ESTUDO DE UM COMPILADOR/INTERPRETADOR RPG

PARA O TERMINAL INTELIGENTE

Luiz Alfredo Soares Garcindo

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

APROVADA POR:

Guilherme Chagas Rodrigues
Presidente

Sueli Mendes dos Santos

Estevam Gilberto De Simone

Rio de Janeiro, RJ - Brasil

MARÇO DE 1982

GARCINDO, LUIZ ALFREDO SOARES

Estudo de um Compilador/Interpretador RPG para o Terminal Inteligente (Rio de Janeiro) 1982.

VII, 215 p. 29,7cm (COPPE - UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1982)

Tese - Universidade Federal do Rio de Janeiro, Faculdade de Engenharia.

1. Assunto: Compiladores e Linguagens Formais I. COPPE/UFRJ II. Título (série).

Para a Gõia e nossos filhos
Tiago, Lucas, Marcos e Luiza

AGRADECIMENTOS

À todos aqueles que direta ou indiretamente colaboraram na realização deste trabalho.

Em particular, ao Professor Guilherme Chagas Rodrigues, pela orientação, e aos Analistas do Núcleo de Computação da UFRJ, José Antônio dos Santos Borges e Paulo César Moraes Melo, pelo apoio recebido.

SINOPSE

Este trabalho é um estudo de um Compilador/Interpretador RPG para o terminal inteligente do Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro.

O estudo parte da definição da Linguagem RPG, adaptada as condições do terminal, e propõe uma configuração lógica de um Compilador/Interpretador para a mesma.

ABSTRACT

This thesis presents the result of a research about a RPG Compiler/Interpreter for an intelligent terminal developed by the Nūcleo de Computação Eletrōnica of Universidade Federal do Rio de Janeiro.

This research includes a definition of the RPG language adequated to the machine and proposes a logical configuration of a Compiler/Interpreter for the referred language.

INDICE

CAPÍTULO I - INTRODUÇÃO

A linguagem RPG foi projetada para facilitar a elaboração de programas pelo usuário, exigindo do mesmo apenas alguns conceitos básicos de processamento de dados. A sua utilização é voltada para a resolução de problemas comerciais. A linguagem utiliza um fluxo padrão de processamento, registro a registro, cabendo ao usuário apenas explicitar as entradas, os cálculos e saídas, relativos ao seu problema. Estas características da linguagem tornam interessante a sua aplicação em computadores de pequeno porte que estão sendo usados atualmente em escala cada vez maior, na área comercial. Mais especificamente, estas características despertaram o interesse do Núcleo de Computação Eletrônica da UFRJ(NCE), de incluir em seus planos o desenvolvimento e implementação de um compilador/interpretador RPG para o terminal Inteligente, equipamento este desenvolvido internamente e montado sobre um microprocessador Intel-8080. Através deste estudo, propomos um modelo básico da linguagem RPG para o Terminal Inteligente, bem como a configuração lógica de um compilador/interpretador para a mesma, procurando indicar uma alternativa para o desenvolvimento do projeto. No capítulo 2 definimos a linguagem apresentando suas especificações.

Uma visão geral dos processos de compilação e interpretação é fornecida no capítulo 3.

No capítulo 4 são apresentadas as especificações do código intermediário proposto.

Apresentamos no capítulo 5 as características do processo de compilação proposto para a linguagem.

No capítulo 6 apresentamos a lógica RPG a ser utilizada e especificações do programa interpretador que executará esta lógica.

Finalmente, no capítulo 7, concluimos o trabalho indicando alguns aspectos que oferecem possibilidades de aperfeiçoamento futuro, bem como fazemos indicação de estudos complementares necessários ao desenvolvimento do projeto.

CAPÍTULO II - ESPECIFICAÇÕES DA LINGUAGEM

2.1. INTRODUÇÃO

A linguagem RPG é voltada para a resolução de problemas comerciais. Ela oferece ao usuário uma série de facilidades, tanto na geração de relatórios como na criação e atualização de arquivos. Um programa em RPG pode ser considerado como um conjunto de declarações de entradas, cálculos e saídas para um fluxo padrão de processamento denominado LÓGICA RPG, com processamento registro a registro.
A codificação é orientada através da utilização de formulários que permitem uma visualização nítida das partes entrada, processamento e saída.
Ao definir a linguagem, nos preocupamos em adaptá-la às condições de uso do Terminal Inteligente, num modelo básico com possibilidade de ampliação.
Na definição, assinalamos com a palavra "PREVISÃO" itens considerados como desejáveis na linguagem que dependerão de novos estudos ou de maiores recursos no Terminal Inteligente.
Entre as diversas facilidades que a linguagem poderá oferecer ao usuário, destacamos:

- o fácil manuseio de entrada e saída, graças à versatilidade do sistema operacional que permite a liberdade de escolha de periféricos sem necessidade de recompilação de programas, e também a concatenação de arquivos.
- a possibilidade de chamar rotinas externas em PLTI ou Assembler.

2.2. ELEMENTOS BÁSICOS

2.2.1. FORMULÁRIOS UTILIZADOS

O modelo proposto para a linguagem utilizará cinco tipos de formulários de codificação, que são:

Formulário de especificações de controle - descrição de arquivos.

Figura II-1

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

GX 17-0047-3 U/M 050
Impresso no Brasil

*No. of forms per pad may very slightly

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic								Card Electro Number
Programmer	Date		Punch								

Page [1][2] of __ Program Identification [75][76][77][78][79][80]

Control Card Specifications

H

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	Debug	MFCM Stacking Sequence	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Model 20: Address to Start	Work Tapes	Overlay Open	Overlap Printer	Binary Search	Tape Error	2152 Checking	Inquiry	Model 20: Read/Write/Compute	Keyboard Output	Sim Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters		Table Load Halt	Shared I/O	Field Print	Formatted Core Dump	RPG to RPG II Conversion	Refer to the specific System Reference Library manual for actual entries.
3 4 5	6	7 8 9	10	11	12 13 14	15	16 17 18 19 20	21	22	23 24 25	26	27 28 29 30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	H																																

File Description Specification

F

Line	Form Type	Filename	File Type: I/O/U/C/D	P/S/C/R/T/D	File Designation: E	End of File: A/D	Sequence: F/V/S/M/D	File Format	Block Length	Record Length	Mode of Processing: L/R	Length of Key Field or of Record Address Field	A/P/I/K	Record Address Type	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8	A/U	R/U/N
																							Continuation Lines: K	Option	Entry							
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27		28	29 30	31	32	33 34	35 36 37 38	39	40 41 42 43 44 45 46	47 48 49 50 51 52			53	54 55 56 57 58 59	60 61 62 63 64 65		66	67 68 69	70	71 72	73 74		
0 2	F																															
0 3	F																															
0 4	F																															
0 5	F																															
0 6	F																															
0 7	F																															
0 8	F																															
0 9	F																															
1 0	F																															
	F																															
	F																															

72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Formulário de especificações de extensão/linhas.

Figura II-2

IBM

IBM do Brasil Ltda.

GX17-0046-2
U/M 050

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page (1 2)

Program Identification (75 76 77 78 79 80)

Extension Specifications

Line (3 4 5)	Form Type (6)	Record Sequence of the Chaining File (7 8)	Number of the Chaining Field (9 10)	From Filename (11 12 13 14 15 16 17 18)	To Filename (19 20 21 22 23 24 25 26)	Table or Array Name (27 28 29 30 31 32)	Number of Entries Per Record (33 34 35)	Number of Entries Per Table or Array (36 37 38 39)	Length of Entry (40 41 42)	P = Packed/B = Binary (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46 47 48 49 50 51)	Length of Entry (52 53 54)	P = Packed/B = Binary (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74)
01	E																	
02	E																	
03	E																	
04	E																	
05	E																	
06	E																	
07	E																	
08	E																	
09	E																	
10	E																	

Line Counter Specifications

Line (3 4 5)	Form Type (6)	Filename (7 8 9 10 11 12 13 14)	1 Line Number (15 16 17)	1 FL or Channel Number (18 19)	2 Line Number (20 21 22)	2 OL or Channel Number (23 24)	3 Line Number (25 26 27)	3 Channel Number (28 29)	4 Line Number (30 31 32)	4 Channel Number (33 34)	5 Line Number (35 36 37)	5 Channel Number (38 39)	6 Line Number (40 41 42)	6 Channel Number (43 44)	7 Line Number (45 46 47)	7 Channel Number (48 49)	8 Line Number (50 51 52)	8 Channel Number (53 54)	9 Line Number (55 56 57)	9 Channel Number (58 59)	10 Line Number (60 61 62)	10 Channel Number (63 64)	11 Line Number (65 66 67)	11 Channel Number (68 69)	12 Line Number (70 71 72)	12 Channel Number (73 74)
11	L																									
12	L																									
13	L																									

Impresso no Brasil

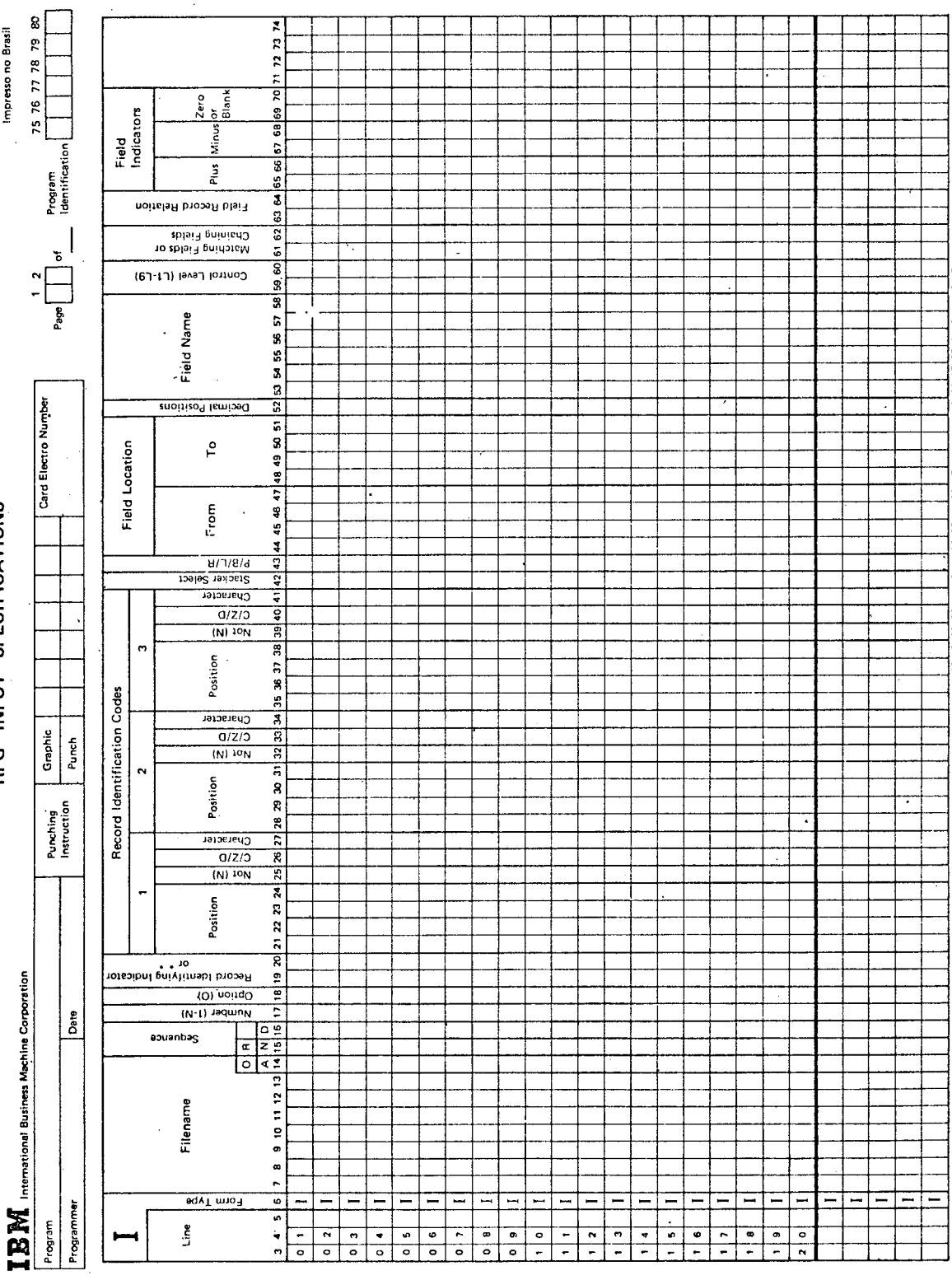


Figura II-3

Formulário de especificações de entrada.

GX 17-0048-2 UM/050
Impresso no Brasil

*No. of forms per pad may vary slightly

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Program		Punchig Instruction	Graphic								Card Electro Number
Programmer	Date		Punch								

Page 1 2 of ____ Program Identification 75 76 77 78 79 80

C	Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup (Factor 2) is: High / Low / Equal	Comments
	3 4 5	6	7 8	9 10 11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55 / 56 57 / 58 59	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
	0 1	C											
	0 2	C											
	0 3	C											
	0 4	C											
	0 5	C											
	0 6	C											
	0 7	C											
	0 8	C											
	0 9	C											
	1 0	C											
	1 1	C											
	1 2	C											
	1 3	C											
	1 4	C											
	1 5	C											
	1 6	C											
	1 7	C											
	1 8	C											
	1 9	C											
	2 0	C											
		C											
		C											
		C											
		C											
		C											

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72

Figura II-4 Formulário de especificações de cálculos.

IBM do Brasil Ltda.

RPG OUTPUT SPECIFICATIONS

GX17-0045-2 U/M 050
Impresso no Brasil

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page [ ][ ] of [ ] Program Identification [ ][ ][ ][ ][ ][ ]

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Constant or Edit Word

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R
	O		O A / R N	D	A	D	D		Not	Not Not	*AUTO				
01	O														
02	O														
03	O														
04	O														
05	O														
06	O														
07	O														
08	O														
09	O														
10	O														
11	O														
12	O														
13	O														
14	O														
15	O														
16	O														
17	O														
18	O														
19	O														
20	O														
	O														
	O														
	O														
	O														

*No. of forms per pad may very slightly

Figura II-5

Formulário de especificações de saída.

a) Formulário de especificações de controle - contendo informações relativas a identificação do programa. A utilização deste formulário é opcional - no caso de não utilização o sistema assumirá uma identificação padrão (figura II-1).

b) Formulário de especificações de descrição de arquivos - contendo informações relativas aos arquivos a serem utilizados no programa (figura II-1).

c) Formulário de especificações de extensões/linhas - contendo informações relativas a utilização da impressora, tamanho do formulário e linha de overflow. A utilização é opcional - se não utilizado o sistema assumirá tamanho e linha de overflow padrão (figura II-2).

d) Formulário de especificações de entrada - utilizado para a descrição dos registros e campos, relativos a cada arquivo de entrada ou atualização do programa (figura II-3).

e) Formulário de especificações de cálculo - será utilizado se o programa necessitar de operações de cálculos (figura II-4).

f) Formulário de especificações de saída - será utilizado para a especificação das saídas necessárias do programa (figura II-5).

2.2.2. DESCRIÇÃO DE TERMOS

a) Caráteres válidos - todos do conjunto ASCII.

b) Nome válido em RPG - nomes de arquivos, campos e rótulos devem obedecer às seguintes condições:

   - Ter o tamanho máximo de 6 caracteres.
   - O primeiro caráter deve ser alfabético, os demais podem ser alfabéticos ou numéricos, não sendo permitido caráter branco no meio dos demais.

c) Campos numéricos - campos que possuem especificação de posições decimais nos formulários de especificações.

d) Campos alfanuméricos - campos que não possuem especificação de posições decimais nos formulários RPG.

2.2.3. LÓGICA RPG - VISÃO BÁSICA

Todo o programa em RPG é executado dentro de um padrão de lógica RPG. A figura abaixo mostra a lógica RPG para o caso de programa relativamente simples envolvendo apenas um arquivo de entrada. Três são os passos básicos em cada programa :

a) Ler informação de entrada.

b) Execução dos cálculos (processamento).

c) Execução das saídas.

De acordo com a lógica RPG, cálculos e saídas são efetuados em dois diferentes tempos num ciclo de programa: tempo de detalhe e tempo de total.

OPERAÇÕES EM TEMPO DE TOTAL

As operações de cálculos ou saídas em tempo de total são normalmente executadas para um grupo relacionado de registros que constituem o que denominamos grupo de controle. As operações são efetuadas quando há uma quebra de controle, ou seja, quando um campo escolhido como campo de controle sofre alteração de conteúdo na passagem de processamento de um registro para o seguinte (passagem de ciclo).
As operações de cálculo em tempo de total são identificadas no formulário específico, colocando-se um indicador do tipo L (indicador de nível) nas colunas 7-8. As operações de saída em tempo de total são identificadas no formulário específico, colocando-se um "T" na coluna 15.

OPERAÇÕES EM TEMPO DE DETALHE

Operações de cálculos não condicionadas por indicadores de nível nas colunas 7-8, são chamadas cálculos detalhe. Operações de saída especificadas com as letras "H" ou "D" na coluna 15 do respectivo formulário, são chamadas saídas detalhe.

As operações detalhe são executadas para cada registro, satisfeitas as condições previstas por indicadores.
Num ciclo de programa, as operações detalhe são executadas após as operações de total. Operações de total manipulam com dados acumulados de registros anteriores. Operações de detalhe, no registro que ocasionou uma quebra de controle, são efetuadas após o término das operações de total.

OBSERVAÇÕES SOBRE O FLUXO LÓGIGO SIMPLIFICADO (conforme figura II-6).

Passo 1.

Antes do primeiro registro ser lido é feita a impressão de linhas de cabeçalho de 1a. página. Após a leitura do 1º registro, neste passo são impressas as linhas de cabeçalho e detalhe ("H" ou "D" na posição 15 do formulário de saída). Linha "H" ou "D"-saídas detalhe.

Passo 2.
São desligados os indicadores de uso pelo programador.

Passo 3.
Lê um registro, faz a identificação e o indicador associado ao registro é ligado, para uso posterior no ciclo em processamento.

Passos 4 e 11.
Verifica se o registro lido é o último do arquivo.
Se for último registro, liga indicador de último registro (LR)e indicadores de controle de nível e encaminha para cálculo e saída de total.

Passos 5 e 12.
Verifica se houve quebra de controle. Se houve liga indicadores de nível apropriados.

Passo 6.
Verifica se o ciclo vigente é o primeiro. Se for, não executa cálculos e saída de total.

Passo 7.
Se o indicador de último registro estiver ligado neste momento,

FLUXO SIMPLIFICADO: LÓGICA RPG

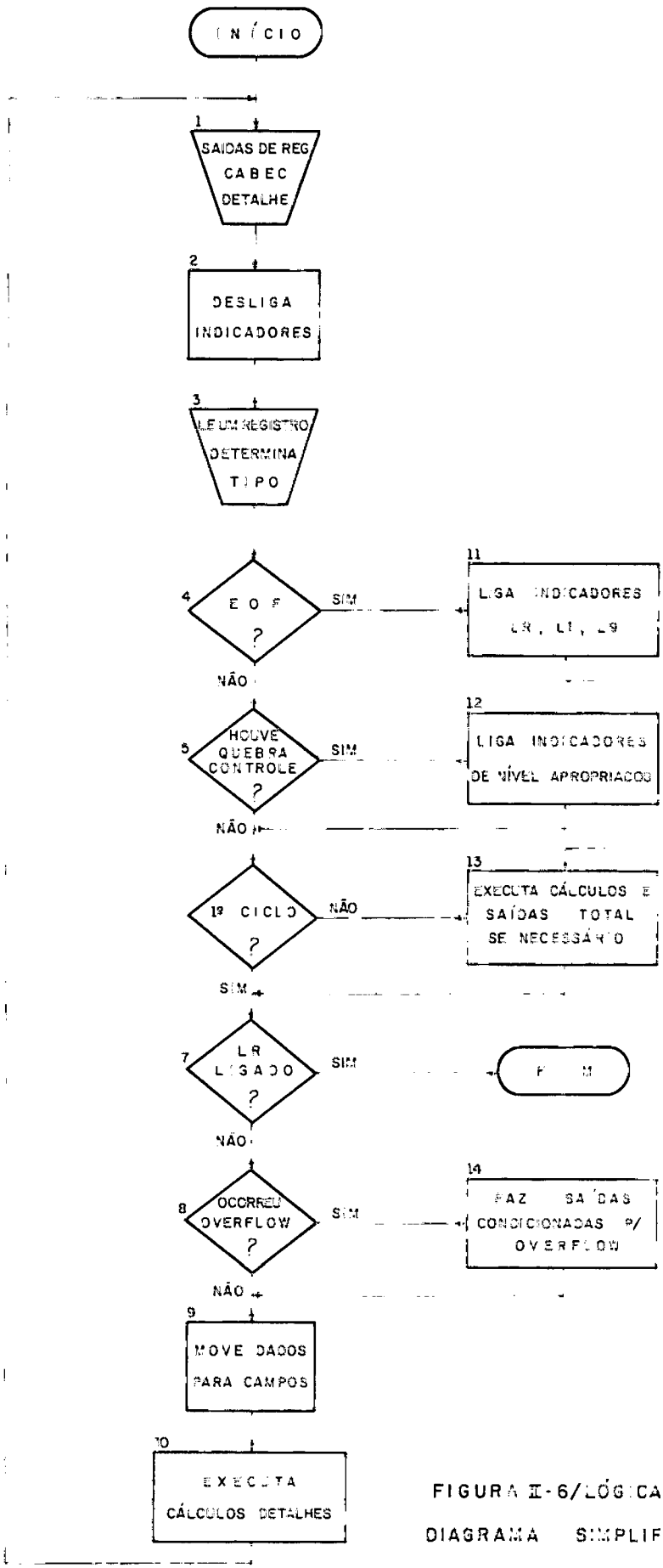


FIGURA II-6/LÓGICA RPG

DIAGRAMA SIMPLIFICADO

encerra o processamento.

Passo 8.
Se for atingida a linha de overflow na impressão de relatórios, executa saídas condicionadas pelo indicador de overflow.

Passo 9.
Move para áreas reservadas aos campos, o conteúdo correspondente obtido no último registro lido.

Passo 10.
Executa operações de cálculo detalhe (não condicionadas por indicadores de nível - quebra de controle).

Passo 13.
Executa cálculos e saídas de total (cálculos e saídas condicionadas por indicadores de nível) se houve quebra de controle.

NOTA: Maiores detalhes relacionados com a lógica RPG podem ser vistos na parte deste trabalho que trata do programa de interpretação.

2.2.4. O TERMINAL INTELIGENTE - CONFIGURAÇÃO

a) O Terminal Inteligente do NCE conta com a seguinte configuração básica:
   - Unidade central - contando com microprocessador INTEL-8080 com até 64K bytes de memória (4K ROM - 60K RAM)
   - Unidade de disco com capacidade para 2,4 MB.
   - Teclado, vídeo, impressora.

b) Software disponível.
   - Montador assembler.
   - Biblioteca para guardar programas objeto.
   - Programa ligador (REFEX) - editor de referências externas.
   - Carregador de programas (CPROG).
   - Editor de textos interativo.
   - Compilador para a linguagem PLTI, usada para desenvolvimento de Software, com as seguintes características:
     - Dois tipos de variáveis: Byte e Address.
     - Aceite vetores de 1 dimensão.

- Variável BASED (para endereçamento indireto).
- Definição hierarquizada de variáveis e rótulos, definição de variáveis globais.
- Comandos estruturados.
- Facilidades de depuração.
- Permite ligação com rotinas em Assembler.
- Permite compilação de subrotinas em separado.
- Comandos de entrada e saída READ, WRITE, SEEK etc.
- Permite a definição de arquivos e seus atributos (processamento sequencial e direto).

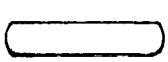
- Rotinas de entrada e saída.
- Programas utilitários.

2.3. DIAGRAMA SINTÁTICO

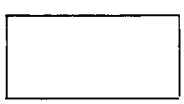
Tendo em vista as características de linguagem optamos pela representação da sintaxe através de um diagrama sintático, com representação do fluxo linha a linha, coluna a coluna.

Este gráfico não só permitirá uma visão completa da linguagem , mas também, facilitará o entendimento do processo de reconhecimento, a ser adotado pelo programa compilador (figura II-7).

Convenções para simbologia:

 - representa uma sequência de caracteres-símbolo terminal.

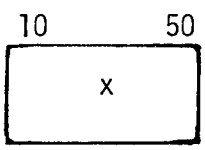
 - representa um caractere-símbolo terminal.

 - representa uma variável sintática que é definida por outro diagrama sintático.

 - representa quais os símbolos podem ser concatenados.

NOTA: Como a codificação nos diversos formulários é feita coluna a coluna, usamos apontadores início e fim do campo, quando for necessário indicar o posicionamento no formulário.

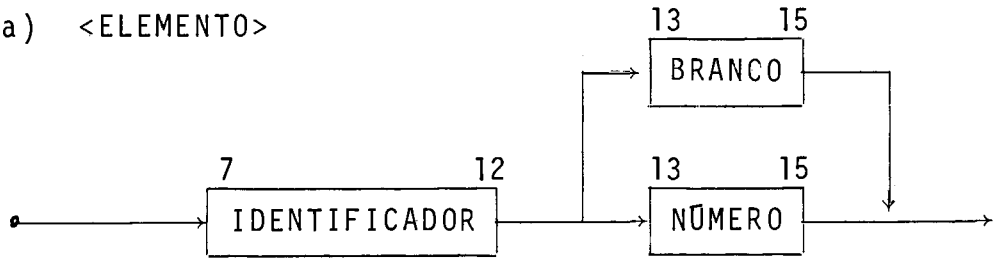
Assim, por exemplo, temos:



- x variável sintática representando conteúdo do campo com início na coluna 10 e fim na coluna 50.

Exemplos:

a) <ELEMENTO>



ELEMENTO é um identificador colocado nas colunas 7 a 12 seguidas por BRANCO ou número nas colunas 13 a 15.

b) Continuação de fluxo na linha seguinte:

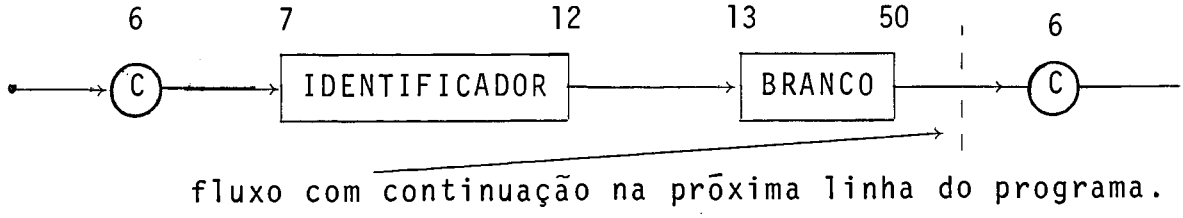
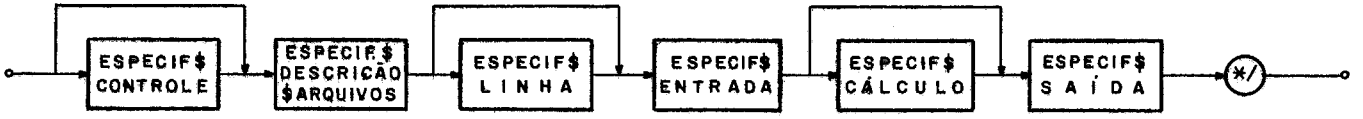
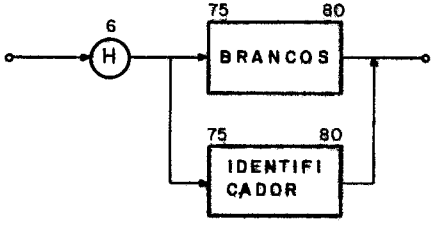


DIAGRAMA SINTÁTICO

⟨ PROGRAMA ⟩



⟨ ESPECIF$CONTROLE ⟩



⟨ ESPECIF$LINHA ⟩

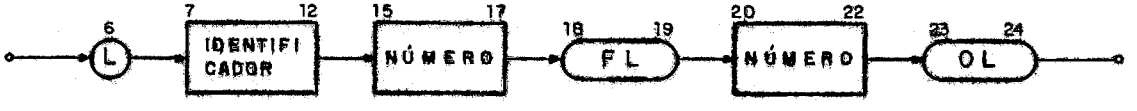


FIGURA II-7/ DIAGRAMA SINTÁTICO

DIAGRAMA SINTÁTICO

⟨ ESPECIF$DESCRIÇÃO$ARQUIVOS ⟩

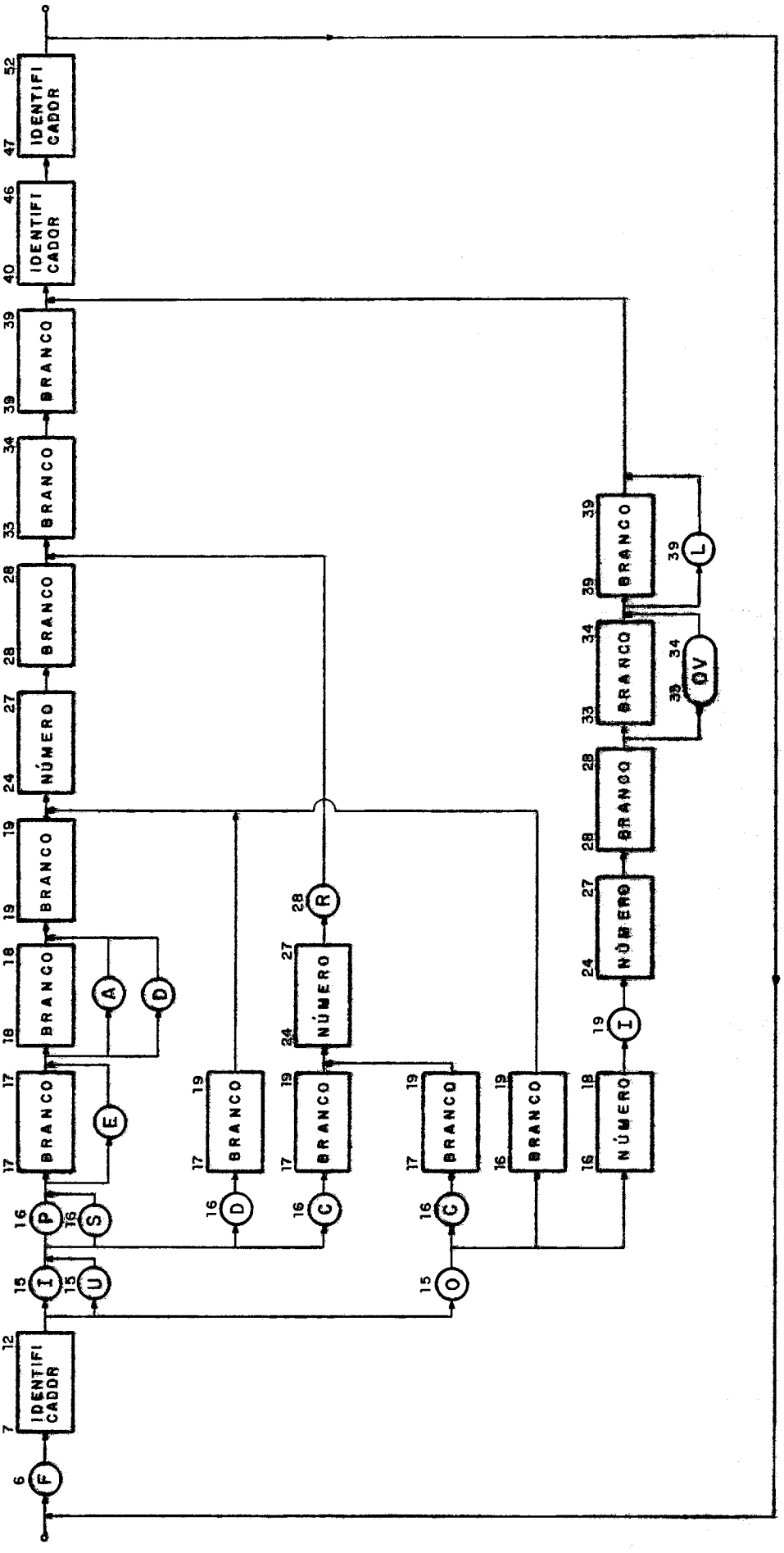
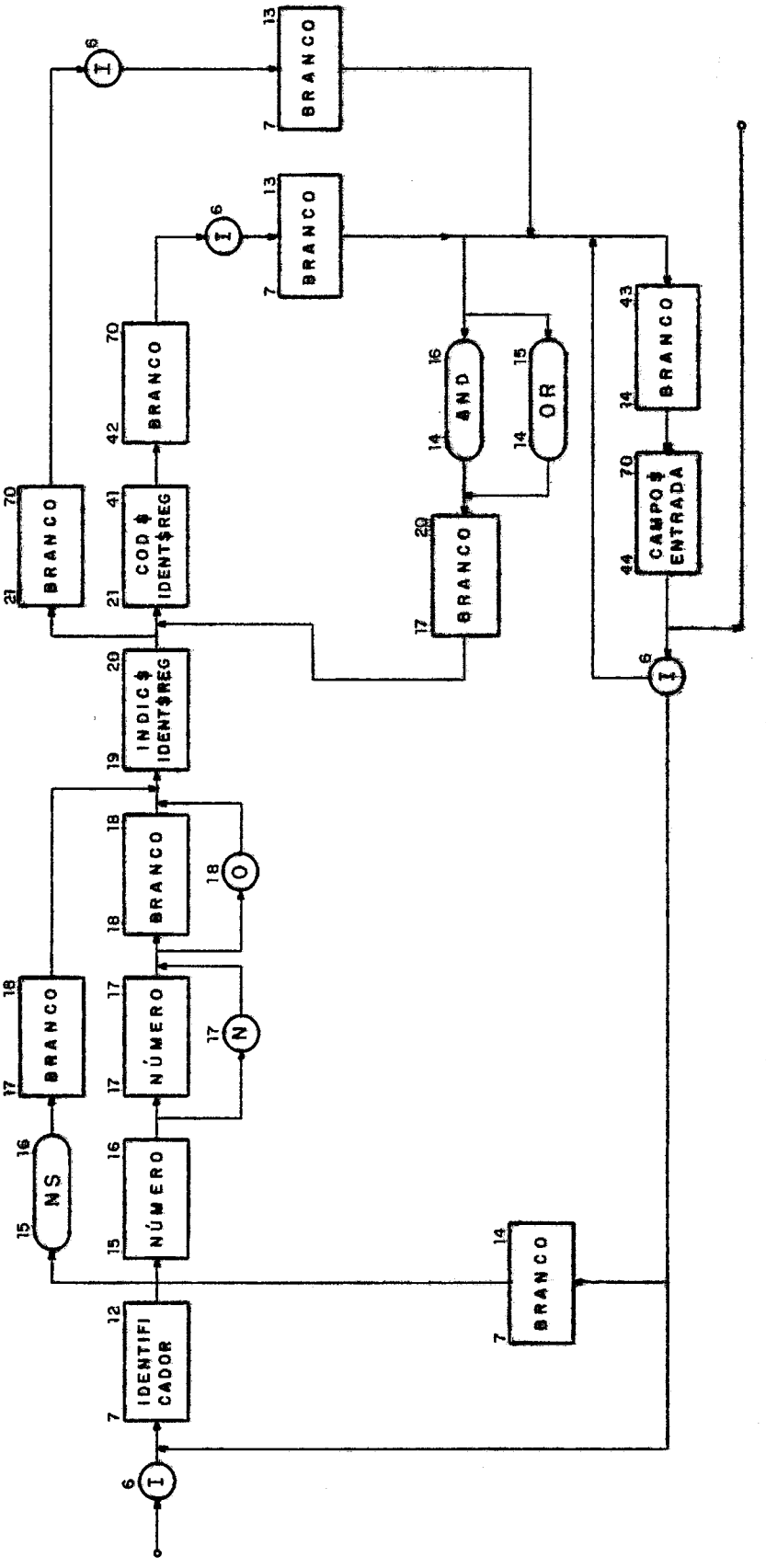


DIAGRAMA SINTÁTICO

< ESPECIF$ENTRADA >



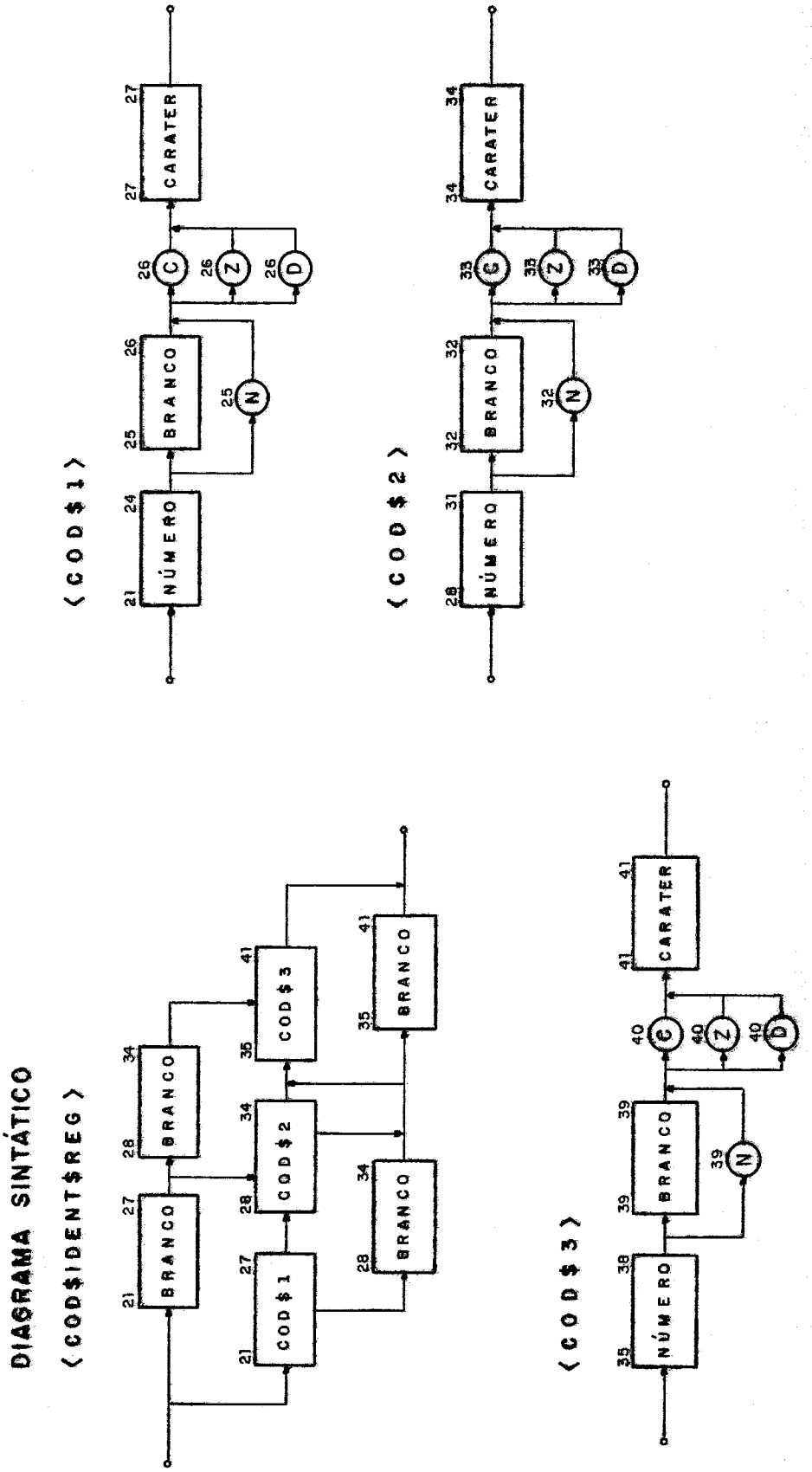
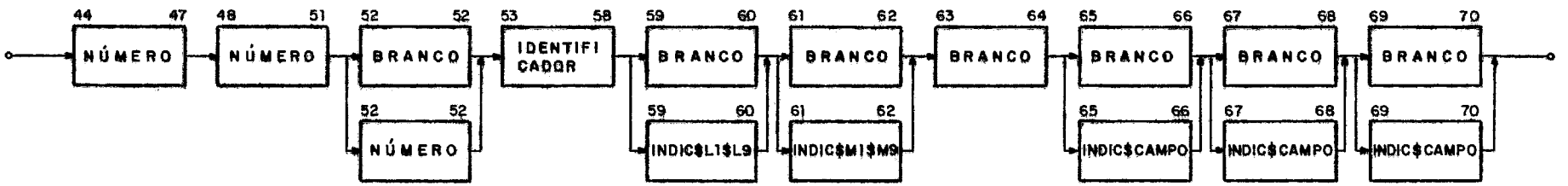


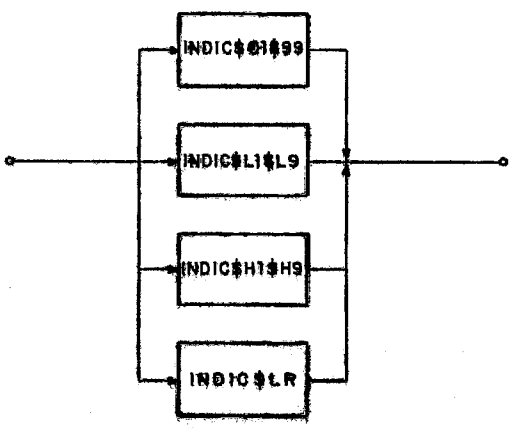
DIAGRAMA SINTÁTICO
⟨ COD$IDENT$REG ⟩
BRANCO
COD$1
BRANCO
COD$2
BRANCO
COD$3
BRANCO
BRANCO
⟨ COD$1 ⟩
NÚMERO
BRANCO
C
Z
D
N
CARATER
⟨ COD$2 ⟩
NÚMERO
BRANCO
C
Z
D
N
CARATER
⟨ COD$3 ⟩
NÚMERO
BRANCO
C
Z
D
N
CARATER

DIAGRAMA SINTÁTICO

⟨ CAMPO$ENTRADA ⟩



⟨ INDIC$IDENT$REG ⟩



⟨ INDIC$CAMPO ⟩

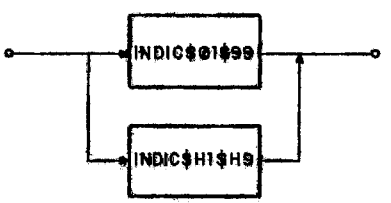


DIAGRAMA SINTÁTICO

⟨ESPECIF$SAÍDA⟩

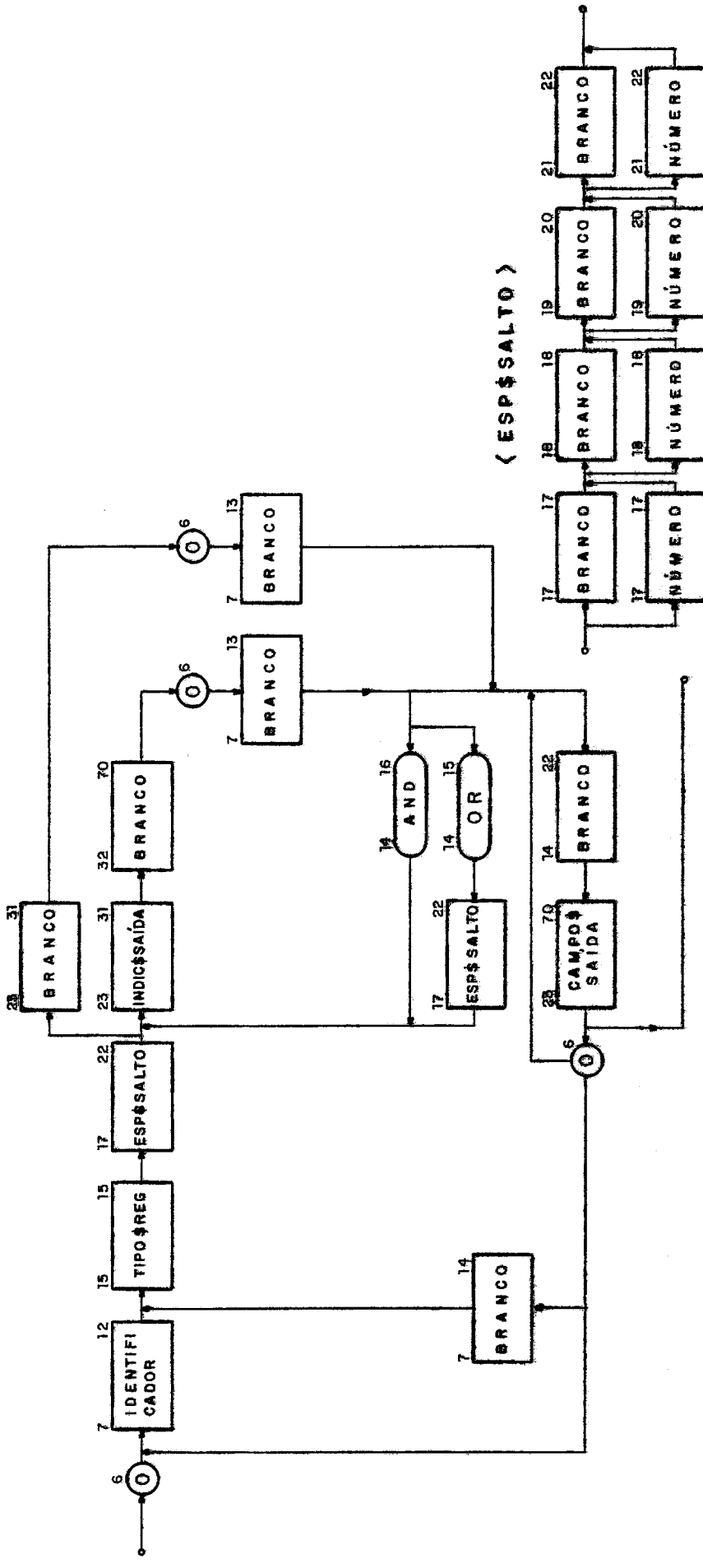
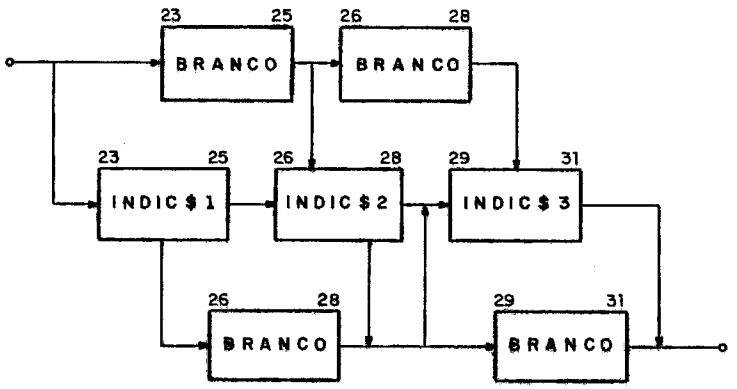
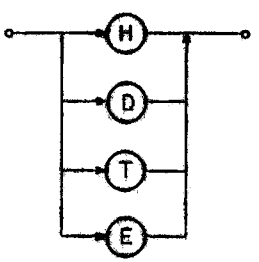


DIAGRAMA SINTÁTICO

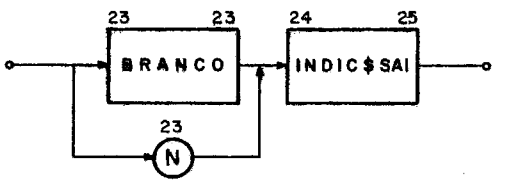
⟨ INDIC$SAÍDA ⟩



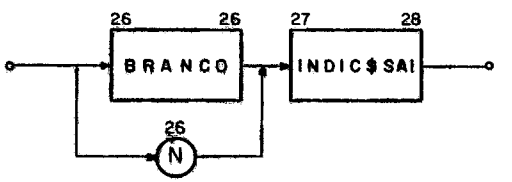
⟨ TIPO$REG ⟩



⟨ INDIC$1 ⟩



⟨ INDIC$2 ⟩



⟨ INDIC$3 ⟩

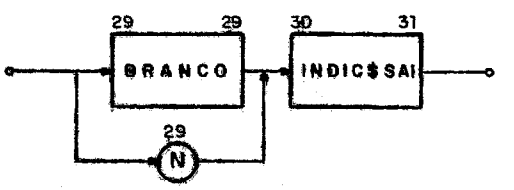
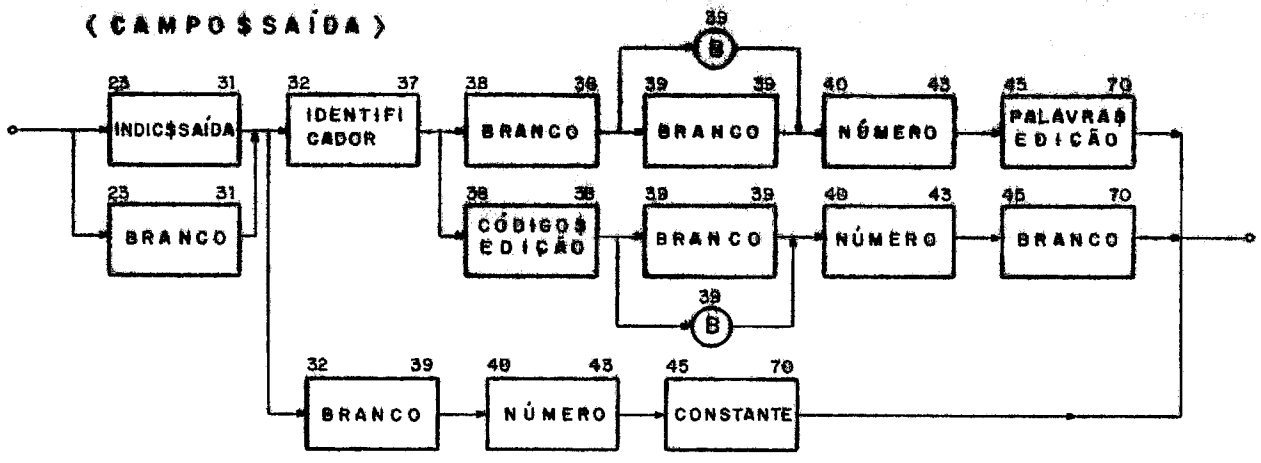
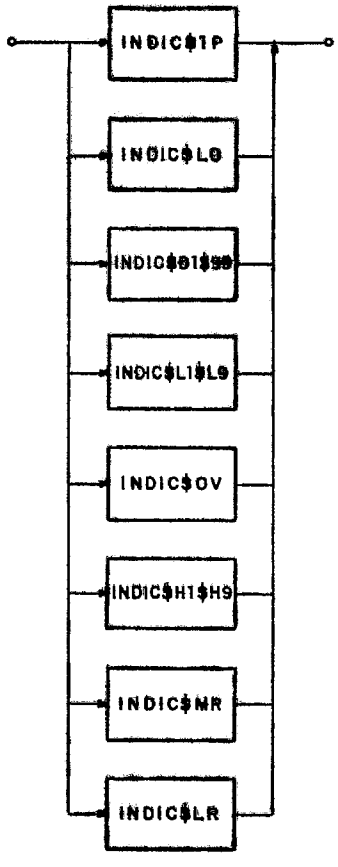


