GERADOR DE COMPILADORES DE MICROASSEMBLER

FREDERICO JOSÉ BANDEIRA DE MELLO E NOVAES

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

Aprovada por:

José Lucas Mourão Rangel Netto

JOSÉ LUCAS MOURÃO RANGEL NETTO
(Presidente)

Edil Severiano Tavares Fernandes

EDIL SEVERIANO TAVARES FERNANDES

Eduardo Lessa Peixoto de Azevedo

EDUARDO LESSA PEIXOTO DE AZEVEDO

Rio de Janeiro, RJ, BRASIL

AGOSTO DE 1982

NOVAES, FREDERICO J. B. M.

GERADOR DE COMPILADORES DE MICROASSEMBLER - RJ - 1982

iii, 76p. (COPPE - UFRJ, M. Sc., Engenharia de Sistemas, 1982)

TESE - UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

1. Teoria dos Compiladores I. COPPE/UFRJ II.Título(Série)

Sinopse

Essa tese apresenta o projeto e a implementação de um sistema gerador de compiladores para montar microlinguagens, voltados para o uso de microprogramação.

O projeto nasceu da necessidade de uma ferramenta geral de microprogramação que servisse tanto a engenheiros de hardware quanto de software.

O sistema foi totalmente implantado no computador COBRA-5ØØ e programado na linguagem LPS.

Abstracts

This thesis presents the design and implementation of a system which generates compilers to assembly microprogramming languages.

The aim of the project was to create a general tool for either software or hardware engineers to write their microprograms.

The system is already available in the COBRA-5ØØ computer, and it was programmed in LPS.

ÍNDICE

CAPÍTULO I - INTRODUÇÃO

1. DESCRIÇÃO DO PROBLEMA

Com o advento das memórias de leitura exclusiva (ROM's), a utilização de microprogramação tornou-se uma ferramenta básica de projetos de hardware de computadores com certo nível de sofisticação.

A estrutura do microprograma é definida de acordo com o módulo da arquitetura do computador que a caracteriza - unidade central de processamento, canais de controle, etc - e, evidentemente, varia de máquina para máquina.

Existem dois métodos de tradução para o microprograma: o primeiro exige o trabalho meticuloso do engenheiro de hardware que manipula diretamente a informação binária, naturalmente sujeito a vários tipos de erro; o segundo através de um projeto de software, o tradutor de micro-assembler, que permite ao engenheiro de hardware trabalhar com os mnemonicos dessa linguagem, o que lhe dá ganhos em tempo, eficiencia e correção de erros.

Na COBRA tem-se experiência em ambos os métodos de tradução citados. Ficou comprovado no caso da unidade central de processamento do COBRA-5ØØ as vantagens do método de tradução a partir do projeto de software.

A necessária diversificação da linha 5ØØ, com a previsão da utilização de variados microassembler, levou à definição desse projeto. Assim, obtem-se um instrumento geral para a área de microprogramação.

Duas características marcantes do montador de microprograma o distingue de outros montadores : o formato multi-campo redefinível do código objeto e as necessidades especiais do usuário desse montador.

Um segmento de microprograma deve especificar muitas coisas : a seqüência do fluxo de controle do microprograma, códigos de controle para a unidade lógica aritmética (ULA), endereços de registradores, condições de controle de relógio e de inibições para chaveamento, constantes de bits para comparações ou para máscaras, etc.

A microinstrução típica, portanto, é um padrão de bits de muitos campos. Cada campo, ou microordem, tem um comprimento em bits.

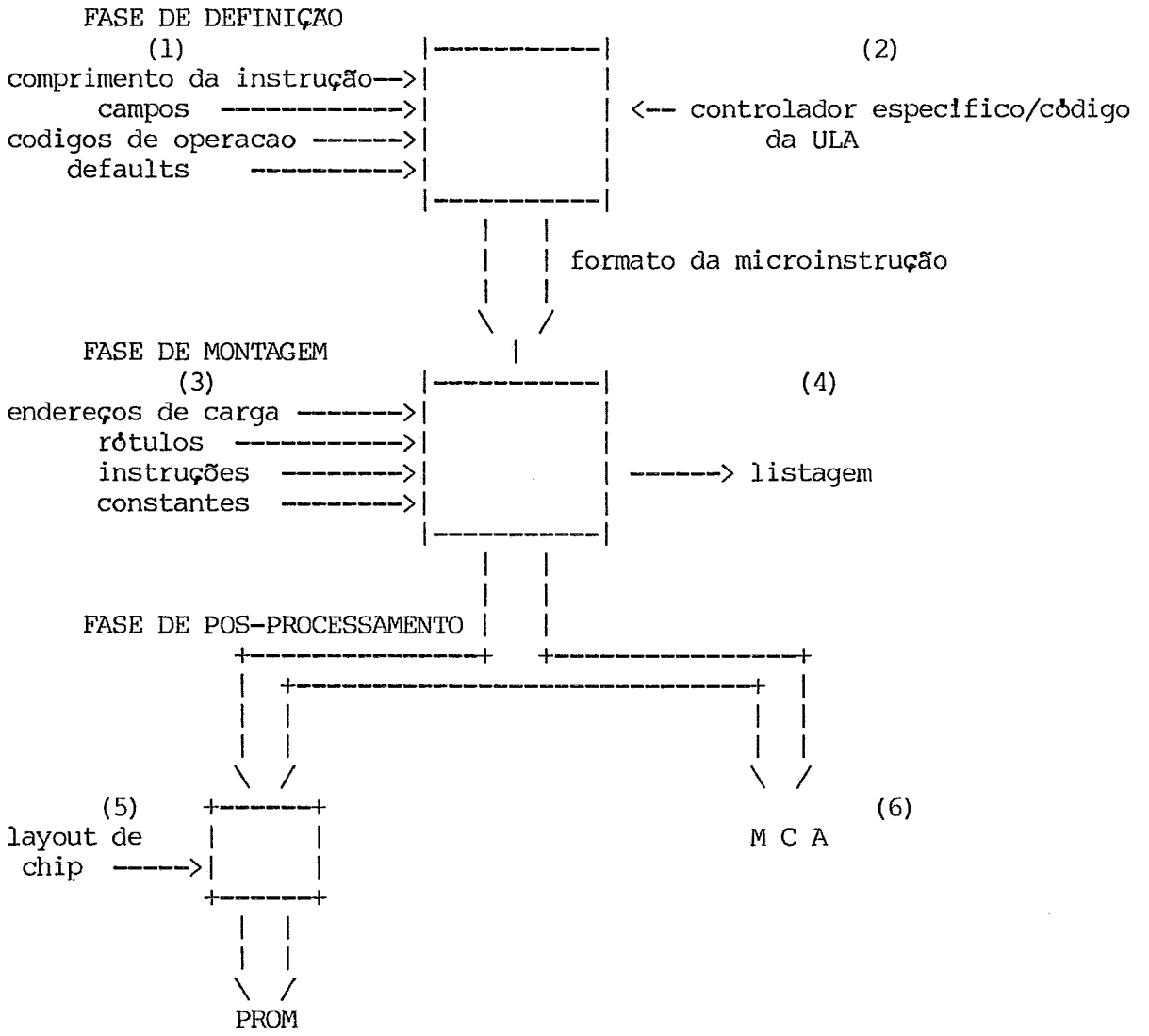
Um microprograma típico para microprocessador é aprontado em tres passos - conforme figura. Cada passo requer etapas de processamento. Na fase de definição , o formato da microinstrução é definido com seus campos e códigos de operação conforme especificação do projetista (1). Nessa fase o montador faz uso de definições relativamente fixas da familia de chips usada (2).

Na fase de montagem , o projetista entra com a "linguagem de montagem" (3) para gerar um programa de microinstruções, selecionando seqüências de códigos de operação e valores. Nessa fase obtem-se a listagem do microprograma montado (4).

A fase de pós-processamento reformata o código objeto binário para seu uso. O uso pode ser a implementação em PROM (5) ou a simulação em memória de controle alterável (6).

CAPÍTULO I - INTRODUÇÃO
1. DESCRIÇÃO DO PROBLEMA

FIGURA



CAPÍTULO I - INTRODUÇÃO
2. ORGANIZAÇÃO DA TESE

Os capítulos em que se divide o presente trabalho apresentam a seguinte estrutura:

.CAPÍTULO II - FUNDAMENTOS TEÓRICOS

Apresenta uma rápida visão sobre a estrutura de um compilador e as etapas em que é geralmente esquematizado. Todos os algoritmos utilizados nas diversas etapas são aí descritos.

.CAPÍTULO III - PROJETO GERAL DO SISTEMA

Apresenta os conceitos básicos de microprogramação seguindo a linha teórica proposta no livro Foundations of Microprogramming - Architecture, Software and Applications dos autores A. K. Agrawala e T. G. Rauscher. Compara as duas formas usuais de construção de microprogramas. Apresenta o esquema geral do Gerador de Compiladores de Microassembler a partir daí desenvolvido.

.CAPÍTULO IV - MANUAL DE LÓGICA

Descreve em detalhes a implementação dos dois módulos em que se divide o Gerador de Compiladores. Apresenta as principais rotinas de cada módulo.

.CAPÍTULO V - MANUAL DE UTILIZAÇÃO

Descreve a forma de uso de cada módulo, apresentando informações sobre o uso de constantes, identificadores, diretiva de execução e sintaxe da gramática de especificação de cada módulo.

.CAPÍTULO VI - CONCLUSÕES

Faz um balanço do uso prático desse trabalho e apresenta possíveis extensões.

.ANEXOS

A bibliografia usada, as listagens dos módulos e um exemplo de utilização compõem os anexos.

CAPÍTULO II - FUNDAMENTOS TEÓRICOS

1. ESTRUTURA DE UM COMPILADOR

Um compilador tem como entrada um programa-fonte que se deseja traduzir para um programa-objeto em instruções de máquina, caracterizado como seu produto de saída. Esse processo de tradução é tão complexo que é tratado como uma série de subprocessos chamados fases.

A primeira fase é a de análise léxica, que divide o texto-fonte em grupos de caracteres afins. Os identificadores são tratados como pares ordenados onde a primeira informação é o código inteiro que o caracteriza e a segunda pode ser um ponteiro para uma tabela. A esse grupo de caracteres assim identificado chama-se token. Os tokens geralmente encontrados são palavras reservadas, identificadores, números e símbolos de operação e pontuação.

A segunda fase é a de análise sintática que grupa os tokens em estruturas sintáticas. A estrutura sintática pode ser vista como uma árvore cujas folhas são os tokens.

A última fase, a de geração de código, produz o programa-objeto. Essa fase pode ser dividida em tres partes : a geração de um código intermediário que a partir do analisador sintático cria uma cadeia de instruções simples que não especifica ainda detalhes como por exemplo os registradores a serem usados em cada operação; a otimização de código, que é opcional, cuja saída tambem é um código intermediário tentativamente aperfeiçoado em tempo e/ou espaço; e a geração de código propriamente dita que decide alocação de memória para dados, seleciona código para acessar dados e seleciona registros de trabalho.

Embora não constituam fases de um compilador, duas ferramentas dessa estrutura devem ser ressaltadas : a gerência da tabela de símbolos que armazena os dados (identificadores, números reais ou inteiros,palavras reservadas) e a manipulação de erros, que comunica ao programador o diagnóstico dos erros encontrados e orienta sua correção.

A divisão do programa compilador em fases não necessàriamente implica que cada fase deve ocorrer após o término da anterior. Essa divisão é para fins de estudo e projeto. A divisão em passos do compilador pode ser a mais conveniente para fins de implementação.

CAPÍTULO II - FUNDAMENTOS TEÓRICOS
2. ANÁLISE LÉXICA COM AUTÔMATO FINITO

Assim como um reconhecedor de uma linguagem L é um programa que a partir de uma cadeia de entrada responde se ela é uma sentença de L ou não, a parte do analisador léxico que identifica a existencia de tokens é um reconhecedor da linguagem que define os tokens.

Expressões regulares são a notação natural para descrever a linguagem dos tokens.

Autômato finito é um diagrama de transição, com símbolos terminais apenas, para linguagens regulares, isto é, aquelas que podem ser definidas por expressões regulares.

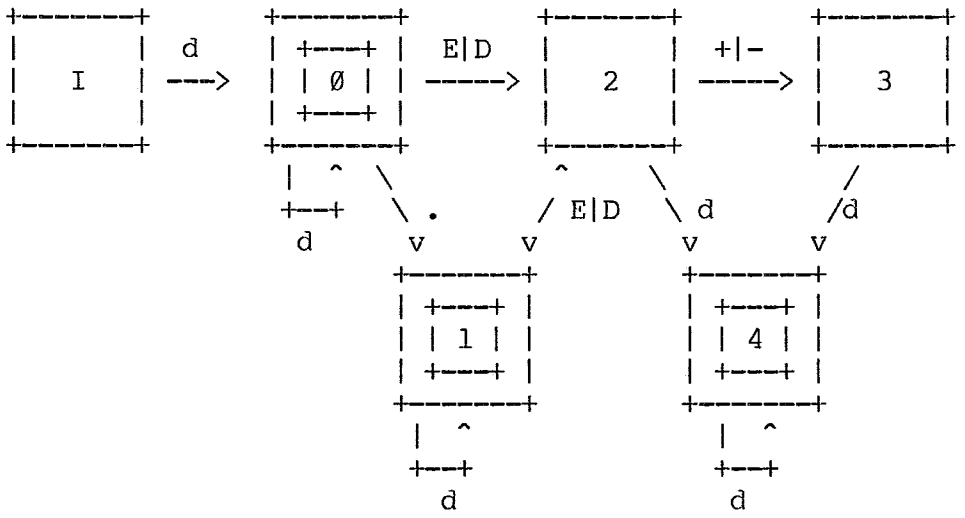
A seguir apresentamos um exemplo do uso de autômato finito na descrição dos números inteiros ou reais conforme sua definição na linguagem LPS (conforme consta na referência 8 da Bibliografia).

A expressão regular que define os números em LPS, conforme a notação de Viena, é:

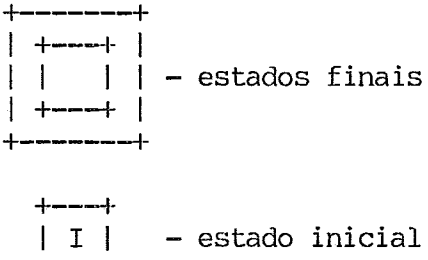
```
                 ...      ...              ...
NUM={d}    [.[d]    ][{E|D}[+|-]d    ]
```

onde d = Ø|1|...|9; E indica precisão simples e D precisão dupla.

Automato Finito definido a partir dessa sentença é expresso da seguinte forma:



Nesse diagrama são adotadas as convenções:



CAPÍTULO II - FUNDAMENTOS TEÓRICOS
2. ANÁLISE LÉXICA COM AUTÔMATO FINITO

Todas as situações não previstas são consideradas como de erro léxico - por exemplo o aparecimento no estado Ø do caracter "+".

CAPÍTULO II - FUNDAMENTOS TEÓRICOS
3. ANÁLISE SINTÁTICA DESCENDENTE

Para gramáticas não ambíguas, a seqüência de producoes usadas para derivar qualquer sentença é única e determina a estrutura sintática dessa sentença. São definidos dois tipos de derivação associados ao não terminal a ser expandido: derivação à direita e à esquerda.

A análise sintática verifica se existe uma derivação na gramática que corresponda à sentença desejada, tendo como partida o simbolo inicial dessa gramática. Essa verificação pode basear-se num dos dois tipos de derivação, caracterizando-se nos dois métodos de análise : DESCENDENTE, baseado na derivação mais à esquerda e ASCENDENTE, baseado na derivação mais à direita, considerada na ordem inversa.

Tomemos como exemplo a gramática G1({E,T,F},{a,+,*,(,)},P,E), cujas produções são assim definidas:

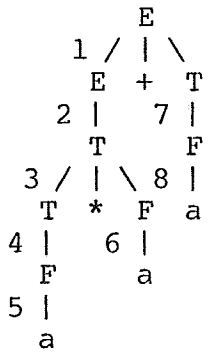
```
P :  1. E => E + T
     2. E => T
     3. T => T * F
     4. T => F
     5. F => (E)
     6. F => a
```

Seja a sentenca S1 : a*a+a.

A estrutura sintática de S1 pode ser determinada através das produções utilizadas numa derivação mais à esquerda :

E=>E+T=>T+T=>T*F+T=>F*F+T=>a*F+T=>a*a+T=>a*a+F=>a*a+a

A árvore de derivação para S1 (com a ordem de uso das produções indicada) é:



O resultado da análise sintática descendente de uma sentença é portanto a seqüência de produções que a geram ou a seqüência de configurações da árvore de derivação.

Abordaremos nesse trabalho o método LL(1) de análise sintática descendente, que é na verdade muito semelhante à representação tabular do método de descida recursiva.

CAPÍTULO II - FUNDAMENTOS TEÓRICOS

3. ANÁLISE SINTÁTICA DESCENDENTE

3.1. Método LL(1)

Pode ser mostrado que uma gramática G é LL(1) se e apenas se para A => alfa | beta, duas produções distintas em G onde A é um símbolo não-terminal e alfa e beta duas seqüências de símbolos, são verdadeiras as seguintes condições :

1. Não exista símbolo terminal que inicie cadeias derivadas a partir de alfa e beta.

2. Pelo menos uma entre as cadeias alfa e beta pode derivar a cadeia vazia.

3. Se beta $\overset{*}{\Rightarrow}$ epsilon, onde epsilon significa a cadeia vazia, alfa não deriva nenhuma cadeia que comece com algum terminal pertencente a FOLLOW(A).

A função FOLLOW(A) é definida para um não-terminal A como o conjunto de terminais que podem aparecer imediatamente à direita de A em alguma forma sentencial. Se A for o símbolo mais à direita para alguma forma sentencial, deve ser incluído no conjunto de terminais de FOLLOW(A) o símbolo que indique fim de cadeia - nesse trabalho indicado por $.

O método LL(1) se utiliza de uma entrada com a sentença a ser analisada, uma pilha que contém a descrição do resto da sentença de entrada, uma tabela de análise preditiva e uma saída.

O não terminal que se encontra no topo da pilha representa (ou seja deve derivar) a parte inicial do trecho ainda não lido da seqüência de entrada; a parte já lida da seqüência de entrada corresponde aos não terminais já desempilhados anteriormente. A presença de um terminal na pilha indica que ele deve ocorrer na seqüência de entrada na posição correspondente.

O nome LL(1) provem de tres fatos : a análise léxica começar pela esquerda (LEFT) da seqüência de entrada; conduzir a análise de acordo com uma derivação mais à esquerda (LEFTMOST) e decidir baseado na análise de 1 símbolo de entrada.

CAPÍTULO II - FUNDAMENTOS TEÓRICOS
3. ANÁLISE SINTÁTICA DESCENDENTE
3.2. Algoritmo de Construção da Tabela T de Análise Preditiva

O analisador LL(1) compõe-se de um algoritmo padrão, cujo funcionamento segue uma tabela T (tabela de análise preditiva) que para cada simbolo não-terminal (do topo da pilha) e para cada símbolo de entrada indica a produção a ser utilizada.

Construção de T :

1. Para todas as produções A => alfa da gramática G, repita os passos 2 e 3.

2. Para cada terminal a pertencente a FIRST(alfa), coloque a produção A => alfa na entrada T[A,alfa] da tabela.

3. Se epsilon pertencer a FIRST(alfa), coloque A => alfa em T[A,b], onde b pertence a FOLLOW(A).

4. Cada entrada de T não preenchida corresponde a uma situação de erro.

A função FIRST(alfa) é o conjunto de terminais que começam cadeias derivadas a partir de alfa. Se alfa ==> epsilon, então epsilon pertence a FIRST(alfa).

Os algoritmos que definem os elementos das funções FIRST e FOLLOW estão descritos nas páginas 188 e 189 do livro The Principles of Compiler Design - AHO e ULLMAN.

CAPÍTULO II - FUNDAMENTOS TEÓRICOS
3. ANÁLISE SINTÁTICA DESCENDENTE
3.3. Algoritmo de Análise Preditiva

A análise preditiva funciona a partir do símbolo X que se encontra no topo da pilha e do terminal a que é o atual símbolo da sentença de entrada. Esses dois símbolos determinam uma das tres seguintes ações :

1. Se X=a=$: o analisador para e anuncia fim de análise.

2. Se X=a $: X é retirado da pilha e é lido o próximo símbolo de entrada.

3. Se X é não terminal : consulta a tabela T de análise preditiva, podendo esta entrada de T ser de 2 tipos :

   i. T[X,a]={X=>U V W} - substitui-se X na pilha por W V U (U no topo).
   ii. T[X,a]=ERRO - é chamada a rotina de recuperação de erros.

Trabalhemos em cima de um exemplo a partir da gramática G1 do exemplo anterior, modificada para torná-la LL(1).

1. E => T E1
2. E1 => + E
3. E1 => epsilon
4. T => F T1
5. T1 => * T
6. T1 => epsilon
7. F => (E)
8. F => a

Tabela de análise preditiva :

	a	(	)	+	*	$
E	1	1				
T	4	4				
F	8	7				
T1			6	6	5	6
E1			3	2		3

CAPÍTULO II - FUNDAMENTOS TEÓRICOS
3. ANÁLISE SINTÁTICA DESCENDENTE
3.3. Algoritmo de Análise Preditiva

Analisemos a sentença a*(a+a) :

PILHA	ENTRADA	PRODUÇÃO USADA	ATITUDE DO ALGORITMO
$E	a*(a+a)$	1	3.i
$E1 T	a*(a+a)$	4	3.i
$E1 T1 F	a*(a+a)$	8	3.i
$E1 T1 a	a*(a+a)$	-	2
$E1 T1	*(a+a)$	5	3.i
$E1 T *	*(a+a)$	-	2
$E1 T	(a+a)$	4	3.i
$E1 T1 F	(a+a)$	7	3.i
$E1 T1)E(	(a+a)$	-	2
$E1 T1)E	a+a)$	1	3.i
$E1 T1)E1 T	a+a)$	4	3.i
$E1 T1)E1 T1 F	a+a)$	8	3.i
$E1 T1)E1 T1 a	a+a)$	-	2
$E1 T1)E1 T1	+a)$	6	3.i
$E1 T1)E1	+a)$	2	3.i
$E1 T1)E +	+a)$	-	2
$E1 T1)E1 T	a)$	4	3.i
$E1 T1)E1 T1 F	a)$	8	3.i
$E1 T1)E1 T1 a	a)$	-	2
$E1 T1)E1 T1	)$	6	3.i
$E1 T1)E1	)$	3	2
$E1 T1)	)$	-	3.i
$E1 T1	$	6	1
$E1	$	3	
$	$	-	

Aqui o analisador anuncia fim de sentença de entrada sem erro e fornece a seqüência de produções usadas para obter a sentença S1 a partir do símbolo inicial da gramática com derivações mais à esquerda.

CAPÍTULO II - FUNDAMENTOS TEÓRICOS

4. ANÁLISE SINTÁTICA DEPENDENTE DE CONTEXTO E SÍNTESE SEMÂNTICA

Existem convenções sintáticas que não podem ser expressas na gramática livre-de-contexto da linguagem de programação. Os exemplos mais comuns dessas convenções são:

- Nenhum identificador pode ser declarado mais de uma vez;

- Todas as variáveis devem ser definidas antes do uso;

- Os argumentos de uma chamada de função devem ser compatíveis tanto em número quanto em atributos com a definição da função.

Um compilador, para observar se regras desse tipo estão sendo obedecidas, necessita do que se convencionou chamar de análise sintática dependente de contexto.

Semântica de uma linguagem de programação é um conjunto de regras que dão significado aos programas escritos nessa linguagem. Essas regras associam estruturas sintáticas a objetos que representem o significado dessas estruturas. São exemplos de ação representadas nessas regras:

- A abertura de espaço em memória para variáveis;

- A execução de instruções nas linguagens interpretadas;

- A geração de uma tabela.

A síntese semântica gera a partir da análise sintática e dos atributos semânticos, herdados ou sintetizados, o produto-objeto da compilação.

Ainda não foi definido nenhum meio para especificar tanto a análise sintática dependente de contexto como a semântica de uma linguagem que ajude satisfatoriamente a construção de um compilador.

Em linguagens onde a análise sintática é feita através do método descendente podemos unir as duas fases acima descritas através da introdução de novos símbolos na sua gramática que indiquem pontos de chamada de rotinas de ação.

Esses símbolos, aqui chamados símbolos de ação, são colocados:

1 - No início da produção, quando ela é escolhida pelo algoritmo;

2 - Entre os símbolos da produção, bastando inserir um indicador de chamada de rotina entre os símbolos correspondentes, ao empilhar o lado direito da produção.

CAPÍTULO III - PROJETO GERAL DO SISTEMA

1. MICROPROGRAMAÇÃO

Microprogramação é tradicionalmente um método alternativo de implementação de instruções de linguagem de máquina num computador, podendo reduzir a complexidade e a inflexibilidade de sua unidade de controle. Baseia-se na observação que uma operação complexa é completamente definida por uma seqüência de operações primitivas e que unidades de memória podem ser usadas para armazenar informações sobre operações primitivas. Cada operação primitiva é representada na memória como uma microordem. A unidade de memória que contém as microordens é chamada de memória de controle ou memória de microprograma.

Executar uma microordem significa a busca dessa instrução pela unidade de controle e a geração dos sinais de controles necessários à ativação da operação primitiva relativa a essa microordem.

Embora microprogramas contenham informações que controlam o hardware a um nível primitivo, eles são armazenados na memória e são executados como programas. Isso dá a microprogramação um nível de software tanto quanto de hardware - daí o termo firmware, que designa microprogramas residentes na memória de controle do computador.

Embora ainda a microprogramação seja um tipo de programação vale enfatizar alguns aspectos que a distinguem do conceito comum de programação:

1 - O nível de controle - microinstruções exercem controle direto sobre recursos primitivos de hardware, enquanto programas manipulam estruturas de dados definidos para usuário.

2 - O repertório de instruções - o conjunto de microordens é geralmente pequeno, enquanto o conjunto de operações de programa pode ser grande.

3 - Paralelismo - a máquina permite execução simultânea de várias microordens.

4 - Memória de controle - microprogramas são geralmente armazenados em unidades de memória especiais que são muito rápidas e caras, logo representam um recurso limitado.

O desenvolvimento de memórias rápidas de leitura e escrita (RAM's) fizeram com que a microprogramação deixasse de ser apenas instrumento para a área de hardware sendo utilizada também pelos projetistas de software para, por exemplo, partes do sistema operacional normalmente muito executados em ambientes multi-usuários.

CAPÍTULO III - PROJETO GERAL DO SISTEMA

2. CONSTRUÇÃO DE MICROPROGRAMAS

Antes de microprogramar um computador para uma aplicação especifica, deve-se conhecer a arquitetura desse computador. Armado com tal informação, parte-se para a definição da microinstrução com as microordens que a constituem. Cada microordem é associada a um conjunto de mnemonicos.

As próximas fases são de escrita na linguagem microassembler, formada pelos mnemônicos já definidos, e tradução dessa linguagem para a representação binária que a máquina possa interpretar.

As etapas finais são de documentação e de inevitáveis alterações que sempre se fazem necessárias, sejam por erro de avaliação sejam por readequação de projeto ou inclusao de novas características.

Caso a tradução para a representação binária seja manual, a probabilidade de erros aumenta consideravelmente em função do fator humano assim como o descompasso com a documentação. A necessidade de alterações torna mais pesado esse trabalho principalmente por causa da mudança dos endereços que provoca a introdução de novas microinstruções.

A tradução por software para a representação binária cria facilidades de escrita, geração e depuração de microprogramas. Cria também, através do uso de rótulos no micro-programa, a facilidade de relocação automática de novas microinstruções. Como argumento final de sua maior utilidade há a geração automática da documentação.

CAPÍTULO III - PROJETO GERAL DO SISTEMA

3. A ESPECIFICAÇÃO DO MICROASSEMBLER

O projeto do microassembler começa pela especificação da informação necessária para controlar os recursos de hardware e como essa informação é montada numa palavra da microinstrução.

De início deve ser determinado o número de recursos de hardware que cada microinstrução controla; ou seja, o número de microordens que cada microinstrução pode executar. Aí fica também determinado o número de bits que compõe a palavra da microinstrução.

No projeto de microassembler mais simples possível, não há codificação na representação da microinstrução: cada bit representa uma microordem.

Num projeto de codificação direta, as microordens que acessam um recurso qualquer de hardware são representadas por um campo de bits em vez de bits individuais. Por exemplo, consideremos uma unidade de lógica aritmética (ULA) que faça oito diferentes operações aritméticas e lógicas. Se não houver codificação, oito bits são necess;arios para especificar cada uma das operações. Entretanto, uma microinstrução só necessita uma das operações da ULA a cada vez; logo num projeto de codificação direta cada operação pode ser especificada com tres bits.

Num projeto de codificação de dois níveis, alem de microordens mutuamente exclusivas serem combinadas para formar campos, o significado de um campo não depende apenas do seu valor mas também do valor contido geralmente em outro campo de microinstrução.

Durante a execução de uma microinstrução é necessário determinar o endereço da próxima microinstrução a ser executada. Existem básicamente duas técnicas de seqüenciamento de microinstrução: na primeira, o endereço de controle da próxima microinstrução a ser executada é igual ao da microinstrução sendo executada mais um; na outra, cada microinstrução contém um campo com o endereço de controle da próxima microinstrução a ser executada.

CAPÍTULO III - PROJETO GERAL DO SISTEMA

4. DEFINIÇÃO DESSE PROJETO

Esse projeto foi definido a partir de reuniões com projetistas de hardware da COBRA, que manifestaram suas necessidades e reuniões com o orientador da tese, onde se apontava formas práticas para a concretização dos objetivos.

Nas reuniões com os projetistas de hardware ficaram definidas as principais bases do projeto:

1 - A representação da microinstrução deve aceitar codificação de dois níveis.

2 - O número máximo de mnemônicos de uma microordem é 256.

3 - O uso de constantes em substituição a conjunto de microordens, podendo mesmo substituir toda a microinstrução.

4 - O número máximo de bits de uma microordem é 16.

5 - Tamanho máximo da palavra de memória de controle: 1 word = 32.767

Pretende-se posteriormente expandir o projeto de forma a incluir:

1 - Crítica em tempo de montagem de combinações mutuamente exclusivas em uma mesma microinstrução.

2 - Implementação de macro-expansores que facilitem o uso para projetistas de software básico.

Durante o projeto foram decididos detalhes de implementação tais como uso de análise sintática LL(1), uso de função de hash para preenchimento e consulta da tabela de símbolos, definição precisa do método de análise léxica a ser usada.

CAPÍTULO III - PROJETO GERAL DO SISTEMA

5. ESTRUTURA GERAL DO GERADOR DE COMPILADORES DE MICROASSEMBLER

O trabalho aqui desenvolvido está dividido em duas partes: a primeira opera sobre a especificação do microassembler - informações sobre o número de bits da palavra da microinstrução, sobre o número de bits de cada microordem e seus mnemonicos associados, sobre o default de preenchimento de cada microordem, sobre o tipo de codificação usado (direta ou de dois níveis), sobre o campo da microinstrução reservado para endereçamento - gerando uma tabela com as informações necessárias à segunda parte do trabalho que é a fase de montagem.

A fase de montagem opera com duas entradas: a tabela acima descrita e o microprograma-fonte a ser montado. A saída dessa fase depende do usuário: se ele a deseja para projetos de hardware, a saída é a transcrição binária dos mnemonicos e endereços no microprograma-fonte; se ele a deseja para projetos de software, a saída é um módulo-objeto a ser relocado e ligado com outros módulos-objeto do seu sistema.

Ambas as fases acima descritas foram programadas na linguagem LPS. A linguagem LPS - projetada para o desenvolvimento de software dos produtos da COBRA - é estruturada e de alto nível. Permite, porém, ao programador usar trechos em assembler, o que a caracteriza como uma linguagem híbrida.

Essas duas fases foram implementadas no computador Cobra 53Ø. Os sistemas da série Cobra 5ØØ são computadores que podem assumir configurações de mini e médio porte. Sua unidade central de processamento é microprogramada e é implementada em suas funções principais através da tecnologia bipolar do tipo "LSI Bit-Slice". Possui capacidade de endereçar 1 Mbyte de memória principal. Seu sistema operacional é um conjunto de processos que permite ao usuário atuar em tempo compartilhado, processamento em lotes, entrada de dados e comunicação de dados. A alocação de memória é gerenciada por um processo que tanto pode atuar através de alocação dinâmica quanto pela utilização de partições fixas.

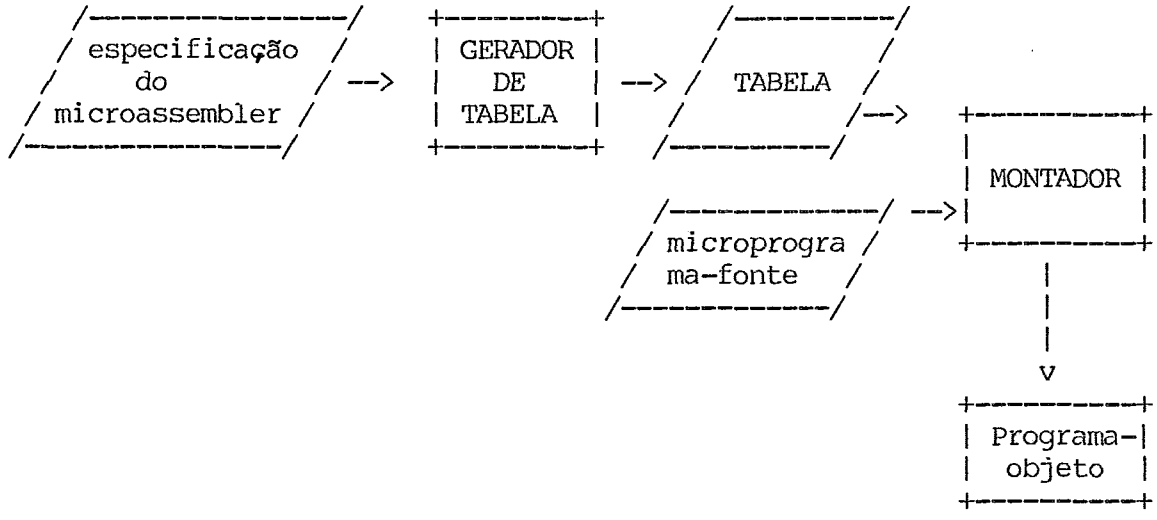
CAPITULO IV - MANUAL DE LÓGICA
1. INTRODUÇÃO

O GERADOR DE COMPILADORES está dividido em dois módulos: o GERADOR DE TABELA e o MONTADOR.

O GERADOR DE TABELA aceita como entrada a especificação do microassembler gerando como saída a tabela relativa a esse microassembler.

O MONTADOR possui duas entradas: a tabela do microassembler e o microprograma-fonte. A saída dessa fase será o programa-objeto a ser utilizado na gravação de PROM'S ou o módulo-objeto a ser posteriormente link-editado.

O esquema do GERADOR DE COMPILADORES é o seguinte:



A ativação de cada módulo é definido no respectivo manual de utilização.

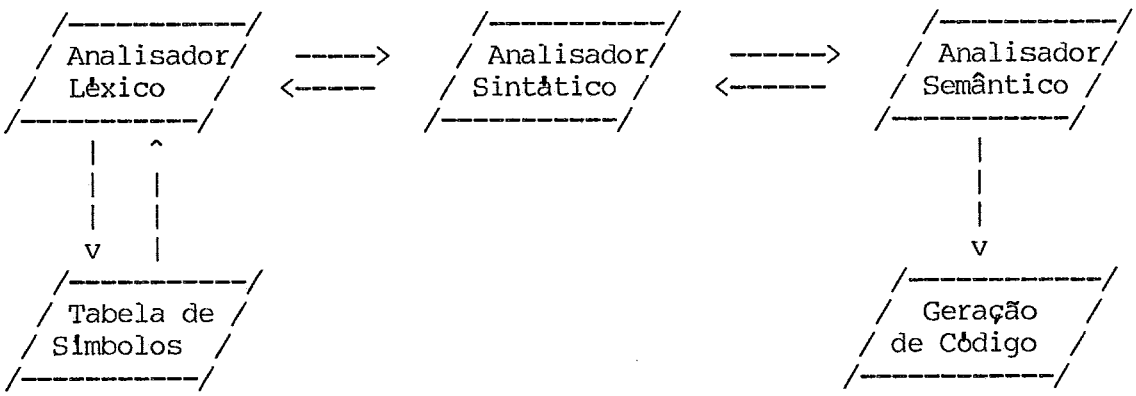
CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.1 Introdução

A entrada desse módulo é um programa escrito numa linguagem especialmente definida para especificar as caracteristicas de qualquer microassembler. Logo esse módulo e um compilador cujo código gerado é a tabela a ser usada na fase de montagem.

Em função da simplicidade da linguagem, que é do tipo PASCAL, foi escolhido como método para análise sintática a técnica LL(1).

O algoritmo do analisador sintático preditivo conforme consta de Principles of Compiler Design, AHO-ULLMAN, sec.5.5 - é usado como o nucleo que comanda esse módulo. Esse nucleo controla tanto o analisador léxico, que gerencia a tabela de símbolos, quanto o analisador semantico, que gera o codigo.

ESQUEMA DE FUNCIONAMENTO:



CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.2 Analisador Sintático

A partir da gramática definida no Manual de Utilização, constroi-se uma nova gramática compactada de forma a facilitar o trabalho do analisador sintatico. Monta-se entao a tabela LL(1) que sera consultada para a A pilha de simbolos e preenchida inicialmente co primeira produção da gramática, que será substituido por outras produções conforme a indicação da tabela LL(1).

A cadeia de símbolos terminais de entrada é uma sequencia de tokens, reconhecidos pelo analisador léxico.

2.2.1 Gramática compactada.

```
Ø     S -> DV maquina id: tam = dec, enc = LIT, campos DC fim $

1,2   DV -> constante id = N ; LDM DV1
           |DV1

3,4   N -> bin
          |hex

5,6   LDM -> id = N ; LDM
            |epsilon

7,8   DV1 -> tipo id : DTC ; LTC
            |epsilon

9,1Ø  DTC -> tam = dec , enc = LIT , ET
            |id

11,12 LIT -> id
            |N

13,14 ET -> literal
           |(id LM)

15,16 LM -> , id LM
           |epsilon

17,18 LTC -> id : DTC ; LTC
            |epsilon

19,2Ø DC -> id : pos = dec, DTC ; LC DC1
           |caso id : pos = dec, DTC de id : (id : pos : dec,
              DTC; LC) LF ; fim LC

21,22 LC -> id : pos = dec , DTC ; LC
           |epsilon

23,24 DC1 -> caso id : pos = dec, DTC de id : (id : pos = dec,
               DTC; LC); LF fim LC
            |epsilon
```

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.2 Analisador Sintático

25,26 LF -> id : (id : pos = dec, DTC; LC) ; LF
|epsilon

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.2 Analisador Sintático

2.2.2 Construção de Tabela LL(1).

2.2.2.1 Definição de FIRST e FOLLOW das Produções

Conforme os algoritmos de FIRST e FOLLOW constantes no livro Principles of Compiler Design de AHO-ULLMAN, será construída a seguinte tabela para os símbolos não-terminais da gramática compactada:

	FIRST	FOLLOW
DV	constante, epsilon, tipo	maquina
LDM	id,epsilon	tipo,maquina
N	bin,hex	;,,
DV1	tipo, epsilon	maquina
LIT	id,bin,hex	,
ET	literal,(	;,de
LM	,,epsilon	)
LTC	id,epsilon	maquina
DC	id,caso	fim

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.2 Analisador Sintático

DTC	id,tam	;,de
LC	id, epsilon	fim,),caso
DC1	caso, epsilon	fim
LF	id,epsilon	fim

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.2 Analisador Sintático

2.2.2.2 Tabela LL(1)

Gerada igualmente através de algoritmo constante do livro Principles of Compiler Dsign de AHO-ULLMAN.

	Ø	1	2	3	4	5	6	7	8	9	1Ø	11	12	13
	!NT!	!	!	!	!	!	!	!	!	!	!	!	!	!
	!TE!CON!TIP!MAQ!	I	!	B	!	H	!LIT!	D	!CA	!	T	!	F	!	,	!	(	!	)	!
	!ER!STA!O	!UIN!	D	!	I	!	E	!ERA!	E	!SO	!	A	!	I	!		!		!	
	!RM!NTE!	!A	!	!	N	!	X	!L	!	!	!	M	!	M	!		!		!	
Ø	!DV ! 1 ! 2 ! 2 !	!	!	!	!	!	!	!	!	!	!	!	!	!
1	!LDM!	! 6 ! 6 ! 5 !	!	!	!	!	!	!	!	!	!	!	!	!
2	! N !	!	!	!	! 3 ! 4 !	!	!	!	!	!	!	!	!	!
3	!DV1!	! 7 ! 8 !	!	!	!	!	!	!	!	!	!	!	!	!
4	!LIT!	!	!	!11	!12	!12	!	!	!	!	!	!	!	!
5	!ET !	!	!	!	!	!	!13	!	!	!	!	!	!14	!
6	!LM !	!	!	!	!	!	!	!	!	!	!	!15	!	!16	!
7	!LTC!	!	!18	!17	!	!	!	!	!	!	!	!	!	!
8	!DC !	!	!	!19	!	!	!	!	!2Ø	!	!	!	!	!
9	!DTC!	!	!	!1Ø	!	!	!	!	!	! 9	!	!	!	!
1Ø	!LC !	!	!	!21	!	!	!	!	!22	!	!22	!	!	!22	!
11	!DC1!	!	!	!	!	!	!	!	!23	!	!24	!	!	!	!
12	!LF !	!	!	!25	!	!	!	!	!	!	!26	!	!	!	!

