSÍVEL A MEMÓRIA, CONTER MAIS DE UMA PÁGINA SIMULTANEAMENTE PODERÍAMOS FAZER UM CONTRÔLE MAIS COMPLEXO DA PAGINAÇÃO E ESPECIFICARMOS PRIORIDADE DE PERMANÊNCIA DE CERTA PÁGINA NA MEMÓRIA. MAS ESTE ASSUNTO, POR SI SÓ, É BEM COMPLEXO E EXTENSO NÃO CABENDO AQUI UMA DISCUSSÃO MAIS DETALHADA DO MESMO.

QUANTO AS PILHAS DE PARÂMETROS E DE RECURSIVIDADE, DEVERÃO SER ALOCADAS NA MEMÓRIA, IMPONDO-SE MAIORES RESTRIÇÕES AO SEU TAMANHO, JÁ QUE SUA ALOCAÇÃO NO DISCO ABAIXARIA BASTANTE O RENDIMENTO DO SISTEMA POIS SÃO FREQUENTEMENTE MANIPULADAS.

4.4- PROGRAMA OBJETO:

JÁ NOS REFERIMOS ANTERIORMENTE À MONTAGEM DO PROGRAMA OBJETO NO DISCO (VER PARÁGRAFO 1.2.1), TORNANDO ASSIM O PROGRAMA OBJETO GERADO COMPATÍVEL COM O SISTEMA OPERACIONAL.

A MONTAGEM DO PROGRAMA OBJETO NO DISCO APRESENTA AINDA A VANTAGEM DE PODERMOS GERAR UM PROGRAMA OBJETO MAIOR, JÁ QUE O MESMO NÃO COMPARTILHARA DA MEMÓRIA COM O COMPILADOR.

OUTRA VANTAGEM, É UMA FLEXIBILIDADE MAIOR NO SISTEMA DE ENTRADA E SAIDA, COMO SERA DISCUTIDO MAIS ADIANTE (PARÁGRAFO 4.5).

MAS, COMO SABEMOS, O PROGRAMA OBJETO SOMENTE NÃO É SUFICIENTE PARA SE EXECUTAR UM PROGRAMA SNOBOL. É NECESSARIO TAMBÉM A TABELA DE SÍMBOLOS.

SE A ALOCAÇÃO DA TS FOR FEITA NA MEMÓRIA, ESTE FATO NÃO APRESENTARÁ GRANDES PROBLEMAS, POIS AO SER CARREGADO O PROGRAMA OBJETO, JUNTO COM ELE SERÁ CARREGADA A TS. MAS SE A TS FOR ALOCADA NO DISCO, HAVERÁ A NECESSIDADE DE SE CRIAR UM ARQUIVO NA FASE COMPILACAO QUE DEVERÁ SER TRANSMITIDO PARA A FASE EXECUÇÃO. NO SISTEMA OPERACIONAL DO 1130, ISTO SÓ É VIÁVEL ATRAVÉS DO CARTÃO *FILES.

DESTA MANEIRA, AO SER CHAMADO O COMPILADOR SNOBOL, O USUÁRIO DEVE FORNECER O ARQUIVO ONDE SERÁ MONTADA A TS. O MESMO PROCEDIMENTO DEVE SER REALIZADO AO EXECUTAR O PROGRAMA OBJETO. ESTE FATO, DIFICULTA O USO DO COMPILADOR POR UM USUÁRIO COMUM.

OUTRO INCONVENIENTE DA MONTAGEM DO PROGRAMA OBJETO NO DISCO, RESIDE NO FATO DE SER BEM COMPLEXA SUA MONTAGEM, EXIGINDO ROTINAS EXTENSAS E DE DIFÍCIL PROGRAMAÇÃO.

HÁ AINDA O INCONVENIENTE DE SE AUMENTAR O TEMPO DE COMPILAÇÃO, EXIGINDO TAMBÉM RELOCAÇÃO E MONTAGEM DO PROGRAMA OBJETO QUE AUMENTA AINDA MAIS O TEMPO TOTAL DE PREPARAÇÃO DO PROGRAMA OBJETO.

POR TODOS ESTES MOTIVOS, DESACONSELHAMOS A MONTAGEM DO PROGRAMA OBJETO NO DISCO, POIS AS VANTAGENS QUE APRESENTA NÃO COMPENSAM O AUMENTO DE COMPLEXIDADE.

A MONTAGEM DO PROGRAMA OBJETO NO DISCO SÓ SE JUSTIFICA SE QUIZERMOS UMA POTENCIALIDADE MAIOR NO SISTEMA DE ENTRADA E SAIDA.

4.5- ENTRADA E SAIDA:

AO ESTENDERMOS A POTENCIALIDADE DO SISTEMA COM RESPEITO A ENTRADA E SAIDA, TEMOS 2 SOLUÇÕES: 1) CARREGAR NA FASE EXECUÇÃO AS SUBROTINAS DE CONVERSÃO E DE E/S PARA TODOS OS PERIFÉRICOS DISPONÍVEIS OU 2) CARREGAR NA FASE EXECUÇÃO, APENAS AS SUBROTINAS USADAS PELO PROGRAMA FONTE.

A PRIMEIRA SOLUÇÃO, EMBORA MAIS FÁCIL DE SER IMPLANTADA, APRESENTA O INCONVENIENTE DE UM GASTO EXCESSIVO E DESNECESSÁRIO DE MEMÓRIA. POR EXEMPLO, AS SUBROTINAS DE E/S E CONVERSÃO PARA ATENDER AOS APARELHOS 2501, 1403, 1142, CONSOLE E 1132 OCUPAM QUASE 2.000 PALAVRAS. POR ESTE MOTIVO, NÃO ACHAMOS VANTAJOSA ESTA SOLUÇÃO.

A SEGUNDA SOLUÇÃO, EXIGE QUE SE FAÇA A RELOCAÇÃO DAS SUBROTINAS A SEREM MONTADAS. ESTE TRABALHO É POR DEMAIS EXTENSO E COMPLEXO, MAS PODE SER FEITO PELO SISTEMA OPERACIONAL, DESDE QUE O PROGRAMA OBJETO SEJA COLOCADO NO DISCO EM FORMATO COMPATÍVEL COM O DO SISTEMA. ESTA SOLUÇÃO É MAIS VIÁVEL QUE A PRIMEIRA, EMBORA ABAIXE O RENDIMENTO DO SISTEMA.

SE HOUVER INTERESSE POR PARTE DOS USUÁRIOS QUE JUSTIFIQUE UMA EXPANSÃO DAS CAPACIDADES DE E/S, EM DETRIMENTO DO TEMPO DE EXECUÇÃO, TORNA-SE VIÁVEL A IMPLANTAÇÃO DOS RECURSOS DESCRITOS A SEGUIR.

O PRIMEIRO PASSO É A ALOCAÇÃO DO PROGRAMA OBJETO NO DISCO, EM FORMATO COMPATÍVEL COM O SISTEMA OPERACIONAL, PARA QUE NÃO TENHAMOS DE RELOCAR AS SUBROTINAS, COMO FOI DITO ACIMA.

DEVEM TAMBÉM SER IMPLANTADAS AS FUNÇÕES DEFINIDAS READ(NOME,APARELHO) E WRITE(NOME,APARELHO), ONDE NOME É O NOME DE UMA VARIÁVEL E APARELHO E O CÓDIGO DO PERIFÉRICO A QUE ESTA VARIÁVEL SERA ASSINALADA.

AO SER COMPILADA UMA FUNÇÃO READ OU WRITE, O TIPO DA VARIÁVEL É MODIFICADO PARA UM TIPO ESPECIAL, PARA QUE A SUBROTINA DE PROCURA NA TS POSSA TOMAR AS PROVIDENCIAS NECESSÁRIAS PARA PERFAZER A E/S. LEMBRE-SE QUE É A SUBROTINA DE PROCURA NA TS QUE FAZ ESTE CONTRÔLE (VER PARÁGRAFO 2.3.3.3).

HAVERÁ, NAS PALAVRAS QUE PRECEDEM O INÍCIO DO PROGRAMA, UM VETOR DE 9 ENTRADAS, CADA UMA ASSOCIADA A UM APARELHO.

AO SER COMPILADA A FUNÇÃO READ OU WRITE, A POSIÇÃO CORRESPONDENTE AO APARELHO É PREENCHIDA COM UMA CHAMADA PARA A SUBROTINA QUE ATENDE AQUELE APARELHO.

POR EXEMPLO, SE ASSINALARMOS A VARIAVEL SAIDA AO APARELHO 1132, A POSIÇÃO CORRESPONDENTE A IMPRESSORA 1132 SERÁ PREENCHIDA COM UMA CHAMADA PARA A SUBROTINA $1132 QUE SERÁ A SUBROTINA DO SISTEMA SNOBOL QUE IRÁ PERFAZER A SAIDA PELA 1132.

AS FUNÇÕES READ E WRITE SERÃO NA VERDADE PSEUDO-FUNÇÕES E SERVIRÃO PARA ASSINALAR UMA VARIÁVEL A UM

APARELHO DE E/S. DIZEMOS QUE SERÃO PSEUDO-FUNÇÕES PORQUE ELAS NÃO GERARÃO CÓDIGO OBJETO ALGUM, SERVINDO APENAS PARA CONTROLE DE E/S NA FASE COMPILAÇÃO. OUTRA RESTRICAO DESTE MÉTODO E QUE A FUNÇÃO QUE DEFINE A VARIÁVEL COMO DE E/S DEVE PRECEDER QUALQUER RFFERÊNCIA A ESTA VARIÁVEL, EMBORA AS FUNCOES READ E WRITE POSSAM APARECER EM QUALQUER LUGAR ONDE SEJA PERMITIDO UMA FUNÇÃO. ESTA RESTRIÇÃO DEVE SER FEITA, POIS PODE JÁ TER SIDO GERADO CÓDIGO TRATANDO ESTA VARIÁVEL COMO UMA VARIÁVEL COMUM.

NESTE MÉTODO, SÃO PREENCHIDOS SOMENTE AS CHAMADAS PARA AS SUBROTINAS QUE ATENDEM OS APARELHOS DEFINIDOS ATRAVÉS DAS FUNÇÕES READ E WRITE. AO SER CARREGADO NA MEMÓRIA PELO SISTEMA, O PROGRAMA OBJETO SÓ CONTERÁ AS SUBROTINAS DESEJADAS.

4.6- REARRANJO DE MEMÓRIA:

EM PROCESSAMENTO DE CADEIAS, MUITAS VEZES, NÃO FAZEMOS UMA OTIMIZAÇÃO DE MEMÓRIA PARA POUPAR TEMPO DE EXECUÇÃO. DESTE MODO, QUANDO A MEMÓRIA ÚTIL ESGOTAR, HÁ AINDA ESPAÇO MAL APROVEITADO. NESTES CASOS, É CHAMADA PARA A MEMÓRIA UM PROGRAMA QUE REARRANJA AS CADEIAS DE MANEIRA MAIS OTIMIZADA E DESTA FORMA CONSEGUE MAIS ALGUM ESPAÇO DISPONÍVEL.

AO IMPLANTARMOS UM PROGRAMA DE REARRANJO DE MEMÓRIA, POUCA COISA PRECISARIA SER MODIFICADA NO SISTEMA ORIGINAL, POIS ESTE PROGRAMA SERIA CHAMADO PELA SUBROTINA DE ERRO AO SER DETETADO UM CÓDIGO DE ESTOURO DE MEMÓRIA. QUANDO O PROGRAMA ENTRASSE NA MEMÓRIA, SALVARIA OS INDICADORES DA ÁREA DE COMUNICAÇÃO E AS PARTES DA MEMÓRIA QUE DEVERIAM SER RESTAURADAS PARA QUE A EXECUÇÃO PUDESSE SER REINICIADA, DO PONTO EM QUE PAROU, AO SER COMPLETADO O REARRANJO DA MEMÓRIA.

A MANEIRA MAIS INTUITIVA DE SE CONSEGUIR ALGUM ESPAÇO LIVRE É RECOPIAR TODAS AS CADEIAS RETIRANDO OS CARACTERES VAZIOS QUE POR ACASO CONTENHAM. ESTES ESPAÇOS VAZIOS SÃO INSERIDOS PARA COMPLETAR UM NÓ QUANDO REDEFINIMOS A PARTE DE UMA CADEIA QUE CASOU COM O PADRÃO (VER PARÁGRAFO 3.3.7).

ESTA MANEIRA DE REARRANJO, DEPENDENDO DO PROGRAMA, PODE NOS FORNECER MUITO POUCO ESPAÇO DISPONÍVEL, POIS PODEM HAVER POUCOS ESPAÇOS VAZIOS NO MEIO DE UMA CADEIA. A GRANDE DESVANTAGEM QUE APRESENTA É DE NÃO PODERMOS ESPECIFICAR DE ANTE-MÃO QUANTO ESPAÇO GANHAREMOS. ENTÃO, ALGUMAS VEZES, MUITO TEMPO DE COMPUTAÇÃO SERIA GASTO ANTES DE DESCOBRIRMOS QUE NÃO PUDEMOS CONSEGUIR ALGUM ESPAÇO LIVRE.

OUTRA MANEIRA DE SE REALIZAR O REARRANJO DA MEMÓRIA

É RECOPIARMOS AS CADEIAS AUMENTANDO O TAMANHO DO NÓ. AO FAZERMOS ISTO, AUMENTAMOS A MEMÓRIA ÚTIL POIS MENOS ESPAÇOS SERIA GASTO EM APONTADORES. POR EXEMPLO, SE AUMENTARMOS O TAMANHO DO NÓ DE 4 PARA 6 PALAVRAS, GANHAMOS 8% EM MEMÓRIA UTILIZÁVEL.

NESTE ESQUEMA, DEVEMOS TER NA ÁREA DE COMUNICAÇÃO UMA PALAVRA QUE CONTÉM O TAMANHO DO NÓ PARA QUE AS ROTINAS QUE MANIPULAM CARACTERES E APONTADORES SAIBAM QUAL O TAMANHO DO NÓ QUE ESTÁ SENDO USADO.

PARTE V

CONCLUSÕES

5.1 - INTRODUÇÃO:

COM A IMPLANTAÇÃO DO SISTEMA, PUDEMOS VERIFICAR QUE ALGUMAS PARTES DO MESMO PODERIAM SER MELHORADAS, POIS SEUS DESEMPENHOS NÃO FORAM SATISFATÓRIOS.

NESTA PARTE, TECEREMOS ALGUMAS CONSIDERAÇÕES SOBRE ESTES PONTOS. SERÃO TAMBÉM DISCUTIDOS CERTOS ASPECTOS DO DESEMPENHO DO SISTEMA.

5.2 - CÓDIGO ALEATÓRIO:

AO REALIZARMOS A DEPURAÇÃO DO SISTEMA, NOTAMOS QUE HÁ TENDÊNCIAS NO CÓDIGO ALEATÓRIO.

QUANDO O NOME DE UM ELEMENTO FOR COMPOSTO DE APENAS UM SÍMBOLO QUE SEJA LETRA OU DÍGITO, O SEU CODIGO ALEATÓRIO SERÁ 2.

EMBORA SEJAM PERMITIDOS QUANTOS CARACTERES DESEJARMOS NO NOME DE UM ELEMENTO, É UMA PRATICA COMUM EM CERTOS PROGRAMADORES, DAR A VARIÁVEIS NOMES CURTOS DE UMA E DUAS LETRAS. ISTO FAZ COM QUE HAJA UM ACÚMULO MAIOR DE ELEMENTOS EM UMA PARTE DA TS.

AO ESCOLHERMOS O ALGORÍTMO DO CÓDIGO ALEATÓRIO, FIZEMOS UM TESTE SUMÁRIO, USANDO APROXIMADAMENTE 150 NOMES DE VARIAVEIS DEFINIDAS POR USUÁRIOS. A DISTRIBUIÇÃO SE MOSTROU RAZOAVELMENTE DISTRIBUIDA, TALVEZ DEVIDO AO GRANDE NÚMERO DE NOMES DADOS POR USUÁRIOS DIFERENTES. HÁ ENTRETANTO CERTOS USUÁRIOS QUE PREFEREM VARIÁVEIS MENORES DE APENAS 1 OU 2 LETRAS.

DEVIDO A MODULARIDADE DO SISTEMA, TORNA-SE FÁCIL MUDARMOS O ALGORÍTMO DE DETERMINAÇÃO DO CÓDIGO ALEÁTORIO, MAS ACHAMOS QUE ESTA MUDANÇA NÃO É IMPERIOSA POIS AUMENTARIA POUCO O RENDIMENTO DO SISTEMA. ISTO PORQUE A VELOCIDADE DE PROCURA NA TS É PEQUENA COMPARADA COM A DE OUTRAS FACILIDADES DO SISTEMA COMO VARREDURA DE PADRÃO E CONCATENAÇÃO, E ALÉM DISTO, ATINGIRIA APENAS DETERMINADO TIPO DE USUÁRIO.

5.3 - MODULARIDADE:

AO ESCREVERMOS OS PROGRAMAS CONSTITUINTES DO SISTEMA, MUITAS VEZES DEMOS MAIOR PRIORIDADE A FACILIDADE DE PROGRAMAÇÃO DO QUE À MODULARIDADE.

ISTO FOI FEITO POR FALTA DE EXPERIÊNCIA NOSSA NA CONSTRUÇÃO DE UM SISTEMA DESTE TIPO E MOSTROU-SE UMA MÁ FILOSOFIA POIS O SISTEMA PERDERIA MUITO EM GENERALIDADE, FUNCIONANDO APENAS COM A CONFIGURACAO DO 1130 DA COPPE E EXIGIU VARIAS MODIFICACOES PARA GENERALIZAR O SISTEMA.

ESTE INCONVENIENTE APARECEU EM DOIS PONTOS: ENTRADA E SAIDA E MANIPULAÇÃO DE APONTADORES.

NA SAIDA DE DADOS, REALIZAMOS A SAIDA DIRETAMENTE DA PRÓPRIA SUBROTINA, FAZENDO ELA PRÓPRIA A CONVERSÃO PARA O CÓDIGO DA IMPRESSORA 1403. COMO SÃO DUAS AS SUBROTINAS QUE REALIZAM SAIDA, PARA FUNCIONARMOS EM UMA CONFIGURAÇÃO QUE SÓ TENHA A IMPRESSORA 1132, NECESSITAMOS MODIFICAR AS DUAS SUBROTINAS. UMA SOLUÇÃO MAIS RACIONAL, FOI FAZER UMA SÓ SUBROTINA DE SAIDA, USADA PELAS OUTRAS, QUE USA O CÓDIGO EBCDIC E MODIFICAR APENAS ESTA SUBROTINA PARA TROCAR A IMPRESSORA DO SISTEMA.

ESTE PROBLEMA NA ENTRADA NÃO FOI CRÍTICO POIS OS CÓDIGOS DE ENTRADA DAS LEITORAS 1442,2501 E DO TECLADO DO CONSOLE SÃO IDÊNTICOS. O PROBLEMA SURGIRIA SE QUIZESSEMOS REALIZAR A ENTRADA ATRAVÉS DA LEITORA DE FITA DE PAPEL 1134.

QUANTO AOS APONTADORES, EMBORA TENHAMOS UMA SUBROTINA QUE MANIPULA COM OS MESMOS, MUITAS VEZES, PARA SIMPLIFICAR A PROGRAMAÇÃO, ACHAMOS DIRETAMENTE O CAMPO DO APONTADOR DE UM NÓ. ISTO APRESENTA O INCONVENIENTE DE TERMOS O SISTEMA DEPENDENTE DO TAMANHO DO NÓ, NÃO SENDO POSSÍVEL A IMEDIATA IMPLANTAÇÃO DA TÉCNICA DE REARRANJO DA MEMÓRIA DESCRITA NO PARÁGRAFO 4.5.

COMO ESTA FALHA SÓ TEM CONSEQUÊNCIAS PARA A IMPLANTAÇÃO DO REARRANJO DE MEMÓRIA, DEIXAMOS PARA REALIZAR AS MODIFICAÇÕES DEVIDAS AO IMPLANTARMOS A TÉCNICA DE REARRANJO DA MEMÓRIA.

5.4 - CARTÕES DE CONTINUAÇÃO:

O SISTEMA UTILIZADO PARA CONTRÔLE DOS CARTÕES DE CONTINUAÇÃO MOSTROU-SE MUITO COMPLEXO E DE DIFÍCIL PROGRAMAÇÃO E DEPURAÇÃO, DEVIDO A ROTINA QUE TERMINA A DECLARAÇÃO ANTERIOR.

UM SISTEMA MAIS FÁCIL CONSISTIRIA EM AO DETETARMOS O FINAL DE UM CARTÃO, TESTARMOS O PRIMEIRO CARACTER DO PRÓXIMO CARTÃO, QUE NESTE PONTO JÁ ESTARIA NA MEMÓRIA, POIS O SISTEMA ADMITE A TÉCNICA DE ÁREA DUPLA PARA LEITURA. SE O PRIMEIRO CARACTER FÔSSE UM PONTO, ESTE CARTÃO SERIA EDITADO E A COMPILAÇÃO CONTINUARIA DE ONDE PAROU. ESTE SISTEMA FAZ COM QUE SEJAM SUPRIMIDAS DUAS ROTINAS DO ATUAL SISTEMA: A QUE IDENTIFICA ONDE A COMPILAÇÃO ANTERIOR TERMINOU E A QUE TERMINA A DECLARAÇÃO ANTERIOR.

EMBORA A MODIFICAÇÃO PARA ESTE NOVO METODO ACARRETE UM GANHO DE MEMÓRIA, IMPLICA EM REESCREVER TODO O CORPO PRINCIPAL DO COMPILADOR, O QUE EM NOSSA OPINIÃO NÃO COMPENSARIA, A NAO SER QUE SE FAÇA PREMENTE UM ESPAÇO MAIOR DE MEMÓRIA DISPONÍVEL.

5.5 - COMPILADOR X INTERPRETADOR:

O NOSSO PROPÓSITO AO REALIZAR ESTE TRABALHO, FOI O DE IMPLANTAR UM COMPILADOR SNOBOL, ESTANDO PORTANTO FORA DE DISCUSSAO A VIABILIDADE DE IMPLANTARMOS UM INTERPRETADOR AO INVÉS DE COMPILADOR.

A NOSSA ESCOLHA FOI BASEADA SOMENTE EM UM INTERESSE PESSOAL POR ESTA ÁREA DE PESQUISA.

MAS A ESCOLHA NOS PARECEU ACERTADA POIS, SNOBOL E UMA LINGUAGEM INERENTEMENTE LENTA E UM INTERPRETADOR TEM SEMPRE O RENDIMENTO MENOR QUE UM COMPILADOR, DEVIDO AO FATO DE SER NECESSÁRIA UMA TRADUÇÃO NA HORA DA EXECUÇÃO.

5.6 - DESEMPENHO DO SISTEMA:

5.6.1 - CAPACIDADE:

O PROGRAMA USADO PARA MEDIRMOS A CAPACIDADE DE ARMAZENAMENTO DE CARACTERES DO SISTEMA, FOI O PROGRAMA TESTE 2 (FREQUÊNCIA DAS LETRAS) APRESENTADO NO APÊNDICE II.

O TESTE FOI REALIZADO COM UM COMPUTADOR 1130 DE 32K DE MEMÓRIA. CONSISTIU EM COLOCAR COMO DADOS DO PROGRAMA TESTE UMA QUANTIDADE GRANDE DE CARTÕES E VERIFICARMOS QUANTOS CARTÕES FORAM LIDOS ANTES DE DAR ESTOURO DE MEMÓRIA.

COMO PODEMOS VERIFICAR, O PROGRAMA TESTE CONSISTE DE DUAS PARTES: 1) LEITURA DE DADOS E FORMACAO DE UMA CADEIA (TEXTO) QUE CONTEM OS CARTÕES JA LIDOS E 2) CÁLCULO DA FREQUÊNCIA DAS LETRAS. PARA O TESTE DE CAPACIDADE, APENAS A PRIMEIRA PARTE NOS INTERESSA.

O PROGRAMA TESTE NÃO FOI FEITO VISANDO UM APROVEITAMENTO MÁXIMO DA CAPACIDADE, POIS PARA JUNTAR CADA LINHA NA VARIAVEL TEXTO, O FAZ PELA DECLARACAO:

```
TEXTO = TEXTO LINHA
```

COMO SABEMOS, A CONCATENAÇÃO DE LINHA COM TEXTO E

REALIZADA EM UMA VARIÁVEL INTERMEDIÁRIA ANTES DE SUBSTITUIR A VARIÁVEL TEXTO. DESTE MODO, AO CRESCERMOS A VARIÁVEL TEXTO INDEFINIDAMENTE, CAUSAREMOS UM ESTOURO DE MEMÓRIA QUANDO TEXTO TIVER APROXIMADAMENTE A METADE DA CAPACIDADE TOTAL.

NO TESTE DE CAPACIDADE, FORAM LIDOS 248 CARTÕES ANTES DE HAVER UM ESTOURO DE MEMÓRIA, DANDO PARA A VARIÁVEL TEXTO UM TAMANHO DE 19.840 CARACTERES. E PELOS MOTIVOS EXPOSTOS ACIMA, AVALIAMOS A CAPACIDADE TOTAL DA MEMÓRIA EM APROXIMADAMENTE 40.000 CARACTERES, O QUE NOS PARECE BEM RAZOÁVEL.

5.6.2 - VELOCIDADE:

O MAIOR PROBLEMA QUE TIVEMOS, PARA AVALIAR O DESEMPENHO DO SISTEMA QUANTO A VELOCIDADE, FOI O DE NÃO TERMOS UM OUTRO SISTEMA SEMELHANTE PARA A COMPARAÇÃO.

O UNICO SISTEMA SNOBOL QUE TIVEMOS ACESSO FOI UM INTERPRETADOR REALIZADO PARA O COMPUTADOR IBM 1620. PARA DIMINUIRMOS O EFEITO DA EFICIÊNCIA DOS COMPUTADORES, LEVAMOS EM CONTA O DESEMPENHO DOS MESMOS EM UM TESTE REALIZADO PELA AUERBACH CO. (11). NESTE TESTE, O 1130 MOSTROU SER CERCA DE 3,5 VEZES MAIS VELOZ QUE O 1620 PARA PROCESSAMENTOS EM QUE HA PREDOMINÂNCIA DE PROCESSAMENTO SOBRE ENTRADA E SAIDA.