DIAGRAMA SINTÁTICO

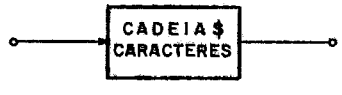
⟨ CAMPO $ SAÍDA ⟩



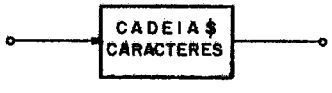
⟨ INDIC$SAI ⟩



⟨ PALAVRA $ EDIÇÃO ⟩



⟨ CONSTANTE ⟩



⟨ CODIGO $ EDIÇÃO ⟩

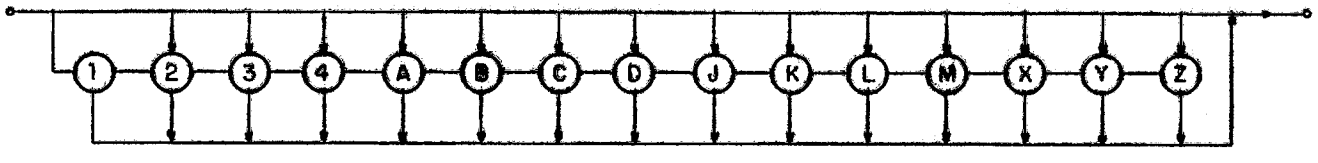
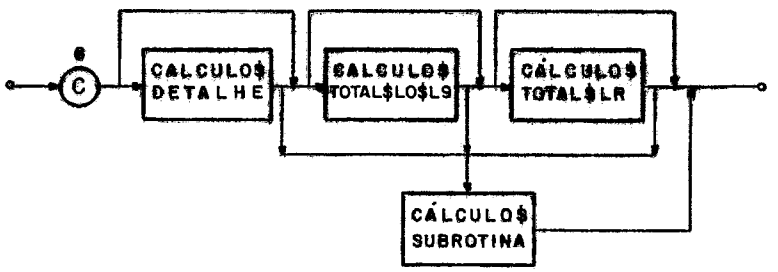
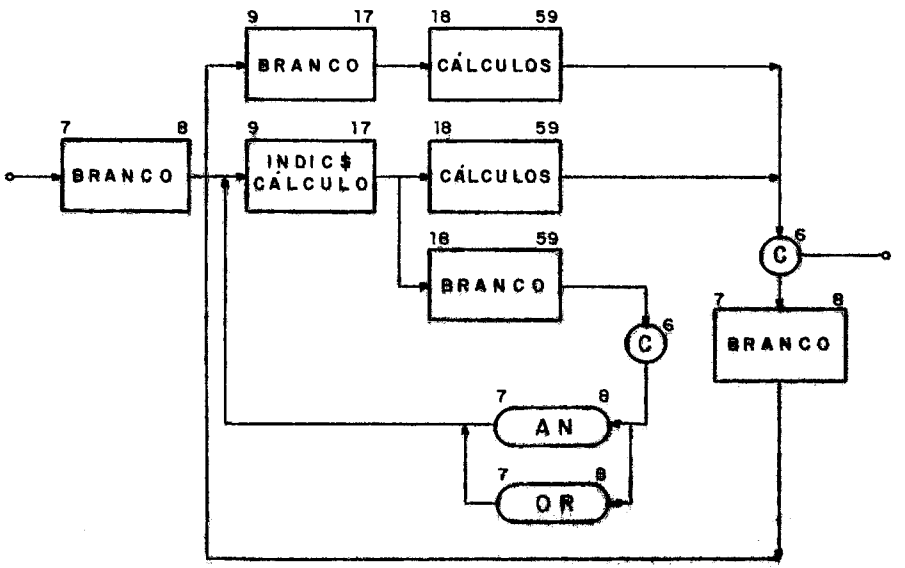


DIAGRAMA SINTÁTICO

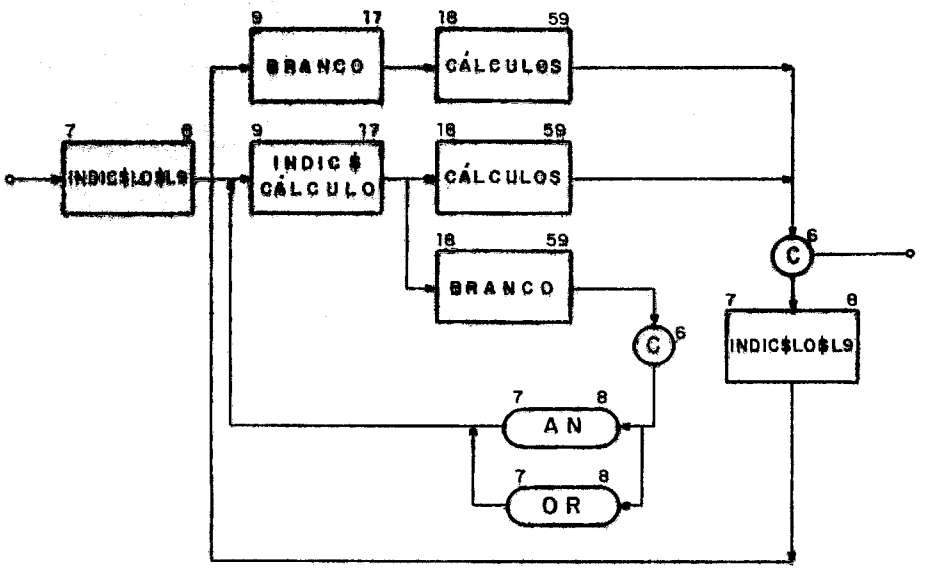
⟨ ESPECIF$CÁLCULOS ⟩



⟨ CÁLCULO$DETALHE ⟩



⟨ CÁLCULO$TOTAL$LO$L9 ⟩



⟨ CÁLCULO$TOTAL$LR ⟩

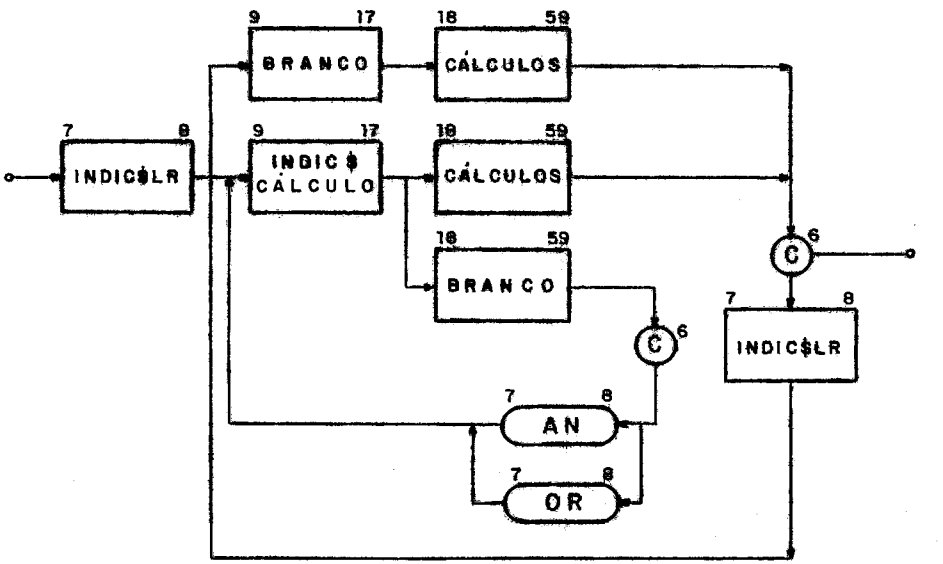
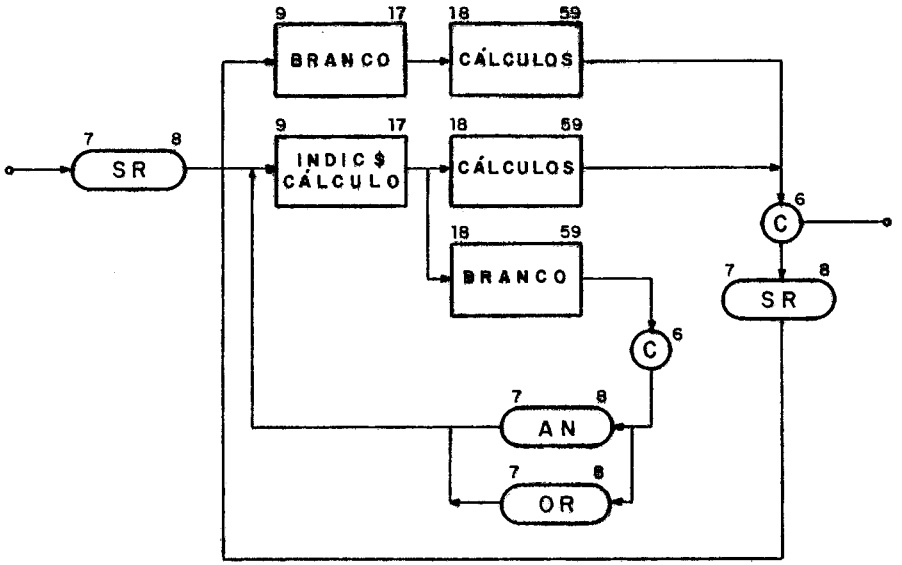
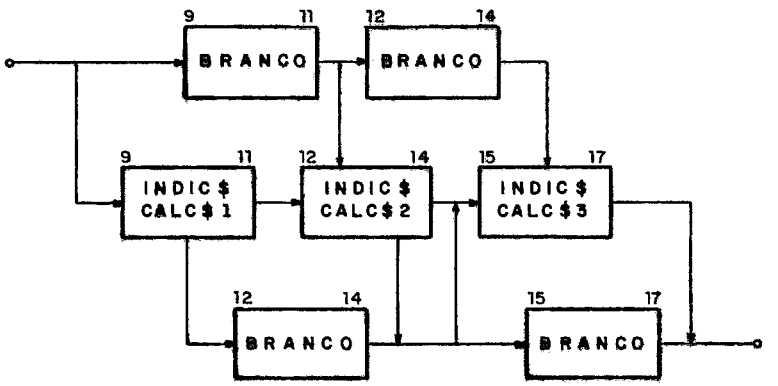


DIAGRAMA SINTÁTICO

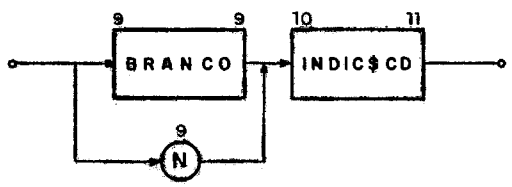
⟨ CÁLCULO $ SUBROTINA ⟩



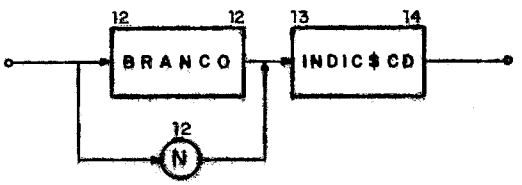
⟨ INDIC$CÁLCULO ⟩



⟨ INDIC$CALC$1 ⟩



⟨ INDIC$CALC$2 ⟩



⟨ INDIC$CALC$3 ⟩

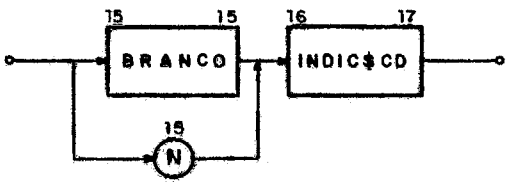
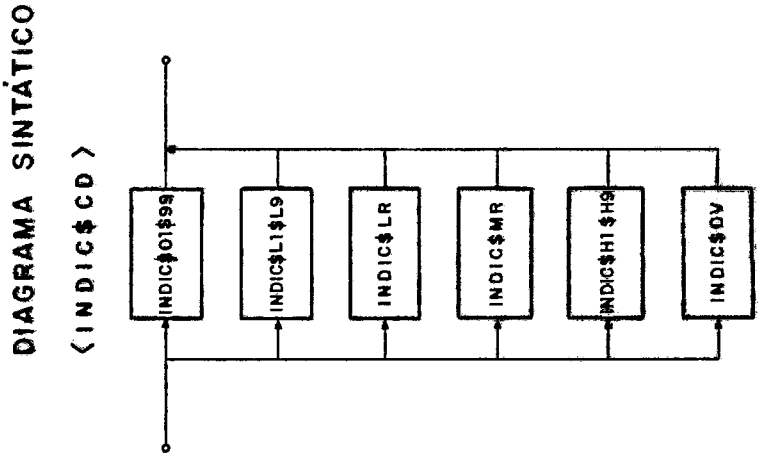


DIAGRAMA SINTÁTICO



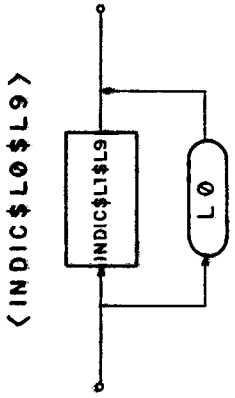
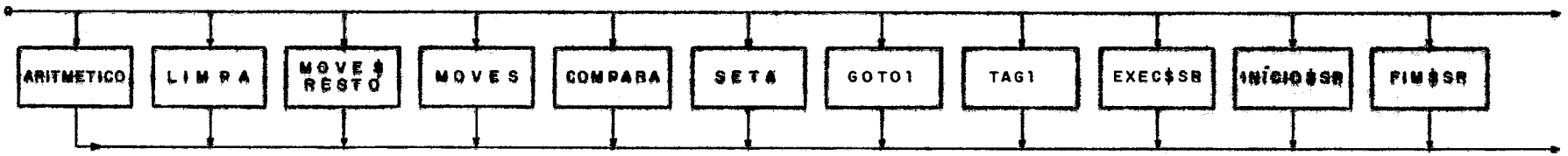
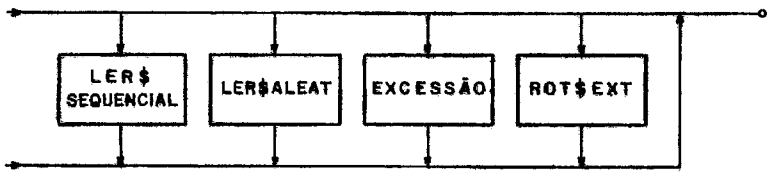


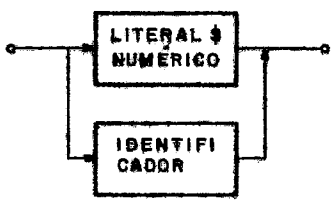
DIAGRAMA SINTÁTICO

⟨ CÁLCULOS ⟩

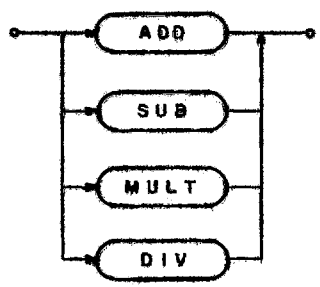




⟨ FATOR$ARITM ⟩

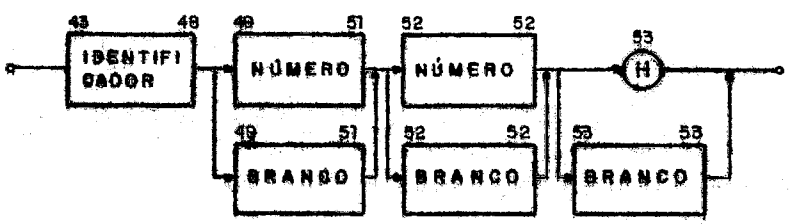


⟨ OPER$ARITM ⟩



⟨ ARITMETICO ⟩

⟨ RESULT$ARITM ⟩



⟨ INDICADOR$RESULTADO ⟩

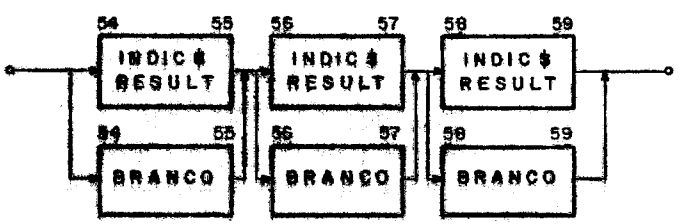
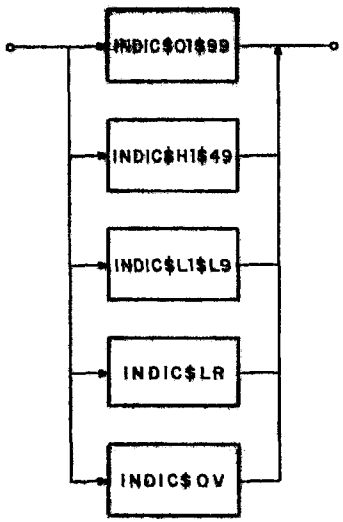
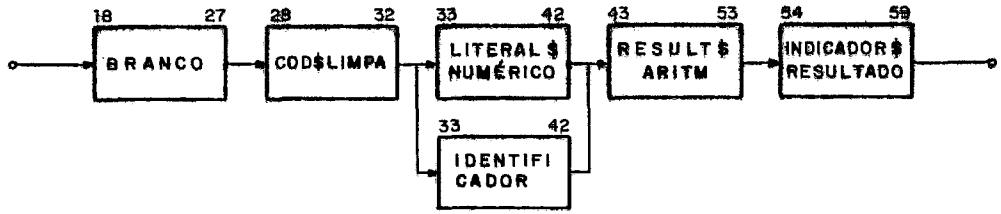


DIAGRAMA SINTÁTICO

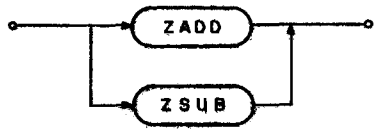
⟨ INDIC$RESULT ⟩



⟨ LIMPA ⟩



⟨ COD$LIMPA ⟩



⟨ MOVE$RESTO ⟩

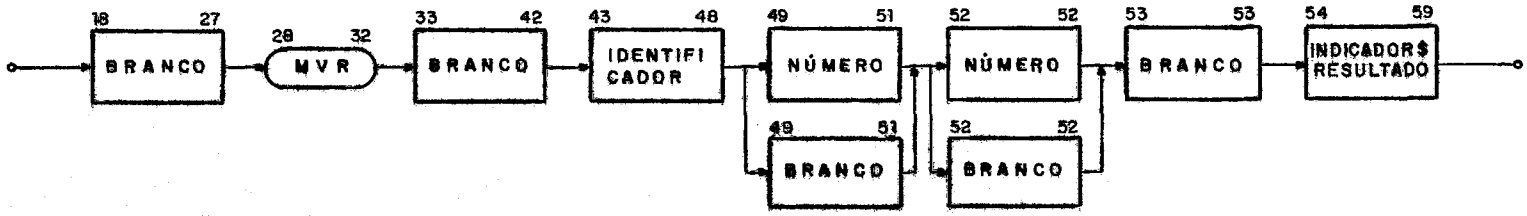
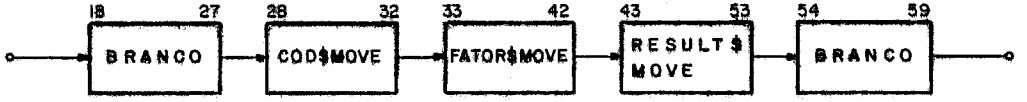
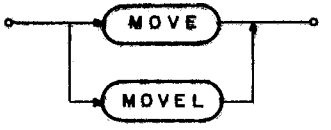


DIAGRAMA SINTÁTICO

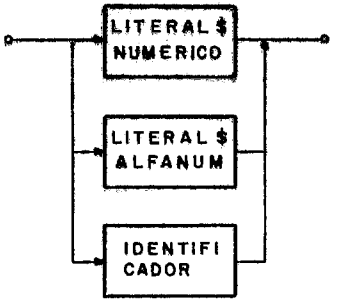
⟨ MOVES ⟩



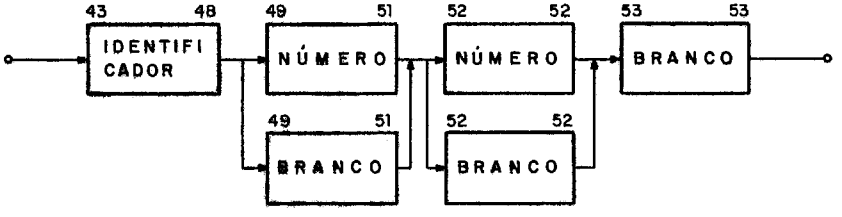
⟨ COD$MOVE ⟩



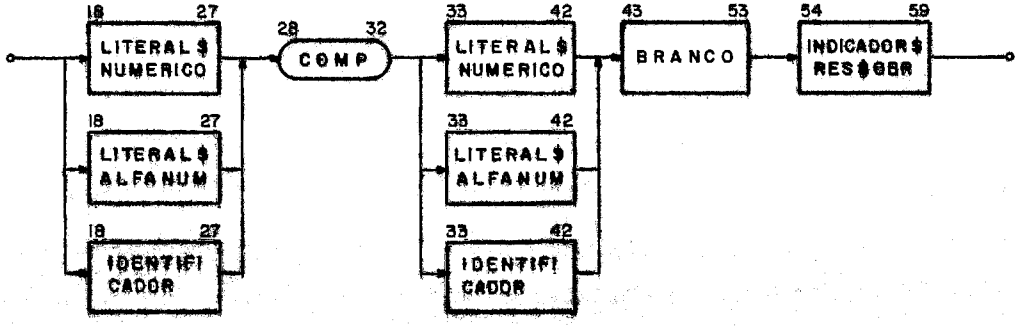
⟨ FATOR$MOVE ⟩



⟨ RESULT$MOVE ⟩



⟨ COMPARA ⟩



⟨ INDICADORRESOBR ⟩

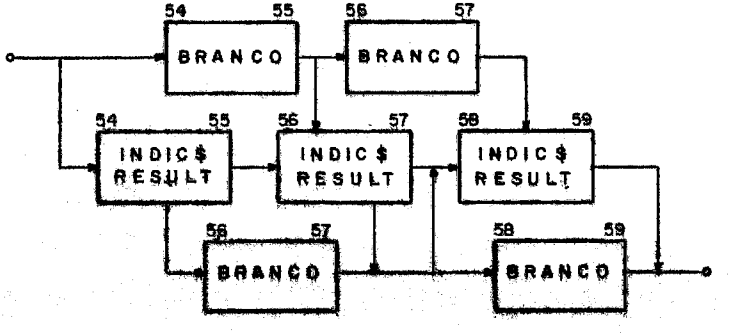
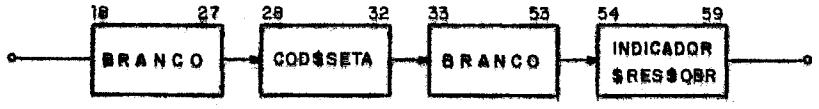
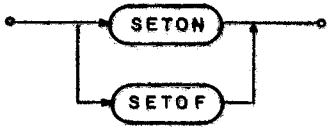


DIAGRAMA SINTÁTICO

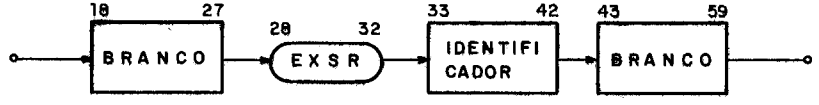
⟨ SETA ⟩



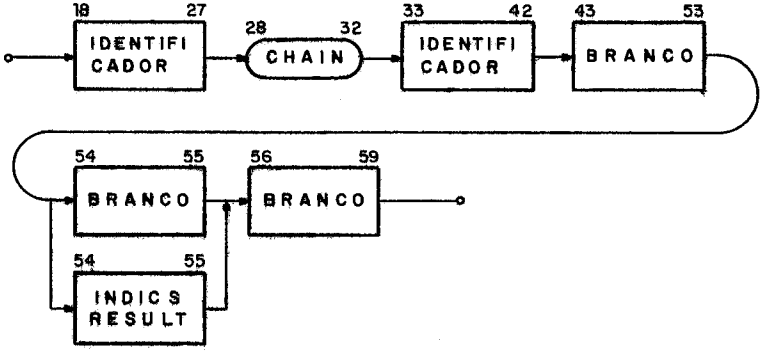
⟨ COD $ SETA ⟩



⟨ EXEC$SR ⟩

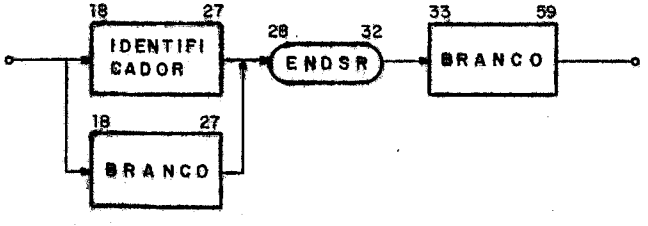


⟨ LER $ ALEAT ⟩

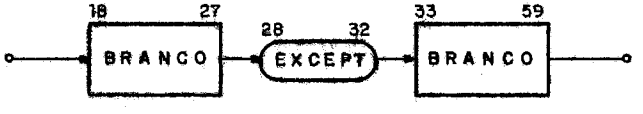


⟨ INÍCIO $ SR ⟩

⟨ FIM $ SR ⟩



⟨ EXCESSÃO ⟩



⟨ LER $ SEQUENCIAL ⟩

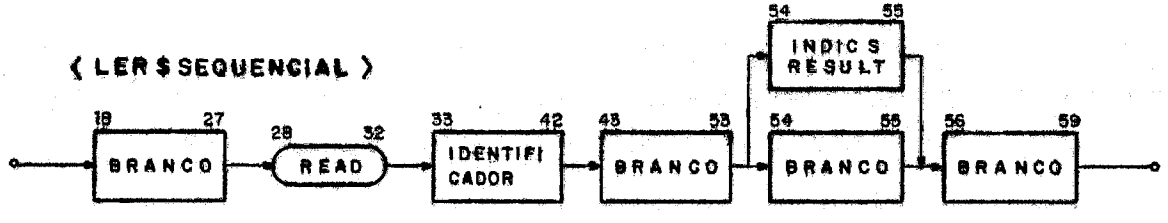
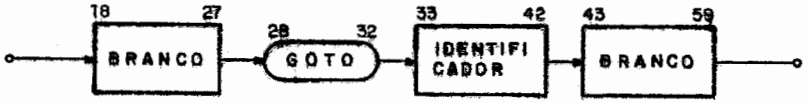
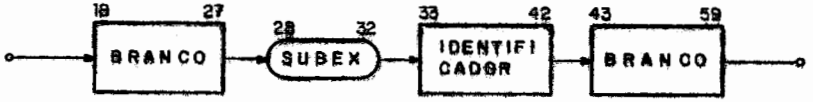


DIAGRAMA SINTÁTICO

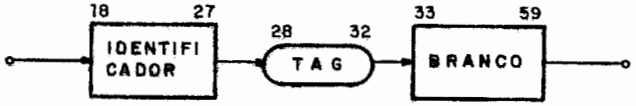
⟨ GOTO 1 ⟩



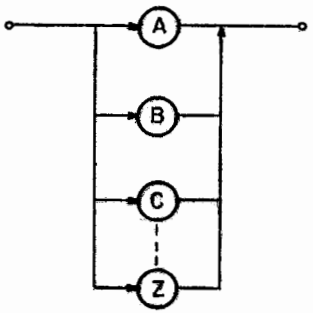
⟨ ROT$EXT ⟩



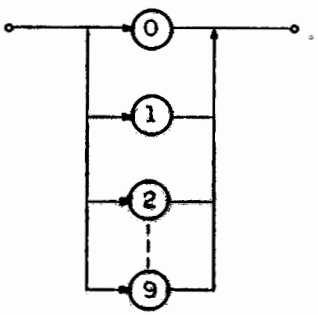
⟨ TAG 1 ⟩



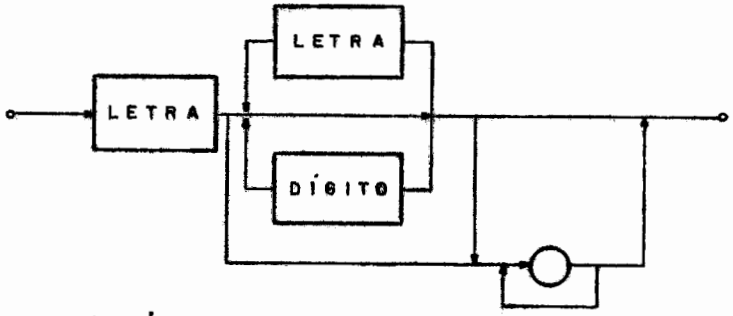
⟨ LETRA ⟩



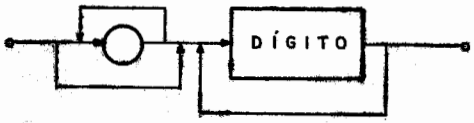
⟨ DÍGITO ⟩



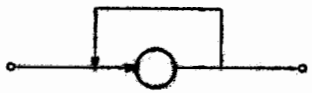
⟨ IDENTIFICADOR ⟩



⟨ NÚMERO ⟩



⟨ BRANCO ⟩



⟨ CARATER ⟩

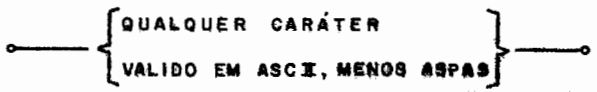
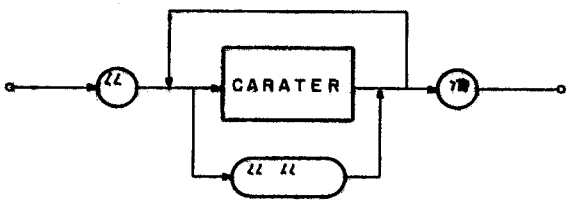


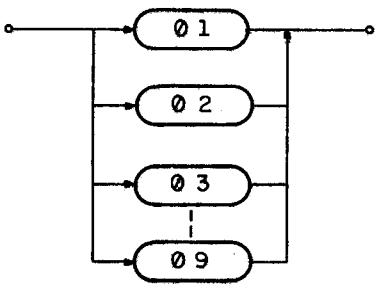
DIAGRAMA SINTÁTICO

⟨ LITERAL $ ALFANUM ⟩

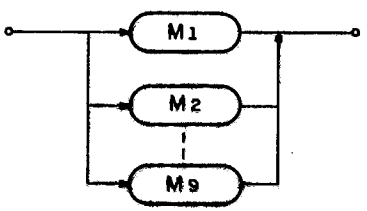
⟨ CADEIA $ CARACTERES ⟩



⟨ INDIC $ 01 $ 99 ⟩

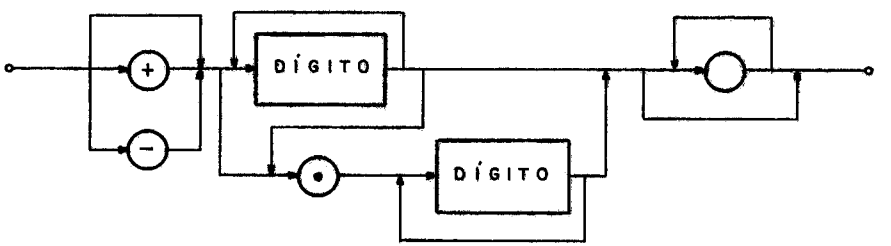


⟨ INDIC $ M1 $ M9 ⟩

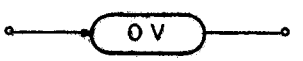


⟨ INDIC $ 1P ⟩

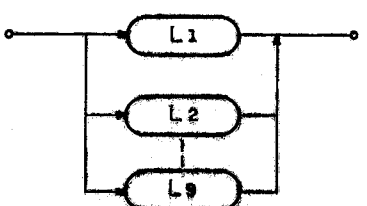
⟨ LITERAL $ NUMÉRICO ⟩



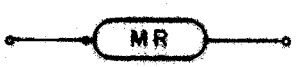
⟨ INDIC $ OV ⟩



⟨ INDIC $ LI $ L9 ⟩

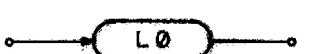


⟨ INDIC $ MR ⟩



⟨ INDIC $ LR ⟩

⟨ INDIC $ LO ⟩



2.4. DESCRIÇÃO DAS ENTRADAS NOS FORMULÁRIOS

2.4.1. ENTRADAS COMUNS NOS FORMULÁRIOS RPG

Serão definidas nesta parte as entradas comuns aos formulários de codificação RPG.

Entradas:

1. Colunas 1 - 2 (página).
2. Colunas 3 - 5 (linha).
3. Coluna 3 (tipo).
4. Coluna 7 (comentário).
5. Colunas 75 - 80 (identificação do programa).

* COLUNAS 1 - 2 (PÁGINA).

Entrada		Explicação
01 - 99	-	nº da página.
/*	-	indica final de especificação de dados fonte.
**	-	seguido de um branco na coluna 3 é um delimitador para tabela (PREVISÃO).

As colunas 1 - 2 servem para enumerar os formulários usados no programa. Podem ser usados mais de um formulário do mesmo tipo. Feito o preenchimento eles devem ser colocados na ordem abaixo especificada, em ordem crescente de nº de página.

1. Cartão de controle e descrição de arquivos.
2. Extensão e contador de linha.
3. Entrada.
4. Cálculo.
5. Saída.

* COLUNAS 3 - 5 (LINHA)

Entrada		Explicação
Qualquer nº	-	nº da linha.

As colunas 3 - 5 servirão para enumerar as linhas de cada página.
A linha do cartão de especificações de controle, será sempre 01. A numeração das demais linhas deverá estar em ordem crescente.

* COLUNA 6 (TIPO).

Esta coluna contém pré-impressa uma letra que identifica o tipo de formulário.

Entrada		Explicação
H	-	Formulário de cartão de especificação.
F	-	Formulário de especificações de descrição de arquivos.
E	-	Formulário de especificações de extensões (PREVISÃO).
L	-	Formulário de especificações de contador de linhas.
I	-	Formulário de especificações de entrada.
C	-	Formulário de especificações de cálculo.
O	-	Formulário de especificações de saída.

* COLUNA 7 (COMENTÁRIOS).

Entrada		Explicações
*	-	Linha de comentário.

A linha de comentário é identificada como um asterisco na coluna 7. Qualquer caráter permitido em RPG pode aparecer nesta linha. Uma linha de comentário não pode ser escrita na linha de especificação de cartão de controle.

* COLUNAS 75-80 (IDENTIFICAÇÃO DO PROGRAMA).

Entrada		Explicação
Nome válido em RPG	-	Identificação do Programa.
Branco	-	RPGOBJ é assumido.

As colunas 75-80 do cartão de especificações de controle, são usadas para se especificar o nome do programa objeto. A entrada deverá ser utilizada quando o programa objeto for catalogado de forma permanente na biblioteca objeto. Se estas colunas estiverem em branco, o compilador assume que a entrada é RPGOBJ.

Nos demais cartões do programa fonte, estas colunas são ignoradas pelo compilador.

2.4.2. ESPECIFICAÇÕES DE CONTROLE

Teremos um cartão por programa, que fornece informações ao compilador RPG. Se este cartão é esquecido, um cartão em branco é assumido (figura II-1).

* COLUNAS 1 - 2, 3 - 5 (PÁGINA), (LINHA).
  Ver entradas comuns.

* COLUNA 6 - (TIPO DE FORMULÁRIO).
  Para cada programa teremos um cartão de controle com o caráter "H" na coluna 6. Se as colunas 75 a 80 do cartão estiverem em branco, ou se não houver este cartão, o nome do programa assumido é "RPGOBJ".

* COLUNA 7 - 74.
  A utilização destas colunas deverá ser definida, se necessário, na fase de implementação.

* COLUNAS 75 - 80.
  Estas colunas são utilizadas para a especificação do nome do programa para fins de catalogação. Defaut é "RPGOBJ".

2.4.3. ESPECIFICAÇÕES DE DESCRIÇÃO DE ARQUIVOS

Estas especificações são necessárias para cada arquivo utilizado no programa. Somente uma linha é requerida por arquivo (figura II-1).

* COLUNAS 1-2, 3-5 (PÁGINA), (LINHA).
  Entradas comuns, ver item 2.2.

* COLUNA 6 (TIPO DE FORMULÁRIO).
  Esta coluna deve ser preenchida com o caráter "F".

* COLUNA 7 - 14 (NOME DE ARQUIVO).
  Estas colunas são usadas para indicar o nome de cada arquivo utilizado no programa - NOME INTERNO. O nome do arquivo deve

ser nome válido em RPG, começando na coluna 7.

Nota: As colunas 13-14 são ignoradas pelo compilador.

* COLUNA 15 (TIPO DE ARQUIVO).

Entrada		Explicação
I	-	Arquivo de entrada.
O	-	Arquivo de saída.
U	-	Arquivo de atualização.

Esta coluna especifica como será utilizado o arquivo.

- Arquivo de entrada.
  Arquivos de entrada constituem a fonte de dados para o programa. Todos os arquivos de entrada, exceto os relativos a tabelas (PREVISÃO) devem ser descritos também no formulário de entradas.

- Arquivo de saída.
  Arquivos de saída são aqueles cujos registros serão escritos pelo programa.

- Arquivo de Atualização.
  São arquivos em disco nos quais o programa lê um registro, atualiza campos e coloca o registro no local onde foi lido. Arquivos de atualização devem ser descritos no formulário de especificações de entrada e no formulário de especificações de saída. Em cada ciclo de um programa pode ser atualizado apenas um registro.

  Em se tratando de arquivo encadeado (CHAINED) ou arquivo em demanda pode ser feita uma atualização em tempo de detalhe, em tempo de total ou em tempo de excessão. Todos os demais arquivos são atualizados em tempo de detalhe, durante o mesmo ciclo do programa.

* COLUNA 16 (USO DO ARQUIVO).

Entrada		Explicação
P	-	Arquivo primário
S	-	Arquivo secundário
C	-	Arquivo encadeado (CHAINED)
T	-	Arquivo tabela (PREVISÃO)

D	-	Arquivo de demanda
R	-	Arquivo de endereços de registros (PREVISÃO)

Esta coluna identifica o uso de um determinado arquivo.

- Arquivo Primário - é o arquivo principal do programa. Cada programa deve ter um e somente um arquivo primário.
  Este arquivo pode ser um arquivo de entrada ou de atualização.

- Arquivos Secundários - são processados na ordem em que aparecem no formulário de especificações de descrição de arquivos.

- Arquivo encadeado (CHAINED FILES)
  Um arquivo encadeado (CHAINED) é um arquivo em disco que é lido aleatóriamente ou carregado na operação "CHAIN". Um arquivo encadeado (CHAINED), pode ser de entrada, saída ou atualização.

- Arquivo de tabelas - PREVISÃO
  São arquivos sequenciais de entrada que contêm entradas de tabelas.

- Arquivo de endereços de registros - PREVISÃO.
  É um arquivo de entrada, que indica quais os registros que devem ser lidos num arquivo em disco, e a ordem de leitura. Este arquivo deve ser definido também no formulário de extensões.

- Arquivos de demanda - pode ser um arquivo de entrada ou atualização. A leitura neste arquivo somente é feita através do comando READ, a ser especificado no formulário de cálculos.

* COLUNA 17 (FIM DE ARQUIVO)

Entrada		Explicação
E	-	Todos os registros do arquivo devem ser processados antes do final do programa.
Ø	-	1. Se a coluna está em branco para todos os arquivos de entrada, todos os registros de cada arquivo serão processados antes do final do processamento.

2. O programa pode terminar tenham ou não sido processados todos os registros do arquivo.

Esta coluna se aplica para arquivos de entrada ou atualização, em processamento que envolva mais de um arquivo de entrada. Usa-se para indicar se o programa deve ou não terminar, antes do processamento de todos os registros.

* COLUNA 18 (SEQUÊNCIA)

Entrada		Explicação
A	-	- Deve ser verificada a sequência: registros em ordem crescente.
D	-	- Deve ser verificada a sequência: registros em ordem decrescente.
Ø	-	Não será verificada a sequência de registros.

Esta coluna se aplica para arquivos de entrada ou atualização utilizados como arquivos primários ou secundários , quando é necessária a verificação da sequência dos registros. A especifivação dos campos a serem verificados é feita no formulário de entrada, posições 61 e 62. A verificação da sequência é necessária no caso de uso de campos de casamento em processamento de mais de um arquivo.

* COLUNA 19 - USO DE ARQUIVO NA IMPRESSORA

Se o arquivo é de saída e usa o periférico impressora, preencher com a letra "I". Caso contrário, deixar em branco.

* COLUNAS 20-23 (TAMANHO DO BLOCO).

Estas colunas serão ignoradas pelo compilador,uma vez que o próprio terminal faz blocagem em disco.

* COLUNAS 24-27 (TAMANHO DO REGISTRO)

Entrada		Explicação
1-512	-	nº de bytes em cada registro
Ø	-	assume tamanho = 80 bytes

* COLUNA 28 (MODO DE PROCESSAMENTO)

Entrada		Explicação
Ø	-	1. Consecutivo 2. Sequencial por chave (PREVISÃO)

L - Sequencial dentro de limites (PREVISÃO).

R - 1. Randômico por nº relativo de registro.
2. Randômico por arquivo de endereços (PREVISÃO).
3. Randômico por chave (PREVISÃO).

Esta coluna é usada para indicar o método pelo qual os registros serão lidos no arquivo, ou para indicar que um arquivo de acesso direto será carregado.

- Arquivos primários, secundários ou de demanda:

Organização	Métodos Possíveis
Sequencial	1. Consecutivamente. 2. Por arquivo de endereços (PREVISÃO).
Direto	1. Consecutivamente. 2. Por arquivo de endereços (PREVISÃO).
Indexado (PREVISÃO)	1. Consecutivamente. 2. Sequencial por chave. 3. Sequencial dentro de limites. 4. Por arquivo de endereços.

- Arquivos encadeados (CHAINED), através da operação "CHAIN".

Organização	Métodos Possíveis
Sequencial	Randômico por número relativo de registro.
Direto	Randômico por número relativo de registro.
Indexado	Randômico por número relativo de registro ou por chave.

- Processamento consecutivo - os registros são processados na ordem em que aparecem no arquivo.

- Processamento randômico - é requerido o uso da operação"CHAIN" no formulário de especificações de cálculo, para ler ou escrever registros.

- Processamento dentro de limites (PREVISÃO) - Somente para arquivo indexados, via operação SETLL a ser usada no formulário de especificações de cálculo.

- Processamento por arquivo de endereços (PREVISÃO).
  Este arquivo deverá ser produzido por um utilitário de SORT do TI. Cada registro conterá o nº relativo de um registro a ser lido num arquivo em disco.

- Processamento sequencial por chave (PREVISÃO).
  Utilizado para arquivos indexados. Os registros são lidos em ordem crescente de chave.

* COLUNAS 29 - 30 (TAMANHO DE CAMPO CHAVE OU DE CAMPO DE ENDEREÇO DE REGISTRO) - (PREVISÃO).

Entrada		Explicação
nº	-	Tamanho de campo chave ou de campo de endereço de registro.

Estas colunas são aplicáveis para arquivos em disco ou para arquivos de endereços de registros.

* COLUNA 31 (TIPO DE ENDEREÇO DE REGISTRO).

Entrada		Explicação
A (PREVISÃO)	-	Chaves de registros são utilizados no processamento ou carregamento de arquivos indexados.
I (PREVISÃO	-	O arquivo está sendo processado por arquivo de endereços ou é um arquivo de endereços.
∅	-	1. NºS relativos são usados no processamento de arquivos sequenciais diretos. 2. Um arquivo sequencial ou direto está sendo carregado. 3. Registros são lidos consecutivamente. 4. O arquivo não é de endereços de registros.

Esta coluna se aplica para arquivos de entrada, atualização ou arquivos de saída encadeados (CHAINED); ela indica a maneira pela qual os registros no arquivo são identificados.

* COLUNA 32 (ORGANIZAÇÃO DO ARQUIVO).

Entrada		Explicação
∅	-	Arquivos sequenciais ou diretos.
I (PREVISÃO)	-	Arquivo indexado.
T (PREVISÃO)	-	Arquivo de endereços de registro.

* COLUNAS 33 - 34 (INDICADOR DE OVERFLOW).

Entrada		Explicação
∅	-	Nenhum indicador de Overflow é utilizado (Overflow automático).
OV	-	É utilizado indicador de Overflow (Controle pelo programador).

Esta coluna se aplica para arquivos de saída na impressora, para condicionar a impressão de registros quando é atingida a linha de overflow. Está prevista a utilização de apenas um arquivo na impressora por programa. A linguagem apresenta duas opções ao programador no controle de overflow:

1. Controle de overflow automático - Neste caso ocorrem as seguintes condições ao ser atingida a linha de overflow:
   - Linhas de detalhes (já no ciclo) são impressas.
   - Linhas de total são impressas se requeridas.
   - O formulário avança para nova página.
   - O indicador de overflow é desligado (controle interno).
2. Controle de overflow pelo programador - Neste caso ocorrem as seguintes condições ao ser atingida a linha de overflow:
   - Linhas de detalhes (já no ciclo) são impressas.
   - Linhas de total são impressas se requeridas.
   - São impressas linhas de total condicionadas por overflow.
   - O formulário avança para nova página se houver especificação de salto (SKIP) para linhas de cabeçalho ou total.
   - Linhas de cabeçalho e detalhe são impressas se condicionadas, por overflow.
   - É desligado indicador de overflow (controle interno).

* COLUNAS 35 - 38 (INÍCIO DO CAMPO CHAVE) (PREVISÃO).

Entrada		Explicação
1-512	-	Posição no registro onde começa o campo de chave.

Aplica-se somente a arquivos indexados. O nº colocado deve terminar na coluna 38. O campo de chave de um registro é aquele que identifica o registro; ele deve estar na mesma posição em todos os registros do arquivo.

* COLUNA 39 (CÓDIGO DE EXTENSÃO).

Entrada		Explicação
E (PREVISÃO)	-	O arquivo é descrito também no formulário de extensões.
L	-	O arquivo é descrito também no formulário de contador de linhas.

- Os arquivos de saída para a impressora podem ser descritos no formulário de contador de linhas, com especificação de tamanho de formulário e linha de overflow.
- Tabelas de arquivos de endereços de registros, devem ser descritos no formulário de extensões (PREVISÃO).

* COLUNAS 40 - 46 (PERIFÉRICO-NOME DA ROTINA DE ENTRADA E SAÍDA DO TI).

Estas colunas especificam a rotina de entrada/saída a ser utilizada. Em tempo de execução poderá haver concatenação de arquivos. Assume DUMMY se omitido. Nome ajustado à esquerda.

* COLUNAS 47 - 52 (NOME EXTERNO DO ARQUIVO).

Preencher com o nome externo do arquivo. Se estas colunas estiverem em branco, ou se o nome for inválido, assume o nome interno.

* COLUNAS 53 - 65 (NÃO UTILIZADAS).

* COLUNA 66 (ADIÇÃO DE REGISTROS NUM ARQUIVO) (PREVISÃO).

Estas colunas serão utilizadas para a adição de registros num arquivo indexado.

* COLUNAS 67 - 74.

Não utilizadas no TI.

* COLUNAS 75 - 80.

Ver entradas comuns, item 2.2.

2.4.4. ESPECIFICAÇÕES DE EXTENSÃO (PREVISÃO).

Estas especificações são usadas para descrever arquivos de endereços de registros e tabelas (figura II-2).

TABELAS

Tabelas podem ser usadas em tempo de execução do programa. As tabelas são acessadas referenciando-se um item de cada vez, através da operação LOKUP (formulário de cálculos).

Temos:

- Tabelas em tempo de compilação - São compiladas junto com o programa e tornam-se parte do programa objeto.
- Tabela em tempo de pré-execução - São carregadas na memória antes do início da execução do programa RPG.
- Tabelas em tempo de execução - São lidas como dados de entrada ou criadas durante a execução de cálculos do programa.

As tabelas podem ser ainda simples ou relacionadas.

ARQUIVO DE ENDEREÇOS DE REGISTROS

É um arquivo de entrada que indica quais os registros que devem ser lidos num arquivo em disco e a ordem de leitura.

2.4.5. ESPECIFICAÇÕES DE CONTADOR DE LINHA

Estas especificações são usadas para a especificação de linhas de overflow e tamanho de formulário, para arquivo na impressora. Se não houver este tipo de especificação no programa, é assumido: linha de overflow = 60; tamanho do formulário = 66 (figura II-2).

* COLUNAS 1-2, 3-5 (PÁGINAS), (LINHA).

Ver entradas comuns.

* COLUNA 6 - (TIPO DE FORMULÁRIO).

A letra "L" deve aparecer nesta coluna.

* COLUNAS 7 - 14 (NOME DE ARQUIVO).

Preencher com o nome do arquivo de saída na impressora. Deve ser nome válido em RPG, começando na coluna 7. O arquivo deve estar definido no formulário de especificações de descrição de arquivos.

* COLUNAS 15 - 17 (NÚMERO DE LINHAS POR PROGRAMA).

Estas colunas devem conter o nº de linhas disponíveis no formulário ou página, a ser usado.

* COLUNAS 18 - 19 (ESPECIFICAÇÃO).

Estas colunas devem conter a cadeia 'FL', indicando que a entrada anterior (colunas 15 - 17) se refere a tamanho do formulário.

* COLUNAS 20 - 22 (LINHA DE OVERFLOW).

Conterão o nº de linha de overflow para o programa.

* COLUNAS 23 - 24 (ESPECIFICAÇÃO).

Estas colunas deverão conter a cadeira 'OL', indicando que a entradaprecedente é de especificação de linha de overflow.

* COLUNAS 25 - 74 - Não serão usadas.

* COLUNAS 75 - 80 (IDENTIFICAÇÃO DO PROGRAMA).

Ver entradas comuns.

2.4.6. ESPECIFICAÇÕES DE ENTRADA

Neste formulário são descritos os registros e campos relativos a cada arquivo de entrada e atualização, especificado no formulário de descrição de arquivos (figura II-3).
As colunas 7 a 42 são utilizadas para a identificação dos arquivos e seus registros e as relações destes com outros registros do arquivo.
As colunas 43 a 44 são utilizadas para a descrição dos campos dos registros.

As descrições de campo devem começar pelo menos uma linha abaixo das entradas de identificação de arquivos e registros.

* COLUNAS 1 - 2, 3 - 5 (PÁGINA), (LINHA).

Ver entradas comuns.

* COLUNA 6 (TIPO DE FORMULÁRIO).

A letra I deve aparecer nesta coluna.

* COLUNAS 7 - 14 (NOME DO ARQUIVO).

Pode ser o nome de um arquivo de entrada ou atualização, que deverá ter sido descrito no formulário de descrição de arquivos. Se este nome é esquecido, é assumido o nome do último arquivo descrito. Este nome deverá ser nome válido em RPG, começando na coluna 7, com 6 caracteres no máximo

Nota: As colunas 13 - 14, são ignoradas pelo compilador na apropriação do nome do arquivo.