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.2 Analisador Sintático

2.2.3 - Pilha de Símbolos

É implementada através da criação de dois vetores: um armazena o topo das produções indicadas pela tabela LL(1)- no programa GERATAB essa variável chama-se PILHA. ENDER.SIMB-e o outro armazena o número de símbolos existentes no lado direito dessa produções - no programa chama-se NUM.SIMB.LADO.DIR. Considera-se topo de uma produção o seu símbolo inicial, contado da esquerda par a direita. Esses vetores admitem ate 287 posições que equivalem à soma do número máximo de mmenônicos num campo (256) com o número máximo de símbolos no lado direito de uma produção (31).

2.2.4 - Armazenamento das Produções de Gramática

As produções serão armazenadas em tabelas, onde cada produção está descrita conforme a seguinte expressão regular:

$$PROD = (NSIMB\ SIMB)^{*}$$

NSIMB - byte que indica o número de símbolos do lado direito da produção

SIMB = TIPOSIMB QUALSIMB

TIPOSIMB = TERM | NTERM

TERM = "Ø" - indica símbolo terminal

NTERM = "1" - indica símbolo não-terminal

$$QUALSIMB = BIT^{7}$$

BIT = "Ø"|"1"

QUALSIMB é uma informação fornecida pelo analisador léxico e corresponde à identificação de determinado token.

Seja por exemplo a produção 7 constante no item 2.2.1. O seu armazenamento será da forma:

```
BYTE(*)LD7.DV1=(6,TIPO,ID,DOIS.PT,DTC,PT.VG,LTC);
```

onde 6 representa o número de símbolos do lado direito da produção; TIPO, ID,DOIS.PT e PT.VG são símbolos terminais armazenados em bytes onde o bit mais significativo vale Ø; DTC e LTC são símbolos não-terminais igualmente armazenados em bytes mas com o bit mais significativo ligado.

Para facilitar o acesso às produções assim respresentadas, foi criado um vetor - chamado PONT.LADO.DIR - onde são armazenados os ponteiros para o inicio de cada produção.

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.3 Analisador Léxico

Esse segmento do GERADOR DE TABELA reconhece como entrada textos que obdecem à seguinte expressão regular:

```
                                        *
TEXTO = ((PAL!NUM) (BRANCOS!SU))

PAL = id |"maquina"|"tam"|"enc"|"campos"|"fim"|
      "constante"|"tipo"|"literal"|"pos"|"caso"|"de"

NUM = BIN | DEC | HEX
           +
BIN = $ BIT
         +
DEC = ARAB

ARAB = BIT|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"

BIT = "Ø"|"1"
             +
HEX = # HEXAD

HEXAD = ARAB|"A"|"B"|"C"|"D"|"E"|"F"
                +
BRANCOS = (" ")

SU = ":"|","|"="|";"|"("|")"|&"
```

A saída do analisador léxico dará informações sobre qual token foi reconhecido, o tamanho da cadeia e o endereço onde ela inicia, os dois últimos pela pilha de atributos.

No caso de identificador, será colocado tambem na pilha de atributos o seu endereço na tabela de símbolos.

No caso de numeros será colocado o valor armazenado em uma palavra (16 bits).

A tabela de símbolos é preenchida sequencialmente mas só é acessada através de uma tabela de ponteiros preenchida por uma função de HASH. (ver item 3.4.2)

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.3 Analisador Léxico

Cada entrada da tabela de símbolos ocupa 16 bytes divididos nos campos:

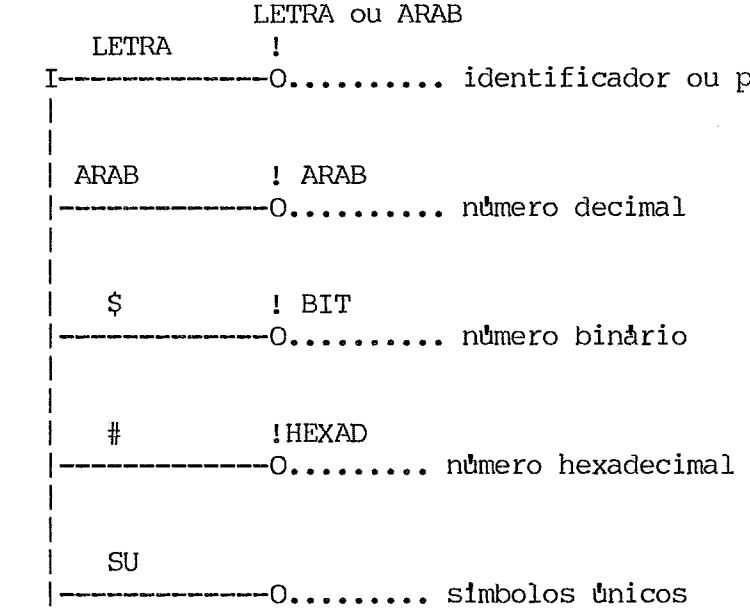
1 - Nome do identificador - 9 bytes.

2 - Controle da entrada - 1 byte - indica se o identificador é nome de mnemonico, de tipo ou de literal.

3 - Ponteiro para próxima entrada - 2 bytes - indica próxima entrada no caso de colisão na tabela de ponteiros.

4 - Os 4 bytes que faltam dependem do tipo de entrada:
- Se nome de mnemonico: 1 byte para o número de bits que o mnem onico possui, 2 bytes para a descrição dos bits.
- Se nome de tipo: 1 byte para tamanho do campo, 1 byte para default de preenchimento, 2 bytes para tecla auxiliar da lista de mnemonicos.

O autônomo finito, implementado através de um comando CASE, para interpretar a expressão regular definida acima possui a seguinte forma gráfica:



A classe de um token é, portanto, determinada pelo seu primeiro caracter. A sua implementação faz-se através da indexação do primeiro caracter do token num vetor que possui para cada caracter ASCII o valor da sua classe - LETRA = Ø, DECIMAL=1,BINARIO=2, HEXADECIMAL=3,SIMBOLOS UNICOS =4 e OUTROS=5.

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.4 Analisador Semântico

A análise semântica foi feita com a inclusão de pontos de chamada de rotinas de ação semântica na gramática compactada descrita no item 2.2.1. O analisador semântico controla a geração de código.

2.4.1 - Gramática Compactada com Símbolos de Ação Semântica.

```
S -> DV maquina id: tam = dec, enc = LIT {6}, campos DC fim $

DV -> constante id = N {Ø}; LDM DV1
     |DV1

N -> bin
    |hex

LDM -> id = N {Ø}; LDM
      |epsilon

DV1 -> tipo id {1}: DTC ; LTC
      |epsilon

DTC -> tam = dec , enc = LIT {3}, ET {5}
      |id {4}

LIT -> id {2}
      |N

ET -> literal
     |(id {12} LM)

LM -> , id {12} LM
     |epsilon

LTC -> id {1} : DTC ; LTC
      |epsilon

DC -> id : pos = dec {7}, DTC {8}; LC DC1
     |caso id : pos = dec {7}, DTC {9} de id {1Ø}: (id : pos : dec,
       {7} DTC {8}; LC); LF , fim {11} LC

LC -> id : pos = dec {7}, DTC {8}; LC
     |epsilon

DC1 -> caso id : pos = dec {7}, DTC {9} de id {1Ø}: (id : pos = dec{7},
       DTC {8}; LC); LF fim {11} LC
       |epsilon

LF -> id {1Ø} : (id : pos = dec {7}, DTC {8}; LC) ; LF
     |epsilon
```

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.4 Analisador Semântico

Os pontos de chamada de ação semântica (ou símbolos de ação semântica) são os números escritos, entre chaves. Cada número corresponde a uma determinada ação semântica - ver item 2.6.1Ø.

2.4.2 Código Gerado

Será uma tabela a ser armazenada em arquivo de organização seqüencial, onde cada entrada ou registro corresponde a um campo ou microordem, a menos do primeiro registro que guarda informações sobre a máquina como um todo.

O primeiro registro é composto pelas seguintes informações:

- nome da máquina - 9 bytes - corresponde ao <nome-máquina> - ver item 1.5.

- tamanho em bits da palavra a ser gerada - 2 bytes

- default de preenchimento do programa-objeto em endereços não definidos no microprograma-fonte - 1 byte

- números de número de campos possíveis - cada número de campos possíveis ocupa um byte - essa informação se justifica conforme o tipo de microprogramação for de codificação direta ou de dois níveis.

Os outros registros referem-se aos campos da microinstrução que podem ser fixos, condicionais, variáveis ou literais.

Todos os campos possuem formato identico até o byte 5 :

Posição do campo : 1 word - refere-se à posição inicial do campo na microinstrução.

Controle : 1 byte - indica se o campo é FIXO, CONDICIONAL, VARIAVEL ou LITERAL.

Tamanho : 1 byte - indica o número de bits ocupados por esse campo.

Default de preenchimento : 1 byte - refere-se ao valor de preenchimento quando essa microordem é deixada em branco no microprograma.

Se o campo for fixo ou condicional essa seqüência será seguida por 9*(2**TAM) bytes, onde serão armazenados os mnemônicos nos lugares relativos aos bits que representam. Se o campo for variável, haverá 2 bytes para receber a cadeia de bits do mnemonico que o sinaliza; depois teremos o mesmo do descrito acima. Campo literal não precisa de mais informações, pois será sempre preenchido por um endereço no microprograma passado através de constante ou rótulo.

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.5 Arquivos Utilizados

2.5.1 Arquivo de entrada

Contém o programa-fonte escrito na linguagem descrita no item 1.5 do MANUAL DE UTILIZAÇÃO.

2.5.2 Arquivo de saída

Contém a tabela necessária a montagem de qualquer microprograma referente a essa máquina. É criado durante a execução do GERADOR DE TABELA com o nome que designa a máquina. Essa informação e passada na linguagem de especifição do microassembler.

2.5.3 Arquivo de listagem

Contém a listagem do programa-fonte com cabeçalho, erros e outras informações.

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.6 Principais Rotinas

2.6.1 Introdução

Como já foi dito, o algoritmo de parser preditivo controla o programa como um todo (ver Fundamentos Teóricos). Antes da execução desse algoritmo, são chamadas rotinas que analisam as informações dos arquivos de entrada e saída, que empilham a produção inicial da gramática e iniciam a análise léxica devolvendo o primeiro token.

2.6.2 ANALISA.PARMS

Função:

Analise léxica dos parâmetros passados na diretiva de execução do GERATAB e o preenchimento de uma área com as informações passadas. Caso não tenha havido opção por passagem de parâmetros são assumidos os defaults.

Sintaxe dos parâmetros :

```
                              |<unidade-de-disco> |
  E : <nome-arq-entrada> [ <                    > ] ; ES ;
                              |V=<volume>        |
```

Rotinas Chamadas:

EXIJA
SIMBOLO
MOVE.NOME
AVANCA.PT
VOLUME
DISCO
ERRA.PARMS

2.6.3 ABRE.ARQ

Função:

Alocar e abrir arquivo de entrada. Preencher e imprimir cabeçalhos de página e de inicio de compilação. Abrir arquivo de listagem.

Rotina Chamada:

IMPR.CABEC

2.6.4 PUSH.LADO.DIR

Chamada:

PUSH.LADO.DIR (PRODUÇAO)
parâmetro PRODUÇAO - indica o número da produção a ser empilhada.

Função:

Empilhar os lados direitos das produções da gramática compactada na pilha

de simbolos. Reconhecer se a produção a ser empilhada é vazia. Controlar estouro de pilha de símbolos.

2.6.5 PEGA.PROX.TOKEN

Função:

Ler arquivo com programa-fonte, desprezando registros com comentários. Ao fim dessa rotina, é devolvido ao analisador sintático o próximo token a ser validado.

Rotinas Chamadas:

LE.ARQENT
ANALISADOR.LEXICO

Rotinas que a Chamam:

ANALISADOR.SINTATICO
ERRO

2.6.6 ANALISADOR.LEXICO

Função:

Após desprezar os caracteres brancos, divide em classes a cadeia de caracteres recebidos, implementando um autômato finito: identificador ou palavra reservada (classe Ø), número decimal (classe 1), número binário (classe 2), número hexadecimal (classe 3), símbolos únicos (classe 4), símbolos inesperados (classe 5). Em cada uma dessas categorias, preenche a pilha de atributos, gerenciando a tabela de símbolos no caso de identificadores.

Rotinas Chamadas:

COME.BRANCOS
LETRA
DIGITO
AVANCA.PT
ERRO
MONTA.CADEIA
HASH

Rotina que a Chama:

PEGA.PROX.TOKEN

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.6 Principais Rotinas

2.6.7 HASH

Função:

É uma word procedure que implementa uma função de hash e devolve o endereço de entrada na tabela de símbolos para o identificador.

A função de hash passa por tres fases:
- fase de ou-exclusivo - é feito ou-exclusivo entre words compostas pela associação de cada dois caracteres do identificador, sendo que de cada caracter é retirado o 1o. e 5o. bit mais significativo para então ser compactado em 6 bits e ser transcrito na word.
- fase de potência - a word resultante da fase anterior é levada ao quadrado e então são tomados os 1Ø bits menos significativos para entrada na tabela de símbolos.
- fase de colisão - em caso de colisao é preenchido o ponteiro da entrada na tabela de símbolos para a próxima entrada livre dessa tabela.

Rotina Chamada:

PREENCHE.TSEQ.SIMB

Rotina que a Chama:

ANALISADOR.LEXICO

2.6.8 POP.TOPO

Função:

Desempilhar o símbolo do topo da pilha de símbolos.

Rotinas que a chamam:

ANALISADOR.SINTATICO
ERRO
ACAO.SEMANTICA

2.6.9 ERRO

Chamada:

ERRO (NUM.ERRO)

Função:

Apontar onde e qual erro foi detectado. No caso de erro sintático, é adotada a técnica do panic mode. Contar número de erros e advertências.

Rotinas que a chamam:

SE.TRUNCADO
ANALISADOR.LEXICO

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.6 Principais Rotinas

ANALISADOR.SINTATICO
PREENCHE.LISTA.MNEM
ACAO.SEMANTICA

2.6.1Ø ACAO.SEMANTICA

Chamada:

AÇÃO.SEMANTICA (AÇÃO)
parametro AÇÃO - indica o número da ação semantica desejada.

Função:

Fazer as ações semanticas conforme gramática compactada modificada para análise dos atributos. São as seguintes:

Ação Ø: DV --> constante id = N {Ø}; LDM DV1
------ LDM --> id = N {Ø}; LDM

- Situação da Pilha de Atributos (P.A.):
  Ø : ID.ENDER (entrada na tab simb)
  1 : NUM DE BITS.
  2 : Valor do Binário ID.CONSTANTE

- Se ID.JA'.DECL, ERRO.

- Entrada é ID.CONSTANTE (CTL.CPO.TAB:=CPO.TAB.CTE).

- Armazenar na tab simb as informações da pilha.

- Esvaziar a pilha.

Ação 1: DV1 --> tipo id {1}: DTC; LTC

LTC --> ID {1} : DTC ; LTC

- Situação da P.A.:
  Ø : ID.ENDER

- Se ID.JA'.DECL, ERRO.

- Entrada é ID.TIPO (CTL.CPO.TAB:=CPO.TAB.TIP).

Ação 2: LIT --> id {2}

- Situação da P.A.:
  Ø : ID.ENDER de ID.TIPO
  1 : Valor do decimal referente a TAM.CPO
  2 : ID.ENDER de ID.CONSTANTE

CAPITULO IV - MANUAL DE LÓGICA
2. GERADOR DE TABELA
2.6 Principais Rotinas

- Se ID.NAO.DECL, ERRO.

- Se ID.NAO.CONSTANTE, ERRO.

- Substitui na P.A. ID.ENDER de ID.CONSTANTE pelas informações NUM DE BITS e valor binário referente a essa entrada na tab simb.

- Situação da P.A.:
  Ø : ID.ENDER de ID.TIPO
  1 : Valor do decimal referente a TAM.CPO
  2 : NUM DE BITS de ENC.CPO
  3 : Valor do binário referente a ENC.CPO

Ação 3: DTC --> tam = dec, enc = LIT {3}, ET {5}

- Situação da P.A.:
  Ø : ID.ENDER de ID.TIPO
  1 : Valor do decimal referente a TAM.CPO
  2 : NUM DE BITS de ENC.CPO
  3 : Valor do binário referente a ENC.CPO

- Armazena na tab simb, na entrada ID.ENDER, as informações constantes na P.A.

- Se TAM.DIFER.NUM.DE.BITS, ADVERTENCIA. Serão considerados os bits mais a direita de ENC.CPO

- Deixar na pilha apenas ID.ENDER.

Ação 4: DTC --> id {4}

- Situação da P.A.:
  Ø : ID.ENDER de ID.TIPO.
  1 : ID.ENDER de ID.TIPO já declarado anteriormente.

- Se ID.NAO.DECL, ERRO.

- Se ID.NAO.TIPO, ERRO.

- Se ID.AINDA.NAO.PREENCHIDO, ERRO.

- Mover informações da entrada da tab simb em P.A.(1) para P.A.(Ø).

- Esvaziar P.A.

Ação 5: DTC --> tam = dec, enc = LIT {3}, ET {5}

- Situação da P.A. se ID.TIPO for LITERAL:
  Ø : ID.ENDER de ID.TIPO.

- Situação da P.A. se ID.TIPO não LITERAL:

Ø : ID.ENDER de ID.TIPO.
1 : ID.ENDER de ID.CONSTANTE.
2 : ID.ENDER de ID.CONSTANTE.
.
.
.
N : ID.ENDER de ID.CONSTANTE.

- Se LITERAL, CTL.CPO.TAB := CPO.TAB.LIT

- Se não LITERAL, transcrever endereços de ID.CONSTANTE para a tabela auxiliar, passando a primeira posição dessa tabela símbolos (campo PONT.LISTA).

- Esvaziar P.A.

Ação 6: S --> DV máquina id: tam = dec, enc = LIT {6}, campos DC fim $

- Situação da P.A.:
  Ø : ID.ENDER de ID.MAQ.
  1 : Valor decimal de TAM.PAL
  2 : NUM DE BITS de ENC.PAL
  3 : Valor binario de ENC.PAL

- Se ID.JA'.DECL, ERRO.

- Transcrever para AREA.DE.CODIGO: ID.MAQ, TAM.PAL e ENC.PAL.

- Se TAM.PAL.DIFER.NUM.DE.BITS, ADVERTENCIA.

- Início da fase de geração de código.

- Criação do arquivo seqüêncial que guarda esse código.

- Gravação do primeiro registro do código com as informações em AREA.DE.CODIGO.

- Esvaziar P.A.

Ação 7: ... id: pos = dec {7}, DTC ...

- Situação da P.A.:
  Ø : ID.ENDER de ID.CPO.
  1 : Valor decimal da posição de ID.CPO na palavra

- Se ID.JA.DECL, ERRO.

- Entrada na tab simb é ID.TIPO (CTL.CPO.TAB:=CPO.TAB.TIP).

- Código gerado para campo fixo (CTL.CPO.COD:=CPO.COD.FIX).

- Desempilhar posição de ID.CPO passando para AREA.DE.CODIGO.

- Checar se essa posição é igual a do campo anterior mais o tamanho do campo anterior. Se não for, ERRO.

- POS.ANTERIOR:=POS.ID.CPO

- Situação da P.A.:
  Ø : ID.ENDER de ID.CPO.

Ação 8: ... id : pos = dec {7}, DTC {8}; LC ...

- P.A. vazia.

- Informações a serem passadas para AREA.DE.CODIGO estão em END.TAB (entrada para tab simb):
  TAM.CPO, ENC.CPO, se campo é LITERAL, se não LITERAL lista de mnemônicos constantes da tabela auxiliar.

- Se campo está dentro de um CASO (informação passada pela AÇÃO 1Ø), é preenchido a descrição binária que caracteriza o uso desse campo (MNEM.CAR) assim como o controle de campo é preenchido como variável (CTL.CPO.COD:=! or CPO.COD.VAR).

- Gravação de registro de campo de microinstrução no código com as informações da AREA.DE.CODIGO.

- TAM.ANTERIOR:=TAM.CPO.COD.

- Número de campos da microinstrução é incrementado.

Ação 9: ... id : pos : dec {7}, DTC {9} de id {1Ø} ...

- Repete Ação 8.

- Controle do campo é condicional.

- Controle não pode ser literal.

- Guarda posição e tamanho do campo condicional.

- Guarda informações de CPO.COND em área inviolável.

- Define, pelo tamanho do campo condicional, o número de números de campo

Ação 1Ø: ... id: pos = dec {7}, DTC {9} de id {1Ø}: (...

- Situação da P.A.:
  Ø : ID.ENDER de ID.CPO.CASO

- Se ID.NAO.DECL, ERRO.

- Se ID.NAO.CONSTANTE, ERRO.

- Se ID.NAO.ESP.LISTA.COND, ERRO.

- Campos seguintes estão dentro de CASO, sinalizados por ID.CPO.CASO

- Armazena valor binário de ID.CPO.CASO.

- Trata POS.ANTERIOR e TAM.ANTERIOR para compatibilizar a cada ID.CPO.CASO.

- Esvaziar P.A.

Ação 11: ... DTC {8}; LC); LF fim {11} LC

- P.A. vazia.

- Campos seguintes não estão mais dentro de CASO.

Ação 12: ET --> (id {12} LM)

- Se ID.NAO.DECL, ERRO.

Rotinas Chamadas:

POP.TOPO
ERRO
GRAVA.COD
PREENCHE.LISTA.MENM

2.6.11 PREENCHE.LISTA.MNEM

Chamada:

PREENCHE.LISTA.MNEM (ACAO 9)
parâmetro AÇÃO 9 - indica, sendo verdadeiro ou falso, se esta rotina está sendo chamada a partir da ação semântica 9 ou ação semântica 8, respectivamente.

Função:

Preencher e gravar no arquivo-objeto a área de código relativa a lista de mnemônicos associados a determinado campo. O mnemônico é colocado na posição relativa ou seu valor.

Rotinas Chamadas:

ERRO
GRAVA.COD

Rotina que a Chama:

ACAO.SEMANTICA

CAPITULO IV - MANUAL DE LÓGICA

3. MONTADOR

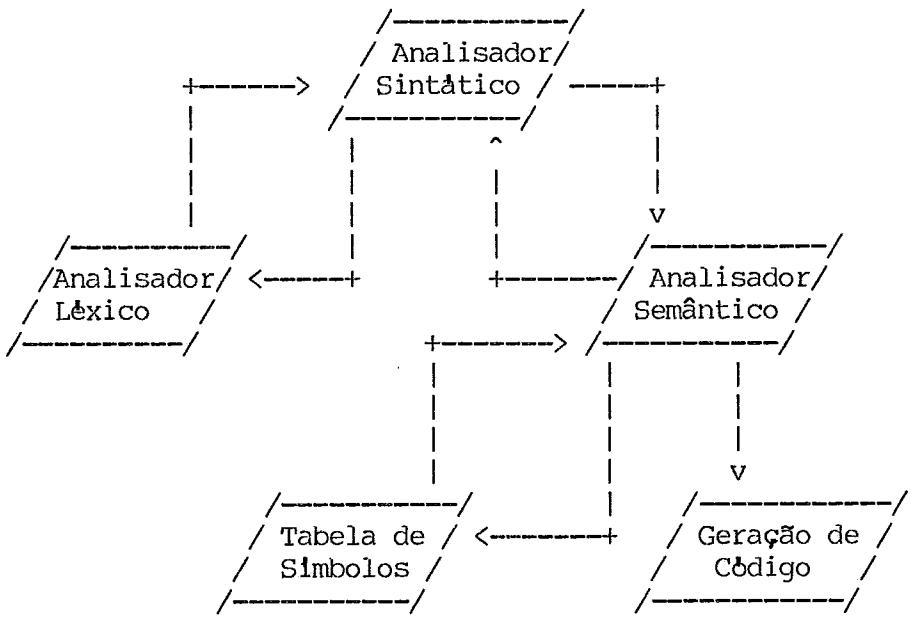
3.1 Introdução

Esse módulo exige como entrada a tabela gerada no módulo anterior e o programa escrito na microlinguagem. Essa microlinguagem - cuja gramatica está descrita no Manual de Utilizacao, item 2.5 - é na verdade uma lista de microinstruções compostas por mnemônicos e endereços absolutos de memória.

Em função da simplicidade da gramática associado a falta de recursão à esquerda na sua descrição, adotou-se tambem nesse módulo o método LL(1) para a análise sintática.

O núcleo desse módulo é o algoritmo sintático preditivo, assim como no módulo GERADOR DE TABELA. Esse núcleo controla os analisadores léxicos e semânticos - que além de gerar o código, gerencia a tabela de símbolos.

ESQUEMA DE FUNCIONAMENTO



CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.2 Analisador Sintático

A seqüência algorítmica desse módulo é idêntica a do módulo anterior: a partir da gramática compactada, baseada na definida para o programa-fonte desse módulo, monta-se uma tabela LL(1) específica a ser consultada para cada situação nova entre a pilha de símbolos, preenchida pelos lados direitos das produções da gramática compactada começando com a produção inicial, e a cadeia de tokens traduzida pelo analisador léxico a partir do texto de entrada. As situações não previstas na tabela LL(1) equivalem a erros sintáticos.

3.2.1 Gramática Compactada

Para efeito de análise sintática, a gramática a ser trabalhada é a seguinte:

```
Ø      S -> P$

1      P -> máquina id A LI.

2,3    A -> id E
          | B

4,5    E -> D ; A
          | MIS

6,7    D -> = N
          | externa

8,9    N -> dec
          | hex

1Ø,11 MIS -> : LC
             | LC1

12,13 LC -> C LC1
            | CT LC1

14,15 C -> id
           | epsilon

16,17 LC1 -> , LC
             | epsilon

18,19 CT -> bin
            | hex

2Ø,21,22  B -> org hex
               | LC1
               | CT LC1

23,24 LI -> ; I LI
            | epsilon

25,26 I -> B
           | id MIS
```

CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.2 Analisador Sintático

3.2.2 Construção da tabela LL(1).

3.2.2.1 Definição de FIRST e FOLLOW das produções

	FIRST	FOLLOW
P	máquina	$
A	id,org,,,bin, hex,epsilon	;,.
B	org,,,bin,hex, epsilon	;,.
CT	hex,bin	;,.,;
E	=,externa,:,,, epsilon	;,.
D	=,externa	;
N	dec,hex	;
I	id,org,bin,hex, ,,epsilon	;,.
C	id,epsilon	;,.,,
MIS	:,,,epsilon	;,.
LC	id,bin,hex,,, epsilon	;,.
LC1	,,epsilon	;,.
LI	;,epsilon	.

CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.2 Analisador Sintático

3.2.2.2 - Tabela LL(1)

	Ø	1	2	3	4	5	6	7	8	9	1Ø	11
		ma qui na	ex ter na	org	dec	hex	dec	id	:	=	.	,	;
Ø	P	1	-	-	-	-	-	-	-	-	-	-	-
1	A	-	-	3	3	3	-	2	-	-	3	3	3
2	B	-	-	2Ø	22	22	-	-	-	-	21	21	21
3	LCl	-	-	-	-	-	-	-	-	-	17	16	17
4	CT	-	-	-	18	19	-	-	-	-	-	-	-
5	E	-	4	-	-	-	-	-	5	4	5	5	5
6	D	-	7	-	-	-	-	-	-	6	-	-	-
7	MIS	-	-	-	-	-	-	-	1Ø	-	11	11	11
8	N	-	-	-	-	9	8	-	-	-	-	-	-
9	LC	-	-	-	13	13	-	12	-	-	12	12	12
1Ø	C	-	-	-	-	-	-	14	-	-	15	15	15
11	LI	-	-	-	-	-	-	-	-	-	24	-	23
12	I	-	-	25	25	25	-	26	-	-	25	25	25

3.2.3 Pilha de Símbolos

É simulada através da criação de dois vetores: um armazena o número de símbolos dos lados direitos das produções a serem empilhadas - no programa MONTADOR esse vetor de variáveis chama-se NSIMB.LDS - e o outro armazena o topo dessa produção (considera-se topo o símbolo inicial, da esquerda para a direita) - no programa chama-se PEND.SIMB. Esses vetores admitem até 127 posições - valor de TPSIMB.

3.2.4 Armazenamento das produções

Idêntico ao constante no item 2.2.4 relativo ao GERADOR DE TABELAS

CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.3 Analisador Léxico

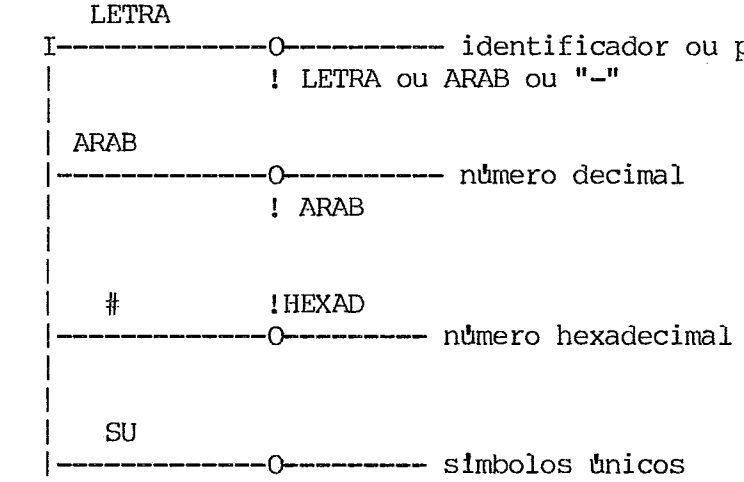
Esse segmento admite cadeias de caracteres como entrada de acordo com a seguinte gramática regular:

```
                                          *
TEXTO: ((PALAVRA|NUM|BRANCOS) DELIM)

PALAVRA: ("maquina","org","id","externa")

NUM: (BIN , DEC , HEX)
           +
BIN = $ BIT

BIT = "Ø"|"1"
         +
DEC = ARAB

ARAB = BIT|"2"|...|"9"
             +
HEX = # HEXAD

HEXAD = ARAB|"A"|"B"|...|"F"
              +
BRANCOS: (" ")

DELIM: (BRANCOS,";",".",":",",","=","&")
```

A saída desse segmento, que implementa um algoritmo baseado em autômato finito, devolve as informações de qualificação do token, número de símbolos que o compoem e endereço do inicio de sua cadeia de símbolos.

CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.3 Analisador Léxico

O autômato finito implementado através do comando CASE possui uma forma gráfica semelhante à do GERADOR DE TABELAS:



As classes dos tokens nesse segmento podem ser:

LETRA = Ø, DECIMAL = 1, HEXADECIMAL = 2, SIMBOLOS.UNICOS = 3.

CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.4 Analisador Semântico

A Análise semântica foi implementada pela inclusão de símbolos de ação semântica na gramática compactada. O analisador semântico gerencia a tabela de símbolos e gera o código de saída do MONTADOR.

3.4.1 Gramática compactada com símbolos de ação semântica.

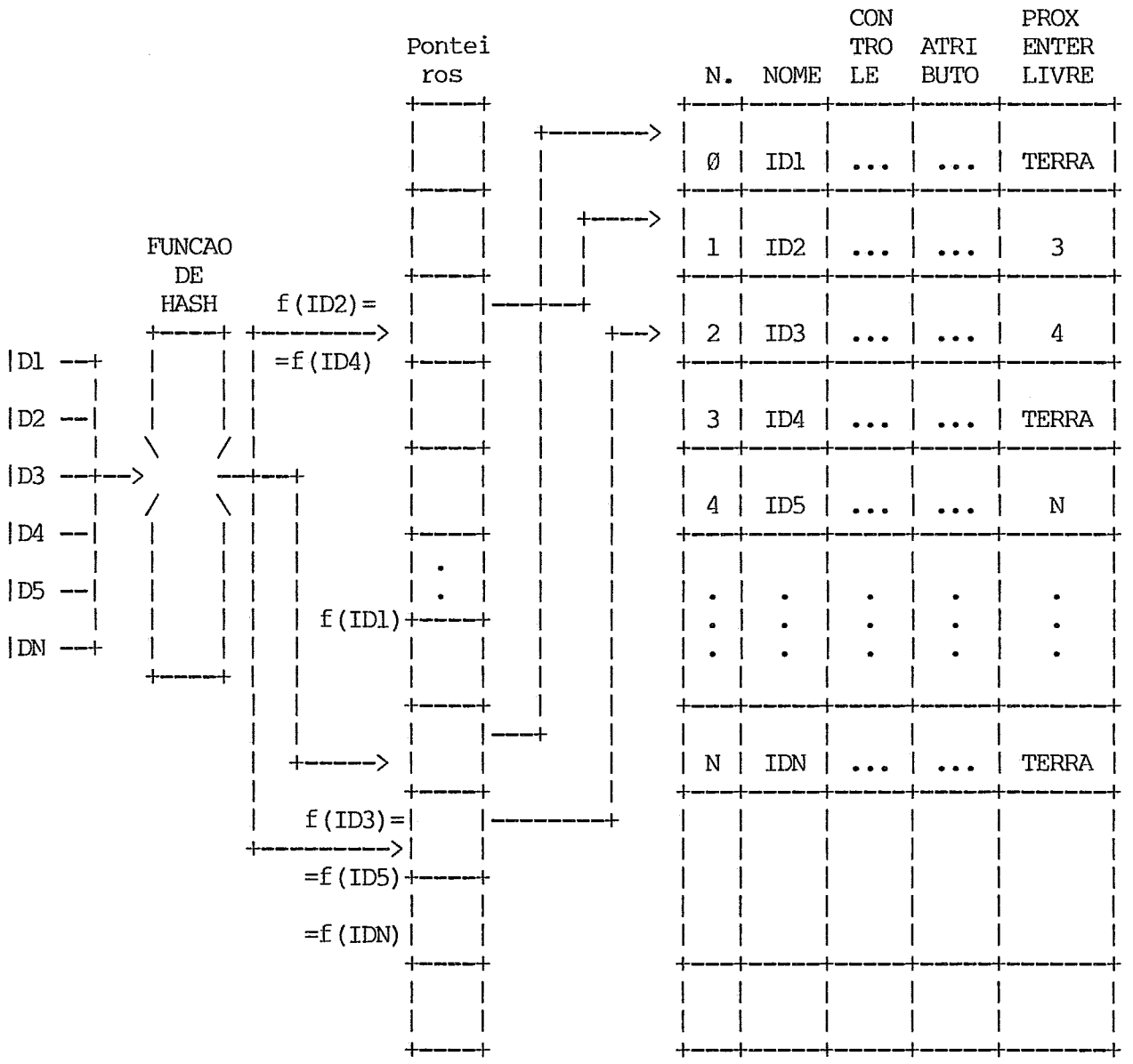
Os símbolos de ação semântica estão entre chaves e cada um deles corresponde a uma chamada de rotina semântica - ver item 3.7.8.

```
S -> P$

P -> maquina id {5}; A LI {4}.

A -> id E
    |B

E -> D;A
    |MIS

D -> = {Ø} N
    |externa {1}

N -> dec
    |hex

MIS -> {2} : LC
      |{3} LC1

LC -> C LC1
     |CT LC1

C -> id {3}
    |{3} epsilon

LC1 -> ,LC
      | epsilon

CT -> {6} bin
     |{6} hex

B -> org {7} hex
    |{3} LC1
    |CT LC1

LI -> {4}; I LI
     | epsilon

I -> B
    |id MIS
```

CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.4 Analisador Semântico

3.4.2 Tabela de Símbolos

A tabela de símbolos é preenchida seqüêncialmente conforme o aparecimento no programa-fonte das variáveis externas, constantes ou rótulos. Ela é acessada através de uma tabela de ponteiros preenchida por uma função de hash idêntica aquela do módulo GERADOR DE TABELAS. No caso de colisão existe um campo na tabela de símbolos que serve de ponteiro para a próxima entrada livre - conforme figura.



- Campos de cada entrada da tabela de símbolos.

1 - Nome - 9 posições para armazenar os caracteres que formam o identificador.

2 - Controle - 1 byte que indica se a entrada e' de identificador de constante, de variável externa ou rótulo.

3 - Atributo - 1 word - no caso de identificador de constante armazena seu valor; no caso de variável externa recebe o número de ordem dessa variável; e no caso de rótulo, recebe o valor do endereço representado por esse identificador.

4 - Próxima entrada livre - 1 word - aponta a próxima entrada onde está preenchido o identificador cuja função de hash é identica a desse identificador.

3.4.3 Código Gerado

São gerados dois tipos de código, conforme o uso que se deseja fazer do microcódigo.

3.4.3.1 Opção Hardware

É gerado um código que é a imagem em bits da seqüência de microinstruções do programa-fonte.

Esse código é utilizado, por exemplo, como entrada num queimador de PROM's.

3.4.3.2 Opção Software

É gerado código de módulo-objeto para trabalhar sob o SOD da linha COBRA-5ØØ.

O módulo-objeto gerado é formado pelos seguintes blocos:

1 - Bloco Nome - guarda informações sobre tipo, nome, número de variáveis externas e tamanho total da área do módulo-objeto

2 - Bloco Externo - informa nome e número de ordem das variáveis externas. Cada variável externa possui um bloco.

3 - Bloco Texto - guarda informações sobre bits de relocação e o próprio código gerado - a imagem em bits da seqüência de microinstruções. Os bits de relocação indicam a fase de link-edição a posição das variáveis externas

4 - Bloco Fim - indica fim do texto de código gerado.

CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.4 Analisador Semântico

3.4.4 Algoritmo para Pesquisa de Rótulos

3.4.4.1 Se rótulo precede microinstrução - Sintaxe : id : ...

3.4.4.1.1 - Rótulo já existe na tabela de símbolos

- Se campo de endereço já está preenchido, erro por duplicação de rótulos.

- Senão, preenche campo de atributo na tabela de símbolos com a posição da atual microinstrução no microcódigo.

3.4.4.1.2 - Rótulo não existe na tabela de símbolos

- Preenche campos de nome e atributo na tabela de simbolos.

3.4.4.2 Rótulo em campo literal de microinstrução

3.4.4.2.1 Rótulo já existe na tabela de símbolos:

- Se campo de atributo já preenchido, ele é transcrito para a área de código.

- Senão, área de código recebe o endereço do rótulo na tabela de símbolos e é empilhado a posição da atual microinstrução.

3.4.4.2.2 - Rótulo não existe na tabela de símbolos:

- Preenche o campo de nome.

- Área de código recebe o endereço do rótulo na tabela de símbolos.

- Empilha-se a posição da atual microinstrução.

3.4.4.3

Após o reconhecimento do fim do microprograma, a pilha de rótulos com endereço indefinido é seguida sendo a área de código preenchida com o campo de endereço da tabela de símbolos.

CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.5 Uso de memória adicional

Faz-se necessario o uso de memória adicional para armazenamento do arquivo-tabela. Essa área deve ser equivalente a área do arquivo-tabela (informação constante da listagem do módulo GERADOR DE TABELA) mais 3000 bytes necessarios para a pilha de trabalho do compilador LPS e para a área de abertura do arquivo-tabela.

CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.6 Arquivos Utilizados

3.6.1 Arquivos de entrada

- Arquivo-tabela - arquivo de saída do módulo GERADOR DE TABELA.

- Arquivo-fonte - contém o microprograma-fonte.

3.6.2 Arquivos de saída

- Arquivo-objeto - contém o código gerado.

- Arquivo-listagem - contém a listagem do microprograma-fonte, com informações de posição de memória de cada microinstrução, lista de erros e cabeçalhos.

- Arquivos-rótulos - arquivo opcional que contém a lista de rótulos usados no microprograma com seus respectivos endereços. É uma importante ferramenta para depuração de erros no microprograma-fonte.

CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.7 Principais Rotinas Locais

O MONTADOR foi dividido em cinco grandes blocos: o bloco de análise sintática que inclui o algoritmo de análise sintática preditiva e rotinas de abertura de arquivos, de operações na pilha de símbolos, de erros no microprograma-fonte e fim de compilação; o bloco de análise dos parâmetros de entrada com informações sobre os arquivos utilizados durante a compilação; o bloco de análise léxica que inclui o algoritmo de autômato finito e rotinas de leitura e impressão do arquivo-fonte; o bloco de análise semântica que implementa através de um comando CASE os pontos de chamada de rotinas semânticas, gerencia a tabela de símbolos e gera o código para a opção de saída para hardware; e o bloco opcional de geração de módulo-objeto, usado quando a opção de saída destina-se a aplicações de software.

3.7.1 Rotinas do bloco de análises dos parâmetros

3.7.1.1 ANALISA.PARMS

Função:

Análise léxica dos parâmetros passados na diretiva de execução do MONTAMIC. Armazena as informações corretas num buffer especial.

Sintaxe dos parâmetros

```
                    | <unid-disco>  |
E: <arq-fonte> [ <                  > ] ; T: <arq-tabela>
                    | <volume>      |

  | <unid-disco> |                          | <unid-disco> |
[ <              > ] ; S: <arq-objeto> [ <              > ] ; ES;
  | <volume>     |                          | <volume>     |

                      | <unid-disco> |     | SOFT |
R: <arq-rótulos> [ <                  > ] ; <      > ;
                      | <volume>     |     | HARD |
```

Essas informações são opcionais podendo ser supridas por alocações prévias a execução do MONTAMIC.

Em caso de erro nas informações, o programa termina com a mensagem: "ERRO NA PASSAGEM DOS PARÂMETROS".

Rotinas Chamadas:

SIMBOLO
EXIJA
CABECA.PARM
MOVE.INFO.ARQ
LETRA
AVANCA.PT
ERRA.PARMS

3.7.2 Rotinas do bloco de análise sintática

3.7.2.1 ABRE.ARQS

Função:

Abre arquivo-tabela e preenche a memória adicional com seu conteúdo. Aloca unidade de listagem (default é IL01). Abre arquivo-fonte. Cria, se não existir, e abre arquivo objeto com organização seqüêncial. Imprime cabeçalho de início de compilação. Cria arquivo para rótulos, se o usuário assim tiver optado.

Rotinas Chamadas:

ERRO
IMPR.CABEC
NUM.LOG

3.7.2.2 PUSH.LADO.DIR

Chamada:

PUSH.LADO.DIR (PRODUCAO)
parâmetro PRODUCAO - indica o número da produção a ser empilhada.

Função:

Empilha os lados direitos das produções da gramática compactada na pilha de símbolos. Controla estouro da tabela de símbolos.

Rotinas que a chamam:

ANALISADOR.SINTATICO
ERRO

3.7.2.3 POP.TOPO

Função:

Desempilha o símbolo do topo da pilha de símbolos.