O TESTE DE DESEMPENHO DO SISTEMA SNOBOL, FOI TAMBÉM O TESTE 2 DO APENDICE II QUE E UM PROGRAMA EM QUE HÁ PREDOMINÂNCIA DA COMPUTAÇÃO SOBRE A E/S. PARA PADRONIZAR OS DADOS, FORAM COLOCADOS 20 CARTÕES EM BRANCO NOS 2 TESTES REALIZADOS.

O RESULTADO OBTIDO NO 1130 FOI:

INICIALIZAÇÃO	10 SEG.
COMPILAÇÃO E LISTAGEM	15 SEG.
INIC. FASE EXECUÇÃO	3 SEG.
LEITURA DOS DADOS	29 SEG.
CÁLCULO	50 SEG.
IMPRESSÃO DA RESPOSTA	5 SEG.
TOTAL:	1 MIN 52 SEG.

O RESULTADO OBTIDO NO 1620 FOI:

INICIALIZAÇÃO	4 SEG.
LEITURA E LISTAGEM	9 SEG.
LEITURA DOS DADOS	21 SEG.
CÁLCULO	475 SEG.

IMPRESSÃO DA RESPOSTA --------- 6 SEG.
TOTAL: ---------- 8 MIN 35 SEG.

SE CONSIDERARMOS A TAREFA INTEIRA, O SISTEMA SNOBOL 1130 E CERCA DE 30% MAIS VELOZ QUE O 1620. MAS DEVEMOS NOTAR QUE SE A TAREFA FOSSE MENOR, É POSSIVEL QUE O SNOBOL 1130 SEJA INCLUSIVE MAIS LENTO QUE O SISTEMA SIMILAR DO 1620, DEVIDO AO FATO DE ESTE NÃO TER A FASE COMPILAÇÃO.

SE CONSIDERARMOS APENAS O TEMPO DE CÁLCULO, VERIFICAMOS QUE O SISTEMA SNOBOL 1130 E CERCA DE 170% MAIS VELOZ QUE O SEMELHANTE 1620.

ESTAS PERCENTAGENS FORAM CALCULADAS PELA FÓRMULA:

PERC = 1 - T(1620) / (T(1130) * 3,5)

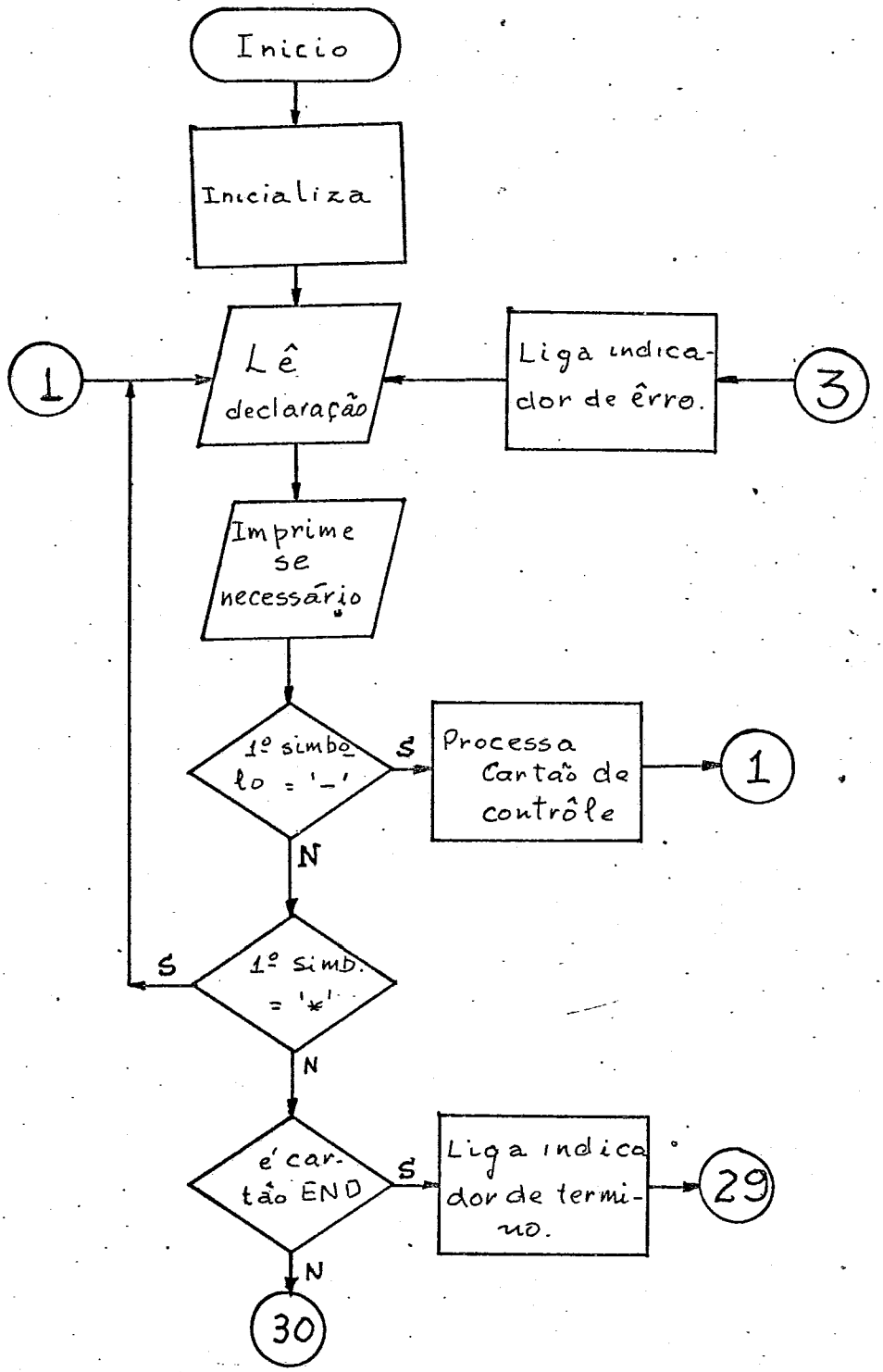
PELOS TEMPOS OBTIDOS, PODEMOS NOTAR QUE, EMBORA OS APARELHOS DE E/S DO 1130 EM QUE FOI REALIZADO O TESTE SEJAM MAIS VELOZES QUE OS DO 1620, OS TEMPOS DE E/S SÃO APROXIMADAMENTE EQUIVALENTES. ISTO E DEVIDO AO FATO DE NÃO USARMOS ÁREA DUPLA DE E/S PARA O SISTEMA 1130.

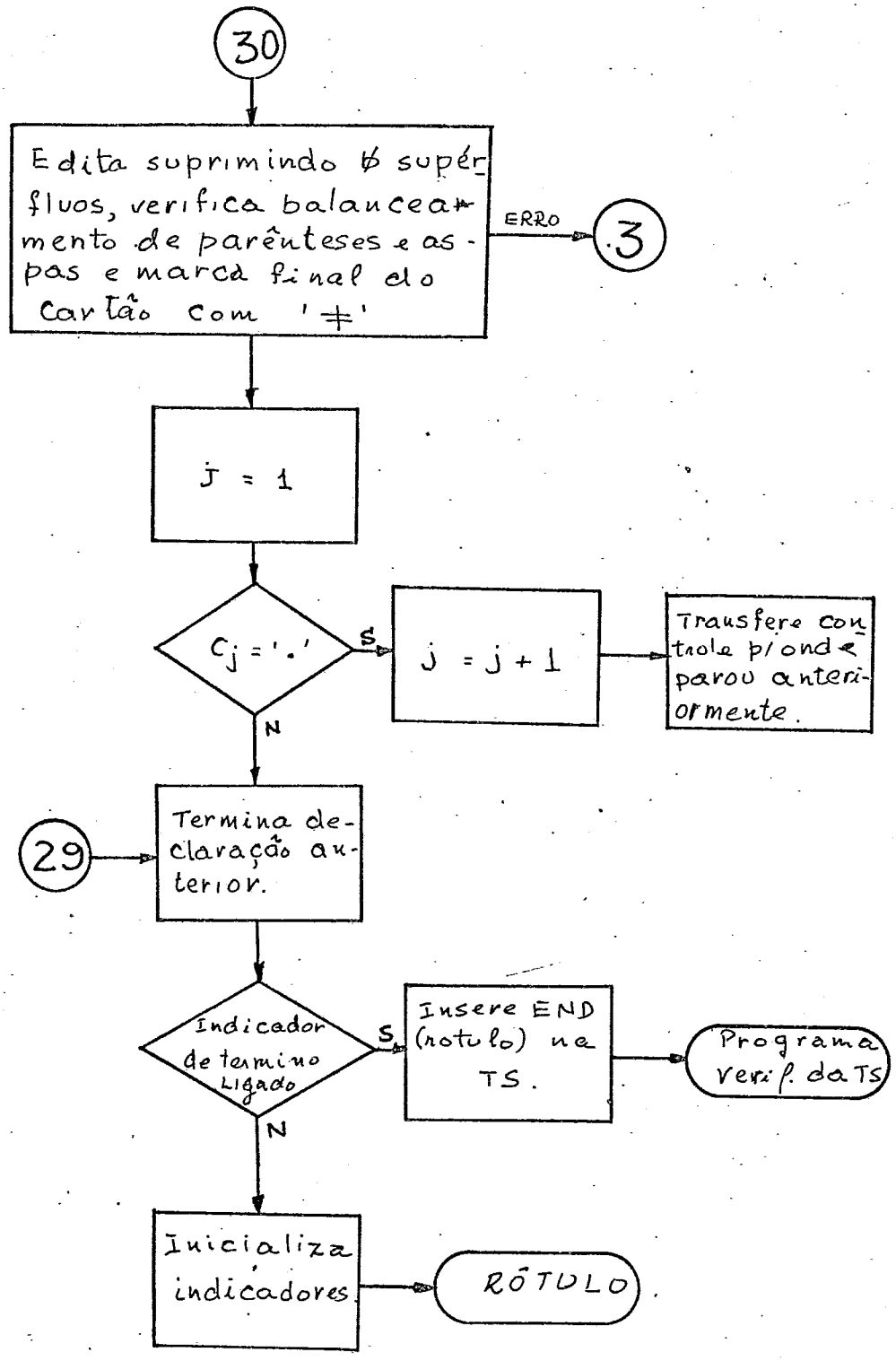
EM RESUMO, O RENDIMENTO DO SISTEMA SNOBOL 1130 E MUITO BOM COMPARANDO COM O SEMELHANTE DO COMPUTADOR 1620.

APÊNDICE I

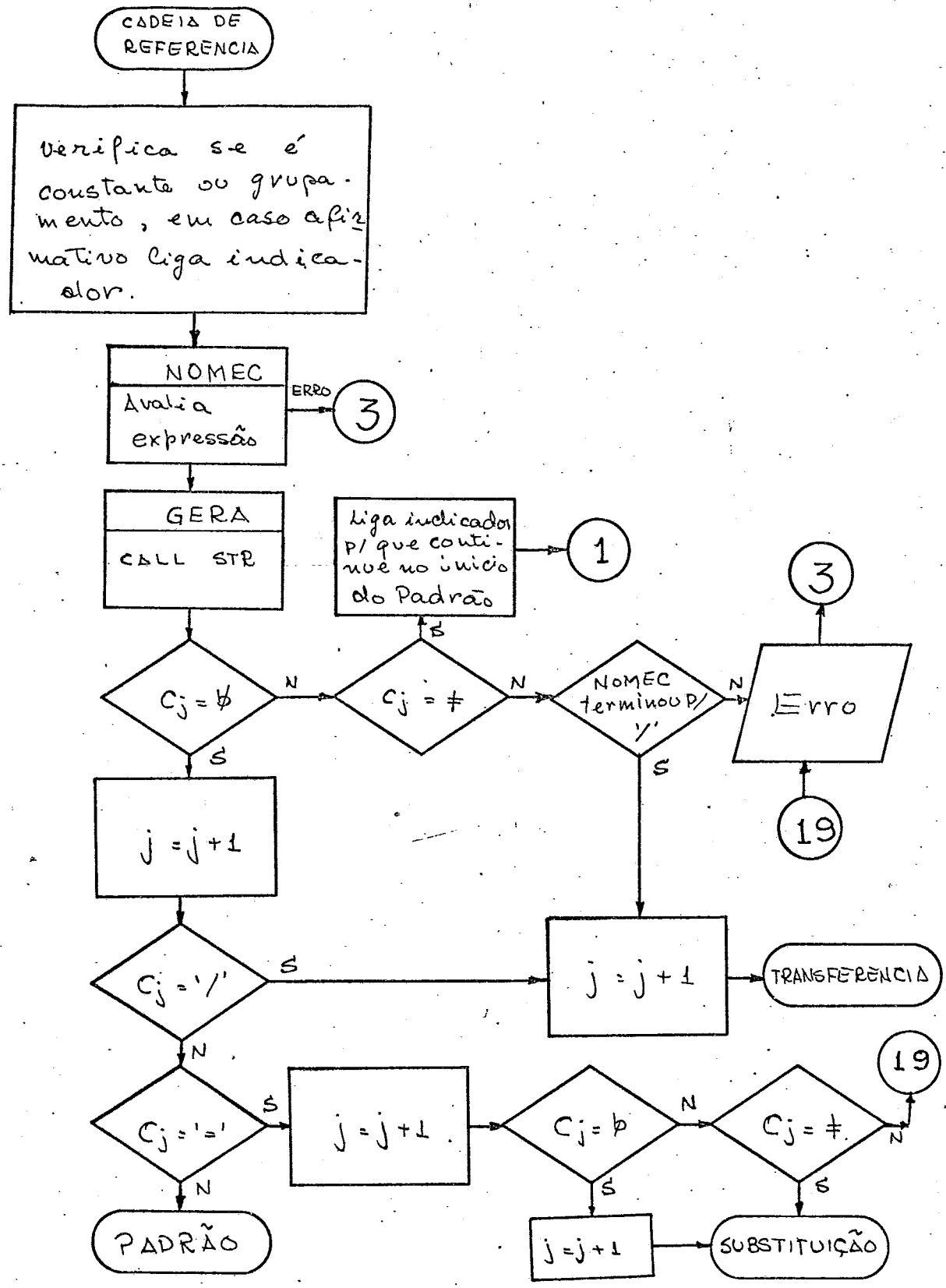
DIAGRAMAS DE BLOCO

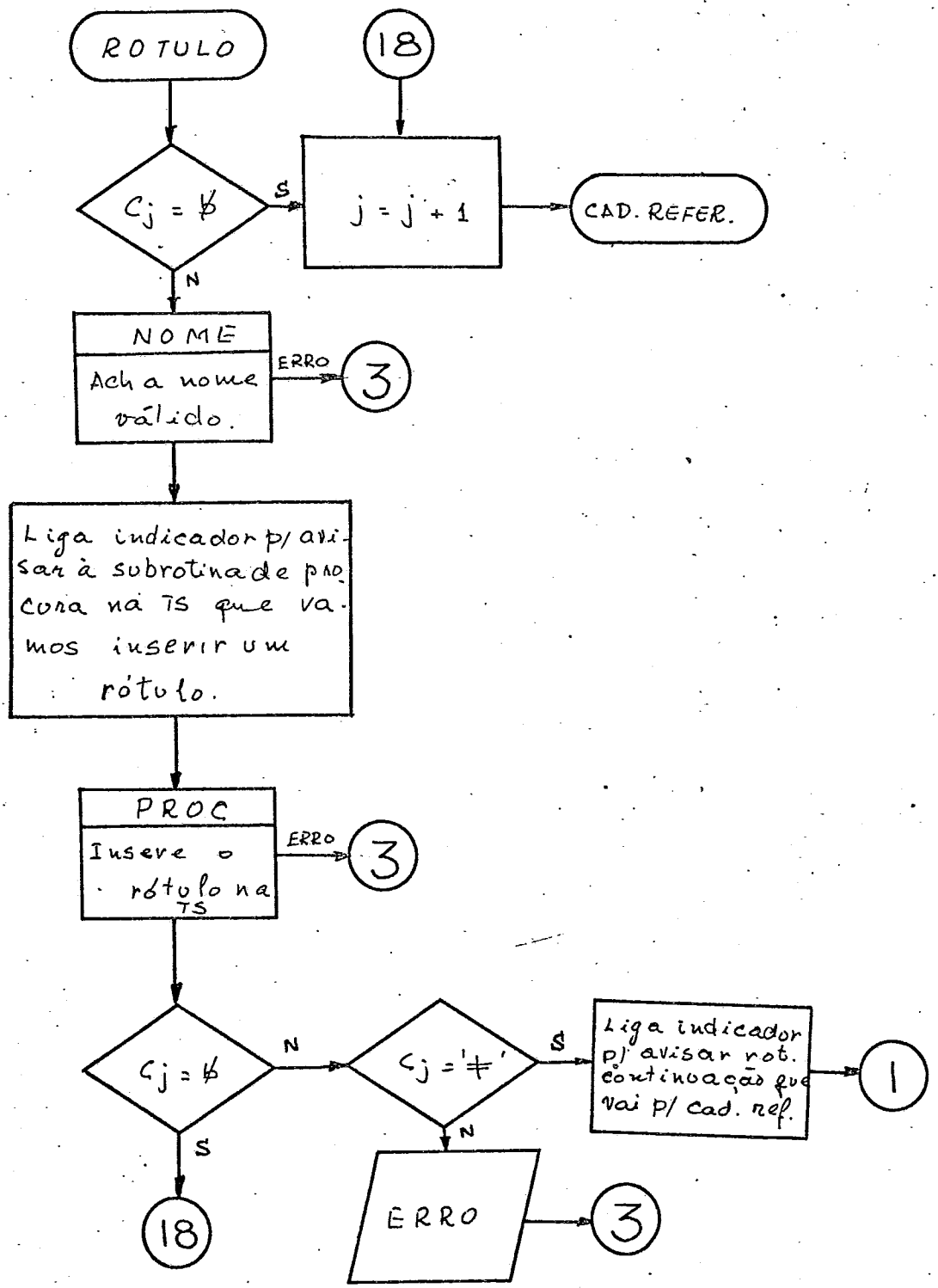
CORPO PRINCIPAL DO COMPILADOR

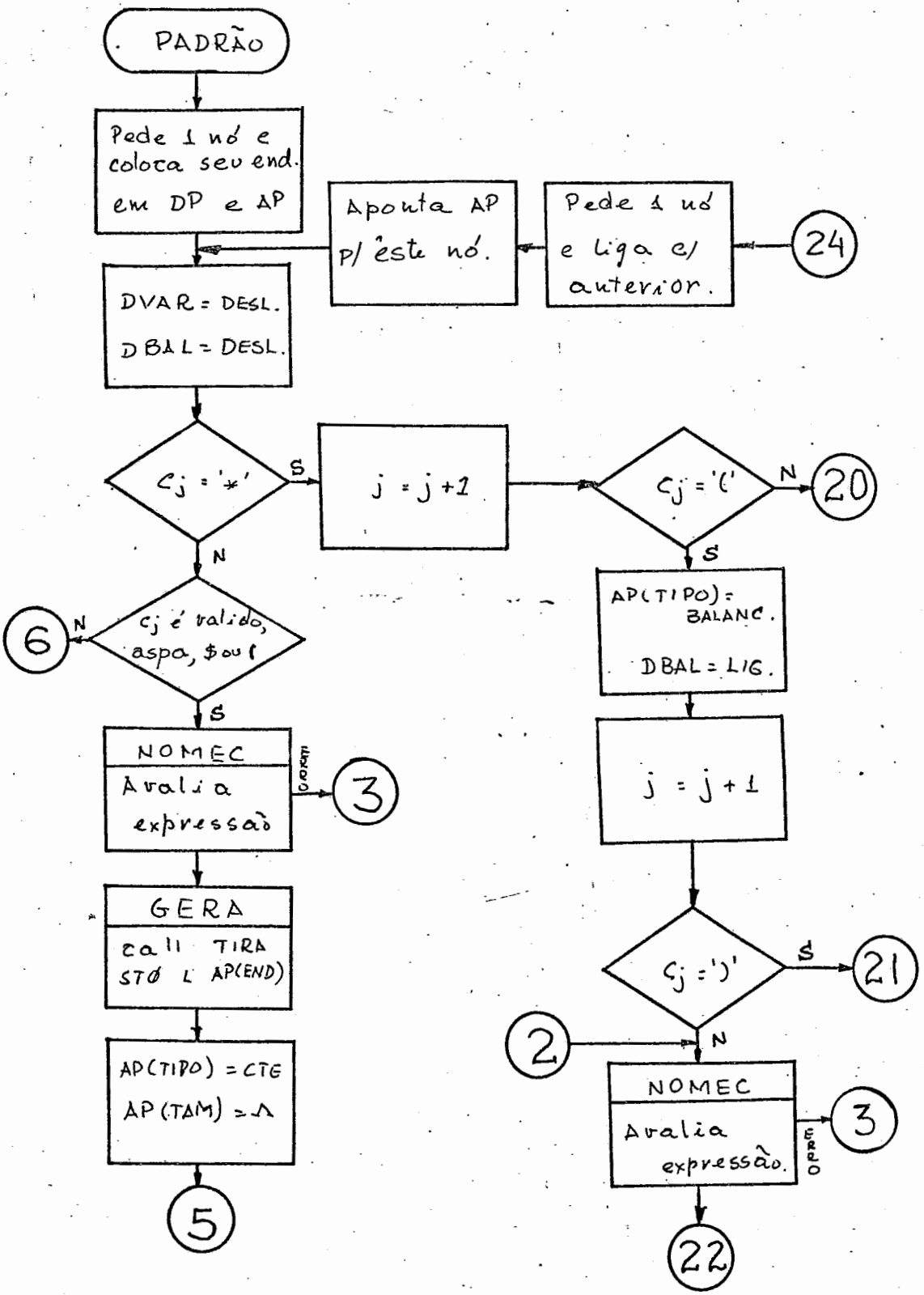


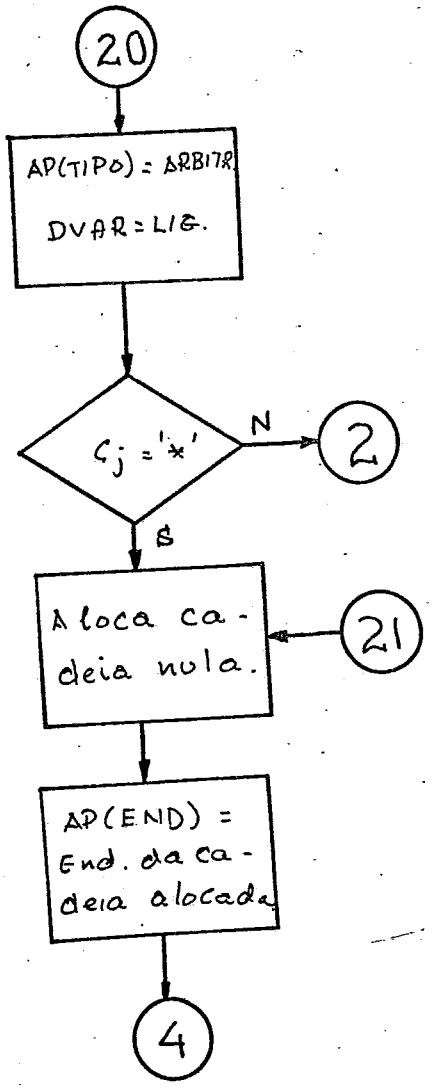


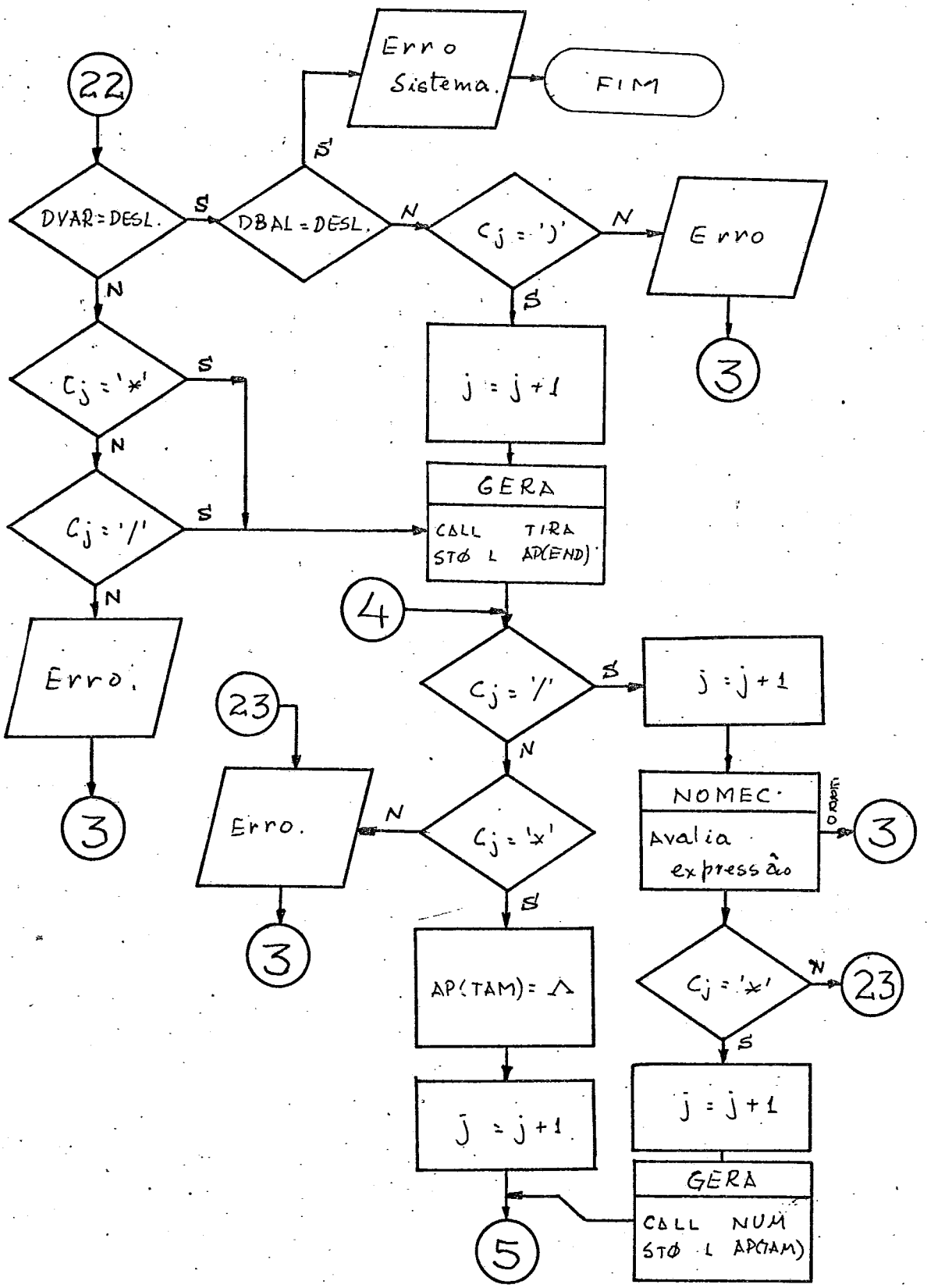
NOTA: o vetor C contem o cartão.