* COLUNAS 15 - 16 (SEQÜÊNCIA).

Entrada	Explicação
Dois caracteres - alfabéticos (NS)	Nenhuma seqüência deve ser verificada.
Um nº de dois dígitos -	Seqüência deverá ser verificada (seqüência de grupos de registros).

Estas colunas podem conter:

a) Entrada numérica - para assinalar uma sequência de tipos de registros de um arquivo e para indicar que a seqüência deve ser verificada pelo programa. A seqüência deverá se iniciar com "01".

b) Entrada alfabética (NS) - para indicar que não deve ser verificada seqüência de tipos de registros.

Num arquivo, as entradas alfabéticas devem preceder as entradas numéricas. Um tipo de registro fora de ordem ocasiona a parada do sistema. O programa poderá ser reativado ignorando o registro, e lendo o próximo registro.

* COLUNA 17 (NÚMERO).

Entrada		Explicação
Ø	-	Tipos de registros sem verificação de seqüência. (colunas 15 - 16 possuem entradas alfabéticas).
1	-	Somente um registro deste tipo está presente no grupo sequencial.
N	-	Um ou mais registros deste tipo podem estar presentes no grupo sequencial.

Esta coluna é usada se verificação de seqüência de grupos de registros está sendo utilizada. Linhas AND e OR não recebem especificação nesta coluna.

* COLUNA 18 (OPCIONAL).

Entrada		Explicação
Ø	-	O registro deste tipo deve estar presente em cada grupo sequencial.
O	-	O registro deste tipo pode ou não estar em cada grupo sequencial (opcional).

Esta coluna é utilizada quando a seqüência de tipos de registros está sendo verificada.
Linhas 'AND' e 'OR' não possuem entrada nesta coluna. É aplicada a entrada da linha anterior.

* COLUNAS 19 - 20 (INDICADOR DE IDENTIFICAÇÃO DE REGISTRO).

Entrada		Explicação
01 - 99	-	Indicador de identificação de registros.
L1 - L9	-	Indicador de controle de nível.
LR	-	Indicador de último registro.
H1 - H9	-	Indicador de parada.

Estas colunas são usadas para que se possa identificar os diversos tipos de registros a serem processados. Quando um determinado tipo de registro é selecionado para processamento o seu correspondente indicador é ligado (todos os demais indicadores estarão desligados neste momento, exceto no caso de processamento de arquivos ENCADEADOS e de DEMANDA em que os registros são lidos através dos comandos "CHAIN" e "READ", haven-

do a possibilidade de vários indicadores estarem ligados).

- Quando se usa um indicador de controle de nível como indicador de identificação de registro, feita a leitura somente é ligado este indicador (os demais permanecem desligados).
- As linhas relativas as relações 'AND' e 'OR' não recebem preenchimento das colunas 19 e 20.

* COLUNAS 21 - 41 (CÓDIGOS DE IDENTIFICAÇÃO DE REGISTROS).

- Um arquivo poderá possuir diferentes tipos de registros para processamento. A identificação é feita através de códigos a serem inseridos em determinadas posições dos registros. Em tempo de leitura, o tipo do registro é determinado em função dos códigos. Se os dados de um registro preenchem as características de mais de um tipo de registro, será assumido o 1º tipo. Se o arquivo possuir um só tipo de registro os códigos poderão permanecer em branco.
- Em cada linha temos 3 (três) códigos relacionados por 'AND'. Um tipo de registro poderá ser identificado por mais de três códigos utilizando-se as relações 'AND' e 'OR' nas linhas seguintes.

- POSIÇÃO (COLUNAS 21-24, 28-31, 35-38).

Entrada		Explicação
Ø	-	Nenhum código de identificação é necessário.
1-512	-	Posição do código no registro (alinhado a direita).

- NOT (N) (COLUNAS 25, 32, 39).

Entrada		Explicação
Ø	-	O código está presente na coluna.
N	-	O código não está presente na coluna.

- C/D/Z (COLUNAS 26, 33, 46).

Entrada		Explicação
C	-	Caráter.
D	-	Porção de zona do caráter.
Z	-	Porção de dígito do caráter.

NOTA: Em tempo de execução, lido um registro, se o mesmo não se identifica com as especificações do programa, o processamento é interrompido. O operador poderá dar continuidade a execução, sendo desprezado o registro não identificado, e sendo lido novo registro no arquivo.

Relação 'AND'

Em cada linha podemos explicitar no máximo 3 (três) caracteres de identificação. Se o código de identificação consiste de mais de três caracteres, linhas de 'AND' poderão ser usadas acicionalmente.
No caso de serem necessárias mais linhas com a relação 'AND',escrever a palavra AND nas colunas 14-16 e continuar a especificação.

Relação 'OR'

Um determinado tipo de registro pode ser identificado dois códigos diferentes, neste caso, as linhas de 'OR' podem ser usadas de tal forma que qualquer uma delas identifique o registro.

* COLUNA 42

Não será utilizada.

* COLUNA 43

Não será utilizada. Campos somente de formado decimal zonado.

Observações:

1) Os dados serão armazenados na forma não compactada. No caso de números o sinal será especificado na posição de zona do byte mais a direita.
2) O RPG não faz verificação dos dados numéricos. O valor de dígito do caráter é o seu valor numérico.

* COLUNAS 44 - 51 (LOCALIZAÇÃO DO CAMPO NO REGISTRO).

Entrada		Explicação
2 números ≤ 512	-	Começo e fim do campo.

Estas entradas descrevem a localização do campo no re

gistro. Para apenas uma posição repete-se o mesmo número para posição inicial e posição final.
Tamanho máximo de 1 campo numérico: 15 posições.
Tamanho máximo de 1 campo alfanumérico: 255 posições.

* COLUNA 52 (POSIÇÃO DECIMAL).

Entrada		Explicação
Ø	-	Campo alfanumérico.
0-9	-	Nº de posições para campo numérico.

Se o campo é numérico esta coluna deverá ser preenchida.

O nº de posições deve ser menor ou igual ao tamanho do campo. Campos numéricos sem posições decimais preencher com zero (Ø).

* COLUNAS 53 - 58 (NOME DO CAMPO).

Entrada		Explicação
Nome válido em RPG	-	Nome do campo

Usa-se estas colunas para especificar os campos encontrados nos registros de entrada. Devem ser indicados os nomes dos campos para todos os tipos de registro. Todavia, somente serão especificados os nomes dos campos que serão usados no programa.

- Nomes dos campos - devem ser nomes válidos em RPG, ajustados à esquerda.
  Dentro de um mesmo tipo de registro, os nomes devem ser diferentes.
  Todavia, em registro de tipos diferentes, os nomes podem se repetir, mesmo que referenciem campos em posições diferentes, desde que possuam o mesmo tipo de dados e tamanho; neste caso é alocada a mesma área de memória.
- Campos numéricos podem ter o tamanho máximo de 15 (quinze) caracteres.
- Campos alfanuméricos podem ter o tamanho máximo de 255 caracteres.
- Cada campo é especificado em uma linha.

- Se um mesmo campo de um registro for usado numérico e alfanumérico, o campo deverá ser definido 2 (duas) vezes com nomes diferentes.

* COLUNAS 59 - 60 (NÍVEL DE CONTROLE).

Entradas		Explicação
∅	-	O campo descrito não é um campo de controle.
L1-L9	-	O campo descrito é um campo de controle.

Estas colunas são usadas para assinalar indicadores de controle de nível para campos de entrada.

- Campo de controle - São campos para os quais são especificados indicadores de controle. Quando um registro, contendo campo de controle é lido, a informação do campo de controle é comparada com a informação do mesmo campo de controle do registro anterior. Quando ocorre uma quebra de controle (conteúdo dos campos é diferente) o indicador de nível associado é ligado e poderá condicionar operações de cálculo e de saída no programa.
- Grupos de controle - São conjuntos de registros que possuem mesma informação no campo de controle.

Os indicadores de nível são em número de 9. Quando um indicador é setado, todos os demais, de nível inferior, também o são. Por exemplo, se é setado o indicador L3, também serão setados L2 e L1. Os vários níveis de quebra devem ser utilizados pelo programador, conforme suas necessidades.
Normalmente, estes indicadores são usados para:

- Condicionar cálculos a serem efetuados quando muda a informação no campo de controle.
- Condicionar saídas de totais a serem executadas após acumulações num Grupo de controle.
- Condicionar cálculos de detalhe ou operações de saída, a serem efetuadas no registro que ocasionou a quebra de controle (primeiro registro de um novo grupo de controle).

Observações quanto ao uso dos indicadores:

- Se o mesmo indicador de nível de controle for usado em diferentes tipos de registros ou em arquivos diferentes, os campos associados devem ter mesmo tamanho e tipo.
- Nomes de campos são ignorados em operações de controle de nível.
- Campos de controle com posições decimais são tratados como se não houvesse posições decimais.
- Se um campo é especificado como numérico apenas a porção de dígito é considerada (o sinal é ignorado).
- Campos de controle são inicializados com zeros, o que resulta numa quebra de controle falsa no 1º ciclo do programa (não é considerada quebra de controle).
- Se diferentes tipos de registro num arquivo, não possuem o mesmo nº de campos de controle, uma quebra indesejável poderá ocorrer na execução.
- Não há seqüência determinada para se escrever níveis de controle. Assim L2 pode aparecer antes de L1.

* COLUNAS 61 - 62 (CAMPOS DE COMBINAÇÃO - MATCH)

Entrada		Explicação
M1-M9	-	Nível de combinação

Uma entrada nestas colunas indica:

1. Combinação entre os campos (arquivos diferentes) e teste de seqüência quando tivermos processamento de 2 (dois) ou mais arquivos de entrada ou atualização.
2. Teste de seqüência quando tivermos somente um arquivo de entrada ou atualização com campos de combinação especificados.

A utilização destas colunas somente é válida para arquivos primários e secundários (para arquivos encadeados e de demanda não é permitido).

* PROCESSAMENTO DE MÚLTIPLOS ARQUIVOS.

Aplica-se para programas que leem registros de um arquivo primário e de um ou mais arquivos secundários.

- PROCESSAMENTO SEM CAMPOS DE COMBINAÇÃO

Os arquivos são processados separadamente, registro a registro na seguinte ordem:

1. Arquivo primário
2. Arquivos secundários, na ordem em que eles forem especificados no formulário de descrição de arquivos:

- PROCESSAMENTO COM CAMPOS DE COMBINAÇÃO (OU CASAMENTO).

Neste caso os registros são selecionados de acordo com o conteúdo dos campos de combinação. É lido um registro de cada arquivo e os campos de combinação são comparados. A seleção de um registro para processamento é feita da seguinte forma:

1. Quando um registro de um arquivo primário casa com um registro de um dos arquivos secundários, ele é selecionado para o processamento. Neste caso, é ligado o indicador MR, que poderá ser usado para condicionar operações de cálculo ou de saída no ciclo de processamento, com o registro selecionado.
2. Quando não há casamento entre campos especificados, se os registros estiverem em ordem crescente, será selecionado o registro com menor valor no campo de casamento; se os registros estiverem em ordem decrescente, será selecionado o registro com maior valor no campo de casamento.
3. Quando um registro é selecionado num arquivo, o próximo registro do mesmo, é lido, e no começo do próximo ciclo do programa o novo registro é comparado com os demais que estiverem no buffer de entrada, e um deles será selecionado.
4. Registros sem campos de combinação podem ser incluídos nos arquivos. Estes registros são selecionados antes dos registros com campos de combinação. Se dois ou mais registros a serem comparados não possuírem campos de combinação, a seleção é feita conforme a prioridade dos arquivos:
   - Arquivo primário
   - Arquivos secundários na ordem em que são descritos no formulário de descrição de arquivos.
5. É válido, haver num programa, um arquivo com campos de combinação, e todos os demais arquivos sem especificação de cam-

pos de combinação. Neste caso, são processados inicialmente todos os arquivos sem campos de combinação especificados (de acordo com prioridade dos arquivos); em seguida é processado o arquivo com campos de combinação e o teste de seqüência é aplicado nestes campos.

- Observações:

1. Um programa com múltiplos arquivos de entrada, ocorre a combinação entre campos se pelo menos um tipo de registro dos 2 (dois) arquivos possuem campos de casamento especificados.
2. Se campos de combinação são especificados para diversos registros no programa, o mesmo número e valor de nível deve ser usado para cada um dos registros. Estes campos devem possuir mesmo tamanho e tipo por nível.
3. Se mais de um campo de combinação é usado para um tipo de registro, todos os campos são tratados como somente um campo de combinação (em ordem descendente de nível M9-M1).
4. Um mesmo valor de nível de um campo de combinação não pode ser usado duas vezes para um mesmo tipo de registro.
5. Campos de combinação podem ser numéricos ou alfanuméricos.
6. Campos de combinação com posições decimais são tratados como se não tivessem posições decimais. Assim, por exemplo 5.45 é considerado igual a 545.
7. O sinal de campos numéricos é ignorado no processo de combinação. Assim, por exemplo, -5 é considerado igual a 5.

* TESTE DE SEQÜÊNCIA.

O teste de seqüência para registros de entrada ou atualização permitirá a verificação da ordem ascendente ou descendente dos mesmos, conforme o(s) campo(s) especificado(s) com os elementos M1 - M9 nas colunas 61 e 62.
A seqüência ascendente ou descendente dos registros do arquivo deve ser especificada no formulário de especificação de arquivos, coluna 18.
Uma entrada nas colunas 61 e 62 indica que os registros devem ser checados para que seja verificado se os mesmos estão real-

mente na seqüência especificada.

* COLUNAS 63 - 64 - Não serão utilizadas.

* COLUNAS 65 - 70 (INDICADORES DE CAMPO).

Entrada		Explicação
01-99	-	Indicador numérico
H1-H9	-	Indicador de parada (para checar condições de erro nos dados).

Estas colunas são preenchidas com indicadores que serão ligados se determinadas condições relativas aos dados de entrada forem satisfeitas.

As condições são três:

- MAIS (PLUS) - Colunas 65-66 - campo numérico especificado nas colunas 53-58 é maior do que zero.
- MENOS (MINUS) - colunas 67-68 - campo numérico especificado nas colunas 53-58 é negativo.
- ZERO ou BRANCO - colunas 69-70 - se o campo numérico especificado nas colunas 53-58 contém zeros ou brancos, ou se o campo alfanumérico contém brancos.

NOTA: Para campos alfanuméricos, as posições 65 a 68 devem estar em branco.

Os indicadores H1-H9 são utilizados para paradas na execução do programa quando uma condição não desejada ocorre nos dados de entrada. Neste caso, o registro que contém o campo será completamente processado antes do término do programa.

* COLUNAS 71 - 74.

Não serão utilizadas.

* COLUNAS 75 - 80 (IDENTIFICAÇÃO DO PROGRAMA).

Ver entradas comuns.

2.4.7. ESPECIFICAÇÕES DE CÁLCULOS

Nos formulários de especificações de cálculos são descritas as operações a serem efetuadas com os dados, numa ordem determinada pelo programador. Cada especificação de cálculo pode ser dividida em 3 partes (figura II-4):

- Condição: estabelece em que condições o cálculo deverá ser efetuado (tempo de detalhe - tempo de total).
- Operação: estabelece qual a operação a ser feita e os dados sobre os quais ela será executada.
- Testes: indica quais os testes a serem efetuados com os resultados obtidos.

Indicadores são utilizados para registrar os testes e os mesmos podem ser usados para condicionar outras operações.

* COLUNA 1 - 2, 3 - 5 (PÁGINA), (LINHA).

Ver entradas comuns.

* COLUNA 6 (TIPO DE FORMULÁRIO).

Preenchida com a letra 'C'.

* COLUNAS 7 - 8 (NÍVEL DE CONTROLE)

Entrada		Explicação
Ø	-	A operação de cálculo será efetuada em tempo de detalhe, para cada ciclo do programa.
LØ	-	A operação de cálculo será efetuada em tempo de total em cada ciclo do programa.
L1-L9	-	A operação de cálculo será efetuada em tempo de total, quando a apropriada quebra de controle ocorrer.
LR	-	A operação de cálculo será efetuada após o processamento do último registro.
SR	-	A operação de cálculo é parte de uma sub rotina.
AN-OR	-	Estabelece relações de 'AND' e 'OR' entre linhas de indicadores.

As colunas 7-8 são usadas para quatro diferentes finalidades:

- Para executar operações em tempo de total quando determinada quebra de controle ocorrer.
- Para executar operações de cálculo a serem efetuadas após a leitura do último registro.
- Para indicar que a operação faz parte de uma subrotina (interna).
- Para especificar que certas linhas de indicadores estão numa relação AND/OR.

Observação:

Ordem de codificação: cálculos de detalhe (branco nas colunas 7-8), cálculos de total (Lo ou L1 a L9 nas colunas 7-8), cálculos condicionados por LR nas colunas 7-8, cálculos de subrotina (SR nas posições 7-8).

INDICADORES DE CONTROLE DE NÍVEL

O indicador Lo fica ligado durante toda a execução do programa. É normalmente usado quando não são utilizados campos de controle e são necessários cálculos em tempo de total.
Os demais indicadores L1-L9, condicionam cálculos para quando houver quebra de controle. Observe-se que uma quebra de controle para um certo nível faz com que fiquem ligados os indicadores de nível inferior. A excessão ocorre quando um indicador de controle de nível (L1-L9) é usado como indicador de identificação de registro ou quando é setado pela operação SETON; neste caso somente o identificador referenciado é ligado, os demais permanecem sem alteração de estado.

INDICADOR DE ÚLTIMO REGISTRO (LR)

Este indicador é ligado automaticamente após as leitura e processamento do último registro. As operações a serem efe

tuadas após a leitura do último registro, devem ser condicionadas pelo LR. Na codificação as operações condicionadas por LR devem ser escritas após aquelas condicionadas por L0-L9, ou após cálculos de detalhe se não houver cálculos em tempo de total.

LINHAS DE SUBROTINA

O símbolo 'SR' nas colunas 7-8 indicam que a linha faz parte de uma subrotina.
Linhas de subrotina devem ser especificadas por último (após linhas de detalhes e total).

AND / OR

Através do uso de AND/OR, mais de uma linha pode ser usada para condicionadamente de cálculos.

* COLUNAS 9 - 17 (INDICADORES).

Entrada		Explicação
∅	-	A operação é executada em cada ciclo.
Ø1-99	-	Indicadores de campos, indicadores de identificação de registros ou indicadores de resultados, usados em algum lugar do programa.
L1-L9	-	Indicadores de controle de nível assinalados em algum lugar.
LR	-	Indicador de último registro.
MR	-	Indicador de registro de combinação.
H1-H9	-	Indicadores de parada assinalados em algum lugar.
OV	-	Indicador de overflow.

Podem ser colocados até 3 indicadores por linha, se o indicador não deve estar ligado para condicionar uma operação , colocar a letra N antes do indicador.
Utilizando linhas de 'AND' e 'OR', muitos indicadores poderão condicionar os cálculos.

Comentários sobre as entradas:

1. Se as porções 7-8 e 9-17 estiverem em branco, o cálculo especificado na linha será executado em tempo detalhe.

2. O uso de indicador de identificação de registro determina o tipo de registro para o qual o cálculo será efetuado.

3. Um indicador de campo, controla a execução de um cálculo conforme o conteúdo do registro de entrada.

4. Indicadores de resultado são utilizados quando o cálculo a ser efetuado depende do resultado de um cálculo anterior.

5. Os indicadores de controle de nível especificados no formulário de entrada ou no próprio formulário de cálculos, podem ser usados nestas colunas.
   Se indicadores de controle são usados nestas colunas e não nas colunas 7-8, a operação é efetuada em tempo de cálculo de detalhe, somente no primeiro registro do nível de controle especificado.

6. Indicador do último (LR), somente é usado nestas colunas quando é ligado durante os cálculos. Normalmente o indicador LR é usado nas colunas 7-8 para condicionar cálculos de final de programa.

7. Indicador de combinação (MR), é usado para condicionar uma operação a ser feita quando houver combinação de registros.

8. Indicadores de parada (H1-H9) - podem ser usados quaisquer indicadores de parada especificados nos formulários de entrada ou de cálculos (resultados), para prevenir operações quando erros na entrada ou nos cálculos são encontrados.
   Isto é necessário porque o registro de entrada é completamente processado antes de o programa parar.

9. Indicador de overflow - o indicador de overflow, especificado no formulário de descrição de arquivos, pode ser usado para condicionar operações a serem efetuadas quando é encontrada a última linha a ser impressa numa página.

10. Relações entre as colunas 7-8 e colunas 9-17: em um ciclo de programa, todas as operações condicionadas por indicadores de controle de nível nas colunas 7-8 (operações em tempo de to-

tal) são feitas antes das operações não condicionadas por indicadores de controle de nível nas colunas 7-8 (operações em tempo de detalhe).
Quando um indicador de nível de controle é usado nas colunas 9-17 e as colunas 7-8 não são usadas (tempo de detalhe), a operação condicionada pelo indicador é executada somente sobre o registro que ocasionou a quebra do controle.

* COLUNAS 18 - 27 (FATOR 1) E COLUNAS 33 - 42 (FATOR 2).

Estas colunas são usadas para a especificação dos nomes dos campos ou dos literais, envolvidos na operação.
As entradas devem começar na coluna 18 para fator 1, e na coluna 33 para fator 2.
As entradas para fator 1 e fator 2 podem ser:

1. Nome de campo já definido anteriormente.
2. Literal:
   Alfanumérico - colocar entre apóstrofes; tamanho máximo de 8 caracteres; é válido qualquer caráter, inclusive o branco ; não pode ser usado em operações aritméticas.
   Numérico - tamanho máximo de 10 caracteres; o sinal + ou - pode ser usado; brancos não podem aparecer num literal numérico.
3. Somente para fator 1: label para operações TAG, BEGSR ou ENDSR
4. Somente para fator 2: label para operações GOTO ou EXSR; nome de arquivo para operações CHAIN, READ.

* COLUNAS 28 - 32 (OPERAÇÃO)

Entradas nestas colunas especificam as operações a serem executadas usando fator 1, fator 2 e campo resultado.
Cada operação é especificada colocando-se o seu código a partir da posição 28.

a) OPERAÇÕES ARITMÉTICAS

Estas operações somente podem ser efetuadas com campos ou literais numéricos. O resultado também deve ser numérico.
No caso do uso dos três campos, pode ocorrer:
a) Fator 1, fator 2 e resultado são campos diferentes.

b) Fator 1, fator 2 e resultado são o mesmo campo.

c) Fator 1 e fator 2 são o mesmo campo mas diferentes do campo resultado.

d) Fator 1 ou fator 2 igual ao campo resultado.

O tamanho de qualquer campo envolvido numa operação aritmética não pode ultrapassar a 15 caracteres (se ultrapassar haverá truncamento). Os resultados terão sinal positivo ou negativo.

Operações:

ADD(SOMA) - fator 2 é adicionado ao fator 1. A soma é colocada no campo resultado. Fator 1 e fator 2 não se alteram na operação.

SUB (SUBTRAÇÃO) - fator 2 é subtraído do fator 1. A diferença é colocada no campo resultado. Fator 1 e fator 2 não se alteram na operação.

Z-ADD (ZERA E SOMA) - fator 2 é somado a um campo de zeros e a soma é colocado no campo resultado.

Z-SUB (ZERA E SUBTRAI) - fator 2 é subtraído de um campo de zeros. A diferença é colocada no campo resultado (usada para trocar sinal de um campo).

MULT (MULTIPLICAÇÃO) - fator 1 é multiplicado por fator 2. O produto é colocado no campo resultado. Fator 1 e fator 2 não se alteram.

DIV (DIVISÃO) - fator 1 (dividendo) é dividido por fator 2 (divisor) e o quociente é colocado no campo resultado . Fator 2 não pode ser zero. Qualquer resto resultante da operação é perdido, a não ser que seja usada a operação MVR (MOVE RESTO) como a operação seguinte.

MVR (MOVE RESTO) - esta operação move resto de uma operação anterior de divisão para um campo colocado como campo resultado. Esta operação deve seguir a operação de divisão e estar condicionada pelos mesmos indicadores. O tamanho máximo do resto é de 15 posições, incluindo posições decimais.

b) OPERAÇÕES DE MOVIMENTAÇÃO.

O fator 1 não é usado. As operações movem todo ou parte do fator 2 para o campo resultado.

A conversão de campos numéricos para alfanuméricos e vice-ver

sa é obtida estipulando o tipo desejado no campo resultado . Posições decimais são ignoradas na movimentação de campos numéricos.

Operações:

MOVE (MOVE) - esta operação faz a movimentação de caracteres do fator 2 para as posições mais a direita do campo resultado. A movimentação começa com o caráter mais a direita. Na passagem de alfanumérico para numérico os dados são convertidos e depois movidos para o campo resultado; posições em branco são convertidos para zeros; a porção de zona do último byte a direita é convertida para um sinal correspondente e usada como sinal do campo resultado.

MOVEL (MOVE A ESQUERDA) - esta operação movimenta caracteres do fator 2 para as posições mais a esquerda do campo resultado. A movimentação começa com o caráter mais a esquerda. Na conversão alfa para numérico, valem as mesmas regras do MOVE; com relação ao sinal, mesmo que o caráter mais a direita não tenha sido transferido, a sua parte de zona é usada como sinal do campo resultado.

MLLZO, MLHZO, MHHZO, MHLZO (MOVE ZONAS, NAS DIVERSAS MODALIDADES) - estas operações são usadas para movimentação da porção de zonas dos caracteres de um campo (PREVISÃO).

c) OPERAÇÕES DE COMPARAÇÃO E TESTE

Estas operações testam campos para certas condições . O resultado é mostrado pelos indicadores assinalados nas colunas 54-59.

TESTE-Z (TESTE ZONA) - esta operação testa a zona do caráter mais a esquerda do campo resultado que deve ser alfanumérico. Fator 1 e fator 2 não são usados (PREVISÃO).

COMP (COMPARAÇÃO) - esta operação efetua uma comparação entre o fator 1 e o fator 2. Como resultado da comparação, indicadores são ligados, da seguinte forma:

MAIOR (HIGH) - se fator 1 > fator 2
MENOR (LOW) - se fator 1 < fator 2
IGUAL (IQUAL)- se fator 1 = fator 2

Fator 1 e fator 2 devem ser ambos numéricos ou ambos alfanuméricos.

Os campos são alinhados antes da comparação.

Se os campos são alfanuméricos são alinhados pelo seu caráter mais a esquerda. Se um campo é menor é completado com brancos.

Se os campos são numéricos, são alinhados pelo ponto decimal.

d) SETANDO INDICADORES

SETON-SETOF (LIGA/DESLIGA INDICADORES) - estas operações são usadas para ligar e desligar indicadores colocados nas colunas 54-59.

Observações:

1. Indicadores que não podem ser ligados: 1P, MR, Lo.
2. Indicadores que não podem ser desligados: 1P, MR, LR, Lo.

e) INSTRUÇÕES DE SALTO.

GO TO (GOTO) - instruções de salto para instrução rotulada (instrução rotulada - ver operação TAG).

Podemos ter GOTO para linha anterior ou para linha posterior, nas especificações, não pode haver salto de um cálculo condicionado por indicador de nível (colunas 7-8) para outro que não o é, e vice-versa. No caso de subrotinas, o salto não pode alcançar um cálculo fora da sub-rotina ou vice-versa.

TAG (RÓTULO) - esta operação indica o rótulo para uma operação de salto (GOTO). O fator 1 contém o rótulo (máximo 6 caracteres) a ser escrito a partir da coluna 18. O rótulo deve ser único.

LOKUP (PESQUISA EM TABELA) (PREVISÃO) - usada para pesquisa em tabela para encontrar um elemento especial.

f) SUB-ROTINAS INTERNAS

A linguagem permite e utilização de sub-rotinas internas, contendo um conjunto de operações. As linhas de sub-rotina devem ser identificadas por 'SR' nas colunas 7-8, temos três códigos:

BEGSR (COMEÇO DA SUB-ROTINA) - este código de operação serve para indicar o início da sub-rotina. Fator 1 deve conter o nome da sub-rotina.

ENDSR (FIM DA SUB-ROTINA) - este código serve para indicar o fim da sub-rotina. Fator 1 pode ser um rótulo. O programa ao encontrar este código volta para o próximo comando após o comando EXSR.

EXSR (EXECUTE SUB-ROTINA) - esta operação solicita a execução da sub-rotina. Fator 2 deve conter o nome da sub-rotina. Este nome deve aparecer numa instrução BEGSR.
Campos usados em sub-rotinas podem ser definidos dentro ou fora da sub-rotina.
Um programa pode ter várias sub-rotinas, todavia uma rotina não pode ser escrita dentro de outra. Uma sub-rotina pode chamar outra sub-rotina, não pode chamar ela mesma ou outra que a chame.

g) SUB-ROTINAS EXTERNAS

A linguagem permite o chamamento de sub-rotinas externas escritas em PLTI ou ASSEMBLER.
Código da operação: SUBEX.
Fator 2: nome da sub-rotina (nome válido em RPG).
A execução poderá ser condicionada por indicadores de nível (colunas 7-8) e/ou por indicadores de cálculos (colunas 9-17).

h) OPERAÇÕES ESPECIAIS PARA CONTROLE DE ENTRADA/SAÍDA

O ciclo normal RPG pode ser alterado para permitir operações de entrada e saída durante execução das operações de cálculo.
Operações:
READ - esta operação é usada para ler um arquivo definido, como arquivo de demanda, a leitura é feita sequencialmente. Fator 2 deve conter o nome do arquivo onde será lido um registro. Um indicador pode ser usado nas colunas 58-59. Este indicador será ligado se no processo de leitura for alcançado o fim do arquivo.
CHAIN - esta operação é usada para ler um arquivo em disco durante a execução de cálculos. A operação é usada para processamento randômico em arquivos sequenciais e diretos (indexado também). Fator 1 deverá conter chave ou nº relativo de registro. É usado chave para arquivos indexados (PREVISÃO) e nº relativo de registro para arquivos sequenciais e diretos. Fator 2 conterá o nome do arquivo a ser lido. Colunas 54 - 55 poderão conter um indicador que será ligado se o registro não for encontrado; se não for usado indicador e não encontrado o registro o programa para.

EXCEPT (EXCESSÃO) - esta operação permite que registros sejam escritos em tempo de cálculos em execução. Quando em uso basta preencher as colunas de código de operação (28-32) e colunas 7-17, se houver condicionamento. As demais colunas devem estar em branco. Em conjunto no formulário de saída devem ser especificadas as saídas de excessão com a letra 'E' na coluna 15.

* COLUNAS 43 - 48 (CAMPO DE RESULTADO - NOME).

Entrada		Explicação
Nome do campo	-	Nome válido em RPG
Nome da Tabela	-	(PREVISÃO)

Esta coluna é usada para a especificação de campo resultado de uma operação.
Pode ser utilizado um campo já definido em outros formulários ou no próprio formulário de cálculos. Além disso pode ser definido um novo campo, utilizando um nome ainda não usado.
No caso de estar sendo criado um novo campo, preencher também as colunas 49-52 onde são fixados os atributos.
No caso de utilização de um campo já definido, entradas nas colunas 49-52 não são necessárias, porém, se houver, devem coincidir com a definição anterior do campo.
O nome deve ser escrito a partir da coluna 43.

* COLUNAS 49 - 51 (CAMPO RESULTADO - TAMANHO).

Entrada		Explicação
1-255	-	Tamanho do campo resultado.
∅	-	Campo já definido anteriormente.

O programador deverá dimensionar adequadamente o campo resultado, pois caso o resultado não caiba no campo a ele destinado, haverá truncamento.
Campos numéricos tem o tamanho máximo de 15 caracteres. Campos alfanuméricos poderão ter até 255 caracteres.

* COLUNA 52 (POSIÇÕES DECIMAIS)

Entrada		Explicação
∅	-	Campo alfanumérico ou campo numérico descrito anteriormente.

0-9	-	Número de casas decimais num campo de resultado.

Usar esta coluna para indicar o número de posições depois da vírgula num campo de resultado. Se o campo é número e não possuir posições decimais, preencher com zero (Ø); o número de posições decimais não pode ser maior do que o tamanho do campo.
Se o número de posições decimais é maior do que o requerido pela operação, o campo é preenchido com zeros a direita.
Caso contrário, em que o número de posições é menor do que o necessário, os dígitos da direita são truncados.

* COLUNA 53 (MEIO AJUSTE - ARRENDONDAMENTO)

Entrada		Explicação
Ø	-	Sem arrendondamento.
H	-	Com arrendondamento (MEIO AJUSTE).

Esta coluna indica que o conteúdo do campo resultado será arrendondado. Isto é feito, somando 5 (-5 se o número é negativo) ao número a direita da última posição decimal para este campo. As casas decimais a direita da posição especificada para o campo são então truncadas.

* COLUNAS 54 - 59 (INDICADORES DE RESULTADO).

Entrada		Explicação
01-99	-	Qualquer indicador numérico.
H1-H9	-	Qualquer indicador de parada.
L1-L9	-	Qualquer indicador de controle de nível.
LR	-	Indicador de último registro.
OV	-	Indicador de overflow.

Estas colunas são usadas para:

1. Testar o valor de um campo de resultado após uma operação aritmética.
2. Para checar saídas das operações CHAIN, COMP, TESTZ, LOKUP (PREVISÃO), TESTB (PREVISÃO).
3. Para especificar quais indicadores para SETON ou SETOF.
4. Para indicar fim de arquivo para a operação READ.

TESTANDO RESULTADOS

Normalmente somente os indicadores 01-99 e H1-H9, são utilizados. O indicador utilizado é ligado se a condição é satisfeita, e poderá ser usado em outras operações ou na saída.

* COLUNAS 54 - 55 (MAIS OU MAIOR - PLUS OU HIGH)

Colocar um indicador nestas colunas para saber:

1. Se o campo resultado numa operação aritmética é positivo.
2. Se o fator 1 é maior do que fator 2 numa operação de comparação.
3. Se fator 2 é maior do que fator 1 numa operação de LOKUP em tabela (PREVISÃO).
4. Se a operação CHAIN não foi bem sucedida.

* COLUNAS 56 - 57 (MENOR ou MENOR - MINUS ou LOW).

Colocar um indicador nestas colunas para saber:

1. Se o resultado de 1 operação aritmética é negativo.
2. Se fator 1 é menor do que fator 2 numa operação aritmética.
3. Se fator 2 é menor do que fator 1 numa operação de LOKUP em tabela (PREVISÃO).
4. Resultados de TESTZ ou TESB (PREVISÃO).

* COLUNAS 58 - 59 (ZERO ou IGUAL).

Colocar um indicador nestas colunas para saber:

1. Se o resultado de uma operação aritmética é zero.
2. Se fator 1 igual a fator 2 numa operação de comparação.
3. Se fator 2 é igual a fator 1 numa operação de LOKUP em tabela (PREVISÃO).
4. Se fim de arquivo foi encontrada para um arquivo de demanda usando operação READ.

SETANDO INDICADORES.

Entrar com até 3 indicadores nas operações SETON e SETOF a serem ligados ou desligados.

* COLUNAS 60 - 74 (COMENTÁRIOS)

Entrar nestas colunas com comentários que forem necessários.

* COLUNAS 75 - 80 (IDENTIFICAÇÃO DO PROGRAMA).

Ver entradas comuns.

2.4.8. ESPECIFICAÇÕES DE SAÍDA

Através destas especificações são descritos registros de saída. Temos duas partes em cada especificação:

1. Descrição dos registros de saída (colunas 7-31).
2. Descrição dos campos que indicam a posição e o formato do dado no registro.

As especificações de campo devem ser escritas uma linha abaixo das especificações de registro (figura II-5).

* COLUNAS 1 - 2 (PÁGINAS E COLUNAS 3 - 5 (LINHA).

Ver entradas comuns.

* COLUNA 6 (TIPO DE FORMULÁRIO).

A letra 'O' deve aparecer nesta coluna.

* COLUNAS 7 - 14 (NOME DE ARQUIVO).

São utilizadas para identificar o arquivo cujos registros serão colocados em saída. O nome do arquivo deve começar na coluna 7 e deve ser válido em RPG.

Nota: Colunas 13-14 são ignoradas na apropriação de nome.

* COLUNAS 14 - 16 (AND/OR).

Estas colunas são usadas para se especificar linhas AND/OR para operações de saída.

* COLUNA 15 (TIPO).

Entrada	Explicação

H	-	Registro cabeçalho.
D	-	Registro de detalhe.
T	-	Registro de total.
E	-	Registro de excessão (a serem escritos durante o tempo de cálculos).

Esta coluna especifica o tipo de registro a ser escrito.

Ordem preferencial de entrada: registros cabeçalho, detalhe, total e excessão.

* COLUNAS 16 - 18 (ADIÇÃO DE REGISTROS) (PREVISÃO).

Estas colunas serão utilizadas para a adição de registros num arquivo indexado.

* COLUNAS 17 - 22 (ESPACEJAMENTO E SALTO).

Estas colunas são usadas para indicar espacejamento e salto de linha, no arquivo impressora. Se preenchidas com branco, espacejamento simples ocorre automaticamente após cada linha impressa. Se ambos, espacejamento e salto são especificados na mesma linha, eles são executados na seguinte ordem:

1. Salto antes
2. Espaço antes
3. Salto depois
4. Espaço depois.

* COLUNAS 17 - 18 (ESPAÇO).

Entrada		Explicação
Ø	-	Sem espacejamento.
1	-	Espaço simples.
2	-	Espaço duplo.
3	-	Espaço triplo.

O espacejamento é usado para linhas em uma página. Pode ser indicado espaço antes ou depois de uma linha a ser impressa.

* COLUNAS 19 - 22 (SALTO).

Entrada		Explicação
Ø	-	Sem salto
01-12	-	Canal da fita do carro.

O salto se refere ao desvio de uma linha para outra sem parar nas linhas intermediárias. Pode ser indicado salto antes ou depois de uma linha a ser impressa.

* COLUNAS 23 - 31 (INDICADORES DE SAÍDA).

Entrada		Explicação
01-99	-	Qualquer indicador de resultado de campo ou de identificação de registro, previamente especificado.
L1-L9	-	Qualquer indicador de controle de nível previamente especificado.
H1-H9	-	Qualquer indicador de parada previamente especificado.
OV	-	Indicador de overflow (assinalado na descrição de arquivos para o arquivo).
MR	-	Indicador de casamento.
LR	-	Indicador de último registro.
IP	-	Indicador de primeira página.
LO	-	Indicador nível zero.

Usar nestas colunas os indicadores que condicionarão as saídas. Podem ser usados até 3 indicadores por linha. As colunas 23, 26 e 29 podem conter branco ou a letra 'N' que indica que o indicador não deverá estar ligado para ocorrer a escrita. Podemos usar os indicadores para condicionar a escrita de uma linha como um todo e também condicionar certos campos de uma linha. As relações AND/OR podem ser usadas para estender a definição de indicadores de saída para registro de saída.
Linhas de AND/OR não podem ser usadas para condicionar campos.

INDICADOR DE PRIMEIRA PÁGINA (1P).
É utilizado normalmente para a impressão de cabeçalhos na primeira página de relatórios. É usado comumente em conexão com indicador de overflow para permitir a impressão de cabeçalhos também em outras páginas.

O indicador 1P somente pode ser usado em linhas de detalhe e cabeçalho (D-H). As linhas condicionadas por 1P são impressas antes do processamento de qualquer registro.

* COLUNAS 32 - 37 (NOME DE CAMPO).

Entrar nestas colunas com:

- Nome de campo previamente usado no programa.

Se é usado um nome de campos nestas colunas, as colunas 7-22 deverão estar em branco.

* COLUNA 38 (CÓDIGO DE EDIÇÃO).

Esta coluna deverá ser utilizada na edição de campos numéricos, para:

1. Suprimir zeros não significativos.
2. Omitir o sinal na posição de baixa ordem.
3. Pontuar um campo numérico.

A tabela abaixo, sumariza os códigos de edição a serem utilizados.

VÍRGULA	SALDO ZERO NA IMPRESSORA	SEM SINAL	CR	-	X - REMOVE SINAL MAIS Y - EDITA CAMPO COM DATA Z - SUPRESSÃO DE ZEROS
SIM	SIM	1	A	J	
SIM	NÃO	2	B	K	
NÃO	SIM	3	C	L	
NÃO	NÃO	4	D	M	

Cada código de edição pontua de forma diferente. Todos os códigos suprimem zeros não significativos.
Quando se usa código de edição na coluna 38, não pode ser usada palavra de edição nas colunas 45-47.

* COLUNA 39 (BRANCO APÓS).

Entrada		Explicação
Ø	-	Campo não será alterado após escrita.
B	-	Campo será alterado após escrita (se nu

mérico sera zerado; se alfanumérico será preenchido com brancos).

Esta coluna é usada quando se necessita preencher um campo com zeros ou brancos após a impressão do mesmo.

* COLUNAS 40 - 43 (POSIÇÃO FINAL DE UM REGISTRO DE SAÍDA).

Usar estas colunas para indicar a posição de um campo ou constante num registro de saída. Entrar apenas com um número que indica a posição do caráter mais a direita do campo ou constante. Para o caso de uso de disco, o maior número é 512. Para o caso de impressora o maior número é 132. Na codificação cuidar no dimensionamento, considerando os campos editados.

* COLUNA 44 (NÃO SERÁ UTILIZADA).

* COLUNAS 45 - 70 (CONSTANTE OU PALAVRA DE EDIÇÃO).

Estas colunas são usadas para especificar uma constante ou uma palavra de edição (máscara de edição).

CONSTANTE
Uma constante é qualquer informação não mutável no programa. Normalmente são palavras ou símbolos utilizados em cabeçalhos.

Regras para constantes:

1. Nome de campo (colunas 32 - 37) deve estar em branco.
2. Uma constante deve ser colocada entre apóstrofes.
3. Uma apóstrofe numa constante deve ser representada por 2 (duas) apóstrofes.
4. Até 24 caracteres de uma constante podem ser colocados numa linha, linhas adicionais podem ser usadas, mas cada linha será considerada uma constante. Posição final, nas colunas 40-43.

PALAVRA DE EDIÇÃO (MÁSCARA).
A palavra de edição oferece maior flexibilidade na pontuação de um campo numérico do que o código de edição.
É especificado diretamente:

- Se vírgulas, pontos decimais e supressão de zeros são necessários.

- Se o sinal negativo deve ser impresso.
- Se constantes devem aparecer na impressão.

Utilizando palavras de edição, deve-se levar em conta:

- Código de edição não deve ser usado.
- O campo deve ser numérico.
- Deve ser especificado posição final de saída do registro.
- Uma palavra deve ser colocada entre apóstrofes.
- Qualquer caráter disponível para impressão pode ser impresso (alguns tem uso especial, como veremos).
- Uma palavra de edição não pode ultrapassar 24 caracteres.
- O nº de caracteres de substituição (caracteres que não exigem uma posição na linha de saída), deve ser igual ao tamanho do campo a ser editado (casos especiais serão vistos adiante).
- Haverá supressão de zeros não significativos, a não ser que um zero ou asterisco seja especificado na palavra edição. O zero ou asterisco indica que o último zero à esquerda deve ser substituído por branco ou asterisco.
- Zeros ou asteriscos seguindo o zero ou asterisco mais a esquerda são tratados como constantes; eles, neste caso, não são tratados como caracteres de substituição.

OBSERVAÇÕES SOBRE EDIÇÃO

- Deve-se deixar espaço suficiente na linha de impressão para a palavra de edição.
- Caracteres de substituição - são caracteres que não requerem posição na linha. Eles são:
  - Ø - se usado supressão de zeros.
  - * - se usado preenchimento com asterisco.
  - ₿ - branco.
  - $ - se aparece imediatamente a esquerda de um zero de supressão.
- Os caracteres dolar fixo, dolar flutuante, ponto decimal, vírgula, & (representando branco), sinal negativo (- ou CR) e constantes, requerem espaço na linha de impressão.

2.

PALAVRA DE EDIÇÃO	DADO FONTE	SAÍDA EDITADA
'##,###,##Ø,##&CR'	0000000005-	##########.05#CR
'###,###,#Ø.##CR**'	0034567890-	###Ø345,678,90CR**

* COLUNAS 71 - 74 - Não serão usadas.

* COLUNAS 75 - 80 (IDENTIFICAÇÃO DO PROGRAMA).

Ver entradas comuns.

CAPÍTULO III - VISÃO DO SISTEMA

O processo de compilação e interpretação passará pelas seguintes fases:

1. Fase de Compilação.

Nesta fase será realizada a análise do programa fonte e a geração do código intermediário (conjunto de descritores) em formato compatível de entrada para o programa linqueditor do terminal REFEX.
Ela será dividida em duas partes:

Parte 1: análise do programa fonte com geração do código intermediário, da tabela de arquivos lógicos (TAL) e da tabela de símbolos externos (TSE). Os elementos serão gerados numa primeira alternativa na memória.

Parte 2: encerrada a parte de análise e geração será realizada a montagem, em disco, do módulo objeto em formato compatível de entrada para o programa linqueditor do terminal REFEX.

2. Fase de linquedição - programa REFEX.

Nesta fase o programa linqueditor REFEX resolve realocação e chamadas externas, carregando a subrotina de interpretação (SUBROTINA INTERPRETA), chamada no texto, e criando em disco o módulo linqueditado.

3. Fase de carregamento e execução (interpretação) - programa CPROG.

O CPROG é a parte do Sistema Operacional (SOCO) do Terminal, responsável pela preparação de programas para processamento. Pode fornecer, como resultado, tanto o formato imagem do programa como o seu processamento imediato. De um modo geral, o CPROG faz, nesta fase:

- Lê e analisa comando de execução.
- Monta as informações de controle (blocos de controle) para a concatenação de arquivos, tabela resolvida de arquivos lógicos e tabela de arquivos físicos.
- Monta as rotinas de entrada e saída.

- Inicializa blocos de controle para arquivos em disco.
- Carrega o programa desejado e transfere o comando para ele (subrotina INTERPRETA é acionada).

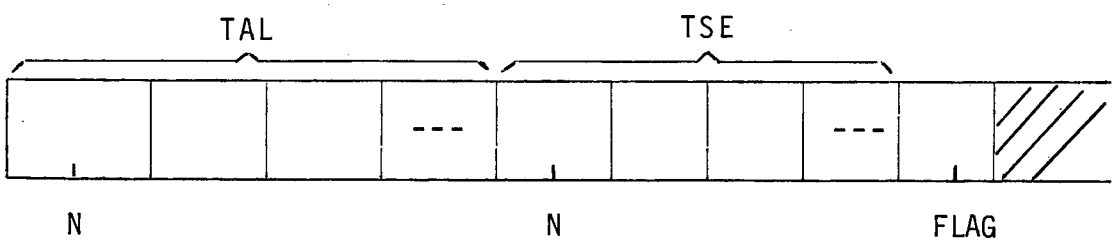
Se o usuário quiser uma cópia do programa a mesma poderá ser guardada na biblioteca no formato imagem de memória. Este procedimento facilitará uma carga futura deste módulo, que não necessitará passar pela construção de blocos de controle. Entretanto, um programa guardado em imagem de memória não poderá ter especificações de arquivos modificados.

Observações

a) Módulo objeto.

REGISTROS DE TABELAS (TAL X TSE)	REGISTROS DE TEXTO

- Registros de tabelas - um ou mais registros no início do arquivo contendo a tabela de arquivos lógicos (TAL) e a tabela de símbolos externos (TSE).
  Tabela de arquivos lógicos (TAL) - contém dados sobre os arquivos definidos pelo programa.
  Tabela de símbolos externos (TSE) - contém as referências externas ao programa bem como as possíveis entradas que possam ser referenciadas por outros programas.
  Concatenando os registros de tabelas temos:



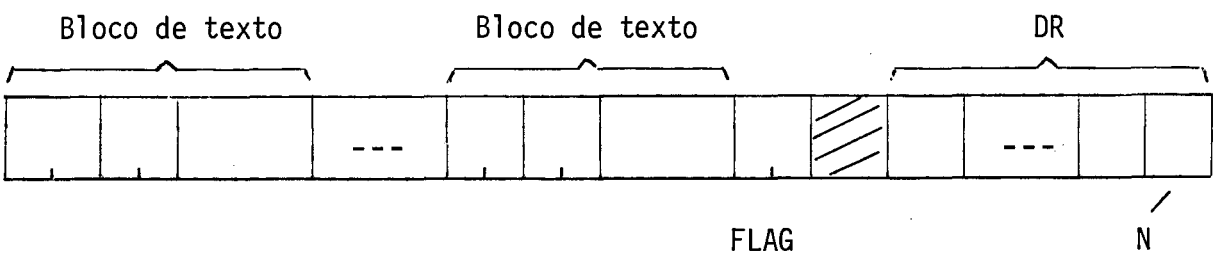
Descrevendo:

```
N  address,                  % Nº DE ENTRADAS NA TABELA
1  TAL,
   2 NOME$INTERNO (5) byte,     %NOME INTERNO DO ARQUIVO
   2 NOME$EXTERNO (5) byte,     %NOME EXTERNO DO ARQUIVO
   2 PERIFÉRICO (5) byte,       %NOME DE ROTINA DE ENTRADA/SAÍDA
   2 TAMANHO$REGISTRO address,  %TAMANHO DO REGISTRO
```

```
1 TSE,                          %EM RPG:
  2 NOME (5) byte,              %NOME DE ARQUIVO OU SUBROTINA EXTERNA
  2 TIPO byte,                  %TIPO=Ø - SUBROTINA EXTERNA
                                %TIPO=1 - ARQUIVO
                                %TIPO=2 - PONTO DE ENTRADA (USADO QUAN-
                                %DO FOR ENTRADA PARA NOME DE PROGRAMA)
  2 DESLOCAMENTO address,       %USADO NO CASO DE TIPO=2, ENTRADA  PARA
                                %NOME DE PROGRAMA.(DESLOCAMENTO=Ø)
```

NOTA: Primeira entrada da TSE: nome de programa

- Registros de texto - um ou mais registros, após os registros de tabelas, contendo o código gerado e as entradas no dicionário de relocabilidade (DR) correspondentes.
  O dicionário de relocabilidade indica quais os endereços do código gerado que dependem de relocação, e o texto corresponde ao código gerado. Texto e DR são montados em registros fixos. Dentro de cada registro são colocados o texto e as entradas do DR correspondentes (Texto e DR crescem em sentido contrário).
  Para cada registro temos:

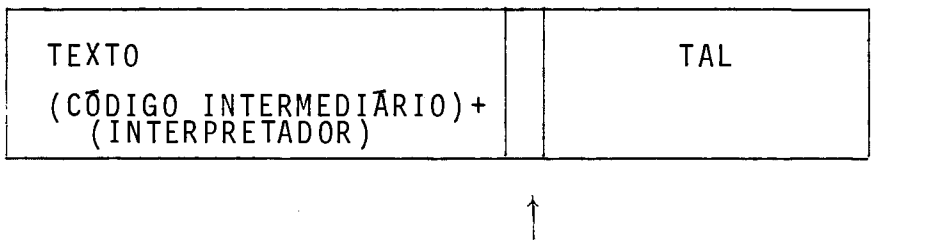


Descrevendo:

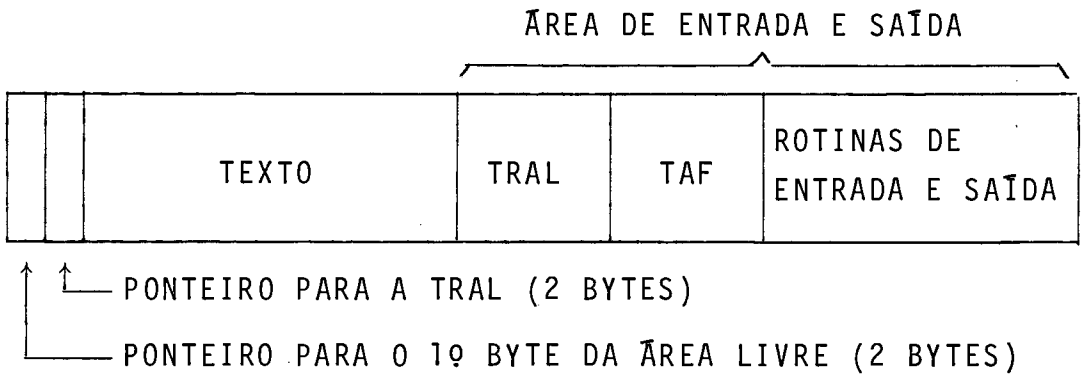
```
1 BLOCO$TEXTO
  2 TAMANHO address,
  2 ENDEREÇO$CARGA address,
  2 TEXTO (TAMANHO-1) byte,

FLAG address,
1 DR
  2 TIPO$INDICE byte,           %TIPO OR INDICE
                                %TIPO:DC
                                %INDICE:Nº DE ORDEM DO SÍMBOLO NA
                                %TSE
```

```
  2 DESLOCAMENTO address;
N byte          %Nº DE ENTRADAS
```

b) Módulo REFEX (LINQUEDITADO)



c) Módulo imagem



d) Tabela resolvida de arquivos lógicos (TRAL) e Tabela de arquivos físicos (TAF).