Rotinas que a Chamam:

ANALISADOR.SINTATICO
ERRO
ACAO.SEMANT

3.7.2.4 ERRO

Chamada:

ERRO (NUM.ERRO)
parâmetro NUM.ERRO - leva o número do erro detectado.

Função:

Aponta onde e qual erro foi detectado. No caso de erro sintático, aplica-se a técnica do panic mode, procurando-se o próximo símbolo ";" na pilha de símbolos. No caso de erros graves, liga-se um sinal de que arquivo-objeto não deve ser gerado. No caso de erro fatal, fim de programa.

Rotinas que a Chamam:

GERA.CODIGO
PREENCHE.TSEQ.SIMB
TRATA.CPO.LIT
PROC.MNEM
ACAO.SEMANT
SE.TRUNCADO
ANALISADOR.LEXICO
PUSH.LADO.DIR
ABRE.ARQS
ANALISADOR.SINTATICO

Rotinas Chamadas:

PEGA.PROX.TOKEN
POP.TOPO
IMPR.CABEC
PUSH.LADO.DIR

3.7.2.5 FIM.DE.GERACAO

Função:

Preenche as áreas de código relativas a rótulos cujos endereços estavam indefinidos. Preenche lista de informações de fim de listagem. Se opção de arquivo de saída for módulo-objeto para utilização de software, chama rotina GRAVA.MOD.OBJ. Grava arquivo de rótulos, se necessário. Termina a execução do MONTAMIC.

Rotinas Chamadas:

GRAVA.OU.REGRAVA
IMPR.CABEC
GERA.CODIGO
GRAVA.MOD.OBJ

3.7.3 Rotinas do bloco de análise léxica

3.7.3.1 PEGA.PROX.TOKEN

Função:

Le o arquivo-fonte, deprezando registros com comentário. Controla a chamada ao analisador léxico devolvendo ao analisador sintático o próximo token a ser validado.

Rotinas Chamadas:

LE.ARQFONT
ANALISADOR.LEXICO

Rotinas que a Chamam:

ANALISADOR.SINTATICO
ERRO

3.7.3.2 ANALISADOR.LEXICO

Função:

Desprezando os caracteres brancos, devolve a qualificação do token, seu tamanho e seu endereço de início. No caso de número decimal, devolve tambem seu valor. Implementa um autômato finito, dividindo nas seguintes classes as cadeias de caracteres recebidas: identificador ou palavra reservada (classe Ø), número decimal (classe 1), número hexadecimal (classe 2), número binário (classe 3), símbolos únicos, próprios ou não a gramática (classe 4).

Rotinas Chamadas:

COME.BRANCOS
MONTA.CADEIA
LETRA
DIGITO
ERRO

Rotina que a Chama:

PEGA.PROX.TOKEN

3.7.4 Rotinas do bloco de análise semântica

CAPITULO IV - MANUAL DE LÓGICA
3. MONTADOR
3.7 Principais Rotinas Locais

3.7.4.1 ACAO.SEMANT

Chamada:

ACAO.SEMANT (ACAO)
parâmetro AÇÃO - indica o número da rotina de chamada de ação semântica.

Função:

Através do comando condicional estruturado CASE, controlado pelo parâmetro da rotina, é executado uma das seguintes ações semânticas:

Ação Ø: D --> = {Ø} N

- ID é declaração de constante.
- Chama função de hash para inserir ID na tabela de símbolos.
- Se ID.JA.DECL, ERRO.
- Preenche campos de controle e atributo (com o valor numerico associado ao identificador de constante) da tabela de símbolos.

Ação 1: D --> externa {1}

- ID é declaração de variável externa
- Chama função de hash para inserir ID na tabela de símbolos.
- Se ID.JA.DECL, ERRO
- Preenche campos de controle e atributo (com número de ordem relativo a atual variável externa) da tabela de símbolos.
- Se código gerado para hardware, ERRO.

Ação 2: MIS --> {2} : LC

- ID é declaração de rótulo precedendo microinstrução.
- Chama função de hash para inserir ID na tabela de símbolos.
- Se ID.JA.DECL e campo de atributo na tabela de símbolos ja preenchido, ERRO.
- Preenche campos de controle e atributo (com endereço da microinstrução) da tabela de símbolos.

Ação 3: MIS --> {3} LC1

C --> id {3}
C --> {3} epsilon

B --> {3} LC1

- ID é um campo (microordem) da microinstrução. Campo pode ser fixo (ID é mnemônico), condicional (ID é mnemônico), literal (ID é rótulo, constante ou variável externa), variável literal (ID é rótulo, constante ou variável externa) e variável não literal (ID é mnemônico). Se ID é mnemônico, é pesquisada seqüêncialmente a tabela de mnemônicos, trazida pelo arquivo-tabela, relativa a esse campo e o seu valor é transcrito para a área de código. Se ID é rótulo, constante ou variável externa, é pesquisada a tabela de símbolos; no caso de rótulo é seguido o algoritmo de pesquisa de rótulos (ver item 3.4.4) e nos outros dois casos é transcrito para a área de código o conteúdo do campo de atributos da tabela de símbolos.

- Se campo condicional, o valor do mnemônico é guardado para sinalizar os campos variáveis seguintes.

Ação 4: P --> máquina id {5} ; A LI {4}.
------ LI --> {4} ; I LI

- Fim da atual microinstrução

- Atualizar posição no microcódigo e número da próxima microinstrução.

- Listar código gerado da atual microinstrução. Passa esse código para o buffer-objeto.

Ação 5: P --> máquina id {5} ; A LI {4}.

- Início do microprograma.

- Se nome de máquina diferente do nome do arquivo-tabela, ERRO.

Ação 6: CT --> {6} hex
------- CT --> {6} bin

- Campo da microinstrução é um valor numérico binário ou hexadecimal.

- Se o número de bits desse valor numérico for igual ou superior ao número de bits da área de código que falta ser preenchido, considera se finalizada a atual microinstrução. Se for menor, considera-se que esse valor numérico representa apenas o atual campo.

Ação 7: B --> org {7} hex

- Mudança de posição de endereço no microcódigo.

- Algoritmo implementado:

1. - Se mudança de posição significa avanço no microcódigo:

1.1 - Constata-se se esse avanço significa pular registros do arquivo-objeto.

1.2 - Constata-se no caso de pular registros se esses registros já existem. Se não existem são gravados com seu conteúdo valendo o default de preenchimento do microcódigo.

2. - Se mudança de posição significa recuo no microcódigo:

2.1 - Armazena-se maior posição alcançada no microcódigo.

2.2 - Constata-se se esse recuo significa mudar de registro.

2.3 - No caso de mudanca de registro, regrava-se o atual antes da leitura do indicado.

Rotinas Chamadas:

POP.TOPO
HASH
ERRO
PROC.MNEM
TRATA.CPO.LIT
GR.REG.CH
GERA.CODIGO
RW.AT
GRAVA.OU.REGRAVA
GR.REG.CH

3.7.4.2 PROC.MNEM

Chamada:

PROC.MNEM (ECPO)
parâmetro ECPO - indica o endereço da descrição do campo (se campo fixo ou variável).

Função:

Pesquisa seqüêncial dos elementos da tabela de mnemônicos de determinado campo da microinstrução. Se não for encontrado será pesquisada a tabela de símbolos para verificar se identificador é constante ou variável externa.

Rotinas Chamadas:

GERA.CODIGO
HASH
ERRO

3.7.4.3 TRATA.CPO.LIT

Função:

Trata o campo literal de microinstrução: se é identificador de constante ou variável externa transcreve atributo de tabela de símbolos para área de código e se é rótulo implementa pesquisa de rótulo na tabela de símbolos quando rótulo está no campo literal da microinstrução (ver item 3.4.4).

Rotinas Chamadas:

GERA.CODIGO
HASH
ERRO

3.7.4.4 HASH

Função:

É uma word procedure que implementa uma função de hash e devolve o endereço de entrada na tabela de símbolos para o identificador.

A função de hash passa por tres fases:
- fase de ou-exclusivo - é feito ou-exclusivo entre words compostas pela associação de cada dois caracteres do identificador, sendo que de cada caracter é retirado o 1o. e 5o. bit mais significativo para então ser compactado em 6 bits e ser transcrito na word.
- fase de potência - a word resultante da fase anterior é levada ao quadrado e então são tomados os 1Ø bits menos significativos para entrada na tabela de símbolos.
- fase de colisão - em caso de colisao é preenchido o ponteiro da entrada na tabela de símbolos para a próxima entrada livre dessa tabela.

Rotina Chamada:

PREENCHE.TSEQ.SIMB

Rotina que a Chama:

ANALISADOR.LEXICO

CAPITULO V - MANUAL DE UTILIZAÇÃO

1. GERADOR DE TABELAS

1.1 Constituintes Básicos

1.1.1 Conjunto de Caracteres

- LETRAS -> A B ... X Y Z
- ALGARISMOS DECIMAIS -> Ø 1 2 3 4 5 6 7 8 9
- VAZIO -> " "
- CARACTERES ESPECIAIS -> ; , ( ) : = & # $

1.1.2 Identificadores

São utilizados para designar as constantes, os tipos, o nome da máquina e os nomes dos campos que compõem a máquina.

Os identificadores de constante e tipo são pré-declarados no início do programa, sendo a constante associada a um valor numérico e o tipo a um conjunto de parâmetros.

As constantes são usadas para designar os mnemônicos associados a microordens.

Os tipos são geralmente usados quando dois campos da microinstrução possuem o mesmo conjunto de parâmetros - tamanho em bits do campo, default de preenchimento e lista de mnemônicos associados a microordens ou, na ausência dessa lista, a informação de que esse campo é de endereçamento.

A formação dos identificadores obedece as seguintes regras:

- Devem começar por letra.
- Podem ser constituídos por qualquer combinação de letras e algarismos decimais.
- Possuem no máximo 9 caracteres.

CAPITULO V - MANUAL DE UTILIZAÇÃO

1. GERADOR DE TABELAS

1.1 Constituintes Básicos

1.1.3 Palavras Reservadas

Existe um conjunto de 11 palavras que embora satisfaçam as condições de formação para identificadores não podem ser usadas como tal.

São as seguintes:

- CAMPOS
- CASO
- CONSTANTE
- DE
- ENC
- FIM
- LITERAL
- MAQUINA
- POS
- TAM
- TIPO

1.1.4 Valores Numéricos

. DECIMAIS - são os números representados na base 10, constituídos pelos algarismos de 0 a 9.

. BINÁRIOS - são os números representados na base 2, constituídos dos algarismos decimais 0 e 1 precedidos pelo caracter $. Possuem no máximo 16 caracteres.

. HEXADECIMAIS - são os números representados na base 16, constituídos pelos algarismos decimais de 0 a 9 e pelas letras de A a F, precedidos pelo caracter #. Possuem no máximo 4 caracteres.

CAPITULO V - MANUAL DE UTILIZAÇÃO

1. GERADOR DE TABELAS

1.2 Declarações

O programa-fonte para entrada no GERADOR DE TABELA é uma lista de declarações, composta por declarações de dados e declarações de formato da máquina a qual o microprograma estará vinculado. As declarações de dados são opcionais.

Esse programa-fonte é a descrição dos campos (ou microordens) que compõem cada microinstrução. Essa descrição equivale a definição dos seguintes parâmetros:

1 - Posição relativa de cada campo na palavra a ser gerada a cada microinstrução completa;

2 - Tamanho em bits do campo;

3 - Default de preenchimento desse campo;

4 - Lista de mnemônicos a ele associados com respectivos valores.

1.2.1 Declarações de Dados

São usadas para definir propriedades de algumas utilidades do programa e associá-las a identificadores. Sempre antecedem as declarações do formato de máquina.

1.2.1.1 Declarações de Constante

Associam a um identificador um valor numérico, que será escrito necessariamente sob a forma binária ou hexadecimal. Esse identificador designará um mnemônico de campo (ou microordem) de cada microinstrução. O bloco de declarações de constante é precedido pela palavra reservada CONSTANTE.

Se o número binário ou hexadecimal passado possui um número de caracteres superior ao permitido, ele será truncado, sempre à esquerda, o número de caracteres necessários a se atingir o máximo permitido.

```
EXEMPLO: CONSTANTE PREF = $ØØØ;
                   REF = #A;
```

CAPITULO V - MANUAL DE UTILIZAÇÃO
1. GERADOR DE TABELAS
1.2 Declarações

1.2.1.2 Declarações de Tipo

Associam a um identificador um conjunto de parâmetros relativo as informações de um determinado campo (ou microordem): quantos bits esse campo ocupa, qual o default de preenchimento e a lista de mnemônicos que o compõe ou se é um campo de rótulos. Essa declaração é usada para campos que ocupam posições diferentes na micro-instrução embora sejam idênticos nos parâmetros acima definidos. O parâmetro que indica o tamanho do campo é passado através de um valor numérico escrito em decimal e o que indica o default de preenchimento através de um valor numérico escrito em binário ou hexadecimal ou de um identificador, já pré-declarado, que designa uma declaração de constante. A lista de mnemônicos é uma lista de identificadores, todos pré-declarados, onde cada um designa uma declaração de constante.

O bloco de declarações de tipo deve ser precedido pela palavra reservada TIPO.

```
EXEMPLO:
TIPO RAM : TAM = 3, ENC = R2,(RØ,R1,R2,R3,R4);
     CPOLABEL : TAM = 5, ENC = $ØØØØØ, LITERAL;
```

Neste exemplo, RAM designa um campo de 3 bits, default de preenchimento de valor igual ao de R2 e com a lista de mnemônicos entre parenteses. CPOLABEL designa um campo de endereçamento de 5 bits com default de preenchimento de valor Ø.

CAPITULO V - MANUAL DE UTILIZAÇÃO
1. GERADOR DE TABELAS
1.2 Declarações

1.2.2 Declarações do formato da máquina

Caracterizam a estrutura da microinstrução. Seu uso é obrigatório no programa. Vejamos o exemplo:

```
Ø1 MAQUINA CPU5ØØ : TAM = 16, ENC = #FF,
Ø2          CAMPOS BYTE : POS = Ø, TAM = 1, ENC = $Ø, (P,B);
Ø3                 PROXENDER : POS = 1, CPOLABEL;
Ø4                 RAM1 : POS = 6, RAM;
Ø5                 RAM2 : POS = 9, RAM;
Ø6                 CASO COND : POS = 12, TAM = 1, ENC = V, (F,V)
Ø7                      DE F : (ZERO:POS = 13, TAM = 1, ENC = $Ø, (S,N);
Ø8                                SHIFT:POS = 14, TAM = 2, ENC = $Ø1,
Ø9                                     (QREG,NOP,RAMA,RAMB););
1Ø                           V : (FUNCAO : POS = 13, TAM = 3, ENC = DIV,
11                                     (CLRMB, PREF, REF, DIV););
12                 FIM
13          FIM
```

Na linha Ø1 do programa é especificado o nome da máquina para a qual se destina o microprograma (CPU5ØØ), o tamanho de cada uma de suas microinstruções (16 bits), assim como o seu default de preenchimento (#FF). A palavra MAQUINA é reservada.

O arquivo de saída, com a tabela gerada, chamar-se-á CPU5ØØ.

Na linha Ø2 inicia-se a definição dos campos que compoem cada microinstrução. O campo BYTE, que ocupa a posição Ø, possue o seguinte conjunto de parâmetros: 1 bit de tamanho, default de preenchimento da microordem igual ao binário $Ø, e lista de mnemônicos associados (no caso P e B).

As linhas Ø3, Ø4 e Ø5 indicam campos cujos conjuntos de parâmetros já devem ter sido anteriormente designados na declaração de tipo.

A linha Ø6 mostra que o valor do mnemônico desse campo (F ou V) serve como sinalisador de quais campos prosseguem a microinstrução.

As linhas de Ø7 a 11 mostram outros campos de microinstrução.

A linha 12 serve para finalizar a estrutura CASO e a linha 13 para finalizar a descrição dos campos, e portanto, o programa.

CAPITULO V - MANUAL DE UTILIZAÇÃO
1. GERADOR DE TABELAS
1.3 Ativação do Gerador de Tabelas.

O programa executável é chamado GERATAB e para a sua ativação faz-se necessário informações sobre o arquivo com o programa-fonte, o arquivo para listagem e se caso já exista o arquivo-objeto de saída, ele deve ser esvaziado.

Essas informações podem ser passadas através de alocações prévias ou através da área de parâmetros.

A unidade de listagem padrão adotada é o SPOOL, alocado a unidade lógica número 6. Se se desejar outra unidade de listagem ela deve ser alocada a unidade 6 com RDEF ligado.

Através da passagem de parâmetros pode ser informado o nome do arquivo que contém o programa-fonte ou o número lógico já utilizado na sua alocação através da sintaxe E:{<nome-arq>|<num-logico>}; e pode tambem ser esvaziado o arquivo de saída, caso já exista, através da sintaxe ES.

A informação <nome-arq> inclui, além de até 8 caracteres com o nome do arquivo que contém o programa-fonte, duas opções excludentes escritas entre colchetes: a informação sobre o disco ou o volume onde se encontra o fonte.

Se o arquivo com o programa-fonte não for passado como parâmetro, deve ser alocado préviamente a unidade lógica número 7.

O arquivo de saída com a tabela gerada será criado na unidade de trabalho com o mesmo nome especificado para a máquina que se destina o microassembler.

Exemplos de Ativação:

1) :EX GERATAB PA
>E:FCPU500[V=00001];ES;

CAPITULO V - MANUAL DE UTILIZAÇÃO
1. GERADOR DE TABELAS
1.4 ERROS

1.4.1 Mensagens de Erros graves - impedem a geração do arquivo de saida:

Ø - Construção não permitida pela linguagem de especificação.

1 - Identificador duplicado.

2 - Identificador não declarado ou não especificado.

3 - Identificador de TIPO não encontrado.

4 - Posição do atual campo está diferente da soma da posição do campo anterior com o tamanho anterior.

5 - Campo condicional (especifica CASO) não pode ser campo de rótulo.

6 - Mnemônico não pertence a lista de mnemônicos do campo condicional.

7 - Dois mnemônicos do mesmo campo possuem mesmo valor.

8 - Posição do atual campo maior que tamanho total da micro-instrução.

9 - Tamanho do atual campo maior que 16 bits e atual campo não é rótulo.

1.4.2 Mensagens de advertências:

1Ø - Identificador com mais de 9 caracteres-truncados à direita.

11 - Decimal com mais de 4 caracteres-truncados à esquerda.

12 - Binário com mais de 16 caracteres-truncados à esquerda.

13 - Hexadecimal com mais de 4 caracteres-truncados à esquerda.

14 - Caracter não especificado na linguagem de especificação.

15 - Tamanho do atual campo diferente do número de bits do parâmetro ENC.

16 - Tamanho do atual campo diferente do número de bits do mnemônico - o número é alinhado à direita.

1.4.3 Mensagens de Erro Fatal - significa fim do programa:

17 - Seqüência de erros faz analisador sintático se perder. Fim do programa.

CAPITULO V - MANUAL DE UTILIZAÇÃO
1. GERADOR DE TABELAS
1.5 BNF DA GRAMÁTICA

```
<programa> --> <lista-dec>

<lista-dec> -->  <dec-val> ; <dec-maq>
               ! <dec-maq>

<dec-val> --> <dec-const> ; <dec-tipo>
            ! <dec-const>
            ! <dec-tipo>

<dec-const> --> CONSTANTE <lista-def-mnem>

<lista-def-mnem> --> <mnem>=<num>;<lista-def-mnem>
                   ! <mnem>=<num>

<mnem> --> [cadeia de caracteres alfanuméricos]

<num> --> <bin> ! <hex>

<bin> --> [cadeia binária]

<hex> --> [cadeia hexadecimal]

<dec-tipo> --> TIPO <lista-tipo-cpo>

<lista-tipo-cpo> --> <nome-tipo-cpo> : <tipo-cpo>;<lista-tipo-cpo>
                   ! <nome-tipo-cpo> : <tipo-cpo>

<nome-tipo-cpo> --> [cadeia de caracteres alfanuméricos]

<tipo-cpo> --> <carac-cpo> , <espec-tipo>
             ! <nome-tipo-cpo>

<carac-cpo> --> TAM=<dec> , ENC=<lit>

<dec> --> [cadeia decimal]

<lit> --> <mnem> ! <num>

<espec-tipo> --> LITERAL ! (<lista-mnem>)

<lista-mnem> --> <mnem>,<lista-mnem>
               ! <mnem>
```

CAPITULO V - MANUAL DE UTILIZAÇÃO
1. GERADOR DE TABELAS
1.5 BNF DA GRAMÁTICA

<dec-maq> --> MAQUINA <nome-máquina> : <carac-cpo> , CAMPOS <descr-cpo> FIM

<nome-máquina> --> [cadeia de caracteres alfanuméricos]

<descr-cpo> --> <parte-fixa>
! <parte-var>
! <parte-fixa> <parte-var>
! <parte-fixa> <parte-var> <parte fixa>
! <parte-var> <parte fixa>

<parte-fixa> --> <lista-cpo>;

<lista-cpo> --> <cpo> ; <lista-cpo>
! <cpo>

<cpo> --> <nome-cpo> : POS = <dec> , <tipo-cpo>

<nome-cpo> --> [cadeia de caracteres alfanuméricos]

<parte-var> --> CASO <cpo> DE <lista-formato> FIM

<lista-formato> ---> <mnem> : (<lista-cpo>;) ; <lista-formato>
! <mnem> : (<lista-cpo>;)

CAPITULO V - MANUAL DE UTILIZAÇÃO
1. GERADOR DE TABELAS
1.6 Observações

. A estrutura da sintaxe da linguagem do programa-fonte é similar a da linguagem PASCAL, com a declaração do formato da máquina definida em cima da estrutura RECORD dessa linguagem.

. O programa executável é chamado GERATAB.

. Ao final da listagem é informado o nome do arquivo de saída, o número de UA's ocupadas por esse arquivo, o número de erros e de advertências. Caso haja algum erro o arquivo de saída nao é gerado.

CAPITULO V - MANUAL DE UTILIZAÇÃO

2. MONTADOR

2.1 Constituintes Básicos

2.1.1 Conjunto de Caracteres

- LETRAS -> A B ... X Y Z
- ALGARISMOS DECIMAIS -> Ø 1 2 3 4 5 6 7 8 9
- VAZIO -> " "
- CARACTERES ESPECIAIS -> : = . , ; & # $

2.1.2 Identificadores

Designam as declarações de nome da máquina, de constantes e de variáveis externas e são usados ainda na designação de rótulos para microinstrução e mnemônicos de campo (ou microordem) que compoem a microinstrução.

Os identificadores de constantes, sempre associados a um valor numérico, e de variáveis externas, que serão definidos em tempo de ligação com outros modulos-objeto, são pré-declarados no inicio do programa.

A formação desses identificadores é identica a aqueles do GERADOR DE TABELAS (ver 1.1.2), porém admite-se caracter "-" como separador.

2.1.3 Palavras Reservadas

Formam um conjunto de 3 palavras:

- MAQUINA
- EXTERNA
- ORG

2.1.4 Valores Numéricos

Assim como no GERADOR DE TABELAS - item 1.1.4 - podem ser DECIMAIS, BINARIOS e HEXADECIMAIS.

A diferença está no limite de caracteres dos valores numéricos binários - no máximo 256 - e dos hexadecimais - no máximo 64.

CAPITULO V - MANUAL DE UTILIZAÇÃO
2. MONTADOR
2.2 Programa-fonte

O programa-fonte para entrada no MONTADOR é um conjunto de declarações de dados e lista de microinstruções.

2.2.1 Declarações de dados

São usados para pré-definir valores do campo literal da microinstrução. Devem preceder a lista de microinstruções.

. CONSTANTES:
Associam a um identificador um valor numérico decimal ou hexadecimal.

A sua sintaxe é da forma : <identificador> = <valor numérico>;

. VARIÁVEIS EXTERNAS:
Associam a um identificador a informação de que o valor do campo literal só será preenchido em tempo de ligação. Cada identificador de variável externa deve ser seguido da palavra reservada EXTERNA. Podem ser no máximo 512 identificadores com um uso máximo de 4096 vezes ao longo do programa.

A sua utilização é permitida apenas em aplicações para a área de software.

O seu uso exige que o campo literal possua 16 bits.

2.2.2 Lista de microinstruções

É um conjunto formado por instruções e pseudo-instruções.

. INSTRUÇÃO:
É uma seqüência de campos - representados por mnemônicos ou constantes numéricas hexadecimal ou binária - separados por virgulas, precedida ou não por rótulo.

. PSEUDO-INSTRUÇÃO:
Existem duas pseudo-instruções possíveis:

- Pseudo-instrução ORG: utilizada para informar a posição da microinstrução. Sintaxe: ORG <número-hexadecimal>;

- Pseudo-instrução constante: utilizada para substituir, por uma constante numérica hexadecimal ou binária, toda a microinstrução.

CAPITULO V - MANUAL DE UTILIZAÇÃO
2. MONTADOR
2.2 Programa-fonte

2.2.3 Rótulos

São identificadores que podem preceder a microinstrução ou ocupar o seu campo literal.

Quando a precedem devem ser seguidos de caracter especial ":" e associam ao identificador o valor da posição da microinstrução no microcódigo servindo portanto como referência de endereçamento.

A tabela de símbolos, que abriga as declarações de dados e os rótulos, possui um tamanho máximo de 1Ø24 entradas. Esse tambem é o limite para o uso de rótulos em campos literais sem que o rótulo já tenha sido associado a número de microinstrução, ou seja, sem ter ainda precedido microinstrução.

2.2.4 Campo da microinstrução

Cada campo da microinstrução deve ser preenchido por um dos elementos da lista de mnemônicos a ele definida no programa-fonte de entrada no GERATAB ou por uma constante numérica hexadecimal ou binária.

Essa constante numérica pode substituir também os campos que faltam para completar a microinstrução. Por exemplo seja uma microinstrução de 64 bits em 15 campos; suponha que já foram passados os dois campos iniciais que ocupam 4 bits. Pode então ser passado uma constante numérica para os 6Ø bits restantes.

CAPITULO V - MANUAL DE UTILIZAÇÃO
2. MONTADOR
2.3 Ativação

O programa executável é chamado MONTAMIC e para a sua execução faz-se necessário informações sobre área adicional de memória e os arquivos nela envolvidos.

A área adicional de memória deve ser igual a soma da área do arquivo-tabela, produto do GERATAB, com 3ØØØ bytes.

Os arquivos envolvidos na execução são: arquivo com programa-fonte, arquivo-tabela, arquivo objeto de saída, arquivo para listagem e arquivo com rótulos.

O arquivo para listagem é considerado a impressora ILØ1, que é alocado a unidade lógica 6. Se se desejar outro arquivo, ele deve ser alocado a unidade lógica 6 com RDEF ligado.

Os outros arquivos envolvidos podem ser definidos das seguintes formas:

. Pré-alocações: O arquivo-fonte associado à unidade lógica 7, o arquivo-tabela e 5 e o arquivo-objeto à 8.

. Passagem de parâmetros:

```
                                                          | SOFT |
Sintaxe - E:<espec-arq>; T:<espec-arq>; S:<espec-arq>; ES; <      >;
                                                          | HARD |
          R:<espec-arq>;
```

<espec-arq> --> <nome-arq> | <num-lógico>

Parâmetro ES - indica que arquivo-objeto deve ser esvaziado

Parâmetro SOFT,HARD - indica que saida é módulo-objeto ou copia de micromemória. Default: HARD.

<nome-arq> - inclui os até 8 caracteres que formam o identificador de nome de arquivo, seguidos, opcionalmente, da informação de disco ou volume a que pertence este arquivo.

<num-lógico> - indica o número lógico já utilizado na alocação a esse arquivo

CAPITULO V - MANUAL DE UTILIZAÇÃO
2. MONTADOR
2.4 LISTA DE ERROS

2.4.1 Mensagens de advertências:

Ø - Identificador, maior que 9 caracteres, truncado à direita

1 - Número binário truncado à esquerda.

2 - Número decimal, maior que 5 caracteres, truncado à esquerda.

3 - Número hexadecimal truncado à esquerda.

4 - Caracter inesperado.

2.4.2 Mensagens de Erros fatais - significam fim do programa.

5 - Fim inesperado do programa por perda de controle do compilador.

6 - Estouro da tabela de símbolos.

7 - Nome de máquina não está de acordo com o especificado na tabela.

8 - RDEF desligado para alocação de unidade de listagem.

9 - ILØ1 já está alocada (provavelmente não foi liberada do SPOOL).

1Ø - Arquivo-objeto já existe e não foi passado parâmetro para esvaziar (EO;)

11 - Estouro da pilha de símbolos.

2.4.3 Mensagens de Erros Graves - impedem a geração do arquivo-objeto.

12 - Erro sintático.

13 - Identificador não declarado.

14 - Mnemônico inválido.

15 - Excesso de micro-ordens na composição da micro-instrução.

16 - Identificador já declarado.

17 - Rótulo duplicado.

18 - Faltam micro-ordens na composição da micro-instrução.

19 - Estouro da tabela de uso de variável externa.

2Ø - Estouro da tabela de uso de rótulo indefinido.

21 - Estouro da tabela de declaração de variável externa.

22 - Variável externa declarada com opção "HARD".

CAPITULO V - MANUAL DE UTILIZAÇÃO
2. MONTADOR
2.5 BNF DA GRAMÁTICA

```
<programa> --> <cabeçalho>; <lista-de-declarações> <lista-de-instruções>.

<cabeçalho> --> MAQUINA <nome-máquina>

<nome-máquina> --> [cadeia de caracteres alfanuméricos]

<lista-de-declarações> --> <decl> ; <lista-de-declarações>
                         ! [cadeia vazia]

<decl> --> <decl-constante>
         ! <decl-externa>

<decl-constante> --> <ident> = <valor>

<ident> --> [cadeia de caracteres alfanuméricos]

<valor> --> DEC
          ! HEX

<decl-externa> --> <ident> EXTERNA

<lista-de-instruções> --> <instrução>;<lista-de-instruções>
                        !<instrução>

<instrução> --> <pseudo-instrução-org>
              !<micro-instrução>

<pseudo-instrução-org> --> ORG <número-hexadecimal>

<micro-instrução> --> <micro-instrução-rotulada>
                    !<micro-instrução-simples>

<micro-instrução-rotulada> --> <rótulo>:<micro-instrução-simples>

<rótulo> --> [cadeia de caracteres alfanuméricos]

<micro-instrução-simples> --> <lista-de-campos>

<lista-de-campos> --> <campo>,<lista-de-campos>
                    ! <constante>,<lista-de-campos>
                    ! <constante>
                    ! <campo>

<campo> --> <mnemônico>
          ! <rótulo>
          ! <ident>
          ! [cadeia vazia]

<mnemônico> --> [cadeia de caracteres alfanuméricos]

<constante> --> <número-hexadecimal>

<número-hexadecimal> --> HEX
```

CAPITULO VI - CONCLUSOES

O Gerador de Compiladores de Microassembler revelou-se um instrumento útil tanto para a microprogramação da unidade central de processamento(CPU) da linha COBRA-5ØØ quanto para a transposição de rotinas de software básico para microprogramas implementados na memória de controle alterável(MCA).

No caso da microprogramação da CPU, demonstrou a maior facilidade da ferramenta de software se comparada à tradicional construção manual. Mostrou tambem a maior flexibilidade para alterações ocasionadas por erros eventuais ou aumento do microprograma pela inclusão de novas caracteristicas, como no caso da implementação de ponto flutuante por microprograma.

Para o uso da MCA procurou-se microprogramar partes do Sistema Operacional(despachante, tratamento de interrupções, troca de mensagens) normalmente muito executados em ambientes multi-usuários e partes das bibliotecas de programas COBOL e FORTRAN para acelerar sua execução. O curto prazo(cerca de tres meses) da realização dessa microprogramação foi possivel em parte à existência desse instrumento de geração de microcódigo.

A partir da microprogramação dessas rotinas, a melhoria do desempenho da máquina foi de 5% a 15%, dependendo do nivel de entrada/saida do teste.

Algumas possibilidades se abrem para a extensão desse sistema:

- a incorporação de controles na especificação da microlinguagem que permitam ao projetista de hardware indicar campos distintos na microinstrução que devam ser mutuamente exclusivos;

- o uso de macros que facilitariam ao projetista de software implementar rotinas básicas do sistema operacional, de bibliotecas de linguagem ou mesmo rotinas de uso geral;

- a previsível definição de novos produtos por parte da COBRA certamente levará à incorporação de novas caracteristicas nesse Gerador de Compiladores.

Esse trabalho aqui apresentado demandou quatro meses entre especificação e implementação de seu núcleo principal, um mes para documentação e dois meses para mudanças necessárias em função da geração de módulo-objeto para aplicações de software.

A ocupação de memória das duas fases se deu da seguinte forma: O GERADOR DE TABELA ocupa uma área de 8,3K bytes de programa e 24,6K bytes de dados enquanto o MONTADOR ocupa 12,2K bytes de área de programa e 33,3K bytes de área de dados.

ANEXOS
ANEXO A - BIBLIOGRAFIA

1 - AHO - ULLMANN - Principles of Compiler Design.

2 - MICHAEL POWERS - JOSÉ HERNANDEZ - Microprogram Assemblers for bit-slice microprocessors - COMPUTER - July 1978 -pp 108-120.

3 - NIKLAUS WIRTH - KATHLEEN JENSEN -Pascal : user manual and report.

4 - VÁRIOS AUTORES - Compiler Construction : an advance course.

5 - DAVID GRIES - Compiler Construction for Digital Computer.

6 - DONALD KNUTH - The Art of Computer Programming - Vol. 3 : SORTING AND SEARCHING.

7 - LIGIA BARROS - Análise LL : Tratamento de Erros Sintáticos.

8 - MANUAL DE PROGRAMAÇÃO LPS - COBRA 500.

9 - LEWIS II - ROSENKRANTZ - STEARNS - Compiler Design Theory.

10 - ASHOK K. AGRAWALA - TOMLINSON G. RAUSCHER - Foundations of Microprogramming - Architecture, Software and Applications