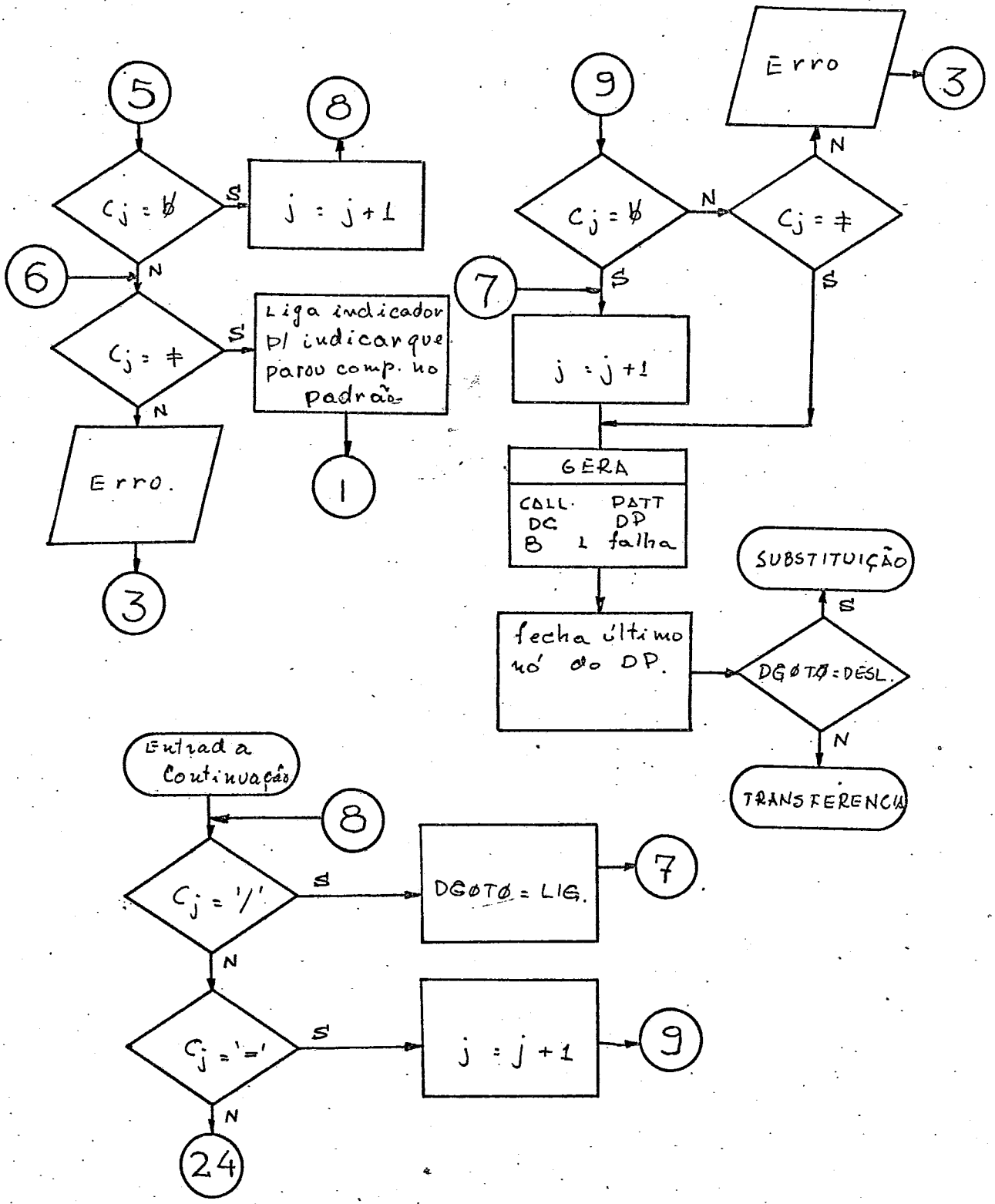

CADEIA DE REFERENCIA
verifica se é constante ou grupamento, em caso afirmativo liga indicador.
NOMEC
Avalia expressão
ERRO
3
GERA
CALL STR
Liga indicador p/ que continue no inicio do Padrão
1
Cj = ϕ
N
Cj = ≠
N
NOMEC terminou p/ ')'
N
Erro
3
19
S
S
S
j = j + 1
j = j + 1
TRANSFERENCIA
Cj = '/'
S
N
Cj = '='
S
j = j + 1
Cj = ϕ
N
Cj = ≠
N
19
N
S
S
PADRÃO
j = j + 1
SUBSTITUIÇÃO

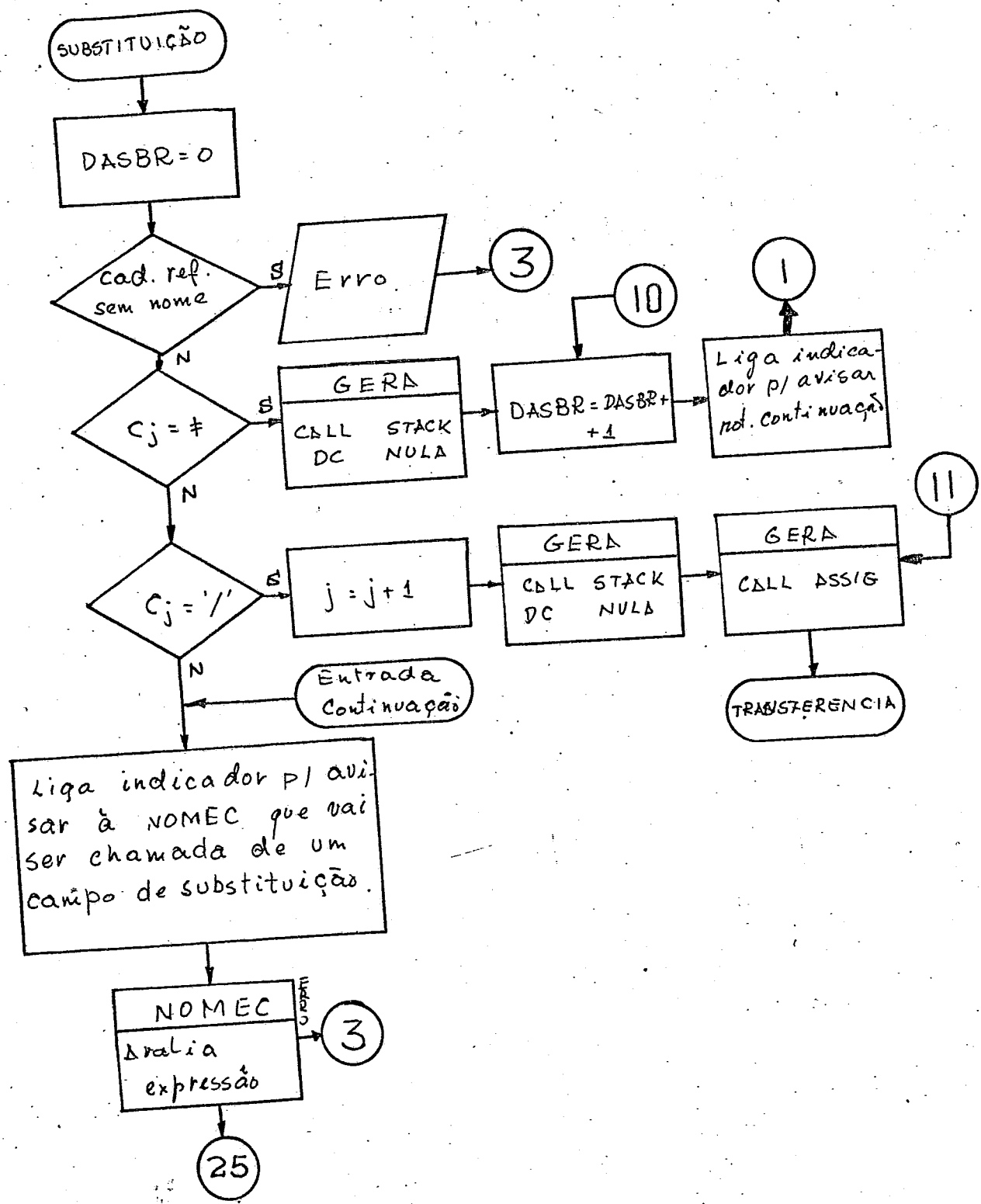

ROTULO
18
C_j = ∅
S
j = j + 1
CAD. REFER.
N
NOME
Acha nome válido.
ERRO
3
Liga indicador p/ avisar à subrotina de proc. na TS que vamos inserir um rótulo.
PROC
Insere o rótulo na TS
ERRO
3
C_j = ∅
N
C_j = '+'
S
Liga indicador p/ avisar rot. continuação que vai p/ cad. ref.
1
S
18
N
ERRO
3

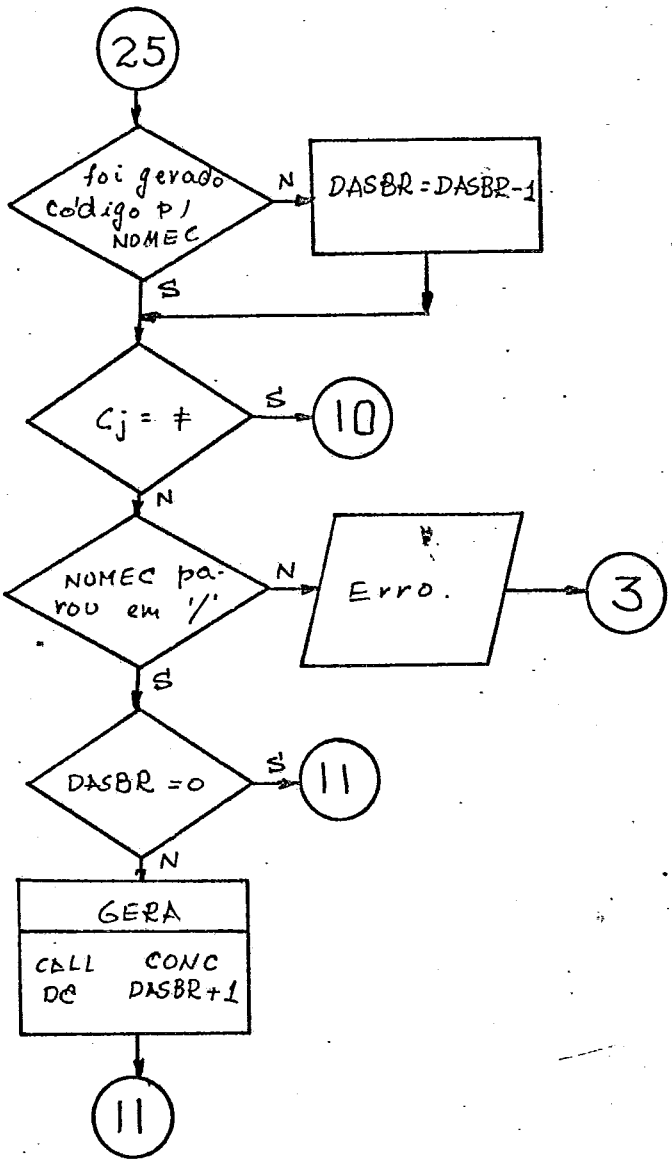

PADRÃO
Pede 1 nó e coloca seu end. em DP e AP
Aponta AP p/ êste nó.
Pede 1 nó e liga c/ anterior.
24
DVAR = DESL.
DBAL = DESL.
Cj = '*'
S
j = j+1
Cj = '('
N
20
S
N
Cj é valido, aspa, $ ou (
N
6
S
AP(TIPO) = BALANC.
DBAL = LIG.
NOMEC
Avalia expressão
Erro
3
j = j+1
GERA
call TIRA STØ L AP(END)
Cj = ')'
S
21
AP(TIPO) = CTE
AP(TAM) = Λ
2
N
NOMEC
Avalia expressão.
Erro
3
5
22

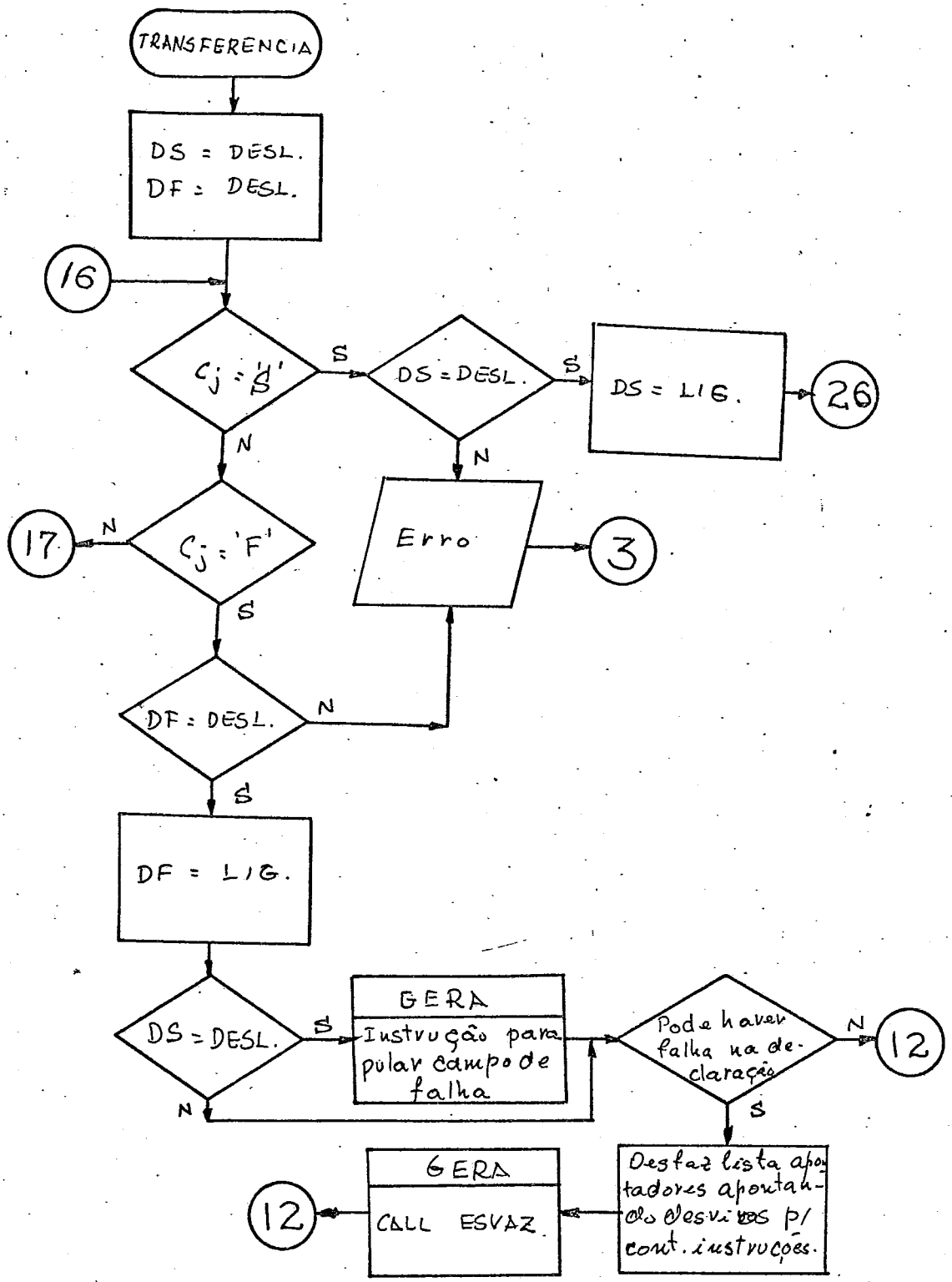

20
AP(TIPO) = ARBITR
DVAR = LIG.
Cj = '*'
N
2
S
Aloca ca-deia nula.
21
AP(END) = End. da ca-deia alocada
4

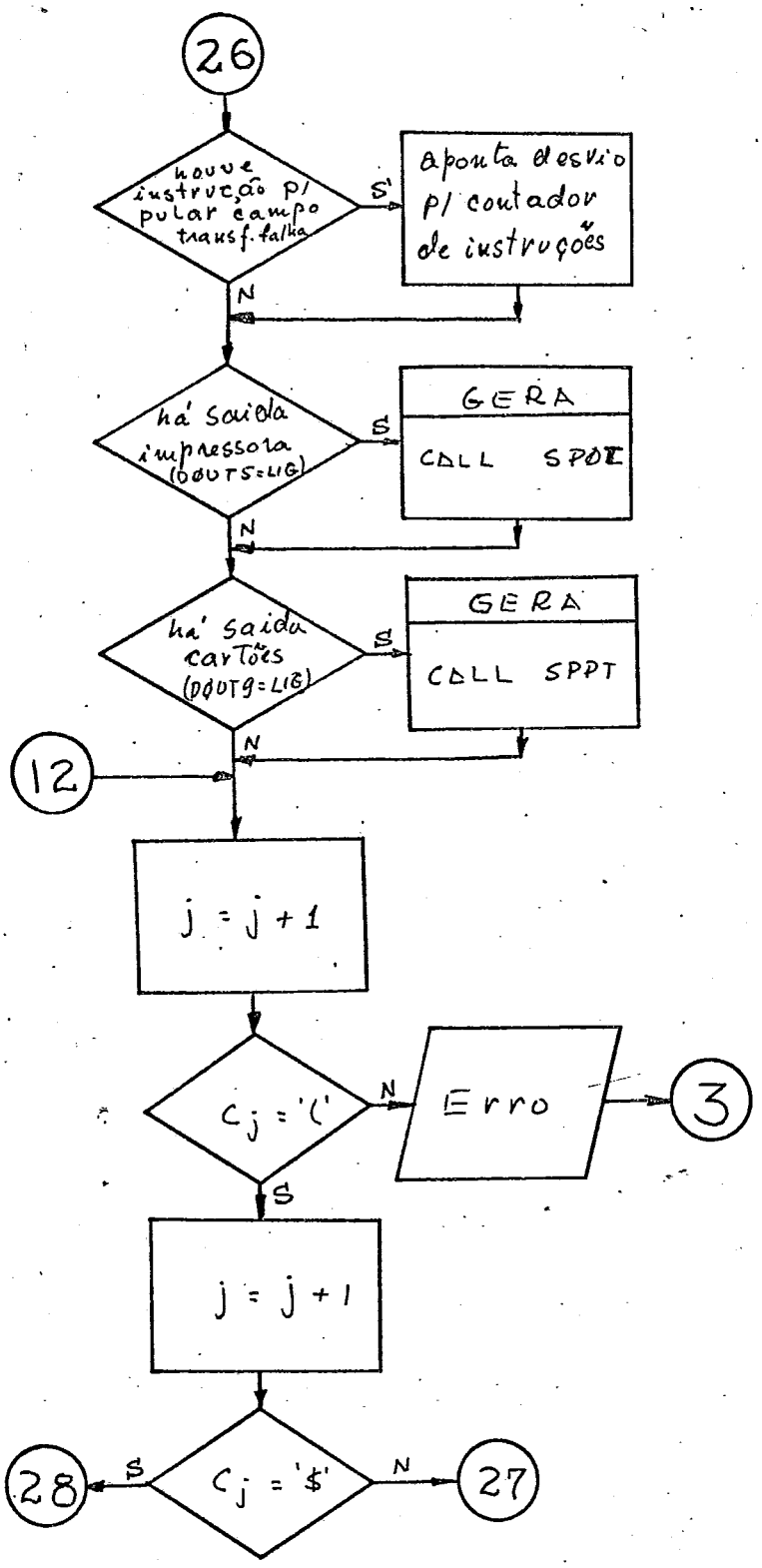

22
Erro Sistema.
FIM
S
DVAR=DESL.
S
DBAL=DESL.
N
Cj = ')'
N
Erro
S
3
N
Cj = '*'
S
j = j+1
N
Cj = '/'
S
GERA
CALL TIRA
STØ L AP(END)
N
4
Erro.
3
23
Cj = '/'
S
j = j+1
N
NOMEC
Avalia expressão
variável
3
Erro.
N
Cj = 'x'
3
S
AP(TAM) = Λ
Cj = 'x'
N
23
S
j = j+1
j = j+1
GERA
CALL NUM
STØ L AP(TAM)
5

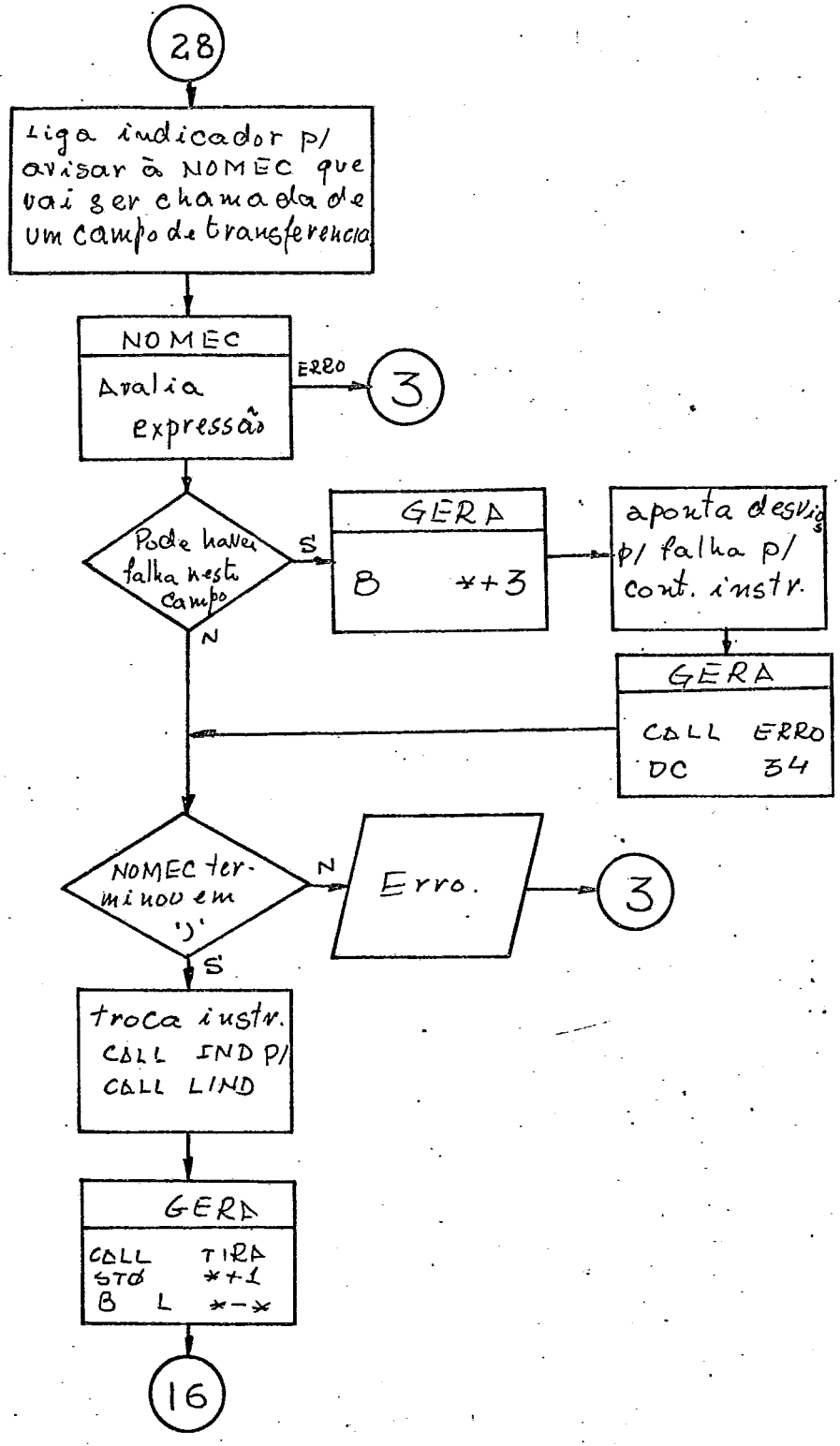

5
8
$C_j = \not b$
S
j = j+1
6
N
$C_j = \neq$
S
Liga indicador p/ indicar que parou comp. no padrão
N
Erro.
3
1
9
Erro
3
$C_j = \not b$
N
$C_j = \neq$
N
S
S
7
S
j = j+1
GERA
CALL DC B
PATT DP 1 falha
fecha último nó do DP.
DGØTØ = DESL.
S
SUBSTITUIÇÃO
N
TRANSFERENCIA
Entrada a Continuação
8
$C_j = '/'$
S
DGØTØ = LIG.
7
N
$C_j = '='$
S
j = j+1
9
N
24