TRAL - contém as informações referentes ao arquivo lógico e é a interface entre a função de entrada e saída e o arquivo físico onde a função deve atuar. Ou seja, a TRAL seleciona o arquivo físico a ser afetado pela função.

TAF - contém as informações referentes ao arquivo físico, isto é, nome, situação atual do arquivo e tamanho. A TAF está associada a rotina de entrada e saída que executará as funções no arquivo que ela descreve.

CAPÍTULO IV - CÓDIGO INTERMEDIÁRIO

Chamaremos de código intermediário, um conjunto de descritores que conterão as informações do programa fonte, necessárias ao processo de interpretação segundo a lógica fixa RPG.

Estes descritores estão organizados em estruturas de listas de forma a propiciar uma recuperação rápida das informações pelo programa interpretador.

A geração é efetuada na fase de compilação sucedendo a análise léxica, sintática e semântica de cada especificação.

No início da área de geração, existem quatro(4) apontadores que dividem a estrutura em partes distintas que são:

- ELEMENTOS PARA ENTRADA
- ELEMENTOS PARA SAÍDA
- ELEMENTOS PARA CÁLCULOS: CÁLCULOS EM TEMPO DE DETALHE
  CÁLCULOS EM TEMPO DE TOTAL

4.1 ELEMENTOS PARA ENTRADA

Nesta parte estão contidas as informações relativas a arquivos de entrada, seus registros e campos, em listas encadeadas (Figura IV-1). Contém também áreas auxiliares, para dados, para controle de nível (áreas nível), para controle de campos de combinação (áreas match), para controle de sequência de registros , e áreas de buffer para a recepção de registros de dados.

As informações são provenientes dos formulários do fonte: ESPECIFICAÇÕES DE DESCRIÇÃO DE ARQUIVOS E ESPECIFICAÇÕES DE ENTRADA.

Quanto as listas temos:

- Lista de descritores de arquivos de entrada (ou simplesmente, lista dos arquivos de entrada).
- A cada nó da lista dos arquivos de entrada, temos vinculada uma lista de descritores de registros especificados para o arquivo, também chamada de lista de registros de entrada.
- A cada nó da lista de registros de entrada temos vinculadas 3(três) listas:

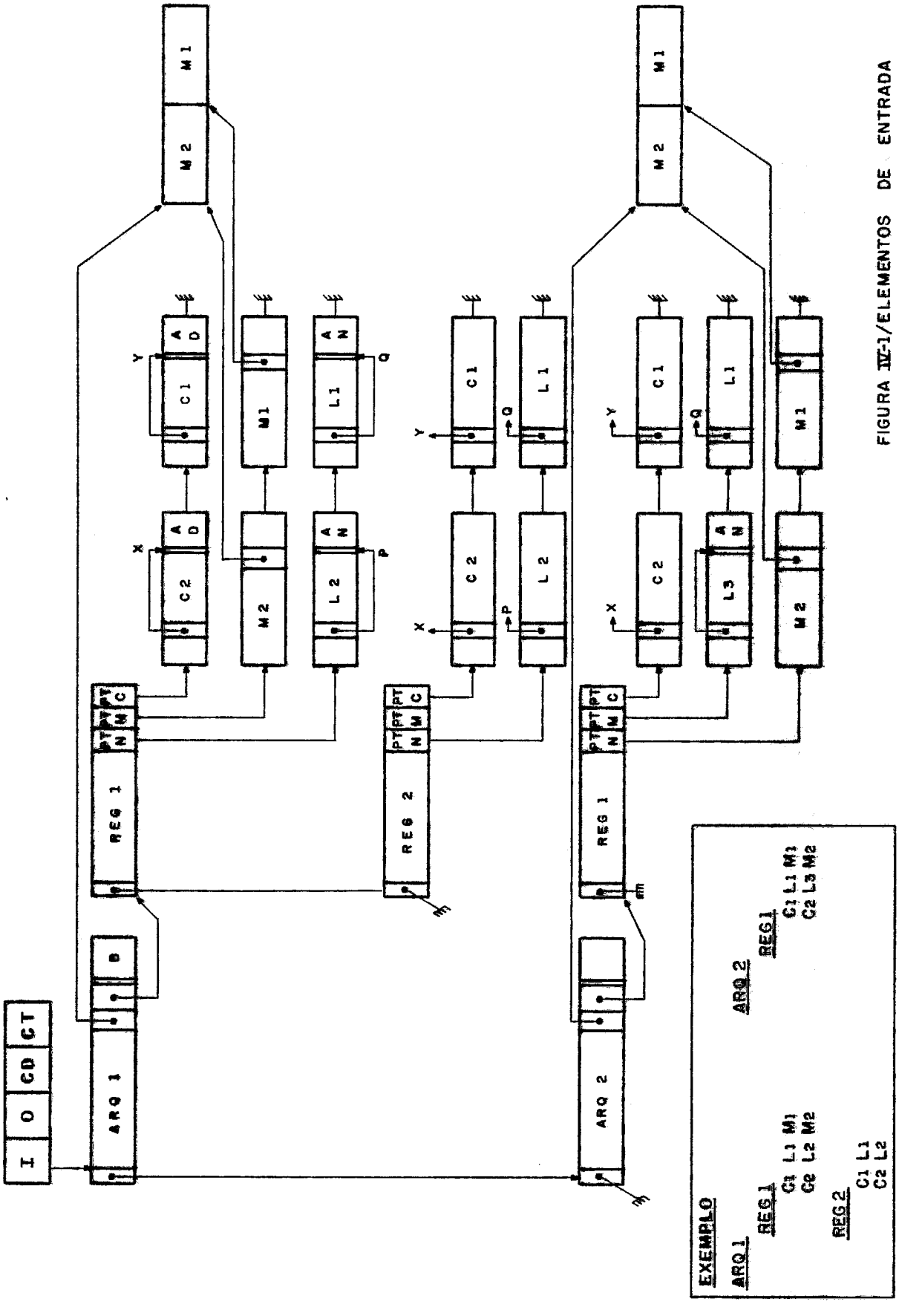


FIGURA IV-1/ELEMENTOS DE ENTRADA

- Lista de descritores dos campos para movimentação (lista move).
- Lista de descritores dos campos com indicador de nível especificado (lista nível).
- Lista de descritores dos campos com indicador de match especificado (lista match).

Com relação as áreas auxiliares, temos:

- Buffer de entrada - um por arquivo, para a recepção dos registros de dados lidos.
- Área de dados - é reservada uma área para cada nome de campo especificado num registro, para armazenamento de dados.
- Área nível - é reservada uma área para cada nome de campo com indicador de nível especificado.
- Área match - é reservada uma área para cada arquivo que possui registros com campos contendo especificação de indicador de match (combinação).
- Área de controle de sequência - reservada uma área para cada arquivo que possui registros com controle de sequência especificado.

A seguir, descreveremos com maior detalhamento os elementos da estrutura.

4.1.1 - LISTAS DE DESCRITORES DE ARQUIVOS DE ENTRADA

a) Características.

É uma lista encadeada - inserção de cada nó após o último nó inserido. Cabeça da lista no início da área de geração. Informações provenientes do formulário de especificações de descrição de arquivos.

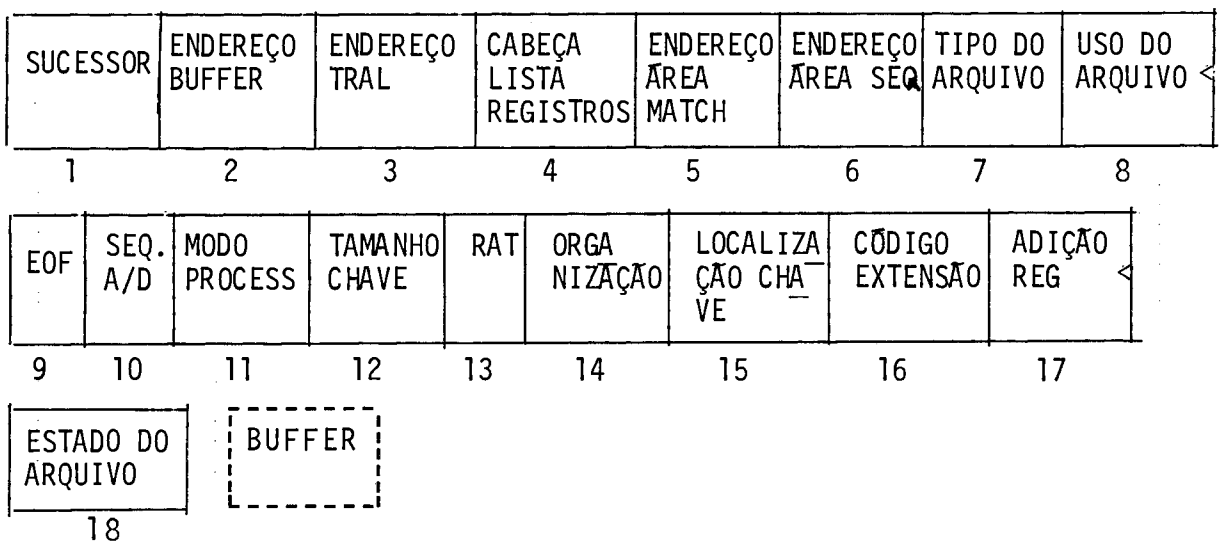
Esta lista contém informações relativas a cada arquivo de entrada especificado para o programa. Contém também ponteiros que são:

- Endereço da área de buffer utilizada para o armazenamento do registro fonte em tempo de interpretação.
- Endereço da tabela resolvida de arquivos lógicos utilizada nas operações de entrada/saída em tempo de interpretação.

- Endereço da área match, utilizada no processo de seleção de registro, no caso de processamento de múltiplos arquivos de entrada com campos de casamento especificados.
- Endereço da área de controle de sequência utilizada na verificação de sequência de registros no arquivo.
- Cabeça da lista de descritores de registros vinculados ao arquivo.

Na fase de interpretação esta lista é usada basicamente no processo de leitura de arquivos. Através dela são acessadas demais informações relativas aos arquivos de entrada.

b) Elementos dos nós.



1. Elo da lista (address) - numérico.
2. Endereço da área de buffer (address) - numérico.
   VALOR = endereço nó + tamanho do nó.
3. Endereço da tabela resolvida de arquivos lógicos (address) - numérico.
4. Cabeça da lista de registros correspondente ao arquivo (address)-numérico.
5. Apontador para área match (address) - numérico.
6. Apontador para área de controle de sequência de registros (address) - numérico.
7. Tipo de arquivo (byte) - caráter.
   VALORES: "I" - Arquivo de entrada.
   "U" - Arquivo de atualização.
8. Uso do arquivo (byte) - caráter.
   VALORES: "P" - principal.
   "S" - secundário.

"D" - demanda.
"C" - encadeado.

9. Fim de arquivo (byte) - caráter.
   VALORES: "E"
   "Ø"
10. Sequência (byte) - caráter - usado em processamento de múltiplos arquivos.
    VALORES: "Ø" - sem match.
    "A" - campos match em ordem crescente.
    "D" - campos match em ordem decrescente.
11. Modo de processamento (byte) - caráter.
    VALORES: "Ø" - processamento sequencial.
    "R" - processamento randômico.
12. Tamanho da chave (byte) - numérico - PREVISÃO para arquivos indexados.
13. Tipo de endereço de registro (byte) - caráter - PREVISÃO para arquivos de endereços de registros.
14. Tipo de organização de arquivos (byte) - caráter - PREVISÃO de uso para arquivos indexados.
15. Localização da chave (address) - numérico - PREVISÃO de uso para arquivos indexados.
16. Código de extensão (byte) - caráter.
    VALORES: "Ø" - sem uso formulário extensão.
    "E" - com uso formulário para tabelas (PREVISÃO).
17. Adição de registros (byte) - caráter - PREVISÃO para arquivos indexados.
18. Estado do arquivo (byte) - numérico - campo a ser usado pelo programa interpretador para indicar se o arquivo está ou não na condição de fim de arquivo. Valores: Ø - não; 1 - sim.

4.1.2 - LISTA DE DESCRITORES DE REGISTROS DE ENTRADA

a) Características - lista encadeada - inserção de cada nó após o último inserido. Cabeça da lista no nó descritor do arquivo correspondente . Informações provenientes do formulário de especificações de entrada.

A lista contém os elementos para a identificação dos registros a serem lidos num determinado arquivo. Os nós contêm ainda

campos para a verificação de sequência de registros e ponteiros (cabeças) para listas de especificações dos campos: Campos de movimentação, campos com controle de nível especificado (indicadores L1-L9) e campos com indicador de match especificado (indicadores M1-M9).

Por condição de sintaxe, os descritores de registros sem controle de sequência antecedem, na lista, os descritores de registros com controle de sequência.

Em tempo de interpretação, lido um registro de um determinado arquivo, a lista é acessada e percorrida para que seja feita a identificação do mesmo. Identificado o registro é verificado se ele se apresenta na sequência desejada (se houver tipos de registros com controle de sequência).

Se o registro for selecionado para processamento, poderão ser acessadas as informações relativas a seus campos.

b) Elementos dos nós.

SUCESSOR	Nº DE SEQUÊNCIA	Nº DE REG. NA SEQUÊNCIA	OPCIONAL	INDICADOR IDENTIF. REGISTRO	CABEÇA LISTA NÍVEL	CABEÇA LISTA MATCH
1	2	3	4	5	6	7

CABEÇA LISTA MÖVE	IDENTIFICAÇÃO CADEIA INDICADORES	✻FF
8	9	

1. Elo da lista (address) - numérico.
2. Número de sequência (byte) - numérico - número de sequência para registros com controle de sequência.
   VALORES:=Ø - sem controle de sequência.
   ≠Ø - com controle de sequência.
3. Número de registros na sequência (byte) - numérico - usado se há controle de sequência.
   VALORES:=✻20 - se não especificado.
   =Ø - com controle de sequência (1 a N).
   >Ø - com controle de sequência determinado.
4. Opcional (byte) - numérico
   VALORES:=✻FF - se especificado (registro opcional).
   =✻20 - se não especificado (registro não opcio-

nal,

5. Indicador de identificação de registro (byte) - numérico.
   VALORES:=Ø - não especificado.
   ≠Ø - especificado.
6. Cabeça da lista nível (address) - numérico.
7. Cabeça da lista match (address) - numérico.
8. Cabeça da lista move (address) - numérico.
9. Identificação do registro - cadeia de tamanho variável, conten
   do:
   - Posição (address) - numérico - posição no registro.
     (VALORES:=1 a (tamanho do registro) - se especificado.
     =Ø - se não especificado.
   - NOT (byte) - numérico.
     VALORES:= 0 - se não especificado.
     =✱FF - se especificado.
   - C/Z/D (byte) - numérico - caráter, zona ou dígito.
     VALORES:=✱FF - caráter.
     =✱FØ - zona.
     =✱ØF - dígito.
   - Caráter (byte) - caráter de identificação.

   Notas:

   1. Especificações na mesma linha - esta implícita a relação AND; em linha de continuação, pode haver relação OR ou re lação AND.
      Para linha de continuação é gerado no início da cadeia:
      ✱FD - se relação AND.
      ✱FE - se relação OR.
   2. No final da cadeia de identificação é gerado um FLAG=✱FF. Se não houver especificação haverá apenas o 'FLAG=✱FF.

4.1.3 - LISTA DE DESCRITORES DE CAMPOS DE MOVIMENTACAO

a) Características - lista encadeada - inserção de cada nó na frente do último nó inserido.

Cabeça da lista no nó descritor do arquivo correspondente. Informações provenientes do formulário de especificações de entrada.

A lista contém as informações relativas aos campos especificados para os registros de um arquivo, necessárias ao processo de movimentação do conteúdo do buffer de entrada para as áreas de dados correspondentes. Os nós contêm também os indicadores usados no programa a serem ligados mediante conteúdo dos campos.

Em tempo de interpretação, a lista é percorrida, são obtidos os endereços e movidos os dados do registro recém lido para as áreas destinadas aos campos. A partir deste momento, os campos ficam disponíveis para uso no programa.

b) Elementos dos nós.

SUCESSOR	ENDEREÇO DO CAMPO	INÍCIO DO CAMPO	RELAÇÃO DE CAMPO	INDICADORES DE CAMPOS INDIC$1	INDIC$2	INDIC$3
1	2	3	4	5		

1. Elo da lista (address) - numérico.
2. Endereço do campo (address) - numérico - endereço da área de dados. (Nota: na área de dados: Tamanho e Tipo).
3. Início (address) - numérico - posição inicial do campo no registro (registro lido na área de buffer).
4. Relações entre campos (byte) - numérico - não usado(PREVISÃO).
5. Indicadores de campos (3 indicadores) (byte) - numérico.
   VALORES: $\neq \emptyset$ - se especificado - indicador - nº associado ao indicador.
   $= \emptyset$ - se não especificado.

4.1.4 - LISTA DE DESCRITORES DE CAMPOS COM CONTROLE DE NIVEL

a) Características - lista encadeada - inserção de cada nó na frente do último nó inserido. Cabeça da lista do nó descritor do arquivo correspondente. Informações provenientes do formulário de especificações de entrada.

Esta lista ficará vazia se no registro não houver campos com controle de nível especificados.

A lista contém as informações dos campos dos registros de um arquivo, para os quais existe especificação de indicador de controle de nível (L1-L9).

Em tempo de interpretação, selecionado um registro para

processamento, a lista é percorrida para a verificação de quebra de controle.

Para tanto é acessado o buffer de entrada e comparado o conteúdo relativo ao campo com o conteúdo da área nível correspondente. Se os valores comparados são diferentes, houve quebra de controle e são ligados o indicador especificado e todos os indicadores de nível inferior.

b) Elementos dos nós.

SUCESSOR	ENDEREÇO ÁREA NÍVEL	INÍCIO DO CAMPO	TAMANHO DO CAMPO	TIPO/PD	NÍVEL
1	2	3	4	5	6

ÁREA NÍVEL

1. Elo da lista (address) - numérico.
2. Endereço da área nível (address) - numérico.
3. Início (address) - numérico - início do campo no registro lido (área de buffer).
4. Tamanho (byte) - numérico - tamanho do campo.
   VALORES: $\leq$ 15 - se campo numérico.
   $\leq$ 255 - se campo alfanumérico.
5. Tipo/Posições decimais (byte) - numérico.
   VALORES: = 20 - se campo alfanumérico.
   Entre Ø e 9 - se campo numérico - indicando o nº de posições decimais.
6. Nível (byte) - numérico - número associado a cada indicador de nível.

4.1.5 - LISTA DE DESCRITORES DE CAMPOS DE COMBINACAO

a) Características - lista encadeada - inserção dos nós de forma ordenada decrescente segundo nível match. Cabeça da lista no nó descritor do arquivo correspondente. Informações provenientes do formulário de especificações de entrada. Esta lista ficará vazia se no registro não houver campos de combinação especificados.

A lista contém informações relativas aos campos para os quais foi especificado indicador de combinação (match) (M1-M9).

Em tempo de interpretação, no processo de seleção de regis

tro a ser processado (Processamento de múltiplos arquivos - verificação de sequência), a lista é percorrida e são movidos para a área match correspondente ao arquivo, o conteúdo do registro lido relativo a cada campo especificado.

b) Elementos dos nós.

SUCESSOR	ENDEREÇO ÁREA MATCH	INÍCIO DO CAMPO	TAMANHO DO CAMPO	TIPO/PD	NÍVEL MATCH
1	2	3	4	5	6

1. Elo da lista (address) - numérico.
2. Endereço da área match (address) - numérico.
3. Início do campo (address) - numérico - posição inicial do campo no registro lido no buffer de entrada.
4. Tamanho do campo (byte) - numérico.
   VALORES: ≤ 15 - se numérico.
   ≤255 - se alfanumérico.
5. Tipo/Posições decimais (byte) - numérico - especifica o tipo do campo e se numérico o nº de casas decimais.
   VALORES: =*20 - se campo alfanumérico.
   Entre Ø e 9 - se campo numérico indicando o nº de posições decimais.
6. Nível match (byte) - numérico - número associado a indicador de nível match (M1-M9).

4.1.6 - AREAS AUXILIARES NA ENTRADA

São áreas vinculadas à estrutura.

a) Buffer de entrada - temos para cada arquivo de entrada um buffer de recepção dos dados lidos através de rotinas de entrada. Este buffer é gerado logo a seguir de cada nó de descritor de arquivo.

b) Área de dados para campos - são áreas de armazenamento das informações relativas a cada campo.
A cada nome de campo corresponde a uma área alocada. Na primeira definição de 1 campo do programa, é gerada a área correspondente, logo após o seu nó descritor na lis

ta de campos.

Na área de dados, temos:

TIPO/PD	TAMANHO	DADOS
1	2	

1. Tipo/Posições decimais (byte) numérico.
   VALORES: =>20 - se campo alfanumérico.
   Entre Ø e 9 - se campo numérico - nº de casas decimais.
2. Tamanho (byte) - numérico.
   VALORES: ≤ 15 - se campo numérico.
   ≤255 - se campo alfanumérico.

c) Área match - é gerada quando pelo menos um registro do arquivo possui especificação de campos de combinação, em processamento de múltiplos arquivos de entrada. Para esta área é movido em tempo de processamento, o conteúdo dos campos match especificados, para que seja feita a seleção de registro a ser processado. Para cada arquivo, contendo registros com campos de combinação, será gerada uma área match.

d) Área nível - é gerada quando para um registro do arquivo houver campos com controle de nível especificado. Nesta área é sempre armazenado o último valor do campo a ser comparado, em cada ciclo da execução do programa, com o novo valor do campo. Quando o último valor for diferente do anterior haverá uma quebra de controle. É gerada uma área para cada tipo de indicador de nível especificado (assim uma área para L1, uma área para L2, etc.). A área é gerada em seguida ao nó descritor do campo nível, quando o indicador aparece pela primeira vez no fonte.

e) Área de controle de sequência de registros - é gerada uma área para cada arquivo, se o mesmo possue registros com controle de sequência. A inicialização é feita na fase de compilação com os dados do descritor do 1º registro com controle de sequência.

APONTADOR PARA 1º REGISTRO COM CONTROLE	Nº DE SEQUÊNCIA ESPERADO	Nº DE REGISTROS ESPERADOS	OPCIONAL	CONTADOR
1	2	3	4	5

1. Apontador para o 1º nó descritor de registro con controle de sequência (address) - numérico - inicializado na fase de compilação.
2. Número de sequência esperado (byte) - numérico.
   VALOR: ≠Ø
3. Número de registros esperados (byte) - numérico.
   VALORES: =Ø - se nº indeterminado (1 ou mais).
   ≠Ø - se nº determinado.
4. Opcional (byte) - numérico - indica se o registro esperado é ou não opcional.
   VALORES: =X'FF' - se é registro opcional.
   =X'2Ø' - se registro não é opcional.
5. Contador (byte) - numérico - serve para contar o número de registros com número de ocorrência indeterminado (1 ou mais).

4.2 ELEMENTOS DE SAIDA

Nesta parte estão contidas, em estrutura de listas, as informações do programa fonte, relativos aos arquivos de saída, seus registros e campos (figura IV-2). As informações são provenientes dos formulários de codificação: ESPECIFICAÇÕES DE DESCRIÇÃO DE ARQUIVOS, ESPECIFICAÇÕES DE CONTADOR DE LINHAS E ESPECIFICAÇÕES DE SAÍDA. Vinculados a estrutura temos áreas para dados (constantes e palavras de edição) e um buffer de saída.

Quanto as listas temos:

- Lista de descritores de arquivos de saída (ou lista de arquivos de saída).
- A cada nó da lista de arquivos de saída estão vinculadas três (3) listas de descritores de registros de saída, que são:
  - Lista de descritores de registros de saída em tempo de cabeçalho/detalhe.
  - Lista de descritores de registros de saída em tempo de total.
  - Lista de descritores de registros de saída em tempo de exces

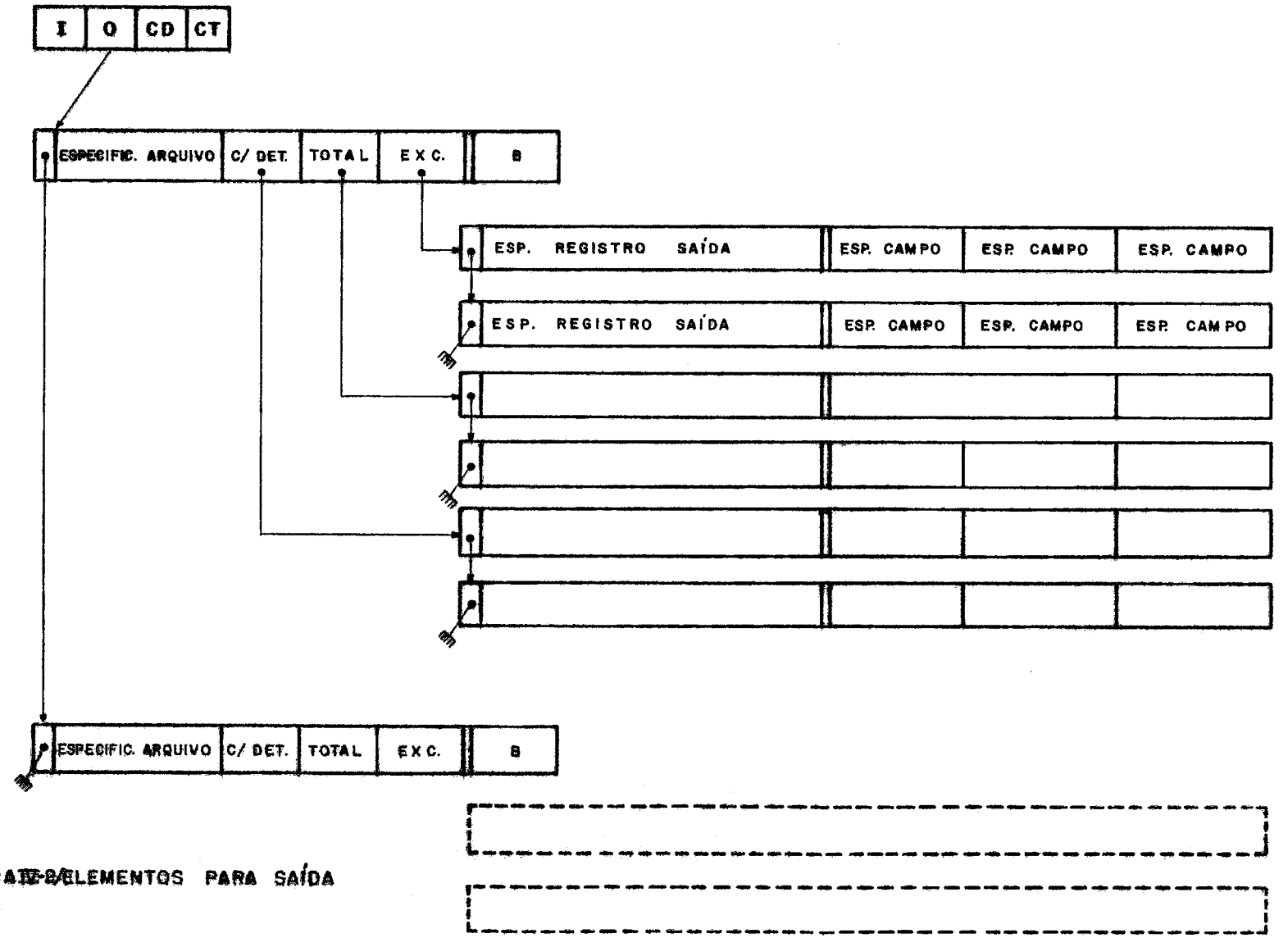


FIGURA 12 - ELEMENTOS PARA SAÍDA

são.

A cada nó da lista de descritores de registros (cabeçalho/detalhe, total ou excessão) existe vinculada uma lista de descritores de campos de saída.

A estrutura é acessada pelo programa interpretador cada vez que for acionado o processo de saída para o programa em execução.

De um modo geral, no caso de saída em tempo de cabeçalho/detalhe, total ou excessão ocorre:

- é percorrida a lista de descritores de arquivos de saída e verificado para cada arquivo, quais os registros que atendem as condições de saída.
- Para os registros que satisfizerem as condições, são verificados quais os campos que atendem as condições de saída (campos condicionados).
- Feitas as verificações é executada a montagem do registro no buffer de saída, procedendo-se a edição se necessário.
- É chamada uma rotina que executará a saída.

A seguir descreveremos a estrutura com maior detalhamento.

4.2.1 - LISTA DE DESCRITORES DE ARQUIVOS DE SAIDA

a) Características - lista encadeada, com inserção de cada nó após o último inserido.
   O cabeça da lista no início da área de geração.

Nesta lista, os nós contém informações relativas aos arquivos de saída previstos no programa fonte. Contém também ponteiros que são os cabeças de listas de registros de saída em tempo de cabeçalho/detalhe, total e excessão, correspondentes ao arquivo.

Na fase de interpretação esta lista se constitui em elemento básico de consulta para a execução das saídas.

SUCESSOR	ENDEREÇO DO BUFFER	ENDEREÇO DA TRAL	TIPO DO ARQUIVO	USO DO ARQUIVO	MODO PROCESS.	TAMANHO DA CHAVE
1	2	3	4	5	6	7

R A T	ORGANIZAÇÃO	INDICADOR OVERFLOW	LOCALIZAÇÃO DA CHAVE	ADIÇÃO REG.	IMPRESSORA	TAMANHO FORMUL.	LINHA OVERF.
8	9	10	11	12	13	14	15

CABEÇA LISTA REGISTROS H/D	CABEÇA LISTA REGISTROS TOTAL	CABEÇA LISTA REGISTROS EXCESSÃO
16	17	18

1. Elo da lista (address) - numérico.
2. Endereço do buffer de saída (address) - numérico.
   VALOR: Endereço do nó + tamanho do nó.
3. Endereço da tabela resolvida de arquivos lógicos - (address)- numérico.
4. Tipo do arquivo (byte) - caráter.
   VALORES: 'U' - arquivo de atualização.
   'O' - arquivo de saída.
5. Uso de arquivo (byte) - caráter.
   VALORES: = 'P' - principal (atualização).
   = 'S' - secundário (atualização).
   = 'D' - demanda (atualização).
   = 'C' - encadeado.
6. Modo de processamento (byte) - caráter.
   VALORES: = 'Ø' - processamento sequencial (consecutivo).
   = 'R' - processamento randômico.
7. Tamanho da chave (byte) - numérico - PREVISÃO para arquivos indexados.
8. Tipo de endereço de registro (byte) - caráter - PREVISÃO.
9. Organização (byte) - caráter - PREVISÃO para uso em arquivos indexados
10. Indicador de overflow (byte) numérico.
    VALORES: =Ø - sem indicador de overflow - controle de overflow automático em arquivo de impressora.
    =1 - com indicador de overflow - controle de overflow pelo programador.
11. Localização da chave (address) - numérico - PREVISÃO para arquivos indexados.
12. Adição de registros (byte) - caráter - PREVISÃO para arquivos indexados.

13. Identificação do arquivo impressora (byte) - caráter.
    VALORES: "I" - arquivo na impressora.
             "Ø" - arquivo não é impressora.
14. Tamanho do formulário (byte) - numérico.
    VALORES: =Ø - para periférico diferente da impressora.
             ≠Ø - para periférico impressora.
                  Se não houver especificação de contador de linhas assume tamanho = 66.
15. Linha de overflow (byte) - numérico.
    VALORES: =Ø - para periférico diferente na impressora.
             ≠Ø - para periférico impressora - se não houver especificação de contador de linhas, assume 6Ø.
16,17,18. Cabeça de listas de registros de saída.

4.2.2 - LISTAS DE DESCRITORES DE REGISTROS DE SAIDA

a) Características - a estrutura é a mesma para registros em tempo de cabec/detalhe, total e excessão. Os cabeças destas listas (3) estão no nó descritor do arquivo correspondente.
A inserção de cada nó é feita após o último nó inserido.
A lista é encadeada.

Nesta lista os nós contêm especificações relativas aos registros de saída e ponteiro (cabeça) para a lista de campos correspondentes.

b) Elementos dos nós.

SUCESSOR	INDICADOR 1P/OV	CABEÇA DA LISTA DE CAMPOS	ESPAÇO ANTES	ESPAÇO DEPOIS	SALTO ANTES	SALTO DEPOIS	CONDIÇÕES SAÍDA CADEIA INDIC.	'*'
1	2	3	4		5		6	

---	ESPAÇO ANTES	ESPAÇO DEPOIS	SALTO ANTES	SALTO DEPOIS	CONDIÇÕES SAÍDA CADEIA INDICADORES	'*'	FLAG *FF
							7

1. Elo da lista (address) - numérico.
2. Indicadores de 1º página e overflow (byte).
   Indica se o registro possui no condicionamento de saída especificados os indicadores 1P e OV.

O bit de mais alta ordem se refere ao indicador 1P.
O bit de mais baixa ordem se refere ao indicador OV.
VALORES: bit=Ø - não especificado.
bit=1 - especificado.

3. Cabeça da lista de descritores de campos de saída (address) - numérico.
4. Espaço antes/depois (byte) - numérico.
Utilizado para arquivo na impressora.
VALORES (antes/depois):=Ø - sem espaço.
=1 - espaço simples.
=2 - espaço duplo.
=X20 - não especificado.
5. Salto (antes/depois) byte - numérico.
VALORES (antes/depois): =01 a 12 - especificado.
=X20 - não especificado.
6. Indicadores de saída - cadeia de tamanho variável contendo:
- NOT(byte) - numérico.
VALORES: =XFF - se especificado.
=XØØ - se não especificado.
- Indicador (byte) - numérico - valor: nº associado a cada indicador.
O final da cadeia é marcado por um FLAG='X', na posição NOT. (Se não houver cadeia aparecerá o FLAG). Continuação na linha seguinte: é gerado no início XFD se linha AND e XFE se linha OR

Nota: No condicionamento de saída poderá haver mais de uma cadeia de indicadores; é o caso de haver uma linha de continuação através relação OR com espacejamento e/ou salto diferente para a saída.

7. FLAG(byte) - é gerado um FLAG=XFF indicativo de final do descritor de registro de saída.

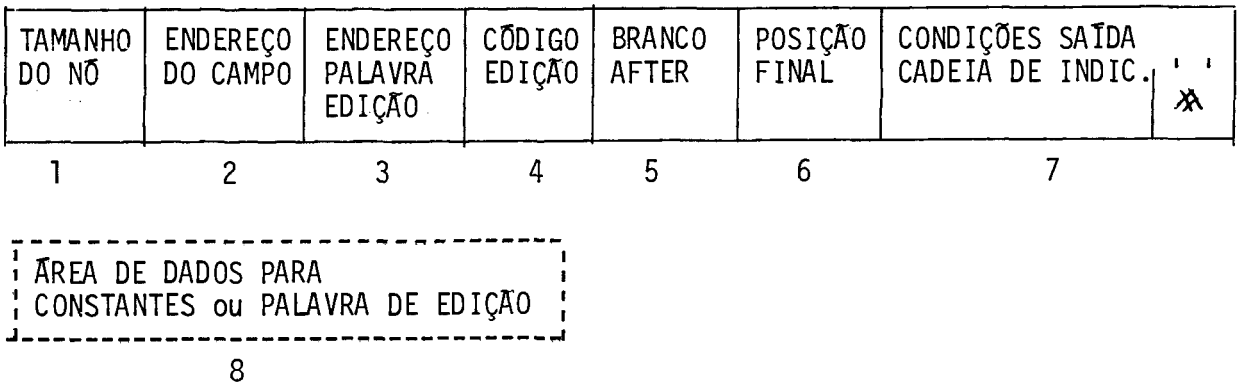
4.2.3 - LISTA DE DESCRITORES DE CAMPOS DE SAIDA

a) Características - é uma lista de alocação contínua na área de geração. A inserção de cada nó é feita após o último nó inserido. O cabeça desta lista está no descritor do registro correspondente.

No fim da lista é gerada uma marca FLAG =X'FF'.

A lista contém especificações dos campos de saída. Observa-se que também os campos podem sofrer condicionamento para a saída através do uso de indicadores. No caso de constantes ou de campos com máscara de edição prevista, é gerada em sequência ao nó da lista uma área para armazenamento destes elementos (constantes ou palavra de edição).

b) Elementos dos nós.



1. Tamanho do nó (byte) - numérico.
2. Endereço do campo (address) - numérico - na fase de geração, pode ser endereço de campo já definido anteriormente coletado na tabela de símbolos, ou endereço de constante definida para a saída. Em ambos os casos, tamanho e tipo estarão definidos na área de dados.
3. Endereço da palavra de edição (address) - numérico - aponta para a área de dados gerada na sequência se houver palavra de edição.
4. Código de edição (byte) - caráter.
   VALORES: 1-2-3-4-A-B-C-D-J-K-L-M-X-Y-Z - se especificado
   "Ø" - se não especificado.
5. Branco after (byte) - caráter.
   VALORES: "Ø" - se não especificado.
   "B" - se especificado.
6. Posição final (address) - numérico - define a posição do campo no registro.
7. Condições de saída - cadeia composta de
   - NOT (byte) - negação da presença do indicador - numérico.
   - VALORES: =X'ØØ' - não especificado.
     =X'FF' - se especificado.
   - Indicador (byte) - numérico - nº associado ao indicador.

8. Área de dados - cadeia de tamanho variável que será gerada se o campo for editado com máscara de edição ou se o campo for constante.
   Conterá portanto máscara de edição ou constante. Tamanho e tipo serão gerados em conjunto na área de dados para cada elemento:
   Tamanho (byte) - numérico - tamanho de constante ou máscara de edição.
   Tipo (byte) - numérico.
   - Valor: X 20 - (cadeia de caracteres - campo alfanumérico)

4.2.4 - AREAS AUXILIARES PARA SAIDA

a) Buffer de saída - área para onde serão movidos campos de saída para a escrita, com tamanho = tamanho de registro previsto para o arquivo. Esta área é gerada após o nó descritor do arquivo correspondente.
b) Área de dados - já descrita, considerada componente do nó descritor de campo de saída (No caso de constante ou palavra de edição).

4.3 ELEMENTOS PARA CALCULOS

Nesta parte estão contidas as informações relativas as operações de cálculos a serem efetuadas provenientes do formulário ESPECIFICAÇÕES DE CÁLCULOS (Figura IV-3). Temos uma lista de descritores das operações a serem realizadas em tempo de detalhe, e outra das operações a serem realizadas em tempo de total.

Estas listas possuem os cabeças no início da área de geração e os seus elementos possuem a mesma descrição.

Para as operações que utilizam literais e para aquelas em que novos campos são definidos, é gerada área auxiliar para armazenamento de dados, em sequência aos descritores da operação.

Abaixo detalharemos a estrutura.

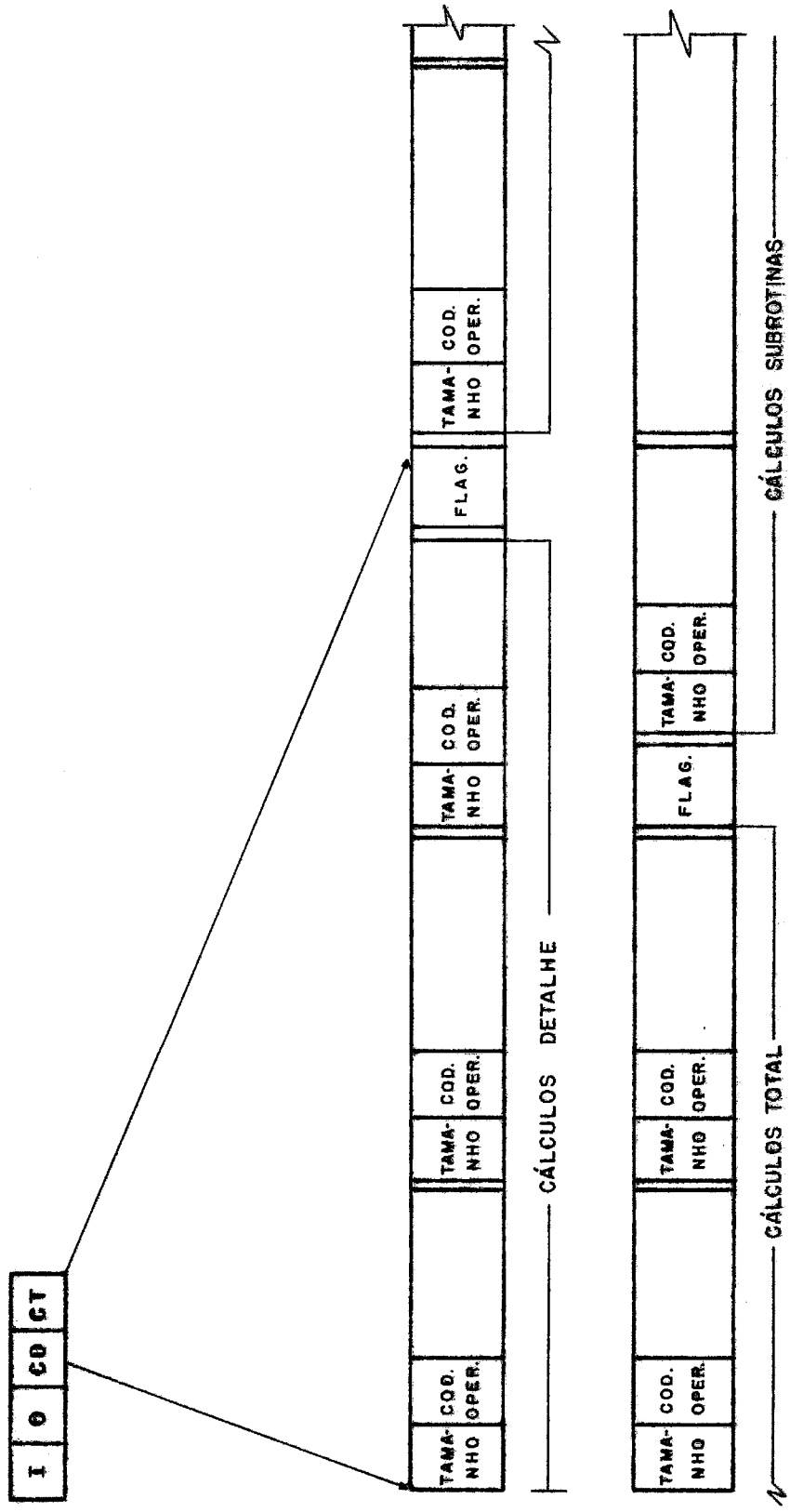
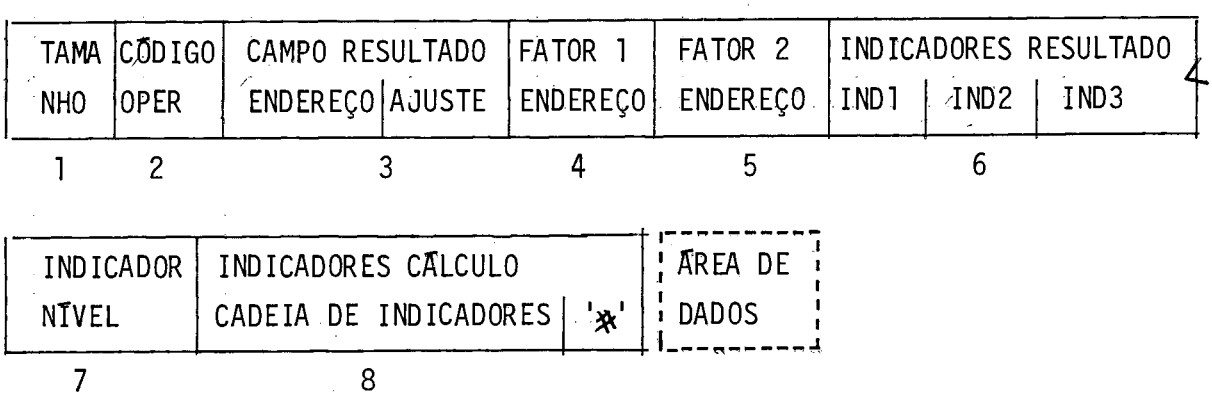


FIGURA III-3/ELEMENTOS PARA CÁLCULOS

4.3.1 - LISTA DE DESCRITORES DE CALCULOS (TEMPO DE DETALHE - TEMPO DE TOTAL)

a) Características - os nós descritores de cada operação são alocados sequencialmente na área de geração, na ordem em que aparecem no programa fonte. Por força de sintaxe, inicialmente são alocados os nós das operações em tempo de detalhe, após as operações em tempo de total, e por último as operações relativas a subrotinas.
O final das listas citadas é marcado através de uma marca FLAG=Ø.

d) Elementos dos nós.
As operações foram separadas em classes conforme suas características.
* Classe 1
Operações ADD, SUB, MULT, DIV - códigos 1,2,3,4, respectivamente.



Descrição dos campos:

1. Tamanho (byte) - numérico - tamanho do nó + tamanho da área de dados (quando for gerada).
2. Código (byte) - numérico - código da operação.
3. Campo resultado
   - Endereço (address) - numérico - endereço do campo resultado. Se o campo está sendo definido pela primeira vez, é gerado espaço para ele na área de dados e este endereço apontará para o espaço alocado (ver item 4.3.2 que trata da área de dados).
   - Ajuste (byte) - caráter.

Valores: "Ø" - não especificado.
"H" - especificado.

4. Fator 1 (byte) - numérico - endereço da área de dados relativa ao campo.
   No caso de fator 1 ser um literal, o endereço apontará para a área de dados gerada após o nó descritor (ver item 4.3.2 que trata da área de dados).
5. Fator 2 - idem fator 1
6. Indicadores de resultado - podem ser usados até 3 indicadores. Para cada um, temos:
   Indicador (byte) - numérico.
   Valor: = valor numérico relativo ao indicador se o mesmo for especificado.
   = Ø - se não especificado.
7. Indicador de nível (byte) - numérico - número associado ao indicador de nível.
   VALORES: =Ø - se não especificado.
   ≠Ø - se especificado, indicando o nº associado ao indicador.
8. Indicadores de cálculo - cadeia de indicadores de tamanho variável contendo:
   - NOT (byte) - número - negação.
   VALORES: =FF - se especificado.
   =ØØ - se não especificado.
   - Indicador (byte) - numérico.
   Valor: nº associado a cada indicador.
   O final da cadeia é marcada por um FLAG = "", na posição NOT. Se não houver cadeia, aparecerá apenas o FLAG.
   Continuação na linha seguinte: é gerada FD se linha AND, e FE se linha OR.

* Classe 2

Operações: Z-ADD e Z-SUB, códigos 5, 6.

TAMANHO	CÓDIGO OPER	CAMPO RESULTADO ENDEREÇO	AJUSTE	FATOR 2 ENDEREÇO	INDICAD. RESULTADOS IND1	IND2	IND3	INDICADOR NÍVEL

INDICADORES CÁLCULO CADEIA INDICAD ''

ÁREA DE DADOS

Descrição dos campos: idem classe 1 (campos comuns).

* Classe 3

Operação MVR, código 7.

TAMANHO	CÓDIGO OPER.	CAMPO RESULTADO ENDEREÇO	INDICAD. RESULTADOS IND1	IND2	IND3	INDICADOR NÍVEL

INDICAD. CÁLCULO CADEIA DE INDICADORES	'#'	ÁREA DE DADOS

Descrição dos campos: idem classe 1 (campos comuns).

* Classe 4

Operações MOVE, MOVEL, códigos 8 e 9, e operações MLLZO, MHHZO, MLHZO e MHLZO(PREVISÃO) códigos 10, 11, 12, 13.

TAMANHO	CÓDIGO OPER	CAMPO RESULTADO ENDEREÇO	FATOR 2 ENDEREÇO	INDICADOR NÍVEL	INDIC. CÁLCULOS CADEIA INDIC	'#'

ÁREA DE DADOS

* Classe 5

Operação COMP, Código 14.

TAMANHO	CÓGIGO	FATOR 1 ENDEREÇO	FATOR 2 ENDEREÇO	INDICAD. RESULTADO IND1	IND2	IND3	INDICADOR NÍVEL

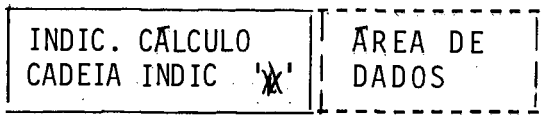
INDICAD. CÁLCULO CADEIA INDICADORES	'#'	ÁREA DE DADOS

Descrição dos campos: idem classe 1 (campos comuns).

* Classe 6

Operação: TESTZ, código 15 (PREVISÃO).

TAMANHO	CÓDIGO OPER	CAMPO RESULTADO ENDEREÇO	INDIC RESULTADOS IND1	IND2	IND3	INDICADOR NÍVEL



Descrição dos campos: idem classe 1 (campos comuns).

* Classe 7

Operações SETON e SETOF - códigos 16,17-

TAMANHO	CÓDIGO OPER	INDIC. RESULTADOS IND1	IND2	IND3	INDICADOR NÍVEL	INDIC CÁLCULOS CADEIA IDICADORES	'*'

Descrição dos campos - idem classe 1 (campos comuns).