11 - MANUAL DE REFÊRENCIA DO LIGADOR COBRA-500.

```
LPS/500 X.06  GERADOR DE TABELA-GERADOR DE MONTADORES  E:FTAB      23/09/82 11:58 PAG:00

ARQUIVO FONTE: FTAB     (DP08)    VOLUME=10101     VERSAO=V.01  CODIGO=MICROA
EQUIPAMENTO:    C500
PARAMETROS:     E:FTAB;S:RTAB;ES
OPCOES INICIAIS: LF  MI C LA  FS  S  D0  D1  D2  D3  D4  D5  D6  D7  PFS

0002 0000-D 00 ?LPS,D,"TABELA"
0003 0000-D 00 &
0004 0000-D 00 & ENTRADA : PROGRAMA-FONTE ESCRITO NUMA LINGUAGEM ESPECIALMENTE
0005 0000-D 00 &           DEFINIDA PARA A ESPECIFICACAO DAS CARACTERISTICAS DE
0006 0000-D 00 &           QUALQUER MICRO-LINGUAGEM.
0007 0000-D 00 &
0008 0000-D 00 & SAIDA   : TABELA DE MNEMONICOS POR MICRO-ORDEM CUJO CONJUNTO FORMA
0009 0000-D 00 &           CADA MICRO-INSTRUCAO DA MICRO-LINGUAGEM.
0010 0000-D 00 &
0011 0000-D 00 & CARACTERISTICAS DO PROGRAMA :
0012 0000-D 00 &
0013 0000-D 00 &  - TABELA DE SIMBOLOS ACESSADA POR FUNCAO DE HASH.
0014 0000-D 00 &
0015 0000-D 00 &  - METODO DE PARSING : LL(1) COM TABELA.
0016 0000-D 00 &
0017 0000-D 00 &  - RECUPERACAO DE ERROS : PANIC MODE.
0018 0000-D 00 &
0019 0000-D 00 & DATA DE CODIFICACAO : 23/6/81.
0020 0000-D 00 &
0021 0000-D 00 & DATA DE CONSOLIDACAO : 28/7/81.
0022 0000-D 00 &
```

```
0024 0000-D 00 BEGIN
0025 0000-P 01 &
0026 0000-P 01 & CONSTANTES :
0027 0000-P 01 &
0028 0000-P 01 CONSTANT ARQ.ENTR    =01000;
0029 0000-D 01 CONSTANT BRANCO      = " ";
0030 0000-D 01 CONSTANT DEFAULT.E.S= 0FF;
0031 0000-D 01 CONSTANT FALSO       = 0  ;
0032 0000-D 01 CONSTANT FIM.DE.ARQ  = 1  ;
0033 0000-D 01 CONSTANT FIM.DE.REG  = 000;
0034 0000-D 01 CONSTANT JA.EXISTE   = 102;
0035 0000-D 01 CONSTANT NAO.CANC    =08000;
0036 0000-D 01 CONSTANT RDEF.DESL   = 68 ;
0037 0000-D 01 CONSTANT SEQL        =00100;
0038 0000-D 01 CONSTANT SETA        = "↑";
0039 0000-D 01 CONSTANT UL.ENTR     = 7  ;
0040 0000-D 01 CONSTANT UL.LIST     =  6 ;
0041 0000-D 01 CONSTANT UL.SAI      = 240;
0042 0000-D 01 CONSTANT UL.TRAB1    = 251;
0043 0000-D 01 CONSTANT VAZIO       =0FEFF;
0044 0000-D 01 CONSTANT VERO        = 1  ;
0045 0000-D 01 CONSTANT M.BRANCO    =02020;
0046 0000-D 01 &
0047 0000-D 01 CONSTANT ESTOURO.DA.PILHA = 286; % N.MAX.MNEM.CPO+30
0048 0000-D 01 CONSTANT ESTOURO.PILHA.M1 = ESTOURO.DA.PILHA+1 ;
0049 0000-D 01 CONSTANT MAX.BIN          = 17 ;
0050 0000-D 01 CONSTANT MAX.CAR          = 9  ;
0051 0000-D 01 CONSTANT MAX.DEC          = 5  ;
0052 0000-D 01 CONSTANT MAX.HEX          = 5  ;
0053 0000-D 01 CONSTANT MAX.PERCORRE     = 10 ;
0054 0000-D 01 CONSTANT MAX.PROD         = 27 ;
0055 0000-D 01 CONSTANT NLINPAG          = 56 ;
0056 0000-D 01 CONSTANT N.MAX.MNEM.CPO   = 256;
0057 0000-D 01 CONSTANT N.TERM.TAB       = 14 ;
0058 0000-D 01 CONSTANT NUM.ENTR.TSIMB   =1024;
0059 0000-D 01 CONSTANT TAM.AREA.COD     =9*N.MAX.MNEM.CPO+5;
0060 0000-D 01 CONSTANT TAM.ENTR.TAB     = 16 ;
0061 0000-D 01 CONSTANT TAM.MAX.REG.ENTR = 80 ;
0062 0000-D 01 CONSTANT TAM.MAX.TAM      = 16 ;
0063 0000-D 01 CONSTANT TAM.TAB          =TAM.ENTR.TAB*NUM.ENTR.TSIMB;
0064 0000-D 01 CONSTANT TAM.TAB.AUX      = N.MAX.MNEM.CPO+1;
```

```
0066 0000-D 01 CONSTANT BINARIO = 2 ;
0067 0000-D 01 CONSTANT DECIMAL = 1 ;
0068 0000-D 01 CONSTANT HEXADEC = 3 ;
0069 0000-D 01 CONSTANT OUTROS  = 5 ;
0070 0000-D 01 CONSTANT PAL     = 0 ;
0071 0000-D 01 CONSTANT SIMB.UN = 4 ;
0072 0000-D 01 &
0073 0000-D 01 CONSTANT CPO.COD.COND= 1 ;
0074 0000-D 01 CONSTANT CPO.COD.FIX = 0 ;
0075 0000-D 01 CONSTANT CPO.COD.LIT = 2 ;
0076 0000-D 01 CONSTANT CPO.COD.VAR = 4 ;
0077 0000-D 01 &
0078 0000-D 01 CONSTANT CPO.TAB.CIE = 0 ;
0079 0000-D 01 CONSTANT CPO.TAB.TIP = 080 ;
0080 0000-D 01 CONSTANT CPO.TAB.LIT = CPO.TAB.TIP OR 040 ;
0081 0000-D 01 &
0082 0000-D 01 CONSTANT NTERM = 080 ;
0083 0000-D 01 CONSTANT TERM  = 000 ;
0084 0000-D 01 CONSTANT SEMANT= NTERM OR 040 ;
```

```
0086 0000-D 01 CONSTANT DC   = NTERM OR 8 ;
0087 0000-D 01 CONSTANT DC1  = NTERM OR 11;
0088 0000-D 01 CONSTANT DTC  = NTERM OR 9 ;
0089 0000-D 01 CONSTANT DV   = NTERM OR 0 ;
0090 0000-D 01 CONSTANT DV1  = NTERM OR 3 ;
0091 0000-D 01 CONSTANT ET   = NTERM OR 5 ;
0092 0000-D 01 CONSTANT LC   = NTERM OR 10;
0093 0000-D 01 CONSTANT LDM  = NTERM OR 1 ;
0094 0000-D 01 CONSTANT LF   = NTERM OR 12;
0095 0000-D 01 CONSTANT LIT  = NTERM OR 4 ;
0096 0000-D 01 CONSTANT LM   = NTERM OR 6 ;
0097 0000-D 01 CONSTANT LTC  = NTERM OR 7 ;
0098 0000-D 01 CONSTANT N    = NTERM OR 2 ;
0099 0000-D 01 &
0100 0000-D 01 CONSTANT ABRE              = TERM OR 12;
0101 0000-D 01 CONSTANT BIN               = TERM OR 4 ;
0102 0000-D 01 CONSTANT CAMPOS            = TERM OR 21;
0103 0000-D 01 CONSTANT CASO              = TERM OR 8 ;
0104 0000-D 01 CONSTANT COMENT            = TERM OR 18;
0105 0000-D 01 CONSTANT CONSTANTE         = TERM OR 0 ;
0106 0000-D 01 CONSTANT DE                = TERM OR 7 ;
0107 0000-D 01 CONSTANT DEC               = TERM OR 19;
0108 0000-D 01 CONSTANT DOIS.PT           = TERM OR 16;
0109 0000-D 01 CONSTANT ENC               = TERM OR 20;
0110 0000-D 01 CONSTANT FECHA             = TERM OR 13;
0111 0000-D 01 CONSTANT FIM               = TERM OR 10;
0112 0000-D 01 CONSTANT FIM.DE.CADEIA     = TERM OR 14;
0113 0000-D 01 CONSTANT HEX               = TERM OR 5 ;
0114 0000-D 01 CONSTANT ID                = TERM OR 3 ;
0115 0000-D 01 CONSTANT IGU               = TERM OR 17;
0116 0000-D 01 CONSTANT LITERAL           = TERM OR 6 ;
0117 0000-D 01 CONSTANT MAQUINA           = TERM OR 2 ;
0118 0000-D 01 CONSTANT POSI              = TERM OR 22;
0119 0000-D 01 CONSTANT PT.VG             = TERM OR 15;
0120 0000-D 01 CONSTANT TAM               = TERM OR 9 ;
0121 0000-D 01 CONSTANT TIPO              = TERM OR 1 ;
0122 0000-D 01 CONSTANT VG                = TERM OR 11;
0123 0000-D 01 CONSTANT QS.OUTROS         = TERM OR 23;
0124 0000-D 01 &
0125 0000-D 01 CONSTANT SEMANT0 = SEMANT OR 0 ;
0126 0000-D 01 CONSTANT SEMANT1 = SEMANT OR 1 ;
0127 0000-D 01 CONSTANT SEMANT2 = SEMANT OR 2 ;
0128 0000-D 01 CONSTANT SEMANT3 = SEMANT OR 3 ;
0129 0000-D 01 CONSTANT SEMANT4 = SEMANT OR 4 ;
0130 0000-D 01 CONSTANT SEMANT5 = SEMANT OR 5 ;
0131 0000-D 01 CONSTANT SEMANT6 = SEMANT OR 6 ;
0132 0000-D 01 CONSTANT SEMANT7 = SEMANT OR 7 ;
0133 0000-D 01 CONSTANT SEMANT8 = SEMANT OR 8 ;
0134 0000-D 01 CONSTANT SEMANT9 = SEMANT OR 9 ;
0135 0000-D 01 CONSTANT SEMANT10= SEMANT OR 10;
```

```
0136 0000-D 01 CONSTANT SEMANI11= SEMANT OR 11;
0137 0000-D 01 CONSTANT SEMANI12= SEMANT OR 12;
```

```
0139 0000-D 01 CONSTANT ERRO.SINTATICO                  = 0 ; & TERM DA PILHA DIF DO DA ENT
0140 0000-D 01 CONSTANT ERRO.ID.JA.DECL                 = 1 ; & ID DUPLICADO
0141 0000-D 01 CONSTANT ERRO.ID.NAO.DECL.OU.NAO.ESP     = 2 ; &
0142 0000-D 01 CONSTANT ERRO.ID.TIPO.NAO.ESPEC          = 3 ; & ID.TIPO NAO ESTA' NA TAB
0143 0000-D 01 CONSTANT ERRO.POS.PALAVRA                = 4 ; & POSICAO CPO NAO E' A ATUAL
0144 0000-D 01 CONSTANT ERRO.COND.EH.LIT                = 5 ; & CPO COND E' RUTULO
0145 0000-D 01 CONSTANT ERRO.MNEM.NAO.COND              = 6 ; & MNEM CASO NAO E' DA LISTA
0146 0000-D 01 CONSTANT ERRO.ID.COM.MESMO.VALOR         = 7 ; & 2 ID'S COM MESMO VALOR
0147 0000-D 01 CONSTANT ERRO.POS.MAI.TAM                = 8 ; & POSICAO MAIOR QUE TAM PALAV
0148 0000-D 01 CONSTANT ERRO.TAM.MAX.TAMANHO            = 9 ; & TAM CPO > 16
0149 0000-D 01 CONSTANT ERRO.TRUNC.ID                   = 10; & TRUNC DE ID - ADV
0150 0000-D 01 CONSTANT ERRO.TRUNC.DEC                  = 11; & TRUNC DE DEC - ADV
0151 0000-D 01 CONSTANT ERRO.TRUNC.BIN                  = 12; & TRUNC DE BIN - ADV
0152 0000-D 01 CONSTANT ERRO.TRUNC.HEX                  = 13; & TRUNC DE HEX - ADV
0153 0000-D 01 CONSTANT ERRO.ASC.INESP                  = 14; & CAR NAO E' DA GRAMATICA - A
0154 0000-D 01 CONSTANT ERRO.TAM.DIF.DEF                = 15; & TAM.CPO DIF DE TAM.DEFAULT
0155 0000-D 01 CONSTANT ERRO.BITS.DIF.TAM               = 16; & AJUSTE A DIREITA - ADV
0156 0000-D 01 CONSTANT ERRO.FIM.INESPERADO             = 17; &
0157 0000-D 01 CONSTANT ERROS.NAO.GERA.COD              = ERRO.TRUNC.ID;
0158 0000-D 01 CONSTANT E.ADV   = 2 ;
0159 0000-D 01 CONSTANT E.GRAVE = 6 ;
0160 0000-D 01 CONSTANT E.TERM  = 8 ;
```

```
LPS/500 X.06  GERADOR DE TABELA-GERADOR DE MONTADORES  L:FTAB      23/09/82 11:58 PAG:00

0162 0000-D 01 &
0163 0000-D 01 & TABELAS :
0164 0000-D 01 &
0165 0000-D 01 BYTE(*) TAB.LL1 =(1,2,2,1,10:BRANCO,                          & DV
0166 000E-D 01                  BRANCO,6,6,5,10:BRANCO,                      & LDM
0167 001C-D 01                  4:BRANCO,3,4,8:BRANCO,                       & N
0168 002A-D 01                  BRANCO,7,8,11:BRANCO,                        & DV1
0169 0038-D 01                  3:BRANCO,11,12,12,8:BRANCO,                  & LIT
0170 0046-D 01                  6:BRANCO,13,5:BRANCO,14,BRANCO,              & ET
0171 0054-D 01                  11:BRANCO,15,BRANCO,16,                      & LM
0172 0062-D 01                  2:BRANCO,18,17,10:BRANCO,                    & LTC
0173 0070-D 01                  3:BRANCO,19,4:BRANCO,20,5:BRANCO,            & DC
0174 007E-D 01                  3:BRANCO,10,5:BRANCO,9,4:BRANCO,             & DTC
0175 008C-D 01                  3:BRANCO,21,4:BRANCO,22,BRANCO,22,2:BRANCO,22,& LC
0176 009A-D 01                  8:BRANCO,23,BRANCO,24,3:BRANCO,              & DC1
0177 00A8-D 01                  3:BRANCO,25,6:BRANCO,26,3:BRANCO);           & LF
0178 00B6-D 01 &
0179 00B6-D 01 BYTE(*) TAB.SIMB.UN = (",",",","(",")",",",".",";",":","=","&");
0180 00BE-D 01 BYTE(*) TAB.PAL.RES = ("CONSTANTE",CONSTANTE,"TIPO      ",TIPO,
0181 00D2-D 01                       "TAM       ",TAM,"ENC       ",ENC,"LITERAL   ",LITER
0182 00F0-D 01                       "MAQUINA   ",MAQUINA,"CAMPOS    ",CAMPOS,
0183 0104-D 01                       "FIM       ",FIM,"POS       ",POSI,"CASO      ",
0184 0121-D 01                       CASO,"DE        ",DE,9:BRANCO,ID);
0185 0136-D 01 & TABELA DE CLASSE PARA SCANNER :
0186 0136-D 01 BYTE(*) CLASSE = (35:OUTROS,HEXADEC,BINARIO,OUTROS,SIMB.UN,OUTROS,2:SIMB
0187 0160-D 01                  2:OUTROS,SIMB.UN,3:OUTROS,10:DECIMAL,2:SIMB.UN,OUTROS,
0188 0174-D 01                  3:OUTROS,26:PAL,37:OUTROS);
0189 01B6-D 01 GLOBAL BYTE(80) .PARMS;
```

```
0191 0206-D 01 & GRAMATICA DO TEXTO DAS TABELAS DAS PRODUCOES :
0192 0206-D 01 &                                *
0193 0206-D 01 & LADO.DIR = NUM.SIMB SIMB
0194 0206-D 01 &
0195 0206-D 01 & NUM.SIMB = BYTE - NUMERO DE SIMBOLOS DO LADO DIREITO DAS PRODUCOES
0196 0206-D 01 &
0197 0206-D 01 & SIMB = TIPOSIMB (QUALSIMB!ACAO.SEMNTICA)
0198 0206-D 01 &
0199 0206-D 01 & TIPOSIMB = TERM ! NTERM - REPRESENTADO POR UM BIT
0200 0206-D 01 &
0201 0206-D 01 & TERM = 0
0202 0206-D 01 &
0203 0206-D 01 & NTERM = 1
0204 0206-D 01 &                    7
0205 0206-D 01 & QUALSIMB = BIT  - SAIDA DO SCREENER
0206 0206-D 01 &                        6
0207 0206-D 01 & ACAO.SEMANTICA = "1" BIT
0208 0206-D 01 &
0209 0206-D 01 BYTE(*) LD0.S = (17,DV,MAQUINA,ID,DOIS.PT,TAM,IGU,DEC,VG,ENC,IGU,LIT,SEM
0210 0213-D 01                  VG,CAMPOS,DC,FIM,FIM.DE.CADEIA); & P->DV MAQUINA ID : T
0211 0218-D 01                                                   & ENC=LIT [6] ,CAMPOS
0212 0218-D 01 &
0213 0218-D 01 BYTE(*) LD1.DV = (8,CONSTANTE,ID,IGU,N,SEMANTO,PT.VG,LDM,DV1);
0214 0221-D 01                  & DV -> CONSTANTE ID = N [0] ; LDM DV1
0215 0221-D 01 &
0216 0221-D 01 BYTE(*) LD2.DV = (1,DV1);
0217 0223-D 01                       & DV -> DV1
0218 0223-D 01 &
0219 0223-D 01 BYTE(*) LD3.N = (1,BIN); & N->BIN
0220 0225-D 01 &
0221 0225-D 01 BYTE(*) LD4.N = (1,HEX); & N->HEX
0222 0227-D 01 &
0223 0227-D 01 BYTE(*) LD5.LDM = (6,ID,IGU,N,SEMANTO,PT.VG,LDM); &LDM->ID = N [0] ; LDM
0224 022E-D 01 &
0225 022E-D 01 BYTE(*) LD6.LDM = (0); & LDM->EPSILON
0226 022F-D 01 &
0227 022F-D 01 BYTE(*) LD7.DV1 = (7,TIPO,ID,SEMANT1,DOIS.PT,DTC,
0228 0235-D 01                    PT.VG,LTC); & DV1->TIPO ID [1] ; DTC ; LTC
0229 0237-D 01 &
0230 0237-D 01 BYTE(*) LD8.DV1 = (0); & DV1->EPSILON
0231 0238-D 01 &
0232 0238-D 01 BYTE(*) LD9.DTC = (11,TAM,IGU,DEC,VG,ENC,IGU,LIT,SEMANT3,VG,ET,SEMANT5);
0233 0244-D 01                      &DTC->TAM = DEC,ENC = LIT [3],ET [5]
0234 0244-D 01 &
0235 0244-D 01 BYTE(*) LD10.DTC = (2,ID,SEMANT4); &DTC->ID [4]
0236 0247-D 01 &
0237 0247-D 01 BYTE(*) LD11.LIT = (2,ID,SEMANT2); & LIT->ID [2]
0238 024A-D 01 &
0239 024A-D 01 BYTE(*) LD12.LIT = (1,N); &LIT->N
0240 024C-D 01 &
```

```
0241 024C-D 01 BYTE(*) LD13.ET = (1,LITERAL); & ET->LITERAL
0242 024E-D 01 &
0243 024E-D 01 BYTE(*) LD14.ET = (5,ABRE,ID,SEMANT12,LM,FECHA); &ET-> (ID [12] LM)
0244 0254-D 01 &
0245 0254-D 01 BYTE(*) LD15.LM = (4,VG,ID,SEMANT12,LM); &LM-> , ID [12] LM
0246 0259-D 01 &
0247 0259-D 01 BYTE(*) LD16.LM = (0); &LM->EPSILON
0248 025A-D 01 &
0249 025A-D 01 BYTE(*) LD17.LTC = (6,ID,SEMANT1,DOIS.PT,DTC,
0250 025F-D 01                     PT.VG,LTC); &LTC->ID [1] ; DTC ; LTC
0251 0261-D 01 &
0252 0261-D 01 BYTE(*) LD18.LTC = (0); &LTC->EPSILON
0253 0262-D 01 &
0254 0262-D 01 BYTE(*) LD19.DC = (12,ID,DOIS.PT,POSI,IGU,DEC,SEMANT7,VG,DTC,SEMANT8,
0255 026C-D 01                    PT.VG,LC,DC1);
0256 026F-D 01                    &DC->ID : POS = DEC [7] , DTC [8] ; LC DC1
0257 026F-D 01 &
0258 026F-D 01 BYTE(*) LD20.DC = (32,CASO,ID,DOIS.PT,POSI,IGU,DEC,SEMANT7,VG,DTC,
0259 0279-D 01                    SEMANT9,DE,ID,SEMANT10,DOIS.PT,ABRE,ID,DOIS.PT,
0260 0281-D 01                    POSI,IGU,DEC,SEMANT7,VG,DTC,SEMANT8,PT.VG,LC,FECHA,
0261 028B-D 01                    PT.VG,LF,FIM,SEMANT11,LC); &DC->CASO ID : POS = DEC [
0262 0290-D 01                                               & [9] DE ID [10] : (ID : P
0263 0290-D 01                                               & DEC [7] ,DTC [8] ; LC )
0264 0290-D 01                                               & FIM [11] LC
0265 0290-D 01 &
0266 0290-D 01 BYTE(*) LD21.LC = (11,ID,DOIS.PT,POSI,IGU,DEC,SEMANT7,VG,DTC,
0267 0299-D 01                   SEMANT8,PT.VG,LC);
0268 029C-D 01                   &LC->ID : POS = DEC [7] , DTC [8] ; LC
0269 029C-D 01 &
0270 029C-D 01 BYTE(*) LD22.LC = (0); &LC->EPSILON
0271 029D-D 01 &
0272 029D-D 01 BYTE(*) LD23.DC1 = (32,CASO,ID,DOIS.PT,POSI,IGU,DEC,SEMANT7,
0273 02A5-D 01                    VG,DTC,SEMANT9,DE,ID,SEMANT10,DOIS.PT,
0274 02AC-D 01                    ABRE,ID,DOIS.PT,POSI,IGU,DEC,SEMANT7,
0275 02B3-D 01                    VG,DTC,SEMANT8,PT.VG,LC,FECHA,PT.VG,
0276 02BA-D 01                    LF,FIM,SEMANT11,LC);
0277 02BE-D 01                    &DC1->CASO ID : POS = DEC [7] ,DTC [9] DE ID [10] :
0278 02BE-D 01                    &      (ID : POS = DEC [7] , DTC [8] ) ; LF FIM [11]
0279 02BE-D 01 &
0280 02BE-D 01 BYTE(*) LD24.DC1 = (0); & DC1->EPSILON
0281 02BF-D 01 &
0282 02BF-D 01 BYTE(*) LD25.LF = (18,ID,SEMANT10,DOIS.PT,ABRE,ID,DOIS.PT,POSI,IGU,
0283 02C8-D 01                   DEC,SEMANT7,VG,DTC,SEMANT8,
0284 02CD-D 01                   PT.VG,LC,FECHA,PT.VG,LF);
0285 02D2-D 01                   &LF->ID [10] : (ID : POS = DEC [7] ,DTC [8] : LC ) ;
0286 02D2-D 01 &
0287 02D2-D 01 BYTE(*) LD26.LF = (0); & LF->EPSILON
0288 02D3-D 01 &
```

```
0290 02D3-D 01 &
0291 02D3-D 01 & TABELA DE PONTEIROS PARA AS PRODUCOES :
0292 02D3-D 01 &
0293 02D3-D 01 WORD(*) PONT.LADO.DIR = (ALD0.S,ALD1.DV,ALD2.DV,
0294 02D9-D 01                         ALD3.N,ALD4.N,ALD5.LDM,ALD6.LDM,ALD7.DV1,
0295 02E3-D 01                         ALD8.DV1,ALD9.DIC,ALD10.DIC,ALD11.LIT,
0296 02EB-D 01                         ALD12.LIT,ALD13.ET,
0297 02EF-D 01                         ALD14.ET,ALD15.LM,ALD16.LM,ALD17.LIC,ALD18.LIC,
0298 02F9-D 01                         ALD19.DC,ALD20.DC,ALD21.LC,
0299 02FF-D 01                         ALD22.LC,ALD23.DC1,ALD24.DC1,ALD25.LF,ALD26.LF)
0300 0309-D 01 & TABELA DE ENDERECO DE HASH
0301 0309-D 01 WORD(*) THASH.END = NUM.ENTR.TSIMB : VAZIO ;
0302 0B09-D 01 & TABELA DE SIMBOLOS :
0303 0B09-D 01 BYTE(*) TAB.SIMB = TAM.TAB:BRANCO;
0304 4B09-D 01 & TABELA AUX PARA LISTA DE MNEMONICOS :
0305 4B09-D 01 WORD(*) TAB.AUX = TAM.TAB.AUX : 82020;
0306 4D0B-D 01 WORD END.TAB.AUX.LIVRE = ATAB.AUX;
0307 4D0D-D 01 WORD PONT.TAB.AUX = 0;
0308 4D0F-D 01 WORD(*) ENTR.TAB.AUX REF END.TAB.AUX.LIVRE;
0309 4D0F-D 01 & AREA PARA PREENCHIMENTO DE CODIGO GERADO :
0310 4D0F-D 01 BYTE(*) AREA.COD = TAM.AREA.COD : BRANCO;
0311 5614-D 01 & SE PRIMEIRO REG :
0312 5614-D 01 BYTE(9) NOME.MAQ POS AREA.COD;
0313 5614-D 01 WORD TAM.TOT POS NOME.MAQ + 9;
0314 5614-D 01 BYTE ENC.TOT POS TAM.TOT + 2;
0315 5614-D 01 & SE CAMPO :
0316 5614-D 01 WORD POS.CPO.COD POS AREA.COD;
0317 5614-D 01 BYTE CTL.CPO.COD POS POS.CPO.COD + 2;
0318 5614-D 01 BYTE TAM.CPO.COD POS CTL.CPO.COD + 1;
0319 5614-D 01 BYTE ENC.CPO.COD POS TAM.CPO.COD + 1;
0320 5614-D 01 & SE CAMPO FIXO OU CONDICIONAL :
0321 5614-D 01 BYTE(*) CPO.FIX POS ENC.CPO.COD + 1;
0322 5614-D 01 & SE CAMPO VARIAVEL ( LITERAL OU NAO ) :
0323 5614-D 01 WORD MNEM.CAR POS ENC.CPO.COD + 1;
0324 5614-D 01 & SE CAMPO VARIAVEL NAO LITERAL :
0325 5614-D 01 BYTE(*) CPO.VAR POS MNEM.CAR + 2;
```

```
0327 5614-D 01 & CABECALHO DE COMECO DE PAGINA :
0328 5614-D 01 BYTE(*) CABEC =(" GERATAB A.03",42:BRANCO,"F:12345678 X.XX-X XX/XX/XX X
0329 566C-D 01                    " PAG:XXXX");
0330 5675-D 01 BYTE(*) NOME.CABEC POS CABEC+57;
0331 5675-D 01 BYTE(*) DATA.CABEC POS CABEC+73;
0332 5675-D 01 BYTE(*) PAG.CABEC  POS CABEC+93;
0333 5675-D 01 BYTE(*) VER.CABEC  POS CABEC+9;
0334 5675-D 01 BYTE(*) VERF.CABEC POS CABEC+66;
0335 5675-D 01 & CABECALHO DE FIM DE GERACAO :
0336 5675-D 01 BYTE(*) FIM.GERA = (" OBJETO: ",8:BRANCO,"[      V=      ] VERSAO=",
0337 569C-D 01                                 "      CODIGO=          REGISTROS");
0338 56BE-D 01 BYTE(*) NOME.FIM POS FIM.GERA+9;
0339 56BF-D 01 BYTE(*) DISCO.FIM  POS FIM.GERA+18;
0340 56BF-D 01 BYTE(*) VOL.FIM POS FIM.GERA+25;
0341 56BF-D 01 BYTE(*) COD.FIM POS FIM.GERA+53;
0342 56BF-D 01 BYTE(*) REG.FIM POS FIM.GERA+60;
0343 56BF-D 01 BYTE(*) VER.FIM POS FIM.GERA+39;
0344 56BF-D 01 BYTE(*) FIM.GERA2 =(" ** FIM GERATAB ** ADVERTENCIAS:1234 ERROS:1234 TAM
0345 56F7-D 01                      "   UA'S");
0346 56FE-D 01 BYTE(*) ERRO.FIM POS FIM.GERA2+43;
0347 56FE-D 01 BYTE(*) ADV.FIM  POS FIM.GERA2+32;
0348 56FE-D 01 BYTE(*) UA.FIM  POS FIM.GERA2+56;
0349 56FE-D 01 & CABECALHOS PADRAO LINGUAGEM COBRA
0350 56FE-D 01 BYTE(*) CABEC.FON = (" FONTE: XXXXXXXX[XXXX V=XXXXXX] VERSAO=X.XX-X CODIG
0351 5732-D 01                      "XXXXXX");
0352 5738-D 01 BYTE(*) NOME.FON POS CABEC.FON + 8 ;
0353 5738-D 01 BYTE(*) DISC.FON POS CABEC.FON + 17 ;
0354 5738-D 01 BYTE(*) VOL.FON POS CABEC.FON + 24 ;
0355 5738-D 01 BYTE(*) VER.FON POS CABEC.FON + 38 ;
0356 5738-D 01 BYTE(*) COD.FON POS CABEC.FON + 52 ;
0357 5738-D 01 BYTE(*) CABEC.EQUIP = (" EQUIPAMENTO: C500") ;
0358 574A-D 01 BYTE(*) CABEC.PARMS = (" PARAMETROS: ",80:BRANCO) ;
0359 57A7-D 01 BYTE(*) VER.TAB = 8 : BRANCO;
```

```
0361 57AF-D 01 &
0362 57AF-D 01 & VARIAVEIS LOCAIS :
0363 57AF-D 01 &
0364 57AF-D 01 WORD ENDER.TOPO.PILHA.SIMB;
0365 57B1-D 01 BYTE TOPO.PILHA.SIMB REF ENDER.TOPO.PILHA.SIMB;
0366 57B1-D 01 &
0367 57B1-D 01 WORD A,B;
0368 57B5-D 01 WORD ERRO.L;
0369 57B7-D 01 &
0370 57B7-D 01 BYTE CONT;
0371 57B8-D 01 WORD CONTA.ADV = 0 ;
0372 57BA-D 01 WORD CONTA.CAR ;
0373 57BC-D 01 WORD CONTA.ERRO=0;
0374 57BE-D 01 BYTE CONTA.LINHAS.PAG ;
0375 57BF-D 01 WORD CONTA.PAG =1;
0376 57C1-D 01 WORD NB.GRAVA ;
0377 57C3-D 01 WORD NB.GRAVA.COND;
0378 57C5-D 01 WORD NB.LIDOS ;
0379 57C7-D 01 WORD NUM.LINHAS=0;
0380 57C9-D 01 &
0381 57C9-D 01 BYTE(ESTOURO.PILHA.M1) NUM.SIMB.LADO.DIR ;
0382 58E8-D 01 WORD(ESTOURO.PILHA.M1) PILHA.ENDER.SIMB ;
0383 5B26-D 01 WORD PONT.PILHA = -1 ;
0384 5B28-D 01 &
0385 5B28-D 01 WORD(N.MAX.MNEM.CPO) PILHA.ATRIBUTOS;
0386 5D28-D 01 WORD CONT.PILHA.ATRIBUTOS=-1;
0387 5D2A-D 01 &
0388 5D2A-D 01 BYTE(N.MAX.MNEM.CPO) N.CPO = N.MAX.MNEM.CPO : 0 ;
0389 5E2A-D 01 BYTE TCOND = 0 ;
0390 5E2B-D 01 & VAR ANALISE LEXICA
0391 5E2B-D 01 WORD INIC.TOKEN;
0392 5E2D-D 01 BYTE NUM.PROD ;
0393 5E2E-D 01 BYTE QUAL.SIMB = 0 ;
0394 5E2F-D 01 WORD TAM.TOKEN ;
0395 5E31-D 01 BYTE TRUNCADO = FALSO;
0396 5E32-D 01 &
0397 5E32-D 01 BYTE(*) REG.GRAVA = ("XXXX",8:BRANCO,81:FIM.DE.REG);
0398 5E8F-D 01 BYTE(*) REG.LIDO POS REG.GRAVA + 12;
0399 5E8F-D 01 WORD PT.REG.LIDO ;
0400 5E91-D 01 BYTE CAR.ATUAL REF PT.REG.LIDO;
0401 5E91-D 01 WORD UL.FONTE = UL.ENTR ;
0402 5E93-D 01 &
0403 5E93-D 01 BYTE(*) REG.ERRO =("MENSAGEM XXX",80:BRANCO);
```

```
0405 5EEF-D 01 & VAR HASH
0406 5EEF-D 01 WORD CONTA.COLISAO=0;
0407 5EF1-D 01 WORD END.TAB;
0408 5EF3-D 01 WORD ENTR.TAB;
0409 5EF5-D 01 BYTE(9) GID ;
0410 5EFE-D 01 BYTE I;
0411 5EFF-D 01 BYTE ID.DECL;
0412 5F00-D 01 WORD N.ENTR=0;
0413 5F02-D 01 WORD PT.ID;
0414 5F04-D 01 BYTE(*) CARAC.ID REF PT.ID;
0415 5F04-D 01 WORD REC.XOR;
0416 5F06-D 01 BYTE(*) CAR POS 0;
0417 5F06-D 01 WORD PALAVRA;
0418 5F08-D 01 & VAR ANALISE SEMANTICA
0419 5F08-D 01 BYTE CPO.COND = FALSO;
0420 5F09-D 01 WORD END.CPO;
0421 5F0B-D 01 WORD END.CPO.COND;
0422 5F0D-D 01 WORD DESCR.BITS.CPO.COND;
0423 5F0F-D 01 BYTE NAO.GERA.COD = FALSO;
0424 5F10-D 01 WORD POS.ANTERIOR;
0425 5F12-D 01 WORD POS.COND;
0426 5F14-D 01 WORD TAM.ANTERIOR;
0427 5F16-D 01 WORD TAM.COND;
0428 5F18-D 01 WORD TAM.MAX.PAL;
0429 5F1A-D 01 &
0430 5F1A-D 01 BYTE (*) ENTRADA REF END.TAB;
0431 5F1A-D 01 BYTE (9) NOME POS ENTRADA;
0432 5F1A-D 01 WORD PROX.ENDER POS NOME+9;
0433 5F1A-D 01 BYTE CTL.CPO.TAB POS PROX.ENDER+2;
0434 5F1A-D 01 & SE ID.CONSTANTE :
0435 5F1A-D 01 BYTE NUM.BITS POS CTL.CPO.TAB+1;
0436 5F1A-D 01 WORD DESCR.BITS POS NUM.BITS+1;
0437 5F1A-D 01 & SE ID.TIPO :
0438 5F1A-D 01 BYTE TAM.CPO.TAB POS CTL.CPO.TAB+1;
0439 5F1A-D 01 BYTE ENC.CPO.TAB POS TAM.CPO.TAB+1;
0440 5F1A-D 01 WORD PONT.LISTA POS ENC.CPO.TAB+1;
0441 5F1A-D 01 WORD END.TAB.AUX.OCUP;
0442 5F1C-D 01 WORD(*) ENTR.TAB.AUX.OCUP REF END.TAB.AUX.OCUP;
0443 5F1C-D 01 WORD NUM.ENTR POS ENTR.TAB.AUX.OCUP;
0444 5F1C-D 01 BYTE(*) NOME.ARQ.SAI = 8:BRANCO;
0445 5F24-D 01 & ANALISE DE PARAMETROS :
0446 5F24-D 01 BYTE(*) REC.PARMS = TAM.MAX.REG.ENTR : BRANCO ;
0447 5F74-D 01 BYTE(9) NOM.ARGENT  POS REC.PARMS ;
0448 5F74-D 01 WORD TIPO.UD POS NOM.ARGENT+9 ;
0449 5F74-D 01 WORD UNID POS TIPO.UD+2 ;
0450 5F74-D 01 WORD VOL POS UNID+2 ;
0451 5F74-D 01 WORD TAM.PARMS POS .PARMS;
0452 5F74-D 01 BYTE(*) CAD.PARMS POS TAM.PARMS+2 ;
0453 5F74-D 01 WORD FSV.SE.EXISTE = FALSO ;
0454 5F76-D 01 WORD COND.FIM = 0 ;
```

```
0456 5F78-D 01 &
0457 5F78-D 01 & ROTINAS EXTERNAS :
0458 5F78-D 01 &
0459 5F78-D 01 PROCEDURE TEMPOMAX(WORD ERESP); SUPERVISOR 7;
0460 5F78-D 01 PROCEDURE FIMPROG(WORD ECOND); SUPERVISOR 3;
0461 5F78-D 01 WORD PROCEDURE ALLOCATE(WORD CONTR,TIP,UNI,V,ENOM,PRM); EXTERNAL;
0462 5F78-D 01 WORD PROCEDURE CREATEDISK(WORD A,B,C,D,E,F,G,H); EXTERNAL;
0463 5F78-D 01 WORD PROCEDURE FILEECB(WORD CONTR,ENDER,DESLOC,TAM); EXTERNAL;
0464 5F78-D 01 WORD PROCEDURE FILENAME(WORD CONTR,ENDER); EXTERNAL;
0465 5F78-D 01 WORD PROCEDURE FILEUNIT(WORD CONTR,ENDER); EXTERNAL;
0466 5F78-D 01 WORD PROCEDURE FILEVOL (WORD CONTR,ENDER); EXTERNAL;
0467 5F78-D 01 WORD PROCEDURE TOTREC(WORD CONTR); EXTERNAL;