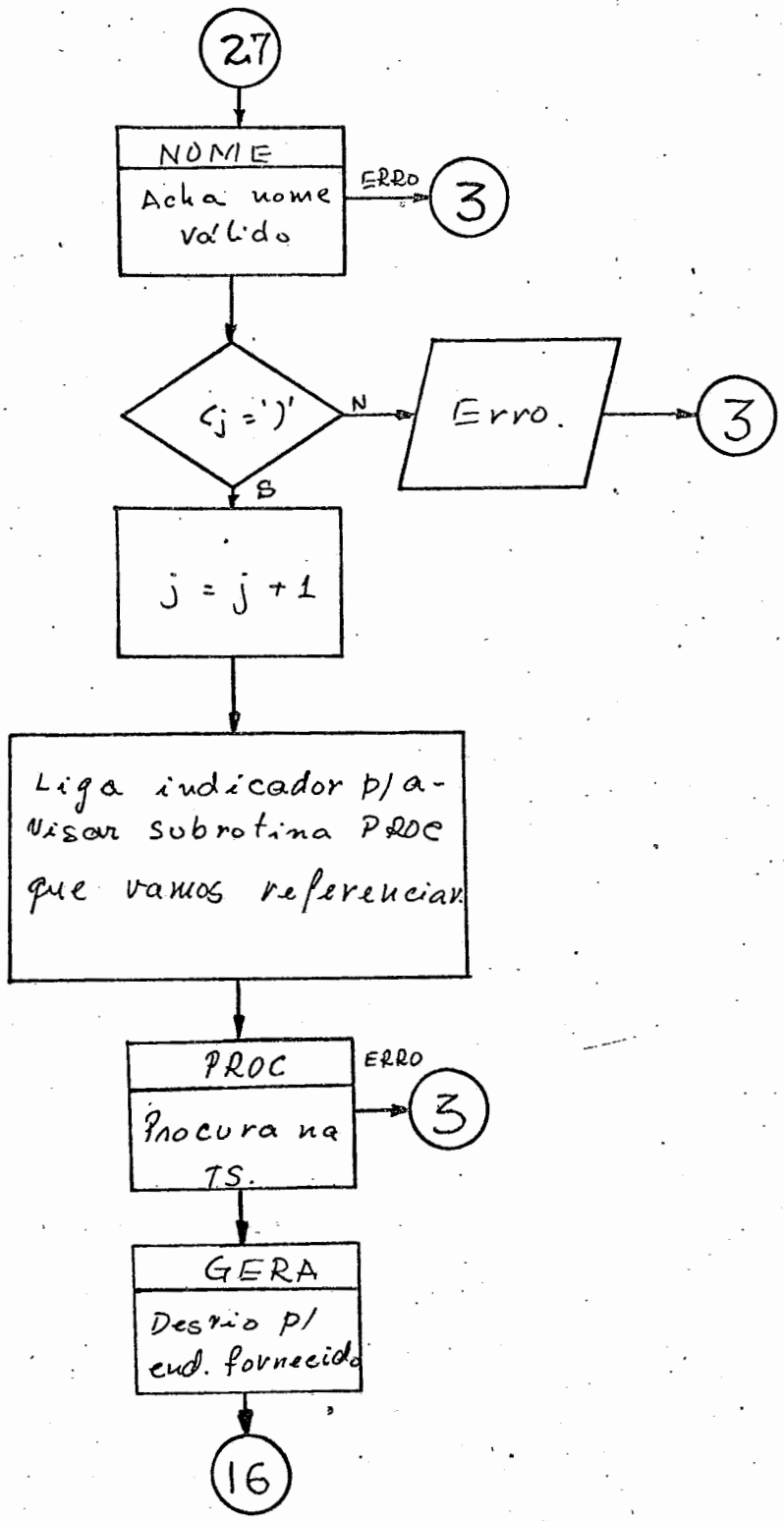

SUBSTITUIÇÃO
DASBR=0
cad. ref. sem nome
S
Erro
3
N
Cj = ≠
S
GERA
CALL STACK
DC NULA
DASBR = DASBR + +1
10
Liga indicador p/ avisar rot. continuação
1
N
Cj = '/'
S
j = j+1
GERA
CALL STACK
DC NULA
GERA
CALL ASSIG
11
N
Entrada Continuação
TRANSFERENCIA
Liga indicador p/ avisar à NOMEC que vai ser chamada de um campo de substituição.
NOMEC
Analia expressão
ERRO
3
25

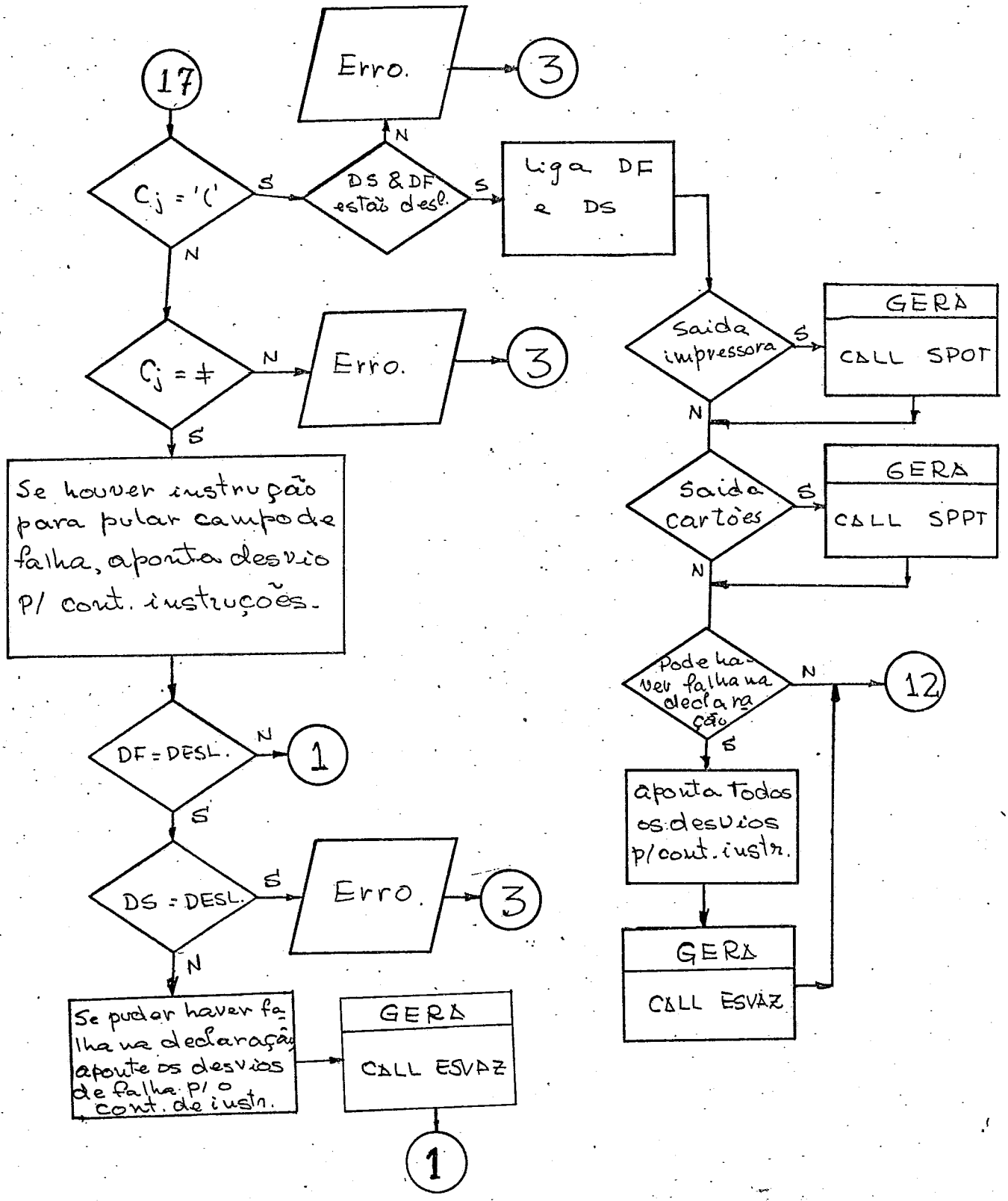

25
foi gerado código p/ NOMEC
N
DASBR = DASBR-1
S
Cj = ≠
S
10
N
NOMEC parou em '/'
N
Erro.
3
S
DASBR = 0
S
11
N
GERA
CALL CONC
DC DASBR+1
11

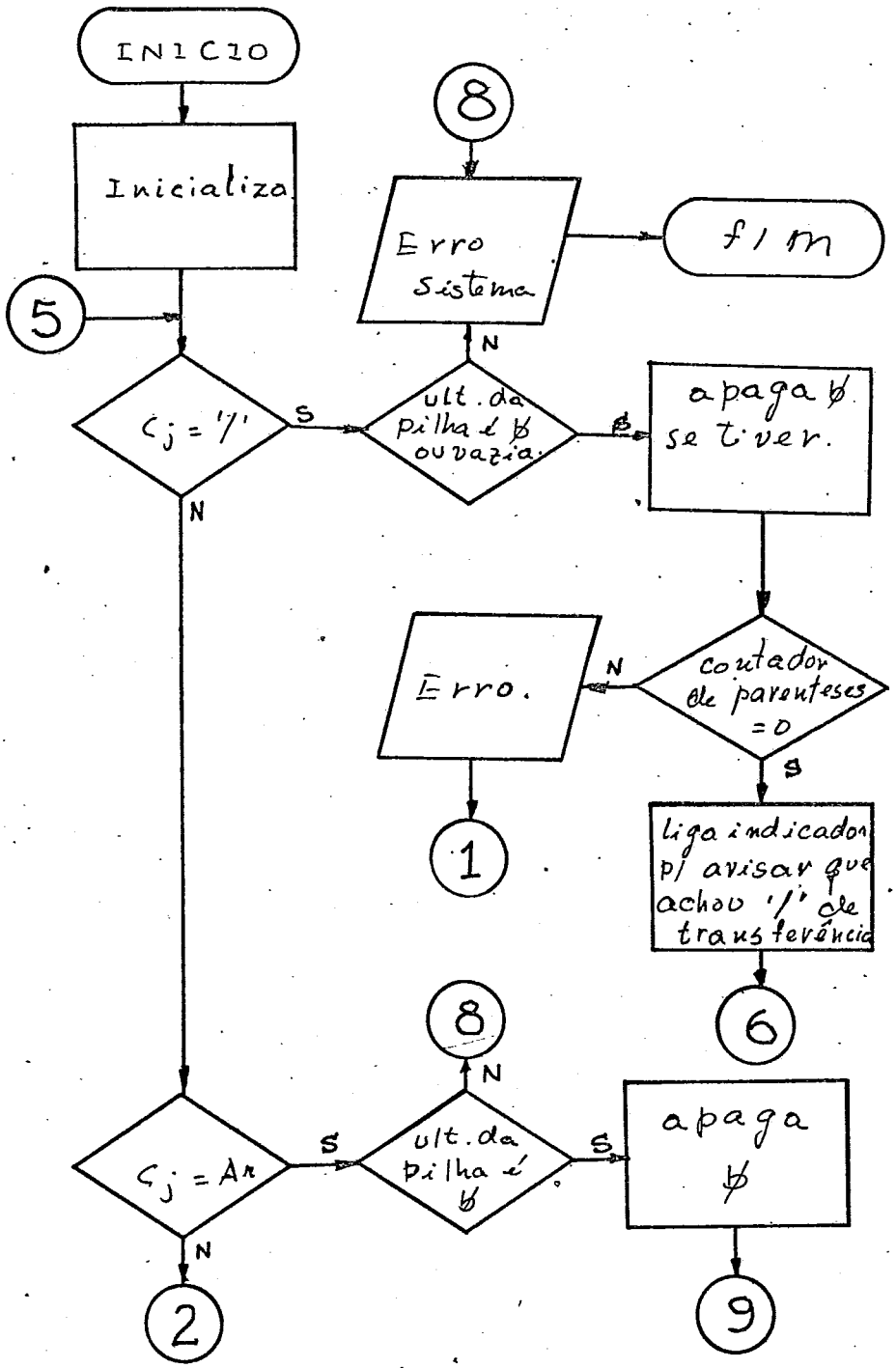

TRANSFERENCIA
DS = DESL.
DF = DESL.
16
Cj = 'G'
S
DS = DESL.
S
DS = LIG.
26
N
N
17
N
Cj = 'F'
Erro
3
S
DF = DESL.
N
S
DF = LIG.
DS = DESL.
S
GERA
Instrução para pular campo de falha
Pode haver falha na declaração
N
12
N
S
12
GERA
CALL ESVAZ
Desfaz lista apontadores apontando desvios p/ cont. instruções.


26
houve instrução p/ pular campo transf. falha
S
aponta desvio p/ contador de instruções
N
há saida impressora (DOUTS=LIG)
S
GERA
CALL SPOT
N
há saida cartões (DOUTG=LIG)
S
GERA
CALL SPPT
N
12
j = j + 1
Cj = '('
N
Erro
3
S
j = j + 1
Cj = '$'
S
28
N
27

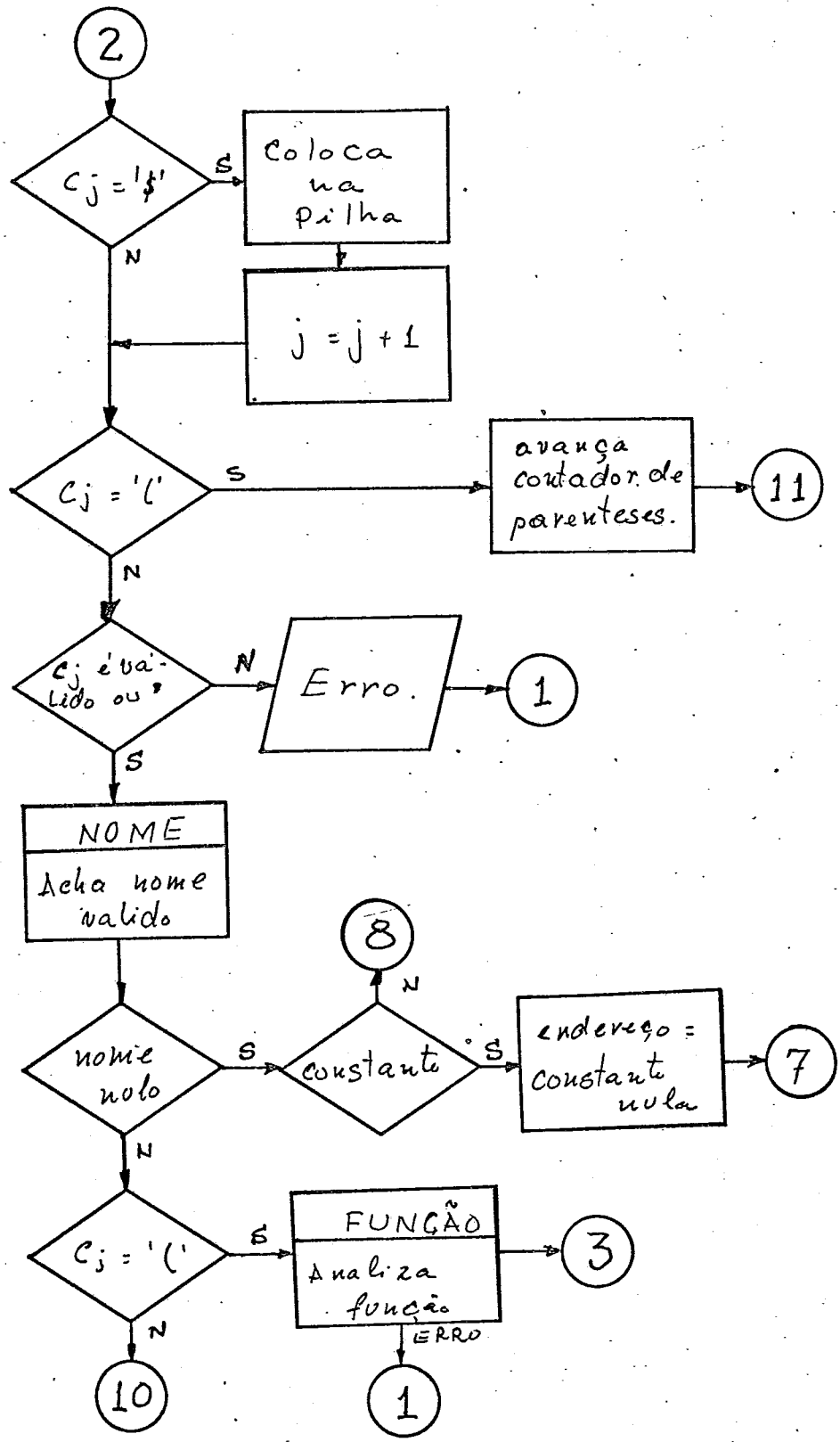

28
Liga indicador p/ avisar à NOMEC que vai ser chamada de um campo de transferencia
NOMEC
Avalia expressão
ERRO
3
Pode haver falha neste campo
S
N
GERA
B *+3
aponta desvio p/ falha p/ cont. instr.
GERA
CALL ERRO
DC 34
NOMEC terminou em ')'
N
Erro.
3
S
troca instr.
CALL IND P/
CALL LIND
GERA
CALL TIRA
STO *+1
B L *-*
16

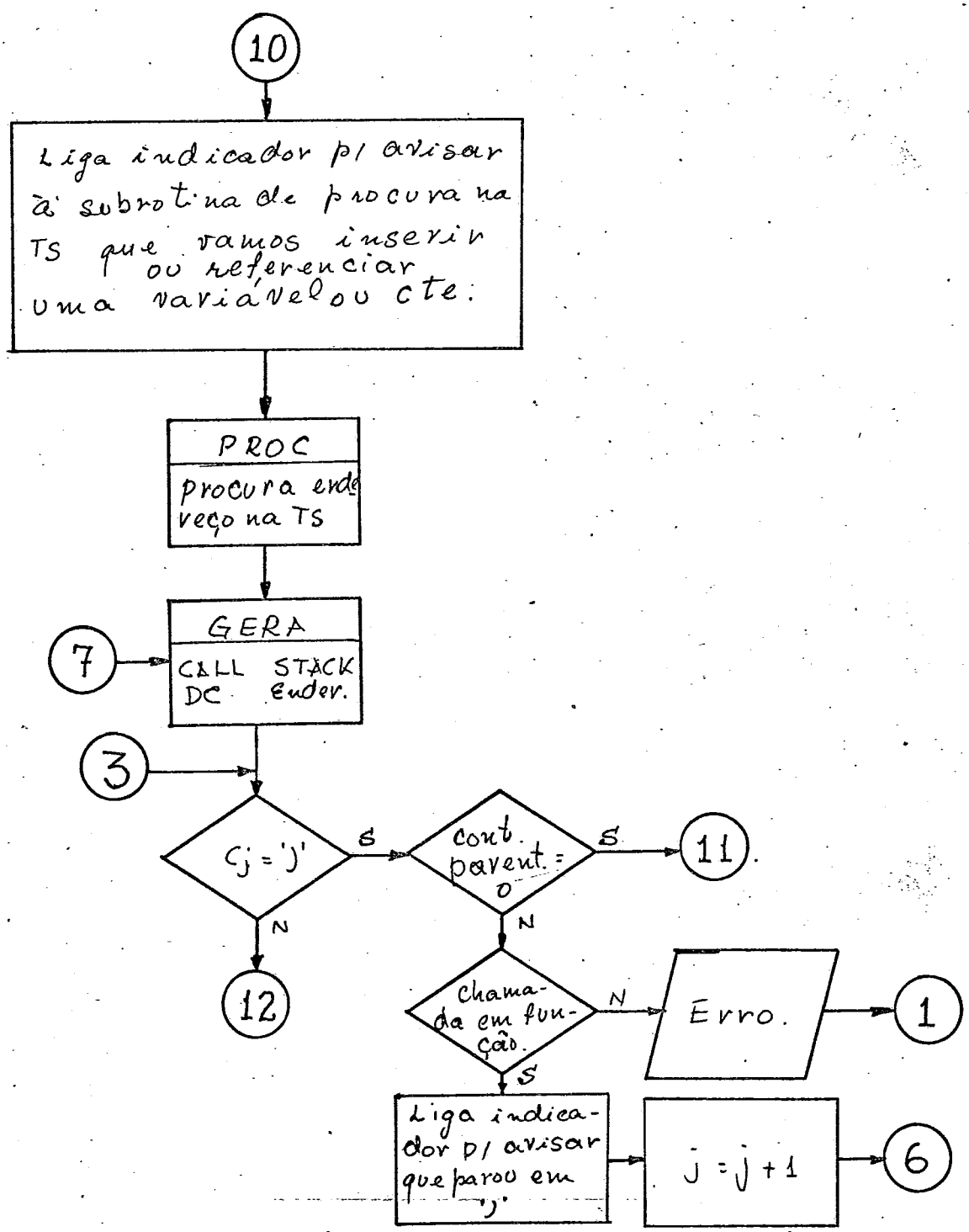

27
NOME
Acha nome válido
ERRO
3
Cj = ')'
N
Erro.
3
S
j = j + 1
Liga indicador p/ avisar subrotina PROC que vamos referenciar
PROC
Procura na TS.
ERRO
3
GERA
Desvio p/ end. fornecido
16

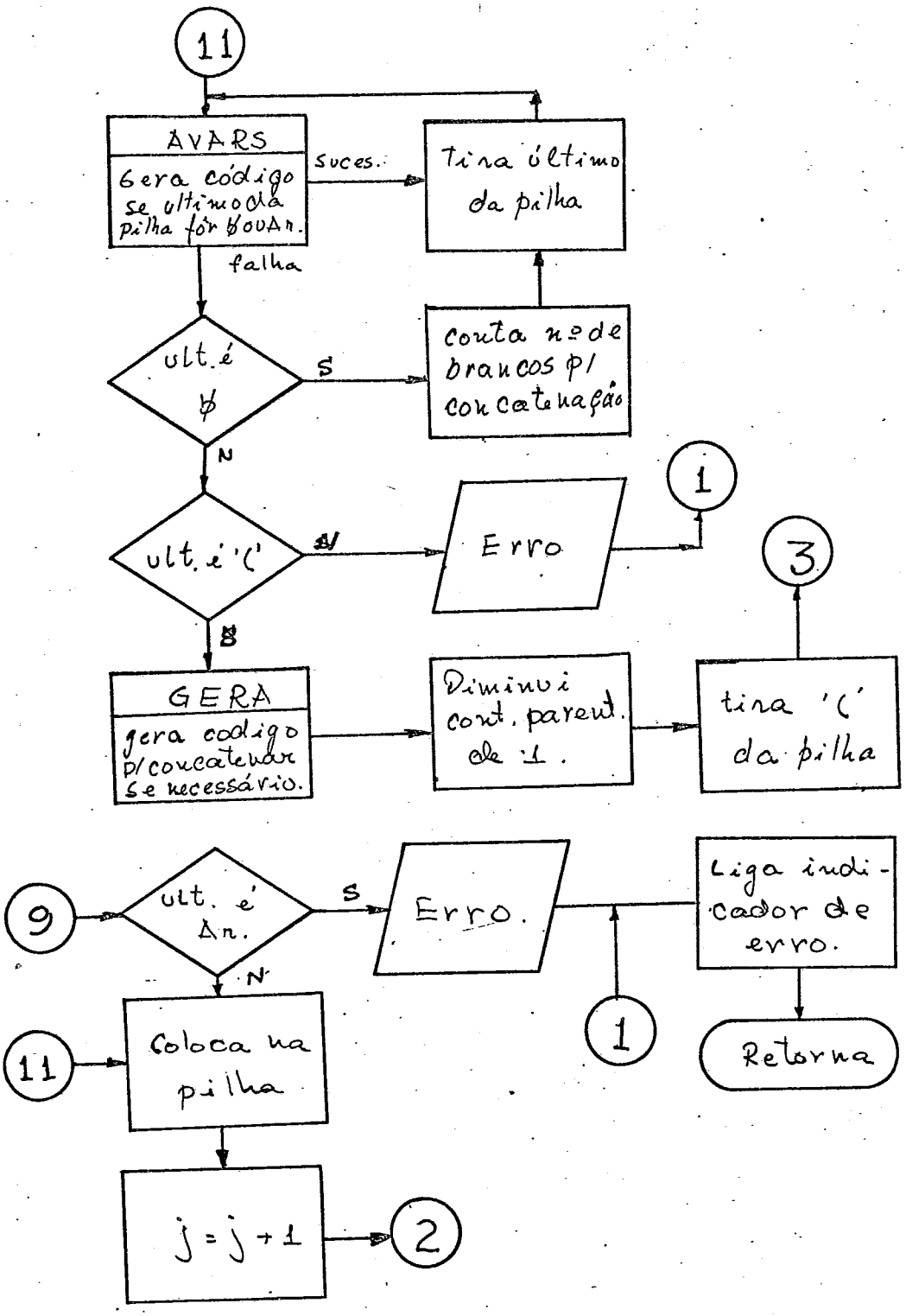

17
Erro.
3
Cj = '('
S
DS & DF estão desl.
S
Liga DF e DS
N
N
Cj = ±
N
Erro.
3
S
Se houver instrução para pular campo de falha, aponta desvio p/ cont. instruções.
Saida impressora
S
GERA
CALL SPOT
N
Saida cartões
S
GERA
CALL SPPT
N
Pode haver falha na declaração
N
12
S
aponta Todos os desvios p/cont. instr.
GERA
CALL ESVAZ
DF = DESL.
N
1
S
DS = DESL.
S
Erro.
3
N
Se puder haver falha na declaração aponte os desvios de falha p/ o cont. de instr.
GERA
CALL ESVAZ
1