* Classe 8

Operações GO TO, EXSR, códigos 18, 19-

TAMANHO	CÓDIGO OPER.	FATOR 2 ENDEREÇO	INDICADOR NÍVEL	INDICADORES CÁLCULOS CADEIA INDICADORES	'*'

Descrição dos campos:

- Fator 2, no caso, é endereço de operação rotulada:
  TAG - se GO TO
  BEGSR - se EXSR
- Demais campos - idem classe 1 (campos comuns).

* Classe 9

Operação READ, código 2Ø

TAMANHO	CÓDIGO OPER.	FATOR 2 ENDEREÇO	INDICADOR NÍVEL	INDICADOR RESULTADO	INDIC. CÁLCULOS CADEIA INDICADORES	'*'

Descrição dos campos:

- Fator 2 (address) - numérico - endereço do nó descritor do arquivo a ser lido.
- Indicador de resultado (byte) - numérico - apenas um indicador.
  Valores: Ø - não especificado.
  >Ø - nº associado ao indicador.
- Demais elementos - idem classe 1 (campos comuns).

* Classe 10

Operação CHAIN, código 21

TAMANHO	CÓDIGO OPER	FATOR 1 Nº RELATIVO	FATOR 2 ENDEREÇO	INDICADOR NÍVEL	INDIC RESULTADO	INDICAD. CÁLCULOS CADEIA INDIC	' '

Descrição dos campos:

- Fator 1 (address) - numérico - número relativo do registro a ser lido (SEEK).
- Fator 2 (address) - numérico - endereço do nó descritor do arquivo a ser lido.
- Indicador de resultado (byte) - numérico - apenas um indicador. Valores: Ø - não especificado.
  >Ø - especificado (nº associado ao indicador).
- Demais elementos - idem classe 1 (campos comuns).

* Classe 11

Operação EXCEPT, código 22

TAMANHO	CÓDIGO OPER	INDICADOR NÍVEL	INDIC. CÁLCULOS CADEIA INDICAD	'*'

Descrição dos campos - idem classe 1 (campos comuns).

* Classe 12

Operação SUBEX, código 23

Esta operação faz chamamento para rotina externa (PLTI ou ASSEMBLER).

TAMANHO	CÓDIGO OPER	FATOR 2 ENDEREÇO	INDICADOR NÍVEL	INDICADORES CÁLCULOS CADEIA DE INDIC.	'*'

Descrição dos campos:

- Fator 2 (address) - numérico - endereço de salto para sub-rotina externa.
- Demais campos - idem classe 1 (campos comuns).

* Classe 13

Operação ENDSR, que determina fim de subrotina - código 24.

TAMANHO	CÓDIGO OPER

Descrição dos campos - idem classe 1 (campos comuns).

4.3.2 - AREAS AUXILIARES

Área de dados - gerada nas seguintes condições:

a) Se fator 1 ou fator 2 são literais, numéricos ou alfanuméricos (para operações que permitem o uso de literais).

b) Se campo de resultado está sendo definido no formulário de cálculos.

Para elementos gerados nesta área, são gerados também os elementos tamanho e TIPO/Posições decimais.

- Tamanho (byte) - numérico.
  Valores: ≤ 15 - se campo numérico.
  ≤ 255 - se campo alfanumérico.
- Tipo (byte) - numérico.
  Valores: = 20 - se campo (ou literal) alfanumérico.
  ≥ Ø - nº de posições decimais, se campo (ou literal) numérico.

CAPÍTULO V - O PROCESSO DE COMPILAÇÃO

5.1 VISÃO GERAL

Verificando as especificações da linguagem observamos que as unidades sintáticas são agrupadas por tipo de especificação e são identificadas através de seu posicionamento no registro fonte.

Observamos ainda que um programa em RPG pode ser considerado como um conjunto de declarações (entradas, cálculos e saídas) para um fluxo padrão de processamento (LÓGICA RPG).

Considerando as características especiais da linguagem, optamos pelo tipo de reconhecimento sintático descendente, específico para a mesma, construído sistematicamente de acordo com o conjunto de regras que mapeiam o fluxo sintático.

Este esquema apresenta a vantagem de ser bastante simples, e de fácil manuseio, facilitando a sua implementação.

Para tanto, teremos um programa principal denominado "COMPILA" chamando quatro(4) módulos de sub-rotinas, cada qual analisando e gerando o código intermediário para determinado tipo de especificação:

- Módulo 1 - sub-rotina COMP$HFDL - trata de especificações de controle, de descrição de arquivos e de extensões/linha.
- Módulo 2 - sub-rotina COMP$ENTRADA - trata das especificações de entrada (registros e campos) relativas a arquivos de entrada.
- Módulo 3 - sub-rotina COMP$CÁLCULOS - trata das especificações de cálculos.
- Módulo 4 - sub-rotina COMP$SAÍDA - trata das especificações de saída (registros e campos) relativas a arquivos de saída, e encerrando a compilação monta o código objeto em formato de entrada compatível ao programa linqueditor, em disco.

Sintaticamente, os módulos são independentes entre si, o que possibilita o tratamento isolado de cada um. Assim, eles são colocados um por vez na memória (OVERLAY), evitando-se problemas de dimensionamento. Sempre na memória teremos um conjunto de SUB-ROTINAS BÁSICAS, auxiliares no reconhecimento lexico, na tabela de símbolos, na geração do código e na escrita de mensagens de erro.

Como principais áreas globais, teremos:
- Tabela de símbolos
- Área de indicadores - vetor que fornece o estado dos indicadores em tempo de compilação (definição e uso).
- Tabela de arquivos lógicos (TAL).
- Tabela de símbolos externos (TSE).
- Buffer para registro fonte.
- Área de geração do código intermediário.

Abaixo descreveremos os elementos.

5.2. SUB-ROTINAS BASICAS

Estas sub-rotinas estarão sempre na memória atendendo aos módulos em overlay.

* * NO RECONHECIMENTO LÉXICO

O reconhecimento léxico para a linguagem foge ao modelo convencional uma vez que é feito para posições determinadas do registro fonte, onde se espera encontrar determinado elemento. Assim sendo, adotaremos um conjunto de sub-rotinas cada qual atendendo uma finalidade.

* * SUB-ROTINA SCAN

```
SCAN:proc(RESULTADO,COLUNA,PT$CADEIA,TAM$CADEIA)
     result(RESULTADO).
```

Descrição

Procura, da esquerda para a direita, na cadeia apontada por PT$-CADEIA, de tamanho TAM$CADEIA, o caráter contido na área de buffer do registro fonte apontado pela variável COLUNA.
Se não encontrou, devolve a variável RESULTADO igual a zero.
Se encontrou, devolve a variável RESULTADO igual ao nº de ordem (1 a TAM$CADEIA) do byte igual ao caráter pesquisado.

* SUB-ROTIMA SCAN$BRANCO

```
SCAN$BRANCO:proc(BRANCO,COLUNA,TAM$BRANCO)
            result(BRANCO)
```

Descrição

Verifica se determinado campo de área de buffer do registro fonte, a partir da posição indicada pela variável COLUNA, com tamanho fornecido pela variável TAM$BRANCO, é uma cadeia de caracteres brancos.
Se a cadeia é formada por brancos, retorna variável BRANCO=TRUE; caso contrário retorna variável BRANCO=FALSE.

* SUB-ROTINA SCAN$NUM

```
SCAN$NUM:proc(VALOR,ERRO,COLUNA,TAM$CAMPO,BR$ESQ)
         result(VALOR,ERRO)
```

Descrição

A rotina verifica se campo da área de buffer do registro fonte, a partir da posição indicada pela variável COLUNA, com tamanho indicado pela variável TAM$CAMPO, contém uma cadeia de caracteres numéricos, com ou sem brancos à esquerda.
A variável BR$ESQ indicada se a cadeia pode ou não possuir brancos à esquerda. Se BR$ESQ=Ø, brancos à esquerda são indesejáveis. Se BR$ESQ=1, nada é requerido.
Na variável result, retornam os seguintes valores:
VALOR - valor binário da cadeia numérica
ERRO Ø - condições satisfeitas (é número) - valor disponível
ERRO 1 - campo branco
ERRO 2 - não é número válido.

* SUB-ROTINA SCAN$NOME

```
SCAN$NOME:proc(ERRO,COLUNA)result(ERRO).
```

Descrição

Verifica se determinado campo contido na área de buffer do regis

tro fonte, a partir da posição indicada pela variável COLUNA, é nome válido em RPG.
Se é nome, devolve a variável ERRO igual a FALSE, caso contrário devolve a variável ERRO igual a TRUE.

```
*   SUB-ROTINA SCAN$IND

    SCAN$IND:proc(ERRO,VALOR$INDIC,CÓDIGO,COLUNA)
             result(ERRO,VALOR$INDIC)
```

Descrição

Verifica se determinado campo de área de buffer do registro fonte, a partir da posição indicada pela variável COLUNA, possui um dos indicadores válidos em RPG ou cadeias 'NS', 'SR', 'AN', 'OR' ou 'Ø', conforme requer a variável CÓDIGO.
Valores da variável CÓDIGO:

CÓDIGO	INDICADORES	OUTRAS CADEIAS
1(COD$0199)	01 - 99	
2(COD$L1L9)	L1 - L9	
3(COD$LR)	LR	
4(COD$H1H9)	H1 - H9	
5(COD$OV)	OV	
6(COD$B)		BRANCO
7(COD$MR)	MR	
8(COD$LO)	LO	
9(COD$1P)	1P	
10(COD$M1M9)	M1 - M9	
11(COD$NS)		NS
12(COD$SR)		SR
13(COD$AN)		AN
14(COD$OR)		OR

Na variável VAL$INDIC retorna um número que é associado a cada indicador para uso na geração e interpretação.

VAL$INDIC

INDICADORES	NºS ASSOCIADOS
01 - 99	01 - 99
H1 - H9	100 - 108
L1 - L9	109 - 117
1P	118
LO	119
LR	120
MR	121
OV	122
HO	123

Na variável ERRO retorna:
Se cadeia não está no campo - ERRO=TRUE; caso contrário, ERRO= FALSE.

```
*  SUB-ROTINA SCAN$LITERAL$NUMÉRICO
   SCAN$LITERAL$NUMÉRICO:proc(ERRO,COLUNA,TAMANHO,ENDER$DESTINO)
                         result(ERRO);
```

Descrição
A sub-rotina verifica incialmente se campo da área de buffer do registro fonte, a partir da posição indicada pela variável COLUNA, com tamanho indicado pela variável TAMANHO, contém um literal numérico.
Se o campo não possui literal numérico, retorna valor TRUE na variável ERRO.
Se o campo possui literal numérico, a sub-rotina seta a variável ERRO=FALSE e move o conteúdo do campo no formato decimal zonado para a área apontada pela variável ENDER$DESTINO.
NOTA: A sub-rotina será usada na compilação de especificações de cálculo (fator 1 ou fator 2).
Literal numérico - ver especificações da linguagem.

```
*   SUB-ROTINA SCAN$LITERAL$ALFANUMÉRICO

    SCAN$LITERAL$ALFANUMÉRICO:proc(ERRO,COLUNA,TAMANHO,ENDER$DES
                                   TINO)result(ERRO);
```

Descrição

A sub-rotina verifica inicialmente, se campo de área de buffer do registro fonte, a partir da posição indicada pela variável COLUNA, com tamanho indicado pela variável TAMANHO, contém um literal alfanumérico.
Se o campo não possui literal alfanumérico, retorna o valor TRUE na variável ERRO.
Se o campo possui literal alfanumérico a sub-rotina seta a variável ERRO=FALSE, e move o conteúdo do campo para a área apontada pela variável ENDER$DESTINO.
NOTA: A sub-rotina será usada na compilação de especificações de cálculo (fator 1 e fator 2) e de especificações de saída.

```
* * NA GERAÇÃO DE CÓDIGO

*   SUB-ROTINA GERA1

    GERA1:proc(VALOR$BYTE);
```

Descrição

Se não houve erro fatal, gera na área do código intermediário o valor byte contido na variável VALOR$BYTE.

```
*   SUB-ROTINA GERA2

    GERA2:proc(VALOR$ADDRESS);
```

Descrição

Se não houve erro fatal, gera na área do código intermediário dois bytes com o valor contido na variável VALOR$ADDRESS.

```
*   SUB-ROTINA GERA3

    GERA3:proc(VALOR$BYTE,ENDER$MEMO);
```

Descrição

Se não houve erro fatal, gera na área do código intermediário no endereço apontado pela variável ENDER$MEMO, o valor byte contido na variável VALOR$BYTE.

```
*   SUB-ROTINA GERAN
    GERAN:proc(VALOR$BYTE,N$VEZES);
```

Descrição

Se não houve erro fatal, gera na área do código intermediário o valor byte contido na variável VALOR$BYTE, no número de vezes fornecido pela variável N$VEZES.

```
* * NO MANUSEIO DA TABELA DE SÍMBOLOS
*   SUB-ROTINA BUSCA$TS
    BUSCA$TS:proc(ACHOU,COLUNA,TAMANHO,TIPO)
             result(ACHOU);
```

Descrição

Faz uma busca na tabela de símbolos, verifica se o identificador contido na área de buffer do registro fonte, a partir da posição indicada pela variável COLUNA, com tamanho especificado pela variável TAMANHO, e tipo especificado pela variável TIPO, está na tabela de símbolos.
A palavra identificador, neste caso, pode ser além de nome de arquivo, campo ou rótulo, nome de indicadores de nível (L1-L9) e match (M1-M9), cujos atributos também são guardados na tabela de símbolos. Ver detalhes da rotina na parte deste trabalho que trata da tabela de símbolos.
Na variável ACHOU, retorna os valores:
se identificador está na TS-ACHOU=TRUE,
caso contrário, ACHOU=FALSE.

* SUB-ROTINA INSERE$TS

INSERE$TS:proc(COLUNA,TAMANHO,TIPO)

Descrição

Insere na tabela de símbolos, o identificador contido na área do buffer do registro fonte, na posição indicada pela variável COLUNA, com tamanho e tipo especificados pelas variáveis TAMANHO e TIPO. Ver detalhes da rotina na parte deste trabalho que trata da tabela de símbolos.

* * MENSAGENS DE ERRO

* SUB-ROTINA ROT$ERRO

ROT$ERRO:proc(NÚMERO$ERRO,TIPO$ERRO)

Descrição

Imprime mensagem de erro de acordo com o conteúdo da variável NÚMERO$ERRO. As mensagens de erro estão contidas em disco (um registro de 80 posições para cada mensagem) em ordem crescente de número do erro.
A rotina acessa o arquivo diretamente de acordo com o conteúdo de NÚMERO$ERRO, traz para a memória a mensagem, e escreve.
Através do parâmetro TIPO$ERRO é informado a rotina se o erro é terminal (T) ou de advertência (W).

5.3 TABELA DE SIMBOLOS

A tabela de símbolos conterá entradas para: nome de arquivo, nome de campo, nome de rótulo (operações GOTO-TAG), nome de rótulo (operações EXSR,BEGSR e ENDSR), nome de indicador de nível, nome de indicador de match.

Organização e métodos de acesso

Optamos pelo uso do método que utiliza HASH, com endereçamento encadeado; neste método, temos os seguintes elementos básicos:

a) Tabela de hash (TAB$HASH-VETOR) - denominada tabela de espalhamento, com K palavras enumeradas de Ø a K-1.
b) Tabela de símbolos propriamente dita.
c) Ponteiro para a próxima posição livre da tabela de símbolos (PTLIVRE).
d) Elemento de ligação, na tabela de símbolos.

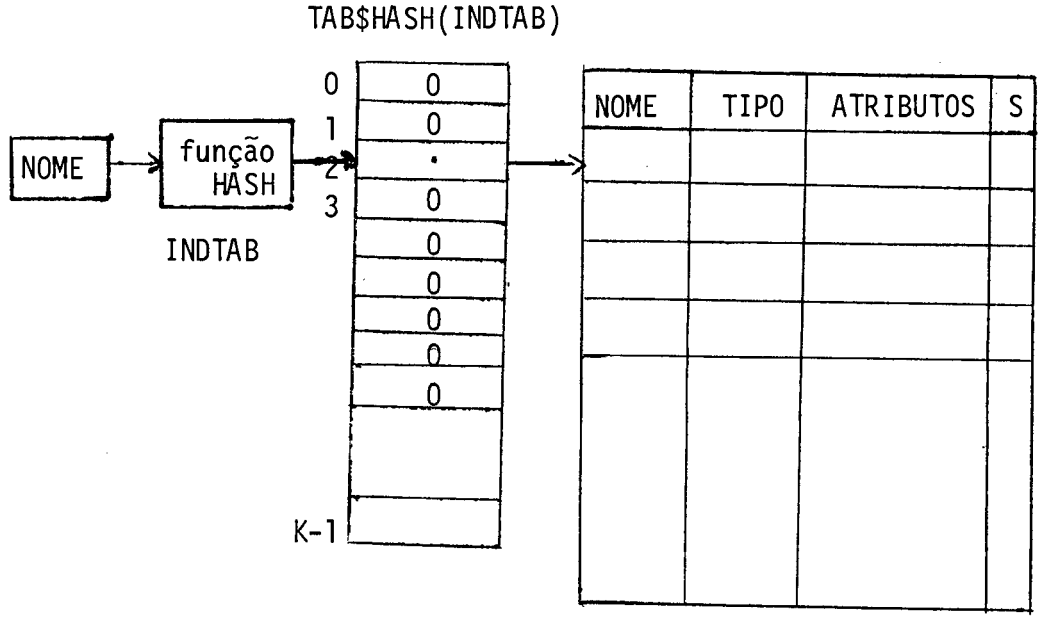


Figura - Tabela de símbolos

Na tabela de espalhamento cada palavra é um ponteiro (cabeça de lista) para K listas encadeadas na tabela de símbolos, nas quais são armazenadas as entradas e seus atributos. Em cada lista teremos os nomes que sofreram colisão no HASH.

No processo de busca temos:

- aplica-se a função de HASH na cadeia de caracteres obtendo-se um número inteiro INDTAB (entre Ø e K-1).
- se TAB$HASH(IND$TAB)=Ø, retorna: ELEMENTO NÃO ENCONTRADO.
- se TAB$HASH(IND$TAB)≠Ø, o elemento poderá ou não estar na tabela. A verificação é feita percorrendo a lista encadeada com cabeça em TAB$HASH(IND$TAB). Se o elemento for encontrado na lista, retorna "ELEMENTO ENCONTRADO", caso contrário retorna "ELEMENTO NÃO ENCONTRADO".

No processo de inserção (caso de não ter sido encontrado o elemento na busca), temos:

- coloca elemento na próxima posição livre da tabela.
- faz encadeamento:
  - Sucessor (PTLIVRE) = TABHASH(INDTAB)
  - TABHASH(INDTAB) = PTLIVRE
- PTLIVRE = PTLIVRE + 1.

Notas: 1) feita a busca com sucesso, ou a inserção, o registro inserido ficará disponível para uso.

2) a função de hash deverá abranger a cadeia de caracteres que compõem o nome mais o tipo (TIPO - ver abaixo).
IND=FHASH(CADEIA"NOME+TIPO")
INDTAB=MOD(IND,K) K:Nº PRIMO, de preferência

3) Alocação: na memória.

4) Tamanho previsto: 1.0K bytes

Organização dos dados

a) Entrada para arquivos

- NOME (5) byte - cadeia de caracteres relativa ao nome.
- TIPO byte - valor numérico=1.
- ATRIBUTOS$1
  Referência - bit 1 Ø - definido
  1 - usado
  Overflow - bit 2 Ø - sem overflow
  1 - com overflow
  Impressora - bit 3 Ø - sem uso da impressora
  1 - com uso da impressora
- ATRIBUTO$2 byte
  Tipo do arquivo - caráter 'I' - entrada
  'O' - saída
  'U' - atualização
- ATRIBUTO$3 address
  Tamanho do registro - valor numérico
- ATRIBUTO$4 byte
  Uso do arquivo - caráter 'p' - principal
  'S' - secundário
  'D' - demanda
  'C' - CHAIN

- ATRIBUTO$5 byte
  Ascendente/descendente - caráter 'A'
  "D"
  "Ø"
- ATRIBUTO$6 address
  Endereço do descritor - numérico.
- SUCESSOR address - elemento de ligação.

b) Entrada para campos

- NOME (5) byte - cadeia de caracteres.
- TIPO byte - valor numérico=2.
- ATRIBUTO$1 byte
  Referência - bit 1: Ø - definido
  1 - usado
- ATRIBUTO$2 byte
  Nº de ordem do registro - valor numérico.
- ATRIBUTO$3 address
  Tamanho do campo - valor numérico.
- ATRIBUTO$4 byte
  Posições decimais/TIPO: *20 - alfanumérico
  nº de Ø a 9 - numérico.
- ATRIBUTO$5 byte
  Meio ajuste - caráter: "H" - com meio ajuste
  "Ø" - sem meio ajuste
- ATRIBUTO$6 address
  Endereço área de dados - numérico.
- SUCESSOR address - elemento de ligação.

c) Entrada para rótulo GOTO/TAG

- NOME (5) byte - cadeia de caracteres (rótulo).
- TIPO byte - valor numérico=5.
- ATRIBUTO$1 byte
  Não usado.
- ATRIBUTO$2 byte
  Não usado
- ATRIBUTO$3 address
  Cabeça de lista de referências futuras - valor numérico.
- ATRIBUTO$4 byte
  Não usado

- ATRIBUTO$5 byte
  Não usado.
- ATRIBUTO$6 address
  Endereço de destino - valor numérico.
- SUCESSOR - address - elemento de ligação.

d) Entrada para rótulo EXSR/BEGSR

- NOME (5) byte - cadeia de caracteres (rótulo).
- TIPO byte - valor numérico=6.
- ATRIBUTO$1 byte
  Não usado.
- ATRIBUTO$2 byte
  Não usado.
- ATRIBUTO$3 address
  Cabeça de lista de referências futuras.
- ATRIBUTO$4 byte - não usado.
- ATRIBUTO$5 byte - não usado.
- ATRIBUTO$6 address
  Endereço de destino - valor numérico.
- SUCESSOR address - elemento de ligação.

e) Entrada para indicador de nível (L1 - L9)

- NOME (5) byte - cadeia de caracteres.
- TIPO byte - valor numérico=3.
- ATRIBUTO$1 byte
  Referência - bit 1: Ø - definido
  1 - usado.
- ATRIBUTO$2 byte
  Nº de ordem de registro - valor numérico.
- ATRIBUTO$3 address
  Tamanho do campo - valor numérico.
- ATRIBUTO$4 byte
  Posições decimais/tipo: * 20 - alfanumérico
  Nº Ø a 9 - numérico.
- ATRIBUTO$5 byte - não usado.
- ATRIBUTO$6 address
  Endereço área nível - valor numérico.
- SUCESSOR address - elemento de ligação.

f) Entrada para indicador match (M1 - M9)

- NOME (5) byte - cadeia de caracteres.
- TIPO byte - valor numérico=4.
- ATRIBUTO$1 byte
  Referência - bit 1: Ø - definido
  1 - usado.
- ATRIBUTO$2 byte
  Nº de ordem do registro - valor numérico.
- ATRIBUTO$3 address
  Tamanho do campo - numérico.
- ATRIBUTO$4 byte
  Posições decimais/tipo: * 20 - alfanumérico
  Nº Ø a 9 - numérico.
- ATRIBUTO$5 byte - não usado.
- ATRIBUTO$6 address - não usado.
- SUCESSOR address - elemento de ligação.

5.4 DESCRICÃO DO PROGRAMA COMPILADOR E SEUS MÓDULOS

Abaixo descreveremos o programa compilador e as sub-rotinas que compõem. Maiores detalhes relativos aos algoritmos podem ser vistos no APÊNDICE A.

5.4.1 PROGRAMA COMPILA

Descrição

Executa a análise do programa fonte e a geração do código intermediário em formato de entrada compatível com o programa linqueditor (REFEX).
É o programa principal que inicializa o processo de compilação e coordena o carregamento (em OVERLAY) dos quatro (4) módulos responsáveis pelo processo de compilação.

5.4.2 SUB-ROTINA COMP$HFDL-MÓDULO 1

```
COMP$HFDL:proc;
```

Descrição

Faz análise das especificações contidas nos formulários de especificações de controle, de descrição de arquivos, de extensão/linha, e coloca o código intermediário correspondente na área de geração.
Com as especificações de arquivos, cria também a tabela de arquivos lógicos (TAL). Com a especificação de controle descobre o nome do programa e o guarda em área reservada. Insere elementos na tabela de símbolos para posterior consulta.

Esquema Básico

```
INICIALIZA ÁREAS;
LÊ E ESCREVE REGISTRO FONTE COLOCANDO NA ÁREA DENOMINADA BUFFER$REG;
CONSISTE REGISTRO FONTE DO TIPO 'H', GUARDANDO NOME DO PROGRAMA EM ÁREA
RESERVADA;
do while HOUVER REGISTRO FONTE DO TIPO DESCRIÇÃO DE ARQUIVO (TIPO 'F');
   CONSISTE REGISTRO FONTE DO TIPO 'F';
   GERA ELEMENTOS NAS LISTAS DE DESCRITORES DE ARQUIVOS DE ENTRADA E SAÍ-
   DA;
   INSERE ELEMENTOS NA TABELA DE SÍMBOLOS;
   INSERE ELEMENTOS NA TABELA DE ARQUIVOS LÓGICO(TAL);
   LÊ E ESCREVE REGISTRO FONTE;
end while;
CONSISTE REGISTRO FONTE DE EXTENSÃO/LINHA (TIPO 'L');
GERA ELEMENTOS RELATIVOS AO REGISTRO FONTE TIPO 'L';
end;
```

5.4.3 SUB-ROTINA COMP$ENTRADA - MÓDULO 2

```
COMP$ENTRADA:proc;
```

Descrição

Faz a análise das informações contidas no formulário de especificações de entrada, que contêm dados relativos a registros, campos, campos com especificações de nível de controle, campos com especificação de combinação (MATCH), para cada arquivo de entrada ou atualização. Executa a geração do código intermediário para os elementos consistidos, que são listas de descritores de re

gistros, de campos, de campos de controle de nível, de campos de combinação, e área auxiliares.

Esquema Básico

```
if NÃO FOR FORMULÁRIO DE ENTRADA then ACUSA ERRO E ENCERRA COMPILAÇÃO;
do while FOR ESPECIFICAÇÃO DE ENTRADA;
   VERIFICA INICIALMENTE SE ESPECIFICAÇÕES DE CAMPO PRECEDEM ESPECIFICA-
   ÇÕES DE REGISTROS;
   VERIFICA ARQUIVO ESPECIFICADO: NOME, CONDIÇÕES DE DEFINIÇÃO E USO -
   SALVA INFORMAÇÕES DA TABELA DE SÍMBOLOS;
   do while NÃO ACABOU ARQUIVO E FOR FORMULÁRIO DE ESPECIF. DE ENTRADA;
      do while FOR ESPECIFICAÇÃO DE REGISTRO;
         if 1ª LINHA DE ESPECIFICAÇÃO DE REGISTRO
            then CONSISTE ESPECIF. DE 1ª LINHA DE REGISTRO GERANDO ELEMEN
                 TOS;
            else CONSISTE ESPECIF. DE LINHA DE CONTINUAÇÃO (AND/OR),  GE
                 RANDO ELEMENTOS;
         LÊ E ESCREVE REGISTRO FONTE;
         if not ESPECIFICAÇÃO DE ENTRADA
            then ENCERRA COMPILAÇÃO ENTRADA;
            else do;
                  if FOR ESPECIFICAÇÃO DE CAMPO
                     then do;
                           GERA FLAG PARA NÓ DESCRITOR REGISTRO;
                           INSERE NÓ NA LISTA DE REGISTROS;
                           INDICA QUE ACABOU ESPECIF. REGISTRO;
                          end;
                end;
      end while REGISTRO;
      do while FOR ESPECIFICAÇÃO DE CAMPO;
       * CONSISTE ESPECIF. CAMPOS - GERA NÓ DESCRITOR; GERA ÁREA DE DADOS
         - INSERE NÓ NA LISTA MOVE;
       * CONSISTE INDICADORES DE NÍVEL - GERA NÓ DESCRITOR LISTA NÍVEL -
         GERA ÁREA NÍVEL - INSERE NÓ NA LISTA NÍVEL;
       * CONSISTE INDICADORES MATCH - GERA NÓ NA LISTA MATCH - INSERE  NÓ
         NA LISTA MATCH;
         LÊ - ESCREVE;
         if ACABOU ESPECIFICAÇÃO DE CAMPO then INDICA QUE ACABOU CAMPOS;
      end while CAMPOS;
```

```
        VERIFICA Nº INDICADORES MATCH NO REGISTRO (SE HOUVE);
        if ACABOU ESPECIF. ARQUIVO then INDICA QUE ACABOU ARQUIVO;
    end while ARQUIVO;
    GERA ÁREA MATCH RELATIVA AO ARQUIVO (SE HOUVE CAMPOS COM INDICADOR
    MATCH) - GERA ENDEREÇOS DE ÁREA MATCH NOS DESCRITORES DAS LISTAS MA
    TCH E NO DESCRITOR DE ARQUIVO;
    GERA ÁREA DE CONTROLE DE SEQUÊNCIA (SE HOUVE REGISTROS COM CONTROLE
    DE SEQUÊNCIA) - GERA ENDEREÇO DA ÁREA NO DESCRITOR DE ARQUIVO;
end while ESPECIFICAÇÕES DE ENTRADA;
end COMP$ENTRADA;
```

Notas Complementares

1. Nomes de campos, indicadores de nível e match

Nomes de campos são ignorados em operações de controle de nível e de controle de match (não existe vinculação nome de campo x indicador). Cada nome de campo, indicador de nível ou indicador de match pode aparecer uma única vez em cada registro. O controle é feito associando-se a cada registro um número de ordem de especificação no programa fonte. A tabela de símbolos conterá para cada campo ou indicador (nível ou match) um atributo indicando em que registro foi referenciado, pela última vez, o campo ou indicador.

2. Campos de sequência, número e opção, para controle de sequência de registros

Verificação sintática conforme especificação da linguagem e geração conforme definição do código intermediário.
Observações importantes:

a) o primeiro registro, no controle, deve conter nº de sequência=01;

b) registros sem controle de sequência devem anteceder registros com controle de sequência;

c) os elementos do primeiro registro com controle de sequência do arquivo, devem ser salvos para geração da área de controle de sequência.

- Mensagens de erro previstas: 35 a 39.

3. Códigos de identificação de registros

- Verificação sintática conforme especificação da linguagem e geração conforme previsão para o código intermediário.

- Observação: o número indicativo da posição deve ser ≤ tamanho do registro
- Mensagens de erro previstas: 41 a 44.

4. Indicadores de match (campos de combinação)
   - Verificação sintática conforme especificações da linguagem e geração conforme previsão para o código intermediário.
   - Observações:
     a) o número de indicadores match deve ser o mesmo em cada registro do programa em que houver especificação;
     b) cada indicador deve ser usado uma única vez em cada registro;
     c) o tamanho e tipo deve ser o mesmo para cada especificação de um indicador match.
- Mensagens de erros previstas: 57 a 59.

5.4.4 SUB-ROTINA COMP$CÁLCULOS - MÓDULO 3

```
COMP$CÁLCULOS:proc;
```

Descrição

Faz a análise das operações de cálculos contidas no formulário de especificação de cálculos, gera o código intermediário relativo a cada operação e insere nós nas listas de cálculo de detalhe ou cálculo de total.

Esquema Básico

```
do while HOUVER ESPECIFICAÇÃO DE CÁLCULOS;
   CONSISTE CONTEÚDO DAS COLUNAS 7-8 (INDICADOR DE NÍVEL);
   -VERIFICA SE A ESPECIFICAÇÃO ESTÁ NA ORDEM DE SEQUÊNCIA ADEQUADA;
   -ABRE OU FECHA LISTAS DE CÁLCULOS DE DETALHE E TOTAL, MARCANDO LIMITES
    DE SALTO PARA OPERAÇÕES GO TO E EXECUÇÃO DE SUB-ROTINAS (EXSR);
   -SALVA INDICADOR DE NÍVEL PARA GERAÇÃO POSTERIOR;
   CONSISTE CONTEÚDO DAS COLUNAS 9-17 (INDICADORES DE CÁLCULOS)
   GERANDO CADEIA DE INDICADORES NUM BUFFER P/POSTERIOR USO NA GERAÇÃO DO
   NÓ DESCRITOR DA OPERAÇÃO;
   DESCOBRE O TIPO DE OPERAÇÃO, CONSISTE ELEMENTOS E FAZ A GERAÇÃO COMPLE
   TA DO NÓ DESCRITOR CORRESPONDENTE;
end while;
```

Notas Complementares

1. Indicadores de cálculos, colunas 9-17

Os indicadores colocados nestas colunas (indicadores de uso pelo programador: indicadores de identificação de registros 01-99, indicadores de nível L1-L9, indicadores de nível L1-L9, indicadores de match M1-M9, e indicadores de parada H1-H9), devem ter sido definidos no formulário de especificações de entrada ou mesmo no formulário de especificações de cálculos como indicadores de resultados.

Como foi visto anteriormente, a cadeia de indicadores condicionantes de operações de cálculos, é gerada inicialmente num buffer, tendo em vista que o tipo de operação ainda não está determinado na fase de consistência de indicadores. O tamanho do buffer condicionará o nº de indicadores a ser utilizado por operação.

2. Operações - comentários

As operações de cálculo serão consistidas e geradas utilizando-se as rotinas básicas previstas, e a geração será feita de acordo com o código intermediário previsto para cada uma.

Comentaremos abaixo, as operações de salto, que possuem um tratamento especial.

OPERAÇÕES GOTO , TAG, EXSR, BEGSR e ENDSR

As operações GOTO/TAG e EXSR/BEGSR sofrerão mesmo tratamento com relação a marcação de endereços. Serão utilizadas listas de referências futuras para a resolução de endereços não resolvidos. Serão listas encadeadas com cabeça na tabela de símbolos (posição relativa ao rótulo armazenado), com nós alocados na área de geração, nos lugares de endereços a serem resolvidos.

A inclusão de nós será feita sempre no início das listas. Uma vez encontrado um rótulo desejado será percorrida a lista respectiva e marcados os endereços destino nas operações de salto gerados (GOTO-EXSR).

Após a marcação a lista torna-se-á vazia.

Na tabela de símbolos, teremos entrada para rótulo de GOTO e para rótulo de EXSR. Como atributos, teremos:

- Cabeça de lista de referências futuras - atributo 3
- Endereço destino - atributo 6

A operação ENDSR, quando rotulada, sofrerá o mesmo tratamento

da instrução TAG.

Na geração: serão geradas instruções para operações GOTO e EXSR. Quando for encontrada uma operação ENDSR será gerado um campo flag, indicativo de final de subrotina.
Vejamos procedimento geral para tratamento das instruções GOTO e TAG.

Operação GOTO

```
ANALISA OPERAÇÃO E GERA CÓDIGO E TAMANHO;
BUSCA POR RÓTULO NA TABELA DE SÍMBOLOS(TS);
if ENCONTROU RÓTULO NA TS
   then do;
        if ENDEREÇO DE DESTINO NÃO DEFINIDO; %RÓTULO DEFINIDO POR INSTRUÇÃO
        %GOTO(ENDEREÇO=Ø)
        then do;
             INSERE NÓ NA LISTA DE REFERÊNCIAS FUTURAS; end;
        else do; %ENDEREÇO DE DESTINO DEFINIDO - HOUVE OPERAÇÃO TAG
             VERIFICA LIMITES DE SALTO;
             GERA ENDEREÇO DESTINO PARA OPERAÇÃO GOTO;
            end;
       end;
   else do;
        INSERE RÓTULO NA TS;
        INSERE NÓ NA LISTA DE REFERÊNCIAS FUTURAS;
       end;
```

Operação TAG

```
ANALISA OPERAÇÃO;
BUSCA POR RÓTULO NA TABELA DE SÍMBOLOS;
if ENCONTROU RÓTULO NA TS
   then do;
        if ENDEREÇO DESTINO NÃO DEFINIDO
           then do;
                MARCA ENDEREÇO DESTINO NA TS;
                PERCORRE LISTA DE REFERÊNCIAS FUTURAS
                GERANDO ENDEREÇO DE DESTINO PARA INSTRUÇÕES GOTO, VERIFI-
                CANDO LIMITES DE SALTO;
               end;
           else ACUSA ERRO; %RÓTULO DUPLICADO
       end;
```

```
else do;
      INSERE RÓTULO NA TS;
      MARCA ENDEREÇO DESTINO NA TS;
     end;
```

Para as operações EXSR e BEGSR, será utilizado o mesmo procedimento utilizado em GOTO e TAG.

Nota: Operação EXSR - execute subrotina
Operação BEGSR - início de subrotina (rotulada)
Operação ENDSR - fim de subrotina (rotulada ou não)
Operação TAG - destino de GOTO (rotulada)

5.4.5 SUB-ROTINA COMP$SAÍDA - MÓDULO 4

```
COMP$SAÍDA:proc;
```

Destino

Faz a análise das especificações contidas no formulário de especificações de saída, que contém informações relativas a registros e campos de cada arquivo de saída ou atualização. Executa a geração do código intermediário para os elementos consistidos (lista de descritores de registros e campos).

Esquema Básico:

```
do while FOR ESPECIFICAÇÃO DE SAÍDA;
   VERIFICA INICIALMENTE SE ESPECIFICAÇÃO DE CAMPO ANTECEDE ESPECIFICAÇÃO
   DE REGISTRO;
   VERIFICA ARQUIVO ESPECIFICADO; NOME, CONDIÇÕES DE DEFINIÇÃO E USO-SAL-
   VA INFORMAÇÕES DA TABELA DE SÍMBOLOS;
   do while NÃO ACABOU ESPECIF. DO ARQUIVO E FOR
      FORMULÁRIO DE SAÍDA;
      do while FOR ESPECIF. DE REGISTRO;
         if 1ª LINHA DE ESPECIFICAÇÃO DE REGISTRO;
               then CONSISTE ESPECIF. DE 1ª LINHA DE REGISTRO GERANDO ELE
                    MENTOS;
               else CONSISTE ESPECIF. DE LINHA DE CONTINUAÇÃO (AND/OR) GE
                    RANDO ELEMENTOS;
         LÊ E ESCREVE REGISTRO FONTE;
         if NÃO É ESPECIFIC. DE SAÍDA;
```

```
                then ENCERRA COMPILAÇÃO DE SAÍDA;
                else do;
                     VERIFICA SE É ESPECIF. DE CAMPO;
                     if FOR ESPECIF. DE CAMPO;
                        then do;
                             GERA FIM (FLAG) PARA NÓ DESCRITOR DE REGISTRO
                             E INSERE NÓ NA LISTA DE REGISTROS DE SAÍDA;
                             FAZ INDICAÇÃO QUE ACABOU ESPEC. DE REGISTRO;
                            end;
                    end;
        end while ESPECIF. REGISTRO;
        do while FOR ESPECIF. CAMPO;
           CONSISTE ESPECIF. CAMPO GERANDO ELEMENTOS;
           INSERE NÓ GERADO NA LISTA DE CAMPOS;
           LÊ E ESCREVE;
           if ACABOU ESPECIF. CAMPO;
              then GERA FLAG NO FIM DA LISTA DE CAMPOS, INDICA QUE ACABOU ES-
                   PECIF. CAMPO;
        end while ESPECIF. CAMPO;
        VERIFICA SE TERMINOU ESPECIF. DO ARQUIVO;
        if TERMINOU FAZ INDICAÇÃO QUE TERMINOU ARQUIVO;
     end while ARQUIVO;
  end while ESPEC. SAÍDA;
  %ENCERRA COMPILAÇÃO
  MONTA EM DISCO, O CÓDIGO GERADO EM FORMATO DE ENTRADA COMPATÍVEL COM O PRO-
  GRAMA LINQUEDITOR(REFEX);
  end;
```

Notas Complementares

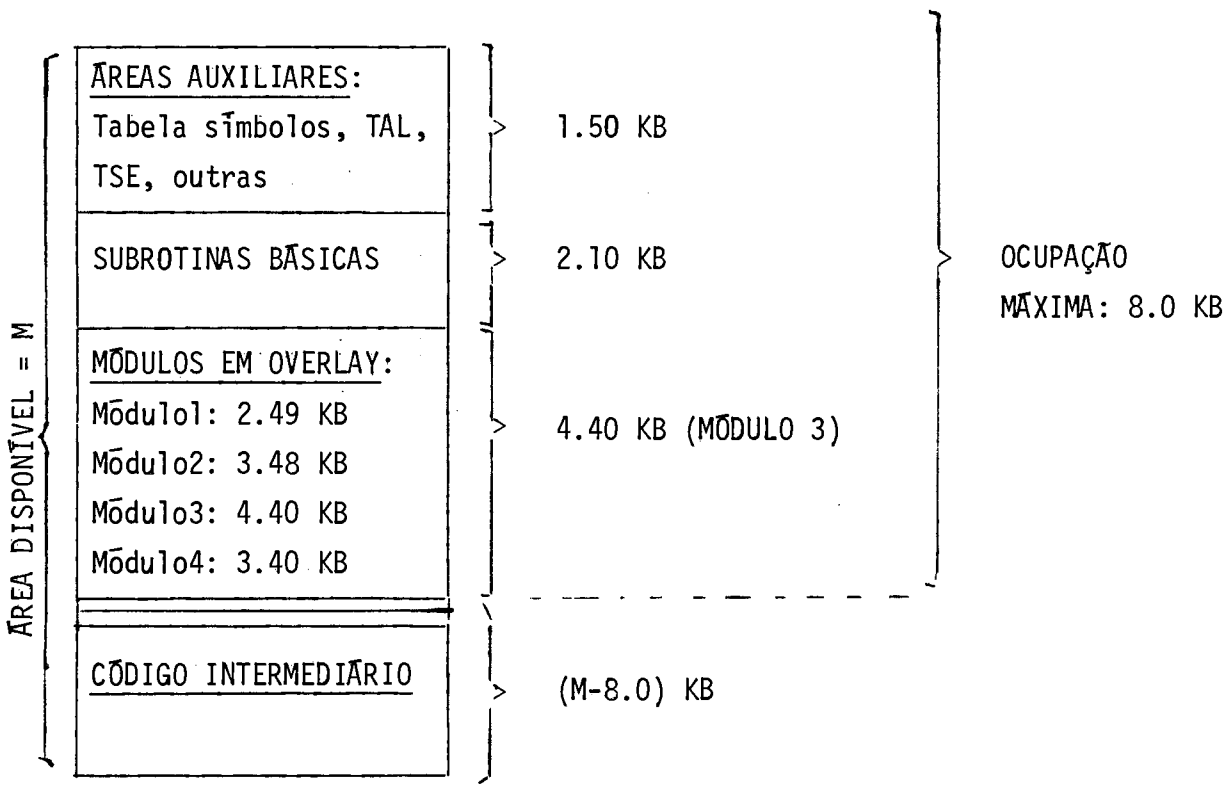
1. Espaço e salto na saída (SPACE/SKIP)-colunas 17-22
   - Verificação sintática conforme especificações da linguagem e geração conforme previsão para o código intermediário.
   - Observações importantes:
     a) Especificações de espaço e salto somente são válidos para arquivos na impressora.
     b) Verificar limitações do nº de linhas de espacejamento e salto.
   - Mensagens de erro previstas: 159, 160, 184.

2. Indicadores de saídas-colunas 23-31

- Verificação sintática conforme especificações da linguagem e geração conforme previsão para o código intermediário.
- Observações importantes:
  a) Indicador especificado tem que ter sido definido anteriormente nos formulários de entrada ou de cálculos.
  O controle de definição e uso é feito através de um vetor CONTROLE$IND(132) Byte, sempre na memória, em que cada posição indica o estado no indicador. O vetor é indexado pelo número que é associado a cada indicador no processo de compilação.
  Assim:
  CONTROLE$IND(01) - fornece o estado do indicador de identificação de registro 01.
  Estados: Ø - não definido
  1 - definido
  2 - definido e usado.
  Indicadores internos: estado inicial=2.
  b) Indicador de overflow, se usado, tem que ter sido especificado para arquivo, na impressora, correspondente (formulário de especificações de descrição de arquivos).
  c) Indicador 1P somente pode ser usado para arquivo na impressora.
  d) Linhas AND/OR, devem possuir pelo menos uma especificação de indicadores.
  e) Indicadores de saída para campos podem ser no máximo em número de três (sem linha AND/OR).
- Mensagens de erro previstas: 161 a 167, 171, 172.

5.5. ALOCACÃO DE MEMÓRIA - DADOS EXPERIMENTAIS

O programa principal de compilação e subrotinas correspondentes foram codificados em PLTI e implementados experimentalmente no terminal, tendo-se obtido a ocupação de memória mostrada a baixo:



A área disponível para o código intermediário (gerado na memória) será de (M-8.0) KB, aproximadamente.

CAPÍTULO VI - O PROCESSO DE INTERPRETAÇÃO

6.1 - A LÓGICA RPG PARA O INTERPRETADOR

Os programas em RPG são executados num mesmo padrão lógico, denominado 'LÓGICA RPG'.

A execução da LÓGICA RPG será tarefa do programa interpretador, a ser visto no próximo título deste trabalho, para o terminal inteligente.

Com o auxílio da figura(VI-1), descreveremos os passos da LÓGICA RPG, apontando desde já as subrotinas associadas a cada um, no programa interpretador. Maiores detalhes podem ser vistos na própria rotina de interpretação.

DESCRIÇÃO DOS PASSOS (Numeração na figura(VI-1)).

1. Inicialização
   - São desligados todos os indicadores, menos os indicadores 1P e LØ.
   - É verificado se o programa é de processamento com multiplos arquivos (ao menos um arquivo primário e 1 arquivo secundário).

2. São executadas saídas condicionadas pelo indicador 1P (indicador de 1ª página). Esta saída é feita uma só vez. SUBROTINA: EXEC$SAÍDA$1P.

3. São executadas as saídas do tipo cabeçalho/detalhe cujas condições estiverem satisfeitas. Se atingida linha de overflow na impressão de relatórios no arquivo impressora, é ligado o indicador de overflow. SUBROTINA EXEC$SAÍDA.

4. O interpretador verifica indicadores de parada. Se houver algum indicador ligado é chamada a SUBROTINA PARADA, que faz comunicação com o operador (salto para o passo 32). VER SUBROTINAS TESTA$PARADA - PARADA.