0468 5F78-D 01 WORD PROCEDURE VERSION (WORD CONTR,ENDVER); EXTERNAL;
0469 5F78-D 01 WORD PROCEDURE VERSIONED(WORD CONTR,END.VER); EXTERNAL;
```

```
0471 5F78-D 01 &
0472 5F78-D 01 & ROTINAS LOCAIS :
0473 5F78-D 01 &
0474 5F78-D 01 &
0475 5F78-D 01 PROCEDURE POP.TOPO;
0476 5F78-D 01 &
0477 5F78-D 01 & FUNCAO : DESEMPILHAR O SIMBOLO DO TOPO DA PILHA. SE ESSE SIMBOLO FOR
0478 5F78-D 01 &          O ULTIMO DA PRODUCAO, TRAZ A PRODUCAO DE BAIXO.
0479 5F78-D 01 &
0480 5F78-D 01 & ROTINAS QUE A CHAMAM : ANALISADOR.SINTATICO
0481 5F78-D 01 &                        ERRO
0482 5F78-D 01 &                        ACAO.SEMANTICA
0483 5F78-D 01 &
0484 5F78-D 01 BEGIN
0485 0000-P 02 IF (NUM.SIMB.LADO.DIR(PONT.PILHA):=!-1) = 0
0486 0008-P 02 THEN
0487 000E-P 02       & ATUAL TOPO E' ULTIMO DE SUA PRODUCAO
0488 000E-P 02       BEGIN
0489 000E-P 03       ENDER.TOPO.PILHA.SIMB:=PILHA.ENDER.SIMB(PONT.PILHA:=!-1);
0490 0020-P 03       RETURN;
0491 0022-P 03       END;
0492 0022-P 02 PILHA.ENDER.SIMB(PONT.PILHA):=ENDER.TOPO.PILHA.SIMB:=I+1;
0493 0034-P 02 END POP.TOPO;
0494 5F78-D 01 &
0495 5F78-D 01 PROCEDURE GRAVA.COD(WORD NUM.BYTES);
0496 5F78-D 01 &
0497 5F78-D 01 & FUNCAO : GRAVAR O CODIGO GERADO.
0498 5F78-D 01 &
0499 5F78-D 01 & ROTINAS QUE A CHAMAM : PREENCHE.LISTA.MNEM
0500 5F78-D 01 &                        ACAO.SEMANTICA
0501 5F78-D 01 &
0502 5F78-D 01 BEGIN
0503 003F-P 02 WRITE(UL.SAI,AAREA.COD,NUM.BYTES);
0504 0050-P 02 END GRAVA.COD;
```

```
0506 5F7A-D 01 PROCEDURE ERRO(WORD NUM.ERRO);
0507 5F7A-D 01 &
0508 5F7A-D 01 & FUNCAO : APONTAR ONDE E QUAL ERRO FOI DETECTADO, CONTAR NUMERO DE ERRO
0509 5F7A-D 01 &          VERTENCIAS.
0510 5F7A-D 01 &
0511 5F7A-D 01 & ROTINAS QUE A CHAMAM : SE.TRUNCADO
0512 5F7A-D 01 &                        ANALISADOR.LEXICO
0513 5F7A-D 01 &                        ACAO.SEMANTICA
0514 5F7A-D 01 &                        PREENCHE.LISTA.MNEM
0515 5F7A-D 01 &                        ANALISADOR.SINTATICO
0516 5F7A-D 01 &
0517 5F7A-D 01 BEGIN
0518 005A-P 02 PROCEDURE PEGA.PROX.TOKEN;
0519 5F7C-D 02   EXTERNAL;
0520 5F7C-D 02 PROCEDURE IMPR.CABEC;
0521 5F7C-D 02   EXTERNAL;
0522 5F7C-D 02 FILL(AREG.ERRO+12,80,BRANCO);
0523 0070-P 02 REG.ERRO(11+PT.REG.LIDO-AREG.LIDO):=SETA;
0524 0082-P 02 BINASCII(NUM.ERRO,AREG.ERRO+9,3);
0525 0098-P 02 WRITE(CUL.LIST,AREG.ERRO,92); & IMPRIME NUMERO E POSICAO DO ERRO
0526 00A6-P 02 IF (CONTA.LINHAS.PAG:=I+1) > NLINPAG
0527 00AA-P 02 THEN
0528 00B2-P 02       IMPR.CABEC;
0529 00B6-P 02 IF NUM.ERRO = ERRO.SINTATICO
0530 00B6-P 02 THEN
0531 00BC-P 02       & PANIC MODE
0532 00BC-P 02       BEGIN
0533 00BC-P 03       REPEAT
0534 00BC-P 03                POP.TOPO
0535 00BC-P 03       UNTIL TOPO.PILHA.SIMB = PT.VG
0536 00C0-P 03             OR
0537 00C6-P 03             TOPO.PILHA.SIMB = CAMPOS;
0538 00D4-P 03       REPEAT
0539 00D4-P 03                PEGA.PROX.TOKEN
0540 00D4-P 03       UNTIL QUAL.SIMB = TOPO.PILHA.SIMB;
0541 00E2-P 03       END;
0542 00E2-P 02 IF NUM.ERRO < ERROS.NAO.GERA.COD
0543 00E2-P 02 THEN
0544 00EA-P 02       & CONTA NUMERO DE ERROS
0545 00EA-P 02       BEGIN
0546 00EA-P 03       NAO.GERA.COD:=VERO;
0547 00EE-P 03       CONTA.ERRO:=I+1;
0548 00F2-P 03       COND.FIM:=F.GRAVE;
0549 00F6-P 03       END
0550 00F6-P 02 ELSE
0551 00F8-P 02       & CONTA NUMERO DE ADVERTENCIAS
0552 00F8-P 02       BEGIN
0553 00F8-P 03       CONTA.ADV:=I+1;
0554 00FC-P 03       IF NOT NAO.GERA.COD
0555 00FC-P 03       THEN
```

```
0556 0104-P 03              COND.FIM:=E.ADV;
0557 0108-P 03          END;
0558 0108-P 02 END ERRO;
```

```
0560 5F7C-D 01 PROCEDURE PREENCHE.LISTA.MNEM(BYTE ACA09);
0561 5F7C-D 01 &
0562 5F7C-D 01 & FUNCAO : PREENCHE A AREA DE CODIGO RELATIVA A LISTA DE MNEMONICOS
0563 5F7C-D 01 &          ASSOCIADOS A DETERMINADO CAMPO. O MNEMONICO E' COLOCADO
0564 5F7C-D 01 &          NA POSICAO RELATIVA AO SEU VALOR.
0565 5F7C-D 01 &
0566 5F7C-D 01 & ROTINA QUE A CHAMA : ACAO.SEMANTICA
0567 5F7C-D 01 &
0568 5F7C-D 01 BEGIN
0569 0112-P 02 TAM.ANTERIOR:=TAM.CPO.COD:=TAM.CPO.TAB;
0570 0122-P 02 ENC.CPO.COD:=ENC.CPO.TAB;
0571 012C-P 02 IF CPO.COND
0572 012C-P 02 THEN
0573 0132-P 02       & CAMPO ESTA' DENTRO DE UM 'CASO', LOGO E' VARIAVEL
0574 0132-P 02       BEGIN
0575 0132-P 03       END.CPO:=ACPO.VAR;
0576 013A-P 03       MNEM.CAR:=DESCR.BITS.CPO.COND;
0577 0142-P 03       CTL.CPO.COD:=! OR CPO.COD.VAR;
0578 014C-P 03       NB.GRAVA:=7;
0579 0150-P 03       N.CPO(DESCR.BITS.CPO.COND):=!+1;
0580 0158-P 03       END
0581 0158-P 02 ELSE & CAMPO E' FIXO
0582 015A-P 02       BEGIN
0583 015A-P 03       END.CPO:=ACPO.FIX;
0584 0162-P 03       NB.GRAVA:=5;
0585 0166-P 03       FOR A:=0 TO TCOND
0586 016A-P 03       DO
0587 0178-P 03            N.CPO(A):=!+1;
0588 018C-P 03       END;
0589 018C-P 02 IF CTL.CPO.TAB = CPO.TAB.LIT
0590 0192-P 02 THEN
0591 0198-P 02       & CAMPO LITERAL
0592 0198-P 02       IF ACA09
0593 0198-P 02       THEN
0594 019E-P 02            & CAMPO CONDICIONAL NAO PODE SER LITERAL
0595 019E-P 02            ERRO(ERRO.COND.EM.LIT)
0596 01A6-P 02       ELSE
0597 01A8-P 02            CTL.CPO.COD:=! OR CPO.COD.LIT
```

```
0599 01A8-P 02 ELSE
0600 01B4-P 02       BEGIN
0601 01B4-P 03       A:=1;
0602 01B8-P 03       B:=0;
0603 01BC-P 03       REPEAT
0604 01BC-P 03            & DETERMINA O VALOR 2 ELEVADO A TAM.CPO.COND
0605 01BC-P 03            BEGIN
0606 01BC-P 04            A:=!*2;
0607 01C6-P 04            B:=!+1;
0608 01CA-P 04            END
0609 01CA-P 03       UNTIL B = TAM.CPO.COD;
0610 01D4-P 05       FILL(END.CPO,(A:=9*A),BRANCO); & B REPRESENTA O TAMANHO TOTAL DA LI
0611 01EA-P 03                                      & QUENCIAL DE MNEMONICOS A SER PASSA
0612 01EA-P 03                                      & O REGISTRO DE CAMPO(FIXO OU VARIA
0613 01EA-P 03       NB.GRAVA:=!+B;
0614 01F2-P 03       END.TAB.AUX.OCUP:=PONT.LISTA;
0615 01FC-P 03       PONT.TAB.AUX:=0;
0616 0200-P 03       REPEAT
0617 0200-P 03            & PREENCHIMENTO DA LISTA SEQUENCIAL DE MNEMONICOS.
0618 0200-P 03            & CADA MNEMONICO OCUPA A POSICAO QUE SEU VALOR INDICA.
0619 0200-P 03            BEGIN
0620 0200-P 04            END.TAB:=ENTR.TAB.AUX.OCUP(PONT.TAB.AUX:=!+1);
0621 0212-P 04            IF NUM.BITS <> TAM.CPO.COD
0622 0218-P 04            THEN
0623 0220-P 04                 BEGIN
0624 0220-P 05                 ERRO(ERRO.BITS,DIF.TAM);
0625 0228-P 05                 IF NUM.BITS > TAM.CPO.COD
0626 022E-P 05                 THEN
0627 0236-P 05                      & AJUSTE DE BITS A DIREITA
0628 0236-P 05                      DESCR.BITS:=!
0629 023C-P 05                                 AND
0630 023C-P 05                                 (8FFFF SHR (EXPS (16-TAM.CPO.COD)));
0631 0250-P 05                 END;
0632 0250-P 04            IF CAR(A:=END.CPO+DESCR.BITS*9) <> BRANCO
0633 0262-P 04            THEN
0634 026A-P 04                 & DOIS MNEMONICOS COM VALORES IDENTICOS
0635 026A-P 04                 ERRO(ERRO.ID.COM.MESMO.VALOR);
0636 0272-P 04            MBT(ANOME,A,9);
0637 0280-P 04            END
0638 0280-P 03       UNTIL PONT.TAB.AUX = NUM.ENTR;
0639 028A-P 03       END;
0640 028A-P 02 GRAVA.COD(NB.GRAVA);
0641 0294-P 02 END PREENCHE.LISTA.MNEM;
```

```
0643 5F7F-D 01 PROCEDURE ACAO.SEMANTICA(BYTE ACAO);
0644 5F7F-D 01 &
0645 5F7F-D 01 & FUNCAO : FAZER AS ACOES SEMANTICAS CONFORME MODIFICACAO DA GRAMATICA
0646 5F7F-D 01 &          PARA ANALISE DOS ATRIBUTOS HERDADOS E SINTETIZADOS.
0647 5F7F-D 01 &
0648 5F7F-D 01 BEGIN
0649 029F-P 02 POP.TOPO;
0650 02A2-P 02 CASE ACAO OF
0651 02AC-P 02 BEGIN
0652 02AC-P 03       & ACAO 0 : DV -->> CONSTANTE ID = N [0] ; LDM DV1
0653 02AC-P 03                  & LDM -->> ID = N [0] ; LDM
0654 02AC-P 03       BEGIN
0655 02AC-P 04       IF ID.DECL
0656 02AC-P 04       THEN
0657 02B2-P 04            & ID JA' DECLARADO
0658 02B2-P 04            ERRO(ERRO.ID.JA.DECL);
0659 02BA-P 04       CTL.CPO.TAB:=CPO.TAB.CTE; & ID E' CONSTANTE
0660 02C4-P 04       & PREENCHE TABELA DE SIMBOLOS
0661 02C4-P 04       DESCR.BITS:=PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS);
0662 02D4-P 04       NUM.BITS:=PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS:=!-1);
0663 02E8-P 04       CONT.PILHA.ATRIBUTOS:=!-2 & DESEMPILHA TB END.TAB
0664 02E8-P 04       END;
0665 02F0-P 03       & ACAO 1 : DV1 -->> TIPO ID [1] ; DTC ; LTC
0666 02F0-P 03                  & LTC -->> ID [1] : DTC ; LTC
0667 02F0-P 03       BEGIN
0668 02F0-P 04       IF ID.DECL
0669 02F0-P 04       THEN
0670 02F6-P 04            & ID JA' DECLARADO ANTERIORMENTE
0671 02F6-P 04            ERRO(ERRO.ID.JA.DECL);
0672 02FE-P 04       CTL.CPO.TAB:=CPO.TAB.TIP; & ID E' TIPO
0673 0308-P 04       END;
0674 030A-P 03       & ACAO 2 ; LIT -->> ID [2]
0675 030A-P 03       BEGIN
0676 030A-P 04       IF (NOT ID.DECL)  & ID NAO FOI DECLARADO ANTERIORMENTE
0677 0310-P 04          OR
0678 0310-P 04          (CTL.CPO.TAB <> CPO.TAB.CTE) & JA' DECL, MAS NAO E' ID.CONST
0679 0318-P 04       THEN
0680 031E-P 04            ERRO(ERRO.ID.NAO.DECL.OU.NAO.ESP);
0681 0326-P 04       PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS):=NUM.BITS;
0682 0338-P 04       PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS:=!+1):=DESCR.BITS;
0683 034C-P 04       END;
0684 034E-P 03       & ACAO 3 : DTC -->> TAM = DEC , ENC = LIT [3] , ET [5]
0685 034E-P 03       BEGIN
0686 034E-P 04       END.TAB:=PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS-3);
0687 0360-P 04       ENC.CPO.TAB:=PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS);
0688 0370-P 04       TAM.CPO.TAB:=PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS-2);
0689 0384-P 04       IF TAM.CPO.TAB <> PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS-1)
0690 0392-P 04       THEN
0691 039A-P 04            & NUMERO DE BITS DE TAM.CPO DIFERNTE DE ENC.CPO : ADVERTENCIA
0692 039A-P 04            ERRO(ERRO.TAM.DIF.DEF);
```

```
0693 03A2-P 04          CONT_PILHA_ATRIBUTOS:=0;
0694 03A6-P 04          END;
```

```
0696 03A8-P 03          & ACAO 4 : DTC --> ID [4]
0697 03A8-P 03          BEGIN
0698 03A8-P 04          IF (NOT ID.DECL)                    & ID NAO DECLARADO
0699 03AE-P 04             OR
0700 03AE-P 04             (NOT BIT(ACTL.CPO.TAB,0)) & ID NAO E' ID.TIPO
0701 03BE-P 04          THEN
0702 03C4-P 04             ERRO(ERRO.ID.NAO.DECL.OU.NAO.ESP);
0703 03CC-P 04          IF TAM.CPO.TAB = BRANCO
0704 03D2-P 04          THEN
0705 03D8-P 04                & ID NAO ESPECIFICADO
0706 03D8-P 04             ERRO(ERRO.ID.TIPO.NAO.ESPEC);
0707 03E0-P 04          END.TAB:=PILHA.ATRIBUTOS(1);
0708 03E8-P 04          MBT(END.TAB+10,PILHA.ATRIBUTOS(0)+10,4); & TRANSFERE INFO DE ID.TI
0709 03FA-P 04                                                   & PARA ID.TIPO(0)
0710 03FA-P 04          CONT.PILHA.ATRIBUTOS:=-1;
0711 03FE-P 04          END;
0712 0400-P 03          & ACAO 5 : DTC --> TAM = DEC , ENC = LIT [3] , FT [5]
0713 0400-P 03          BEGIN
0714 0400-P 04          END.TAB:=PILHA.ATRIBUTOS(0);
0715 0408-P 04          IF CONT.PILHA.ATRIBUTOS = 0
0716 0408-P 04          THEN
0717 040E-P 04                & ID.ENDER E' DE ID.TIPO LITERAL
0718 040E-P 04             CTL.CPO.TAB:=CPO.TAB.LIT
0719 0414-P 04          ELSE
0720 041A-P 04             BEGIN
0721 041A-P 05             IF TAM.CPO.TAB > TAM.MAX.TAM
0722 0420-P 05             THEN
0723 0426-P 05                   & TAM CAMPO > QUE MAX PERMITIDO
0724 0426-P 05                ERRO(ERRO.TAM.MAX.TAMANHO);
0725 042E-P 05             PONT.LISTA:=END.TAB.AUX.LIVRE;
0726 0438-P 05             PONT.TAB.AUX:=0;
0727 043C-P 05             REPEAT
0728 043C-P 05                   & PREENCHIMENTO DA TABELA AUXILIAR DE LISTA DE MNEMONI
0729 043C-P 05                   BEGIN
0730 043C-P 06                   ENTR.TAB.AUX(PONT.TAB.AUX:=!+1):=
0731 0446-P 06                   PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS);
0732 0454-P 06                   CONT.PILHA.ATRIBUTOS:=!-1;
0733 0458-P 06                   IF PONT.TAB.AUX = TAM.TAB.AUX
0734 0458-P 06                   THEN
0735 0462-P 06                      BEGIN
0736 0462-P 07                      DISPLAY(A"ESTOURO DA TABELA AUXILIAR",26);
0737 046E-P 07                      TERMINATE;
0738 0472-P 07                      END;
0739 0472-P 06                   END
0740 0472-P 05             UNTIL CONT.PILHA.ATRIBUTOS = 0;
0741 0478-P 05             ENTR.TAB.AUX(0):=PONT.TAB.AUX;
0742 0480-P 05             END.TAB.AUX.LIVRE:=!+2*PONT.TAB.AUX+2;
0743 0490-P 05             END;
0744 0490-P 04          CONT.PILHA.ATRIBUTOS:=-1;
0745 0494-P 04          END;
```

```
0747 0496-P 03          & ACAO 6 : ... MAQUINA ID : TAM=DEC,ENC=LIT [6] ,...
0748 0496-P 03          BEGIN
0749 0496-P 04          IF CAR(PILHA.ATRIBUTOS(0)+9) <> BRANCO
0750 049C-P 04          THEN
0751 04A4-P 04              & ID JA DECL
0752 04A4-P 04              ERRO(ERRO.ID.JA.DECL);
0753 04AC-P 04          MBT(PILHA.ATRIBUTOS(0),ANOME.MAQ,9); & LEVA ID.MAQ P/ PRIM REG DO
0754 04BA-P 04          MBT(ANOME.MAQ,ANOME.ARQ.SAI,8);
0755 04C8-P 04          ENC.TOT:=PILHA.ATRIBUTOS(3) AND 00FF;
0756 04D2-P 04          TAM.MAX.PAL:=TAM.TOT:=PILHA.ATRIBUTOS(1);
0757 04DE-P 04          IF PILHA.ATRIBUTOS(2) <> 8
0758 04DE-P 04          THEN
0759 04E6-P 04              & NUM BITS DIFER TAM : ADV
0760 04E6-P 04              ERRO(ERRO.TAM.DIF.DEF);
0761 04EE-P 04          TAM.ANTERIOR:=POS.ANTERIOR:=0;
0762 04F6-P 04          IF (CREATEDISK(ESV.SE.EXISTE OR NAO.CANC OR UL.SAI,
0763 0502-P 04                         DEFAULT.E.S,DEFAULT.E.S,DEFAULT.E.S,
0764 050E-P 04                         ANOME.MAQ,SEQL,DEFAULT.E.S,DEFAULT.E.S))=JA.EXISTE
0765 0530-P 04          THEN
0766 0536-P 04              & ARQ DE SAIDA JA' EXISTE E NAO FOI MANDADO ESVAZIAR
0767 0536-P 04              BEGIN
0768 0536-P 05              DISPLAY(A"ARQUIVO DE SAIDA JA EXISTE. FIM DE PROGRAMA.",44);
0769 0542-P 05              COND.FIM:=E.TERM;
0770 0546-P 05              FIMPROG(ACOND.FIM);
0771 054E-P 05              END;
0772 054F-P 04          GRAVA.COD(12); & GRAVA PRIM REG DO CODIGO-OBJETO
0773 0556-P 04          CONT.PILHA.ATRIBUTOS:=-1;
0774 055A-P 04          END;
0775 055C-P 03          & ACAO 7 : ... ID : POS = DEC [7] , DTC ...
0776 055C-P 03          BEGIN
0777 055C-P 04          IF ID.DECL
0778 055C-P 04          THEN
0779 0562-P 04              ERRO(ERRO.ID.JA.DECL);
0780 056A-P 04          CTL.CPO.TAB:=CPO.TAB.TIP;
0781 0574-P 04          CTL.CPO.COD:=CPO.COD.FIX;
0782 0578-P 04          POS.CPO.COD:=PILHA.ATRIBUTOS(1);
0783 0580-P 04          CONT.PILHA.ATRIBUTOS:=0;
0784 0584-P 04          IF POS.CPO.COD <> POS.ANTERIOR + TAM.ANTERIOR
0785 0584-P 04          THEN
0786 0592-P 04              & POSICAO INDEVIDA DE CAMPO
0787 0592-P 04              ERRO(ERRO.POS.PALAVRA);
0788 059A-P 04          POS.ANTERIOR:=POS.CPO.COD;
0789 05A2-P 04          IF POS.CPO.COD >= TAM.MAX.PAL
0790 05A2-P 04          THEN
0791 05AC-P 04              & POSICAO DE CAMPO > TAM PALAVRA
0792 05AC-P 04              ERRO(ERRO.POS.MAT.TAM);
0793 05B4-P 04          END;
0794 05B6-P 03          & ACAO 8 ; ...ID : POS = DEC [7] , DTC [8] ; LC ...
0795 05B6-P 03          PREENCHE.LISTA.MNEM(FALSO);
```

```
0797 05C0-P 03          & ACAO 9 :  ...ID : POS = DEC [7] , DTC [9] DE ID [10] ...
0798 05C0-P 03          BEGIN
0799 05C0-P 04          CTL.CPO.COD:=! OR CPO.COD.COND;
0800 05CA-P 04          TCOND:=TAM.COND:=TAM.CPO.TAB;
0801 05DA-P 04          FOR A:=0 TO ICOND-1
0802 05DE-P 04          DO
0803 05F0-P 04             & NUMERO DE NUMEROS DE CAMPOS
0804 05F0-P 04             N.CPO(A+1):=N.CPO(A);
0805 060E-P 04          PREENCHE.LISTA.MNEM(VERO);
0806 0616-P 04          POS.COND:=POS.CPO.COD;
0807 061E-P 04          END.CPO.COND:=AAREA.COD+TAM.AREA.COD-NB.GRAVA-10;
0808 0630-P 04          MBT(AAREA.COD,END.CPO.COND,NB.GRAVA); & ARMAZENA TODO O REG COM CA
0809 0640-P 04          NB.GRAVA.COND:=NB.GRAVA;
0810 0648-P 04          END;
0811 064A-P 03          & ACAO 10 :  ...POS=DEC [7] ,DTC [9] DE ID [10] : ( ...
0812 064A-P 03          BEGIN
0813 064A-P 04          IF (NOT ID.DECL)                    & ID NAO DECLARADO
0814 0650-P 04             OR
0815 0650-P 04             (CTL.CPO.TAB <> CPO.TAB.CIE) & ID NAO CONSTANTE
0816 0658-P 04          THEN
0817 065E-P 04             ERRO(ERRO.ID.NAO.DFCL.OU.NAO.ESP);
0818 0666-P 04          MBT(PILHA.ATRIBUTOS(0),END.CPO.COND+NB.GRAVA.COND,9);
0819 0678-P 04          A:=5;B:=0;
0820 0680-P 04          WHILE COMPARE(END.CPO.COND+NB.GRAVA.COND,9,END.CPO.COND+A,9) <> 0
0821 0698-P 04          DO
0822 069C-P 04             & BUSCA SEQUENCIAL DO ID DO CAMPO COND QUE SINALIZA LISTA DE CA
0823 069C-P 04             BEGIN
0824 069C-P 05             B:=!+1;
0825 06A0-P 05             A:=!+9;
0826 06A4-P 05             END;
0827 06A6-P 04          IF B =((NB.GRAVA.COND-5)/9)
0828 06B2-P 04          THEN
0829 06B8-P 04             & ID SINALIZADOR NAO ESPERADO NA LISTA DE MNEM DE COND
0830 06B8-P 04             ERRO(ERRO.MNEM.NAO.COND);
0831 06C0-P 04          CPO.COND:=VERO;
0832 06C4-P 04          DESCR.BITS.CPO.COND:=B;
0833 06CC-P 04          CONT.PILHA.ATRIBUTOS:=-1;
0834 06D0-P 04          TAM.ANTERIOR:=TAM.COND;
0835 06D8-P 04          POS.ANTERIOR:=POS.COND;
0836 06E0-P 04          END;
0837 06E2-P 03          & ACAO 11 : ...DTC [8] ; LC ) ; LF FIM [11] PF
0838 06E2-P 03          CPO.COND:=FALSO;
0839 06E8-P 03          & ACAO 12 : ET --> ( ID [12] LM )
0840 06E8-P 03          IF NOT ID.DECL
0841 06E8-P 03          THEN
0842 06F0-P 03             ERRO(ERRO.ID.NAO.DECL.OU.NAO.ESP);
0843 06FA-P 03 END;
0844 0718-P 02 END ACAO.SEMANTICA;
0845 5FC8-D 01 PROCEDURE SF.TRUNCADO(BYTE NUM.ERRO);
0846 5FC8-D 01 &
```

```
0847 5FC8-D 01 & FUNCAO : ENVIAR AVISO DE TRUNCAMENTO DE CADEIA
0848 5FC8-D 01 &
0849 5FC8-D 01 & ROTINA QUE A CHAMA : ANALISADOR LEXICO
0850 5FC8-D 01 &
0851 5FC8-D 01 BEGIN
0852 0722-P 02 IF TRUNCADO
0853 0722-P 02 THEN
0854 0728-P 02       BEGIN
0855 0728-P 03       ERRO(NUM.ERRO);
0856 0732-P 03       TRUNCADO:=FALSO;
0857 0736-P 03       END;
0858 0736-P 02 END SE.TRUNCADO;
```

```
0860 5FC9-D 01 BYTE PROCEDURE DIGITO;
0861 5FC9-D 01 &
0862 5FC9-D 01 & FUNCAO : RECONHECEDOR DE DIGITOS (0 A 9).
0863 5FC9-D 01 &
0864 5FC9-D 01 BEGIN
0865 073A-P 02 DIGITO:= CONTA.CAR < NB.LIDOS
0866 073A-P 02             AND
0867 0742-P 02             CLASSE(CAR.ATUAL) = DECIMAL;
0868 0756-P 02 END DIGITO;
0869 5FCA-D 01 PROCEDURE AVANCA.PT;
0870 5FCA-D 01 &
0871 5FCA-D 01 & FUNCAO : INCREMENTA PONTEIROS DE POSICAO DE REGISTRO LIDO.
0872 5FCA-D 01 &
0873 5FCA-D 01 BEGIN
0874 075E-P 02 TAM.TOKEN:=!+1;
0875 0762-P 02 PT.REG.LIDO:=!+1;
0876 0766-P 02 CONTA.CAR:=!+1;
0877 076A-P 02 END AVANCA.PT;
0878 5FCA-D 01 PROCEDURE COME.BRANCOS;
0879 5FCA-D 01 &
0880 5FCA-D 01 & FUNCAO : PULAR CARACTER BRANCO.
0881 5FCA-D 01 BEGIN
0882 076C-P 02 WHILE CAR.ATUAL = BRANCO
0883 076C-P 02 DO
0884 0774-P 02     AVANCA.PT;
0885 077A-P 02 END COME.BRANCOS;
0886 5FCA-D 01 BYTE PROCEDURE LETRA;
0887 5FCA-D 01 &
0888 5FCA-D 01 & FUNCAO : RECONHECEDOR DE LETRAS.
0889 5FCA-D 01 &
0890 5FCA-D 01 BEGIN
0891 077E-P 02 LETRA:=CONTA.CAR < NB.LIDOS
0892 077E-P 02             AND
0893 0786-P 02             CLASSE(CAR.ATUAL) = PAL;
0894 0798-P 02 END LETRA;
```

```
0896 5FCB-D 01 WORD PROCEDURE BITBIN(WORD ECAD,TCAD);
0897 5FCB-D 01 &
0898 5FCB-D 01 & FUNCAO : TRANSFORMAR CADEIAS BINARIAS DE ATE' 16 BITS NUMA PALAVRA.
0899 5FCB-D 01 &
0900 5FCB-D 01 & ROTINA QUE A CHAMA : ANALISADOR.LEXICO
0901 5FCB-D 01 &
0902 5FCB-D 01 BEGIN
0903 07AC-P 02 PALAVRA:=0;
0904 07B0-P 02 WHILE TCAD > 0
0905 07B0-P 02 DO
0906 07B6-P 02     BEGIN
0907 07B6-P 03     IF CAR(ECAD)
0908 07BA-P 03     THEN
0909 07C0-P 03          & CARACTER E' "1"
0910 07C0-P 03          SET(@PALAVRA,16-TCAD);
0911 07D0-P 03     ECAD:=!+1;
0912 07D4-P 03     TCAD:=!-1;
0913 07D8-P 03     END;
0914 07DA-P 02 BITBIN:=PALAVRA;
0915 07E2-P 02 END BITBIN;
0916 5FD1-D 01 PROCEDURE PREENCHE.TSEQ.SIMB(WORD ENDER);
0917 5FD1-D 01 &
0918 5FD1-D 01 & FUNCAO : PREENCHER O CAMPO DE NOME DA TABELA DE SIMBOLOS.
0919 5FD1-D 01 &
0920 5FD1-D 01 & ROTINA QUE A CHAMA : HASH
0921 5FD1-D 01 &
0922 5FD1-D 01 BEGIN
0923 07F0-P 02 WORD CONTEUDO REF ENDER;
0924 5FD3-D 02 MBI(INIC.TOKEN,CONTEUDO:=@TAB.SIMB+TAM.ENTR.TAB*N.ENTR,TAM.TOKEN);
0925 0810-P 02 IF (N.ENTR:=!+1) = NUM.ENTR.TSIMB
0926 0814-P 02 THEN
0927 081E-P 02     & ESTOURO DA TABELA DE SIMBOLOS
0928 081E-P 02       BEGIN
0929 081E-P 03       DISPLAY(@"ESTOURO NA TABELA DE SIMBOLOS. FIM DE PROGRAMA.",47);
0930 082A-P 03       COND.FIM:=E.TERM;
0931 082E-P 03       FIMPROG(@COND.FIM);
0932 0836-P 03       END;
0933 0836-P 02 ID.DECL:=FALSO;
0934 083A-P 02 END PREENCHE.TSEQ.SIMB;
```

```
0936 6002-D 01 WORD PROCEDURE HASH;
0937 6002-D 01 &
0938 6002-D 01 & FUNCAO : DETERMINAR O ENDERECO DE ENTRADA NA TABELA DE SIMBOLOS, QUE
0939 6002-D 01 &          SERA' PREENCHIDA COM ID'S DE ROTULO, CONSTANTE E VARIAVEL EXT
0940 6002-D 01 &          A FUNCAO DE HASH PASSA POR TRES FASES :
0941 6002-D 01 &            - FASE DE OU-EXCLUSIVO - E' FEITO OU-EXCLUIVO ENTRE WORD'S
0942 6002-D 01 &              COMPOSTAS PELA ASSOCIACAO DE CADA DOIS CARACTERES DO ID,
0943 6002-D 01 &              SENDO QUE DE CADA CARACTER E' RETIRADO O 1. E 5. BIT MAIS
0944 6002-D 01 &              SIGNIFICATIVO PARA ENTAO SER COMPACTADO EM 6 BITS E TRANS
0945 6002-D 01 &              NA WORD.
0946 6002-D 01 &            - FASE DE POTENCIA - A WORD RESULTANTE DA FASE ANTERIOR E'
0947 6002-D 01 &              ELEVADA AO QUADRADO E ENTAO SAO TOMADOS OS 10 BITS MENOS
0948 6002-D 01 &              SIGNIFICATIVOS PARA ENTRADA NA TABELA DE SIMBOLOS.
0949 6002-D 01 &            - FASE DE COLISAO - EM CASO DE COLISAO E' USADO UM INCREMEN
0950 6002-D 01 &              QUE DEPENDE DOS DOIS PRIMEIROS CARACTERES DO ID.
0951 6002-D 01 &
0952 6002-D 01 & ROTINA QUE A CHAMA : ANALISADOR.LEXICO
0953 6002-D 01 &
0954 6002-D 01 BEGIN
0955 083E-P 02 PT.ID:=&GID;
0956 0846-P 02 FILL(PT.ID,9,BRANCO);
0957 0854-P 02 MBI(INIC.TOKEN,PT.ID,TAM.TOKEN);
0958 0864-P 02 REC.XOR:=BRANCO;
0959 086A-P 02 FOR I:=0 TO 3
0960 086E-P 02 DO
0961 087A-P 02     & FASE OU-EXCLUSIVO :
0962 087A-P 02     BEGIN
0963 087A-P 03     REC.XOR:=I XOR (((CARAC.ID(0) AND 80007) OR ((CARAC.ID(0) SHR 1) AND
0964 0890-P 03                        OR
0965 0890-P 03                          ((((EXPS CARAC.ID(1)) RTL 6) AND 801C0)
0966 089C-P 03                           OR
0967 089C-P 03                            ((EXPS(CARAC.ID(1) SHR 1) RTL 6) AND 80E00)));
0968 08B8-P 03     PT.ID:=I+2;
0969 08BC-P 03     END;
0970 08C8-P 02 REC.XOR:=((REC.XOR*REC.XOR) AND 803FF);&FASE DE POTENCIA
```

```
0972 08D8-P 02 IF THASH.END(REC.XOR) = VAZIO
0973 08DE-P 02 THEN
0974 08E8-P 02       & ENTRADA VAZIA NA TAB HASH ==> ID NAO DECL
0975 08E8-P 02       BEGIN
0976 08E8-P 03       PREENCHE.TSEQ.SIMB(@END.TAB);
0977 08F2-P 03       HASH:=THASH.END(REC.XOR):=END.TAB;
0978 0904-P 03       RETURN;
0979 090C-P 03       END;
0980 090C-P 02 END.TAB:=THASH.END(REC.XOR);