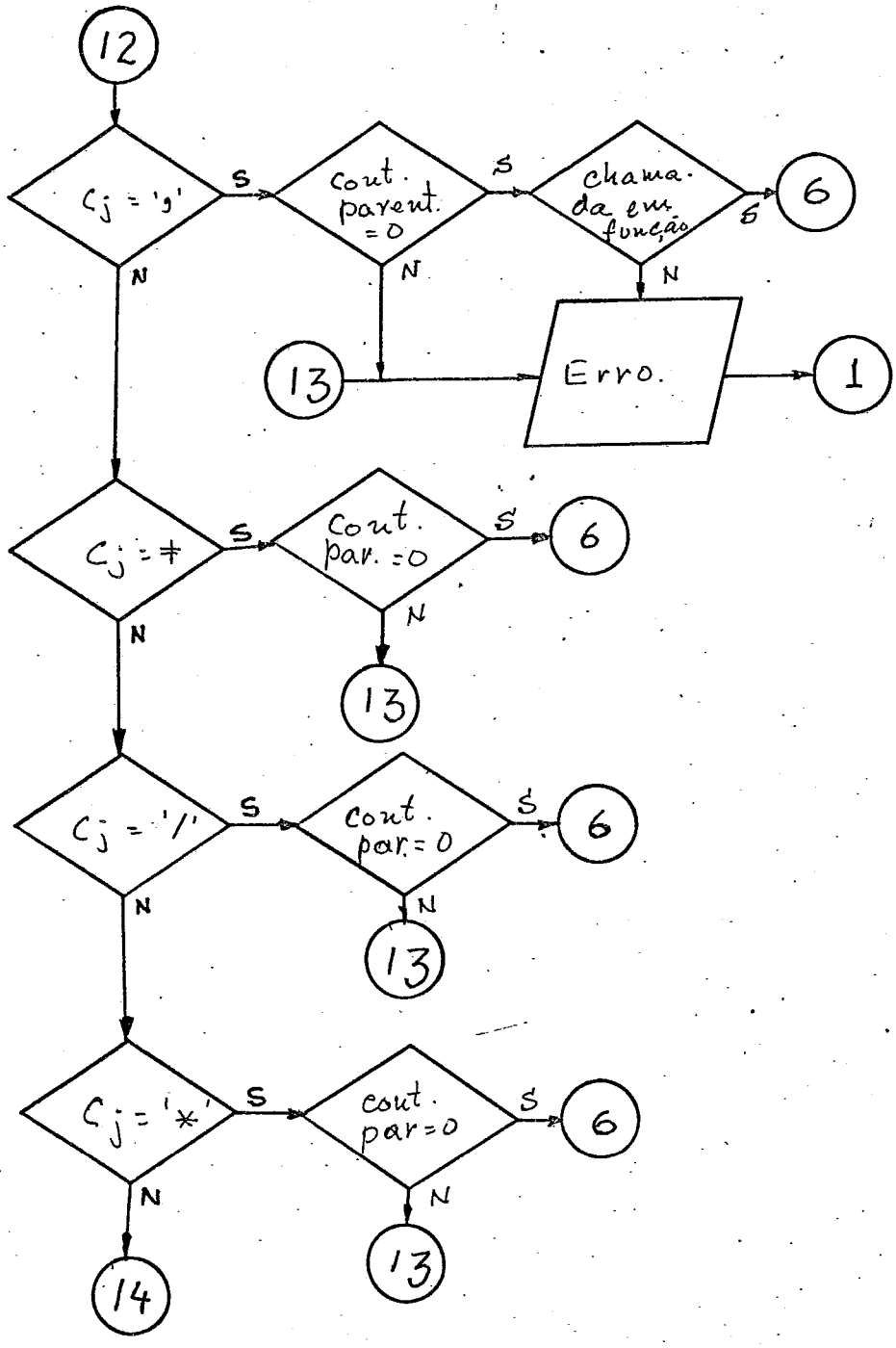


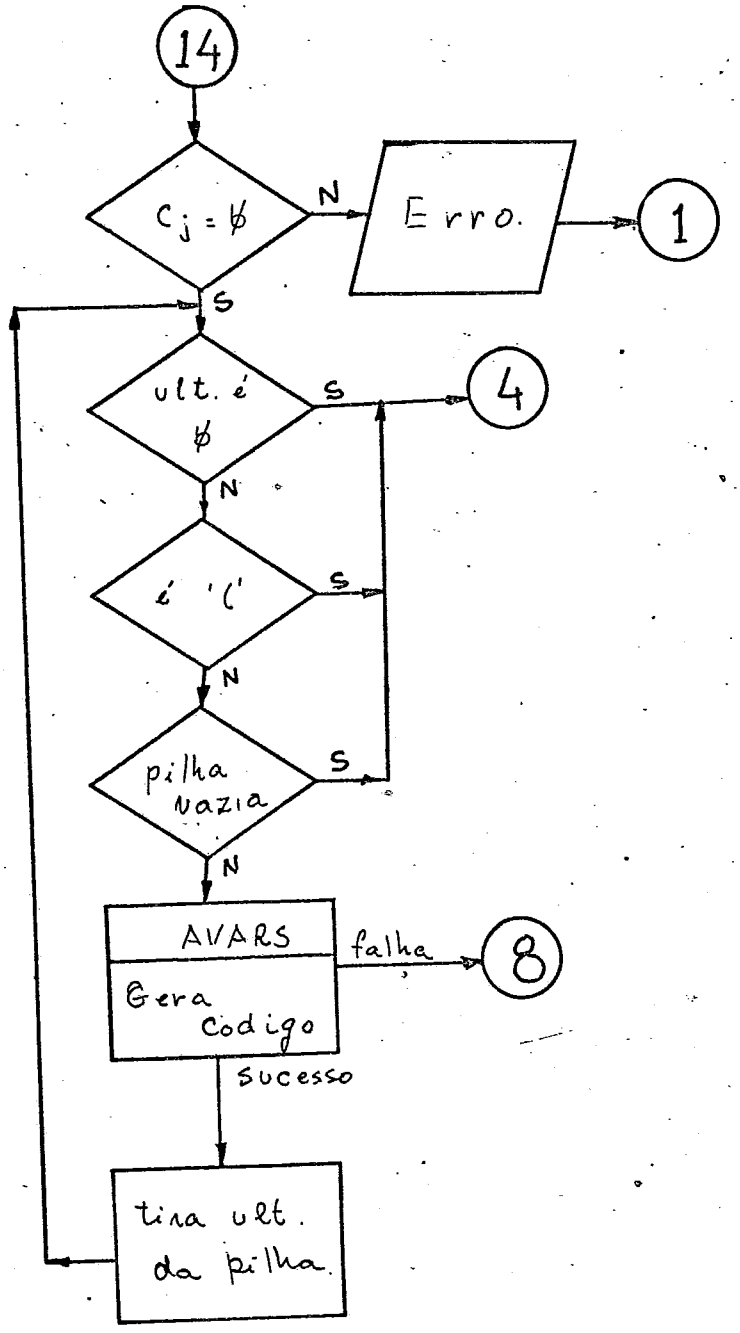
NOTA: pilha refere-se à pilha de operadores.

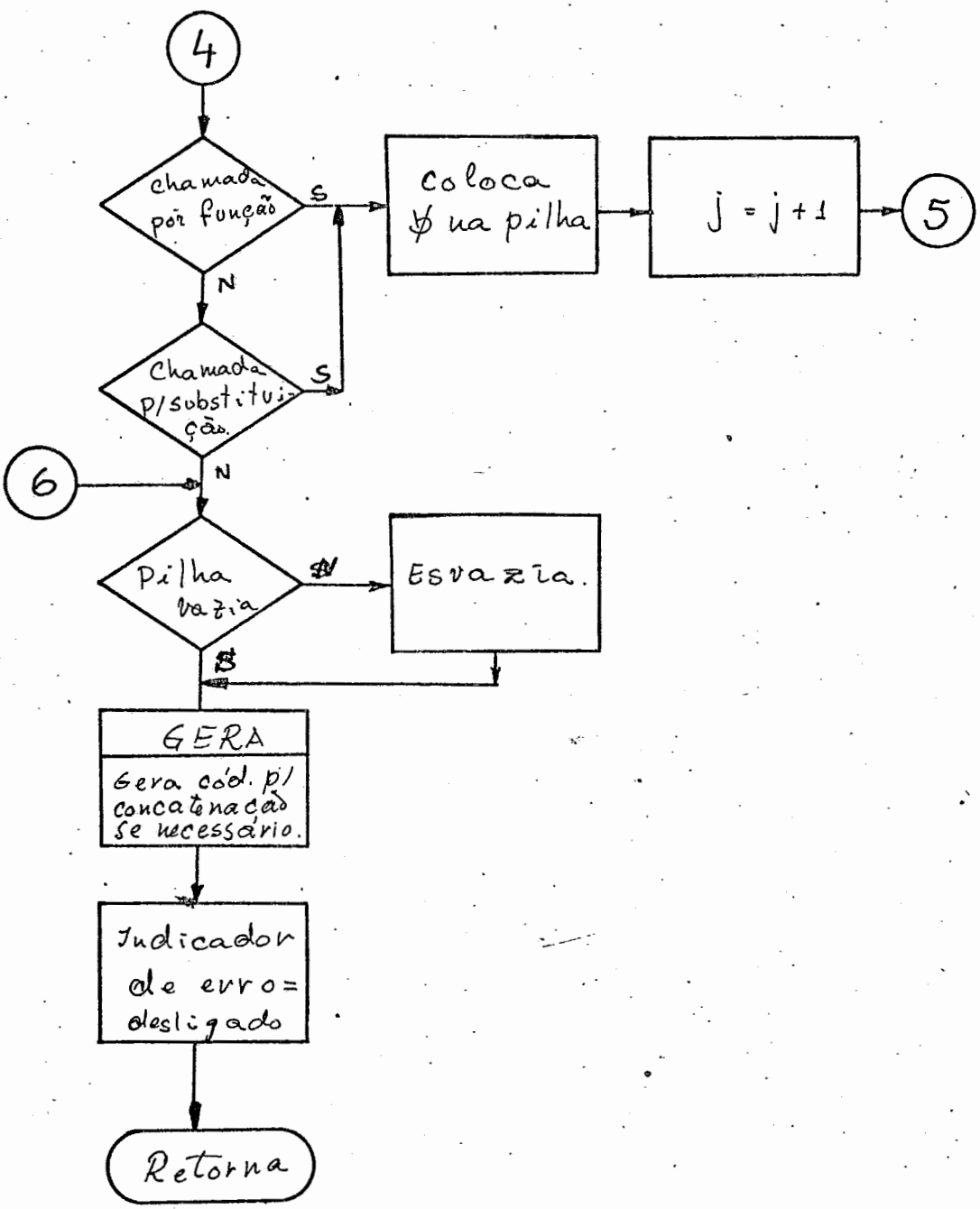

2
Cj = '$'
S
Coloca na pilha
N
j = j + 1
Cj = '('
S
avança contador de parenteses.
11
N
Cj é valido ou '
N
Erro.
1
S
NOME
Acha nome valido
8
N
nome nulo
S
constante
S
endereço = constante nula
7
N
Cj = '('
S
FUNÇÃO
Analiza função
3
ERRO
N
10
1


10
Liga indicador p/ avisar à subrotina de procura na TS que vamos inserir ou referenciar uma variável ou cte.
PROC
procura endereço na TS
GERA
CALL STACK DC ender.
7
3
Cj = 'j'
S
N
cont. parent. = 0
S
N
11
12
Chamada em função.
N
S
Erro.
1
Liga indicador p/ avisar que parou em 'j'
J = J + 1
6

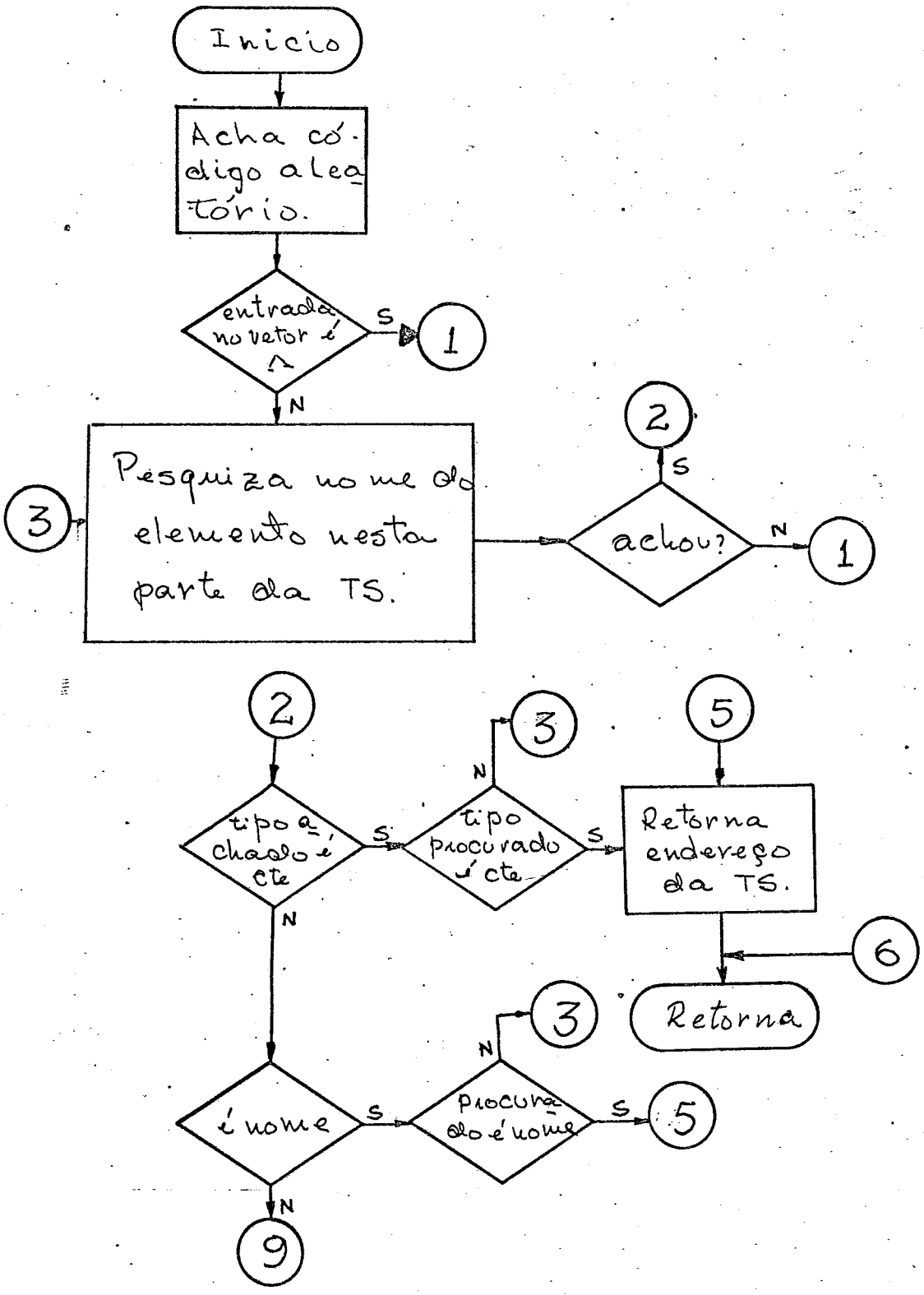

11
AVARS
Gera código se último da pilha for ϕ ou An.
sucess.
falha
Tira último da pilha
ult. é ϕ
S
N
Conta nº de brancos p/ concatenação
ult. é '('
S
N
Erro
1
3
GERA
gera código p/ concatenar se necessário.
Diminui cont. parent. de 1.
tira '(' da pilha
9
ult. é An.
S
N
Erro.
Liga indicador de erro.
1
Retorna
11
Coloca na pilha
j = j + 1
2

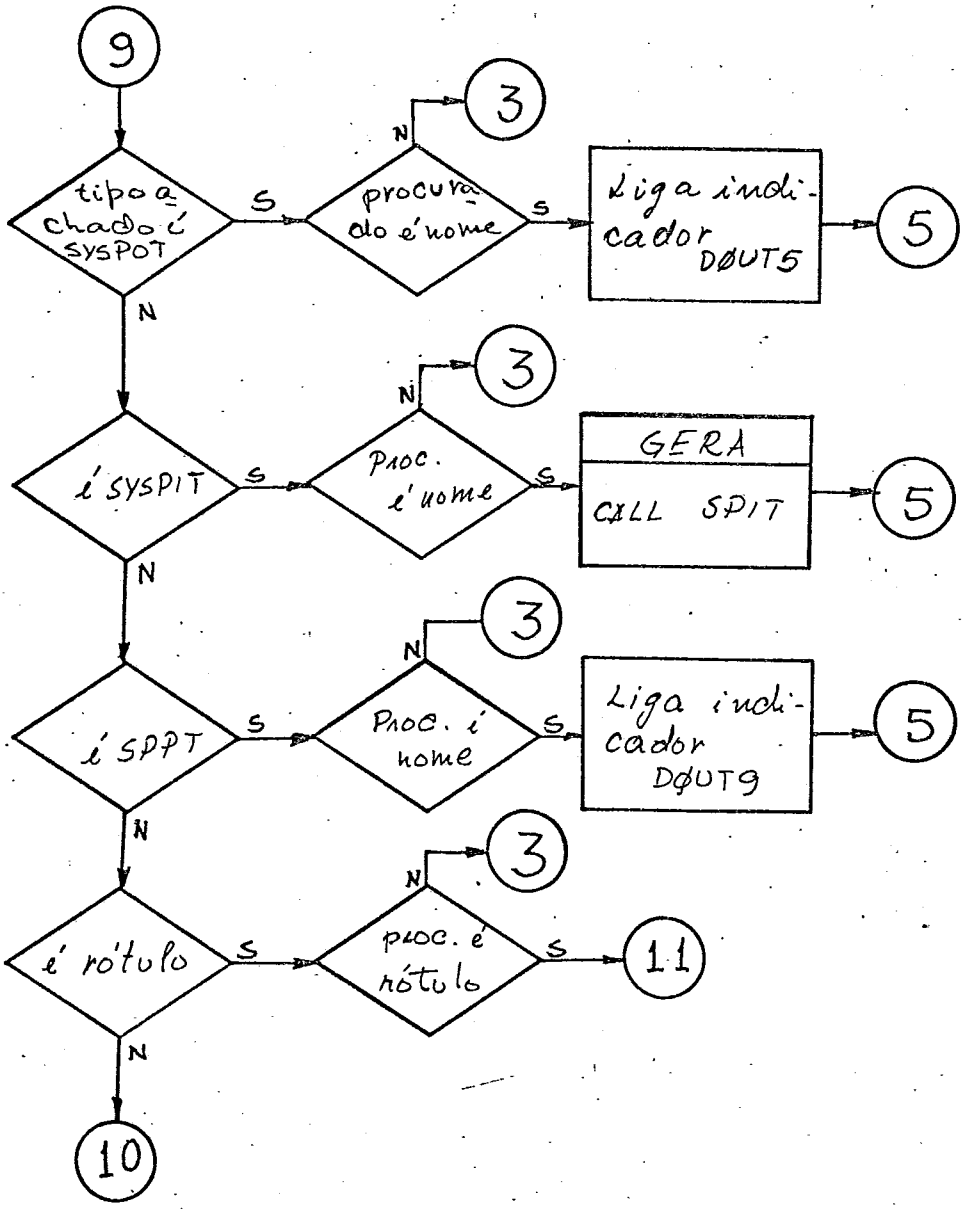

12
Cj = 'g'
S
cont. parent. = 0
S
chama. da em função
S
6
N
N
N
13
Erro.
1
Cj = +
S
cont. par. = 0
S
6
N
N
13
Cj = '/'
S
cont. par. = 0
S
6
N
N
13
Cj = '*'
S
cont. par = 0
S
6
N
N
13
14

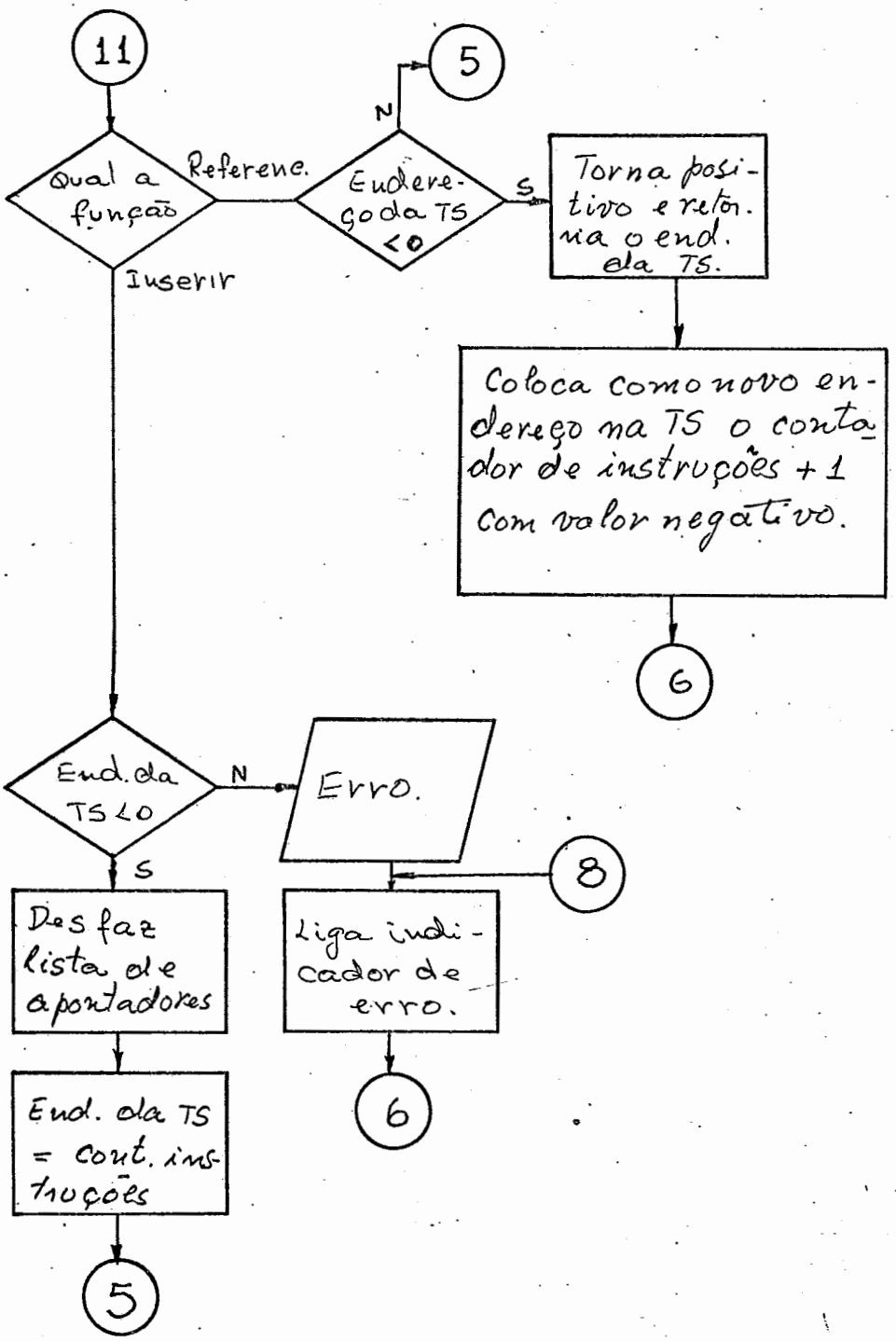

14
$C_j = \phi$
N
Erro.
1
S
ult. é ϕ
S
4
N
é '('
S
N
pilha vazia
S
N
AVARS
Gera Codigo
falha
8
sucesso
tira ult. da pilha.