FLUXO : LÓGICA RPG

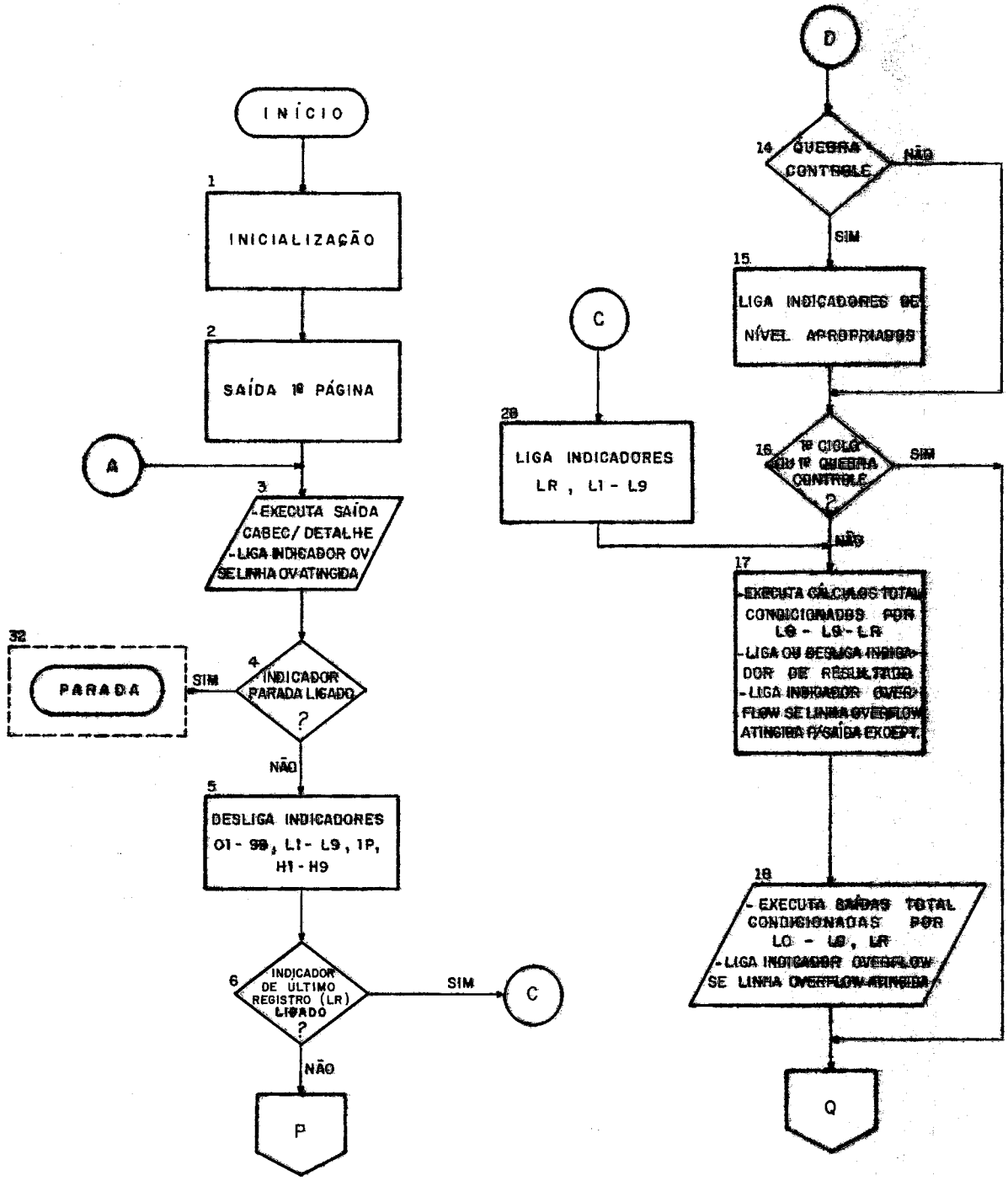
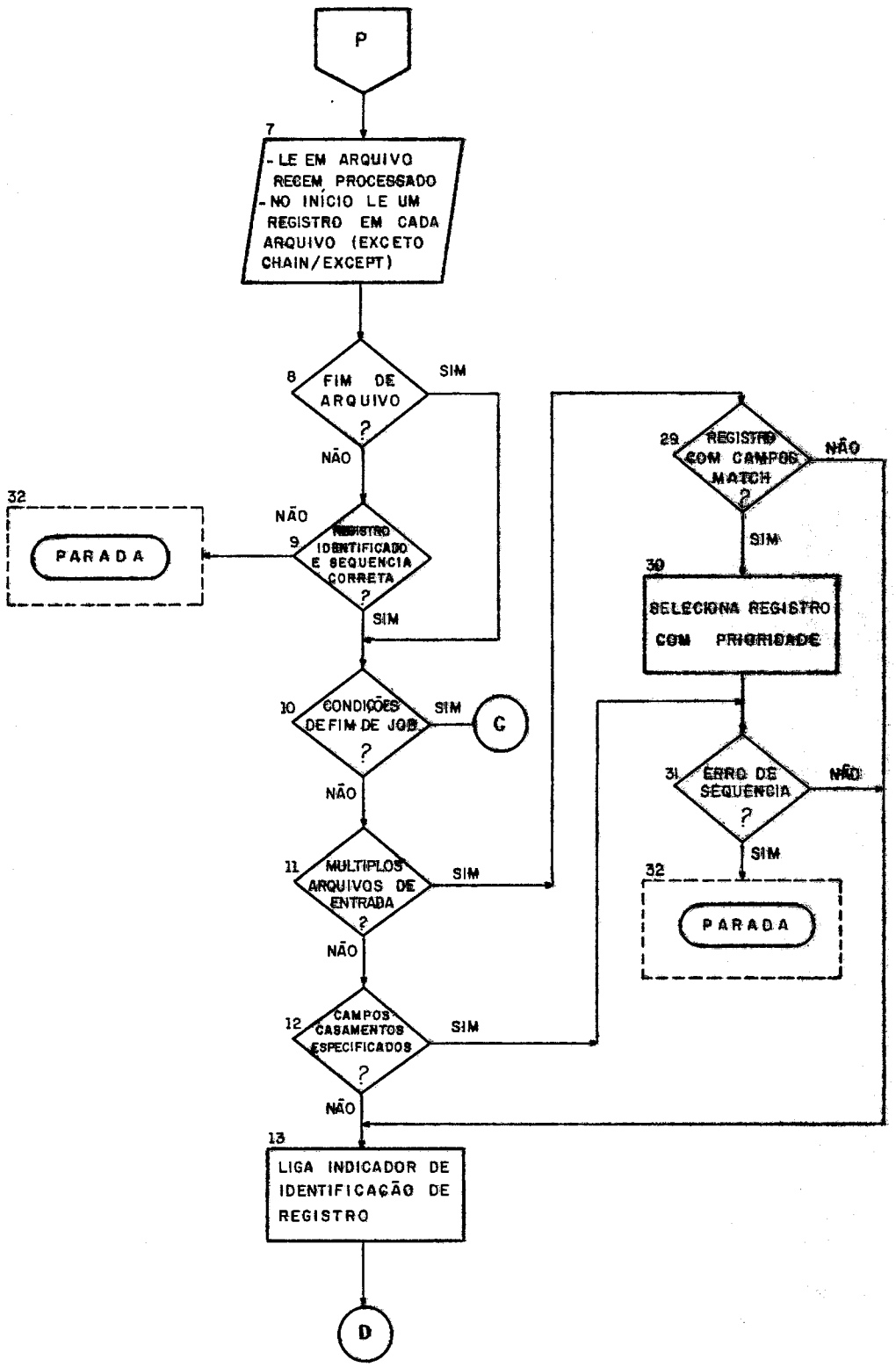
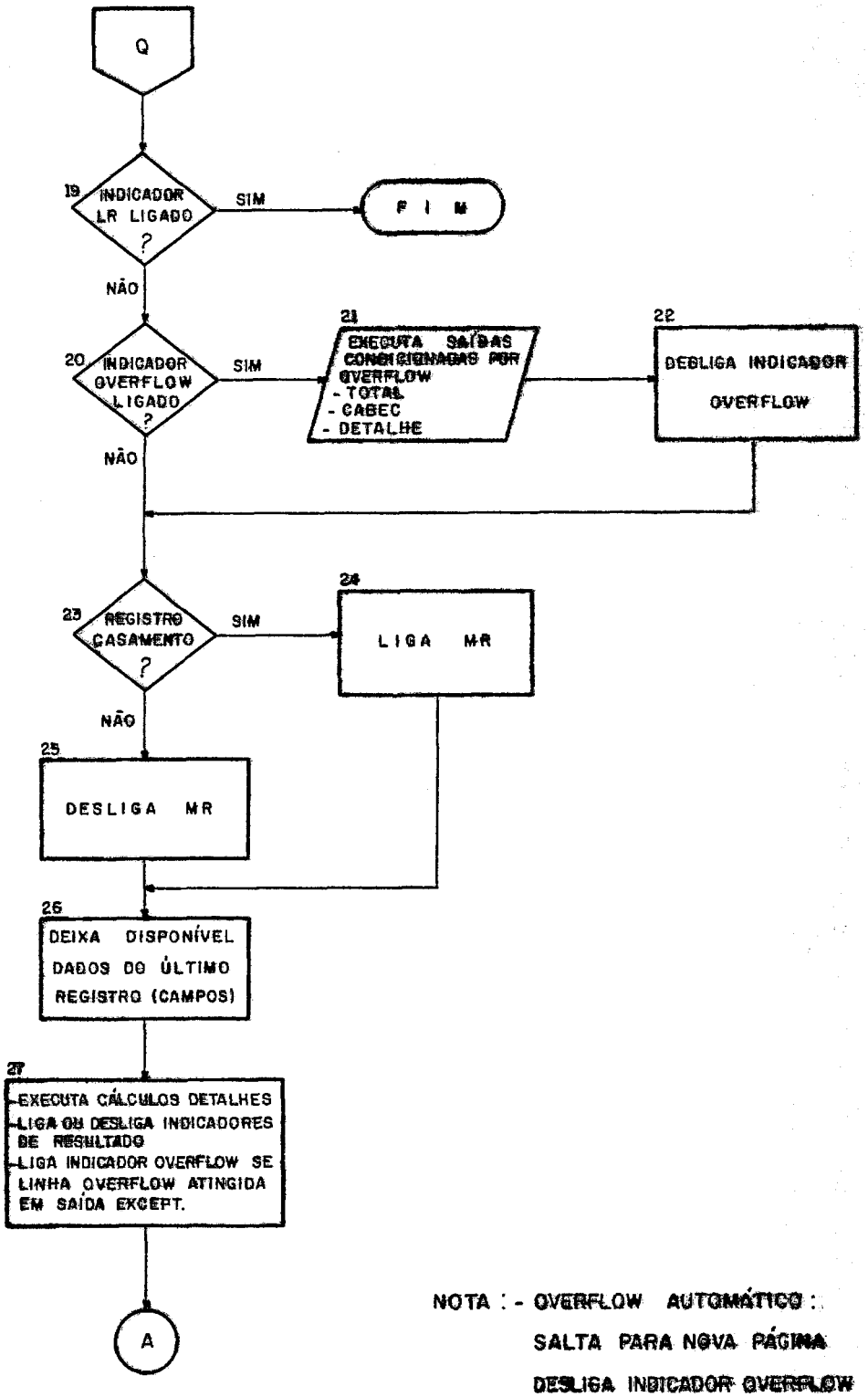


FIGURA VI-1/LÓGICA RPG


P
7
- LE EM ARQUIVO RECEM PROCESSADO
- NO INÍCIO LE UM REGISTRO EM CADA ARQUIVO (EXCETO CHAIN/EXCEPT)
8
FIM DE ARQUIVO ?
SIM
NÃO
32
PARADA
NÃO
9
REGISTRO IDENTIFICADO E SEQUENCIA CORRETA ?
SIM
10
CONDIÇÕES DE FIM DE JOB ?
SIM
C
NÃO
11
MULTIPLOS ARQUIVOS DE ENTRADA ?
SIM
NÃO
12
CAMPOS CASAMENTOS ESPECIFICADOS ?
SIM
NÃO
13
LIGA INDICADOR DE IDENTIFICAÇÃO DE REGISTRO
D
29
REGISTRO COM CAMPOS MATCH ?
NÃO
SIM
30
SELECIONA REGISTRO COM PRIORIDADE
31
ERRO DE SEQUÊNCIA ?
NÃO
SIM
32
PARADA


Q
19
INDICADOR LR LIGADO ?
SIM
F I M
NÃO
20
INDICADOR OVERFLOW LIGADO ?
SIM
21
EXECUTA SAÍDAS CONDICIONADAS POR OVERFLOW
- TOTAL
- CABEC
- DETALHE
22
DESLIGA INDICADOR OVERFLOW
NÃO
23
REGISTRO CASAMENTO ?
SIM
24
L I G A M R
NÃO
25
D E S L I G A M R
26
DEIXA DISPONÍVEL DADOS DO ÚLTIMO REGISTRO (CAMPOS)
27
-EXECUTA CÁLCULOS DETALHES
-LIGA OU DESLIGA INDICADORES DE RESULTADO
-LIGA INDICADOR OVERFLOW SE LINHA OVERFLOW ATINGIDA EM SAÍDA EXCEPT.
A
NOTA : - OVERFLOW AUTOMÁTICO :
SALTA PARA NOVA PÁGINA
DESLIGA INDICADOR OVERFLOW

5. São desligados os indicadores de identificação de registros (01-99), indicadores de contrôle de nível (L1 - L9) e indicadores de parada (H1 - H9).

6. O interpretador verifica se o indicador de último registro (LR) está ligado.
Caso afirmativo, salta para o passo nº 28.

7. O programa lê registro do arquivo recém processado. No início, lê um registro de cada arquivo. Arquivos encadeados e de demanda não são lidos neste passo. Para estes a leitura é feita em tempo de execução de cálculos através de comandos READ e CHAIN, especificados pelo usuário.

8. O interpretador verifica se a condição de fim de arquivo ocorreu. Se ocorreu, salta para o passo nº 10.

9. O interpretador verifica a identificação do registro lido (se confere com as especificações do formulário de entrada) e verifica também a seqüência de entrada, se foi especificado contrôle de seqüência. Ver detalhes nas SUBROTINAS IDENTIFICA$REG e VERIFICA$SEQUENCIA.
Caso o registro não seja identificado, ou esteja fora de seqüência, salta para o passo nº 32, onde é chamada a SUBROTINA PARADA.

10. Se condições de fim de programa foram alcançadas, salta para passo 28. No caso de processamento com multiplos arquivos , todos aqueles que possuem especificação 'E' na coluna 17 devem estar em condição de fim de arquivo. Ver SUBROTINA TESTAFIMJOB.

11. Se o processamento é com multiplos arquivos de entrada (ao menos 1 arquivo principal e 1 secundário) é necessário selecionar um registro. Neste caso, salta para passo 29. Ver SUBROTINAS SELECIONA$MATCH E VERIFICA$SEQUENCIA$MATCH.

12. Se existe apenas um arquivo de entrada, não é necessário a seleção. Se o registro possui campos match especificados um teste de seqüência é feito, passo 31. Ver SUBROTINA VERIFICA$-

SEQUENCIA$MATCH.

13. É ligado o indicador de identificação para o registro selecionado.

14. O interpretador verifica se houve quebra de contrôle , ou seja, verifica se o conteúdo de campos de contrôle é diferente do conteúdo dos campos no ciclo anterior. Ver SUBROTINA QUEBRA$CONTROLE. Se não houve quebra de controle salta para o passo 16.

15. Se houve quebra de controle, o indicador de nível de controle especificado é ligado, bem como os demais indicadores de nível inferior. Ver SUBROTINA QUEBRA$CONTROLE.

16. Se for 1º ciclo, ou 1ª quebra de controle, o interpretador ignora cálculos e saídas de total e salta para passo nº 19.

17. São executados cálculos de total, se condições satisfeitas (indicadores de controle ligados). Indicadores de resultados são ligados ou desligados. Se atingida linha de overflow através saídas EXCEPT é ligado indicador de overflow. Ver SUBROTINA EXEC$OPERAÇÃO.

18. São executadas saídas total, não condicionadas por indicadores overflow, se condições satisfeitas. Se linha de overflow foi atingida é ligado indicador de overflow. Ver SUBROTINA EXEC$SAIDA.

19. Se o indicador de último registro estiver ligado, teremos o final da execução da interpretação.

20. 21. 22. Se indicador de overflow estiver ligado temos 2 opções:

a) Se contrôle de overflow é feito pelo programador através da especificação no formulário de descrição de arquivos, são executadas as saídas condicionadas por overflow no formulário de especificações de saída, na ordem, saídas de total, saídas de cabeçalho e saídas de detalhe, havendo salto de página conforme es

pecificações.

b) Se controle de overflow é automático, por conta do sistema, é executado salto para nova página, e desligado indicador de overflow.
Ver SUBROTINA EXEC$SAIDA$OV.

23. 24. 25. O indicador MR é ligado se é processamento de múltiplos arquivos e o registro selecionado é um registro de casamento. Caso contrário é desligado.

26. Deixa disponível dados do último registro selecionado.

27. O interpretador executa os cálculos detalhe para operações não condicionadas por indicadores, ou para operações com condições especificadas e satisfeitas. São ligados ou desligados indicadores de resultado.

Se atingida linha de overflow, através saídas previstas para operação EXCEPT, é ligado o indicador de overflow. Ver SUBROTINA EXEC$OPERAÇÃO.

28. É ligado o indicador LR e demais indicadores de controle de nível (L1 - L9) e o processamento continua no passo 17 (Cálculos e saída total).

29. 30. Se um registro sem campos de match especificado, é encontrado num arquivo de entrada que não está no estado de fim de arquivo ele é selecionado.

Quando o registro possue campos de match especificados, e acionado um procedimento de seleção de registros, sendo escolhido o registro com maior prioridade. Ver SUBROTINA SELECIONA$-MATCH.

31. O valor de match do campo é comparado com o valor de match do campo do registro anterior. Se não houver erro de seqüência, o processamento continua para o registro selecionado. Caso contrário é chamada a subrotina PARADA que interrompe o processamento. Ver SUBROTINA VERIFICA$SEQUENCIA$MATCH.

32. A execução do programa sofre interrupção. O interpretador se comunica com o operador que terá opção de continuar ou en

cerrar o processamento. Ver SUBROTINA PARADA.

6.2 - O INTERPRETADOR

A interpretação será realizada pela subrotina INTERPRETA que carregada na memória executará a lógica RPG, utilizando o código intermediário gerado, também na memória.

A subrotina INTERPRETA foi dividida em partes que compõem um conjunto de subrotinas auxiliares básicas, procurando-se fazer cada uma corresponder a um procedimento no fluxo da lógica RPG. Abaixo fazemos a descrição:

Subrotinas previstas: (escopo)

```
1. INTERPRETA: proc;
   2. PARADA: proc (DISPLAY);
   2. IDENTIFICA$REG: proc (ENDER$REG$LIDO) result (ENDER$
                      REG$LIDO);
   2. VERIFICA$SEQUENCIA: proc (ENDER$REG$LIDO);
   2. INICIALIZA$ARQ: proc (TIPO$ARQUIVO);
   2. QUEBRA$CONTROLE: proc (HOUVE$QUEBRA,ENDER$REG$LIDO)
                       result (HOUVE$QUEBRA);
   2. TESTA$IND: proc (OK, ENDER$CADEIA) result (OK);
   2. EXEC$SAIDA: proc (TEMPO$SAIDA),
   2. EXEC$SAIDA$OV: proc (AUTOMÁTICO) result (AUTOMÁTICO);
   2. EXEC$SAIDA$1P; proc;
   2. MONTA$ESCREVE: proc (ENDER$SPACE$SKIP, ENDER$NO$CAM-
                     PO, TIPO$SAIDA);
   2. IO: proc (ENDER$ARQ, ENDER$BUFFER, OPERAÇÃO, NÚMERO$
          SEEK);
   2. TESTA$PARADA: proc;
   2. TESTA$FIM$JOB: proc (FIM$JOB);
   2. TESTA$CONDIÇÕES$SAIDA: proc (SATISFAZ, ENDER$SPACE$
                             SKIP, ENDER$NO$REG) result
                             (SATISFAZ, ENDER$SPACE$SKIP);
   2. SELECIONA$MATCH; proc;
      3. COMPARA$MATCH. proc;
   2. VERIFICA$SEQUENCIA$MATCH: proc;
   2. MOVE$MATCH: proc (ENDER$REG$LIDO);
   2. EXEC$OPERAÇÃO: proc;
```

* * SUB-ROTINA INTERPRETA

INTERPRETA: proc;

Descrição

Executa a lógica RPG utilizando o código intermediário gerado (conjunto de descritores). Áreas com destaque utilizadas pela sub-rotina:

- Área MEMO - conterá o código intermediário
- Áreas de trabalho para controle de sequência match, e para salvar campos match, quando houver processamento com múltiplos arquivos com campos match (de casamento) especificados
- Área de trabalho para armazenamento de valores temporários em subrotinas de cálculos
- Vetor SITUAÇÃO$IND(132)byte que é utilizado para controle de estado de indicadores

Algoritmo

INTERPRETA: proc;

%INICIALIZAÇÃO DE PRINCIPAIS ÁREAS/VARIÁVEIS - PASSO 1

ARQ$PRIMÁRIO$INICIALIZADO=FALSE; %VARIÁVEL LÓGICA PARA CONTROLE DE INICIALI-
%ZAÇÃO DE ARQUIVOS PRIMÁRIOS.

INICIALIZOU$TODOS=FALSE; %VARIÁVEL LÓGICA PARA CONTROLE DE INICIALIZAÇÃO DE
%ARQUIVOS SECUNDÁRIOS.

HOUVE%MATCH=FALSE; %VARIÁVEL LÓGICA PARA CONTROLE DE CASAMENTO ENTRE REGIS-
%TROS.

PRIMEIRAVEZMATCH=TRUE; %VARIÁVEL LÓGICA PARA INDICAR SE É OU NÃO A 1ª CHA-
%MADA PARA A SUB-ROTINA DE SELEÇÃO DE REGISTROS COM
%MATCH.

PRIMEIRO$CICLO=TRUE; %UTILIZADA PARA INDICAÇÃO DE 1º CICLO DO PROCESSAMENTO.

PRIMEIRA$QUEBRA=FALSE; %UTILIZADA PARA INDICAÇÃO DE 1ª QUEBRA DE CONTROLE DE
%NÍVEL DO PROCESSAMENTO.

TIPO$REG=PRIMÁRIO; %VARIÁVEL QUE INDICARÁ O TIPO DE ARQUIVO DO QUAL FOI LIDO
%O ÚLTIMO REGISTRO.

INDICADOR$MR=FALSE; %VARIÁVEL UTILIZADA EM PROCESSAMENTO MULTIFILE, INDICAN-
%DO SE INDICADOR MR DEVE OU NÃO SER LIGADO: HÁ MATCH OU
%NÃO.

DESLIGA TODOS OS INDICADORES NO VETOR SITUAÇÃO$IND, QUE FORNECE O ESTADO DE INDICADORES (LIGADOS/DESLIGADO), MENOS OS INDICADORES 1P (1ª PÁGINA) E LØ;

```
PERCORRE A LISTA DE ARQUIVOS DE ENTRADA (DESCRITORES) FAZENDO:
    i) SALVA ENDEREÇO DO ARQUIVO PRIMÁRIO E SEU BUFFER NAS VARIÁVEIS ENDER$
       ARQ$LER E ENDER$BUFFER$LER, VARIÁVEIS ESTAS QUE FORNECERÃO NO PROCES
       SO DE INTERPRETAÇÃO, OS ENDEREÇOS RELATIVOS AO PRÓXIMO ARQUIVO A SER
       LIDO.
   ii) VERIFICA SE O PROFRAMA POSSUE PROCESSAMENTO COM MÚLTIPLOS ARQUIVOS.
       CASO POSITIVO SETA A VARIÁVEL LÓGICA 'MULTIFILE=TRUE, CASO CONTRÁRIO
       MULTIFILE=FALSE;

%FIM DA INICIALIZAÇÃO * * * * * * * * * *
%EXECUTA SAÍDA DE 1ª PÁGINA - PASSO 2
call EXEC$SAIDA$1P;

%EXECUTA SAIDA CABEÇALHO/DETALHE - PASSO 3
INÍCIO:          %(RÓTULO)
call EXEC$SAIDA(TEMPO$CABEC$DETALHE);

%VERIFICA INDICADORES DE PARADA (H1 - H9) - PASSO 4
call TESTA$PARADA;

%DESLIGA INDICADORES NO VETOR SITUAÇÃO$IND - PASSO 5
DESLIGA INDICADORES DE IDENTIFICAÇÃO DE REGISTRO(01-99) DE CONTROLE DE NÍVEL
(L1-L9), DE 1ª PÁGINA (1P), E DE PARADA (H1-H9);

%TESTA INDICADOR LR - PASSO 6
VERIFICA VETOR SITUAÇÃO$IND;
if INDICADOR LR LIGADO
   then do; LIGA INDICADORES L1-L9; Go to TOTAL; end;

%LÊ ARQUIVO - PASSO 7
if MULTIFILE    %PROCESS. COM MÚLTIPLOS ARQUIVOS DE ENTRADA: 1 ARQ. PRIMÁRIO
                %E 1 OU MAIS ARQ. SECUNDÁRIOS
   then do;
        if not ARQUIVO$PRIMÁRIO$INICIALIZADO
           then call INICIALIZA$ARQ(PRIMÁRIO);
           else if not INICIALIZOU$TODOS
                   then call INICIALIZA$ARQ(SECUNDÁRIO);
        %LÊ ARQUIVO RECÉM PROCESSADO
        X=IO(ENDER$ARQ$LER, END$BUFFER$LER, READ, 0);
        if X=CÓDIGO$ERRO
           then call PARADA(DISPLAY$ERRO);
```

```
         if X=EOF   %FIM DE ARQUIVO
            then MARCA NO DESCRITOR DO ARQUIVO ESTADO=EOF;
       end;
    else do; %PROCESSAMENTO COM 1 ARQUIVO ENTRADA
          X=IO(ENDER$ARQ$LER,ENDER$BUFFER$LER,READ,Ø);
          if X=CÓDIGO$ERRO
             then call PARADA(DISPLAY$ERRO);
          if X=EOF
             then MARCA NO DESCRITOR DO ARQUIVO ESTADO=EOF;
       end;

%TESTA FIM ARQUIVO - IDENTIFICA REGISTRO - VÊ SEQUENCIA REGISTROS
%TESTA FIM DE JOB - PASSOS 8 - 9 - 10
if X <> EOF     %NÃO É FIM DE ARQUIVO
   then do;
         call IDENTIFICA$REG(ENDER$REG$LIDO);
         call VERIFICA$SEQUÊNCIA;
       end;
call TESTA$FIM$JOB(FIM$JOB);
FINAL:            %RÓTULO
if FIM$JOB
   then do;
         LIGA INDICADORES LR,L1-L9;
         go to TOTAL;
       end;

%VERIFICA SE É PROCESS. MULTIFILE, SE FOR PROVIDENCIA
%SELEÇÃO E VERIFICAÇÃO DE SEQUÊNCIA - PASSOS 11, 29, 30, 31
SELECIONA:            %RÓTULO
if MULTIFILE
   then if REGISTRO LIDO POSSUI CAMPOD DE MATCH
           then do; %NÓ DESCRITOR LISTA MATCH≠VAZIA
                 call SELECIONA$MATCH;
                 call VERIFICA$SEQUÊNCIA$MATCH;
              end;
   else            %NÃO É MULTIFILE - PASSO 12
        if REGISTRO LIDO POSSUE CAMPOS MATCH
           then call VERIFICA$SEQUÊNCIA$MATCH;
```

```
%LIGA INDICADOR DE IDENTIFICAÇÃO DE REGISTRO - PASSO 13
PROCESSA:            %RÓTULO
LIGA INDICADOR DE IDENTIFICAÇÃO DE REGISTRO PARA O REGISTRO SELECIONADO;
                              %INDICADOR ESTÁ NO DESCRITOR - PARA LIGAR BAS
                              %TA INDEXAR VETOR SITUAÇÃO$IND E MARCAR 'LIGA
                              %DO'.

%TESTA QUEBRA DE CONTROLE PARA REGISTRO SELECIONADO, LIGA INDICADORES SE  NE
%CESSÁRIO, VERIFICA CICLO, EFETUA CÁLCULOS E SAÍDA DE TOTAL SE NECESSÁRIO  -
 PASSOS 14 - 15 - 16 - 17 - 18
call QUEBRA$CONTROLE(HOUVE$QUEBRA,ENDER$REG$LIDO);
if PRIMEIRO$CICLO
   then do;
        PRIMEIRO$CICLO=FALSE;
        go to CONTINUA;
       end;
if HOUVE$QUEBRA AND PRIMEIRA$QUEBRA=FALSE
   then do;
        PRIMEIRA$QUEBRA=TRUE;
        go to CONTINUA;
       end;
TOTAL:          %RÓTULO
call EXEC$OPERAÇÃO(TEMPO$TOTAL); %EXECUTA OPERAÇÕES SE CONDIÇÕES SATISFEITAS

call EXEC$SAÍDA(TEMPO$TOTAL);    %EXECUTA SAÍDAS TOTAL SE CONDIÇÕES SATISFEI
                              %TAS
%VERIFICA INDICADOR LR - PASSO 19

CONTINUA:       %RÓTULO
if INDICADOR LR LIGADO           %VERIFICANDO VETOR SITUAÇÃO$IND
   then go to FIM;

%VERIFICA INDICADOR DE OVERFLOW - PROVIDENCIA SAÍDAS CONDICIONADAS POR OVER-
%FLOW SE NECESSÁRIO - PASSOS 20 - 21 - 22
if INDICADOR OVERFLOW LIGADO
   then do;
        call EXEC$SAIDA$OV(AUTOMÁTICO);
        if AUTOMÁTICO            %CONTROLE AUTOMÁTICO - PROVIDENCIA SALTO PÁ
                              %GINA
           then do;
```

```
                ACESSA ARQUIVO IMPRESSORA NA LISTA DE DESCRITORES DE  ARQ.
                SAÍDA ;
                ACESSA BUFFER RELATIVO AO ARQUIVO;
                MOVE CARÁTER DE CONTROLE PARA POSIÇÃO RESERVADA DO BUFFER;
                X=IO(ENDER$ARQ,ENDER$BUFFER,WRITE,Ø);
                ATUALIZA CONTADOR DE LINHAS;
                DESLIGA INDICADOR DE OVERFLOW
              end;
        end OVERFLOW;

%LIGA OU DESLIGA INDICADOR MR - CONTROLE MATCH - PASSOS 23, 24, 25
if INDICADOR$MR=TRUE          %SETADO NA SELEÇÃO
   the LIGA INDICADOR MR;
   else DESLIGA INDICADOR MR;

%DEIXA DISPONÍVEL DADOS DO ÚLTIMO REGISTRO SELECIONADO - PASSO 26
PERCORRE LISTA DE CAMPOS MOVE RELATIVA DO REGISTRO SELECIONADO E MOVE O CON
TEÚDO DO BUFFER DE ENTRADA; CORRESPONDENTE AO CAMPO, PARA A ÁREA DE    DADOS
RESERVADA PARA O CAMPO;

%EXECUTA CÁLCULOS DETALHE - PASSO 27
call EXEC$OPERAÇÃO(TEMPO$DETALHE);

%VOLTA AO INÍCIO
go to INÍCIO;

FIM:            %RÓTULO

end INTERPRETA;


* *  SUB-ROTINA PARADA

     PARADA:proc(DISPLAY);
```

Descrição

Faz comunicação com o operador, mostrando o indicador de parada setado no programa ou outros indicadores de erro na execução. Serão oferecidas ao operador as opções abaixo, dependendo do tipo de ocorrência:

opção 1 - cancelamento imediato do programa

opção 2 - cancelamento controlado - o programa executa cálculos

e saídas condicionadas pelo indicador LR e encerra execução.

opção 3 - continua - o controle volta ao programa num ponto adequado (no caso de indicadores de parada volta ao ponto de chamamento da rotina PARADA).

- Indicadores de parada:
  H1 a H9 - de uso pelo programador
  H0 - interno, ligado quando os demais (H1 a H9) estiverem ligados.
- Indicadores de erro:
  A serem definidos na fase de implementação. Principais tipos de erros: registro não identificado, erro de sequência de registros, erro de sequência match, divisão por zero, etc.

Parâmetros:
  DISPLAY (byte) - código da ocorrência.

* * SUB-ROTINA IDENTIFICA$REG

```
IDENTIFICA$REG: proc(ENDER$REG$LIDO)result(ENDER$REG$LIDO);
```

Descrição

Faz identificação do registro lido no buffer de entrada com as especificações do programa fonte.
A identificação é feita percorrendo a lista de registros (lista de descritores de registro), confrontando a descrição do registro com o conteúdo do buffer.
Se o registro lido não for identificado é chamada a rotina PARADA, que interrompe a execução fornecendo opções ao operador.

Parâmetros:

ENDERREGLIDO-endereço do nó descritor relativo ao registro identificado

Algoritmo:

```
IDENTIFICA$REG:proc(ENDER$REG$LIDO)result(ENDER$REG$LIDO);
%PARÂMETRO:ENDER$REG$LIDO-ENDEREÇO DO NÓ DESCRITOR DO REGISTROLIDO;
    ACABOU$REG=FALSE;
ACESSA BUFFER;
ACESSA 1º NÓ DA LISTA DE REGISTRO; %PERCORRE LISTA DE REGISTROS
```

```
do while not ACABOU$REG;
   VERIFICA IDENTIDADE;
   if REGISTRO IDENTIFICADO
      then do;
             ENDER$REG$LIDO=ENDEREÇO DO NÓ DESCRITOR; return;
           end;
      else if SUCESSOR=Ø;
           then ACABOU$REG=TRUE;
           else ACESSA NÓ SUCESSOR;
end while;
call PARADA(COD$ERRO); %NÃO IDENTIFICADO
```

Algoritmo: versão mais refinada

```
IDENTIFICA$REG:proc(ENDER$REG$LIDO)result(ENDER$REG$LIDO);
%VERSÃO MAIS REFINADA EM PLTI
declare
      1 no based PT$AUX,    %VARIÁVEIS EXTERNAS AO BLOCO
      2 POSIÇÃO address,    %PT$REG-APONTADOR NÓ REG
      2 NOT2 byte,          %ENDER$BUFFER-ENDEREÇO DO BUFFER
      2 TIPO byte,          %PT$LISTA$REG-CABEÇA LISTA REG
      2 CAR byte,           %PRÓXIMO$REG-ENDEREÇO DO PRÓXIMO NÓ REG
declare
      FLAG based PT$AUX byte;
      OPERAÇÃO byte; ENDER$REG$LIDO address;
      BUFFER(512)byte based PT$BUFFER;
      ACABOU$REG byte initial Ø; %FALSE
      (VALOR1,VALOR2,VALOR3)byte;
PT$BUFFER=ENDER$BUFFER, %ACESSA BUFFER
PT$REG=PT$LISTA$REG;    %ACESSA NÓ
PT$AUX=PT$REG+12;       %ACESSA CADEIA DE IDENTIFICAÇÃO
do while not ACABOU$REG;
   %VERIFICA IDENTIFICAÇÃO;
   OPERAÇÃO=#FD; VALOR3=TRUE;
   do while OPERAÇÃO <>#FF;
      VALOR2=TRUE;
      do while FLAG <>#FE and FLAG <>#FD and FLAG <>#FF;
         VALOR1=(BUFFER(POSIÇÃO)xor car)and TIPO)=Ø xor NOT2);
         VALOR2=VALOR2 and VALOR1; PT$AUX=PT$AUX+5;
      end while;
```

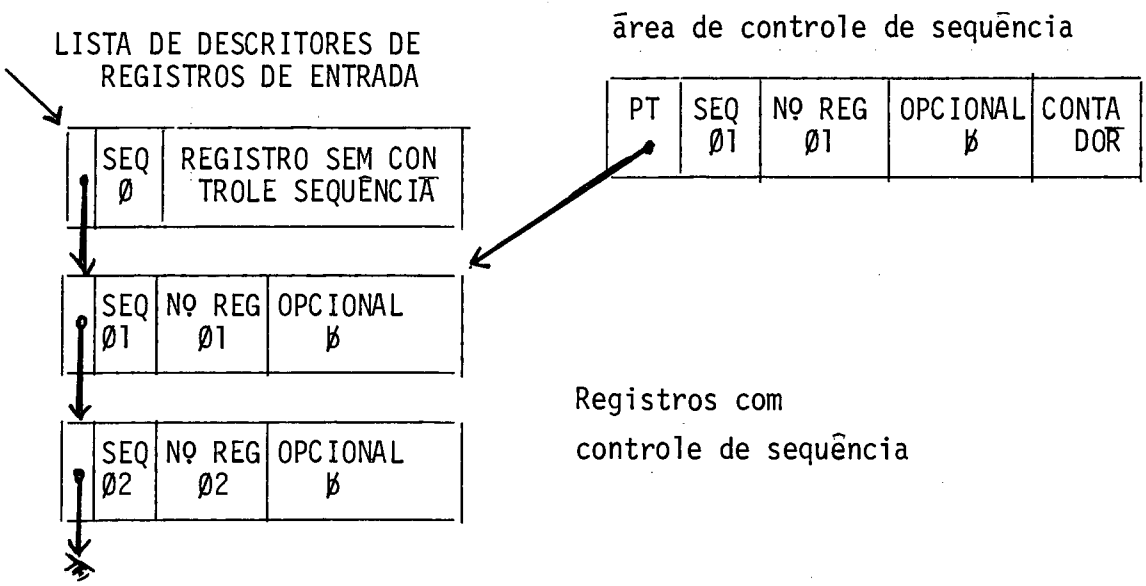
```
      if OPERAÇÃO= ¬FD then VALOR3=VALOR3 and VALOR2;
                       else VALOR3=VALOR3 or VALOR2;
      OPERAÇÃO=FLAG; PT$AUX=PT$AUX+1;
   end while;
    if VALOR3=TRUE then do; ENDER$REG$LIDO=PT$REG; return; end;
                  else if PRÓXIMO$REG=Ø then ACABOU$REG=TRUE;
                                         else do;
                                              PT$REG=PRÓXIMO$REG;
                                              PT$AUX=PT$REG+12;
                                             end;
end while;
call PARADA(COD$ERRO);
```

```
* *  ROTINA VERIFICA$SEQUÊNCIA (ENDER$REG$LIDO);
     VERIFICA$SEQUÊNCIA:proc(ENDER$REG$LIDO)
```

Descrição

A rotina testa a sequência de tipos de registro em grupos sequenciais quando esta opção de processamento é especificada pelo usuário no programa.
É utilizada uma área auxiliar para o controle de sequência. Esta área é gerada e inicializada na fase de compilação, para cada arquivo, se existe controle de sequência; ela indica qual o registro esperado de acordo com o controle sequencial.

Elementos da área de controle:



A área auxiliar é inicializada com os elementos do primeiro registro esperado no grupo sequencial. Feita a verificação esta área é atualizada com elementos do próximo nó descritor.
Quando há controle de sequência, a lista de descritores funciona como uma lista circular.

Algoritmo

```
VERIFICA$SEQUÊNCIA:proc(ENDER$REG$LIDO);
     ACESSA NÓ DESCRITOR DO REGISTRO LIDO;
     if Nº SEQUÊNCIA ESTÁ ESPECIFICADO NO DESCRITOR %HÁ CONTROLE DE SEQUÊNCIA
        then do;
             ACABOU$SEQ=FALSE;
             do while not ACABOU$SEQ;
              if Nº SEQUÊNCIA DO DESCRITOR=Nº SEQUÊNCIA ESPERADO %ÁREA AUXI-
                                                                  %LIAR
                 then do;
                      if Nº DE REGISTROS ESPERADO > 1
                         then  DECREMENTA DE 1 UNIDADE;
                         else
                            if Nº DE REGISTROS ESPERADO É DETERMINADO
                               then do;      %ATUALIZA ÁREA AUXILIAR
                                    ACESSA PRÓXIMA NÓ DESCRITOR DE REGIS-
                                    TROS;
                                    MOVE Nº DE SEQUÊNCIA, Nº DE REGISTROS,
                                    OPÇÃO, PARA ÁREA AUXILIAR;
                                    MOVE ZEROS PARA CONTADOR NA ÁREA AUXI
                                    LIAR;
                                  end;
                               else INCREMENTA DE 1 CONTADOR ÁREA AUXILIAR;
                      ACABOU$SEQ=TRUE;
                   end;
                else if REGISTRO ESPERADO NÃO É OPCIONAL and Nº DE REGIS-
                        TROS ESPERADO É DETERMINADO
                        then call PARADA(COD$ERRO);
                        else if Nº DE REGISTROS ESPERADO NÃO É DETERMINADO
                                and CONTADOR ÁREA AUXILIAR=Ø(NÃO HOUVE  O
                                CORRÊNCIA) and SE REGISTRO NÃO É OPCIONAL
                                then call PARADA(COD$ERRO);
                                else do;   %ATUALIZA ÁREA AUXILIAR
```

```
                                    ACESSA PRÓXIMO NÓ DESCRITOR DE REGIS
                                    TROS;
                                    MOVE Nº DE SEQUÊNCIA, Nº DE REGISTROS,
                                    OPÇÃO, PARA ÁREA AUXILIAR;
                                    MOVE ZEROS PARA CONTADOR NA ÁREA AUXI
                                    LIAR;
                                  end;
            end while;
        end;
  end VERIFICA$SEQUÊNCIA;
```

```
* *  SUB-ROTINA INICIALIZA$ARQ
     INICIALIZA$ARQ:proc(TIPO$ARQUIVO);
```

Descrição

No caso de processamento com múltiplos arquivos de entrada utilizando a técnica de campos de combinação (campos MATCH), cada arquivo deverá sofrer uma inicialização.
Um arquivo primário estará inicializado quando, após n leituras, for encontrado um registro com campos de combinação especificados (M1-M9). Enquanto este tipo de registro não for encontrado , o registro primário é selecionado e processado.
Quando for encontrado um registro com campo de combinação, o registro primário, agora inicializado, não é processado imediatamente. Ao invés disto, o programa interpretador executa a inicialização do primeiro arquivo secundário, da mesma maneira. Este processo continua até a inicialização de todos os arquivos, ou seja, um registro contendo campo de combinação foi lido em cada arquivo.

Parâmetros:

```
     TIPO$ARQUIVO=Ø - inicializa arquivo primário
                  =1 - inicializa arquivo secundário
```

Algoritmo

```
      INICIALIZA$ARQ:proc(TIPO$ARQUIVO);
if TIPO$ARQUIVO=PRIMÁRIO   %TRATA INICIALIZAÇÃO ARQ. PRIMÁRIO
    then do;
         X=IO(ENDER$ARQ$LER,ENDE$BUFFER$LER,READ,Ø);
         if X=CÓDIGO$ERRO then call PARADA(COD$ERRO);
         if X<> EOF   %NÃO É FIM DE ARQUIVO
            then do;
                 %VERIFICA IDENTIDADE DO REGISTRO LIDO
                 call IDENTIFICA$REG(ENDER$REG$LIDO);
                 %VERIFICA SEQUÊNCIA TIPOS DE REGISTRO
                 call VERIFICA$SEQUÊNCIA(ENDER$REG$LIDO);
                 if REGISTRO POSSUE CAMPOS   %VER DESCRITOR MATCH;
                    then do;
                         %MOVE CAMPOS MATCH DO REGISTRO LIDO PARA ÁREA
                         %MATCH CORRESPONDENTE
                         call MOVE$MATCH(ENDER$REG$LIDO);
                         ARQ$PRIMÁRIO$INICIALIZADO=TRUE;
                         ACESSA 1º ARQ. SECUNDÁRIO-DESCRITOR;
                         ENDER$ARQ$LER=@NÓ$ARQUIVO;
                         ENDER$BUFFER$LER=@BUFFER;
                      end;
                    else ABORT PROCESSA; %VOLTA A ROTINA INTERPRETA E  PRO
                                         %CESSA REGISTRO DO ARQ. PRIMÁRIO
               end;
            else do; ARQ$PRIMÁRIO$INICIALIZADO=TRUE;
                 MARCA NÓ DESCRITOR DO ARQUIVO ESTADO=EOF;
               end;
       end INICIALIZAÇÃO DO ARQUIVO PRIMÁRIO;
%TRATA INICIALIZAÇÃO ARQ. SECUNDÁRIO
do while not INICIALIZOU$TODOS;
   X=IO(ENDER$ARQ$LER,ENDER$BUFFER$LER,READ,Ø);
   %ENDEREÇOS SALVOS NO ACESSO
   if X=CÓDIGO$ERRO then call PARADA(COD$ERRO);
   if X<> EOF
      then do;
           call IDENTIFICA$REG(ENDER$REG$LIDO);
           call VERIFICA$SEQUÊNCIA(ENDER$REG$LIDO);
           if REGISTRO POSSUE CAMPOS DE COMBINAÇÃO  %VER DESCRITOR
```

```
                then call MOVE$MATCH(ENDER$REG$LIDO);
                else abort PROCESSA; %VOLTA AO PROGR. INTERPRETADOR E PROCES-
                                     %SA REGISTRO
            end;
        else do; MARCA NÓ DESCRITOR DO ARQUIVO ESTADO=EOF;
                call TESTA$FIM(FIM$JOB);
                if FIM$JOB
                   then abort FINAL; %VOLTA AO INTERPRETADOR P/FINALIZAÇÃO
            end;
    if SUCESSOR=Ø   %PRÓXIMO ARQUIVO SECUNDÁRIO(LISTA DE DESCRITORES)
        then INICIALIZOU$TODOS=TRUE;
        else do; ACESSA PRÓXIMO ARQUIVO SECUNDÁRIO; %DESCRITOR
                ENDER$ARQ$LER=@NÓ ARQUIVO;          %DESCRITOR
                ENDER$BUFFER$LER=@BUFFER;           %DESCRITOR
            end;
end  while;
abort SELECIONA; %VOLTA AO PROGR. INTERPRETADOR P/SELEÇÃO
end INICIALIZA;
```

```
* *  SUB-ROTINA QUEBRA$CONTROLE

     QUEBRA$CONTROLE:proc(HOUVE$QUEBRA,ENDER$REG$LIDO)
                     result(HOUVE$QUEBRA)
```

Descrição:

Verifica se houve quebra de controle no registro selecionado, para os campos com controle de nível especificado.
Temos na estrutura do código, para cada registro, uma lista de campos de controle de nível. Temos também para cada nível uma área de controle de nível, que conterá sempre o último valor lido para o campo(inicializada com zeros ou brancos quando o campo for numérico ou alfanumérico, respectivamente).
A rotina percorre a lista nível, comparando para cada nó, o valor associado no buffer de entrada (registro lido) com o valor contido na área de controle de nível (registro lido anterior).
Se na comparação os valores forem diferentes, houve uma quebra de controle e é ligado o indicador de nível do campo bem como os indicadores de nível inferior.
Na comparação dos campos, se o tipo for numérico é considerado a

penas a parte de digito do byte testado.

Parâmetros:

HOUVE$QUEBRA-result-sim ou não.

ENDERREGLIDO-endereço do nõ descritor do registro lido

Algoritmo

```
QUEBRA$CONTROLE:proc(HOUVE$QUEBRA,ENDER$REG$LIDO)result(HOUVE$QUEBRA);
  %PARÂMETROS:ENDER$REG$LIDO-ENDEREÇO DO NÕ DESCRITOR DO REGISTRO LIDO;
  %           HOUVE$QUEBRA: SIM OU NÃO;
  ACABOU$NÍVEL,HOUVE$QUEBRA=FALSE;
  ACESSA BUFFER;
  if LISTA NÍVEL VAZIA
     then return;%NÃO HÁ CONTROLE DE QUEBRA
  ACESSA 1º NO DA LISTA NÍVEL; %PERCORRE LISTA NÍVEL
  do while not ACABOU$NÍVEL;
     if SUCESSOR=Ø then  ACABOU$NÍVEL=TRUE;
     if CONTEÚDO DA ÁREA NÍVEL=CONTEÚDO DO BUFFER
        then do;%NA COMPARAÇÃO SE CAMPO NUMÉRICO COMPARA
                %SOMENTE A PARTE DE DIGITO DE CADA BYTE;
                %HOUVE QUEBRA
                LIGA INDICADOR DE NÍVEL ESPECIFICADO E INDICADORES DE NÍVEL
                INFERIOR;
                MOVE CAMPO DO BUFFER PARA ÁREA NÍVEL;
                HOUVE QUEBRA=TRUE;
             end;
     ACESSA PRÕXIMO NÓ DA LISTA NÍVEL;
  end while;
end QUEBRA$CONTROLE;
```

* * SUB-ROTINA TESTA$IND

TESTA$IND:proc(OK,ENDER$CADEIA)result(OK);

Descrição:

Tendo como base o vetor de situação de indicadores SITUAÇÃO$IND (132)Byte, a rotina verifica se as condições previstas numa cadeia descritora de indicadores, estão satisfeitas.
Ela é utilizada na verificação dos indicadores que condicionam operações (formulário de cálculos), saídas de registros e saídas de campos (formulários de saídas).

Parâmetros:

ENDER$CADEIA - endereço de cadeia de indicadores a ser testada

OK - condições satisfeitas ou não

Algoritmo:

```
TESTA$IND:proc(OK,ENDER$CADEIA)result(OK);
%PARÂMETROS:OK-CONDIÇÕES SATISFEITAS OU NÃO
%           ENDER$CADEIA-ENDEREÇO DA CADEIA DE INDICADORES A SER TESTADA
OK=FALSE;
ACESSA CADEIA DE INDICADORES;
if NÃO HÁ INDICADORES DE CONDIÇÃO
   then do; OK=TRUE; return;
   end;
ACABOU$CADEIA=FALSE;
do while not ACABOU$CADEIA;
   VERIFICA CONDIÇÕES DE CADA INDICADOR CONSIDERANDO RELAÇÕES AND E OR;
   if FLAG ENCONTRADO
      then ACABOU$CADEIA=TRUE;
end while;
if CONDIÇÕES SATISFEITAS
   then OK=TRUE;
end TESTA$IND;
```

Algoritmo:

```
TESTA$IND:proc(OK,ENDER$CADEIA)result(OK);
   %ALGORITMO COMO MAIOR REFINAMENTO EM PLTI
      Declare
         1 NO based PT;
         2 NOT2 byte;
         2 IND byte;
      Declare FLAG based PT byte;
      Declare (VALOR1,VALOR2,VALOR3,OK,OPERAÇÃO)byte;
      Declare ENDER$CADEIA address;
   OK=FALSE;OPERAÇÃO=#FD;VALOR3=TRUE;
   PT=ENDER$CADEIA;
   if NOT2='#' %NÃO HÁ INDICADORES
      then do; OK=TRUE;return;end;
   do while OPERAÇÃO <>'#';
      VALOR2=TRUE;
      Do while FLAG <>#FE and FLAG <>#FD and FLAG <>'#';
         VALOR1=((SITUAÇÃO$IND(IND)xor#FF)=Ø xor NOT2);
         VALOR2=VALOR2 and VALOR1;
         PT=PT+2;
      end while;
      if OPERAÇÃO=#FD then VALOR3=VALOR3 and VALOR2;
                      else VALOR3=VALOR3 or VALOR2;
      OPERAÇÃO=FLAG;
      PT=PT+1;
   end while;
   if VALOR3=TRUE;
      then OK=TRUE;
   end TESTA$IND;
```

* * SUB-ROTINA EXEC$SAÍDA

```
EXEC$SAÍDA.proc(TEMPO$SAÍDA)
```

Descrição

Providencia as saídas em tempo de cabeçalho/detalhe, total ou excessão, conforme a especificação do parâmetro TEMPO$SAÍDA.

Percorre a lista de arquivos de saída, verifica para cada arquivo quais os registros que atendem as condições impostas pelos indicadores de saída através da rotina TESTA$CONDIÇÕES$SAÍDA, e se condições satisfeitas para os registros, chama a rotina MONTA$ESCREVE que fará verificações finais e encaminhará a saída.

Parâmetros: TEMPO$SAÍDA 1 - tempo de cabec/detalhe
2 - tempo de total
3 - tempo de excesso

Algoritmo

```
      EXEC$SAÍDA:proc(TEMPO$SAÍDA);
%PARÂMETROS:TEMPO$SAÍDA-TEMPO DE CABEC/DETALHE,TOTAL OU EXCESSÃO
%PERCORRE LISTA DE ARQUIVOS DE SAÍDA
 ACESSA 1º NÓ DA LISTA;ACABOU$ARQ$SAÍDA=FALSE; TIPO$SAÍDA=Ø;
 do while not ACABOU$ARQ$SAÍDA;
    if SUCESSOR=Ø
       then ACABOU$ARQ$SAÍDA=TRUE; %ÚLTIMO NÓ
    %SELECIONA LISTA DE REGISTROS CONFORME VARIÁVEL TEMPO$SAÍDA
    do case TEMPO$SAÍDA;
     %CABEC/DETALHE
      ACESSA 1º NÓ DA LISTA DE REGISTROS CABEÇALHO  DETALHE;
     %TOTAL
      ACESSA 1º NÓ DA LISTA DE REGISTROS DE TOTAL;
     %EXCESSÃO
      ACESSA 1º NÓ DA LISTA DE REGISTROS DE EXCESSÃO;
    end case;
    %PERCORRE LISTA DE REGISTROS SELECIONADA
     ACABOU$REG$SAÍDA=FALSE;
    do while not ACABOU$REG$SAÍDA;
     if SUCESSOR=Ø
        then ACABOU$REG$SAÍDA=TRUE;
     %TESTA CONDIÇÕES PARA SAÍDA
     call TESTA$CONDIÇÕES$SAÍDA(SATISFAZ,ENDER$SPACE$SKIP,ENDER$
          NÓ$REG);
     if SATISFAZ
        then call MONTA$ESCREVE(ENDER$SPACE$SKIP,ENDER$NÓ$CAMPO,
                  TIPO$SAÍDA); %TIPO$SAÍDA=Ø
     ACESSA PRÓXIMO NÓ DA LISTA DE REG;
```

```
     end while;
  ACESSA PRÓXIMO NÓ DA LISTA DE ARQUIVOS;
end while;
end EXEC$SAÍDA;
```

```
* *  SUB-ROTINA EXEC$SAÍDA$OV
     EXEC$SAÍDA$OV:proc(AUTOMÁTICO)result(AUTOMÁTICO);
```

Descrição

Providencia a execução das saídas condicionadas por overflow (arquivo impressora).
A rotina foi desenvolvida prevendo-se apenas um arquivo de saída ligado a impressora, dentro das condições de utilização da linguagem.
A rotina, inicialmente procura na lista de arquivos de saída (lista descritora), o arquivo de saída (impressora) que possue especificação de uso do indicador de overflow OV.
Se não encontrou nenhum arquivo com a especificação 'OV', devolve ao programa chamador através da variável AUTOMÁTICO, a indicação de que o controle de overflow é automático.
Se encontrou, percorre a lista de registros de saída em tempo de total, depois a lista de registros de saída em tempo de cabec/detalhe, verificando os nós que possuem indicador de overflow especificado e para estes testando e encaminhando para a saída.

Parâmetros: AUTOMÁTICO - indicador de overflow automático ou não.