0981 091A-P 02 IF COMPARE(INIC.TOKEN,TAM.TOKEN,END.TAB,TAM.TOKEN) = 0
0982 092E-P 02 THEN
0983 0932-P 02       IF SCAND(END.TAB+TAM.TOKEN,9-TAM.TOKEN,BRANCO) = -1
0984 0948-P 02       THEN
0985 094F-P 02            & ID JA DECL
0986 094E-P 02            BEGIN
0987 094E-P 03            ID.DECL:=VERO;
0988 0952-P 03            HASH:=END.TAB;
0989 095A-P 03            RETURN;
0990 095C-P 03            END;
0991 095C-P 02 & COLISAO :
0992 095C-P 02 CONTA.COLISAO:=!+1;
0993 0960-P 02 WHILE PROX.ENDER <> W.BRANCO
0994 0966-P 02 DO
0995 096E-P 02     & PROCURA ENTRADA LIVRE
0996 096F-P 02     BEGIN
0997 096F-P 03     IF COMPARE(INIC.TOKEN,TAM.TOKEN,END.TAB:=PROX.ENDER,TAM.TOKEN) = 0
0998 098A-P 03     THEN
0999 098F-P 03          IF SCAND(END.TAB+TAM.TOKEN,9-TAM.TOKEN,BRANCO) = -1
1000 09A4-P 03          THEN
1001 09AA-P 03               & ID JA' DECLARADO
1002 09AA-P 03               BEGIN
1003 09AA-P 04               ID.DECL:=VERO;
1004 09AE-P 04               HASH:=END.TAB;
1005 09B6-P 04               RETURN;
1006 09B8-P 04               END;
1007 09B8-P 03     END;
1008 09BA-P 02 PREENCHE.TSEQ.SIMB(@PROX.ENDER);
1009 09C6-P 02 HASH:=END.TAB:=PROX.ENDER;
1010 09D4-P 02 END HASH;
```

```
1012 6005-D 01 PROCEDURE MONTA.CADEIA(BYTE LIM.CADEIA);
1013 6005-D 01 &
1014 6005-D 01 & FUNCAO : PREPARA A CADEIA DE ID,PALAVRA RESERVADA,DECIMAL,BINARIO OU
1015 6005-D 01 &          HEXADECIMAL. CONTROLA O TRUNCAMENTO DA CADEIA.
1016 6005-D 01 &
1017 6005-D 01 & ROTINA QUE A CHAMA : ANALISADOR.LEXICO
1018 6005-D 01 &
1019 6005-D 01 BEGIN
1020 09DE-P 02 AVANCA.PT;
1021 09E2-P 02 IF TAM.TOKEN > LIM.CADEIA
1022 09E2-P 02 THEN
1023 09EC-P 02      BEGIN
1024 09EC-P 03        INIC.TOKEN:=!+1;
1025 09F0-P 03        TAM.TOKEN:=!-1;
1026 09F4-P 03        TRUNCADO:=VERO;
1027 09F8-P 03      END;
1028 09F8-P 02 END MONTA.CADEIA;
```

```
1030 6006-D 01 PROCEDURE ANALISADOR.LEXICO;
1031 6006-D 01 &
1032 6006-D 01 & GRAMATICA DO TEXTO DE ENTRADA DO SCANNER :
1033 6006-D 01 &                                          *
1034 6006-D 01 &  TEXTO = ((PAL!NUM) (BRANCOS!SIMB.UN))
1035 6006-D 01 &                                  *8
1036 6006-D 01 &  PAL = LETRA(LETRA!DIGITO)
1037 6006-D 01 &
1038 6006-D 01 &  NUM = DEC!BIN!HEX
1039 6006-D 01 &                 +
1040 6006-D 01 &  DEC = ARAB
1041 6006-D 01 &                     +
1042 6006-D 01 &  BIN = "Ç" BIT
1043 6006-D 01 &                       +
1044 6006-D 01 &  HEX = "Ö" HEXA
1045 6006-D 01 &                       +
1046 6006-D 01 &  BRANCOS = " "
1047 6006-D 01 &
1048 6006-D 01 &  SIMB.UN = ":"!","!"="!";"!"("!")"!"&"
1049 6006-D 01 &
1050 6006-D 01 & GRAMATICA DO TEXTO DE SAIDA DO SCANNER E ENTRADA DO SCREENER :
1051 6006-D 01 &
1052 6006-D 01 &  SAISCAN = (CLASSE,TAM.TOKEN,INIC.TOKEN)
1053 6006-D 01 &
1054 6006-D 01 &  CLASSE = PAL=0
1055 6006-D 01 &           DEC=1
1056 6006-D 01 &           BIN=2
1057 6006-D 01 &           HEX=3
1058 6006-D 01 &           SIMB.UN=4
1059 6006-D 01 &           OUTROS=5
1060 6006-D 01 &
1061 6006-D 01 &  TAM.TOKEN = BYTE - GUARDA O NUMERO DE CARACTERES DO TOKEN
1062 6006-D 01 &
1063 6006-D 01 &  INIC.TOKEN = WORD - GUARDA O ENDERECO DO TOKEN
1064 6006-D 01 &
1065 6006-D 01 & SAIDA DO SCREENER : QUAL.SIMB
1066 6006-D 01 &
1067 6006-D 01 & FUNCAO : TRANSFORMAR O TEXTO DE ENTRADA NO TEXTO DE SAIDA.
1068 6006-D 01 &          USA-SE A TECNICA DE AUTOMATO FINITO, IMPLEMENTADA ATRAVES DO
1069 6006-D 01 &
1070 6006-D 01 & ROTINAS QUE A CHAMA : PEGA.PROX.TOKEN
1071 6006-D 01 &
```

```
1073 6006-D 01 BEGIN
1074 09FA-P 02 COME.BRANCOS;
1075 09FE-P 02 INIC.TOKEN:=PT.REG.LIDO;
1076 0A06-P 02 TAM.TOKEN:=0;
1077 0A0A-P 02 CASE CLASSE(CAR.ATUAL) OF
1078 0A18-P 02 & IMPLEMENTACAO DE AUTOMATO FINITO
1079 0A18-P 02 BEGIN
1080 0A18-P 03       & IDENTIFICADOR OU PALAVRA RESERVADA :
1081 0A18-P 03       BEGIN
1082 0A18-P 04       REPEAT
1083 0A18-P 04            MONTA.CADEIA(MAX.CAR)
1084 0A20-P 04       UNTIL NOT(LETRA OR DIGITO OR CAR.ATUAL="_");
1085 0A3C-P 04       SE.TRUNCADO(ERRO.TRUNC.ID);
1086 0A44-P 04       CONT:=0;
1087 0A48-P 04       FILL((A:=ADDR(TAB.PAL.RES+110)),9,BRANCO);
1088 0A62-P 04       MBI(INIC.TOKEN,A,TAM.TOKEN);
1089 0A72-P 04       WHILE COMPARE (A,9,ADDR(TAB.PAL.RES+CONT),9) <> 0
1090 0A8C-P 04       DO
1091 0A90-P 04          & VERIFICA SE E' PALAVRA RESERVADA
1092 0A90-P 04          CONT:=CONT+10;
1093 0A96-P 04       QUAL.SIMB:=TAB.PAL.RES(CONT+9);
1094 0AA6-P 04       IF QUAL.SIMB=ID
1095 0AA6-P 04       THEN
1096 0AAE-P 04            & POE NA PILHA DE ATRIBUTOS ENDERECO DE ENTRADA NA TABELA DE
1097 0AAE-P 04            PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS:=CONT.PILHA.ATRIBUTOS+1):=HASH;
1098 0AC0-P 04       END;
1099 0AC4-P 03       & NUMERO DECIMAL :
1100 0AC4-P 03       BEGIN
1101 0AC4-P 04       REPEAT
1102 0AC4-P 04            MONTA.CADEIA(MAX.DEC)
1103 0ACC-P 04       UNTIL NOT DIGITO;
1104 0AD6-P 04       SE.TRUNCADO(ERRO.TRUNC.DEC);
1105 0ADE-P 04       QUAL.SIMB:=DEC;
1106 0AE4-P 04       PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS:=CONT.PILHA.ATRIBUTOS+1):=ASCIIBIN(INIC.TOKEN,
1107 0AF2-P 04                                                                    TAM.TOKEN);
1108 0B00-P 04       END;
1109 0B02-P 03       & NUMERO BINARIO :
1110 0B02-P 03       BEGIN
1111 0B02-P 04       REPEAT
1112 0B02-P 04            MONTA.CADEIA(MAX.BIN)
1113 0B0A-P 04       UNTIL CONTA.CAR < NB.LIDOS
1114 0B0A-P 04             AND
1115 0B12-P 04              (CAR.ATUAL <> "0"
1116 0B14-P 04               AND
1117 0B1A-P 04               CAR.ATUAL <> "1");
1118 0B2A-P 04       SE.TRUNCADO(ERRO.TRUNC.BIN);
1119 0B32-P 04       QUAL.SIMB:=BIN;
1120 0B36-P 04       PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS:=CONT.PILHA.ATRIBUTOS+1):=TAM.TOKEN:=TAM.TOKEN-1;
1121 0B4C-P 04       PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS:=CONT.PILHA.ATRIBUTOS+1)
1122 0B56-P 04       :=
```

```
1123 0B56-P 04          BITBIN(INIC.TOKEN+1,TAM.TOKEN);
1124 0B6C-P 04          END;
1125 0B6E-P 03          & NUMERO HEXADECIMAL :
1126 0B6E-P 03          BEGIN
1127 0B6E-P 04          REPEAT
1128 0B6E-P 04                MONTA.CADEIA(MAX.HEX)
1129 0B76-P 04          UNTIL (NOT DIGITO)
1130 0B7C-P 04                AND
1131 0B7C-P 04                  (CAR.ATUAL<"A"
1132 0B7C-P 04                  OR
1133 0B82-P 04                   CAR.ATUAL>"F");
1134 0B92-P 04          SE.TRUNCADO(ERRO.TRUNC.HEX);
1135 0B9A-P 04          QUAL.SIMB:=HEX;
1136 0B9E-P 04          PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS:=I+1):=(TAM.TOKEN:=I-1)*4;
1137 0BB6-P 04          PILHA.ATRIBUTOS(CONT.PILHA.ATRIBUTOS:=I+1)
1138 0BC0-P 04          :=
1139 0BC0-P 04          HEXBIN(INIC.TOKEN+1,TAM.TOKEN);
1140 0BD6-P 04          END;
1141 0BD8-P 03          & SIMBOLOS UNICOS :
1142 0BD8-P 03          BEGIN
1143 0BD8-P 04          QUAL.SIMB:=VG+SCAN(ATAB.SIMB.UN,8,CAR.ATUAL);
1144 0BEE-P 04          AVANCA.PT;
1145 0BF2-P 04          END;
1146 0BF4-P 03          & QUALQUER OUTRO SIMBOLO :
1147 0BF4-P 03          BEGIN
1148 0BF4-P 04          IF QUAL.SIMB = FIM.DE.CADEIA
1149 0BF4-P 04          THEN
1150 0BFC-P 04                BEGIN
1151 0BFC-P 05                ERRO(ERRO.FIM.INESPERADO);
1152 0C04-P 05                COND.FIM:=F.TERM;
1153 0C08-P 05                FIMPROG(ACOND.FIM);
1154 0C10-P 05                END;
1155 0C10-P 04          ERRO(ERRO.ASC.INESP);
1156 0C18-P 04          AVANCA.PT;
1157 0C1C-P 04          QUAL.SIMB:=OS.OUTROS;
1158 0C22-P 04          END;
1159 0C24-P 03 END;
1160 0C34-P 02 END ANALISADOR.LEXICO;
```

```
1162 6006-D 01 GLOBAL PROCEDURE IMPR.CABEC;
1163 6006-D 01 &
1164 6006-D 01 & FUNCAO : IMPRIMIR CABECALHO DE INICIO DE NOVA PAGINA.
1165 6006-D 01 &
1166 6006-D 01 & ROTINAS QUE A CHAMAM : ABRE.ARQ
1167 6006-D 01 &                        LE.ARGENT
1168 6006-D 01 &                        ERRO
1169 6006-D 01 &                        GRAVA.REG.ENT
1170 6006-D 01 &
1171 6006-D 01 BEGIN
1172 0C36-P 02 BYTE(*) PULA4 = 4:80A;
1173 600A-D 02 BYTE PULAPAG = 80C;
1174 600B-D 02 WRITE(UL.LIST,APULAPAG,1);
1175 0C44-P 02 BINASCII(CONTA.PAG,APAG.CABEC,4);
1176 0C52-P 02 WRITE(UL.LIST,ACABEC,97);
1177 0C60-P 02 CONTA.PAG:=!+1;
1178 0C64-P 02 WRITE(UL.LIST,APULA4,1);
1179 0C72-P 02 CONTA.LINHAS.PAG:=0;
1180 0C76-P 02 END;
1181 600B-D 01 PROCEDURE GRAVA.REGENT;
1182 600B-D 01 &
1183 600B-D 01 & FUNCAO : IMPRIME RELATORIO DE SAIDA DO GERADOR DE TABELA.
1184 600B-D 01 &
1185 600B-D 01 & ROTINA QUE A CHAMA : LE.ARGENT
1186 600B-D 01 &
1187 600B-D 01 BEGIN
1188 0C78-P 02 NUM.LINHAS:=!+1;
1189 0C7C-P 02 BINASCII(NUM.LINHAS,AREG.GRAVA,4);
1190 0C8A-P 02 IF (CONTA.LINHAS.PAG:=!+1) > NLINPAG
1191 0C8E-P 02 THEN
1192 0C96-P 02       & PULA PAGINA
1193 0C96-P 02       IMPR.CABEC;
1194 0C9A-P 02 WRITE(UL.LIST,AREG.GRAVA,NB.LIDOS+12);
1195 0CAC-P 02 END GRAVA.REG.ENT;
```

```
1197 600B-D 01 PROCEDURE LF.ARQENT;
1198 600B-D 01 &
1199 600B-D 01 & FUNCAO : LER ARQ DE ENTRADA, SINALIZAR FIM DE ARQ, IMPRIMIR RELATORIO
1200 600B-D 01 &                FONTE.
1201 600B-D 01 &
1202 600B-D 01 & ROTINA QUE A CHAMA : PEGA.PROX.TOKEN
1203 600B-D 01 &
1204 600B-D 01 BEGIN
1205 0CAE-P 02 PT.REG.LIDO:=^REG.LIDO;
1206 0CB6-P 02 IF READ(NAO.CANC OR UL.FONTE,PT.REG.LIDO,^NB.LIDOS) = FIM.DE.ARQ
1207 0CCC-P 02 THEN
1208 0CD2-P 02       BEGIN
1209 0CD2-P 03       QUAL.SIMB:=FIM.DE.CADEIA;
1210 0CD6-P 03       RETURN;
1211 0CD8-P 03       END;
1212 0CD8-P 02 IF COMPARE(PT.REG.LIDO,2,^"??",2) = 0
1213 0CE8-P 02 THEN
1214 0CEC-P 02       & PULA PAGINA
1215 0CEC-P 02       BEGIN
1216 0CEC-P 03       IMPR.CABEC;
1217 0CF0-P 03       LE.ARQENT;
1218 0CF4-P 03       RETURN;
1219 0CF6-P 03       END;
1220 0CF6-P 02 GRAVA.REGENT;
1221 0CFA-P 02 CONTA.CAR:=0;
1222 0CFE-P 02 REG.LIDO(^NB.LIDOS):=FIM.DE.REG;
1223 0D06-P 02 COME.BRANCOS;
1224 0D0A-P 02 END LF.ARQENT;
```

```
1226 600D-D 01 GLOBAL PROCEDURE PEGA.PROX.TOKEN;
1227 600D-D 01 &
1228 600D-D 01 & FUNCAO : CHAMA LEITURA DO ARQ COM A ESPECIFICACAO DA GRAMATICA.
1229 600D-D 01 &          CHAMA SCANNER E SCREENER, DEVOLVE QUAL.SIMB.
1230 600D-D 01 &
1231 600D-D 01 & ROTINAS QUE A CHAMAM : ANALISADOR.SINTATICO
1232 600D-D 01 &                        ERRO
1233 600D-D 01 &
1234 600D-D 01 BEGIN
1235 0D0C-P 02 REPEAT
1236 0D0C-P 02          BEGIN
1237 0D0C-P 03          WHILE CAR.ATUAL = FIM.DE.REG
1238 0D0C-P 03                OR
1239 0D10-P 03                QUAL.SIMB = COMENT
1240 0D12-P 03          DO
1241 0D1E-P 03            & LE PROX REG
1242 0D1E-P 03            BEGIN
1243 0D1E-P 04            QUAL.SIMB:=FALSO;
1244 0D22-P 04            LE.ARQENT;
1245 0D26-P 04            IF QUAL.SIMB=FIM.DE.CADEIA
1246 0D26-P 04            THEN
1247 0D2E-P 04                 RETURN;
1248 0D30-P 04            END;
1249 0D32-P 03          ANALISADOR.LEXICO;
1250 0D36-P 03          END
1251 0D36-P 02 UNTIL QUAL.SIMB <> COMENT
1252 0D36-P 02       AND
1253 0D3C-P 02         QUAL.SIMB <> QS.OUTROS;
1254 0D4A-P 02 END PEGA.PROX.TOKEN;
```

```
1256 600D-D 01 PROCEDURE PUSH.LADO.DIR(BYTE PRODUCAO);
1257 600D-D 01 &
1258 600D-D 01 & FUNCAO : EMPILHAR OS LADOS DIREITOS DAS PRODUCOES DA GRAMATICA.
1259 600D-D 01 &
1260 600D-D 01 BEGIN
1261 0D54-P 02 ENDER.TOPO.PILHA.SIMB:=PONT.LADO.DIR(PRODUCAO);
1262 0D62-P 02 IF (NUM.SIMB.LADO.DIR(PONT.PILHA:=!+1):=TOPO.PILHA.SIMB) = 0
1263 0D72-P 02 THEN
1264 0D74-P 02       & PRODUCAO VAZIA
1265 0D74-P 02       BEGIN
1266 0D74-P 03       PONT.PILHA:=!-1;
1267 0D78-P 03       ENDER.TOPO.PILHA.SIMB:=PILHA.ENDER.SIMB(PONT.PILHA);
1268 0D86-P 03       RETURN;
1269 0D88-P 03       END;
1270 0D88-P 02 PILHA.ENDER.SIMB(PONT.PILHA):=ENDER.TOPO.PILHA.SIMB:=!+1;
1271 0D9A-P 02 IF PONT.PILHA = ESTOURO.DA.PILHA
1272 0D9A-P 02 THEN
1273 0DA4-P 02       BEGIN
1274 0DA4-P 03       DISPLAY(A"ESTOURO NA PILHA DE SIMBOLOS",28);
1275 0DB0-P 03       COND.FIM:=E.TERM;
1276 0DB4-P 03       FIMPROG(ACOND.FIM);
1277 0DBC-P 03       END;
1278 0DBC-P 02 END PUSH.LADO.DIR;
```

```
1280 602A-D 01 PROCEDURE ABRE.ARQ;
1281 602A-D 01 &
1282 602A-D 01 & FUNCAO : ABRIR ARQ ENTR E IMPRIMIR LINHA INICIAL COM CABEC
1283 602A-D 01 &
1284 602A-D 01 BEGIN
1285 0DBE-P 02 BYTE PULA1 = 80A;
1286 602B-D 02 IF NOM.ARQENT(0) <> BRANCO
1287 0DBE-P 02 THEN
1288 0DC6-P 02       & NOME FOI PASSADO COMO PARAMETRO
1289 0DC6-P 02       BEGIN
1290 0DC6-P 03       UL.FONTE:=UL.TRAB1;
1291 0DCC-P 03       ALLOCATE(UL.FONTE,TIPO.UD,UNTD,VOL,ANOM.ARQENT,DEFAULT.E.S);
1292 0DEA-P 03       END;
1293 0DEA-P 02 VERSION(UL.FONTE,AVER.TAB);
1294 0DF8-P 02 VER.TAB(4):=VER.CABEC(0);
1295 0E00-P 02 MBT(AVER.CABEC+2,AVER.TAB[5,2);
1296 0E1A-P 02 VERSIONED(UL.FONTE,AVERF.CABEC);
1297 0E28-P 02 OPENINPUT(ARQ.ENTR OR UL.FONTE,TAM.MAX.REG.ENTR);
1298 0E38-P 02 IF ALLOCATE(UL.LIST OR NAO.CANC,"SR",DEFAULT.E.S,DEFAULT.E.S,
1299 0E48-P 02                                    DEFAULT.E.S,DEFAULT.E.S)=RDEF.DESL
1300 0E56-P 02 THEN
1301 0E5C-P 02       BEGIN
1302 0E5C-P 03       DISPLAY(A"RDEF DESLIGADO PARA LISTAGEM. FIM DE PROGRAMA.",46);
1303 0E68-P 03       COND.FIM:=F.TERM;
1304 0E6C-P 03       FIMPROG(ACOND.FIM);
1305 0E74-P 03       END;
1306 0E74-P 02 & IMPRESSAO DO CABECALHO DE INICIO DE PAGINA
1307 0E74-P 02 FILENAME(UL.FONTE,ANOME.CABEC);
1308 0E82-P 02 DATEHOUR(ADATA.CABEC);
1309 0E8A-P 02 MBT(A" PA",&PAG.CABEC-5,3);
1310 0E9C-P 02 IMPR.CABEC;
1311 0EA0-P 02 & IMPRESSAO DE CABECALHO PADRAO DE INICIO DE COMPILACAO
1312 0EA0-P 02 MBT(ANOME.CABEC,ANOME.FON,8);
1313 0EAE-P 02 FILEUNIT(UL.FONTE,ADISC.FON);
1314 0EBC-P 02 FILEVOL(UL.FONTE,AVOL.FON);
1315 0ECA-P 02 MBT(AVERF.CABEC,AVER.FON,6);
1316 0ED8-P 02 FILEFCB(UL.FONTE,ACOD.FON,10,6);
1317 0EEA-P 02 WRITE(UL.LIST,ACABEC.FON,58);
1318 0EF8-P 02 WRITE(UL.LIST,ACABEC.EQUIP,18);
1319 0F06-P 02 WRITE(UL.LIST,ACABEC.PARMS,13+TAM.PARMS);
1320 0F18-P 02 WRITE(UL.LIST,APULA1,1);
1321 0F26-P 02 CONTA.LINHAS.PAG:=4;
1322 0F2A-P 02 END ABRE.ARQ;
```

```
1324 605C-D 01 PROCEDURE FIM.DE.GERACAO;
1325 605C-D 01 &
1326 605C-D 01 & FUNCAO : IMPRIMIR RESULTADOS FINAIS.
1327 605C-D 01 &
1328 605C-D 01 BEGIN
1329 0E2C-P 02 BYTE(X) PULA3 = 3 : 00A ;
1330 605F-D 02 IF NAO.GERA.COD
1331 0E2C-P 02 THEN
1332 0E32-P 02       & ERRO GRAVE : ARQSAI E' DELETADO
1333 0E32-P 02       PURGE(UL.SAI)
1334 0E3C-P 02 ELSE
1335 0E3E-P 02       BEGIN
1336 0E3E-P 03       MBT(ANOME.ARQ.SAI,ANOME.FIM,8);
1337 0E4C-P 03       FILEFCB(UL.SAI,AA,32,2); & NUM DE UA'S - ARQ SAIDA
1338 0E5E-P 03       BINASCII(A,AUA.FIM,2);
1339 0E6C-P 03       FILEUNIT(UL.SAI,ADISCO.FIM); & DISCO
1340 0E7A-P 03       FILEVOL(UL.SAI,AVOL.FIM); & VOLUME
1341 0E88-P 03       FILEFCB(UL.SAI,ACOD.FIM,10,6); & CODIGO DO OPER
1342 0E9A-P 03       BINASCII(TOTREC(UL.SAI),AREG.FIM,4);
1343 0EB0-P 03       & REGRAVA PRIM REG COM NUMERO DE CAMPOS
1344 0EB0-P 03       NB.LIDOS:=12;
1345 0EB4-P 03       READANY(UL.SAI,AAREA.COD,ANB.LIDOS,1);
1346 0EC8-P 03       MBT(AN.CPO,AAREA.COD+12,TCOND+1);
1347 0EE4-P 03       REWRITE(UL.SAI,AAREA.COD,1COND+13);
1348 0FFA-P 03       VERSION(UL.SAI OR 00100,AVER.TAB);
1349 1008-P 03       VERSIONED(UL.SAI,AVER.FIM);
1350 1016-P 03       MBT(A" CODIG",ACOD.FIM-8,6);
1351 1028-P 03       END;
1352 1028-P 02 WRITE(UL.LIST,APULA3,3);
1353 1036-P 02 BINASCII(CONTA.ERRO,AERRO.FIM,4);
1354 1044-P 02 BINASCII(CONTA.ADV,AADV.FIM,4);
1355 1052-P 02 WRITE(UL.LIST,AFIM.GERA,74);
1356 1060-P 02 WRITE(UL.LIST,AFIM.GERA2,63);
1357 106E-P 02 FIMPROG(ACOND.FIM);
1358 1076-P 02 END;
```

```
1360 6065-D 01 PROCEDURE ERRA.PARMS;
1361 6065-D 01 &
1362 6065-D 01 & FUNCAO : AVISAR ERRO NA PASSAGEM DE PARAMETROS.
1363 6065-D 01 &
1364 6065-D 01 BEGIN
1365 1078-P 02 DISPLAY(A"ERRO NOS PARAMETROS. FIM DE PROGRAMA.",37);
1366 1084-P 02 COND.FIM:=E.TERM;
1367 1088-P 02 FIMPROG(ACOND.FIM);
1368 1090-P 02 END ERRA.PARMS;
1369 608A-D 01 BYTE PROCEDURE SIMBOLO(BYTE CARAC);
1370 608A-D 01 &
1371 608A-D 01 & FUNCAO : CONFERIR CARACTER
1372 608A-D 01 &
1373 608A-D 01 BEGIN
1374 109A-P 02 COME.BRANCOS;
1375 109E-P 02 SIMBOLO:=CAR.ATUAL = CARAC;
1376 10AC-P 02 END SIMBOLO;
1377 608C-D 01 PROCEDURE EXIJA(BYTE CARAC);
1378 608C-D 01 &
1379 608C-D 01 & FUNCAO : EXIGIR CARACTER
1380 608C-D 01 &
1381 608C-D 01 BEGIN
1382 10BA-P 02 IF NOT SIMBOLO(CARAC)
1383 10C4-P 02 THEN
1384 10CA-P 02       ERRA.PARMS;
1385 10CE-P 02 AVANCA.PI;
1386 10D2-P 02 END EXIJA;
```

```
1388 608D-D 01 PROCEDURE MONTA.CAD(BYTE CAD.ALFA,MAX.CAD);
1389 608D-D 01 &
1390 608D-D 01 & FUNCAO : MONTA CADEIA DE CARACTERES NUMERICOS OU ALFANUMERICOS
1391 608D-D 01 &
1392 608D-D 01 BEGIN
1393 10E0-P 02 TAM.TOKEN:=0;
1394 10E4-P 02 INIC.TOKEN:=PT.REG.LIDO;
1395 10EC-P 02 WHILE DIGITO
1396 10EC-P 02         OR
1397 10F0-P 02         (IF CAD.ALFA
1398 10F0-P 02          THEN
1399 10F6-P 02               & CADEIA ALFANUMERICA
1400 10F6-P 02               LETRA
1401 10F6-P 02          ELSE
1402 10FE-P 02               & CADEIA NUMERICA
1403 10FE-P 02               0)
1404 1104-P 02 DO
1405 110A-P 02    AVANCA.PT ;
1406 1110-P 02 IF TAM.TOKEN > MAX.CAD
1407 1110-P 02 THEN
1408 111A-P 02       ERRA.PARMS;
1409 111F-P 02 END MONTA.CAD;
```

```
1411 608F-D 01 PROCEDURE DISCO;
1412 608F-D 01 &
1413 608F-D 01 & FUNCAO : MONTA NO BUFFER INFORMACOES DE DISCO DO ARQUIVO-FONTE
1414 608F-D 01 &
1415 608F-D 01 BEGIN
1416 1120-P 02 EXIJA("D");
1417 1128-P 02 MONTA.CAD(VERO,3);
1418 1132-P 02 MBT(INIC.TOKEN-1,ATIPO.UD,TAM.TOKEN+1);
1419 1146-P 02 END DISCO;
1420 608F-D 01 PROCEDURE VOLUME;
1421 608F-D 01 &
1422 608F-D 01 & FUNCAO : TRAZER ESSA INFURMACAO DA AREA DE PARAMETROS PARA BUFFER.
1423 608F-D 01 &
1424 608F-D 01 BEGIN
1425 1148-P 02 COME.BRANCOS;
1426 114C-P 02 MONTA.CAD(FALSO,5);
1427 1156-P 02 VOL:=ASCIIBIN(INIC.TOKEN,TAM.TOKEN);
1428 1166-P 02 END VOLUME;
1429 608F-D 01 PROCEDURE MOVE.NOME;
1430 608F-D 01 &
1431 608F-D 01 & FUNCAO : MOVER NOME DO ARQ ENTR DA AREA DE PARAM PARA BUFFER
1432 608F-D 01 &
1433 608F-D 01 BEGIN
1434 1168-P 02 AVANCA.PT;
1435 116C-P 02 COME.BRANCOS;
1436 1170-P 02 IF DIGITO
1437 1170-P 02 THEN
1438 1178-P 02       BEGIN
1439 1178-P 03       INIC.TOKEN:=PT.REG.LIDO;
1440 1180-P 03       TAM.TOKEN:=0;
1441 1184-P 03       WHILE DIGITO
1442 1184-P 03       DO
1443 118C-P 03          AVANCA.PT;
1444 1192-P 03       UL.FONTE:=ASCIIBIN(INIC.TOKEN,TAM.TOKEN);
1445 11A2-P 03       RETURN;
1446 11A4-P 03       END;
1447 11A4-P 02 IF NOT LETRA
1448 11A4-P 02 THEN
1449 11AE-P 02       ERRA.PARMS;
1450 11B2-P 02 MONTA.CAD(VERO,MAX.CAR);
1451 11BC-P 02 MBT(INIC.TOKEN,ANOM.ARGENT,TAM.TOKEN);
1452 11CC-P 02 END MOVE.NOME;
```

```
1454 608F-D 01 PROCEDURE ANALISA.PARMS;
1455 608F-D 01 &
1456 608F-D 01 & FUNCAO : ANALISE DOS PARAMETROS PASSADOS NA EXECUCAO DO GERATAB.
1457 608F-D 01 &          SINTAXE DOS PARAMETROS : E:<ARQ> <INFO=DISCO=OU=VOLUME>;ES;
1458 608F-D 01 &
1459 608F-D 01 BEGIN
1460 11CE-P 02 CAD.PARMS(NB.LIDOS:=TAM.PARMS):=FIM.DE.REG;
1461 11DA-P 02 MBI(PT.REG.LIDO:=ACAD.PARMS,ACABEC.PARMS+13,TAM.PARMS);
1462 11F6-P 02 CONTA.CAR:=0;
1463 11FA-P 02 EXIJA("E");
1464 1202-P 02 IF SIMBOLO(":")
1465 120A-P 02 THEN
1466 120E-P 02       & DESCRICAO DO ARQUIVO DE ENTRADA
1467 120E-P 02       BEGIN
1468 120E-P 03       MOVE.NOME;
1469 1212-P 03       IF SIMBOLO("I")
1470 121A-P 03       THEN BEGIN
1471 121E-P 04            AVANCA.PT;
1472 1222-P 04            IF SIMBOLO("V")
1473 122A-P 04            THEN BEGIN
1474 122E-P 05                 & VOLUME E' PASSADO COMO PARAMETRO
1475 122E-P 05                 AVANCA.PT;
1476 1232-P 05                 TIPO.UD:="D ";
1477 1238-P 05                 EXIJA("=");
1478 1240-P 05                 VOLUME;
1479 1244-P 05                 END
1480 1244-P 04            ELSE
1481 1246-P 04                 DISCO;
1482 124A-P 04            EXIJA("I");
1483 1252-P 04            END;
1484 1252-P 03       IF SIMBOLO(FIM.DE.REG)
1485 125A-P 03       THEN
1486 125E-P 03            RETURN;
1487 1260-P 03       IF SIMBOLO(";")
1488 1268-P 03       THEN
1489 126C-P 03            BEGIN
1490 126C-P 04             AVANCA.PT;
1491 1270-P 04             IF SIMBOLO(FIM.DE.REG)
1492 1278-P 04             THEN RETURN;
1493 127E-P 04             EXIJA("E");
1494 1286-P 04             END
1495 1286-P 03       ELSE
1496 1288-P 03            ERRA.PARMS;
1497 128C-P 03       END;
1498 128C-P 02 IF SIMBOLO("S")
1499 1294-P 02 THEN
1500 1298-P 02       BEGIN
1501 1298-P 03       AVANCA.PT;
1502 129C-P 03       FSV.SE.EXISTE:=00300;
1503 12A2-P 03       IF SIMBOLO(";")
```

```
1504 12AA-P 03       THEN
1505 12AE-P 03            AVANCA.PT;
1506 12B2-P 03       EXTJA(FIM.DE.REG);
1507 12BA-P 03       END
1508 12BA-P 02 ELSE
1509 12BC-P 02       ERRA.PARMS;
1510 12C0-P 02 END ANALISA.PARMS;
```

```
1512 608F-D 01 &
1513 608F-D 01 & INICIO DOS COMANDOS :
1514 608F-D 01 &
1515 608F-D 01 TEMPOMAX(A(07F,0FF));
1516 12CA-P 01 ANALISA.PARMS;
1517 12CE-P 01 ABRE.ARQ;
1518 12D2-P 01 PUSH.LADO.DIR(0); & EMPILHA A PRODUCAO INICIAL DA GRAMATICA : S->PÇ
1519 12DA-P 01 PI.REG.LIDO:=AREG.LIDO;
1520 12E2-P 01 PEGA.PROX.TOKEN;
1521 12E6-P 01 & ALGORITMO DO PARSER PREDITIVO - CONFORME DRAGAO ITEM 5.5 :
1522 12E6-P 01 REPEAT
1523 12E6-P 01         BEGIN
1524 12E6-P 02           IF TOPO.PILHA.SIMB >= 0
1525 12E6-P 02           THEN
1526 12EC-P 02                & TOPO DA PILHA DE SIMBOLOS E' TERM OU FIM DE CADEIA
1527 12EC-P 02                IF QUAL.SIMB = TOPO.PILHA.SIMB
1528 12EC-P 02                THEN
1529 12F6-P 02                     BEGIN
1530 12F6-P 03                       IF QUAL.SIMB = FIM.DE.CADEIA
1531 12F6-P 03                       THEN
1532 12FE-P 03                            & PILHA E SEQ DE ENTRADA EM FIM DE CADEIA
1533 12FE-P 03                            FIM.DE.GERACAO;
1534 1302-P 03                       POP.TOPO; & RETIRA O ATUAL TOPO DA PILHA DE SIMB
1535 1306-P 03                       PEGA.PROX.TOKEN;
1536 130A-P 03                       END
1537 130A-P 02                ELSE
1538 130C-P 02                       ERRO(ERRO.SINTATICO)
1539 1314-P 02           ELSE
1540 1316-P 02                IF (TOPO.PILHA.SIMB AND 07F) >= 040
1541 131C-P 02                THEN
1542 1320-P 02                     & TOPODA PILHA INDICA CHAMADA DE ROTINA SEMANTICA
1543 1320-P 02                     ACAO.SEMANTICA(TOPO.PILHA.SIMB AND 03F)
1544 132C-P 02                ELSE
1545 132E-P 02                     & TOPO E' NAO TERM
1546 132E-P 02                     IF (NUM.PROD:=TAB.LL1(EXPZ N.TERM.TAB*(TOPO.PILHA.SIMB
1547 1334-P 02                                          + QUAL.SIMB)) < MAX.PROD
1548 1348-P 02                        AND
1549 134A-P 02                          QUAL.SIMB < N.TERM.TAB
1550 134C-P 02                     THEN
1551 1358-P 02                          & RESULTADO VALIDO NA TABELA LL(1)
1552 1358-P 02                          BEGIN
1553 1358-P 03                          POP.TOPO;
1554 135C-P 03                          PUSH.LADO.DIR(NUM.PROD);
1555 1366-P 03                          END
1556 1366-P 02                     ELSE
1557 1368-P 02                          ERRO(ERRO.SINTATICO);
1558 1370-P 02           END;
1559 1372-P 01 END GERATAB;
```