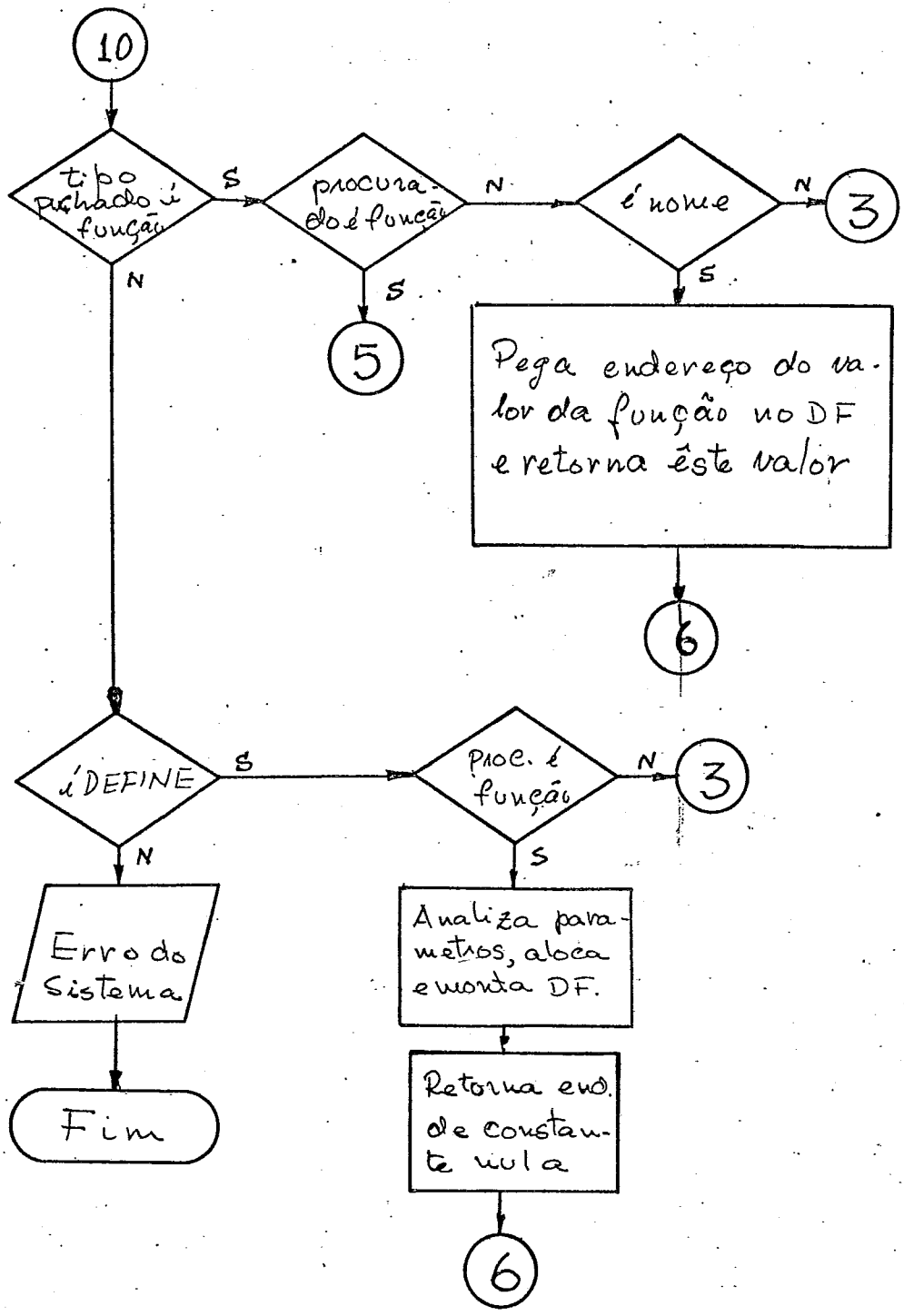

4
chamada por função
S
Coloca ϕ na pilha
j = j + 1
5
N
Chamada p/ substituição.
S
6
N
Pilha vazia
S
Esvazia.
S
GERA
Gera cód. p/ concatenação se necessário.
Indicador de erro = desligado
Retorna

SUBROTINA DE PROCURA NA TS.
(PROC)

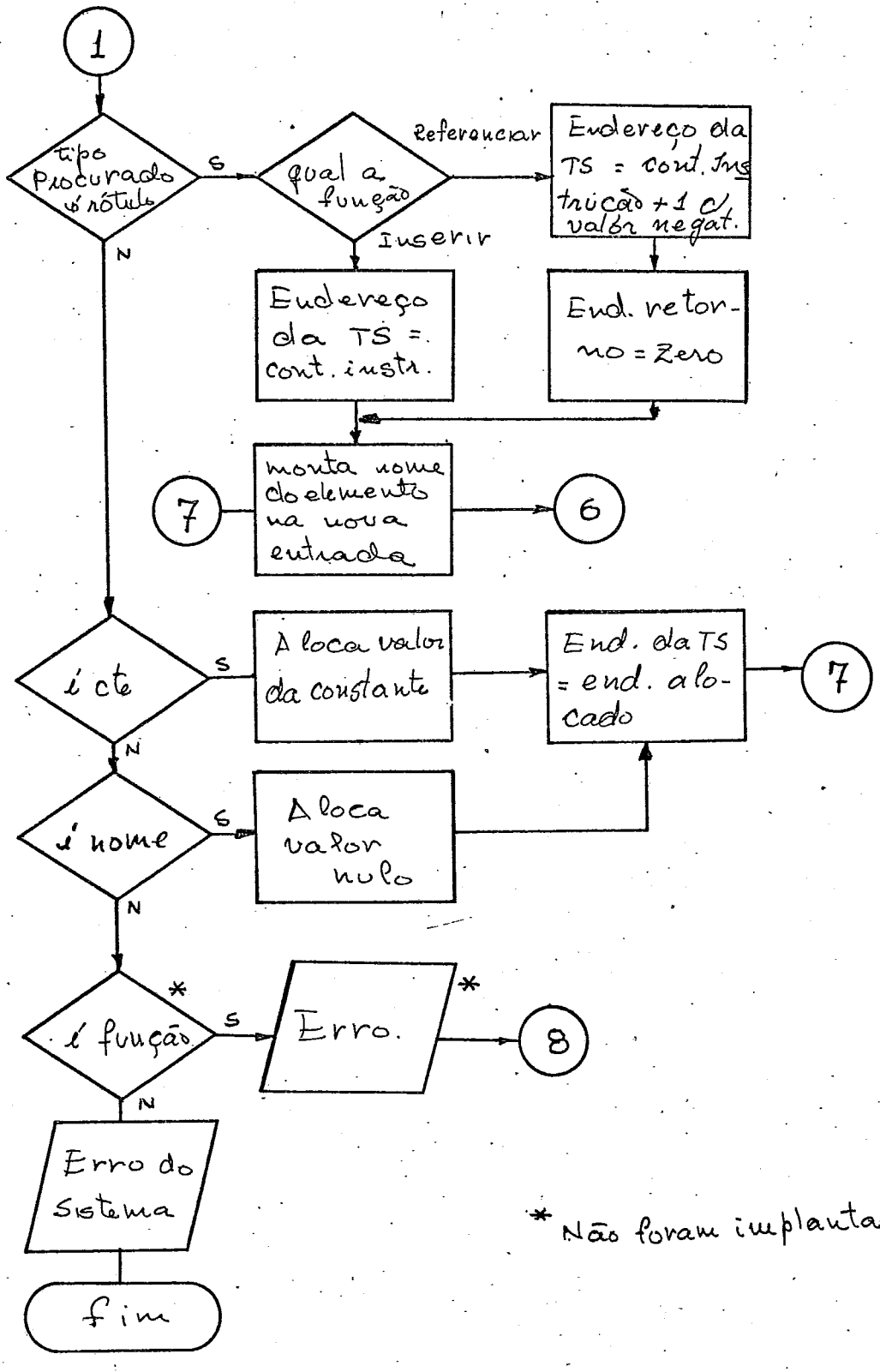



9
tipo achado é SYSPOT
S
N
3
procura do é nome
S
N
Liga indicador DØUT5
5
é SYSPIT
S
N
3
Proc. é nome
S
N
GERA
CALL SPIT
5
é SPPT
S
N
3
Proc. é nome
S
N
Liga indicador DØUT9
5
é rótulo
S
N
3
proc. é rótulo
S
N
11
10


11
Qual a função
Referenc.
Inserir
N
5
Endereço da TS < 0
S
Torna positivo e retorna o end. da TS.
Coloca como novo endereço na TS o contador de instruções + 1 com valor negativo.
6
End. da TS < 0
N
Erro.
8
S
Desfaz lista de apontadores
End. da TS = cont. instruções
5
Liga indicador de erro.
6

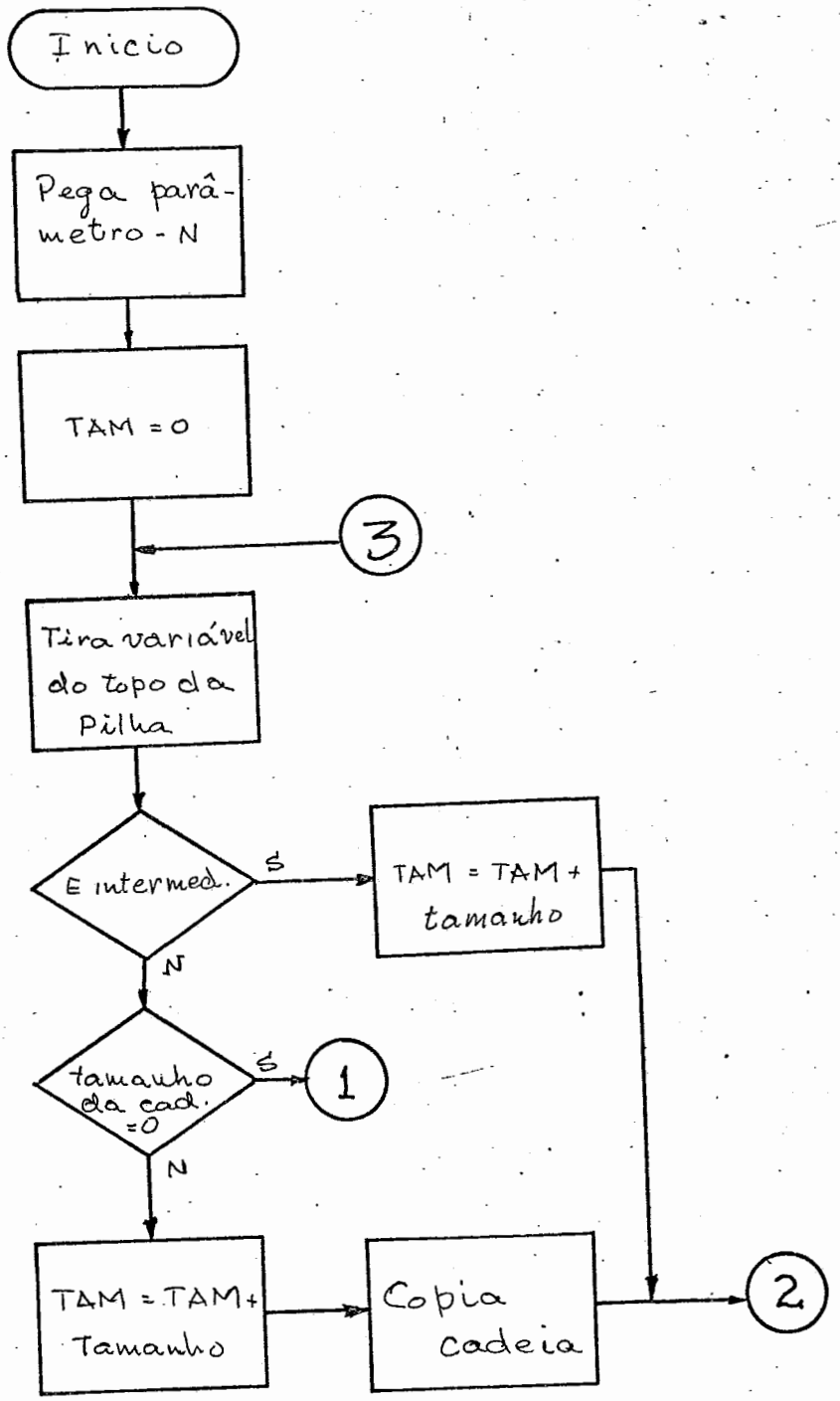


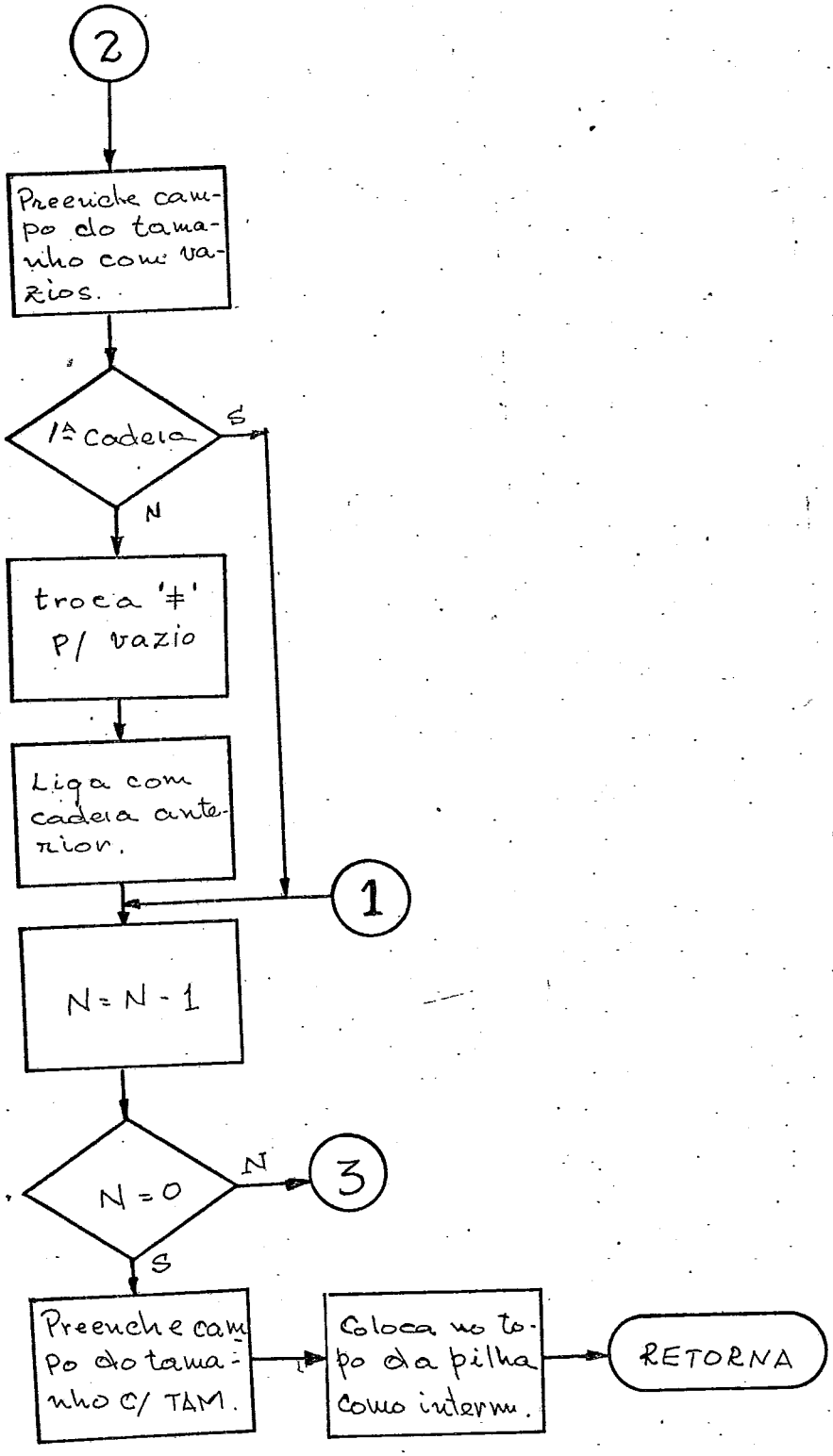
NOTA: Esta parte não foi implantada.



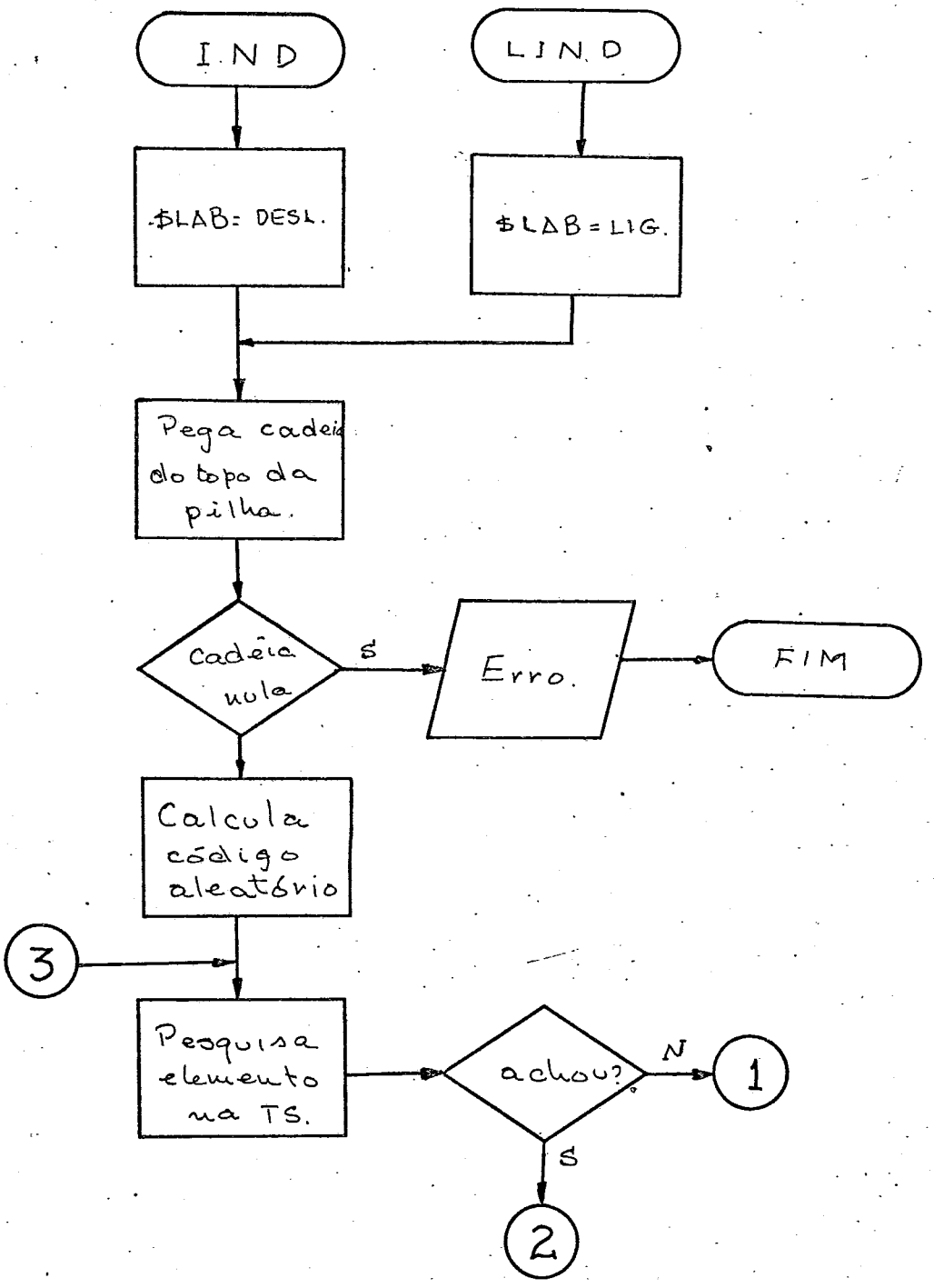
* Não foram implantados.

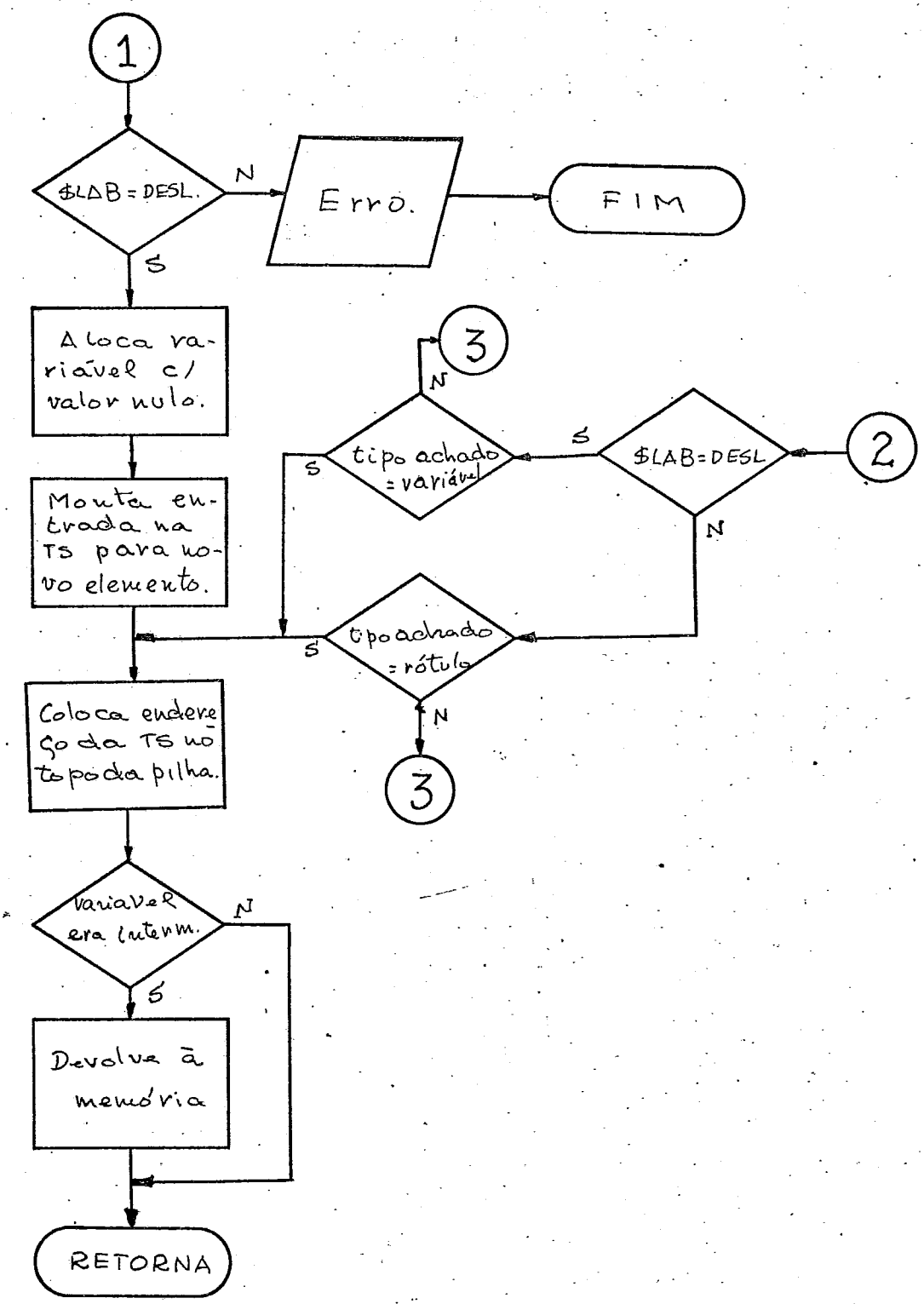
SUBROTINA DE CONCATENAÇÃO (CONC)



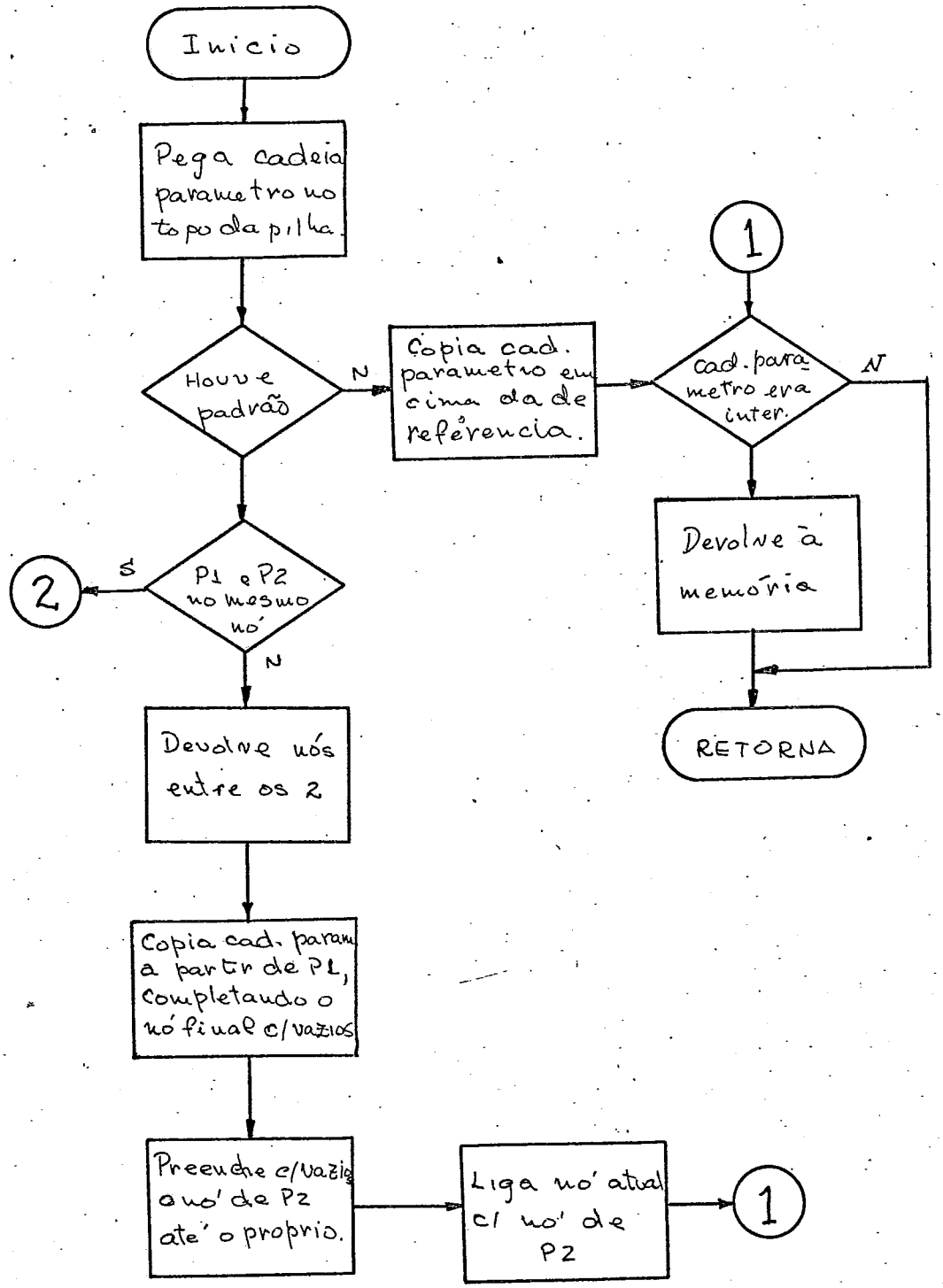

2
Preenche campo do tamanho com vazios.
1ª Cadeia
S
N
troca '#' p/ vazio
Liga com cadeia anterior.
1
N = N - 1
N = 0
N
3
S
Preenche campo do tamanho C/ TAM.
Coloca no topo da pilha como interm.
RETORNA

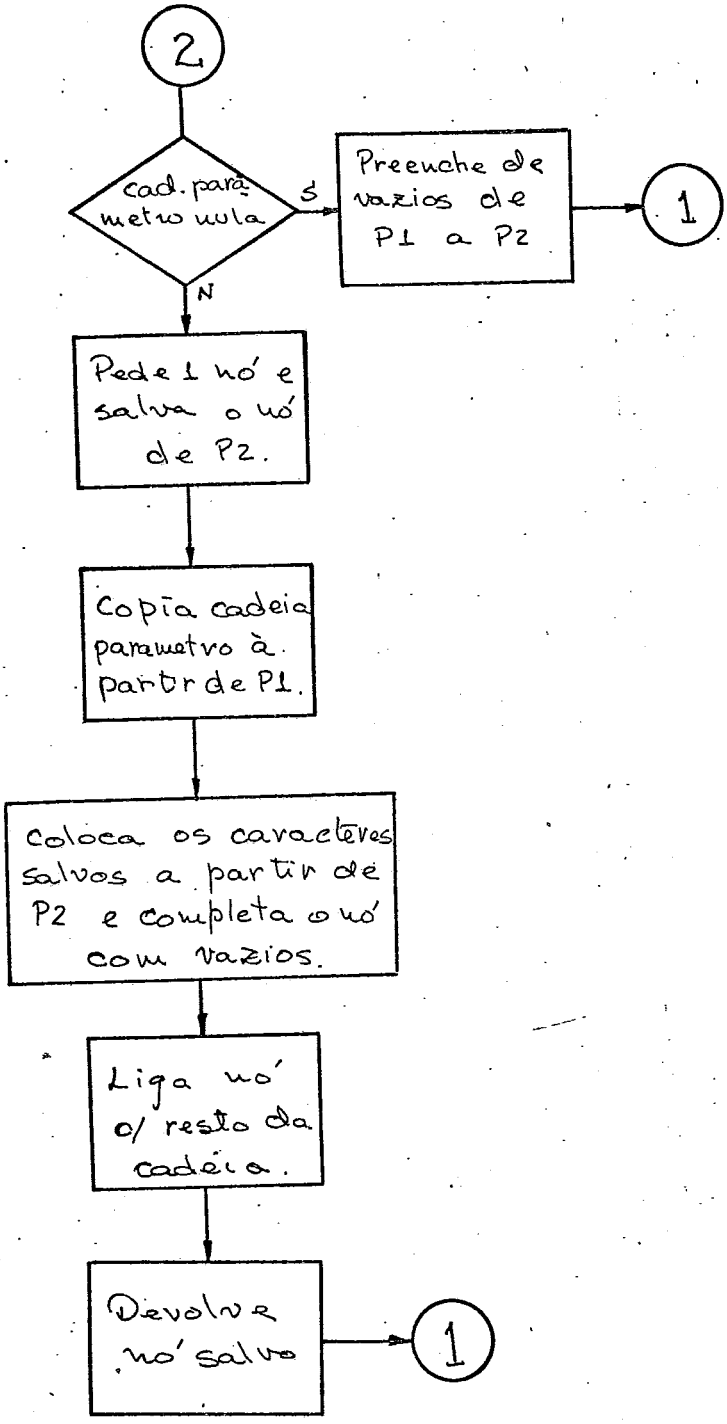
ENDEREÇAMENTO INDIRETO (IND & LIND)