Algoritmo:

```
EXEC$SAÍDA$OV:proc(AUTOMÁTICO)result AUTOMÁTICO;
   %PARÂMETRO:AUTOMÁTICO-OVERFLOW AUTOMÁTICO OU NÃO
   %PERCORRE LISTA DE ARQUIVOS DE SAÍDA ATÉ ENCONTRAR NÓ COM
   %ESPECIFICAÇÃO OV(IMPRESSORA)
   ACESSA 1º NÓ DA LISTA
   ACABOU$OV,ACHOU$OV,AUTOMÁTICO=FALSE;
   do while not ACABOU$OV;
      if NÓ POSSUI ESPECIFICAÇÃO OV
         then ACHOU$OV,ACABOU$OV=TRUE;
         else do;
            if SUCESSOR=Ø then ACABOU$OV=TRUE;
               else ACESSA NÓ SUCESSOR;
            end;
   end while;
   if not ACHOU$OV
      then do;AUTOMÁTICO=TRUE;return;end;%OVERFLOW AUTOMÁTICO
      else do;
         %PERCORRE LISTA DE REG DE TOTAL, DEPOIS DE CABEC/DETALHE
         %TESTANDO E ENCAMINHANDO PARA SAÍDA
         do I=1 to 2;
            if I=1 then ACESSA 1º NÓ DA LISTA DE TOTAL;
                   else ACESSA 1º NÓ DA LISTA DE CABEC/DETALHE;
            ACABOU$REG$OV=FALSE;
            do while not ACABOU$REG$OV;
               if SUCESSOR=Ø then ACABOU$REG$OV=TRUE;
               if REGISTRO POSSUE INDICADOR OV ESPECIFICADO
                  then do:
                     %TESTA CONDIÇÕES SAÍDA
                     call TESTA$CONDIÇÕES$SAÍDA(SATISFAZ,ENDER$SPACE$KIP,EN-
                     DER$NÓ$REG);
                     if SATISFAZ
                        then do:
                           TIPO$SAÍDA=I;
                           call MONTA$ESCREVE(ENDER$SPACE$SKIP,ENDER$NÓCAMPO,
                           TIPO$SAÍDA);
                            end;
                     end;
```

```
            ACESSA NÓ SUCESSOR DA LISTA DE REGISTROS;
          end while;
        end do;
      end;
    DESLIGA INDICADOR DE OVERFLOW
end EXEC$SAÍDA$OV;
```

* * SUB-ROTINA EXEC$SAÍDA$1P

```
     EXEC$SAÍDA$1P:proc;
```

Descrição

Providencia a saída de linhas de cabeçalho e detalhe condicionadas pelo indicador 1P(1ª página - arquivo impressora).
A sub-rotina foi desenvolvida prevendo-se a utilização de um arquivo ligado a impressora, dentro das condições da linguagem.
Inicialmente procura o arquivo impressora na lista de descritores de arquivos de saída. Se não encontrado, o controle volta ao interpretador. Se encontrado, percorre a lista de registros cabec/detalhe vinculada ao arquivo, procurando registros condicionados por indicador 1P, testando condições de escrita para os mesmos e providenciando a saída.

Parâmetros: não tem.

Sub-rotinas chamadas:

MONTA$ESCREVE - monta buffer e providencia escrita

Algoritmo

```
     EXEC$SAÍDA$1P:proc;
%PERCORRE LISTA DOS ARQUIVOS DE SAÍDA ATÉ ARQUIVO IMPRESSORA
ACESSA 1º NÓ DA LISTA;
ACABOU$ARQ,ACHOU$ARQ=FALSE;
do while not ACABOU$ARQ;
 if NÓ É ARQUIVO IMPRESSORA
    then do;
          ACHOU$ARQ,ACABOU$ARQ=TRUE;
        end;
```

```
    else do;
          if SUCESSOR=Ø
             then ACABOU$ARQ=TRUE;
             else ACESSA PRÓXIMO NÓ;
        end;
end while;
if not ACHOU$ARQ
   then return; %NÃO EXISTE ARQUIVO NA IMPRESSORA
%PERCORRE LISTA DE REGISTROS CABEC/DETALHE DO ARQUIVO PROCURANDO POR REGIS-
%TROS CONDICIONADOS POR 1P
ACABOU$CD=FALSE; ACESSA 1º NÓ DA LISTA CABEC/DETALHE;
do while not ACABOU$CD;
 if SUCESSOR=Ø
    then ACABOU$CD=TRUE;
 if REGISTRO POSSUE INDICADOR 1P ESPECIFICADO
    then do;
          %TESTA CONDIÇÕES PARA SAÍDA
          call TESTA$CONDIÇÕES$SAÍDA(SATISFAZ,ENDER$SPACE$SKIP,ENDER$NÓ$
               REG);
          if SATISFAZ
             then call MONTA$ESCREVE(ENDER$SPACE$SKIP,ENDER$NÓ$CAMPO,TIPO$
                  SAÍDA);                          %TIPO$SAÍDA=Ø
        end;
 ACESSA PRÓXIMO NÓ DA LISTA;
end while;
end EXEC$SAÍDA$1P;
```

```
* *  SUB-ROTINA MONTA$ESCREVE

     MONTA$ESCREVE:proc(ENDER$SPACE$SKIP,ENDER$NÓ$CAMPO,TIPO$SAÍDA);
```

Descrição

Satisfeitas as condições para a escrita de um registro, a rotina monta na área de buffer do arquivo em tratamento, o conteúdo dos campos relativos ao registro e providencia a escrita.
Condições na montagem:

a) Campos condicionados por indicadores somente serão montados se condições satisfeitas.

b) Se não especificado no descritor de campos o código de edição nem máscara, não há necessidade de edição, bastando movimentação para a área de Buffer.
c) Se especificado código de edição ou máscara de edição, faz edição e movimenta para a área de Buffer.
d) Para campos com especificação BRANCO/AFTER, são movidos brancos (se campo alfanumérico) ou zeros (se campo numérico), na área de dados respectiva, após montagem do Buffer de saída.

Condições na escrita:

a) Se arquivo em tratamento é impressora, faz controle de impressão e overflow:
   - trata espacejamento e salto;
   - trata condições de overflow, ligando indicador de overflow se necessário;
   - move caráter de controle da impressão para posição reservada do Buffer de saída.
b) Rotina de IO faz saída; havendo erro é interrompido o processamento, com opções ao operador (rotina PARADA).
c) Após a escrita é limpa a área de buffer de saída.

Parâmetros:

ENDER$SPACE$SKIP - endereço dos campos SPACE/SKIP (espacejamento e salto) relativos ao registro em escrita.

ENDER$NÓ$CAMPO - endereço do 1º nó da lista descritora de campos de saída.

TIPO$SAÍDA - utilização no caso de arquivo impressora;
TIPO$SAÍDA Ø - saída em tempo cabec/detalhe total ou excessão.
1 - saída total em tempo de overflow.
2 - saída cabec/detalhe em tempo de overflow.

Algoritmo

```
MONTA$ESCREVE:proc(ENDER$SPACE$SKIP,ENDER$NÓ$CAMPO,TIPO$SAÍDA);
%PARÂMETROS:ENDER$SPACE$SKIP-ENDEREÇO DOS CAMPOS SPACE/SKIP RELATIVO A LINHA
%           ENDER$NÓ$CAMPO-ENDEREÇO DO 1º NÓ DA LISTA DE CAMPOS
%           TIPO$SAÍDA-UTILIZAÇÃO NO CASO DE IMPRESSORA
%                 TIPO$SAÍDA  Ø-SAÍDA CABEC/DETALHE,TOTAL,EXCESSÃO
%                             1-SAÍDA TOTAL CONDICIONADA P/OVERFLOW
%                             2-SAÍDA CAB/DETALHE CONDICIONADA POR OVERFLOW
ACABOU$CAMPOS=FALSE;
ACESSA ÁREA BUFFER;
ACESSA ÁREA SPACE/SKIP;
%VERIFICA E PREPARA CAMPOS DE SAÍDA
ACESSA 1º NÓ DA LISTA DE CAMPOS;
do while not ACABOU$CAMPOS;
   ENDER$CADEIA=ENDEREÇO DA CADEIA DE INDICADORES;
   call TESTA$IND(ENDER$CADEIA,OK);
   if OK     %NÃO HÁ INDICADORES DE CAMPO, OU HÁ INDICADORES C/CONDIÇÕES  SA
             %TISFEITAS
      then do;
           %VERIFICA CONDIÇÕES P/EDIÇÃO OU NÃO, MONTA BUFFER DE SAÍDA, VERI
           %FICA BLANK-AFTER
           ACESSA ÁREA DE DADOS RELATIVA AO CAMPO;
           if NÃO ESPECIFICADO CÓDIGO DE EDIÇÃO NEM MÁSCARA
              then MOVE CONTEÚDO ÁREA DADOS PARA BUFFER;
              else
                 if TEM CÓDIGO DE EDIÇÃO ESPECIFICADO
                    then FAZ EDIÇÃO C/CÓDIGO COLOCANDO RESULTADO NO BUFFER;
                    else if TEM MÁSCARA ESPECIFICADA
                            then do;
                                 ACESSA MÁSCARA;
                                 FAZ EDIÇÃO C/RESULTADO NO BUFFER;
                                end;
           if BRANCO-AFTER ESPECIFICADO
              then if TIPO DE CAMPO NUMÉRICO
                      then MOVE ZEROS P/ÁREA DADOS RELATIVA AO CAMPO;
                      else MOVE BRANCOS P/ÁREA DADOS RELATIVA AO CAMPO;
        end;
   ACESSA NÓ SUCESSOR NA LISTA DE CAMPOS;
   if FLAG ENCONTRADO
```

```
      then ACABOU$CAMPOS=TRUE;
end while;
%ESCREVE
if ARQUIVO DO TIPO IMPRESSORA
    then do;
         %TRATA DE CONTROLE DE IMPRESSÃO E OVERFLOW
         ATRAVÉS ESPECIFICAÇÕES FORNECIDAS TRATA:
         -ESPACEJAMENTO E SALTO
         -CONDIÇÕES DE OVERFLOW, LIGANDO INDICADOR DE OVERLOW SE NECESSÁRIO
         -MOVE CARATER DE CONTROLE DA IMPRESSÃO PARA POSIÇÃO RESERVADA  DO
          BUFFER DE SAÍDA;
       end;
X=IO(ENDER$ARQ,ENDER$BUFFER,OPERAÇÃO,Ø);
%OPERAÇÃO:WRITE P/ARQUIVO SAÍDA
%OPERAÇÃO:REWRITE P/ARQUIVO ATUALIZAÇÃO
if X=COD$ERRO
   then call PARADA(DISPLAY$ERRO);
%LIMPA BUFFER
MOVE BRANCOS PARA ÁREA DE BUFFER DO ARQUIVO;
end MONTA$ESCREVE;
```

```
* *  SUB-ROTINA IO
     IO:proc(ENDER$ARQ,ENDER$BUFFER,OPERAÇÃO,N$SEEK);
```

Descrição

É uma rotina do tipo função que executa as operações de entrada e saída especificadas pelo parâmetro OPERAÇÃO, tendo como base o endereço do descritor do arquivo e endereço do buffer correspondente.

Resultados:

IO=Ø - tudo OK
IO=Ø1 - fim de arquivo
IO=Ø2 - periférico inexistente
IO=Ø3 - arquivo protegido
IO=Ø4 - erro
IO=Ø5 - erro

Parâmetros:

OPERAÇÃO:

OPERAÇÃO=1 - READ - lê um registro
OPERAÇÃO=2 - WRITE - escreve um registro
OPERAÇÃO=3 - REWIND - volta ao início do arquivo
OPERAÇÃO=4 - REWRITE - reescreve o último registro
OPERAÇÃO=5 - END FILE - marca fim de arquivo
OPERAÇÃO=6 - SEEK - posiciona em determinado registro

ENDER$ARQ - endereço do nó descritor de arquivo
ENDER$BUFFER - endereço do Buffer relativo ao arquivo
N$SEEK - nº do registro para a operação SEEK - igual a zero quando não for operação SEEK

* * SUB-ROTINA TESTA$PARADA

```
TESTA$PARADA:proc;
```

Descrição

Verifica indicadores de parada no vetor SITUAÇÃO$IND, que fornece a situação dos indicadores RPG, e mostra através da Sub-rotina PARADA o(s) indicador(es) ligado(s).

* * SUB-ROTINA TESTAFIMJOB

```
TESTA$FIM$JOB:proc(FIM$JOB) result(FIM$JOB);
```

Descrição

Verifica condições de fim de processamento. No caso de processamento com múltiplos arquivos (o processamento possui ao menos um arquivo principal e um arquivo secundário), serão necessárias as seguintes condições para final de processamento:

a) Todos os arquivos de entrada com especificação 'E' na coluna 17 do formulário de descrição de arquivos, deverão estar em estado de fim de arquivos (estado é um campo do descritor de arquivo que recebe uma marca de fim de arquivo durante o processo de interpretação, quando o evento FIM DE ARQUIVO ocor-

re).

b) Se não houver especificação 'E' para os arquivos, então todos os arquivos deverão estar em estado de fim de arquivo para haver condições de fim de processamento.

Parâmetros

FIM$JOB(byte)-sim ou não.

Algoritmo

```
     TESTA$FIM$JOB:proc(FIM$JOB)result(FIM$JOB);
ACABOU$SEQ=FALSE; TEM$E=FALSE; FIM$JOB=TRUE;
ACESSA 1º NÓ DA LISTA DE ARQUIVOS;
do while not ACABOU$ARQ; %PERCORRE A LISTA VERIFICANDO NÓS COM
                         %ESPECIFIC. 'E'
 if ARQUIVO POSSUI ESPECIFIC. 'E'
    then do;
          TEM$E=TRUE;
          if ESTADO <> EOF
             then do;
                   FIM$JOB=FALSE;
                   return;
                  end;
         end;
 if SUCESSOR=Ø
    then ACABOU$ARQ=TRUE;
    else ACESSA PRÓXIMO NÓ DA LISTA;
end while;
if not TEM$E   %NÃO HÁ ARQUIVOS C/ESPECIFIC. 'E'
    then do;   %PERCORRE A LISTA VERIFICANDO SE TODOS OS   ARQUI
               %VOS ESTÃO 'EOF"
          ACABOU$ARQ=FALSE;
          ACESSA 1º NÓ DA LISTA DE ARQUIVOS ENTRADA;
          do while not ACABOU$ARQ;
           if ESTADO <> EOF
              then do;
                    FIM$JOB=FALSE;
                    return;
                   end;
           if SUCESSOR=Ø
```

```
            then ACABOU$ARQ=TRUE;
            else ACESSA PRÓXIMO NÓ;
        end while;
      end;
end TESTA$FIM$JOB;
```

* * SUB-ROTINA TESTA$CONDIÇÕES$SAÍDA

```
TESTA$CONDIÇÕES$SAÍDA:proc(SATISFAZ,ENDER$SPACE$SKIP,ENDER$IND$REG)
                      result(SATISFAZ,ENDER$SPACE$SKIP);
```

Descrição

Verifica se condições para a escrita de um registro, estão satisfeitas.
As condições para a escrita são especificadas através de cadeias de indicadores de saída. No caso de registros de impressão (arq. impressora) poderá haver, em função das necessidades do programa, diferentes especificações de espacejamento e salto, cada uma delas condicionadas por uma cadeia de indicadores diferente (utilização da relação OR).
O algoritmo recebe o endereço do descritor do registro em exame e devolve na variável SATISFAZ a indicação de condições satisfeitas ou não. No caso de registros linhas de impressão, devolve também as especificações de espacejamento e salto ligadas a cadeia de indicadores testada e aceita.

Parâmetros:

SATISFAZ - sim ou não (result)
ENDER$SPACE$SKIP - endereço das especificações de espacejamento e salto (result)
ENDER$NÓ$REG - endereço do descritor de registro, em exame.

Algoritmo

```
TESTA$CONDIÇÕES$SAÍDA:proc(SATISFAZ,ENDER$SPACE$SKIP,ENDER$NÓ$REG)
                        result(SATISFAZ,ENDER$SPACE$SKIP)
%PARÂMETROS:SATISFAZ(BYTE):SIM OU NÃO
%           ENDER$SPACE$SKIP:ENDEREÇO DOS CAMPOS SPACE/SKIP
%           ENDER$REG:ENDEREÇO DO REGISTRO A SER VERIFICADO
%TESTA 1ª CADEIA DE INDICADORES
SATISFAZ=FALSE;
ENDER$CADEIA=ENDEREÇO DA 1ª CADEIA;
ENDER$SPACE$SKIP=ENDEREÇO CAMPOS SPACE/SKIP; %ENDER$CADEIA-4
call TESTA$IND(ENDER$CADEIA),OK);
if OK
   then SATISFAZ=TRUE;
   else do;
        ACABOU$TESTE=FALSE;
        do while not ACABOU$TESTE;
           if not FLAG
              then do;
                   ENDER$CADEIA=ENDEREÇO DA PRÓXIMA CADEIA;
                   ENDER$SPACE$SKIP=ENDER$CAMPO$SPACE$SKIP;
                   call TESTA$IND(INDER$CADEIA,OK);
                   if OK
                      then do;
                           SATISFAZ=TRUE;
                           ACABOU$TESTE=TRUE;
                          end;
                  end;
       end while;
       end;
end TESTA$CONDIÇÕES$SAÍDA;


* *  SUB-ROTINA SELECIONA$MATCH

     SELECIONA$MATCH:proc;
```

Descrição

Faz a seleção de registro a ser processado no caso de processa-

mento com múltiplos arquivos de entrada (programa com 1 arquivo primário e 1 ou mais arquivo secundários), quando houver campos de casamento (campo match) especificados para os registros. Temos para cada arquivo em processamento uma área denominada área match, que conterá os campos de um determinado registro do arquivo para os quais foram especificados indicadores match (são os campos match do registro).

O princípio básico da seleção é o seguinte:

- quando um registro de um arquivo primário casa com um registro de um arquivo secundário, ele é selecionado para processamento. Neste caso deverá ser ligado o indicador MR a ser usado em cálculos ou saídas durante o processamento.
- quando não há casamento entre campos especificados, se os registros estiverem em ordem crescente, será selecionado o registro com menor valor no campo de casamento; se os registros estiverem em ordem decrescente, será selecionado o registro com maior valor no campo de casamento.

No algoritmo a ser visto adiante, consideramos registros em ordem crescente.

A sub-rotina utiliza uma área auxiliar ÁREA$SALVA, no processo de seleção. Utiliza também uma variável lógica INDICADOR$MR que será utilizada pelo interpretador para saber se deve ligar ou desligar o indicador MR a ser utilizado para o condicionamento de cálculos e saídas. Esta variável possue valor 'true' quando há casamento entre campos e permanece neste estado até haver mudança no grupo de controle match.

Algoritmo

```
      SELECIONA$MATCH:proc;
%NA INICIALIZAÇÃO DO INTERPRETADOR TEMOS AS VARIÁVEIS:HOUVE$MATCH=FALSE-ESTA
%VARIÁVEL ASSUME VALOR TRUE
%QUANDO HÁ CASAMENTO ENTRE CAMPOS
%TIPO$REG=PRIMÁRIO-ESTA VARIÁVEL INDICARÁ DE QUE TIPO DE ARQUIVO FOI LIDO  O
%ÚLTIMO REGISTRO(PRIMÁRIO OU SECUNDÁRIO)
%PRIMEIRA$VEZ$MATCH=TRUE-INDICARÁ SE É A 1ª VEZ QUE FOI CHAMADA A ROTINA  DE
%SELEÇÃO
if not PRIMEIRA$VEZ$MATCH
       then do; call MOVE$MATCH(ENDER$REG$LIDO);
            PRIMEIRA$VEZ$MATCH=FALSE;
           end;
if not HOUVE$MATCH
       then do; %NÃO OCORREU MATCH
            DETERMINA MENOR ÁREA MATCH PARA ARQUIVOS SECUNDÁRIOS;
            call COMPARA$MATCH;
           end;
       else do; %HOUVE MATCH
            if TIPO$REG=SECUNDÁRIO   %ÚLTIMO REGISTRO LIDO É DE ARQUIVO SE-
                                     %CUNDÁRIO
               then do;
                    if ÁREA MATCH PARA ESTE ARQUIVO<>ÁREA$SALVA
                       then do;
                            DETERMINA MENOR ÁREA MATCH PARA ARQUIVOS  SE
                            CUNDÁRIOS;
                            if MENOR ÁREA MATCH=ÁREA$SALVA
                               then do;
                                    SELECIONA ESTE REGISTRO SECUNDÁRIO
                                    PARA PROCESSAMENTO;
                                    INDICADOR$MR=TRUE;
                                    TIPO$REG=SECUNDÁRIO;
                                    SALVA ENDEREÇO DO ARQUIVO PARA PRÓXI
                                    MA LEITURA E SEU BUFFER, NAS VERIÁVE
                                    IS END$ARQ$LER E END$BUFFER;
                                   end;
                               else call COMPARA$MATCH;
                            end;
```

```
            else do;
                  SELECIONA ESTE ARQUIVO SECUNDÁRIO PARA PROCESSAMENTO;
                  INDICADOR$MR=TRUE;
                  TIPO$REG=SECUNDÁRIO;
                  SALVA ENDER$ARQ$LER E ENDER$BUFFER$LER;
               end;
         end REGISTRO SECUNDÁRIO;
      else do; %ÚLTIMO REGISTRO LIDO É DE ARQUIVO PRIMÁRIO
            if ÁREA MATCH PARA ARQUIVO PRIMÁRIO=ÁREA$SALVA
               then do;
                  SELECIONA REGISTRO PRIMÁRIO;
                  INDICADOR$MR=TRUE;
                  TIPO$REG=PRIMÁRIO;
                  SALVA ENDER$ARQ$LER E ENDER$BUFFER$LER;
               end;
               else do;
                  DETERMINA MENOR ÁREA MATCH PARA ARQUIVOS SECUNDÁRIOS;
                  if MENOR ÁREA MATCH=ÁREA$SALVA
                     then do;
                        SELECIONA ESTE REGISTRO SECUNDÁRIO;
                        INDICADOR$MR=TRUE;
                        TIPO$REG=SECUNDÁRIO;
                        SALVA ENDER$ARQ$LER E ENDER$BUFFER$LER;
                     end;
                     else call COMPARA$MATCH;
               end;
         end ARQUIVO PRIMÁRIO;
   end HOUVE$MATCH;
end SELECIONA$MATCH;
```

```
* *  SUB-ROTINA COMPARA$MATCH

     COMPARA$MATCH:proc;
```

Descrição

É uma sub-rotina auxiliar da sub-rotina SELECIONA$MATCH. Faz a comparação entre as áreas match dos arquivos em processamento, com vistas à seleção de um registro.
Selecionado um registro, assinala a variável HOUVE$MATCH com true ou false, indicando se houve ou não casamento.
Esta rotina é chamada sempre que houve uma quebra no controle dos campos match.

Algoritmo

```
     COMPARA$MATCH:proc;
%ESTA SUB-ROTINA É AUXILIAR DA SELECIONA$MATCH
if ÁREA MATCH PARA ARQUIVO PRIMÁRIO=MENOR ÁREA MATCH PARA ARQUIVOS SECUNDÁRI
   OS
   then do;   %HOUVE MATCH
        MOVE ÁREA MATCH PRIMÁRIO PARA ÁREA$SALVA;
        SELECIONA REGISTRO PRIMÁRIO PARA PROCESSAMENTO;
        INDICADOR$MR=TRUE;
        HOUVE$MATCH=TRUE;
        TIPO$REG=PRIMÁRIO;   %O PRÓXIMO REGISTRO A SER LIDO SERÁ NO ARQUIVO
                             %PRIMÁRIO
        SALVA ENDEREÇO DO PRÓXIMO ARQUIVO A SER LIDO E DO BUFFER CORRESPON-
        DENTE NAS VARIÁVEIS ENDER$ARQ$LER E ENDER$BUFFER;
      end;
   else do;
        if ÁREA MATCH PARA ARQ. PRIMÁRIO < MENOR ÁREA MATCH PARA ARQ.SECUN-
           DÁRIOS
           then do;
                SELECIONA REGISTRO PRIMÁRIO PARA PROCESSAMENTO;
                INDICADOR$MR=FALSE;
                HOUVE$MATCH=FALSE;
                TIPO$REG=PRIMÁRIO;
                SALVA ENDER$ARQ$LER E ENDER$BUFFER PARA O PRÓXIMO ARQUIVO
                A SER LIDO;
              end;
```

```
            else do;
                 SELECIONA REGISTRO SECUNDÁRIO PARA PROCESSAMENTO;
                 INDICADOR$MR=FALSE;
                 HOUVE$MATCH=FALSE;
                 TIPO$REG=SECUNDÁRIO;
                 SALVA ENDER$ARQ$LER E ENDER$BUFFER PARA O PRÓXIMO ARQUIVO
                 A SER LIDO;
               end;
       end;
end COMPARA$MATCH;
```

```
* *  SUB-ROTINA VERIFICA$SEQUÊNCIA$MATCH

     VERIFICA$SEQUÊNCIA$MATCH:proc;
```

Descrição

Verifica se o último registro selecionado pela rotina SELECIONA$ MATCH está fora de sequência ou seja, verifica se o valor match do registro selecionado é maior do que o valor match do registro selecionado anteriormente (caso de ordenação crescente).
O algoritmo usa uma área auxiliar, no interpretador, para controle de sequência (inicializada no início da interpretação).

Algoritmo

```
     VERIFICA$SEQUÊNCIA$MATCH:proc;
if VALOR MATCH DO ÚLTIMO REGISTRO SELECIONADO ≥ VALOR MATCH PARA
   ÁREA AUXILIAR
   then MOVE VALOR MATCH DO ÚLTIMO REGISTRO SELECIONADO PARA ÁRE
        A AUXILIAR;
   else call PARADA(CODIGO$ERRO);
end VERIFICA$SEQUÊNCIA;
```

* * SUB-ROTINA MOVE$MATCH

MOVE$MATCH:proc(ENDER$REG$LIDO)

Descrição

Move campos MATCH do registro recém lido (no Buffer) para a área match correspondente ao arquivo lido.
No processo percorre a lista match correspondente, acessa o conteúdo dos campos no Buffer do arquivo e os movimenta para a área match.
Na movimentação salva o tamanho da área match para uso posterior na rotina de comparação e verifica também se há campos numéricos movimentados. Após o processo, se houve campos numéricos envolvidos converte a parte de zona dos bytes na área match para zero.

Parâmetros: ENDER$NÓ$REG - endereço do nó descritor na lista de registros.

Algoritmo

```
      MOVE$MATCH:proc(ENDER$REG$LIDO);
%PARÂMETRO:ENDER$REG$LIDO-ENDEREÇO DO REGISTRO LIDO(NÓ DESCRITOR)
ACABOU$MATCH,NUMÉRICO=FALSE; TAMANHO$ÁREA$MATCH=Ø   %EM BYTES
ACESSA BUFFER; %DO ARQUIVO EM PROCESSAMENTO
%PERCORRE LISTA MATCH P/MOVIMENTAÇÃO
ACESSA 1º NÓ DA LISTA MATCH;
do while not ACABOU$MATCH;
 if SUCESSOR=Ø then ACABOU$MATCH=TRUE;
  MOVE CONTEÚDO DO CAMPO APONTADO,NO BUFFER PARA ÁREA MATCH APONTADA;
  INCREMENTA TAMANHO$ÁREA$MATCH;  %VARIÁVEL UTILIZADA NA ROTINA DE COMPARAÇÃO
  if TIPO$CAMPO=NUMÉRICO
     then NUMÉRICO=TRUE;
  ACESSA NÓ SUCESSOR;
end while;
%SE UM DOS CAMPOS MOVIMENTADOS É NUMÉRICO CONVERTE AS ZONAS NA ÁREA MATCH PA-
%RA ZERO
if NUMÉRICO
   then TRANSFORMA ZONAS ÁREA MATCH PARA VALOR Ø;
end MOVE$MATCH;
```

```
* *  SUB-ROTINA EXEC$OPERAÇÃO

     EXEC$OPERAÇÃO:proc(TEMPO);
```

Descrição

Executa operações de cálculo em tempo de detalhe ou em tempo de total.
Para tanto, percorrendo a lista de cálculos de detalhe ou de total, conforme o parâmetro TEMPO, descobre qual a operação a ser executada, testa indicadores e se satisfeitas as condições chama a rotina adequada para a execução da operação.

Parâmetro - TEMPO Ø - detalhe
                   1 - total

Algoritmo:

```
EXEC$OPERAÇÃO:proc(TEMPO);
%PARÂMETRO:TEMPO-DETALHE OU TOTAL
ACABOU$CALC=FALSE;
if TEMPO=TEMPO$TOTAL   %SELECIONA LISTAS DE DESCRITORES
   then if LISTA DE CÁLCULOS DE TOTAL ESTÁ VAZIA   %LISTA DE DESCRITORES
           then return;
           else ACESSA 1º NÓ DA LISTA DE CÁLCULOS DE TOTAL;
   else if LISTA DE CÁLCULOS DE DETALHE ESTÁ VAZIA;
           then return;
           else ACESSA 1º NÓ DA LISTA DE CÁLCULOS DE DETALHE;
   do while not ACABOU$CALC;
      if SUCESSOR=Ø then ACABOU$CALC=TRUE;
      VERIFICA SE AS CONDIÇÕES PARA A EXECUÇÃO DA OPERAÇÃO ESTÃO SATISFEITAS
      (SE EXISTIR CONDIÇÕES)
      SE EM TEMPO DE TOTAL VERIFICA INDICADORES DE NÍVEL E DE CÁLCULO
      SE EM TEMPO DE DETALHE VERIFICA INDICADORES DE CÁLCULO
      USAR ROTINA TESTA$IND(OK) PARA TESTAR INDIC DE CÁLCULO;
      if CONDIÇÕES SATISFEITAS
         then do case OPERAÇÃO;
               call ADD;
               call SUB;
               call DIV;
               call MULT;
```

```
            call Z-ADD;
            call Z-SUB;
            call MVR;
            call MOVE;
            call MOVEL;
            call MLLZO(PREVISÃO);
            call MLHZO(PREVISÃO);
            call MHHZO(PREVISÃO);
            call MHLZO(PREVISÃO);
            call COMP;
            call TESTZ(PREVISÃO);
            call SETON;
            call SETOF;
            call EXSR;
            call GOTO;

            call ENDSR;

            call CHAIN;
            call READ;
            call EXCEPT;
            call SUBEX;
          end case;
      TRATA DOS INDICADORES DE RESULTADO,PARA AS OPERAÇÕES QUE OS USAM;
      ACESSA PRÓXIMA OPERAÇÃO;
    end while;
end EXEC$OPERAÇÃO;
```

OBSERVAÇÕES SOBRE AS OPERAÇÕES DE CÁLCULO

Abaixo faremos um resumo relativo aos procedimentos a serem adotados na execução das operações de cálculos.
Grande parte das operações utiliza o campo indicadores de resultado, ou seja, indicadores são ligados em função dos resultados de operações, quando especificados pelo programador.
Para tratar dos indicadores de resultado, deveremos ter uma sub-rotina que poderá ligar ou desligar os mesmos segundo as necessidades da operação.

OPERAÇÃO CHAIN

Esta operação permite, para arquivos encadeados (CHAINED), a leitura (arquivos de entrada ou atualização) ou a escrita (arquivos de saída), em acesso direto por nº relativo de registro em tempo de cálculos de detalhe ou total. Temos no descritor da operação o endereço do campo cujo conteúdo indica o nº relativo de registro, o endereço do arquivo (descritor) e indicador de resultado para uso no caso de haver problemas na operação de entrada/saída.
Na execução, temos:

1. Acessa nó descritor do arquivo
2. Se o arquivo é de saída, escreve através sub-rotina
   IO.(TRATA-SE DE GERAÇÃO DO ARQUIVO)
   X=IO(ENDER$ARQ,ENDER$BUFFER,SEEK,N$SEEK)-ACESSA
   X=IO(ENDER$ARQ,ENDER$BUFFER,WRITE,Ø)-ESCREVE
   onde: ENDER$ARQ=endereço do arquivo
   ENDER$BUFFER=endereço do buffer de saída
   Operação: SEEK - acessa registro com nº relativo N$SEEK
   WRITE - escreve

Problemas na gravação - é chamada rotina que trata indicadores e é setado indicador se especificado.

3. Se arquivo de entrada ou atualização, faz leitura:
   X=IO(ENDER$ARQ,ENDER$BUFFER,SEEK,N$SEEK)
   X=IO(ENDER$ARQ,ENDER$BUFFER,READ,Ø)
   No caso de problemas na leitura é chamada a sub-rotina que trata indicadores e é setado indicador se especificado.
   Caso a leitura seja bem sucedida faz:
   - chama sub-rotina IDENTIFICA$REG para fazer identificação do

registro lido.

- move conteúdo do registro, relativo aos campos para área de dados dos mesmos (faz isto percorrendo lista MOVE relativa ao registro lido), deixando campos disponíveis para uso.

OPERAÇÃO READ

A operação READ permite a leitura em acesso sequencial, de um registro de arquivo de demanda, em tempo de execução de cálculos (detalhe ou total). Temos no descritor da operação o endereço do arquivo e um indicador de resultado que poderá ser previsto pelo programador para a indicação de erro na leitura ou fim de arquivo.

Temos os seguintes passos na execução:

1. Acessa nó descritor do arquivo
2. Lê registro através da sub-rotina IO
   X=IO(ENDER$ARQ,ENDER$BUFFER,READ,Ø)
   Operação READ - lê sequencial
3. Problemas na leitura
   Se a execução da leitura resultou em erro é chamada a sub-rotina PARADA que se comunicará com o operador.
   Se fim de arquivo ocorreu, será ligado o indicador de resultado, se especificado.
4. Caso leitura bem sucedida, repete o mesmo procedimento previsto para a operação CHAIN.

OPERAÇÃO GO TO

Na execução ocorre o salto para a operação cujo endereço está no descritor.

OPERAÇÃO EXSR

É utilizada uma pilha para armazenamento do endereço da operação que segue a operação EXSR.

Na execução de operações ocorre:

- Armazenamento na pilha, do endereço da próxima instrução.
- Salto para o endereço constante do nó descritor.

OPERAÇÃO ENDSR

Esta operação indica fim de sub-rotina.

Na execução, ocorre: retorno a operação cujo endereço está no to

po da pilha de armazenamento de endereços de retorno.

OPERAÇÃO EXCEPT

Na execução ocorre:

- Acessa lista de arquivos de saída
- Percorre a lista e para os arquivos que possuírem especificada lista de registros de EXCESSÃO(EXCEPT)(lista≠vazia), faz:
  - Percorre lista de registro de excessão
  - Testa condição de saída
  - Providencia escrita dos registros
  - Se for arquivo na impressora, e se for atingida linha de overflow, liga indicador de overflow.

OPERAÇÃO SETON/SETOF

Na execução é chamada a sub-rotina que trata de indicadores de resultado que será acionada para ligar ou desligar indicadores constantes do nó descritor da operação.

OPERAÇÃO MOVE (a direita)

Na execução ocorre a movimentação do conteúdo de fator 2 para o campo resultado. Se o campo resultado é menor do que fator 2, ocorre o truncamento dos bytes mais a esquerda de fator 2.

OPERAÇÃO MOVEL (a esquerda)

Na execução ocorre a movimentação do conteúdo de fator 2 para as posições mais a esquerda do campo resultado. Se o campo resultado é menor do que fator 2, os bytes em excesso da direita de fator 2 não são movimentados.

OPERAÇÃO ARITMÉTICAS

As operações aritméticas deverão ser efetuadas com campos no formato decimal zonado (sinal do campo na parte de zona do byte menos significativa).

OPERAÇÃO COMP

Os campos a serem comparados devem ser ambos numéricos ou alfanuméricos. Se um campo alfanumérico é menor do que o outro ,

o menor é preenchido com brancos à direita antes da comparação.
Campos numéricos são alinhados antes da comparação pela posição do ponto decimal; campos menores são preenchidos com zeros à esquerda e à direita.
Campos alfanuméricos são comparados caráter a caráter.
Campos numéricos são comparados subtraindo-se fator 2 de fator 1 e analisando-se o resultado.
Dependendo do resultado da operação são ligados indicadores de resultados, se especificados, através da sub-rotina que trata de indicadores.

OPERAÇÃO SUBEX

Esta operação permite a execução de sub-rotinas externas. Na execução, é feita o salto para a sub-rotina externa.

CAPÍTULO VII - CONCLUSÃO

O presente trabalho indica uma alternativa para o desenvolvimento de um Compilador/Interpretador RPG, para o Terminal Inteligente.

A linguagem definida, aliada às facilidades que oferece o sistema operacional existente, permitirá um tratamento adequado no processamento de sistemas comerciais, apresentando facilidades na geração de relatórios e na manipulação de arquivos. O modelo utilizado para a linguagem é um modelo básico que poderá ser ampliado em função de novos recursos a serem implantados no terminal, ou em função de novas necessidades a serem detectadas na fase de utilização. Como elementos adicionais para a linguagem destacamos:

i) A possibilidade de uso de arquivos de endereços de registros, que permitirá especificar quais os registros a serem lidos num arquivo, e qual a ordem de leitura. A geração destes arquivos deverá ser feita através de um utilitário de SORT.

ii) O uso de tabelas de consulta.

iii) A possibilidade de processamento randômico e sequencial por chave em arquivos com organização indexada. Para tanto deverá ser preparado o software básico no terminal.

iv) A possibilidade de definição de campos do tipo byte e address (campos com 1 e 2 bytes), de forma compatível com a linguagem PLTI utilizada no terminal. Esta opção, permitirá por exemplo, criar um arquivo utilizando PLTI e listá-lo em relatório através do RPG.

v) Um editor de textos especial para o RPG, tendo-se em vista as características especiais da linguagem no tocante a codificação.

Com relação aos processos de compilação e interpretação, fazemos as seguintes considerações:

i) O código intermediário proposto, que no caso, é o nome dado a um conjunto de descritores (tabelas), organizados em estrutura de listas, é de fácil geração e permite num acesso rápido às informações em tempo de interpretação.

O seu tamanho é relativamente pequeno, uma vez que ele representa um conjunto de especificações para o processamento segundo a lógica fixa RPG. Para efeito de melhor visualização, apresentamos no apêndice "B" um exemplo de programa de atualização de arquivo, ocupando aproximadamente 1450 bytes.

ii) Estimativa de tamanho máximo de programa-como vimos no capítulo V, o programa compilador foi implementado experimentalmente em PLTI e a ocupação máxima com os módulos em overlay foi de 8.0 KB aproximadamente (envolvendo tabelas, áreas auxiliares, rotinas básicas e módulos).
Considerando que o programa interpretador, a ser implementado, não ultrapasse os 8.0 KB (tamanho quase duas vezes maior que o maior módulo de compilação), poderíamos gerar e interpretar programas em RPG de até (M-8.0) KB de código intermediário, ou seja, de até 40 KB, aproximadamente (M=área disponível no terminal para a execução).
Como o tamanho do interpretador será fator de grande importância no projeto, sugerimos que o mesmo seja implementado em Assembler. Uma alternativa que também merece estudo é a de carregamento do interpretador de acordo com as necessidades do programa a ser executado. Assim, por exemplo, se o programa não possuir processamento com múltiplos arquivos de entrada, não seriam carregadas as subrotinas correspondentes ao processo de seleção de registros de entrada.

BIBLIOGRAFIA

1. GRIES, David. Compiler Construction For Digital Computers. Wiley International Edition - 1971.

2. AHO, Alfred V., e ULMAN, Jeffrey D., Principles of Compiler Design. Addison-Wesley - 1977.

3. WIRTH, Niklaus. Algorithms + Data Structures = Programs. Prentice Hall Inc. - 1976.

4. IBM. Manuais de referência da linguagem RPG.

5. NUCLEO DE COMPUTAÇÃO DA UFRJ. Manuais do Terminal Inteligente.

APÊNDICE "A" - MÓDULOS DO PROCESSO DE COMPILACÃO

MÓDULO 1: SUBROTINA COMP$HFDL

```
          COMP$HFDL:proc;
   %INICIALIZA ÁREA DE PONTEIROS DO INÍCIO DA ESTRUTURA
   call GERAN (GERA 8 ZEROS);
   call LÊ$ESCREVE;
   %CONSISTE CARTÃO TIPO 'H'
   if BUFFER$REG(COLUNA 6) <> 'H'   %SE NÃO É CARTÃO TIPO 'H'
      then do;
            call ROT$ERRO(1,W);
            ASSUME NOME DO PROGRAMA:"JOBRPG";
          end;
      move NOME DO PROGRAMA to ÁREA RESERVADA;
   %CONSISTE CARTÃO TIPO 'F'
   call LÊ$ESCREVE;
   if BUFFER$REG(COLUNA 6) <> 'F'
      then do;
            call ROT$ERRO(2,T);  %FALTOU CARTÃO TIPO 'F'
            abort FIM$PROGRAMA;  %ENCERRA COMPILAÇÃO
          end;
   HOUVE$ARQUIVO$PRINCIPAL=FALSE;  %CHAVE PARA VERIFICAR SE HOUVE ARQUIVO
                                   %PRINCIPAL NA COMPILAÇÃO
   %ENQUANTO HOUVER REGISTRO FONTE DO TIPO 'F', FAÇA:
   do while BUFFER$REG(COLUNA 6)='F';
      %CONSISTE TAMANHO DE REGISTRO-COLUNAS 24-27
      call SCAN$NÚMERO(PROCURA POR Nº NAS COLUNAS 24-27);
      if ERRO or VALOR>TAMANHO$MÁXIMO$REG  %512 bytes
         then do; call ROT$ERRO (3,T);  %ASSUME TAMANHO=80
                  TAMANHO$REG=80;
              end;
         else TAMANHO$REG=VALOR;
      %CONSISTE PERIFÉRICO(NOME DE ROTINA I/O)-COLUNAS 40-46
      call SCAN$NOME(COLUNAS 40-46);
      if ERRO then do;
                    call ROT$ERRO(5,W);  %PERIF. INVÁLIDO
                    ASSUME PERIFÉRICO="DUMMY";
                   end;
      %CONSISTE NOME DE ARQUIVO-COLUNAS 7-12
      call SCAN$NOME(COLUNAS 7-12);
```

```
if ERRO
   then do;
         call ROT$ERRO(7,T);  %NOME INVÁLIDO
         go to OUTRO;         %IGNORA RESTO DO REGISTRO
        end;
call BUSCA$TS(NOME DO ARQUIVO DAS COLUNAS 7-12);
if ACHOU
   then do;
         call ROT$ERRO(8,T);  %NOME JÁ DEFINIDO
         go to OUTRO;         %IGNORA RESTO REGISTRO
        end;
%CONSISTE NOME EXTERNO DE ARQUIVO-COLUNAS 47-52
call SCAN$NOME(COLUNAS 47-52);
if ERRO
   then do;
         call ROT$ERRO(6,W);  %NOME INVÁLIDO
         ASSUME NOME INTERNO;
        end;
%CONSISTE TIPO DE ARQUIVO - COLUNA 15
call SCAN(PROCURA POR 'I','O' ou 'U', NA COLUNA 15)
if RESULTADO=ERRO  %I=ENTRADA, U=ATUALIZAÇÃO, O=SAÍDA
   then do;
         call ROT$ERRO(9,T);  %ENTRADA INVÁLIDA
         go to OUTRO;         %IGNORA RESTO DA ESPECIFIC.
        end;
  else TIPO$ARQ=RESULTADO;  %SALVA TIPO
%CONSISTE USO DO ARQUIVO - COLUNA 16
call SCAN(PROCURA POR 'P','S','D','C', ou 'Ø' NA COLUNA 16);
USO$ARQ=RESULTADO;  %S=SECUND, D=DEMANDA, C=CHAIN, Ø=BRANCO, P=PRINCI-
                    %PAL
if USO$ARQ=ERRO
   then do;
         call ROT$ERRO(10,W);  %ENTRADA INVÁLIDA
         if TIPO$ARQ=ENTRADA or
            TIPO$ARQ=ATUALIZAÇÃO
            then ASSUME USO$ARQ=SECUNDÁRIO, COLUNA 16='S';
            else if TIPO$ARQ=SAÍDA
                    then ASSUMO USO$ARQ=BRANCO, COLUNA 16='Ø';
```

```
          end;
    else if TIPO$ARQ=SAÍDA
              and USO$ARQ<>BRANCO
              and USO$ARQ<>CHAIN
             then do;
                  call ROT$ERRO(11,W);  %ENTRADA INVÁLIDA
                  ASSUME USO$ARQ=BRANCO, COLUNA 16='Ø';
                end;
if USO$ARQ=PRINCIPAL and
   HOUVE$ARQUIVO$PRINCIPAL
   then do;
          call ROT$ERRO(12,W);  %ARQ. PRINCIPAL JÁ EXISTIA
          ASSUME USO$ARQ=SECUNDÁRIO, COLUNA 16='S';
        end;
   else HOUVE$ARQUIVO$PRINCIPAL=TRUE;
%CONSISTE COLUNA 17-FIM DE ARQUIVO
call SCAN(PROCURA POR 'E' OU 'Ø', NA COLUNA 17);
if RESULTADO=ERRO
   then do;
          call ROT$ERRO(14,W);  %ENTRADA INVÁLIDA
          ASSUME COLUNA 17='Ø';
        end;
   else if RESULTADO=E  %'E' NA COLUNA 17
             and USO$ARQ<>PRINCIPAL and
                 USO$ARQ<>SECUNDÁRIO
                 then do;
                       call ROT$ERRO(14,W);
                       ASSUME COLUNA 17='Ø';
                     end;
%CONSISTE COLUNA 18-SEQUÊNCIA
call SCAN(PROCURA POR 'A' ou 'D' NA COLUNA 18);
if RESULTADO=ERRO
   then do;
          call ROT$ERRO(15,W);
          ASSUME COLUNA 18='Ø';
        end;
   else if RESULTADO=A
             or RESULTADO=D
             and USO$ARQ<>PRINCIPAL
```

```
            and USO$ARQ<>SECUNDÁRIO
            then do; call ROT$ERRO(15,W);
                     ASSUME COLUNA 18='Ø';
                 end;
%CONSISTE COLUNA 19-INDICA USO IMPRESSORA
call SCAN(PROCURA POR 'I' ou 'Ø', NA COLUNA 19);
if RESULTADO=ERRO
   then do; call ROT$ERRO(16,W);
            ASSUME COLUNA 18='Ø';
        end;
   else do;
         if RESULTADO=I  %IMPRESSORA EM USO
            and TIPO$ARQ<>SAÍDA
            and USO$ARQ<>BRANCO
            then do;
                  call ROT$ERRO(16,W);
                  ASSUME COLUNA 19='Ø';
                 end;
            else if RESULTADO=I
                    then IMPRES=RESULTADO;  %SALVA
        end;
%CONSISTE COLUNA 28-MODO PROCESSAMENTO
call SCAN(PROCURA POR 'R' ou 'Ø' NA COLUNA 28);
if RESULTADO=ERRO                %R=RANDOM
   then do;
         call ROT$ERRO(17,W);
         if USO$ARQ=CHAIN
            then ASSUME COLUNA 28='R';
            else ASSUME COLUNA 28='Ø';
        end;
   else do;
         if RESULTADO=RANDOM
            and TIPO$ARQ<>CHAIN
            then do;
                  call ROT$ERRO(18,W);
                  ASSUME COLUNA 28='Ø';
                 end;
         if RESULTADO=BRANCO1 and USO$ARQ=CHAIN
```

```
            then do;
                  call ROT$ERRO(19,W);
                  ASSUME COLUNA 28='R';
                end;
          end;
%CONSISTE COLUNAS 33-34 - OVERFLOW
call SCAN$BRANCO(COLUNAS 33,34);
if not BRANCO
   then do;
         if COLUNAS 33-34 NÃO CONTÉM CADEIA 'OV'
            then do;
                  call ROT$ERRO(20,W);
                  ASSUME OVERFLOW=Ø;  %SEM OVERF
                end;
            else if TIPO$ARQ<>SAÍDA
                    OR not IMPRES
                    then do;
                          OVERFLOW=Ø;  %SEM OVERF
                          call ROT$ERRO(21,W);
                        end;
                    else OVERFLOW=1;  %C/OVERFLOW
         end;
%CONSISTE COLUNA 39-EXTENSÃO/LINHA
call SCAN(PROCURA POR 'L' ou 'Ø' NA COLUNA 39);
if RESULTADO=ERRO
   then do;
         call ROT$ERRO(22,W);
         ASSUME COLUNA 39='Ø';
        end;
   else if RESULTADO=L and
           (TIPO$ARQ<>SAÍDA or not IMPRES)
           then do;
                 call ROT$ERRO(22,W);
                 ASSUME COLUNA 39='Ø';
                end;
FIM CONSISTÊNCIA * * * * * * * * * * * * * * *
%GERAÇÃO DE NÓ-INSERÇÃO NAS LISTAS DE ARQUIVOS DE ENTRADA E DE SAÍDA
if TIPO$ARQ=ATUALIZAÇÃO
   or TIPO$ARQ=ENTRADA
```

```
        then do; EFETUA   GERAÇÃO   UTILIZANDO   ELEMENTOS
                 CONSISTIDOS CONTIDOS NA ÁREA BUFFER$REG(REGISTRO FONTE),
                 ATRAVÉS DA UTILIZAÇÃO DAS ROTINAS BÁSICAS GERA1, GERA2, GE-
                 RAN, DE ACORDO COM DEFINIÇÕES PARA O CÓGICO INTERMEDIÁRIO
                 (PARTE 4);
                 INSERE NÓ GERADO NA LISTA DE ARQUIVOS DE ENTRADA;
             end;
      if TIPO$ARQ=ATUALIZAÇÃO
         or TIPO$ARQ=SAÍDA
        then do;
             EFETUA GERAÇÃO     UTILIZANDO     ELEMENTOS
             CONSISTIDOS CONTIDOS NA ÁREA BUFFER$REG(REGISTRO FONTE), ATRA-
             VÉS DA UTILIZAÇÃO DAS ROTINAS BÁSICAS GERA1, GERA2, GERAN,  DE
             ACORDO COM DEFINIÇÕES PARA O CÓDIGO INTERMEDIÁRIO (PARTE 4);
             INSERE NÓ GERADO NA LISTA DE ARQUIVOS DE SAÍDA;
           end;
      %INSERÇÃO NA TABELA DE SÍMBOLOS
       INSERE NA TABELA DE SÍMBOLOS, UTILIZANDO ROTINA BÁSICA
       INSERE$TS, OS ELEMENTOS DO ARQUIVO DE ACORDO COM DESCRIÇÃO DE ATRIBU-
       TOS;
      %INSERÇÃO NA TABELA DE ARQUIVOS LÓGICOS(TAL)
       INSERE OS ELEMENTOS: NOME INTERNO, NOME EXTERNO, PERIFÉRICO E TAMANHO
       DE REGISTRO NA TABELA TAL;
OUTRO:call LÊ$ESCREVE;
  end while REGISTRO TIPO 'F';
  if not HOUVE$ARQUIVO$PRINCIPAL
     then call ROT$ERRO(25,T);
  %CONSISTE CARTÃO TIPO 'L' (EXTENSÃO/LINHA) * * * * * * * * * * *
  if BUFFER$REG(COLUNA 6)='L'  %SE EXISTE REG. FONTE TIPO 'L'
  then do; %TESTA NOME ARQUIVO
        call SCAN$NOME(COLUNAS 7-12);
        if ERRO
           then do;
                call ROT$ERRO(26,T);
                go to FIM;
              end;
        call BUSCA$TS(NOME DO ARQUIVO,COLUNAS 7-12);
        if not ACHOU
           then do;
```

```
                call ROT$ERRO(26,T);
                go to FIM;
              end;
        if ATRIBUTO DA TS NÃO INDICA USO IMPRESSORA
           then do;
                call ROT$ERRO(29,T);
                go to FIM;
              end;
        %TESTA TAMANHO DO FORMULÁRIO
        call SCAN$NUM(PROCURA Nº NAS COLUNAS 15-17);
        if ERRO or VALOR FORA DOS LIMITES
           then call ROT$ERRO(27,T);
           else TAMANHO=VALOR;
        %TESTA LINHA DE OVERFLOW
        call SCAN$NUM(PROCURA Nº NAS COLUNAS 20-22);
        if ERRO or VALOR>TAMANHO
           then call ROT$ERRO(28,T);
           else OVERF=VALOR;
        %GERA ELEMENTOS
         GERA NÓ DESCRITOR TAMANHO E OVERFLOW, NA POSIÇÃO ADEQUADA;
      end;
FIM: call LÊ$ESCREVE;
end COMP$HFDL;
```