** FIM LPS ** ADVERTENCIAS:00 ERROS:00 MEMORIA: 01372-P + 06091-D

```
LIGADOR C500  V:V.24

EQUIPAMENTO : C500

PARAMETROS E/OU DIRETIVAS :
E:5,9,10:S:8;ES;LM;M:TABELA;P:.PARMS,80;A:200;V:F01 ;

BIBLIOTECAS DE MODULOS OBJETO :
 005 RTAB      [DP08] VOLUME=10101 CO=MICROA VERSAO=V.01
 009 RLPS      [DP00] VOLUME=00000 CO=SISTEM VERSAO=V.01
 010 ENTSAI    [DP00] VOLUME=00000 CO=SISTEM VERSAO=V.07

OPCOES EM CURSO :  LM; NLN; NLX;  LA;  ES; NOM; NRE; NCR;

ARQUIVO DE SAIDA :
 008 GERATAB   [DP08] VOLUME=10101 CO=MICROA VERSAO=F.01
```

```
LIGADOR C500  V:V.24

MAPA DE MEMORIA

RAIZ

TABELA          0000  0000
REWRITE         1372  6092
.SAIDA          1380  6092
.PREPARAES      1484  60A6
OPEN            1748  60E4
.ABRE           1764  60E4
.AREANALC       1896  6100
READANY         1896  6214
.LE             18A2  6214
PURGE           19FA  622C
CLOSE           1A46  622C
FORWARD         1AB4  6234
.LIBERANL       1AD4  6234
OPENINPUT       1B0A  6234
READ            1B30  6234
SCAN            1B3E  6234
HEXBIN          1B4E  6234
FLAG            1B84  6234
SCAND           1B84  6236
SET             1B94  6236
TERMINATE       1BAE  6236
DISPLAY         1BB6  6238
.SAIMSG         1BD0  623A
OPENUNIT        1C52  624C
BIT             1C60  624C
FILL            1C7A  624C
WRITE           1C8C  624C
VERSIONED       1C9A  624C
VERSION         1D68  625E
TOTREC          1F0A  6270
FILEVOL         1E44  6270
BINDEC          1F7E  6270
.FILEID         1EC4  6270
FILEUNIT        1F1C  6270
FILENAME        1F2C  6270
FILEFCB         1F3C  6270
CREATEDISK      1F4A  6270
EMPTY           205C  6292
ALLOCATE        2098  6292

END FINAL DA SECAO  :  PROG=218C DADOS=62B4


AREA ADICIONAL  :  00C8

*** CONDICAO DE TERMINO : 000 ***
```

```
ARQUIVO FONTE: FRSINT  [DP08]   VOLUME=10101    VERSAO=V.02  CODIGO=MICROA
EQUIPAMENTO:   C500
PARAMETROS:    E:FRSINT;S:RRSINT;ES
OPCOES INICIAIS: LE  NLC LA  ES  S   D0  D1  D2  D3  D4  D5  D6  D7  PFS

0002 0000-D 00 ?LPS,D,"MICROA"
0003 0000-D 00 &
0004 0000-D 00 & ENTRADAS : PROGRAMA-FONTE ESCRITO NA MICRO-LINGUAGEM
0005 0000-D 00 &                F
0006 0000-D 00 &                  TABELA DE CARACTERISTICAS DESSA MICRO-LINGUAGEM, QUE E' A
0007 0000-D 00 &                  SAIDA DA FASE DE GERACAO DE TABELA.
0008 0000-D 00 &
0009 0000-D 00 & SAIDAS : SE O USO DESTINA-SE A APLICACOES EM HARDWARE, A SAIDA E' UM
0010 0000-D 00 &          ARQUIVO-OBJETO PARA ENTRADA NO QUEIMADOR DE PROM'S.
0011 0000-D 00 &          SE O USO DESTINA-SE A APLICACOES EM SOFTWARE, A SAIDA E' UM
0012 0000-D 00 &          MODULO-OBJETO A SER LINK-EDITADO.
0013 0000-D 00 &
0014 0000-D 00 & CARACTERISTICAS DO PROGRAMA :
0015 0000-D 00 &
0016 0000-D 00 & - TABELA DE SIMBOLOS ACESSADO POR FUNCAO DE HASH.
0017 0000-D 00 &
0018 0000-D 00 & - USO DE ADICIONAL DE MEMORIA PARA DADOS.
0019 0000-D 00 &
0020 0000-D 00 & - METODO DO ANALISADOR SINTATICO : LL(1) COM TABELA.
0021 0000-D 00 &
0022 0000-D 00 & - RECUPERACAO DE ERROS : PANIC MODE.
0023 0000-D 00 &
0024 0000-D 00 & DATA DE CODIFICACAO : 10/08/81.
0025 0000-D 00 &
0026 0000-D 00 & DATA DE CONSOLIDACAO : 15/12/81.
0027 0000-D 00 &
```

```
0029 0000-D 00 BEGIN
0030 0000-P 01 &
0031 0000-P 01 & CONSTANTES GERAIS :
0032 0000-P 01 &
0033 0000-P 01 CONSTANT BRANCO       = " " ;
0034 0000-D 01 CONSTANT BYTE.VAZIO   = 0FF ;
0035 0000-D 01 CONSTANT CTL.ASA      = 00200 ;
0036 0000-D 01 CONSTANT DEF.E.S      = 0FFFF ;
0037 0000-D 01 CONSTANT DESL.RELAT   = 13 ;
0038 0000-D 01 CONSTANT ECO          = 00800 ;
0039 0000-D 01 CONSTANT FALSO        = 0 ;
0040 0000-D 01 CONSTANT FIM.ARQ      = 1 ;
0041 0000-D 01 CONSTANT FIM.DE.REG   = 0FF ;
0042 0000-D 01 CONSTANT IL01.JALOC   = 71 ;
0043 0000-D 01 CONSTANT IMP.ASCII    = 00100 ;
0044 0000-D 01 CONSTANT JA.EXISTE    = 102 ;
0045 0000-D 01 CONSTANT NAO.CANC     = 08000 ;
0046 0000-D 01 CONSTANT PARA.LEIT    = 01000 ;
0047 0000-D 01 CONSTANT PARA.SAIDA   = 00800 ;
0048 0000-D 01 CONSTANT RDEF.DESL    = 68 ;
0049 0000-D 01 CONSTANT RTEM         = 00100 ;
0050 0000-D 01 CONSTANT SEQL         = 00100 ;
0051 0000-D 01 CONSTANT SEQL.OBJ     = 00103 ;
0052 0000-D 01 CONSTANT SETA         = "↑" ;
0053 0000-D 01 CONSTANT VAZIO        = 0FFFF ;
0054 0000-D 01 CONSTANT VERO         = 1 ;
0055 0000-D 01 &
0056 0000-D 01 & CONSTANTES DE UNIDADES LOGICAS :
0057 0000-D 01 &
0058 0000-D 01 CONSTANT UL.MOD.OBJ = 242 ;
0059 0000-D 01 CONSTANT UL.RELAT = 6 ;
0060 0000-D 01 &
0061 0000-D 01 & CONSTANTES DE LIMITES :
0062 0000-D 01 &
0063 0000-D 01 CONSTANT MAX.BIN      = 256 ;
0064 0000-D 01 CONSTANT MAX.DEC      = 5 ;
0065 0000-D 01 CONSTANT MAX.HEX      = 64 ;
0066 0000-D 01 CONSTANT MAX.PAL      = 9 ;
0067 0000-D 01 CONSTANT MAX.PERC     = 10 ;
0068 0000-D 01 CONSTANT MAX.REG.OBJ = 1024 ;
0069 0000-D 01 CONSTANT N.TERM.TAB = 12 ;
0070 0000-D 01 CONSTANT NLIN.PAG     = 56 ;
0071 0000-D 01 CONSTANT NUM.ENTR.TSIMB = 1024 ;
0072 0000-D 01 CONSTANT TPILROT      = NUM.ENTR.TSIMB*2 ;
0073 0000-D 01 CONSTANT TAM.AREA.COD = 9*256+5 ;
0074 0000-D 01 CONSTANT TAM.ENTR.TSIMB = 14 ;
0075 0000-D 01 CONSTANT TAM.INFO.ARQ = 15;
0076 0000-D 01 CONSTANT TAM.TSIMB = TAM.ENTR.TSIMB*NUM.ENTR.TSIMB ;
0077 0000-D 01 CONSTANT TEXT.TAB = 512 ;
0078 0000-D 01 CONSTANT TEXT.MIC = 8*TEXT.TAB ;
```

```
0079 0000-D 01 CONSIANT TPSIMB      = 127 ;
```

```
0081 0000-D 01 &
0082 0000-D 01 & CONSTANTES DE CLASSE PARA O SCANNER :
0083 0000-D 01 &
0084 0000-D 01 CONSTANT BINARIO      = 4 ;
0085 0000-D 01 CONSTANT DECIMAL      = 1 ;
0086 0000-D 01 CONSTANT HEXADECIMAL  = 2 ;
0087 0000-D 01 CONSTANT PALAVRA      = 0 ;
0088 0000-D 01 CONSTANT SIMB.UNICO   = 3 ;
0089 0000-D 01 &
0090 0000-D 01 & CONTROLES DO CAMPO NA TABSIMB :
0091 0000-D 01 &
0092 0000-D 01 CONSTANT CTL.TAB.CTE = 0 ;
0093 0000-D 01 CONSTANT CTL.TAB.EXT = 1 ;
0094 0000-D 01 CONSTANT CTL.TAB.ROT = 2 ;
0095 0000-D 01 &
0096 0000-D 01 & CONSTANTES DOS SIMBOLOS DA GRAMATICA :
0097 0000-D 01 &
0098 0000-D 01 CONSTANT  TERM = 800 ;
0099 0000-D 01 CONSTANT NTERM = 880 ;
0100 0000-D 01 CONSTANT SEMANT= NTERM OR 640 ;
0101 0000-D 01 & SIMBOLOS NAO-TERMINAIS :
0102 0000-D 01 CONSTANT   P = NTERM OR 0 ;
0103 0000-D 01 CONSTANT   A = NTERM OR 1 ;
0104 0000-D 01 CONSTANT   B = NTERM OR 2 ;
0105 0000-D 01 CONSTANT LC1 = NTERM OR 3 ;
0106 0000-D 01 CONSTANT  CT = NTERM OR 4 ;
0107 0000-D 01 CONSTANT   E = NTERM OR 5 ;
0108 0000-D 01 CONSTANT   D = NTERM OR 6 ;
0109 0000-D 01 CONSTANT MIS = NTERM OR 7 ;
0110 0000-D 01 CONSTANT   N = NTERM OR 8 ;
0111 0000-D 01 CONSTANT  LC = NTERM OR 9 ;
0112 0000-D 01 CONSTANT   C = NTERM OR 10;
0113 0000-D 01 CONSTANT  LI = NTERM OR 11;
0114 0000-D 01 CONSTANT   I = NTERM OR 12;
0115 0000-D 01 & SIMBOLOS TERMINAIS :
0116 0000-D 01 CONSTANT  MAQUINA          = TERM OR 0 ;
0117 0000-D 01 CONSTANT  EXTERNA          = TERM OR 1 ;
0118 0000-D 01 CONSTANT  ORG              = TERM OR 2 ;
0119 0000-D 01 CONSTANT  BIN              = TERM OR 3 ;
0120 0000-D 01 CONSTANT  HEX              = TERM OR 4 ;
0121 0000-D 01 CONSTANT  DEC              = TERM OR 5 ;
0122 0000-D 01 CONSTANT  ID               = TERM OR 6 ;
0123 0000-D 01 CONSTANT  DOIS.PT          = TERM OR 7 ;
0124 0000-D 01 CONSTANT  IGUAL            = TERM OR 8 ;
0125 0000-D 01 CONSTANT  PT               = TERM OR 9 ;
0126 0000-D 01 CONSTANT  VG               = TERM OR 10;
0127 0000-D 01 CONSTANT  PT.VG            = TERM OR 11;
0128 0000-D 01 CONSTANT  FIM.DE.CADEIA    = TERM OR 12;
0129 0000-D 01 CONSTANT  COMENT           = TERM OR 13;
0130 0000-D 01 CONSTANT  OUTROS           = TERM OR 14;
```

```
0132 0000-D 01 & SIMBOLOS DE ACAO SEMANTICA :
0133 0000-D 01 CONSTANT SEMANTO  = SEMANT OR 0 ;
0134 0000-D 01 CONSTANT SEMANT1  = SEMANT OR 1 ;
0135 0000-D 01 CONSTANT SEMANT2  = SEMANT OR 2 ;
0136 0000-D 01 CONSTANT SEMANT3  = SEMANT OR 3 ;
0137 0000-D 01 CONSTANT SEMANT4  = SEMANT OR 4 ;
0138 0000-D 01 CONSTANT SEMANT5  = SEMANT OR 5 ;
0139 0000-D 01 CONSTANT SEMANT6  = SEMANT OR 6 ;
0140 0000-D 01 CONSTANT SEMANT7  = SEMANT OR 7 ;
0141 0000-D 01 &
0142 0000-D 01 & CONSTANT PARA OS ERROS :
0143 0000-D 01 &
0144 0000-D 01 CONSTANT ERRO.TRUNC.PAL          = 0 ;
0145 0000-D 01 CONSTANT ERRO.TRUNC.BIN          = 1 ;
0146 0000-D 01 CONSTANT ERRO.TRUNC.DEC          = 2 ;
0147 0000-D 01 CONSTANT ERRO.TRUNC.HEX          = 3 ;
0148 0000-D 01 CONSTANT ERRO.CAR.INESP          = 4 ;
0149 0000-D 01 CONSTANT ERRO.FIM.INESPERADO     = 5 ;
0150 0000-D 01 CONSTANT ERRO.EST.TAB.SIMB       = 6 ;
0151 0000-D 01 CONSTANT ERRO.MAQ.NVAL           = 7 ;
0152 0000-D 01 CONSTANT ERRO.RDEF.DESL          = 8 ;
0153 0000-D 01 CONSTANT ERRO.IL01.JALOC         = 9 ;
0154 0000-D 01 CONSTANT ERRO.OBJ.CH             = 10;
0155 0000-D 01 CONSTANT ERRO.EST.PILHA.SIMB     = 11;
0156 0000-D 01 CONSTANT ERRO.SINTATICO          = 12;
0157 0000-D 01 CONSTANT ERRO.ID.NAO.DECL        = 13;
0158 0000-D 01 CONSTANT ERRO.MNEM.INV           = 14;
0159 0000-D 01 CONSTANT ERRO.EXC.CPO            = 15;
0160 0000-D 01 CONSTANT ERRO.ID.JA.DECL         = 16;
0161 0000-D 01 CONSTANT ERRO.ROT.DUP            = 17;
0162 0000-D 01 CONSTANT ERRO.CPO.FALT           = 18;
0163 0000-D 01 CONSTANT ERRO.EST.TAB.USO.EXT    = 19;
0164 0000-D 01 CONSTANT ERRO.EST.TAB.USO.ROT    = 20;
0165 0000-D 01 CONSTANT ERRO.EST.TAB.EXT        = 21;
0166 0000-D 01 CONSTANT ERRO.EXT.HARD           = 22;
0167 0000-D 01 &
0168 0000-D 01 CONSTANT ERROS.NAO.GERA.COD      = ERRO.SINTATICO ;
0169 0000-D 01 CONSTANT ERROS.FATAIS            = ERRO.FIM.INESPERADO ;
0170 0000-D 01 &
0171 0000-D 01 & CONTROLES DE CAMPO NA TABELA DE ENTRADA :
0172 0000-D 01 &
0173 0000-D 01 CONSTANT FIX     =  0 ;
0174 0000-D 01 CONSTANT COND    =  1 ;
0175 0000-D 01 CONSTANT LIT     =  2 ;
0176 0000-D 01 CONSTANT VAR     =  4 ;
0177 0000-D 01 CONSTANT LIT.VAR =  6 ;
0178 0000-D 01 & ERROS PARA CONDICAO DE FIM DE PROGRAMA
0179 0000-D 01 CONSTANT ADVERTENCIA = 2;
0180 0000-D 01 CONSTANT ERRO.GRAVE  = 6;
0181 0000-D 01 CONSTANT ERRO.TERM   = 8;
```

```
0183 0000-D 01 &
0184 0000-D 01 & VARIAVEIS GLOBAIS :
0185 0000-D 01 &
0186 0000-D 01 GLOBAL BYTE(*) CABEC.PARMS = (" PARAMETROS: ",80:BRANCO) ;
0187 005D-D 01 GLOBAL WORD CONTA.LINHAS.PAG = 0 ;
0188 005F-D 01 GLOBAL WORD ERUF.CPO ;
0189 0061-D 01 GLOBAL WORD EPRIM.REG ;
0190 0063-D 01 GLOBAL BYTE INICIO = VERO;
0191 0064-D 01 GLOBAL BYTE NB.PAL;
0192 0065-D 01 GLOBAL WORD NMI.PROG;
0193 0067-D 01 GLOBAL WORD NMI.REG ;
0194 0069-D 01 GLOBAL WORD NUM.LINHA = 0 ;
0195 006B-D 01 & REGISTRO DE GRAVACAO DE FIM DE GERACAO :
0196 006B-D 01 GLOBAL BYTE(*) REG.FIM.GERA = ("-OBJETO: ",8:BRANCO,"[ V= 1 VERS
0197 0092-D 01 "X.XX-X CODIGO=XXXXXX XXXX REGISTROS");
0198 00B5-D 01 & REGISTRO DE SAIDA PARA ARQUIVO OBJETO :
0199 00B5-D 01 GLOBAL BYTE(MAX.REG.OBJ) REG.OBJ ;
0200 04B5-D 01 GLOBAL BYTE(*) REG.RELAT =(" XXXX XXXX ",81:FIM.DE.REG) ;
0201 0513-D 01 GLOBAL WORD TREG.OBJ;
0202 0515-D 01 GLOBAL WORD UL.FONTE;
0203 0517-D 01 GLOBAL WORD UL.OBJ;
0204 0519-D 01 GLOBAL BYTE(*) VER.OBJ = 8 : BRANCO;
0205 0521-D 01 &
0206 0521-D 01 & VARIAVEIS EXTERNAS :
0207 0521-D 01 &
0208 0521-D 01 EXTERNAL BYTE(*) .PARMS;
0209 0521-D 01 EXTERNAL BYTE AT.CPO;
0210 0521-D 01 EXTERNAL BYTE(*) CAD.INFO.ARQS;
0211 0521-D 01 EXTERNAL WORD D.CPO;
0212 0521-D 01 EXTERNAL WORD ERUF.LIT;
0213 0521-D 01 EXTERNAL WORD ECOD.GERADO;
0214 0521-D 01 EXTERNAL WORD EMEM.REC;
0215 0521-D 01 EXTERNAL WORD END.CAD.ARQS;
0216 0521-D 01 EXTERNAL WORD ETAB.SIMB;
0217 0521-D 01 EXTERNAL WORD IND.PROT;
0218 0521-D 01 EXTERNAL WORD N.ENTR;
0219 0521-D 01 EXTERNAL WORD NB;
0220 0521-D 01 EXTERNAL WORD NB.LIDOS;
0221 0521-D 01 EXTERNAL WORD NMI;
0222 0521-D 01 EXTERNAL WORD NPAL.OBJ;
0223 0521-D 01 EXTERNAL WORD(*) PIL.ROT;
0224 0521-D 01 EXTERNAL WORD(*) POS.CAD.INFO.ARQ;
0225 0521-D 01 EXTERNAL WORD PT.REG.LIDO;
0226 0521-D 01 EXTERNAL BYTE QUAL.SIMB;
0227 0521-D 01 EXTERNAL BYTE TEM.ARQ.ROT;
0228 0521-D 01 EXTERNAL BYTE(*) TSEQ.SIMB;
0229 0521-D 01 EXTERNAL WORD(*) UL.NPRF.ALOC;
0230 0521-D 01 EXTERNAL WORD(*) UL.PRE.ALOC;
0231 0521-D 01 EXTERNAL WORD VAL.EMEM;
```

```
0233 0521-D 01 &
0234 0521-D 01 & TABELA DAS PRODUCOES :
0235 0521-D 01 &
0236 0521-D 01 BYTE(*) LD00.S  = (2,P,FIM.DE.CADEIA) ;
0237 0524-D 01          &&&&& S --> P Ç
0238 0524-D 01 &
0239 0524-D 01 BYTE(*) LD01.P  = (8,MAQUINA,ID,SEMANT5,PT.VG,A,LI,SEMANT4,PT) ;
0240 052D-D 01          &&&&& P --> MAQUINA ID [5] ; A LI [4] .
0241 052D-D 01 &
0242 052D-D 01 BYTE(*) LD02.A  = (2,ID,E) ;
0243 0530-D 01          &&&&& A  --> ID E
0244 0530-D 01 &
0245 0530-D 01 BYTE(*) LD03.A  = (1,B);
0246 0532-D 01          &&&&& A  --> B
0247 0532-D 01 &
0248 0532-D 01 BYTE(*) LD04.E  = (3,D,PT.VG,A);
0249 0536-D 01          &&&&& E --> D ; A
0250 0536-D 01 &
0251 0536-D 01 BYTE(*) LD05.E  = (1,MIS);
0252 0538-D 01          &&&&& E --> MIS
0253 0538-D 01 &
0254 0538-D 01 BYTE(*) LD06.D  = (3,IGUAL,SEMANT0,N);
0255 053C-D 01          &&&&& D --> = [0] N
0256 053C-D 01 &
0257 053C-D 01 BYTE(*) LD07.D  = (2,EXTERNA,SEMANT1);
0258 053F-D 01          &&&&& D --> EXTERNA [1]
0259 053F-D 01 &
0260 053F-D 01 BYTE(*) LD08.N  = (1,DEC);
0261 0541-D 01          &&&&& N --> DEC
0262 0541-D 01 &
0263 0541-D 01 BYTE(*) LD09.N  = (1,HEX) ;
0264 0543-D 01          &&&&& N --> HEX
0265 0543-D 01 &
0266 0543-D 01 BYTE(*) LD10.MIS= (3,SEMANT2,DOIS.PT,LC);
0267 0547-D 01          &&&&& MIS--> [2] : LC
```

```
0269 0547-D 01 BYTE(*) LD11.MIS= (2,SEMANT3,LC1);
0270 054A-D 01          &&&&& MIS--> [3] LC1
0271 054A-D 01 &
0272 054A-D 01 BYTE(*) LD12.LC = (2,C,LC1) ;
0273 054D-D 01          &&&&& LC --> C LC1
0274 054D-D 01 &
0275 054D-D 01 BYTE(*) LD13.LC = (2,CT,LC1) ;
0276 0550-D 01          &&&&& LC --> CT LC1
0277 0550-D 01 &
0278 0550-D 01 BYTE(*) LD14.C  = (2,ID,SEMANT3) ;
0279 0553-D 01          &&&&& C  --> ID [3]
0280 0553-D 01 &
0281 0553-D 01 BYTE(*) LD15.C  = (1,SEMANT3) ;
0282 0555-D 01          &&&&& C  --> [3] EPSILON
0283 0555-D 01 &
0284 0555-D 01 BYTE(*) LD16.LC1= (2,VG,LC) ;
0285 0558-D 01          &&&&& LC1--> , LC
0286 0558-D 01 &
0287 0558-D 01 BYTE(*) LD17.LC1= (0);
0288 0559-D 01          &&&&& LC1--> EPSILON
0289 0559-D 01 &
0290 0559-D 01 BYTE(*) LD18.CT = (2,SEMANT6,BIN) ;
0291 055C-D 01          &&&&& CT --> [6] BIN
0292 055C-D 01 &
0293 055C-D 01 BYTE(*) LD19.CT = (2,SEMANT6,HEX) ;
0294 055F-D 01          &&&&& CT --> [6] HEX
0295 055F-D 01 &
0296 055F-D 01 BYTE(*) LD20.B  = (3,ORG,SEMANT7,HEX);
0297 0563-D 01          &&&&& B  --> ORG [7] HEX
0298 0563-D 01 &
0299 0563-D 01 BYTE(*) LD21.B  = (2,SEMANT3,LC1);
0300 0566-D 01          &&&&& B  --> [3] LC1
0301 0566-D 01 &
0302 0566-D 01 BYTE(*) LD22.B  = (2,CT,LC1);
0303 0569-D 01          &&&&& B  --> CT LC1
0304 0569-D 01 &
0305 0569-D 01 BYTE(*) LD23.LI = (4,SEMANT4,PT.VG,I,LI);
0306 056E-D 01          &&&&& LI --> [4] ; I LI
0307 056F-D 01 &
0308 056E-D 01 BYTE(*) LD24.LI = (0);
0309 056F-D 01          &&&&& LI --> EPSILON
0310 056F-D 01 &
0311 056F-D 01 BYTE(*) LD25.I  = (1,B);
0312 0571-D 01          &&&&& I  --> B
0313 0571-D 01 &
0314 0571-D 01 BYTE(*) LD26.I  = (2,ID,MIS);
0315 0574-D 01          &&&&& I  --> ID MIS
0316 0574-D 01 &ORD(*) PONT.LDS=(ALD00.S,ALD01.P,ALD02.A,ALD03.A,ALD04.E,ALD05.E,
0317 0580-D 01                   ALD06.D,ALD07.D,ALD08.N ,ALD09.N ,ALD10.MIS,ALD11.MIS,
0318 058C-D 01                   ALD12.LC,ALD13.LC,ALD14.C,ALD15.C,ALD16.LC1,
```

```
0319 0596-D 01                  ALD17.LC1,ALD18.CT,ALD19.C1,ALD20.B,ALD21.B,ALD22.B,
0320 0542-D 01                  ALD23.L1,ALD24.L1,ALD25.1,ALD26.1);
```

```
0322 05AA-D 01 &
0323 05AA-D 01 & TABELA SINTATICA LL(1) :
0324 05AA-D 01 &
0325 05AA-D 01 BYTE(*) TAB.LL1 = (1,11:BRANCO,                                  & P
0326 05B6-D 01                    2:BRANCO,3,3,3,BRANCO,2,2:BRANCO,3,3,3,       & A
0327 05C2-D 01                    2:BRANCO,20,22,22,4:BRANCO,21,21,21,          & B
0328 05CE-D 01                    9:BRANCO,17,16,17,                            & LC1
0329 05DA-D 01                    3:BRANCO,18,19,7:BRANCO,                      & CT
0330 05E6-D 01                    BRANCO,4,5:BRANCO,5,4,5,5,5,                  & E
0331 05F2-D 01                    BRANCO,7,6:BRANCO,6,3:BRANCO,                 & D
0332 05FE-D 01                    7:BRANCO,10,BRANCO,11,11,11,                  & MIS
0333 060A-D 01                    4:BRANCO,9,8,6:BRANCO,                        & N
0334 0616-D 01                    3:BRANCO,13,13,BRANCO,12,2:BRANCO,12,12,12,& LC
0335 0622-D 01                    6:BRANCO,14,2:BRANCO,15,15,15,                & C
0336 062E-D 01                    9:BRANCO,24,BRANCO,23,                        & LI
0337 063A-D 01                    2:BRANCO,25,25,25,BRANCO,26,2:BRANCO,25,25,25);& I
0338 0646-D 01 & BUFFERS DE LEITURA E GRAVACAO
0339 0646-D 01 BYTE(*) REG.LIDO POS REG.RELAT + DESL.RELAT ;
0340 0646-D 01 BYTE(*) REG.CABEC =("1MONTAMIC A.00",42:BRANCO,"E:XXXXXXXX X.XX-X XX/XX/
0341 0698-D 01                     " XX:XX PAG:XXXX");
0342 06A8-D 01 BYTE(*) NOME.CABEC POS REG.CABEC + 58 ;
0343 06A8-D 01 BYTE(*) VERF.CABEC POS REG.CABEC + 67 ;
0344 06A8-D 01 BYTE(*) DATA.CABEC POS REG.CABEC + 74 ;
0345 06A8-D 01 BYTE(*) PAG.CABEC POS REG.CABEC + 94 ;
0346 06A8-D 01 BYTE(*) VER.CABEC POS REG.CABEC + 10 ;
0347 06A8-D 01 BYTE(*) REG.ERRO = (" *** XXX",83:BRANCO);
0348 0703-D 01 BYTE(*) POS.ERRO POS REG.ERRO+5;
0349 0703-D 01 BYTE(*) POS.SETA POS POS.ERRO+3;
0350 0703-D 01 & AREA ADICTONAL
0351 0703-D 01 EXTERNAL WORD EADIC;
0352 0703-D 01 EXTERNAL WORD TADIC;
0353 0703-D 01 WORD(*) ECPO.MEM REF EADIC;
0354 0703-D 01 & PRIMEIRO REGISTRO DO ARQUIVO-TABELA
0355 0703-D 01 BYTE(*) PRIM.REG REF EPRIM.REG ;
0356 0703-D 01 BYTE(9) NMAQ POS PRIM.REG ; & NOME DA MAQUINA
0357 0703-D 01 WORD    TPAL POS NMAQ + 9 ; & TAM DA PALAVRA A SER GERADA
0358 0703-D 01 BYTE    ENC.OBJ POS TPAL + 2 ; & DEFAULT DE PREENCHIMENTO DO OBJETO
0359 0703-D 01 BYTE(*) NCPO POS ENC.OBJ + 1 ; & VETOR DE NUM CAMPOS POSSIVEIS
0360 0703-D 01 & BUFFER DE CAMPO NO ARQUIVO-TABELA
0361 0703-D 01 BYTE(*) DESCR.CPO REF EBUF.CPO ;
0362 0703-D 01 WORD POS.CPO POS DESCR.CPO ;
0363 0703-D 01 BYTE CTL.CPO POS POS.CPO + 2 ;
0364 0703-D 01 BYTE TAM.CPO POS CTL.CPO + 1 ;
0365 0703-D 01 BYTE ENC.CPO POS TAM.CPO + 1 ;
0366 0703-D 01 & SE CAMPO FIXO OU CONDICIONAL :
0367 0703-D 01 BYTE(*) CPO.FIX POS ENC.CPO + 1 ;
0368 0703-D 01 & SE CAMPO VARIAVEL (LITERAL OU NAO) :
0369 0703-D 01 WORD MNEM.CAR POS ENC.CPO + 1 ;
0370 0703-D 01 BYTE(*) CPO.VAR POS MNEM.CAR + 2 ; & VARIAVEL NAO LITERAL
```

```
0372 0703-D 01 & CABECALHOS DE FIM DE MONTADOR :
0373 0703-D 01 BYTE(*) NOM.OBJ POS REG.FIM.GERA + 9 ;
0374 0703-D 01 BYTE(*) DISCO.FIM POS REG.FIM.GERA + 18 ;
0375 0703-D 01 BYTE(*) VOL.FIM POS REG.FIM.GERA + 25 ;
0376 0703-D 01 BYTE(*) VER.FIM POS REG.FIM.GERA + 39 ;
0377 0703-D 01 BYTE(*) COD.FIM POS REG.FIM.GERA + 53 ;
0378 0703-D 01 BYTE(*) REG.FIM POS REG.FIM.GERA + 60 ;
0379 0703-D 01 BYTE(*) REG.FIM.GERA2 = ("-TAMANHO: XXXXX PALAVRAS DE XXX BYTES") ;
0380 0728-D 01 BYTE(*) NPAL.PRG POS REG.FIM.GERA2 + 10 ;
0381 0728-D 01 BYTE(*) NBYT.PAL POS REG.FIM.GERA2 + 28 ;
0382 0728-D 01 BYTE(*) REG.TPRG =(" ** FIM MONTAMIC ** ADVERTENCIAS:XXXX ERROS:XXXX ",
0383 0759-D 01                    "ROTULOS SEM ENDERECO:XXXX");
0384 0772-D 01 BYTE(*) NERRO POS REG.TPRG + 44 ;
0385 0772-D 01 BYTE(*) NROT POS REG.TPRG + 70 ;
0386 0772-D 01 BYTE(*) NADV POS REG.TPRG + 33 ;
0387 0772-D 01 &
0388 0772-D 01 WORD ESV.SE.EXISTE POS CAD.INFO.ARQS + 60;
0389 0772-D 01 BYTE SOFT POS CAD.INFO.ARQS + 62;
0390 0772-D 01 &
0391 0772-D 01 BYTE(*) INFO.ARQS REF END.CAD.ARQS;
0392 0772-D 01 BYTE(9) NOM.ARQ POS INFO.ARQS;
0393 0772-D 01 WORD TIPO.UD.ARQ POS NOM.ARQ + 9;
0394 0772-D 01 WORD UNTD.ARQ POS TIPO.UD.ARQ + 2;
0395 0772-D 01 WORD VOL.ARQ POS TIPO.UD.ARQ + 4;
0396 0772-D 01 &
0397 0772-D 01 BYTE(*) COD.GERADO REF ECOD.GERADO ;
0398 0772-D 01 &
0399 0772-D 01 BYTE(*) ENTRADA REF ETAB.SIMB ;
0400 0772-D 01 BYTE(9) NID POS ENTRADA ;
0401 0772-D 01 BYTE CTL.TAB POS NID + 9 ;
0402 0772-D 01 WORD ATRIB.TAB POS CTL.TAB + 1 ;
0403 0772-D 01 WORD PROX.ENDER POS ATRIB.TAB + 2 ;
0404 0772-D 01 &
0405 0772-D 01 BYTE(*) REG.ARQ.ROT = (" ARQUIVO DE ROTULOS E ENDERECOS ; XXXXXXXX");
0406 079C-D 01 BYTE(*) NOM.ARQ.ROT POS REG.ARQ.ROT + 34;
0407 079C-D 01 & CABECALHOS PADRAO LINGUAGEM COBRA
0408 079C-D 01 BYTE(*) CABEC.FON = (" FONTE: XXXXXXXX[XXXX V=XXXXX] VERSAO=X.XX-X CODIG
0409 07D0-D 01                    "XXXXX");
0410 07D6-D 01 BYTE(*) NOME.FON POS CABEC.FON + 8 ;
0411 07D6-D 01 BYTE(*) DISC.FON POS CABEC.FON + 17 ;
0412 07D6-D 01 BYTE(*) VOL.FON POS CABEC.FON + 24 ;
0413 07D6-D 01 BYTE(*) VER.FON POS CABEC.FON + 38 ;
0414 07D6-D 01 BYTE(*) COD.FON POS CABEC.FON + 52 ;
0415 07D6-D 01 BYTE(*) CABEC.EQUIP = (" EQUIPAMENTO: C500") ;
```

```
0417 07E8-D 01 &
0418 07E8-D 01 & VARIAVEIS LOCAIS :
0419 07E8-D 01 &
0420 07E8-D 01 WORD AUX,X ; & VARIAVEIS-AUXILIAR
0421 07EC-D 01 WORD ZERO = 0 ;
0422 07EE-D 01 BYTE(*) CAR POS 0;
0423 07EE-D 01 BYTE NUM.PROD ;
0424 07EF-D 01 WORD CONTA.ERRO = 0 ;
0425 07F1-D 01 WORD CONTA.ADV = 0 ;
0426 07F3-D 01 WORD CONTA.ROT = 0 ;
0427 07F5-D 01 BYTE NAO.GERA.COD = FALSO ;
0428 07F6-D 01 WORD TAM.PARMS POS .PARMS;
0429 07F6-D 01 WORD UL.TAB,UL.ROT=246;
0430 07FA-D 01 & PILHA DE SIMBOLOS :
0431 07FA-D 01 WORD END.TOPO.PSIMB ;
0432 07FC-D 01 BYTE TOPO.PSIMB REF END.TOPO.PSIMB ;
0433 07FC-D 01 BYTE TOPO ;
0434 07FD-D 01 BYTE IND.PILHA = -1 ;
0435 07FE-D 01 WORD(TPSIMB) PEND.SIMB ;
0436 08FC-D 01 BYTE(TPSIMB) NSIMB.LDS ;
0437 097B-D 01 & TRATAMENTO REGISTRO LIDO :
0438 097B-D 01 WORD ERRO.L ;
0439 097D-D 01 BYTE CAR.ATUAL REF PT.REG.LIDO ;
0440 097D-D 01 & CONTROLE PARA CABECALHO :
0441 097D-D 01 WORD CONTA.PAG = 1 ;
0442 097F-D 01 BYTE PULA.PAG = 0C ;
0443 0980-D 01 BYTE(*) PULA.4LIN = 4:0A ;
0444 0984-D 01 WORD COND.FIM = 0;
0445 0986-D 01 WORD REG.ATUAL;
```

```
0 47 0988-D 01 &
0 48 0988-D 01 & ROTINAS EXTERNAS :
0449 0988-D 01 &
0450 0988-D 01 WORD PROCEDURE ALLOCATE(WORD A1,A2,A3,A4,A5,A6);
0451 0988-D 01   EXTERNAL;
0452 0988-D 01 PROCEDURE ATRAB(WORD TPLPS);
0453 0988-D 01   EXTERNAL;
0454 0988-D 01 WORD PROCEDURE CREATEDISK(WORD A1,A2,A3,A4,A5,A6,A7,A8);
0455 0988-D 01   EXTERNAL;
0456 0988-D 01 WORD PROCEDURE FILENAME(WORD A,B);
0457 0988-D 01   EXTERNAL;
0458 0988-D 01 WORD PROCEDURE FILEUNIT(WORD A,B);
0459 0988-D 01   EXTERNAL;
0460 0988-D 01 WORD PROCEDURE FILEVOL (WORD A,B);
0461 0988-D 01   EXTERNAL;
0462 0988-D 01 WORD PROCEDURE FILEFCB(WORD CONTR,ENDER,DESLOC,TAM);
0463 0988-D 01   EXTERNAL;
0464 0988-D 01 WORD PROCEDURE FREE(WORD CONTR);
0465 0988-D 01   EXTERNAL;
0466 0988-D 01 WORD PROCEDURE TOTREC(WORD CONTR);
0467 0988-D 01   EXTERNAL;
0468 0988-D 01 WORD PROCEDURE CATALOG(WORD CONTR,END,NOME);
0469 0988-D 01   EXTERNAL;
0470 0988-D 01 WORD PROCEDURE VERSION(WORD CONTR,END,VER);
0471 0988-D 01   EXTERNAL;
0472 0988-D 01 WORD PROCEDURE VERSIONED(WORD CONTR,END,VER);
0473 0988-D 01   EXTERNAL;
0474 0988-D 01 PROCEDURE TEMPOMAX(WORD ERESP); SUPERVISOR 7;
0475 0988-D 01 PROCEDURE FIMPROG(WORD ECOND); SUPERVISOR 3;
0476 0988-D 01 PROCEDURE GRAVA.MOD.OBJ;
0477 0988-D 01   EXTERNAL;
0478 0988-D 01 PROCEDURE ANALISA.PARMS;
0479 0988-D 01   EXTERNAL;
0480 0988-D 01 PROCEDURE PEGA.PROX.TOKEN;
0481 0988-D 01   EXTERNAL;
0482 0988-D 01 PROCEDURE ACAO.SEMANT(BYTE ACAO);
0483 0988-D 01   EXTERNAL;
0484 0988-D 01 PROCEDURE GRAVA.OU.REGRAVA(WORD TAM);
0485 0988-D 01   EXTERNAL;
0486 0988-D 01 PROCEDURE GERA.CODIGO;
0487 0988-D 01   EXTERNAL;
```

```
0489 0988-D 01 GLOBAL PROCEDURE POP.TOPO;
0490 0988-D 01 &
0491 0988-D 01 & FUNCAO : DESEMPILHAR O SIMBOLO DO TOPO DA PILHA DE SIMBOLOS
0492 0988-D 01 &
0493 0988-D 01 BEGIN
0494 0000-P 02 IF (NSIMB.LDS(IND.PILHA):=I-1) = 0
0495 0008-P 02 THEN
0496 000E-P 02       & ATUAL TOPO CORRESPONDE AO ULTIMO SIMBOLO DE SUA PRODUCAO
0497 000E-P 02       & O NOVO TOPO E' O DA PRODUCAO ANTECESSORA
0498 000E-P 02       BEGIN
0499 000E-P 03       END.TOPO.PSIMB:=PEND.SIMB(IND.PILHA:=I-1);
0500 0020-P 03       RETURN;
0501 0022-P 03       END;
0502 0022-P 02 PEND.SIMB(IND.PILHA):=END.TOPO.PSIMB:=I+1;
0503 0034-P 02 END POP.TOPO;
```

```
0505 0988-D 01 GLOBAL PROCEDURE ERRO(WORD NUM.ERRO);
0506 0988-D 01 &
0507 0988-D 01 & FUNCAO : APONTAR O LUGAR E O NUMERO DO ERRO DETECTADO.
0508 0988-D 01 &
0509 0988-D 01 BEGIN
0510 003E-P 02 PROCEDURE PEGA.PROX.TOKEN;
0511 098A-D 02   EXTERNAL;
0512 098A-D 02 PROCEDURE IMPR.CABEC;
0513 098A-D 02   EXTERNAL;
0514 098A-D 02 PROCEDURE PUSH.LADO.DIR(BYTE NUM.PROD);
0515 098A-D 02   EXTERNAL;
0516 098A-D 02 FILL(APOS.SETA,83,BRANCO);
0517 004C-P 02 REG.ERRO(8+PT.REG.LIDO-AREG.LIDO):=SETA;
0518 005E-P 02 BINASCII(NUM.ERRO,APOS.ERRO,3);
0519 006C-P 02 WRITE(UL.RELAT,AREG.ERRO,91);
0520 007A-P 02 DISPLAY(AREG.RELAT,NB.LIDOS+DESL.RELAT);
0521 008A-P 02 DISPLAY(AREG.ERRO,91);
0522 0096-P 02 IF (CONTA.LINHAS.PAG:=!+1) > NLIN.PAG
0523 009A-P 02 THEN
0524 00A2-P 02       IMPR.CABEC;
0525 00A6-P 02 IF NUM.ERRO = ERRO.SINTATICO
0526 00A6-P 02    OR
0527 00AC-P 02    NUM.ERRO = ERRO.MNEM.INV
0528 00AF-P 02    OR
0529 00B8-P 02    NUM.ERRO = ERRO.EXC.CPO
0530 00B8-P 02 THEN
0531 00C4-P 02    BEGIN
0532 00C4-P 03    IF IND.PILHA = 0
0533 00C4-P 03    THEN
0534 00CA-P 03          ERRO(ERRO.FIM.INESPERADO);
0535 00D2-P 03    REPEAT
0536 00D2-P 03          POP.TOPO
0537 00D2-P 03    UNTIL TOPO.PSIMB = SFMANT4;
0538 00DF-P 03    PUSH.LADO.DIR(23);
0539 00E6-P 03    REPEAT
0540 00E6-P 03          PEGA.PROX.TOKEN
0541 00E6-P 03    UNTIL QUAL.SIMB = PT.VG
0542 00EA-P 03          OR
0543 00F0-P 03          QUAL.SIMB = PT;
0544 00FE-P 03    END;
0545 00FE-P 02 IF NUM.ERRO >= ERROS.NAO.GERA.COD
0546 00FF-P 02 THEN
0547 0106-P 02    BEGIN
0548 0106-P 03    NAO.GERA.COD:=VERO;
0549 010A-P 03    CONTA.ERRO:=!+1;
0550 010F-P 03    COND.FIM:=ERRO.GRAVE;
0551 0112-P 03    END
0552 0112-P 02 ELSE
0553 0114-P 02    IF NUM.ERRO >= ERROS.FATAIS
0554 0114-P 02    THEN
```

```
0555 011C-P 02                BEGIN
0556 011C-P 03                DISPLAY(CA"ERRO FATAL",10);
0557 0128-P 03                FIMPROG(CA(0,ERRO,TERM));
0558 0130-P 03                END
0559 0130-P 02          ELSE
0560 0132-P 02                BEGIN
0561 0132-P 03                CONTA.ADV:=!+1;
0562 0136-P 03                IF NOT NAO.GERA.COD
0563 0136-P 03                THEN
0564 013F-P 03                     COND.FIM:=ADVERTENCIA;
0565 0142-P 03                END;
0566 0142-P 02 END ERRO;
```

```
0568 0996-D 01 GLOBAL PROCEDURE IMPR.CABEC;
0569 0996-D 01 &
0570 0996-D 01 & FUNCAO : IMPRIMIR CABECALHO DE PAGINA DO RELATORIO.
0571 0996-D 01 &
0572 0996-D 01 & ROTINAS QUE A CHAMAM : ABRE.AROS
0573 0996-D 01 &                        LE.ARQFONT
0574 0996-D 01 &
0575 0996-D 01 BEGIN
0576 0144-P 02 BINASCII(CONTA.PAG,APAG.CABEC,4);
0577 0152-P 02 WRITECOL.RELAT,AREG.CABEC,98);
0578 0160-P 02 CONTA.PAG:=!+1;
0579 0164-P 02 WRITECOL.RELAT,APULA.4LIN,1);
0580 0172-P 02 CONTA.LINHAS.PAG:=0;
0581 0176-P 02 END IMPR.CABEC;
```

```
0583 0996-D 01 GLOBAL PROCEDURE PUSH.LADO.DIR(BYTE PRODUCAO);
0584 0996-D 01 &
0585 0996-D 01 & FUNCAO : EMPILHAR NA PILHA DE SIMBOLOS OS LADOS DIREITOS DAS PRODUCOES
0586 0996-D 01 &           DA GRAMATICA.
0587 0996-D 01 &
0588 0996-D 01 BEGIN
0589 0180-P 02 END.TOPO.PSIMB:=PONT.LDS(PRODUCAO);
0590 018E-P 02 IF (NSIMB.LDS(IND.PILHA:=!+1):=TOPO.PSIMB) = 0
0591 019F-P 02 THEN
0592 01A0-P 02      & LADO DIREITO E' CADEIA VAZIA
0593 01A0-P 02      BEGIN
0594 01A0-P 03      END.TOPO.PSIMB:=PEND.SIMB(IND.PILHA:=!-1);
0595 01B2-P 03      RETURN;
0596 01B4-P 03      END;
0597 01B4-P 02 PEND.SIMB(IND.PILHA):=END.TOPO.PSIMB:=!+1;
0598 01C6-P 02 IF IND.PILHA = TPSIMB - 1
0599 01C6-P 02 THEN
0600 01CE-P 02      ERRO(ERRO.EST.PILHA.SIMB);
0601 01D6-P 02 END PUSH.LADO.DIR;
0602 0997-D 01 WORD PROCEDURE NUM.LOG(BYTE POS.INFO.ARQ);
0603 0997-D 01 &
0604 0997-D 01 & FUNCAO : DETERMINAR O NUMERO LOGICO A SER ALOCADO AOS ARQUIVOS
0605 0997-D 01 &
0606 0997-D 01 BEGIN
0607 01E0-P 02 IF CAR(END.CAD.ARQS:=POS.CAD.INFO.ARQ(POS.INFO.ARQ)) = BRANCO
0608 01EE-P 02 THEN
0609 01F6-P 02      NUM.LOG:=UL.PRE.ALOC(POS.INFO.ARQ)
0610 01FC-P 02 ELSE
0611 0206-P 02      BEGIN
0612 0206-P 03      AUX:=NUM.LOG:=UL.NPRE.ALOC(POS.INFO.ARQ);
0613 0218-P 03      IF POS.INFO.ARQ = 2
0614 0218-P 03      THEN
0615 0220-P 03           RETURN;
0616 0226-P 03      ALLOCATE(AUX OR NAO.CANC,TIPO.UD.ARQ,UNID.ARQ,VOL.ARQ,^NOM.ARQ,DEF.
0617 0254-P 03      END;
0618 0254-P 02 END NUM.LOG;
```

```
0620 099A-D 01 PROCEDURE ABRE.AROS;
0621 099A-D 01 &
0622 099A-D 01 & FUNCAO : ABRIR OS DOIS ARQUIVOS DE ENTRADA : TABELA E FONTE.
0623 099A-D 01 &          TRAZER TODA A TABELA PARA A AREA ADICIONAL DE MEMORIA.
0624 099A-D 01 &          IMPRIMIR CABECALHO.
0625 099A-D 01 &
0626 099A-D 01 BEGIN
0627 0256-P 02 & TRATAMENTO DE ARQUIVO-TABELA
0628 0256-P 02 UL.TAB:=NUM.LOG(1);
0629 0262-P 02 OPENINPUT(UL.TAB OR PARA.LEIT,TAM.AREA.COD);
0630 0274-P 02 EPRIM.REG:=EBUF.CPO:=EADIC+2*TOTREC(UL.TAB);
0631 028E-P 02 NB.LIDOS:=-9;
0632 0292-P 02 AUX:=0;
0633 0296-P 02 WHILE (ERRO.L:=READ(NAO.CANC OR UL.TAB,EBUF.CPO:=I+NB.LIDOS+9,
0634 02AC-P 02                    ANB.LIDOS)) <> FIM.ARQ
0635 02BA-P 02 DO
0636 02C0-P 02    & LEVA 'A MEMORIA TODA A TABELA
0637 02C0-P 02    BEGIN
0638 02C0-P 03    ECPO.MEM(AUX):=EBUF.CPO;
0639 02CE-P 03    AUX:=I+1;
0640 02D2-P 03    END;
0641 02D4-P 02 ECOD.GERADO:=EBUF.CPO;
0642 02DC-P 02 & ALOCACAO DE IL01 PARA SAIDA
0643 02DC-P 02 IF (AUX:=ALLOCATE(UL.RELAT OR NAO.CANC,"IL","01",DEF.E.S,DEF.E.S,DEF.E.S
0644 02FE-P 02        = RDEF.DESL
0645 02FE-P 02 THEN
0646 0304-P 02    ERRO(ERRO.RDEF.DESL);
0647 030C-P 02 IF AUX = IL01.JALOC
0648 030C-P 02 THEN
0649 0314-P 02    ERRO(ERRO.IL01.JALOC);
0650 031C-P 02 OPEN(PARA.SAIDA OR UL.RELAT OR CTL.ASA);
0651 0326-P 02 & ABERTURA ARQ FONTE E IMPR PRIMEIRO CABEC
0652 0326-P 02 VERSION(UL.FONTE:=NUM.LOG(0),AVER.OBJ);
0653 033E-P 02 VER.OBJ(4):=VER.CABEC(0);
0654 0346-P 02 MBT(AVER.CABEC+2,AVER.OBJ+5,2);
0655 0360-P 02 VERSIONED(UL.FONTE,AVERF.CABEC);
0656 036E-P 02 OPENINPUT(PARA.LEIT OR UL.FONTE,80);
0657 037E-P 02 FILENAME(UL.FONTE,ANOME.CABEC);
0658 038C-P 02 DATEHOUR(ADATA.CABEC);
0659 0394-P 02 MBT(A" PA",APAG.CABEC-5,3);
0660 03A6-P 02 IMPR.CABEC;
0661 03AA-P 02 MBT(ANOME.CABEC,ANOME.FON,8);
0662 03B8-P 02 FILEUNIT(UL.FONTE,ADISC.FON);
0663 03C6-P 02 FILEVOL(UL.FONTE,AVOL.FON);
0664 03D4-P 02 MBT(AVERF.CABEC,AVER.FON,6);
0665 03E2-P 02 FILEFCB(UL.FONTE,ACOD.FON,10,6);
0666 03F4-P 02 WRITE(UL.RELAT,ACABFC.FON,58);
0667 0402-P 02 WRITE(UL.RELAT,ACABEC.EQUIP,18);
0668 0410-P 02 WRITE(UL.RELAT,ACABEC.PARMS,13+TAM.PARMS);
0669 0422-P 02 WRITE(UL.RELAT,APULA.4LIN,1);
```

```
0670 0430-P 02 CONTA.LINHAS.PAG:=4;
0671 0434-P 02 & CRIACAO ARQ OBJETO
0672 0434-P 02 IF CREATEDISK(ESV.SE.EXISTE OR NAO.CANC OR (UL.OBJ:=NUM.LOG(2)),TIPO.UD.
0673 0454-P 02                 UNID.ARQ,VOL.ARQ,ANOM.ARQ,SEQL,DEF.E.S,DEF.E.S) = JA.EXIST
0674 0486-P 02 THEN
0675 048C-P 02       ERRO(ERRO.OBJ.CR);
0676 0494-P 02 NB.PAL:=TPAL/8;
0677 04A4-P 02 TREG.OBJ:=(MAX.REG.OBJ/NB.PAL)*NB.PAL;
0678 04B8-P 02 OPENINPUT(UL.OBJ OR 81800,TREG.OBJ);
0679 04CA-P 02 NMI.REG:=TREG.OBJ/NB.PAL;
0680 04D8-P 02 FILL(AREG.OBJ,TREG.OBJ,ENC.OBJ);
0681 04EE-P 02 & ARQUIVO DE ROTULOS
0682 04EE-P 02 IF TEM.ARQ.ROT
0683 04EE-P 02 THEN
0684 04F4-P 02       IF UL.PRE.ALOC(3) = UL.ROT
0685 04F4-P 02       THEN
0686 04FE-P 02             BEGIN
0687 04FE-P 03             END.CAD.ARQS:=POS.CAD.INFO.ARQ(3);
0688 0506-P 03             CREATEDISK(UL.ROT OR 80300,TIPO.UD.ARQ,UNID.ARQ,VOL.ARQ,ANOM.A
0689 052A-P 03                        DEF.E.S,DEF.E.S,DEF.E.S);
0690 0548-P 03             END
0691 0548-P 02       ELSE
0692 054A-P 02             UL.ROT:=UL.PRE.ALOC(3);
0693 0552-P 02 END ABRE.ARQS;
```

```
0695 099D-D 01 PROCEDURE FIM.DE.GERACAO;
0696 099D-D 01 BEGIN
0697 0554-P 02 GRAVA.OU.REGRAVA(IF NPAL.OBJ > (X:=NB.PAL*(EMEM.REC MOD NMI.REG))
0698 0566-P 02                    THEN NPAL.OBJ
0699 056C-P 02                         ELSE X);
0700 057C-P 02 EBUF.CPO:=EBUF.LIT;
0701 0584-P 02 REG.ATUAL:=0;
0702 0588-P 02 INICIO:=VERO;
0703 058C-P 02 WHILE IND.PROT > 0
0704 058C-P 02 DO
0705 0592-P 02     BEGIN
0706 0592-P 03     FTAB.SIMB:=PIL.ROT(IND.PROT);
0707 05A0-P 03     IF (D.CPO:=ATRIB.TAB) = VAZIO
0708 05AA-P 03     THEN
0709 05B2-P 03          & NAO FOI DEFINIDO ENDERECO PARA ESSE ROTULO
0710 05B2-P 03          BEGIN
0711 05B2-P 04          IF INICIO
0712 05B2-P 04          THEN
0713 05B8-P 04               BEGIN
0714 05B8-P 05               INICIO:=FALSO;
0715 05BC-P 05               NAO.GERA.COD:=VERO;
0716 05C0-P 05               WRITE(UL.RELAT OR ECO,A" ROTULOS SEM ENDERECO :",23);
0717 05D0-P 05               COND.FIM:=ERRO.GRAVE;
0718 05D4-P 05               END;
0719 05D4-P 04          IF (AUX:=SCAN(ANID,9,0FF)) = -1
0720 05E6-P 04          THEN
0721 05EC-P 04               AUX:=9;
0722 05F0-P 04          MBT(ANID,AREG.RELAT+1,AUX);
0723 0602-P 04          WRITE(UL.RELAT OR ECO,AREG.RELAT,AUX+1);
0724 0616-P 04          CONTA.ROT:=I+1;
0725 061A-P 04          IF (CONTA.LINHAS.PAG:=I+1) > NLIN.PAG
0726 061F-P 04          THEN
0727 0626-P 04               IMPR.CABEC;
0728 062A-P 04          IND.PROT:=I-2;
0729 062E-P 04          END
0730 062E-P 03     ELSE
0731 0630-P 03          BEGIN
0732 0630-P 04          IF REG.ATUAL <> (X:=(NMI:=PIL.ROT(IND.PROT:=I-1))/NMI.REG+1)
0733 064C-P 04          THEN
0734 0652-P 04               BEGIN
0735 0652-P 05               IF REG.ATUAL <> 0
0736 0652-P 05               THEN
0737 0658-P 05                    REWRITE(UL.OBJ,AREG.OBJ,NB);
0738 066A-P 05               READANY(UL.OBJ,AREG.OBJ,ANB,REG.ATUAL:=X);
0739 0684-P 05               END;
0740 0684-P 04          MBT(AUX:=AREG.OBJ+(NMI MOD NMI.REG)*NB.PAL,ACOD.GERADO,NB.PAL);
0741 06B2-P 04          AT.CPO:=0;
0742 06B6-P 04          GERA.CODIGO;
0743 06BA-P 04          MBT(ACOD.GERADO,AUX,NB.PAL);
0744 06CA-P 04          IND.PROT:=I-1;
```

```
0745 06CF-P 04            END;
0746 06CF-P 03        END;
0747 06D2-P 02  IF REG.ATUAL <> 0
0748 06D2-P 02  THEN
0749 06D8-P 02        REWRITE(UL.OBJ,AREG.OBJ,NB);
0750 06EA-P 02  NMI.PROG:=IF EMEM.REC > VAL.EMEM
0751 06EA-P 02              THEN EMEM.REC
0752 06F4-P 02              ELSE VAL.EMEM;
0753 0702-P 02  IF NAO.GERA.COD
0754 0702-P 02  THEN
0755 0708-P 02        PURGE(UL.OBJ)
0756 0712-P 02  ELSE
0757 0714-P 02        BEGIN
0758 0714-P 03        FILENAME(UL.OBJ,ANOM.OBJ);
0759 0722-P 03        FILEUNIT(UL.OBJ,ADISCO.FIM);
0760 0730-P 03        FILEVOL(UL.OBJ,AVOL.FIM);
0761 073E-P 03        FILEFCB(UL.OBJ,ACOD.FIM,10,6);
0762 0750-P 03        IF SOFT
0763 0750-P 03        THEN
0764 0756-P 03              GRAVA.MOD.OBJ;
0765 075A-P 03        VERSION(UL.OBJ OR %0100,AVER.OBJ);
0766 076C-P 03        VERSIONED(UL.OBJ,AVER.FIM);
0767 077A-P 03        MBT(A" CODIG",ACOD.FIM-8,6);
0768 078C-P 03        BINASCII(TOTREC(UL.OBJ),AREG.FIM,4);
0769 07A2-P 03        END;
0770 07A2-P 02  WRITE(UL.RELAT,APULA.4LIN,4);
0771 07B0-P 02  IF TEM.ARQ.ROT
0772 07B0-P 02  THEN
0773 07B6-P 02        & VAI SER GRAVADO ARQ COM ROTULOS E ENDERECOS
0774 07B6-P 02        BEGIN
0775 07B6-P 03        WRITE(UL.ROT,ADATA.CABEC-1,9);
0776 07C8-P 03        FOR AUX:=0 TO N.ENTR-1
0777 07CC-P 03        DO
0778 07DC-P 03            & TRANSCREVE ROTULOS DA TAB SIMB PARA ARQ ROT
0779 07DC-P 03            BEGIN
0780 07DC-P 04            ETAB.SIMB:=XTSEQ.SIMB+AUX*TAM.ENTR.TSIMB;
0781 07EC-P 04            IF CTL.TAB = CTL.TAB.ROT
0782 07F2-P 04            THEN
0783 07F8-P 04                  BEGIN
0784 07F8-P 05                  CTL.TAB:=BRANCO;
0785 0802-P 05                  BINHEX(ATRIB.TAB,AATRIB.TAB,4,0);
0786 081A-P 05                  IF (X:=SCAN(ETAB.SIMB,9,BYTE.VAZIO)) <> -1
0787 082C-P 05                  THEN
0788 0832-P 05                        FILL(ETAB.SIMB+X,9-X,BRANCO);
0789 0848-P 05                  WRITE(UL.ROT,ETAB.SIMB,TAM.ENTR.TSIMB);
0790 0858-P 05                  END;
0791 0858-P 04            END;
0792 0864-P 03            MBT(ANOM.ARQ,ANOM.ARQ.ROT,8);
0793 0872-P 03            WRITE(UL.RELAT OR ECO,AREG.ARQ.ROT,41);
0794 0882-P 03        END;
```

```
0795 0882-P 02 BINASCII(CONTA.ERRO,ANERRO,4);
0796 0890-P 02 BINASCII(CONTA.ROT,ANROT,4);
0797 089F-P 02 BINASCII(CONTA.ADV,ANADV,4);
0798 08AC-P 02 BINASCII(NMI.PROG,ANPAL.PRG,5);
0799 08BA-P 02 BINASCII(NB.PAL,ANBYT.PAL,3);
0800 08C8-P 02 WRITE(UL.RELAT OR ECO,AREG.FIM.GERA,74);
0801 08D8-P 02 WRITE(UL.RELAT OR ECO,AREG.FIM.GERA2,37);
0802 08E8-P 02 WRITE(UL.RELAT OR ECO,AREG.TPRG,74);
0803 08F8-P 02 FIMPROG(ACOND.FIM);
0804 0900-P 02 END FIM.DE.GERACAO;
```

```
0806 09BC-D 01 &
0807 09BC-D 01 & INICIO DOS COMANDOS :
0808 09BC-D 01 &
0809 09BC-D 01 TEMPOMAX(X(87F,8FF));
0810 090A-P 01 ATRAB(200);
0811 0914-P 01 ANALISA.PARMS;
0812 0918-P 01 ABRE.ARQS;
0813 091C-P 01 PUSH.LADO.DIR(0);
0814 0924-P 01 PT.REG.LIDO:=AREG.LIDO;
0815 092C-P 01 PEGA.PROX.TOKEN;
0816 0930-P 01 & ALGORITMO DO PARSER PREDITIVO - CONFORME DRAGAO ITEM 5.5 :
0817 0930-P 01 REPEAT
0818 0930-P 01         BEGIN
0819 0930-P 02         IF TOPO.PSIMB >= TERM
0820 0930-P 02         THEN
0821 0936-P 02             & O TOPO DA PILHA DE SIMBOLOS E' TERMINAL
0822 0936-P 02             IF QUAL.SIMB = TOPO.PSIMB
0823 0936-P 02             THEN
0824 0940-P 02                 & SIMBOLO ATUAL DA CADEIA E PILHA IDENTICOS
0825 0940-P 02                 BEGIN
0826 0940-P 03                 IF QUAL.SIMB = FIM.DE.CADEIA
0827 0940-P 03                 THEN
0828 0948-P 03                     FIM.DE.GERACAO;
0829 094C-P 03                 POP.TOPO; & RETIRA ATUAL TOPO
0830 0950-P 03                 PEGA.PROX.TOKEN;
0831 0954-P 03                 END
0832 0954-P 02             ELSE
0833 0956-P 02                 ERRO(ERRO.SINTATICO)
0834 095E-P 02         ELSE
0835 0960-P 02             & O TOPO E' NAO TERMINAL
0836 0960-P 02             IF (TOPO:=TOPO.PSIMB AND 87F) >= 840
0837 0964-P 02             THEN
0838 096E-P 02                 & ACAO SEMANTICA
0839 096E-P 02                 ACAO.SEMANT(TOPO AND 03F)
0840 097A-P 02             ELSE
0841 097C-P 02                 IF (NUM.PROD:=TAB.LL1(EXPZ N.TERM.TAB*TOPO+QUAL.SIMB))<
0842 0992-P 02                     AND
0843 0994-P 02                     QUAL.SIMB < N.TERM.TAB
0844 0996-P 02                 THEN
0845 09A2-P 02                     & RESULTADO VALIDO NA TABELA LL(1)
0846 09A2-P 02                     BEGIN
0847 09A2-P 03                     POP.TOPO;
0848 09A6-P 03                     PUSH.LADO.DIR(NUM.PROD);
0849 09B0-P 03                     END
0850 09B0-P 02                 ELSE
0851 09B2-P 02                     ERRO(ERRO.SINTATICO);
0852 09BA-P 02         END;
0853 09BC-P 01 END MONTAMIC;
```