1
$LAB = DESL.
N
Erro.
FIM
S
Aloca variável c/ valor nulo.
Monta entrada na TS para novo elemento.
Coloca endereço da TS no topo da pilha.
3
N
tipo achado = variável
S
$LAB = DESL
S
2
N
tipo achado = rótulo
S
N
3
variável era interm.
N
S
Devolva à memória
RETORNA

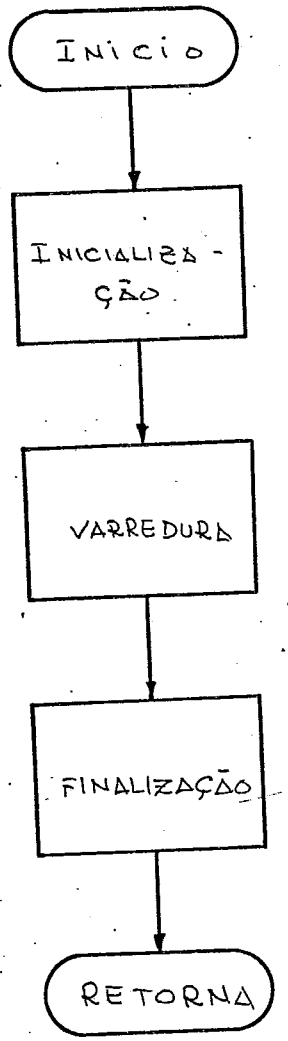
SUBROTINA DE SUBSTITUIÇÃO (ASSIG)

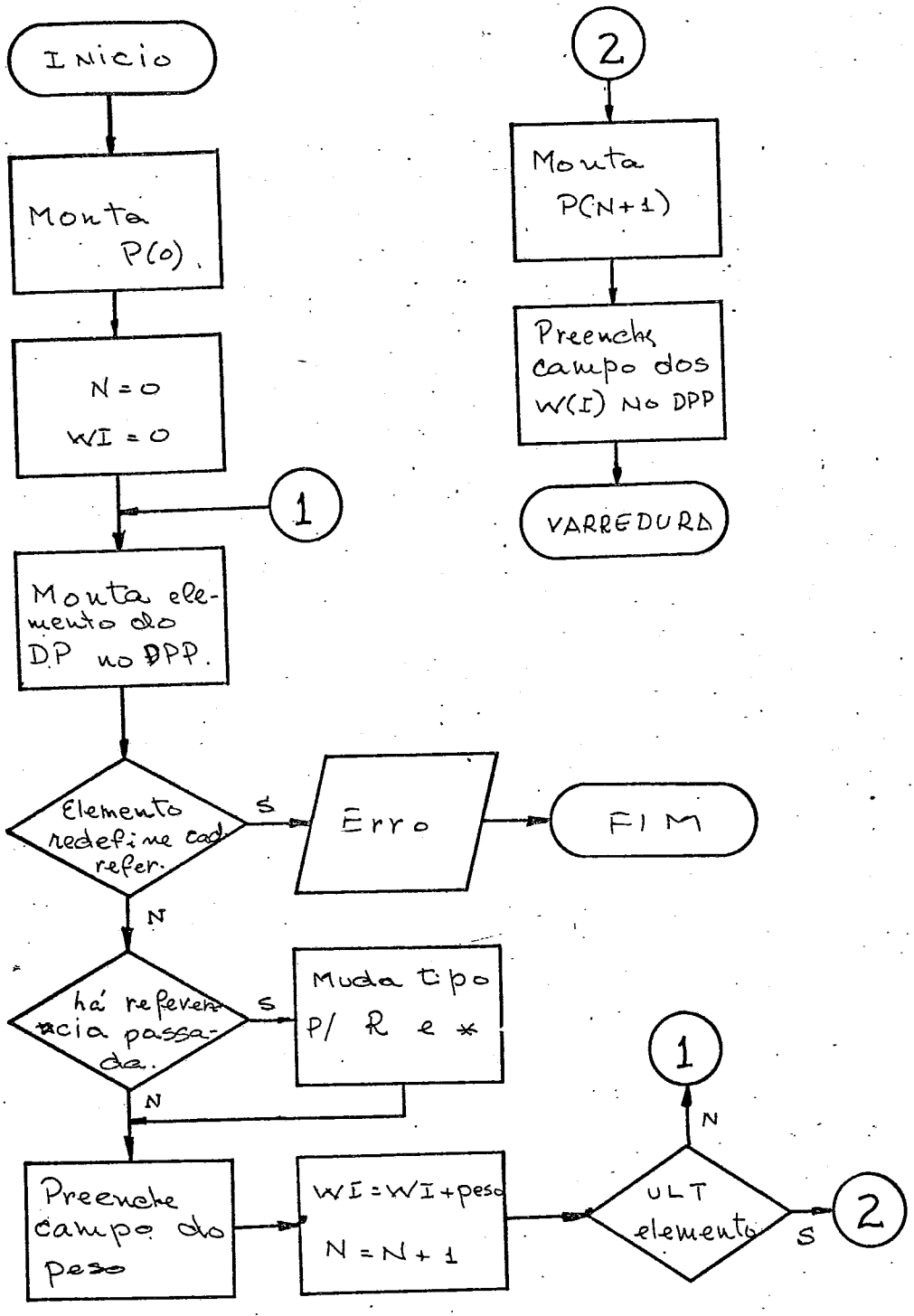



2
cad. parâmetro nula
S
Preenche de vazios de P1 a P2
1
N
Pede 1 nó e salva o nó de P2.
Copia cadeia parametro à partir de P1.
Coloca os caracteres salvos a partir de P2 e completa o nó com vazios.
Liga nó c/ resto da cadeia.
Devolve nó salvo
1

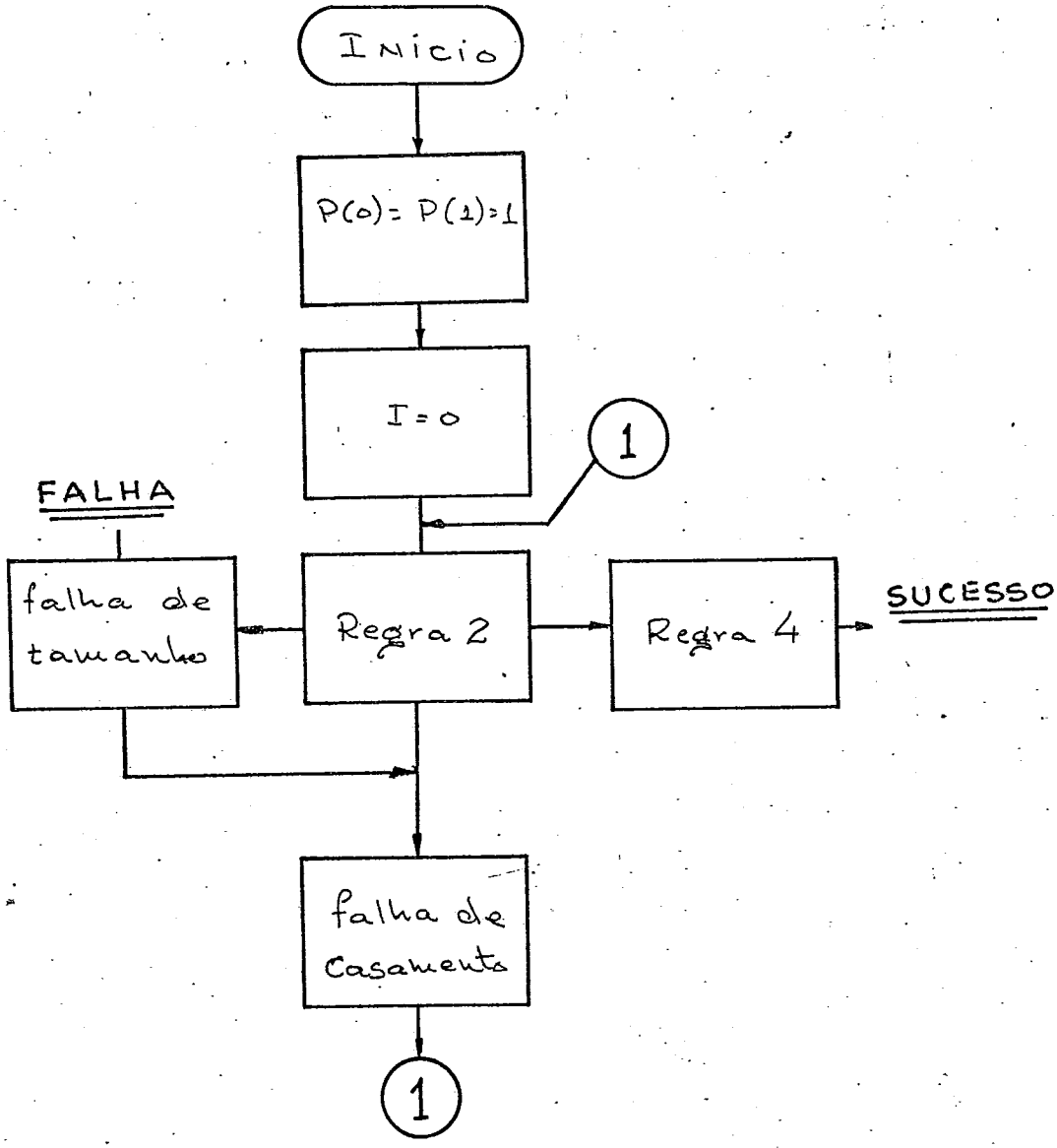
SUBROTINA DE COMPARAÇÃO DO PADRÃO (PATT)

ESTRUTURA GERAL

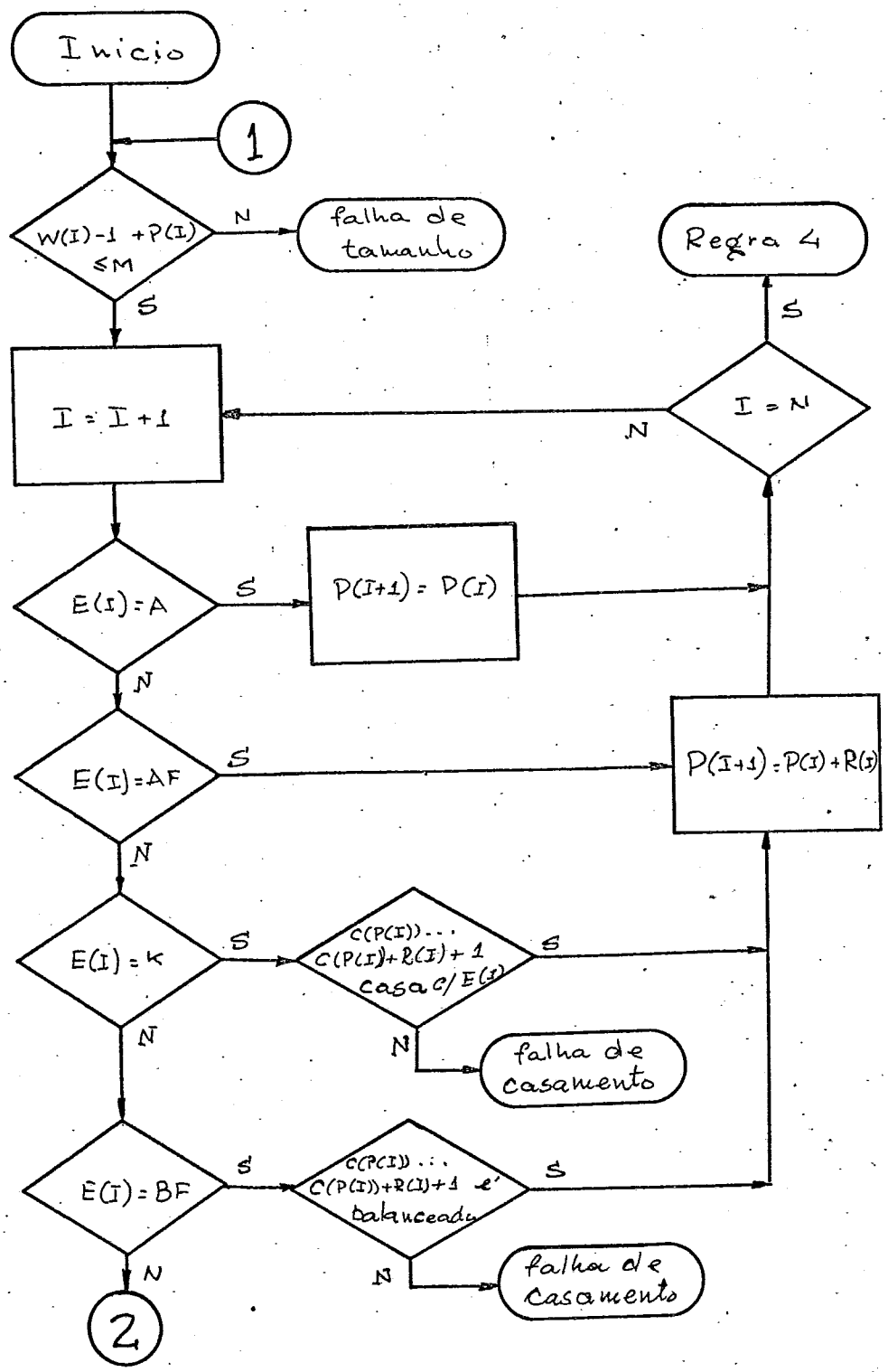


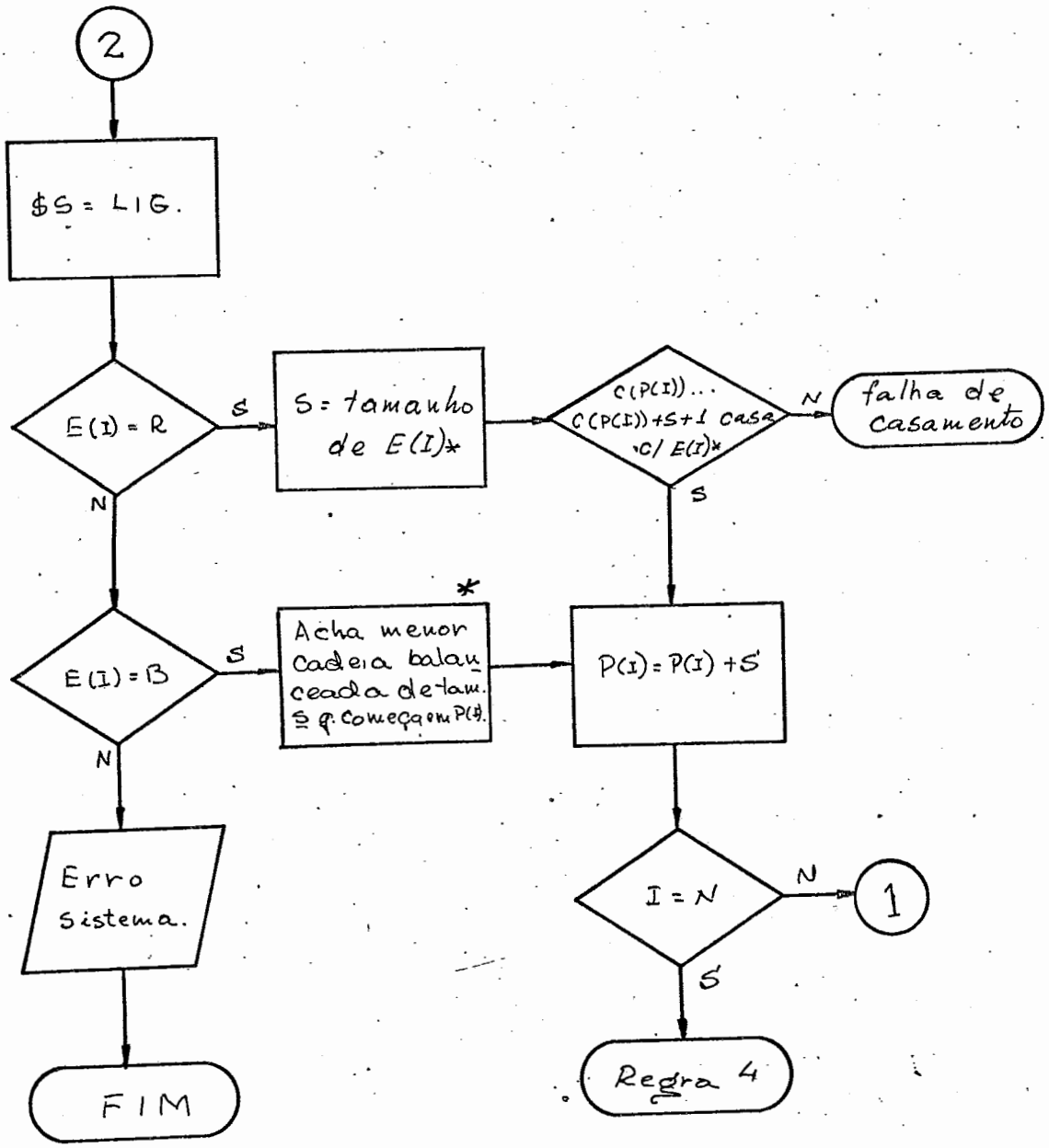

INICIO
Monta P(0)
N = 0
WI = 0
1
Monta elemento do DP no DPP.
Elemento redefine cad refer.
S
Erro
FIM
N
há referencia passada.
S
Muda tipo p/ R e *
N
Preenche campo do peso
WI = WI + peso
N = N + 1
ULT elemento
N
1
S
2
2
Monta P(N+1)
Preenche campo dos W(I) no DPP
VARREDURA

VARREDURA – ESTR. GERAL



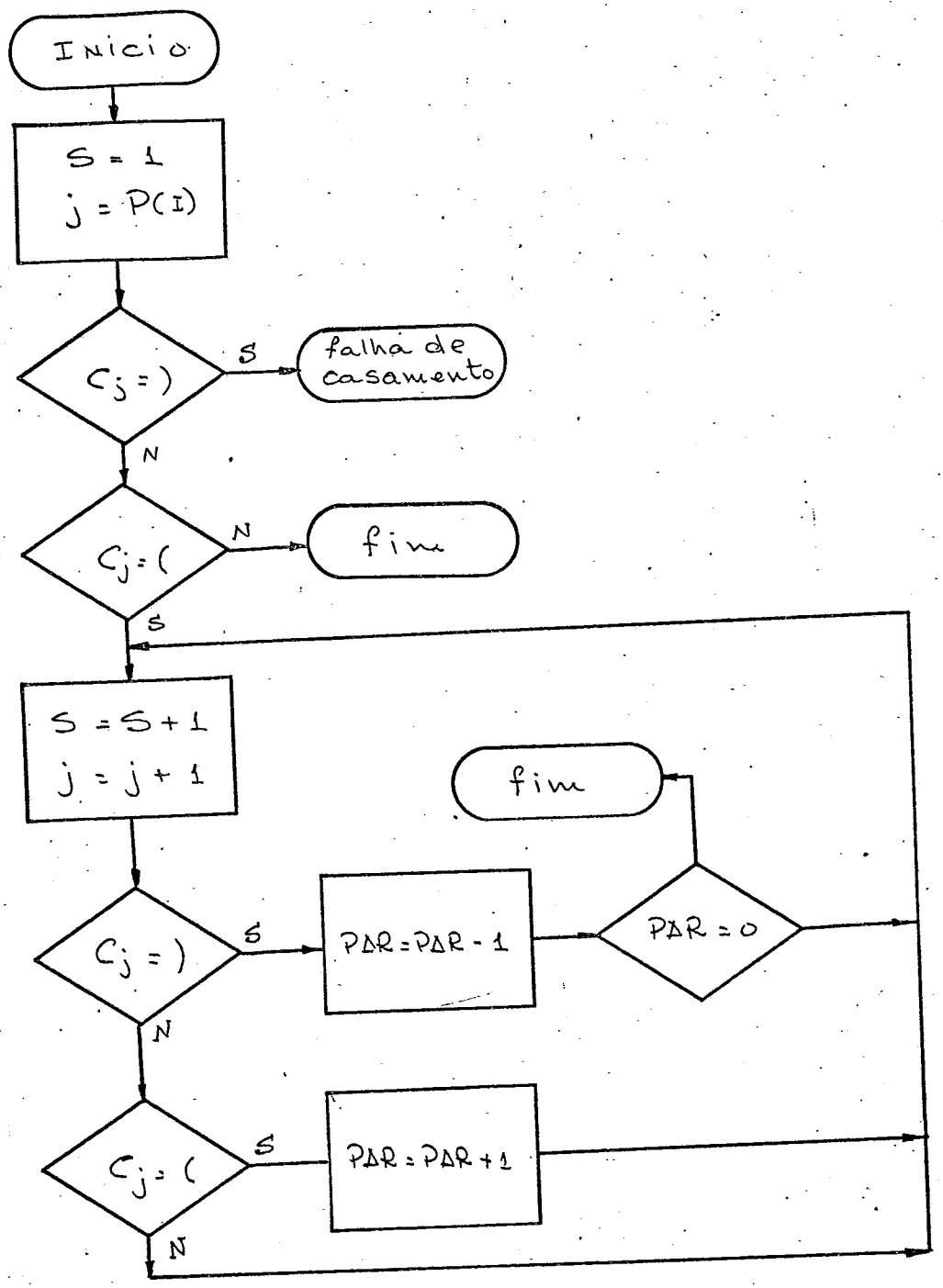
REGRA 2


Inicio
1
W(I)-1 +P(I) ≤M
N
falha de tamanho
S
I = I+1
E(I)=A
S
P(I+1) = P(I)
N
E(I)=AF
S
P(I+1) = P(I)+R(I)
N
E(I)=K
S
C(P(I)) . . . C(P(I))+R(I)+1 casa c/ E(I)
S
N
falha de casamento
E(I)=BF
S
C(P(I)) . . . C(P(I))+R(I)+1 é' balanceada
S
N
falha de casamento
N
2
I = N
N
Regra 4
S

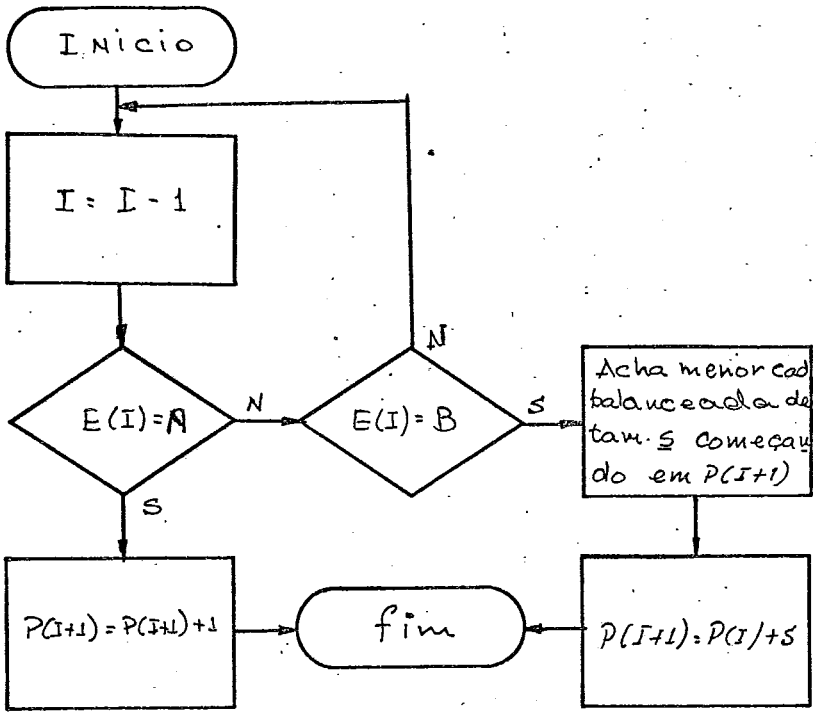


* Detalhado mais adiante.

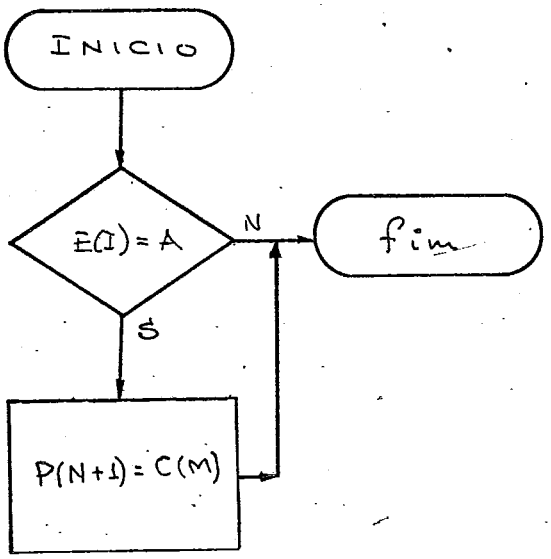
ALGORITMO P/ ACHAR MENOR CADEIA BALANCEADA DE TAMANHO $\leq$ COMEÇANDO EM P(I).