MÓDULO 2: SUBROTINA COMP$ENTRADA

```
          COMP$ENTRADA:proc;
   if BUFFER$REG(COLUNA 6)<>'I'
      then do;
           call ROT$ERRO(190,T); %ESPERADO REGISTRO TIPO 'I'
           abort FIM$COMPILA;    %INTERROMPE ANÁLISE
         end;
   PRIMEIRO$ARQUIVO=TRUE;
   %ENQUANTO HOUVER ESPECIFICAÇÃO DE TIPO 'I', FAÇA:
   do while BUFFER$REG(COLUNA 6)='I';
      ACABOU$ARQ, ACABOU$REG=FALSE;
      PRIMEIRA$LINHA$REG,PRIMEIRO$REG=TRUE;
      if PRIMEIRO$ARQUIVO
         then do; %VERIFICA SE ESPEC. CAMPO ANTECEDE ESPEC. REGISTRO
              call SCAN$BRANCO(COLUNAS 44-70);
              if BRANCO
                 then do; call ROT$ERRO(30,T);
                          ACABOU$REG=TRUE;
                      end;
              PRIMEIRO$ARQUIVO=FALSE;
            end;
      %VERIFICA NOME DO ARQUIVO
      call SCAN$NOME(COLUNA 7);
      if ERRO   %NOME INVÁLIDO
         then call ROT$ERRO(31,T);
         else do; %VERIFICA DEFINIÇÃO
              call BUSCA$TS(NOME DE ARQUIVO);
              if not ACHOU
                 then call ROT$ERRO(32,T);
                 else if (ARQUIVO<>ENTRADA and
                          ARQUIVO<>ATUALIZAÇÃO) or
                          ARQUIVO JÁ ESPECIFICADO
                          then call ROT$ERRO(34,T);
                          else SALVA ELEMENTOS DA TSÍMBOLOS;
            end;
      %ENQUANTO NÃO ACABOU ARQUIVO E FOR FORMULÁRIO ENTRADA
      do while not ACABOU$ARQ and BUFFER$REG(COLUNA 6)='I';
         %ENQUANTO FOR ESPECIFICAÇÃO DE REGISTRO
```

```
do while not ACABOU$REG;
   call SCAN$IND(VÊ SE TEM 'AND' OU 'OR' NAS COLUNAS 14, 15);
   if not EXISTE
      then do; %É ESPECIFIC. DE 1ª LINHA DE REGISTRO
            if not PRIMEIRA$LINHA$REG
               then call ROT$ERRO(33,W);
            if not PRIMEIRO$REG
               then do;
                     call SCAN$BRANCO(COLUNAS 7-14);
                     if not BRANCO then call ROT$ERRO(47);
                   end;
            CONSISTE CAMPOS DE SEQUÊNCIA, NÚMERO, OPÇÃO; COLUNAS 15-
            18, GERANDO ELEMENTOS - VER NOTAS;
            CONSISTE INDICADOR DE IDENTIFICAÇÃO DE REGISTRO, COLUNAS
            19 a 20 - GERA ELEMENTOS, MARCA USO;
            CONSISTE CÓDIGOS DE IDENTIFICAÇÃO DE REGISTROS - COLUNAS
            21 a 38 - VER NOTAS;
            %VERIFICA BRANCOS COLUNAS 43-70;
            call SCAN$BRANCO(COLUNAS 43-70);
            if not BRANCO then call ROT$ERRO(48,W);
            PRIMEIRA$LINHA$REG, PRIMEIRO$REG=FALSE;
         end;
      else do; %É LINHA AND/OR DE CONTINUAÇÃO
            if PRIMEIRA$LINHA$REG=TRUE
            then call ROT$ERRO(45,T); %LINHA AND/OR
                                      %FORA DE ORDEM
            call SCAN$BRANCO(COLUNAS 7-13);
            if not BRANCO
               then call ROT$ERRO(46,W); %IGNORA
            call SCAN$BRANCO(COLUNAS 17-20);
            if not BRANCO then call ROT$ERRO(49,W);
            %CONSISTE CÓDIGOS IDENTIF. REGISTRO - COLUNAS 21-38
            CONSISTE E GERA ELEMENTOS PARA LINHA AND/OR - VER NOTAS
            - 3;
         end;
      call LÊ$ESCREVE; %LÊ REGISTRO FONTE
      if BUFFER$REG(COLUNA 6)<>'I'
         then do;
               call ROT$ERRO(50,T);
```

```
                    ACABOU$ARQ,ACABOU$REG=TRUE;
                end;
            else do;
                    call SCAN$BRANCO(COLUNAS 7,41);
                    if BRANCO  %É ESPECIFIC. DE CAMPO
                       then do;
                            ACABOU$REG,PRIMEIRA$LINHA$REG=TRUE;
                            GERA FLAG NO DESCRITOR DE REGISTRO (FIM DA
                            CADEIA DE CÓDIGOS);
                            if not ERRO TERMINAL
                               then INSERE NÓ NA LISTA DE DESCRITORES DE
                                    REGISTROS;
                           end;
                end;
end while REGISTRO;
%ENQUANTO FOR ESPECIFICAÇÃO DE CAMPO, FAÇA:
do while ESPECIF. CAMPO (BRANCOS NAS COLUNAS 7-41) and
   BUFFER$REG(COLUNA 6)='I';
   %CONSISTE LOCALIZAÇÃO DO CAMPO (FROM)
   call SCAN$NUM(PROCURA Nº COLUNAS 44-47);
   if ERRO
      then do; call ROT$ERRO(52,T); %INVÁLIDO
               VALOR$FROM=1,        %ASSUME 1
           end;
      else if VALOR<1 or VALOR>512
              then do; call ROT$ERRO(52,T);
                       VALOR$FROM=1;
                   end;
              else VALOR$FROM=VALOR;
   %CONSISTE LOCALIZAÇÃO DO CAMPO (TO)
   call SCAN$NUM(PROCURA Nº COLUNAS 48-51);
   if ERRO
      then do;
            call ROT$ERRO(52,T);
            VALOR$TO=1;
           end;
      else if VALOR<1 or VALOR>512
              then do; call ROT$ERRO(52,T);
                       VALOR$TO=1
```

```
                    end;
                else VALOR$TO=VALOR
if VALOR$FROM>VALOR$TO
   then do;
         call ROT$ERRO(53,T);
         VALOR$TO=VALOR$FROM;
       end;
if VALOR$TO>TAMANHO$REG
   then do;
         call ROT$ERRO(41,T);
         VALOR$TO=TAMANHO$REG;
       end;
%CONSISTE POSIÇÕES DECIMAIS(PD)
call SCAN$NUM(PROCURA POR Nº OU BRANCO NA COLUNA 52);
if not Nº NEM 'Ø'
   then do; call ROT$ERRO(54,W); PD=Ø; end; %ASSUME
                                              %NUMÉRICO
   else if É NÚMERO
           then PD=VALOR;
           else PD='Ø';
%VERIFICAÇÃO DE TAMANHO DE CAMPO
TAMANHO=(VALOR$TO-VALOR$FROM)+1;
if CAMPO NUMÉRICO  %PD≠ 'Ø'
   then if TAMANHO>15
           then do; call ROT$ERRO(60,T); %ASSUME 15
                    TAMANHO=15;
                end;
   else do; %CAMPO ALFANUM.
         if TAMANHO>255
            then do;
                  call ROT$ERRO(60,T);
                  TAMANHO=255;
                end;
       end;
%CONSISTE NOME DE CAMPO
call SCAN$NOME (COLUNAS 53-58);
if ERRO then call ROT$ERRO(31,T);
   else do;
         call BUSCA$TS(NOME DO CAMPO);
```

```
         if not ACHOU
            then do;
                  call INSERE$TS(NOME DO CAMPO);
                  INSERE ELEMENTOS DO CAMPO NA TABELA DE SÍMBOLOS:
                  TAMANHO, TIPO, ENDEREÇO, ÁREA DE DADOS, REFERÊNCIA,
                  Nº DE ORDEM DO REGISTRO;
                 end;
            else do; %ACHOU
            if CAMPO NÃO DEFINIDO NO REGISTRO
               then do;
                     ATUALIZA Nº ORDEM DE REGISTRO NA TS;
                     if POSIÇÕES DECIMAIS or TAMANHO<>DADOS NA TS
                     then call ROT$ERRO(55,W);
                    end;
               else call ROT$ERRO(51,T);
                    end;
       end;
GERA ELEMENTOS NO DESCRITOR: ENDEREÇO DO CAMPO, VALOR$FROM; %LISTA DE
                                                            %CAMPOS
    CONSISTE INDICADORES DE CAMPO, MARCANDO USO E GERANDO ELEMENTOS
PARA DESCRITOR DA LISTA MOVE;
if not ACHOU      %CAMPO RECÉM DEFINIDO
   then GERA ÁREA DE DADOS, CONTENDO TAMBÉM TAMANHO E TIPO RELATIVOS
        AO CAMPO;
%INSERE NÓ NA LISTA MOVE
if not ERRO TERMINAL
   then INSERE NÓ NA LISTA DE DESCRITORES DE CAMPOS - LISTA MOVE;
%TRATA NÍVEL DE CONTROLE
call SCAN$BRANCO(COLUNAS 59,68);
if not BRANCO
   then do;
         call SCAN$IND(PROCURA POR INDIC. L1-L9 NAS COLUNAS 59-60);
         if not ENCONTROU L1-L9
            then call ROT$ERRO(57,T); %ASSUME BRANCO
            else do; %ENCONTROU
                  call BUSCA$TS(INDICADOR NÍVEL COLUNAS 59-60);
                  if not ACHOU
                     then do; %NÃO ESTÁ DEFINIDO
                           GERA NÓ LISTA NÍVEL COM ELEMENTOS: ENDERE
```

```
                                    ÇO ÁREA NÍVEL, INÍCIO, TAMANHO, TIPO/POSI-
                                    ÇÕES DECIMAIS, INDICADOR NÍVEL;
                                    INSERE NA TABELA SÍMBOLOS: REFERÊNCIA, Nº
                                    ORDEM REGISTRO, TAMANHO CAMPO, TIPO/POSI-
                                    ÇÕES DECIMAIS;
                                    GERA ÁREA NÍVEL;
                                    INSERE NA TS: ENDEREÇO ÁREA NÍVEL;
                                  end not ACHOU;
                              else do; %JÁ DEFINIDO - ACHOU
                                    if INDICADOR JÁ DEFINIDO NO REGISTRO
                                       then call ROT$ERRO(51,T);
                                       else do;
                                             ATUALIZA Nº ORDEM REGISTRO NA TS;
                                             VERIFICA TIPO/POSIÇÕES DECIMAIS
                                             COM TS;
                                             GERA NÓ LISTA NÍVEL;
                                          end;
                                  end ACHOU;
                            if not ERRO TERMINAL
                               INSERE NÓ NA LISTA NÍVEL;
                            end ENCONTROU;
                     end not BRANCO;
          CONSISTE INDICADORES MATCH - GERA NÓ MATCH E INSERE NA LISTA MATCH -
          VER NOTAS;
          call LÊ$ESCREVE;
          call SCAN$BRANCO(COLUNAS 7-42);
          %SE BRANCO NÃO TERMINOU ESPEC. CAMPO
          end while ESPECIFICAÇÕES CAMPO;
      VERIFICA Nº DE INDICADORES MATCH NO REGISTRO;
      call SCAN$BRANCO(COLUNAS 7-13);
      if BRANCO
         then ACABOU$ARQ=TRUE;
   end while ARQUIVO;
   if HOUVE CAMPOS MATCH ESPECIFICADOS NO ARQUIVO
      then do;
            GERA ÁREA MATCH RELATIVA AO ARQUIVO;
            GERA ENDEREÇOS ÁREA MATCH NOS DESCRITORES LISTA MATCH, E NO DESCRI-
            TOR ARQUIVO;
          end;
   if HOUVE REGISTROS COM CONTROLE DE SEQUÊNCIA NO ARQUIVO
```

```
    then do;
          GERA ÁREA DE CONTROLE DE SEQUÊNCIA
          GERA ENDEREÇO DA ÁREA NO DESCRITOR DE ARQUIVO;
        end;
end while ESPECIFICAÇÕES ENTRADA;
end COMP$ENTRADA;
```

```
MÓDULO 3: SUBROTINA COMP$CÁLCULOS

          COMP$CÁLCULOS:proc;

   %ENQUANTO HOUVER ESPECIFICAÇÃO DE CÁLCULOS, FAÇA:
   do while BUFFER$REG(COLUNA 6)='C';
      %CONSISTE COLUNA 7-8 - INDICADOR DE NÍVEL
      %NOTA-ORDEM:CÁLCULOS DETALHE-CÁLCULOS TOTAL(LO-L9)-CÁLCULOS TOTAL(LR)-
      %CÁLCULOS DE SUBROTINAS(SR)
      call SCAN$IND(COLUNAS 7-8);
      do case CONTEÚDO$COLUNAS$7$8;
         do; %INDICADORES LØ-L9
          if HOUVE CÁLCULO$SUBR or CÁLCULO$LR then call ROT$ERRO (70,T);
             else do;
                   if NÃO HOUVE CÁLCULO$TOTAL
                      then do;
                            if HOUVE CÁLCULO$DETALHE
                               then GERA FLAG; %FIM DETALHE
                            MARCA LIMITE$SALTO;
                            ABRE LISTA CÁLCULO$TOTAL;
                           end;
                  end;
          SALVA INDICADOR DE NÍVEL;
          INDICA QUE HPUVE CÁLCULO$TOTAL;
        end;
         do; %INDICADOR LR
          if HOUVE$CÁLCULO$SUBR
             then call ROT$ERRO(70,T);
             else do;
                   if NÃO HOUVE CÁLCULO TOTAL and NÃO HOUVE CÁLCULO$LR
                      then do;
                            if HOUVE CÁLCULO$DETALHE
                               then GERA FLAG; %FIM DETALHE
                            MARCA LIMITE$SALTO;
                            ABRE LISTA CÁLCULO$TOTAL;
                           end;
                  end;
          SALVA INDICADOR DE NÍVEL;
```

```
    INDICA QUE HOUVE CÁLCULO TOTAL;
  end;
   do; %INDICADOR NÃO EXISTE - BRANCO COLUNAS 7-8
    if HOUVE CÁLCULO$SUBR or CÁLCULO$TOTAL
       or CÁLCULO$LR
       then call ROT$ERRO(70,T);
       else do;
             if NÃO HOUVE CÁLCULO$DETALHE
                then do;
                      MARCA LIMITE$SALTO;
                      ABRE LISTA DETALHE;
                     end;
            end;
    SALVA INDICADOR DE NÍVEL=0; %NÃO EXISTE
    INDICA QUE HOUVE CÁLCULO$DETALHE;
  end;
   do; %SIGLA SUB-ROTINA(SR)
   if NÃO HOUVE CÁLCULO$DETALHE
      and NÃO HOUVE CÁLCULO$TOTAL
      and NÃO HOUVE CÁLCULO$LR
      then call ROT$ERRO(71,T);
      else if NÃO HOUVE CÁLCULO$SUBR
              then GERA FLAG;
   SALVA INDICADOR DE NÍVEL=Ø; %NÃO EXISTE
   INDICA QUE HOUVE CÁLCULO$SUBR
  end;
   do; %'AND' ou 'OR'
    call ROT$ERRO(72,T); %FORA ORDEM
  end;
   do; %OUTROS
    call ROT$ERRO(73,T); %ENTRADA INVÁLIDA
  end;
end case;
%TRATA INDICADORES DE CÁLCULO-COLUNAS 9-17
%CONSISTE INDICADORES GERANDO ELEMENTOS NUM BUFFER PARA USO POSTERIOR
%NA MONTAGEM DO CÓDIGO PARA CADA OPERAÇÃO
call SCAN$BRANCO(COLUNAS 9-17);
if BRANCO
   then do; GERA FLAG INDICADOR DE FIM DE CADEIA DE INDICADORES; %NÃO
```

```
            %HÁ INDICADORES
            INDICA QUE NÃO HÁ INDICADORES;
         end;
      else do;
         CONSISTE INDICADORES E GERA CADEIA DE INDICADORES PARA CÓDIGO
         INTERMEDIÁRIO; %NO BUFFER
         call SCAN$BRANCO(COLUNAS 18-59);
         do while BRANCO COLUNAS 18-59;
            %CONSISTE LINHAS AND/OR
            call LÊ$ESCREVE;
            call SCAN$IND(PROCURA POR 'AND' OU 'OR', COLUNAS 7-8);
            if NÃO É LINHA 'AND' NEM LINHA 'OR'
               then call ROT$ERRO(77,T);
               else do; %É LINHA 'AND' OU LINHA 'OR'
                     GERA INDICAÇÃO DE LINHA 'AND' OU LINHA 'OR' NA CA
                     DEIA DE INDICADORES PARA CÓDIGO INTERMEDIÁRIO;
                     CONSISTE INDICADORES GERANDO ELEMENTOS PARA CÓDI-
                     GO INTERMEDIÁRIO;
                     if NÃO TEM INDICADORES
                        then call ROT$ERRO(78,T);
                     call SCAN$BRANCO(COLUNAS 18-59);
                  end;
            end while BRANCO;
            GERA FIM DE CADEIA DE INDICADORES;
         end;
   %TRATA DEMAIS ELEMENTOS POR CLASSE DE OPERAÇÃO
   VERIFICA CLASSE DE OPERAÇÃO EM TABELA;
   do case CLASSE;
    CONSISTE DEMAIS ELEMENTOS DA OPERAÇÃO GERANDO CÓDIGO INTERMEDIÁRIO FI-
    NAL. OS ELEMENTOS RELATIVOS A INDICADORES SÃO EXTRAÍDOS DO BUFFER UTI
    LIZADO;
   end case;
   call LÊ$ESCREVE;
end while ESPECIFICAÇÕES DE CÁLCULOS;
if NÃO HOUVE CÁLCULOS SUBR
   then GERA FLAG DA ÚLTIMA OPERAÇÃO;
end ESPECIFICAÇÕES CÁLCULO;
```

MÓDULO 4: SUBROTINA COMP$SAÍDA

```
          COMP$SAÍDA:proc;
     PRIMEIRO$ARQUIVO=TRUE %CHAVE CONTROLE P/1º ARQ.
    %ENQUANTO FOR ESPECIFICAÇÃO DE SAÍDA FAÇA:
    do while BUFFER$REG(COLUNA 6)='0';
       ACABOU$ARQUIVO, ACABOU$REG=FALSE;
       PRIMEIRA$LINHA$REG,PRIMEIRO$REG=TRUE; %CONTROLE
       if PRIMEIRO$ARQUIVO                   %SEQUÊNCIA
          then do; %VERIFICA SE ESPECIF.CAMPO ANTECEDE
                   %ESPECIFIC.REG
                call SCAN$BRANCO(COLUNAS 7 a 22);
                if BRANCO
                   then do; call ROT$ERRO(150,T);
                            ACABOU$REG=TRUE; %PROCESSA ESPECIF.CAMPO
                        end;
                PRIMEIRO$ARQUIVO=FALSE; %NÃO USADA MAIS DAQUI EM DIANTE
             end;
       %VERIFICA NOME DO ARQUIVO
       call SCAN$NOME(COLUNA 7);
       if ERRO %NOME INVÁLIDO
          then call ROT$ERRO(183,T);
          else do; %VERIFICA DEFINIÇÃO
                call BUSCA$TS(NOME DE ARQUIVO);
                if not ACHOU
                   then call ROT$ERRO(155,T);
                   else if (ARQUIVO<>SAÍDA AND ARQUIVO<>ATUALIZAÇÃO)
                           or ARQUIVO JÁ ESPECIFICADO
                           then call ROT$ERRO(151,T);
                           else SALVA ELEMENTOS DA TABELA PARA POSTERIOR USO;
             end;
       %ENQUANTO NÃO ACABOU ESPEC. DO ARQUIVO E FOR FORMULÁRIO DE SAÍDA FAÇA:
       do while not ACABOU$ARQUIVO
          and BUFFER$REG(COLUNA 6)='0';
          %ENQUANTO FOR ESPECIFICAÇÃO DE REGISTRO FAÇA:
          do while not ACABOU$REG;
             call SCAN$IND(VE SE TEM 'AND' ou 'OR' NAS COLUNAS 14-15);
             if not EXISTE
                then do; %É ESPECIF. DE PRIMEIRA LINHA DE REGISTRO
```

```
        if not PRIMEIRA$LINHA$REG
           then call ROT$ERRO(153,W);
           %FALTOU ESPECIF. DE CAMPO
        if not PRIMEIRO$REG
           then do;
                  call SCAN$BRANCO(COLUNAS 7-14);
                  if not BRANCO
                     then call ROT$ERRO(154,W);
                end;
        %CONSISTE COLUNA 15 - TIPO DE LINHA
        call SCAN(BUSCA POR 'H', 'D', 'T', OU 'E',NA COLUNA 15);
        if RESULTADO=ERRO %NÃO ACHOU
        then call ROT$ERRO(152,T); %SE ERRO ASSUME 'D'
        %CONSISTE COLUNAS 17-22 - ESPAÇO/SALTO
         CONSISTE E GERA ELEMENTOS DO CÓDIGO INTERMEDIÁRIO, UTI
         LIZANDO ROTINAS BÁSICAS - VER NOTAS;
        %CONSISTE COLUNAS 23-31 - INDICADORES SAÍDA
         CONSISTE E GERA ELEMENTOS DO CÓDIGO INTERMEDIÁRIO, UTI
         LIZANDO ROTINAS BÁSICAS - VER NOTAS;
        %VERIFICA BRANCOS COLUNAS 32-70
        call SCAN$BRANCO(COLUNAS 32-70);
        if not BRANCO
           then call ROT$ERRO(168,W); %ASSUME BRANCO
        PRIMEIRA$LINHA$REG=FALSE;
        PRIMEIRO$REG=FALSE;
     end;
else do; %É LINHA DE CONTINUAÇÃO: AND/OR
        if PRIMEIRA$LINHA$REG=TRUE
           then call ROT$ERRO(169,T); %LINHA AND/OR FORA DE OR-
                                      %DEM
        call SCAN$BRANCO(COLUNA 7-13);
        if not BRANCO then call ROT$ERRO(170,W); %IGNORA CONTEÚ
              %DO COLUNAS 7-13
        %CONSISTE COLUNAS 17-22 - ESPAÇO/SALTO P/ LINHA AND/OR
         CONSISTE E GERA ELEMENTOS DO CÓDIGO INTERMEDIÁRIO, UTI
         LIZANDO ROTINAS BÁSICAS - VER NOTAS;
        %CONSISTE COLUNAS 23-31 - INDICADORES P/LINHA AND/OR
         CONSISTE E GERA ELEMENTOS DO CÓDIGO INTERMEDIÁRIO, UTI
         LIZANDO ROTINAS BÁSICAS - VER NOTAS;
```

```
            %VERIFICA BRANCOS - COLUNAS 32-70
            call SCAN$BRANCOS(COLUNAS 32-70);
            if not BRANCO
               then call ROT$ERRO(168,W); %ASSUME BRANCO
         end;
   call LÊ$ESCREVE; %LÊ REGISTRO FONTE
   if BUFFER$REG(COLUNA 6)<>'0'
      then do; ACABOU$ARQUIVO, ACABOU$REG=TRUE;
               call ROT$ERRO(173,T); %FALTOU CAMPOS
         end;
      else do; %VERIFICA SE É ESPECIF. DE CAMPO
            call SCAN$BRANCO(COLUNAS 7-22)
            if BRANCO %É ESPECIF. CAMPO
               then do;
                     ACABOU$REG, PRIMEIRA$LINHA$REG=TRUE;
                     GERA FLAG; FIM DE DESCRITOR REGISTRO;
                     INSERE NÓ NA LISTA DE REGISTROS (SE NÃO HOUVE
                     ERRO TERMINAL);
                  end;
         end;
end while ESPECIF. REGISTRO;
%ENQUANTO FOR ESPECIF. DE CAMPOS DE SAÍDA, FAÇA;
do while ESPECIF. CAMPOS(BRANCOS NAS COLUNAS 7-22)
   and BUFFER$REG(COLUNA 6)='0';
   %CONSISTE INDICADORES DE SAÍDA P/CAMPOS
    CONSISTE E GERA ELEMENTOS DO CÓDIGO INTERMEDIÁRIO, UTILIZANDO
    ROTINAS BÁSICAS - VER NOTAS;
   %CONSISTE NOME DO CAMPO, COD.EDIÇÃO, PALAVRA EDIÇÃO, POSIÇÃO,
   %BRANCO AFTER;
   call SCAN$BRANCO(COLUNAS 32-37);
   if BRANCO %NÃO HÁ NOME DE CAMPO
      then do;
            VERIFICA SE HÁ CONSTANTE COLUNAS 45-70;
            if CONSTANTE NÃO EXISTE OU COM ERRO
               then do;
                     call ROT$ERRO(174,T);
                     go to FIM$LAÇO; %IGNORA  ESPECIFIC. CAMPO DES-
                                     %TA LINHA
                  end;
```

```
        %CONSTANTE EXISTE
        if CÓDIGO EDIÇÃO or
           BRANCO-AFTER<>'Ø'
           then call ROT$ERRO(175,W); %ASSUME Ø
        GERA CONSTANTE, TIPO, TAMANHO NA ÁREA DE DADOS;
        GERA ENDEREÇO DE CONSTANTE NO NÓ DESCRITOR DE CAMPO;
        GERA ESPECIF. ADEQUADAS P/ELEMENTOS SEM USO NO NÓ CAMPO
        (ENDER.PALAVRA EDIÇÃO, CÓDIGO EDIÇÃO, BRANCO AFTER);
        CONSISTE E GERA CAMPO POSIÇÃO - VER NOTAS;
     end;
  else do; %HÁ NOME DE CAMPO
        call SCAN$NOME(NOME DE CAMPO);
        if ERRO
           then do;
                 call ROT$ERRO(176,T);
                 go to FIM$LAÇO; %IGNORA ESPEC. CAMPO DESTA LI-
                                 %NHA
              end;
        call BUSCA$TS(NOME DE CAMPO); %BUSCA NA TABELA DE SÍMBO
                                      %LOS
        if not ACHOU
           then do;
                 call ROT$ERRO(177,T); %NOME NÃO DEFINIDO
                 goto FIM$LAÇO; %IGNORA ESPEC. DESTA LINHA
              end;
        %NOME VÁLIDO PARA CAMPO
        GERA ENDEREÇO DO CAMPO NO DESCRITOR;
        if CAMPO NUMÉRICO
           then do; %VERIFICA COD.EDIÇÃO E PALAVRA EDIÇÃO
                 CONSISTE PALAVRA DE EDIÇÃO;
                 if PALAVRA VÁLIDA
                    then do;
                          GERA PALAVRA NA ÁREA DE DADOS COM TA-
                          MANHO E TIPO;
                          GERA ENDEREÇO DA PALAVRA NO NÓ DESCRI
                          TOR CAMPO;
                         end;
                    else GERA ENDEREÇO=Ø NO NÓ DESCRITOR;
                         %NÃO EXISTE OU INVÁLIDO
```

```
                        CONSISTE CÓDIGO EDIÇÃO;
                        if CÓDIGO EDIÇÃO INVÁLIDO and<>'Ø'
                           then call ROT$ERRO(178,T);
                           else if CÓDIGO<>'Ø'
                                   and PALAVRA VÁLIDA
                                   then call ROT$ERRO(179,T);
                        GERA CÓDIGO EDIÇÃO;
                     end;
               CONSISTE BRANCO-AFTER - COLUNA 39
               call SCAN(PROCURA POR 'Ø' ou 'B' NA COLUNA 34);
               if RESULTADO=ERRO %NÃO ENCONTROU
                  then do; call ROT$ERRO(180,W);
                           GERA NO DESCRITOR 'B' %ASSUME 'B'
                       end;
                  else GERA VALOR DO BUFFER$REG;
               CONSISTE POSIÇÃO DO CAMPO
               CONSISTE E GERA NO DESCRITOR - VER NOTAS - 3;
            end;
      %INSERE NÓ NA LISTA DE DESCRITORES DE CAMPOS
       CALCULA TAMANHO;
       GERA TAMANHO; %INSERIU
      %LÊ PRÓXIMO REGISTRO
FIM$LAÇO: call LÊ$ESCREVE;
      call SCAN$BRANCO(COLUNAS 7-22);
      if not BRANCO %ACABOU ESPEC. CAMPOS
         then GERA FLAG DE FIM DE LISTA DE CAMPOS;
   end while ESPECIFICAÇÕES CAMPOS;
   %VERIFICA SE TERMINOU ARQUIVO;
   call SCAN$BRANCO(COLUNAS 7-12);
   if BRANCO
      then ACABOU$ARQUIVO=TRUE;
 end while ARQUIVO;
end while ESPECIF. SAÍDA;
%ENCERRA COMPILAÇÃO
MONTA EM DISCO O CÓDIGO GERADO EM FORMATO DE ENTRADA COMPATÍVEL COM O PRO
GRAMA LINQUEDITOR(REFEX);
end COMP$SAÍDA;
```

APÊNDICE "B" - CODIFICACÃO EM RPG - APLICACÃO

PROGRAMA EXEMPLO:

Descrição

O programa faz a atualização de um arquivo de estoques (em disco) e imprime um relatório de ocorrências.

Entradas.

a) Arquivo mestre em disco.

Nome : MESTRE

Tamanho do registro : 80

Conteúdo dos registros :

NOME DO CAMPO	DESCRIÇÃO DO CAMPO	POSIÇÕES
CÓDIGO	Código identificação = 1	1
NITEM	nº do ítem no estoque	2 - 7
DESCR	descrição do ítem	8 - 29
QESTOQ	quantidade em estoque	34 - 39
QENCOM	quantidade em encomenda	40 - 45
MAX	estoque máximo	46 - 49
MIN	estoque mínimo	50 - 53

b) Arquivo de transações (entradas e saídas no estoque)

Nome: TRANSA

Tamanho do registro : 80

Conteúdo do registros :

NOME DO CAMPO	DESCRIÇÃO DO CAMPO	POSIÇÕES
CÓDIGO	Código de identificação s - saída e - entrada	1
NITEM ENTRAD/ SAÍDAS	nº do ítem no estoque entrada ou saída no estoque	2 - 7 8 - 12

Nota: - dois tipos de registros: entrada ou saída.

- registros em qualquer número, e opcionais
- ordem: registros de entrada no estoque, seguem regis-

tros de saídas.

Saídas.

a) Arquivo MESTRE atualizado
b) Relatório da atualização (ocorrências)

H		RELATÓRIO DE SITUAÇÃO DO ESTOQUE					
H	Nº ÍTEM	DESCRIÇÃO	QUANTIDADE EM ESTOQUE	QUANTIDADE ENCOMENDA	TRANSAÇÃO	SALDO MÍNIMO	SALDO MÁXIMO
D	XXXXXX	X————X	XXX,XXX	XXX,XXX		XXXX	XXXX
D		SAÍDA			XX,XXX		
D		SAÍDA			XX,XXX		
D		ENTRADA			XX,XXX		
T			XXX,XXX (NOVA QTDA DE EM ESTOQUE)	XXX,XXX (NOVA QTDA DE ENCOMENDA)		ABAIXO	ACIMA
D	XXXXXX	X————X	XXX,XXX	XXX,XXX		XXXX	XXXX

Nota: - espaço simples em linhas detalhe
- espaço duplo entre as linhas detalhe e total
- cabeçalhos em cada página.

Cálculos.

Acumula entradas e saídas para cada ítem e atualiza campos de quantidade em estoque e quantidade em encomenda. Verifica situação do estoque atualizado com relação aos limites, mínimo e máximo, previstos.

OBSERVAÇÃO: O CÓDIGO INTERMEDIÁRIO RELATIVO A ESTE PROGRAMA OCUPA APROXIMADAMENTE 1450 BYTES.

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

GX 17-0047-3 U/M 050
Impresso no Brasil
*No. of forms per pad may very slightly

IBM International Business Machine Corporation

Program	PROGRAMA 1	Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 01 of __ (1 2) Program Identification PROGR1 (75 76 77 78 79 80)

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	Debug	MFCM Stacking Sequence	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Model 20: Address to Start	Model 20: Work Tapes	Model 20: Overlay Open	Model 20: Overlap Printer	Model 20: Binary Search	Model 20: Type Error	Model 20: 2152 Checking	Model 20: Inquiry	Model 20: Read/Write/Compute	Model 20: Keyboard Output	Sim Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Table Load Halt	Shared I/O	Field Print	Formatted Core Dump	RPG to RPG II Conversion	Refer to the specific System Reference Library manual for actual entries.
(3 4 5)	(6)	(7 8 9)	(10)	(11)	(12 13 14)	(15)	(16)	(17 18 19 20)	(21)	(22)	(23 24 25)	(26)	(27 28 29 30)	(31)	(32)	(33)	(34)	(35)	(36)	(37)	(38)	(39)	(40)	(41)	(42)	(43)	(44)	(45)	(46)	(47)	(48)	(49)	(50)	(51)	(52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74)
0 1	H																																	

File Description Specification

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	Continuation Lines: K	Continuation Lines: Option	Continuation Lines: Entry	A/U	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8
		(7 8 9 10 11 12 13 14)	(15) I/O/U/C/D	(16) P/S/C/R/T/D	(17) E	(18) A/D	(19) F/V/S/M/D	(20 21 22 23)	(24 25 26 27)	(28) L/R	(29 30)	(31) A/P/I/K	(32) I/D/T or 2	(33 34)	(35 36 37 38)	(39)	(40 41 42 43 44 45 46)	(47 48 49 50 51 52)	(53)	(54 55 56 57 58 59)	(60 61 62 63 64 65)	(66)	(67)	(68 69)	(70) R/U/N	(71 72)	(73 74)
0 2	F	TRANSA	I	S		A			80								DISCO															
0 3	F	*																														
0 4	F	MESTRE	U	P		A			80								DISCO															
0 5	F	*																														
0 6	F	IMPR	O			I			96					OV			IMPRES															
0 7	F																															
0 8	F																															
0 9	F																															
1 0	F																															
	F																															
	F																															

(72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1)

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

GX17-0049-2 U/M 050
Impresso no Brasil

Program	PROGRAMA 1	Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 02 of ____ Program Identification

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	TRANSA	01	N	O	20			C	S																				
02	I																				2	7		NITEM	L1	M1				
03	I																				8	12		SAIDAS						
04	I		02	N	O	30			C	E																				
05	I																				2	7		NITEM	L1	M1				
06	I																				8	12		ENTRAD						
07	I	*																												
08	I	*																												
09	I	MESTRE		N	S	40			C	1																				
10	I																				2	7		NITEM	L1	M1				
11	I																				8	29		DESCR						
12	I																				34	39	0	QESTOQ						
13	I																				40	45	0	QENCOM						
14	I																				46	49	0	MAX						
15	I																				50	53	0	MIN						
16	I																													
17	I																													
18	I																													
19	I																													
20	I																													
	I																													
	I																													
	I																													
	I																													
	I																													

GX 17-0048-2 UM:050
Impresso no Brasil
*No. of forms per pad may vary slightly

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Program: PROGRAMA 1 | Punchig Instruction | Graphic | Punch | Card Electro Number
Programmer | Date

Page 03 of __ | Program Identification

C

Line	Form Type	Control Level (L0, L1, ..., L9, LR, SR, AN/OH)	Indicators: And / And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup (Factor 2): High / Low / Equal	Comments
01	C		20 MR	SAIDAS	ADD	TOTSAI	TOTSAI		50			
02	C		30 MR	ENTRAD	ADD	TOTENT	TOTENT		50			
03	C	L1		OESTOQ	SUB	TOTSAI	OESTOQ					
04	C	L1		OESTOQ	ADD	TOTENT	OESTOQ					
05	C	L1		OENCOM	SUB	TOTENT	OENCOM					
06	C	L1		OESTOQ	COMP	MAX					99	
07	C	L1		OESTOQ	COMP	MIN					88	
08	C	L1		TOTSAI	SUB	TOTSAI	TOTSAI					
09	C	L1		TOTENT	SUB	TOTENT	TOTENT					
10	C											
11	C											
12	C											
13	C											
14	C											
15	C											
16	C											
17	C											
18	C											
19	C											
20	C											
	C											
	C											
	C											
	C											
	C											

RPG OUTPUT SPECIFICATIONS

GX17-0045-2 U/M 050
Impresso no Brasil

IBM International Business Machine Corporation

Program	PROGRAMA 7	Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page 04 of __ Program Identification

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Constant or Edit Word

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # (Fetch) (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Output Indicators Not	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	IMPR	H			3	06				1	P					
02	O		OR								0	V					
03	O														63		'RELATORIO DE SITUACAO DO'
04	O														70		'ESTOQUE'
05	O		H		1						1	P					
06	O		OR								0	V					
07	O														9		'NO.ITEM'
08	O														26		'DESCRICAO'
09	O														46		'QUANTIDADE'
10	O														58		'QUANTIDADE'
11	O														68		'TRANSACAO'
12	O														83		'SALDO'
13	O														92		'SALDO'
14	O		H		2						1	P					
15	O		OR								0	V					
16	O														46		'EM ESTOQUE'
17	O														57		'ENCOMENDA'
18	O														84		'MINIMO'
19	O														93		'MAXIMO'
20	O		D		1						4	0					
	O											NITEM			8		
	O											DESCR			34		
	O											QESTOQ1			43		
	O											QENCOM1			56		
	O											MIN Z			81		

*No. of forms per pad may very slightly

IBM do Brasil Ltda.

RPG OUTPUT SPECIFICATIONS

GX17-0045-2 U/M 050
Impresso no Brasil

IBM International Business Machine Corporation

Program	PROGRAMA 1	Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page 05 of __ Program Identification

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Output Indicators And	Field Name	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
									Not	Not	Not	*AUTO	B/A/C/1-9/R			
01	O											MAX	Z	88		
02	O		D			1			20							
03	O													17		'SAIDA'
04	O											SAIDAS	2	68		
05	O		D			1			30							
06	O													19		'ENTRADA'
07	O											ENTRAD	2	68		
08	O		T		12				L1							
09	O											QESTOO	1	43		
10	O													45		' **'
11	O											QENCOM	1	55		
12	O													57		' **'
13	O								88					81		'ABATXO'
14	O								99					92		'ACIMA'
15	O	*														
16	O	*														
17	O	MESTRE	T						L1	40	MR					
18	O											QESTOO		39		
19	O											QENCOM		45		
20	O															
	O															
	O															
	O															
	O															

*No. of forms per pad may vary slightly

APÊNDICE "C" - MENSAGENS DE ERRO PREVISTAS PARA O PROCESSO DE COMPILACAO

MENSAGENS DE ERRO

ERRO TIPO W: ERRO DE ADVERTÊNCIA
ERRO TIPO T: ERRO TERMINAL

1. ESPECIFICAÇÕES DE CONTROLE, DE DESCRIÇÃO DE ARQUIVOS E DE CONTADOR DE LINHA

Nº	TIPO	MENSAGEM
1	W	Faltou cartão tipo 'H' - Nome assumido para o programa 'JOBRPG'.
2	T	Após cartão tipo 'H' deve vir cartão tipo 'F'.
3	T	Tamanho do registro inválido ou > 512 - assume tamanho = 80.
4	W	Reserva.
5	W	Nome para periférico branco ou inválido - assume 'DUMMY'.
6	W	Nome externo para arquivo branco ou inválido - assume nome interno.
7	T	Nome interno para arquivo inválido - ignora especificações para o arquivo.
8	T	Nome de arquivo já definido - ignora especificações para arquivo.
9	T	Entrada inválida coluna 15 (Tipo de arquivo) - ignora especificações para o arquivo.
10	W	Entrada inválida coluna 16 (uso de arquivo) - se tipo de entrada ou atualização assume secundário; se arquivo de saída assume branco.
11	W	Entrada inválida na coluna 16 (uso do arquivo) - para arquivo de saída - assume branco.
12	W	Arquivo principal já definido - assume uso do arquivo = secundário.
13		RESERVA
14	W	Entrada inválida coluna 17 (Fim de arquivo) - assume branco.
15	W	Entrada inválida coluna 18 (Sequência) - assume branco.
16	W	Entrada inválida coluna 19 - assume branco.
17	W	Entrada inválida coluna 28 (MODO DE PROCESSAMENTO) - assume 'R' para arquivo encadeado (CHAINED) - senão assume branco.

Nº	TIPO	MENSAGEM
18	W	Entrada inválida coluna 28 - arquivo não é encadeado - assume branco.
19	W	Entrada inválida coluna 28 para arquivo encadeado (CHAINED) - assume 'R'.
20	W	Entrada inválida colunas 33-34 (OVERFLOW) - assume branco.
21	W	Entrada inválida colunas 33-34 - overflow indicado para arquivo < > arquivo na impressora - assume branco.
22	W	Entrada inválida coluna 39 (código extensão/linha)- assume branco.
23		RESERVA
24		RESERVA
25	T	Faltou especificação para arquivo principal.
26	T	Nome de arquivo para registro tipo''L (LINHA) inválido ou não definido no formulário de especificações de arquivo.
27	T	Tamanho do formulário inválido ou fora dos limites (registro fonte tipo L).
28	T	Linha de overflow inválida ou > tamanho do formulario' (registro fonte tipo L).
29	T	Especificação de linha para arquivo diferente de impressora.

2. ESPECIFICAÇÕES DE ENTRADA

Nº	TIPO	MENSAGEM
30	T	Especificação de campo não precedida da especificaçãp' de arquivo e registro.
31	T	Nome de arquivo de entrada ou de campo inválido.
32	T	Nome de arquivo não definido nas especificações de arquivo ou arquivo não é de entrada nem atualização
33	W	Faltou especificação de campo - continua.
34	T	Arquivo não é de entrada nem de atualização ou já especificado.
35	W	Entrada inválida colunas 15-16 - assume 'NS' para registro fora de grupo sequencial - nº sequencial anterior caso contrário.
36	W	Entrada inválida colunas 15-16; assume 01 para primeiro registro do grupo sequencial - assume número sequencial anterior caso não seja primeiro registro.

Nº	TIPO	MENSAGEM
37	W	Entrada inválida para coluna 17, para sequência controlada _ ASSUME 'N'.
38	W	Entrada inválida para coluna 18, para sequência controlada - assume '0'.
39	T	Entrada inválida colunas 17.18 para sequência sem controle.
40	W	Esquecido preenchimento colunas 19-20, ou preenchimento inválido - assume branco.
41	T	Entrada inválida para posição ou posição maior do que tamanho do registro.
42	W	Entrada inválida para colunas 25,32 ou 39 (NOT) - assume 'N'.
43	W	Entrada inválida para colunas 26-33 ou 40 (CARATER/ZONA/DITO) - assume 'C'.
44	T	Linha AND/OR sem código de identificação de registro.
45	T	Linha AND/OR fora de ordem.
46	W	Linha AND/OR com nome de arquivo especificado.
47		RESERVA
48	W	Especificação de campo na mesma linha de especificação de registro - assume branco.
49	W	Linha AND/OR - com preenchimento das colunas 17-22 - assume branco.
50	T	Faltou especificação de campo.
51	T	Campo, indicador de nível, ou indicador de match já definido no registro.
52	T	Entrada inválida para posição de campo - assume 1.
53	T	Valor$fron = valor$to.
54	T	Entrada inválida para posições decimais - assume zero(numérico).
55	W	Campo previamente definido com tamanho ou tipo (posições decimais) diferentes - assume primeira definição.
56	W	PLUS_MINUS referenciado para campo alfanumérico ou indicado inválido.
57	T	Indicador de nível ou de match inválido - assume branco.
58	T	Tamanho ou tipo de campo de nível de controle ou de match diferente de definição anterior - assume definição anterior.
59	T	Indicador de match diferente do definido no primeiro registro com indicadores match.

Nº	TIPO	MENSAGEM
60	T	Tamanho de campo numérico maior do que 15, ou de campo alafanumérico maior do que 255 - assume tamanho = 15 para numérico e tamanho 255 para alfanumérico.
61	T	Linha de continuação AND/OR, sem haver especificação de 1ª linha de registro.
62		RESERVA - ATÉ Nº 68

3. ESPECIFICAÇÃO DE CÁLCULO

Nº	TIPO	MENSAGEM
69	T	Estourou buffer para indicadores de cálculo.
70	T	Especificações de cálculo fora de ordem.
71	T	Especificações de subrotina não seguem especificações de cálculo detalhe ou total.
72	T	Somente última linha AND/OR pode possuir especificação de cálculo.
73	T	Entrada inválida colunas 7-8 (indicador de nível).
74	T	Entrada inválida colunas (10-11), (13,14) ou (16,17).
75	T	Indicador não definido.
76	W	Entrada inválida para colunas 9-12 ou 15 (NOT) - assume 'N'.
77	T	Faltaram especificações de cálculo no registro fonte anterior.
78	T	Linha AND/OR sem indicadores especificados.
79	T	Operação de cálculo inválida.
80	T	Nome inválido.
81	T	Tipo de campo resultado não permitido para a operação.
82	W	Campo resultado (já definido) com tamanho inválido ou diferente do anterior - assume anterior.
83	W	Campo resultado (já definido) com posições decimais inválido ou diferente do anterior - assume anterior.
84	W	Campo resultado (em definição) com posições decimais inválido - assume zero se numérico, assume branco se alfanumérico.
85	T	Campo resultado (em definição) com posições decimais inválido para operação - assume zero se numérico, e branco se alfanumérico.
86	T	Campo resultado (em definição) com tamnaho inválido - assume tamanho = 15 se numérico e tamanho = 255 se alfanumérico.
87	T	Campo resultado (em definição) com tamanho maior que limite - assume tamanho = 15 para numérico, e tamnaho = 255 para alfanumérico.

Nº	TIPO	MENSAGEM
88	T	Campo resultado com posições decimais maior do que tamanho - assume posições decimais = tamanho.
89	W	Campo resultado - entrada inválida para ajuste -assume 'H'.
90	W	Campo resultado - operação não permite meio ajuste - assume branco.
91	T	Fator 1 ou fator 2 não definidos anteriormente.
92	T	Fator 1 ou fator 2 possui tipo não permitido para a operação.
93	T	Fator 1 ou fator 2 com entrada inválida.
94	T	Operação MVR não segue operação DIV.
95	T	Campo resultado - nome esquecido.
96	T	Fator 1 - nome esquecido.
97	T	Fator 2 - nome esquecido.
98	T	Fator 1 inválido para operação.
99	T	Fator 2 invalido para operação.
100	T	Indicador resultado inválido para operação.
101	T	Campo resultado e meio ajuste invalidos para operação.
102	T	Tipo fator 1 diferente tipo fator 2 para operação.
103	T	Indicador de resultado esquecido para operação.
104	T	Subrotina chamando ela mesmo.
105	T	GOTO fora de limites.
106	T	Subrotina definida dentro de subrotina.
107	T	Nome inválido para rótulo (GOTO - SUBROTINA).
108	T	Nesta operação indicador de cálculo deve estar branco.
109	T	Rótulo em fator 1 já usado em outra operação.
110	T	Operação BEGSR não precede operação ENDSR.
111	T	Rótulo usado em outra operação TAG ou ENDSR.
112		RESERVA - ATÉ Nº 149

4. ESPECIFICAÇÕES DE SAÍDA

Nº	TIPO	MENSAGEM
150	T	Especificação de campo não segue especificação de registro.
151	T	Arquivo não é de saída nem de atualização ou especificado mais de uma vez.
152		Carater inválido na coluna 15 (TIPO DE LINHA) - assume de talhe (D).

Nº	TIPO	MENSAGEM
153	W	Faltou especificação de campo de saída.
154	W	Entrada inválida colunas 7-14 para especificação de registro - assume branco.
155	T	Nome de arquivo de saída ou atualização não especificado no formulário de descrição de arquivos.
156	W	RESERVA
157	W	Linha 'AND' com especificação de space/skip - assume space/skip = branco.
158		RESERVA
159	W	Entrada inválida para space - assume branco.
160	W	Entrada inválida para skip - assume branco.
161	T	Indicador de saída não permitido ou inválido.
162	W	Não há indicador de saída.
163	T	Indicador de saída não definido anteriormente.
164	W	Entrada inválida para colunas 23-26-29 (NOT) - assume 'N'.
165	T	Indicador de overflow não especificado na descrição de arquivos.
166	T	Indicador de overflow especificado para linha EXCEPT.
167	T	Indicador de 1ª página (1P), especificado em linha diferente de cabeçalho/detalhe.
168	W	Especificação de campo na mesma linha de especificação do registro - ignora especificação de campo.
169	T	Linha AND/OR fora de ordem.
170	W	Linha AND/OR com nome de arquivo especificado - ignora nome de arquivo.
171	T	Linha AND/OR sem indicadores especificados na linha anterior.
172	T	Linha AND/OR sem especificação de indicadores.
173	T	Faltou especificação de campo.
174	T	Especificação de campo sem nome de campo e sem constante.
175	W	Especificação de constante com branco/depois ou com especificação de código de edição - assume branco.
176	T	Nome de campo inválido.
177	T	Nome de campo não definido anteriormente.
178	T	Código de edição diferente de branco ou inválido.

Nº	TIPO	MENSAGEM
179	T	Foi especificado código de edição e mascara de edição para um campo numérico.
180	W	Branco/depois inválido - assume 'B'.
181	T	Entrada inválida para posição final de campo.
182	T	Posição final de campo maior do que tamanho deregistro, ou posição final menor do que o tamanho de campo.
183	T	Nome de arquivo inválido.
184	W	Arquivo para impressora sem especificação space/skip - assume space-after = 1
185	W	Código de edição ou palavra de edição especificado para campo alfanumérico - assume branco.

5. OUTRAS MENSAGENS

Nº	TIPO	MENSAGEM
186	T	Esgotou área de geração do código intermediário.
187	T	Esgotou tabela de símbolos.
188	T	Esgotou tabela de arquivos lógicos ou tabela de símbolos externos.
189	T	Esperado registro fonte do tipo 'F' - interrompe análise.
190	T	Esperado registro fonte do tipo 'I' - interrompe análise.
191	W	Adverte - não há especificação de cálculo.
192	T	Esperado registro fonte tipo 'O' - interrompe análise.
193	T	Número de indicadores match diferente do número de indicadores definidos no primeiro registro com campos match.