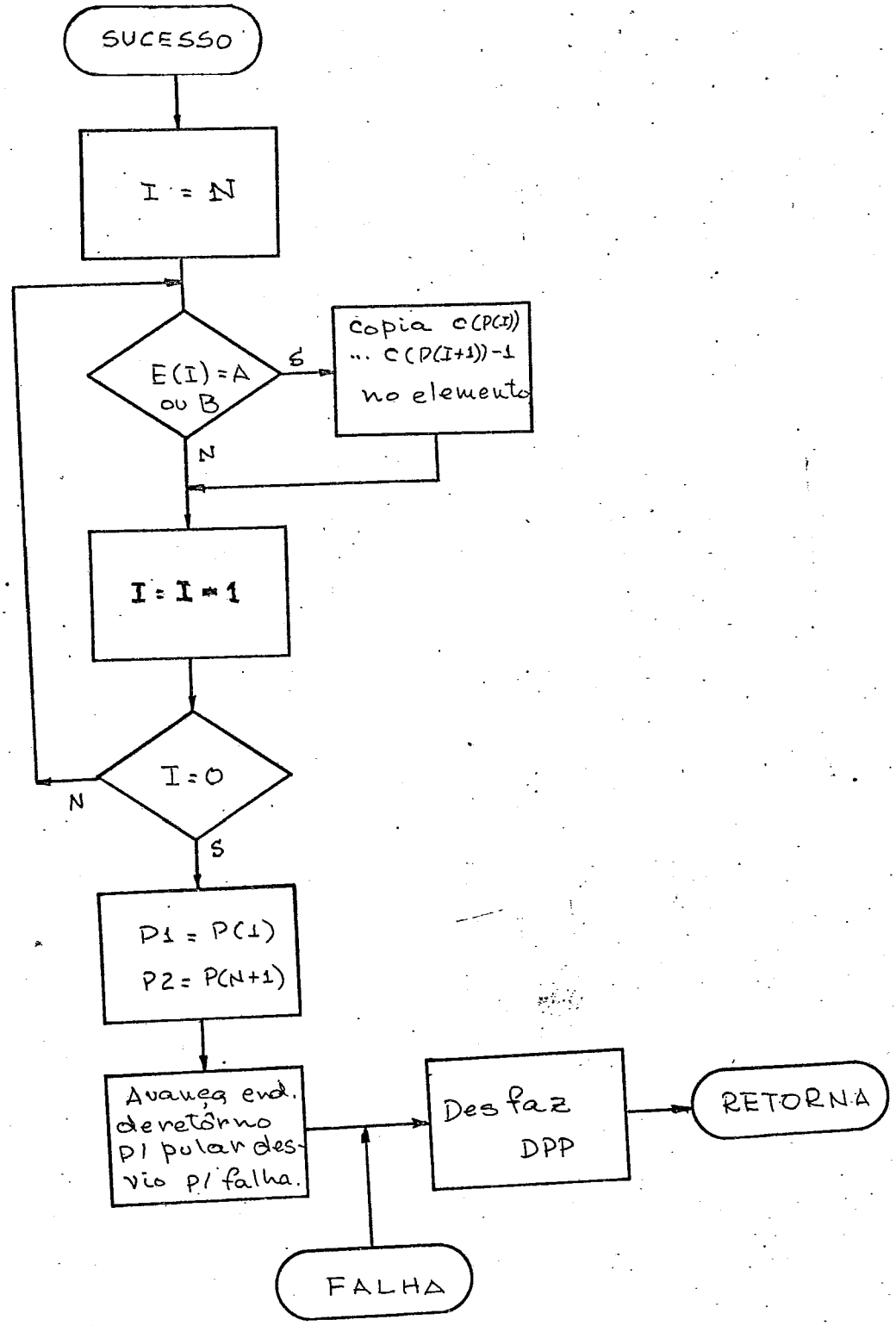
FALHA DE CASAMENTO



REGRA 4



FINALIZAÇÃO



APENDICE II

PROGRAMAS EXEMPLO

```
PAGE   1     N17

// JOB     00FF                                              N17

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         00FF         00FF         0000
                            10FF         0001

V2 M05    ACTUAL 32K   CONFIG 32K

// XEQ SNOBO


-TITLE                  EDITOR DE TEXTO COMPLETO
*
* ESTE PROGRAMA FOI USADO PARA EDITAR O TEXTO DO PRESENTE
* TRABALHO. A SAIDA DA EDICAO FOI REALIZADA EM CARTOES PARA
* POSTERIOR LISTAGEM.
*
* AS CONVENCOES USADAS FORAM AS SEGUINTES -
*
*         O ULTIMO CARTAO E IDENTIFICADO POR $$ .
*
*         O CODIGO $$N$$ FAZ COM QUE SEJAM PULADAS N LINHAS.
*
*         PARAGRAFOS SAO IDENTIFICADOS POR UM BRANCO NA
*         PRIMEIRA COLUNA.
*
*         EMBORA NA EDICAO O PROGRAMA NAO SEPARE SILABAS,
*         ADMITE QUE O TEXTO ORIGINAL TENHA UM HIFEN NA
*         ULTIMA COLUNA NAO BRANCA COMO INDICADOR DE SEPARA-
*         CAO DE SILABAS.
*
*         O PRIMEIRO CARTAO LIDO DEVE CONTER O NUMERO DE
*         CARACTERES POR LINHA DO TEXTO EDITADO.
*
*
* INICIALIZA
*
          BRS   =  '                    '
                   '                    '
*
* LE NUMERO DE CARACTERES POR LINHA EDITADA
*
          SYSPIT     *N*   ' '
*
* LE LINHA
*
LE        LINHA      =  SYSPIT
*
* TESTA SE NECESSARIO PULAR LINHA(S)
*
```

EDITOR DE TEXTO COMPLETO

```
          LINHA  '$$'  *M*  '$$'          /F(TESTA.ULTIMA)
*
* PULA M LINHAS
*
          SYSPPT  =  TEXTO
          SYSPPT  =  '$$'  M  '$$'
          TEXTO  *SYSPOT*  =
PULA      M  =  M  -  '1'
          M  '-'                          /S(LE)
          SYSPOT  =  ' '                  /(PULA)
*
* TESTA ULTIMA LINHA
*
TESTA.ULTIMA  LINHA  '$$'                 /S(ACABOU)
*
* TESTA LINHA EM BRANCO
*
          LINHA  BRS  BRS                 /F(TESTA.PARAGRAFO)
          SYSPPT  =  TEXTO
          SYSPPT  =  ' '
          TEXTO  *SYSPOT*  =
          SYSPOT  =  ' '      /(LE)
*
* TESTA PARAGRAFO
*
TESTA.PARAGRAFO    LINHA  *CARACTER/'1'*
          CARACTER  ' '                   /F(TIRA.BRANCO)
          SYSPPT  =  TEXTO
          TEXTO  *SYSPOT*  =
*
* SUPRIME BRANCOS DO FINAL
*
TIRA.BRANCO        M  =  '79'
PEGA.ULTIMO        LINHA  *LINHA1/M*  *CARACTER/'1'*
          CARACTER  ' '                   /F(JUNTA)
          M  =  M  -  '1'                 /(PEGA.ULTIMO)
*
* TESTA SEPARACAO DE SILABAS E JUNTA LINHA C/ TEXTO
*
JUNTA     CARACTER  '-'       /S(SUPRIME)
          TEXTO  =  TEXTO  LINHA1  CARACTER  ' '  /(EDITA)
SUPRIME            TEXTO     =  TEXTO  LINHA1
*
* ACHA NUMERO DE BRANCOS A SEREM INSERIDOS
*
EDITA     K        =
TESTA.BRANCO  TEXTO  *LINHA/(N - K)*  *CARACTER/'1'*  /F(LE)
          CARACTER  ' '                   /S(DELETE)
          K  =  K  +  '1'
          ((N - K) - '1')  '-'  /S(ERRO)F(TESTA.BRANCO)
*
```

EDITOR DE TEXTO COMPLETO

```
* DELETA LINHA DO TEXTO
*
DELETE   TEXTO  LINHA  ' '  =
*
* INSERE K BRANCOS
*
         LINHA  ' '           /F(ERRO)
         BR  =  ' '
INICIALIZA          IMPRESSAO  =
SEPARA   LINHA  *CARACTER/'1'*          =
         CARACTER  ' '                  /F(REJUNTA)
         IMPRESSAO =  IMPRESSAO  ' '              /(SEPARA)
REJUNTA  LINHA     =  CARACTER  LINHA
TESTA.K  (K - '1')  '-'       /S(IMPRIME)
         LINHA  *PALAVRA*  BR  =        /F(AUMENTA)
         IMPRESSAO = IMPRESSAO  PALAVRA  BR  ' '
         K  =  K  -  '1'                /(TESTA.K)
AUMENTA  BR  =  BR  ' '
         LINHA  =  IMPRESSAO  LINHA          /(INICIALIZA)
*
* IMPRIME LINHA EDITADA
*
IMPRIME  SYSPOT    =  IMPRESSAO LINHA
          SYSPPT  =  IMPRESSAO  LINHA  /(EDITA)
*
* IMPRIME ULTIMA LINHA
*
ACABOU   SYSPOT  =  TEXTO
          SYSPPT  =  TEXTO             /(END)
*
* MENSAGEM DE ERRO
*
ERRO     SYSPOT    = 'LINHA NAO PODE SER EDITADA.'
END
```

COMPILACAO BEM SUCEDIDA

```
PAGE   1    N17

// JOB    00FF 10FF                                          N17

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        00FF        00FF         0000
  0001        10FF        10FF         0001

V2 M05   ACTUAL 32K  CONFIG 32K

// XEQ SNOBO


-TITLE              TESTE 2  -  FREQUENCIA DAS LETRAS
*
* INICIALIZA
*
      SYSPOT = 'TEXTO'
      SYSPOT = ' '
      ALFABETO = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
*
*  LE
*
LE    LINHA = SYSPIT
      SYSPOT = LINHA
      LINHA  '$$'                          /S(CONTA)
      TEXTO = TEXTO LINHA                  /(LE)
*
* CALCULA AS FREQUENCIAS
*
CONTA TEXTO *CARACTER/'1'* =               /F(PULA)
      $CARACTER = $CARACTER + '1'          /(CONTA)
*
* IMPRIME O RESULTADO
*
PULA    SYSPOT = ' '
        SYSPOT = ' '
IMPRIME      ALFABETO  *CARACTER/'1'*  =  /F(END)
 SYSPOT = 'FREQ. DO CARACTER ' CARACTER ' = ' $CARACTER /(IMPRIME)
END


COMPILACAO BEM SUCEDIDA


TEXTO

         O PRESENTE TRABALHO DESCREVE AS DIRETRIZES SEGUIDAS
NA IMPLANTACAO DE UM COMPILADOR SNOBOL PARA O COMPUTADOR IBM
```

1130.

E DISCUTIDA TODA A ESTRUTURA E ORGANIZACAO DO SISTEMA DO PONTO DE VISTA PRATICO, VISANDO A SUA IMPLANTACAO A CURTO PRAZO.

SAO TAMBEM APRESENTADAS A ESTRUTURA E A LOGICA DO SISTEMA, TANTO NA FASE COMPILACAO COMO NA FASE EXECUCAO, COM UMA DESCRICAO SUCINTA DAS ROTINAS PRINCIPAIS.

ALGUMAS FACILIDADES QUE NAO FORAM IMPLANTADAS NO SISTEMA SNOBOL ORIGINAL SAO DESCUTIDAS DE MANEIRA SUPERFICIAL.

$$

FREQ. DO CARACTER A = 70
FREQ. DO CARACTER B = 5
FREQ. DO CARACTER C = 23
FREQ. DO CARACTER D = 23
FREQ. DO CARACTER E = 33
FREQ. DO CARACTER F = 5
FREQ. DO CARACTER G = 5
FREQ. DO CARACTER H = 1
FREQ. DO CARACTER I = 33
FREQ. DO CARACTER J =
FREQ. DO CARACTER K =
FREQ. DO CARACTER L = 13
FREQ. DO CARACTER M = 19
FREQ. DO CARACTER N = 21
FREQ. DO CARACTER O = 41
FREQ. DO CARACTER P = 15
FREQ. DO CARACTER Q = 1
FREQ. DO CARACTER R = 24
FREQ. DO CARACTER S = 37
FREQ. DO CARACTER T = 27
FREQ. DO CARACTER U = 17
FREQ. DO CARACTER V = 3
FREQ. DO CARACTER W =
FREQ. DO CARACTER X = 1
FREQ. DO CARACTER Y =
FREQ. DO CARACTER Z = 3

```
PAGE   1    N17

// JOB    00FF 10FF                                          N17

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        00FF         00FF        0000
  0001        10FF         10FF        0001
                           0CF1        0002

V2 M05   ACTUAL 32K  CONFIG 32K

// XEQ SNOBO


-TITLE              TESTE  3   -  QUANTIAS EM ALFABETICO
*
*
*        ESTE PROGRAMA FOI RETIRADO DA APOSTILA
*        DE SNOBOL 3 PUBLICADA PELO CBPF.
*
*
INIC     CRZ =
         SYSPIT  *ML/'3'*  *M/'3'*  *UNID/'2'*
         K  =  ML  M  UNID
         '00000000'  K                    /S(INIC)
         Q  =  ML  M
         '000000'  Q                      /S(A99)
*
* CASA DOS MILHARES
*
         '000'  ML              /S(MIL)
         '001'  ML              /F(X1)
         CRZ  =  ' HUM MIL'                /(MIL)
X1       '100'  ML                         /F(X15)
         CRZ  =  ' CEM MIL'                /(MIL)
X15      PT  = ' MIL'
X16      CAD  =  ML
         SAIDA  =  'MIL'                   /(COMUM)
*
* CASA DOS MIL
*
MIL      '000'  M                          /S(A98)
         '001'  M                          /F(X2)
         CRZ  =  CRZ  ' HUM CRUZEIRO'                    /(A99)
X2       '100'  M               /F(X3)
         CRZ  =  CRZ  ' CEM CRUZEIROS'                   /(A99)
X3       PT  =  ' CRUZEIROS'
         CAD  =  M
         SAIDA  =  'A99'                   /(COMUM)
A98      CRZ  =  CRZ  ' CRUZEIROS'
A99      CAD  =  'XXX'
```

TESTE 3 - QUANTIAS EM ALFABETICO

```
          B  =  UNID
          '00'  B                 /S(IMP)
          '01'  B                 /F(X7)
          CRZ  =  CRZ  ' E UM CENTAVO'                    /(IMP)
X7        PT  =  ' CENTAVOS'
          SAIDA  =  'IMP'                   /(COMUM)
IMP       SYSPOT  =  K  ' = '  '('  CRZ  ')'
          SYSPOT  =  ' '                    /(INIC)
COMUM     CAD  'XXX'                        /S(X4)
          CAD  *A/'1'*  *B*
          '0'  A                  /S(X4)
          SD  =  'X4A'
          A  =  A  '00'           /($A)
X4A       CRZ  =  CRZ  C
X4        SD  =  'X6'
          B  *C2/'1'*  *D1*
          '0'  C2                 /S($D1)
          Z  =  '20'  -  B
          '0'  Z                  /F(X29)
          SD  =  'X5A'                      /($B)
X29       Z  '-'                  /F($B)
          SD  =  'X5A'
          C2  =  C2  '0'                    /($C2)
X5A       CRZ  =  CRZ D
          SD  =  'X6'
          '0'  D1                 /S(X23)
          CRZ  =  CRZ  ' E'                 /($D1)
X6        CRZ  =  CRZ  U  PT                /($SAIDA)
X23       CRZ  =  CRZ PT                    /($SAIDA)
0         U  =                              /($SD)
1         U  =       ' UM'                  /($SD)
2         U  =       ' DOIS'                /($SD)
3         U  =       ' TRES'                /($SD)
4         U  =       ' QUATRO'              /($SD)
5         U  =       ' CINCO'               /($SD)
6         U  =       ' SEIS'                /($SD)
7         U  =       ' SETE'                /($SD)
8         U  =       ' OITO'                /($SD)
9         U  =       ' NOVE'                /($SD)
10        U  =       ' DEZ'                 /($SD)
11        U  =       ' ONZE'                /($SD)
12        U  =       ' DOZE'                /($SD)
13        U  =       ' TREZE'               /($S4)
14        U  =       ' QUATORZE'            /($SD)
15        U  =       ' QUINZE'              /($SD)
16        U  =       ' DEZESSEIS'           /($SD)
17        U  =       ' DEZESSETE'           /($SD)
18        U  =       ' DEZOITO'             /($SD)
19        U  =       ' DEZENOVE'            /($SD)
20        D  =       ' VINTE'               /($SD)
```

TESTE 3 - QUANTIAS EM ALFABETICO

```
30        D  =      ' TRINTA'              /($SD)
40        D  =      ' QUARENTA'            /($SD)
50        D  =      ' CINQUENTA'           /($SD)
60        D  =      ' SESSENTA'            /($SD)
70        D  =      ' SETENTA'             /($SD)
80        D  =      ' OITENTA'             /($SD)
90        D  =      ' NOVENTA'             /($SD)
100       C  =      ' CENTO E'             /($SD)
200       C  =      ' DUZENTOS'            /($SD)
300       C  =      ' TREZENTOS'           /($SD)
400       C  =      ' QUATROCENTOS'        /($SD)
500       C  =      ' QUINHENTOS'          /($SD)
600       C  =      ' SEISCENTOS'          /($SD)
700       C  =      ' SETECENTOS'          /($SD)
800       C  =      ' OITOCENTOS'          /($SD)
900       C  =      ' NOVECENTOS'          /($SD)
END
```

```
COMPILACAO BEM SUCEDIDA


00090200 = ( NOVECENTOS DOIS CRUZEIROS)

00011000 = ( CENTO E DEZ CRUZEIROS)

00010001 = ( CEM CRUZEIROS E UM CENTAVO)

00001209 = ( DOZE CRUZEIROS NOVE CENTAVOS)
```

APÊNDICE III

GLOSSÁRIO

DEVIDO À FALTA DE PADRONIZAÇÃO DA NOMENCLATURA EM CIÊNCIA DA COMPUTAÇÃO, APRESENTAMOS A SEGUIR UM GLOSSÁRIO DOS TERMOS MAIS IMPORTANTES COM UMA NOTA EXPLICATIVA OU SEU CORRESPONDENTE EM INGLÊS, ALÉM DE ALGUMAS ABREVIATURAS USADAS NO TEXTO.

CADEIA - STRING - SEQUÊNCIA DE CARACTERES.
CADEIA DE REFERÊNCIA - REFERENCE STRING.
CAMPO DE TRANSFERÊNCIA - GO TO FIELD.
CÓDIGO ALEATORIO - HASH CODE.
COMPARAÇÃO DE PADRÃO - PATTERN MATCHING
DELETAR - DELETE - APAGAR.
DF - DESCRITOR DE FUNÇÃO; VER PARAG. 1.2.7.
DP - DESCRITOR DE PADRÃO; VER PARAG. 1.2.6.
DPP - DESCRITOR DE PADRÃO PROVISÓRIO; VER PARAG. 3.3.9.7.
INICIALIZAR - INITIALIZE - DAR VALOR INICIAL AS VARIÁVEIS.
LISTA DE APONTADORES - LINKED LIST - VER PARAG. 1.2.2.
NÓ - NODE.
PARÂMETRO FORMAL - FORMAL PARAMETER - PARÂMETRO DA DEFINIÇÃO DE UMA SUBROTINA.
PARAMETRO REAL - REAL PARAMETER - PARÂMETRO TRANSMITIDO NA CHAMADA DE UMA SUBROTINA.
PILHA - STACK.
ROTINA - PARTE DE PROGRAMA QUE REALIZA CERTA FUNÇÃO.
RÓTULO - LABEL.
SUBSTITUIÇÃO - ASSIGNMENT.
TS - TABELA DE SÍMBOLOS.
VALOR GLOBAL - VALOR DE UMA VARIÁVEL LOCAL FORA DA SUBROTINA.

APÊNDICE IV

DESCRIÇÃO BNF DA LINGUAGEM SNOBOL IMPLANTADA

```
<VAZIO> ::=
<BR> ::= BRANCO
<LETRA> ::= 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
<DÍGITO> ::= 1|2|3|4|5|6|7|8|9|0
<CARACTER ESPECIAL> ::= (|)|,|.|=|+|-|*|/|$|<BR>
<CARACTER> ::= <LETRA>|<DÍGITO>|<CARACTER ESPECIAL>
<U> ::= <BR>|<U><BR>
<CARACTER DE NOME> ::= <LETRA>|<DÍGITO>|.
<NOME EXPLÍCITO> ::= <CARAC.DE NOME>|<NOME EXPL.><CARAC.NOME
<NOME IMPLÍCITO> ::= $<NOME EXPL.>|$<GRUPAMENTO>
<NOME> ::= <NOME EXPL.>|<NOME IMPLICITO>
<CADEIA> ::= <VAZIO>|<CARACTER>|<CADEIA><CARACTER>
<CONSTANTE> ::= '<CADEIA>'
<OPERADOR> ::= +|-|*|/|**
<OPERADOR ARITMETICO> ::= <U><OPERADOR><U>
<OPERANDO> ::= <NOME>|<CONSTANTE>|<GRUPAMENTO>
<EXPRESSAO ARITMÉTICA> ::= <OPERANDO><OPER.ARIT.><OPERANDO>
<GRUPAMENTO> ::= (<EXPRESSÃO>)
<TERMO> ::= <EXPRESSÃO ARIT.>|<OPERANDO>
<EXPRESSÃO> ::= <TERMO>|<EXPRESSAO><U><TERMO>
<NOME DE RÓTULO> ::= <LETRA>|<DIGITO>|
                     <NOME DE RÓTULO><CARACTER DE NOME>
<RÓTULO> ::= <VAZIO>|<NOME DE RÓTULO>
<CADEIA DE REFERENCIA> ::= <EXPRESSÃO>
<NOME DE ELEMENTO> ::= <VAZIO>|<NOME>
<ELEMENTO A> ::= *<NOME DE ELEMENTO>*
<ELEMENTO AF> ::= *<NOME DE ELEM.>/<EXPRESSÃO>*
<ELEMENTO B> ::= *(<NOME DE ELEM.>)*
<ELEMENTO BF> ::= *(<NOME DE ELEM.>)/<EXPRESSÃO>*
<ELEMENTO K> ::= <NOME>|<CONSTANTE>
<ELEMENTO DE PADRÃO> ::= <ELEM.A>|<ELEM.AF>|<ELEM.B>|
                         <ELEM.BF>|<ELEM.K>
<PADRÃO> ::= <VAZIO>|<ELEM.PADRÃO>|<PADRÃO><U><ELEM.PADRÃO>
<PADRÃO> ::= <VAZIO>|<ELEM.DE PADRÃO>|<PADRÃO><ELEM.DE PADRÃO>
<SUBSTITUIÇÃO> ::= <VAZIO>|<U>=|<U>=<U><EXPRESSÃO>
<NOME DE TRANSFERENCIA> ::= <NOME DE RÓTULO>|<NOME IMPL.>
<CONDIÇÃO DE SUCESSO> ::= S(<NOME DE TRANSF.>)
<CONDIÇÃO DE FALHA> ::= F(<NOME DE TRANSF.>)
<INCONDICIONAL> ::= (<NOME DE TRANSF.>)
<CONDIÇÃO SIMPLES> ::= <COND.SUCESSO>|<COND.FALHA>|<INCON.>
<CONDIÇÃO DUPLA> ::= <COND.SUCESSO><COND.FALHA>|
                     <COND.FALHA><COND.SUCESSO>
<TRANSFERENCIA> ::= <VAZIO>|<U>/<CONDIÇÃO SIMPLES>|
                    <U>/<CONDIÇÃO DUPLA>
<CADEIA OPCIONAL> ::= <VAZIO>|<EXPRESSÃO>
<DECLARAÇÃO COM CAD.OPCIONAL> ::= <RÓTULO><U><CAD.OPCIONAL>
                                  <TRANSFERENCIA>
<DECLARAÇÃO> ::= <RÓTULO><U><CAD.REFER.><U><PADRÃO>
                 <SUBSTITUIÇÃO><TRANSFERENCIA>
<DECLARAÇÃO GENÉRICA> ::= <DECL.C/CAD.OPCIONAL>|<DECLARAÇÃO>
<PROGRAMA SONBOL> ::= END|<DECLAR.GENER.><PROGR.SNOBOL>
```

BIBLIOGRAFIA

1 - D.J.FARBER, R.E.GRISWOLD & I.P.POLONSKY
THE SNOBOL 3 PROGRAMMING LANGUAGE
THE BELL SYSTEM TECHNICAL JOURNAL
JULY-AUGUST 1966

2 - ROBERT L. GLASS
AN ELEMENTARY DISCUSSION OF COMPILER/INTERPRETER WRITING
COMPUTING SURVEYS
MARCH 1969, VOL I NO. 1

3 - ALLEN FORTE
SNOBOL 3 PRIMER
THE MIT PRESS - 1967

4 - A.T.DE MEDEIROS & G.SCHWACHHEIM
MANUAL DE SNOBOL 3 PARA O COMPUTADOR IBM 1620
PUBLICAÇÃO DO CENTRO BRASILEIRO DE PESQUISAS FÍSICAS - 1

6 - DAVID L. WILSON
SNOBOL 3
COMMON USERS GROUP PROGRAM FROM IBM - NO. 1.4.024

7 - D. KNUTH
THE ART OF COMPUTER PROGRAMMING - VOL I
ADDISON-WESLEY PUBLISHING CO. - 1968

8 - ALLEN NEWEL
INFORMATION PROCESSING LANGUAGE V MANUAL
PRENTICE HALL, INC. - 1965

9 - JEAN E. SAMMET
PROGRAMMING LANGUAGES: HISTORY AND FUNDAMENTALS
PRENTICE HALL, INC. - 1969

10- R. MORRIS
SCATTER STORAGE TECHNIQUES
COMMUNICATIONS OF ACM
JANUARY 1968

11- F.H.REAGAN, JR.
THE SURPRISING IBM 1130
DATA PROCESSING MAGAZINE
JULY 1966