** FIM LPS ** ADVERTENCIAS:00 ERROS:00 MEMORIA: 009BC-P + 009BE-D

ARQUIVO FONTE: FLEX [DP08] VOLUME=10101 VERSAO=V.00 CODIGO=MICROA
EQUIPAMENTO: C500
PARAMETROS: E:FLEX;S:RLEX;ES
OPCOES INICIAIS: LE NLC LA ES S D0 D1 D2 D3 D4 D5 D6 D7 PF8

```
0002 0000-D 00 ?LPS,S,D,"ANLEX"
0003 0000-D 00 BEGIN
0004 0000-P 01 &
0005 0000-P 01 & CONSTANTES GERAIS :
0006 0000-P 01 &
0007 0000-P 01 CONSTANT BRANCO       = " " ;
0008 0000-D 01 CONSTANT BYTE.VAZIO   = 0FF ;
0009 0000-D 01 CONSTANT DEF.E.S      = 0FFFF ;
0010 0000-D 01 CONSTANT DESL.RELAT   = 13 ;
0011 0000-D 01 CONSTANT FALSO        = 0 ;
0012 0000-D 01 CONSTANT FIM.ARQ      = 1 ;
0013 0000-D 01 CONSTANT FIM.DE.REG   = 0FF ;
0014 0000-D 01 CONSTANT NAO.CANC     = 08000 ;
0015 0000-D 01 CONSTANT PARA.LEIT    = 01000 ;
0016 0000-D 01 CONSTANT PARA.SAIDA   = 00800 ;
0017 0000-D 01 CONSTANT RIEM         = 00100 ;
0018 0000-D 01 CONSTANT UL.RELAT     = 6 ;
0019 0000-D 01 CONSTANT VAZIO        = 0FFFF ;
0020 0000-D 01 CONSTANT VERO         = 1 ;
0021 0000-D 01 &
0022 0000-D 01 & CONSTANTES DE LIMITES :
0023 0000-D 01 &
0024 0000-D 01 CONSTANT MAX.BIN      = 256 ;
0025 0000-D 01 CONSTANT MAX.DEC      = 5 ;
0026 0000-D 01 CONSTANT MAX.HEX      = 64 ;
0027 0000-D 01 CONSTANT MAX.PAL      = 9 ;
0028 0000-D 01 CONSTANT MAX.PERC     = 10 ;
0029 0000-D 01 CONSTANT MAX.REG.OBJ  = 1024 ;
0030 0000-D 01 CONSTANT N.TERM.TAB   = 12 ;
0031 0000-D 01 CONSTANT NLIN.PAG     = 56 ;
0032 0000-D 01 CONSTANT NUM.ENTR.TSIMB = 1024 ;
0033 0000-D 01 CONSTANT TAM.AREA.COD = 9*256+5 ;
0034 0000-D 01 CONSTANT TAM.ENTR.TSIMB = 14 ;
0035 0000-D 01 CONSTANT TAM.INFO.ARQ = 15;
0036 0000-D 01 CONSTANT TAM.TSIMB = TAM.ENTR.TSIMB*NUM.ENTR.TSIMB ;
0037 0000-D 01 CONSTANT TPSIMB       = 127 ;
```

```
0039 0000-D 01 &
0040 0000-D 01 & CONSTANTES DE CLASSE PARA O SCANNER :
0041 0000-D 01 &
0042 0000-D 01 CONSTANT BINARIO      = 4 ;
0043 0000-D 01 CONSTANT DECIMAL      = 1 ;
0044 0000-D 01 CONSTANT HEXADECIMAL  = 2 ;
0045 0000-D 01 CONSTANT PALAVRA      = 0 ;
0046 0000-D 01 CONSTANT SIMB.UNICO   = 3 ;
0047 0000-D 01 &
0048 0000-D 01 & CONSTANT PARA OS ERROS :
0049 0000-D 01 &
0050 0000-D 01 CONSTANT ERRO.TRUNC.PAL       = 0 ;
0051 0000-D 01 CONSTANT ERRO.TRUNC.BIN       = 1 ;
0052 0000-D 01 CONSTANT ERRO.TRUNC.DEC       = 2 ;
0053 0000-D 01 CONSTANT ERRO.TRUNC.HEX       = 3 ;
0054 0000-D 01 CONSTANT ERRO.CAR.INESP       = 4 ;
0055 0000-D 01 CONSTANT ERRO.FIM.INESPERADO  = 5 ;
0056 0000-D 01 &
0057 0000-D 01 & CONSTANTES DOS SIMBOLOS DA GRAMATICA :
0058 0000-D 01 &
0059 0000-D 01 CONSTANT  TERM = 800 ;
0060 0000-D 01 CONSTANT NTERM = 880 ;
0061 0000-D 01 CONSTANT SEMANT= NTERM OR 840 ;
0062 0000-D 01 & SIMBOLOS TERMINAIS :
0063 0000-D 01 CONSTANT  MAQUINA          = TERM OR 0 ;
0064 0000-D 01 CONSTANT  EXTERNA          = TERM OR 1 ;
0065 0000-D 01 CONSTANT  ORG              = TERM OR 2 ;
0066 0000-D 01 CONSTANT  BIN              = TERM OR 3 ;
0067 0000-D 01 CONSTANT  HEX              = TERM OR 4 ;
0068 0000-D 01 CONSTANT  DEC              = TERM OR 5 ;
0069 0000-D 01 CONSTANT  ID               = TERM OR 6 ;
0070 0000-D 01 CONSTANT  DOIS.PT          = TERM OR 7 ;
0071 0000-D 01 CONSTANT  IGUAL            = TERM OR 8 ;
0072 0000-D 01 CONSTANT  PT               = TERM OR 9 ;
0073 0000-D 01 CONSTANT  VG               = TERM OR 10;
0074 0000-D 01 CONSTANT  PT.VG            = TERM OR 11;
0075 0000-D 01 CONSTANT  FIM.DE.CADEIA    = TERM OR 12;
0076 0000-D 01 CONSTANT  COMENT           = TERM OR 13;
0077 0000-D 01 CONSTANT  OUTROS           = TERM OR 14;
```

```
0079  0000-D  01  &
0080  0000-D  01  & VARIAVEIS GLOBAIS :
0081  0000-D  01  &
0082  0000-D  01  GLOBAL WORD CONTA.CAR = 0;
0083  0002-D  01  GLOBAL WORD INIC.ID;
0084  0004-D  01  GLOBAL WORD INIC.TOKEN;
0085  0006-D  01  GLOBAL WORD NB.LIDOS;
0086  0008-D  01  GLOBAL WORD PT.REG.LIDO;
0087  000A-D  01  GLOBAL BYTE QUAL.SIMB;
0088  000B-D  01  GLOBAL BYTE TAM.ID;
0089  000C-D  01  GLOBAL BYTE TAM.TOKEN;
0090  000D-D  01  GLOBAL WORD VAL.DEC;
```

```
0092 000F-D 01 GLOBAL PROCEDURE PEGA.PROX.TOKEN;
0093 000F-D 01 BEGIN
0094 0000-P 02 &
0095 0000-P 02 & FUNCAO : CONTROLAR A LEITURA DO ARQUIVO-FONTE.
0096 0000-P 02 &          CHAMAR O ANALISADOR LEXICO, QUE VAI DEVOLVER O PROXIMO TOKEN.
0097 0000-P 02 &
0098 0000-P 02 &
0099 0000-P 02 & VARIAVEIS EXTERNAS :
0100 0000-P 02 &
0101 0000-P 02 EXTERNAL WORD CONTA.LINHAS.PAG;
0102 000F-D 02 EXTERNAL WORD NUM.LINHA;
0103 000F-D 02 EXTERNAL BYTE(*) REG.RELAT;
0104 000F-D 02 EXTERNAL WORD UL.FONTE;
0105 000F-D 02 EXTERNAL WORD VAL.EMEM;
```

```
0107 000F-D 02 &
0108 000F-D 02 & VARIAVEIS LOCAIS :
0109 000F-D 02 &
0110 000F-D 02 & TABELAS DE CONSULTA :
0111 000F-D 02 BYTE(*) TAB.SIMB.UN = (":","=",".",",",";","&");
0112 0015-D 02 BYTE(*) QS.SIMB.UN = (DOIS.PT,IGUAL,PT,VG,PT.VG,COMENT);
0113 001B-D 02 BYTE(*) PRESERV = ("MAQUINA","EXTERNA","ORG");
0114 002C-D 02 BYTE(*) TAM.PRESERV = (7,7,3);
0115 002F-D 02 BYTE(*) QS.PRESERV = (MAQUINA,EXTERNA,ORG);
0116 0032-D 02 & TABELA DE CLASSE PARA O SCANNER :
0117 0032-D 02 BYTE(*) CLASSE = (35:SIMB.UNICO,HEXADECIMAL,BINARIO,11:SIMB.UNICO,10:DEC
0118 006C-D 02 7:SIMB.UNICO,26:PALAVRA,37:SIMB.UNICO);
0119 00B2-D 02 WORD AUX;
0120 00B4-D 02 & TRATAMENTO REGISTRO LIDO :
0121 00B4-D 02 WORD ERRO.L ;
0122 00B6-D 02 BYTE CAR.ATUAL REF PT.REG.LIDO ;
0123 00B6-D 02 BYTE(*) REG.LIDO POS REG.RELAT+DESL.RELAT;
0124 00B6-D 02 & ANALISADOR LEXICO :
0125 00B6-D 02 BYTE TRUNCADO = FALSO ;
0126 00B7-D 02 BYTE IND.PRESERV = 0 ;
0127 00B8-D 02 BYTE PROX.IND = 1;
0128 00B9-D 02 &
0129 00B9-D 02 & PROCEDIMENTOS EXTERNOS :
0130 00B9-D 02 &
0131 00B9-D 02 PROCEDURE ERRO(WORD NUM,ERRO);
0132 00B9-D 02 EXTERNAL;
0133 00B9-D 02 PROCEDURE IMPR.CABEC;
0134 00B9-D 02 EXTERNAL;
```

```
0136 00B9-D 02 GLOBAL BYTE PROCEDURE LETRA;
0137 00B9-D 02 &
0138 00B9-D 02 & FUNCAO : RECONHECEDOR DE CARACTER ALFABETICO
0139 00B9-D 02 &
0140 00B9-D 02 BEGIN
0141 0006-P 03 LETRA:=CONTA.CAR < NB.LIDOS
0142 0006-P 03         AND
0143 000F-P 03         CLASSE(CAR.ATUAL) = PALAVRA;
0144 0020-P 03 END LETRA;
0145 00BA-D 02 GLOBAL BYTE PROCEDURE DIGITO;
0146 00BA-D 02 &
0147 00BA-D 02 & FUNCAO : RECONHECEDOR DE CARACTERE NUMERICO DECIMAL
0148 00BA-D 02 &
0149 00BA-D 02 BEGIN
0150 002A-P 03 DIGITO:=CONTA.CAR < NB.LIDOS
0151 002A-P 03         AND
0152 0032-P 03         CLASSE(CAR.ATUAL) = DECIMAL;
0153 0046-P 03 END DIGITO;
0154 00BB-D 02 GLOBAL PROCEDURE AVANCA.PT;
0155 00BB-D 02 &
0156 00BB-D 02 & FUNCAO : ATUALIZAR PONTEIROS
0157 00BB-D 02 &
0158 00BB-D 02 BEGIN
0159 004E-P 03 IAM.TOKEN:=!+1;
0160 0052-P 03 PT.REG.LIDO:=!+1;
0161 0056-P 03 CONTA.CAR:=!+1;
0162 005A-P 03 END AVANCA.PT;
0163 00BB-D 02 GLOBAL PROCEDURE COME.BRANCOS;
0164 00BB-D 02 &
0165 00BB-D 02 & FUNCAO : PULAR CARACTER BRANCO
0166 00BB-D 02 &
0167 00BB-D 02 & ROTINAS QUE A CHAMAM : LE.ARQFONT
0168 00BB-D 02 &                       ANALISADOR.LEXICO
0169 00BB-D 02 &
0170 00BB-D 02 BEGIN
0171 005C-P 03 WHILE CAR.ATUAL = BRANCO
0172 005C-P 03 DO
0173 0064-P 03     AVANCA.PT;
0174 006A-P 03 END COME.BRANCOS;
```

```
0176 00BB-D 02 PROCEDURE MONTA.CADEIA(WORD LIM.CAD);
0177 00BB-D 02 &
0178 00BB-D 02 & FUNCAO : TRUNCAR A CADEIA, SEMPRE 'A ESQUERDA, QUANDO LIM.CAD. FOR
0179 00BB-D 02 &          ALCANCADO
0180 00BB-D 02 &
0181 00BB-D 02 BEGIN
0182 0074-P 03 AVANCA.PT;
0183 0078-P 03 IF TAM.TOKEN > LIM.CAD
0184 0078-P 03 THEN
0185 0082-P 03      & TRUNCA 'A ESQUERDA
0186 0082-P 03      BEGIN
0187 0082-P 04      INIC.TOKEN:=I+1;
0188 0086-P 04      TAM.TOKEN:=I-1;
0189 008A-P 04      TRUNCADO:=VERO;
0190 008E-P 04      END;
0191 008F-P 03 END MONTA.CADEIA;
0192 00BD-D 02 PROCEDURE SE.TRUNCADO(BYTE ERRO,TRUNC);
0193 00BD-D 02 IF TRUNCADO
0194 0098-P 02 THEN BEGIN
0195 009E-P 03      ERRO(ERRO,TRUNC);
0196 00A8-P 03      TRUNCADO:=FALSO;
0197 00AC-P 03      END;
```

```
0199 00BE-D 02 PROCEDURE ANALISADOR.LEXICO;
0200 00BE-D 02 &
0201 00BE-D 02 & FUNCAO : DEVOLVER O PROXIMO TOKEN, ATRAVES DA VARTAVEL QUAL.SIMB, AO
0202 00BE-D 02 &          ANALISADOR SINTATICO, EMPILHAR OS ATRIBUTOS DOS TOKENS.
0203 00BE-D 02 &
0204 00BE-D 02 & ROTINAS CHAMADAS : COME.BRANCOS
0205 00BE-D 02 &                    LETRA
0206 00BE-D 02 &                    DIGITO
0207 00BE-D 02 &                    MONTA.CADEIA
0208 00BE-D 02 &                    SE.TRUNCADO
0209 00BE-D 02 &                    ERRO
0210 00BE-D 02 &                    AVANCA.PT
0211 00BE-D 02 BEGIN
0212 00AE-P 03 COME.BRANCOS;
0213 00B2-P 03 INIC.TOKEN:=PT.REG.LIDO;
0214 00BA-P 03 TAM.TOKEN:=0;
0215 00BE-P 03 CASE CLASSE(CAR.ATUAL) OF
0216 00CC-P 03 BEGIN
0217 00CC-P 04          & CLASSE E' PALAVRA :
0218 00CC-P 04          BEGIN
0219 00CC-P 05          REPEAT
0220 00CC-P 05                  MONTA.CADEIA(MAX.PAL)
0221 00D4-P 05          UNTIL NOT ( LETRA
0222 00D4-P 05                    OR
0223 00D8-P 05                    CAR.ATUAL = "-"
0224 00D8-P 05                    OR
0225 00E2-P 05                    DIGITO );
0226 00F0-P 05          SE.TRUNCADO(ERRO.TRUNC.PAL);
0227 00F8-P 05          REPEAT
0228 00F8-P 05                  BEGIN
0229 00F8-P 06                  AUX:=COMPARE(INIC.TOKEN,TAM.TOKEN,
0230 0100-P 06                               APRESERV+7*IND.PRESERV,TAM.PRESERV(IND.PRESERV));
0231 0122-P 06                  IND.PRESERV:=I+1
0232 0122-P 06                  END
0233 0126-P 05          UNTIL AUX = 0
0234 0126-P 05                OR
0235 012A-P 05                IND.PRESERV = 3;
0236 0138-P 05          IF AUX = 0
0237 0138-P 05          THEN
0238 013E-P 05                & E' PALAVRA RESERVADA
0239 013E-P 05                BEGIN
0240 013E-P 06                QUAL.SIMB:=OS.PRESERV(IND.PRESERV-1);
0241 014E-P 06                IND.PRESERV:=PROX.IND;
0242 0156-P 06                END
0243 0156-P 05          ELSE
0244 0158-P 05                & E' ID OU DECL
0245 0158-P 05                BEGIN
0246 0158-P 06                QUAL.SIMB:=ID;
0247 015C-P 06                INIC.ID:=INIC.TOKEN;
0248 0164-P 06                TAM.ID:=TAM.TOKEN;
```

```
0249 016C-P 06              END;
0250 016C-P 05          END;
```

```
0252 0170-P 04          & CLASSE E' DECIMAL :
0253 0170-P 04          BEGIN
0254 0170-P 05          REPEAT
0255 0170-P 05                  MONTA.CADEIA(MAX.DEC)
0256 0178-P 05          UNTIL NOT DIGITO;
0257 0182-P 05          SE.TRUNCADO(ERRO.TRUNC.DEC);
0258 018A-P 05          QUAL.SIMB:=DEC;
0259 018E-P 05          VAL.DEC:=ASCIIBIN(INIC.TOKEN,TAM.TOKEN);
0260 019E-P 05          END;
0261 01A0-P 04          & CLASSE E' HEXADECIMAL :
0262 01A0-P 04          BEGIN
0263 01A0-P 05          REPEAT
0264 01A0-P 05                  MONTA.CADEIA(MAX.HEX)
0265 01A8-P 05          UNTIL (NOT DIGITO)
0266 01AF-P 05                  AND
0267 01AF-P 05                  (CAR.ATUAL < "A"
0268 01AF-P 05                   OR
0269 01B4-P 05                   CAR.ATUAL > "F");
0270 01C4-P 05          SE.TRUNCADO(ERRO.TRUNC.HEX);
0271 01CC-P 05          QUAL.SIMB:=HEX;
0272 01D0-P 05          INIC.TOKEN:=I+1;
0273 01D4-P 05          TAM.TOKEN:=I-1;
0274 01D8-P 05          END;
0275 01DA-P 04          & CLASSE E' SIMBOLO UNICO OU IMPROPRIO DA GRAMATICA :
0276 01DA-P 04          BEGIN
0277 01DA-P 05          IF (AUX:=SCAN(ATAB.SIMB.UN,6,CAR.ATUAL)) = -1
0278 01EF-P 05          THEN
0279 01F4-P 05                  & CARACTER NAO ESPERADO NA GRAMATICA
0280 01F4-P 05                  BEGIN
0281 01F4-P 06                  IF QUAL.SIMB = FIM.DE.CADEIA
0282 01F4-P 06                  THEN
0283 01FC-P 06                          & ERRO ANTERIOR FEZ ANALISE SE PERDER
0284 01FC-P 06                          ERRO(ERRO.FIM.INESPERADO);
0285 0204-P 06                  ERRO(ERRO.CAR.INESP);
0286 020C-P 06                  QUAL.SIMB:=OUTROS;
0287 0210-P 06                  END
0288 0210-P 05          ELSE
0289 0212-P 05                  & CARACTER E' SIMBOLO UNICO
0290 0212-P 05                  QUAL.SIMB:=QS.SIMB.UN(AUX);
0291 021E-P 05          AVANCA.PT;
0292 0222-P 05          END;
```

```
0294 0224-P 04          & CLASSE E' BINARIA :
0295 0224-P 04          BEGIN
0296 0224-P 05          REPEAT
0297 0224-P 05                  MONTA.CADEIA(MAX.BIN)
0298 022E-P 05          UNTIL CAR.ATUAL <> "0"
0299 022E-P 05                  AND
0300 0234-P 05                      CAR.ATUAL <> "1";
0301 0242-P 05          SE.TRUNCADO(ERRO.TRUNC.BIN);
0302 024A-P 05          QUAL.SIMB:=BIN;
0303 024E-P 05          INIC.TOKEN:=I+1;
0304 0252-P 05          TAM.TOKEN:=I-1;
0305 0256-P 05          END;
0306 0258-P 04 END;
0307 0266-P 03 END ANALISADOR.LEXICO;
0308 00BE-D 02 PROCEDURE LE.ARQFONT;
0309 00BE-D 02 &
0310 00BE-D 02 & FUNCAO : LER ARQUIVOS FONTE, IMPRIMIR RELATORIO.
0311 00BE-D 02 & ROTINAS CHAMADAS : IMPR.CABEC COME.BRANCOS
0312 00BE-D 02 BEGIN
0313 0268-P 03 & LE ARQ FONTE :
0314 0268-P 03 PT.REG.LIDO:=AREG.LIDO;
0315 0270-P 03 IF (ERRO.L:=READ(@8000 OR UL.FONTE,PT.REG.LIDO,ANB.LIDOS)) = FIM.ARQ
0316 028A-P 03 THEN
0317 0290-P 03          & FIM DO ARQ FONTE
0318 0290-P 03          BEGIN
0319 0290-P 04          QUAL.SIMB:=FIM.DE.CADEIA;
0320 0294-P 04          RETURN;
0321 0296-P 04          END;
0322 0296-P 03 IF COMPARE(PT.REG.LIDO,2,A"?P",2) = 0
0323 02A6-P 03 THEN
0324 02AA-P 03          & CONTROLE DE PULO DE PAGINA NO RELATORIO
0325 02AA-P 03          BEGIN
0326 02AA-P 04          NUM.LINHA:=I+1;
0327 02AE-P 04          IMPR.CABEC;
0328 02B2-P 04          LE.ARQFONT;
0329 02B6-P 04          RETURN;
0330 02B8-P 04          END;
0331 02B8-P 03 & IMPRIME RELATORIO :
0332 02B8-P 03 NUM.LINHA:=I+1;
0333 02BC-P 03 IF (CONTA.LINHAS.PAG:=I+1) > NLIN.PAG
0334 02C0-P 03 THEN IMPR.CABEC;
0335 02CC-P 03 BINASCII(NUM.LINHA,AREG.RELAT+1,4);
0336 02DC-P 03 BINHEX(VAL.EMEM,AREG.RELAT+7,4,0);
0337 02F8-P 03 WRITE(UL.RELAT,AREG.RELAT,NB.LIDOS+DESL.RELAT);
0338 030A-P 03 CONTA.CAR:=0;
0339 030E-P 03 REG.LIDO(NB.LIDOS):=FIM.DE.REG;
0340 0318-P 03 COME.BRANCOS;
0341 031C-P 03 END LE.ARQFONT;
```

```
0343 00C0-D 02 &
0344 00C0-D 02 & INICIO DOS COMANDOS :
0345 00C0-D 02 &
0346 00C0-D 02 REPEAT
0347 031E-P 02          BEGIN
0348 031E-P 03          WHILE CAR.ATUAL = FIM.DE.REG  & SIMBOLO DE FIM DE REGISTRO
0349 031E-P 03                OR
0350 0324-P 03                QUAL.SIMB = COMENT      & SIMBOLO DE COMENTARIO
0351 0326-P 03          DO
0352 0332-P 03             BEGIN
0353 0332-P 04             QUAL.SIMB:=FALSO;
0354 0336-P 04             LE.ARQFONT;
0355 033A-P 04             IF QUAL.SIMB = FIM.DE.CADEIA
0356 033A-P 04             THEN
0357 0342-P 04                  & ACABOU ARQ FONTE
0358 0342-P 04                  RETURN;
0359 0344-P 04             END;
0360 0346-P 03          ANALISADOR.LEXICO;
0361 034A-P 03          END
0362 034A-P 02 UNTIL QUAL.SIMB <> COMENT
0363 034A-P 02         AND
0364 0350-P 02         QUAL.SIMB <> OUTROS;
0365 035E-P 02 END PEGA.PROX.TOKEN;
0366 00C0-D 01 END;

OBJETO:FLEX    [OP08 V=10101 ] VERSAO=V.00   COD=MICROA   0005 REGISTROS
** FIM LPS **  ADVERTENCIAS:00      ERROS:00  MEMORIA: 00360-P + 00C0-D
```

```
LPS/500 X.06    ANALISADOR DE PARAMETROS - MONTAMIC    E:FPARMS   23/09/82 12:04 PAG:00

ARQUIVO FONTE: FPARMS   [DP08]    VOLUME=10101     VERSAO=V.01   CODIGO=MICROA
EQUIPAMENTO:    C500
PARAMETROS:     E:FPARMS;S:RPARMS;ES
OPCOES INICIAIS: LF  NLC LA  ES  S   D0  D1  D2  D3  D4  D5  D6  D7  PFS


0002 0000-D 00 ?LPS,S,D,"ANPARMS"
0003 0000-D 00 BEGIN
0004 0000-P 01 &
0005 0000-P 01 & CONSTANTES GERAIS :
0006 0000-P 01 &
0007 0000-P 01 CONSTANT BRANCO      = " " ;
0008 0000-D 01 CONSTANT ERRO.TERM   = 8 ;
0009 0000-D 01 CONSTANT FALSO       = 0 ;
0010 0000-D 01 CONSTANT FIM.DE.REG = 0FF ;
0011 0000-D 01 CONSTANT MAX.PAL     = 9 ;
0012 0000-D 01 CONSTANT TAM.INFO.ARQ=15;
0013 0000-D 01 CONSTANT VERO        = 1 ;
0014 0000-D 01 GLOBAL BYTE(*) .PARMS = 80 : 0 ;
0015 0050-D 01 GLOBAL BYTE(*) CAD.INFO.ARQS = (60 : BRANCO,3 : FALSO);
0016 008F-D 01 GLOBAL WORD END.CAD.ARQS;
0017 0091-D 01 GLOBAL WORD(*) POS.CAD.INFO.ARQ = (ACAD.INFO.ARQS, & INFO SOBRE FONTE
0018 0093-D 01                                    ACAD.INFO.ARQS+15, & INFO SOBRE TABEL
0019 0095-D 01                                    ACAD.INFO.ARQS+30, & INFO SOBRE OBJET
0020 0097-D 01                                    ACAD.INFO.ARQS+45); & INFO SOBRE ARQ D
0021 0099-D 01 GLOBAL BYTE TEM.ARQ.ROT = FALSO ;
0022 009A-D 01 GLOBAL WORD(*) UL.NPRE.ALOC = (243,244,245);
0023 00A0-D 01 GLOBAL WORD(*) UL.PRE.ALOC = (7,5,8,246) ;
```

```
0025 00A8-D 01 GLOBAL PROCEDURE ANALISA.PARMS;
0026 00A8-D 01 &
0027 00A8-D 01 & FUNCAO : ANALISE DOS PARAMETROS PASSADOS NA EXECUCAO DO MONTAMIC.
0028 00A8-D 01 &
0029 00A8-D 01 &          OS PARAMETROS OBEDECEM A SEGUINTE SINTAXE :
0030 00A8-D 01 &
0031 00A8-D 01 &          E:<NOME-ARQ-FONTE>;T:<NOME-ARQ-TAB>;S:<NOME-ARQ-OBJ>;ES;!SOFT
0032 00A8-D 01 &                                                              !HARD
0033 00A8-D 01 &            <NOME-ARQ-ROT>;
0034 00A8-D 01 &
0035 00A8-D 01 BEGIN
0036 0000-P 02 &
0037 0000-P 02 & VARIAVEIS EXTERNAS :
0038 0000-P 02 &
0039 0000-P 02 EXTERNAL BYTE(*) CABEC.PARMS;
0040 00A8-D 02 EXTERNAL WORD CONTA.CAR;
0041 00A8-D 02 EXTERNAL WORD INIC.TOKEN;
0042 00A8-D 02 EXTERNAL BYTE INICIO;
0043 00A8-D 02 EXTERNAL WORD NB.LIDOS;
0044 00A8-D 02 EXTERNAL WORD PT.REG.LIDO;
0045 00A8-D 02 EXTERNAL BYTE TAM.TOKEN;
0046 00A8-D 02 &
0047 00A8-D 02 & VARIAVEIS LOCAIS :
0048 00A8-D 02 &
0049 00A8-D 02 WORD TAM.PARMS POS .PARMS;
0050 00A8-D 02 BYTE(*) CAD.PARMS POS (TAM.PARMS + 2;
0051 00A8-D 02 WORD FSV.SE.EXISTE POS CAD.INFO.ARQS + 60;
0052 00A8-D 02 BYTE SOFT POS CAD.INFO.ARQS + 62;
0053 00A8-D 02 BYTE(*) INFO.ARQS REF END.CAD.ARQS;
0054 00A8-D 02 BYTE(9) NOM.ARQ POS INFO.ARQS;
0055 00A8-D 02 WORD TIPO.UD.ARQ POS NOM.ARQ + 9;
0056 00A8-D 02 WORD UNID.ARQ POS TIPO.UD.ARQ + 2;
0057 00A8-D 02 WORD VOL.ARQ POS TIPO.UD.ARQ + 4;
0058 00A8-D 02 BYTE CAR.ATUAL REF PT.REG.LIDO ;
0059 00A8-D 02 BYTE LIGA.OPC = FALSO;
0060 00A9-D 02 BYTE QUAL.PARM;
0061 00AA-D 02 BYTE(*) CAB.PARM = ("E","T","S","R");
0062 00AE-D 02 &
0063 00AE-D 02 & PROCEDIMENTOS EXTERNOS :
0064 00AE-D 02 &
0065 00AE-D 02 PROCEDURE AVANCA.PT;
0066 00AE-D 02   EXTERNAL;
0067 00AE-D 02 PROCEDURE COME.BRANCOS;
0068 00AE-D 02   EXTERNAL;
0069 00AE-D 02 BYTE PROCEDURE DIGITO;
0070 00AE-D 02   EXTERNAL;
0071 00AE-D 02 PROCEDURE FIMPROG(WORD ECOND);
0072 00AE-D 02   SUPERVISOR 3;
0073 00AE-D 02 BYTE PROCEDURE LETRA;
0074 00AE-D 02   EXTERNAL;
```

```
0076 00AE-D 02 PROCEDURE ERRA.PARMS;
0077 00AE-D 02 &
0078 00AE-D 02 & FUNCAO : TERMINAR PROGRAMA AVISANDO NO VIDEO ERRO NOS PARAM.
0079 00AE-D 02 &
0080 00AE-D 02 BEGIN
0081 0004-P 03 DISPLAY(A"ERRO NA PASSAGEM DOS PARAMETROS.",32);
0082 0010-P 03 FIMPROG(A(0,ERRO.TERM));
0083 0018-P 03 END ERRA.PARMS;
0084 00D0-D 02 BYTE PROCEDURE SIMBOLO(BYTE CAR.PROCURADO);
0085 00D0-D 02 &
0086 00D0-D 02 & FUNCAO : DEVOLVE VERO SE CARACTER ATUAL FOR IGUAL AO DESEJADO.
0087 00D0-D 02 &
0088 00D0-D 02 & ROTINAS QUE A CHAMAM : ANALISA.PARMS
0089 00D0-D 02 &                        EXIJA
0090 00D0-D 02 &                        CABECA.PARM
0091 00D0-D 02 &                        MOVE.INFO.ARQ
0092 00D0-D 02 &
0093 00D0-D 02 BEGIN
0094 0022-P 03 COME.BRANCOS;
0095 0026-P 03 SIMBOLO:=CAR.ATUAL = CAR.PROCURADO;
0096 0034-P 03 END SIMBOLO;
0097 00D2-D 02 PROCEDURE EXIJA(BYTE CAR.EXIGIDO);
0098 00D2-D 02 &
0099 00D2-D 02 & FUNCAO : TERMINAR O PROGRAMA POR ERRO NA PASSAGEM DE PARAMETROS CASO
0100 00D2-D 02 &          CARACTER ATUAL NAO FOR O EXIGIDO PELA SINTAXE.
0101 00D2-D 02 &
0102 00D2-D 02 BEGIN
0103 0042-P 03 IF NOT SIMBOLO(CAR.EXIGIDO)
0104 004C-P 03 THEN
0105 0052-P 03      ERRA.PARMS;
0106 0056-P 03 AVANCA.PT;
0107 005A-P 03 END EXIJA;
0108 00D3-D 02 PROCEDURE MONTA.CAD.PARM(BYTE CAD.ALFA,MAX.CAD);
0109 00D3-D 02 &
0110 00D3-D 02 & FUNCAO : MONTAR CONFORME CONDICAO CAD.ALFA CADEIAS ALFANUMERICAS OU
0111 00D3-D 02 &          NUMERICAS.
0112 00D3-D 02 &
0113 00D3-D 02 BEGIN
0114 0068-P 03 TAM.TOKEN:=0;
0115 006C-P 03 INIC.TOKEN:=PT.REG.LIDO;
0116 0074-P 03 WHILE DIGITO
0117 0074-P 03       OR
0118 0078-P 03        (IF CAD.ALFA
0119 0078-P 03         THEN LETRA
0120 007F-P 03         ELSE 0)
0121 008C-P 03 DO
0122 0092-P 03    AVANCA.PT;
0123 0098-P 03 IF TAM.TOKEN > MAX.CAD
0124 0098-P 03 THEN
0125 00A2-P 03      ERRA.PARMS;
```

```
0126 00A6-P 03 END MONTA.CAD;
```

```
0128 00D5-D 02 PROCEDURE VOLUME;
0129 00D5-D 02 BEGIN
0130 00A8-P 03 AVANCA.PI;
0131 00AC-P 03 EXIJA("=");
0132 00B4-P 03 TIPO.UD.ARQ:="D ";
0133 00BE-P 03 COME.BRANCOS;
0134 00C2-P 03 MONTA.CAD.PARM(FALSO,5);
0135 00CC-P 03 VOL.ARQ:=ASCIIBIN(INIC.TOKEN,TAM.TOKEN);
0136 00E2-P 03 END VOLUME;
0137 00D5-D 02 PROCEDURE DISCO;
0138 00D5-D 02 BEGIN
0139 00E4-P 03 EXIJA("D");
0140 00EC-P 03 MONTA.CAD.PARM(VERO,3);
0141 00F6-P 03 MBI(INIC.TOKEN-1,&TIPO.UD.ARQ,TAM.TOKEN+1);
0142 010E-P 03 END DISCO;
```

```
0144 00D5-D 02 PROCEDURE MOVE.INFO.ARQ;
0145 00D5-D 02 &
0146 00D5-D 02 & FUNCAO : PREENCHER CADEIA DE INFORMACOES DOS ARQUIVOS TRATADOS COM
0147 00D5-D 02 &          NOME,TIPO,UNIDADE E VOLUME.
0148 00D5-D 02 &
0149 00D5-D 02 BEGIN
0150 0110-P 03 IF LIGA.OPC
0151 0110-P 03 THEN
0152 0116-P 03       BEGIN
0153 0116-P 04       LIGA.OPC:=FALSO;
0154 011A-P 04       RETURN;
0155 011C-P 04       END;
0156 011C-P 03 COME.BRANCOS;
0157 0120-P 03 IF DIGITO
0158 0120-P 03 THEN
0159 0128-P 03       BEGIN
0160 0128-P 04       INIC.TOKEN:=PT.REG.LIDO;
0161 0130-P 04       TAM.TOKEN:=0;
0162 0134-P 04       WHILE DIGITO
0163 0134-P 04       DO
0164 013C-P 04          AVANCA.PT;
0165 0142-P 04       UL.PRE.ALOC(QUAL.PARM):=ASCIIBIN(INIC.TOKEN,TAM.TOKEN);
0166 015A-P 04       RETURN;
0167 015C-P 04       END;
0168 015C-P 03 IF NOT LETRA
0169 015C-P 03 THEN
0170 0166-P 03       ERRA.PARMS;
0171 016A-P 03 MONTA.CAD.PARM(VERO,MAX.PAI);
0172 0174-P 03 MBT(INIC.TOKEN,END.CAD.ARQS,TAM.TOKEN); & MOVE NOME ARQ
0173 0184-P 03 IF SIMBOLO("I")
0174 018C-P 03 THEN
0175 0190-P 03       BEGIN
0176 0190-P 04       AVANCA.PT;
0177 0194-P 04       IF SIMBOLO("V")
0178 019C-P 04       THEN
0179 01A0-P 04            & VOLUME PASSADO COMO PARAM
0180 01A0-P 04            VOLUME
0181 01A0-P 04       ELSE
0182 01A6-P 04            DISCO;
0183 01AA-P 04       EXIJA("I");
0184 01B2-P 04       END;
0185 01B2-P 03 END MOVE.INFO.ARQ;
```

```
0187 00D5-D 02 BYTE PROCEDURE CABECA.PARM;
0188 00D5-D 02 &
0189 00D5-D 02 & FUNCAO : DEVOLVER VERO SE ATUAL SEGMENTO DA CADEIA DE PARAM FOR
0190 00D5-D 02 &
0191 00D5-D 02 &          INICIADA POR "E","T","R" OU "S" SEGUIDO DE ":".
0192 00D5-D 02 &
0193 00D5-D 02 BEGIN
0194 01B6-P 03 IF (QUAL.PARM:=SCANCACAR.PARM,4,CAR.ATUAL)) = -1
0195 01CA-P 03 THEN
0196 01D0-P 03       & ATUAL INICIO DE SUBCADEIA NAO E' CABECA DE PARAMETRO
0197 01D0-P 03       BEGIN
0198 01D0-P 04       CABECA.PARM:=FALSO;
0199 01D4-P 04       RETURN;
0200 01DC-P 04       END;
0201 01DC-P 03 IF SIMBOLO("S")
0202 01E4-P 03 THEN
0203 01E8-P 03       IF COMPARE(PT.REG.LIDO,4,@"SOFT",4) = 0
0204 01F8-P 03       THEN
0205 01FC-P 03            BEGIN
0206 01FC-P 04            CABECA.PARM:=FALSO;
0207 0200-P 04            RETURN;
0208 0202-P 04            END;
0209 0202-P 03 AVANCA.PT;
0210 0206-P 03 CABECA.PARM:=VERO;
0211 020A-P 03 IF QUAL.PARM = 3
0212 020A-P 03 THEN
0213 0212-P 03       & ARQ COM ROTULOS
0214 0212-P 03       TEM.ARQ.ROT:=VERO;
0215 0216-P 03 IF SIMBOLO(":")
0216 021E-P 03 THEN
0217 0222-P 03       BEGIN
0218 0222-P 04       AVANCA.PT;
0219 0226-P 04       END.CAD.ARQS:=@CAD.INFO.ARQS+QUAL.PARM*TAM.INFO.ARQ;
0220 023B-P 04       END
0221 0238-P 03 ELSE
0222 023A-P 03       IF QUAL.PARM = 0
0223 023A-P 03          AND
0224 023F-P 03          SIMBOLO("S")
0225 0248-P 03       THEN
0226 024E-P 03            BEGIN
0227 024E-P 04            AVANCA.PT;
0228 0252-P 04            LIGA.OPC:=VERO;
0229 0256-P 04            ESV.SE.EXISTE:=@0300
0230 0256-P 04            END
0231 025C-P 03       ELSE
0232 025E-P 03            ERRA.PARMS;
0233 0262-P 03 END CABECA.PARM;
```

```
0235 00DA-D 02 &
0236 00DA-P 02 & INICIO DOS COMANDOS :
0237 00DA-D 02 &
0238 00DA-D 02 CAD.PARMS(NB.LIDOS:=TAM.PARMS):=FIM.DE.REG;
0239 0272-P 02 MRT(PT.REG.LIDO:=ACAD.PARMS,ACABEC.PARMS+13,TAM.PARMS);
0240 028E-P 02 CONTA.CAR:=0;
0241 0292-P 02 WHILE NOT SIMBOLO(FIM.DE.REG)
0242 029A-P 02 DO
0243 02A0-P 02     & A PROCURA DE IDENTIFICACAO DE PARAM CONTINUA ATE' FIM.DE.REG
0244 02A0-P 02     BEGIN
0245 02A0-P 03     IF INICIO
0246 02A0-P 03     THEN INICIO:=FALSO
0247 02A6-P 03     ELSE BEGIN
0248 02AC-P 04          EXIJA(";");
0249 02B4-P 04          IF SIMBOLO(FIM.DE.REG)
0250 02BC-P 04          THEN RETURN;
0251 02C2-P 04          END;
0252 02C2-P 03     IF CABECA.PARM
0253 02C2-P 03     THEN
0254 02CA-P 03          & SUBCADEIA INICIADA POR "E","I","S" OU "R"
0255 02CA-P 03          MOVE.INFO.ARQ
0256 02CA-P 03     ELSE
0257 02D0-P 03          & OPCAO DE INDICAR SE OBJETO E' PARA SOFT OU HARD
0258 02D0-P 03          BEGIN
0259 02D0-P 04          IF COMPARE(PT.REG.LIDO,4,A"SOFT",4) = 0
0260 02E0-P 04          THEN
0261 02E4-P 04               SOFT:=VERO
0262 02E4-P 04          ELSE
0263 02EA-P 04               IF COMPARE(PT.REG.LIDO,4,A"HARD",4) <> 0
0264 02FA-P 04               THEN
0265 02FE-P 04                    ERRA.PARMS;
0266 0302-P 04          WHILE LETRA
0267 0302-P 04          DO AVANCA.PT;
0268 0310-P 04          END;
0269 0310-P 03     END;
0270 0312-P 02 END ANALISA.PARMS;
0271 00E2-D 01 END;

OBJETO:RPARMS  IDP08 V=10101 ] VERSAO=V.01   COD=MICROA  0005 REGISTROS
** FIM LPS **  ADVERTENCIAS:00     ERROS:00  MEMORIA: 80314-P + 800E2-D
```

ARQUIVO FONTE: FSEMANT [DP08] VOLUME=10101 VERSAO=V.01 CODIGO=MICROA
EQUIPAMENTO: C500
PARAMETROS: E:FSEMANT;S:RSEMANT;ES
OPCOES INICIAIS: LF NLC LA ES S 00 01 02 03 04 05 06 07 PFS

```
0002 0000-D 00 ?LPS,S,D,"SEMANT"
0003 0000-D 00 BEGIN
0004 0000-P 01 &
0005 0000-P 01 & CONSTANTES GERAIS ;
0006 0000-P 01 &
0007 0000-P 01 CONSTANT BRANCO      = " " ;
0008 0000-D 01 CONSTANT BYTE.VAZIO = ÖFF ;
0009 0000-D 01 CONSTANT DEF.F.S     = ÖFEFF ;
0010 0000-D 01 CONSTANT FALSO       = 0 ;
0011 0000-D 01 CONSTANT FIM.ARQ     = 1 ;
0012 0000-D 01 CONSTANT FIM.DE.REG = ÖFF ;
0013 0000-D 01 CONSTANT NAO.CANC    = Ö8000 ;
0014 0000-D 01 CONSTANT PARA.LEIT   = Ö1000 ;
0015 0000-D 01 CONSTANT PARA.SAIDA = Ö0800 ;
0016 0000-D 01 CONSTANT SEQL        = Ö0100 ;
0017 0000-D 01 CONSTANT SEQL.OBJ    = Ö0103 ;
0018 0000-D 01 CONSTANT VAZIO       = ÖFFFF ;
0019 0000-D 01 CONSTANT VERO        = 1 ;
0020 0000-D 01 CONSTANT UL.RELAT = 6 ;
0021 0000-D 01 &
0022 0000-D 01 & CONSTANTES DE LIMITES :
0023 0000-D 01 &
0024 0000-D 01 CONSTANT MAX.BIN     = 256 ;
0025 0000-D 01 CONSTANT MAX.DEC     = 5 ;
0026 0000-D 01 CONSTANT MAX.HEX     = 64 ;
0027 0000-D 01 CONSTANT MAX.PAL     = 9 ;
0028 0000-D 01 CONSTANT MAX.PERC    = 10 ;
0029 0000-D 01 CONSTANT MAX.REG.OBJ= 1024 ;
0030 0000-D 01 CONSTANT N.TERM.TAB = 12 ;
0031 0000-D 01 CONSTANT NUM.ENTR.TSIMB = 1024 ;
0032 0000-D 01 CONSTANT TPILROT     = NUM.ENTR.TSIMB*2 ;
0033 0000-D 01 CONSTANT TAM.AREA.COD = 9*256+5 ;
0034 0000-D 01 CONSTANT TAM.ENTR.TSIMB = 14 ;
0035 0000-D 01 CONSTANT TAM.INFO.ARG = 15;
0036 0000-D 01 CONSTANT TAM.TSIMB = TAM.ENTR.TSIMB*NUM.ENTR.TSIMB ;
0037 0000-D 01 CONSTANT TEXT.TAB = 512 ;
0038 0000-D 01 CONSTANT TEXT.MIC = 8*TEXT.TAB ;
0039 0000-D 01 CONSTANT TPSIMB      = 127 ;
0040 0000-D 01 &
0041 0000-D 01 & CONTROLES DO CAMPO NA TABSIMB :
0042 0000-D 01 &
0043 0000-D 01 CONSTANT CTL.TAB.CTE = 0 ;
0044 0000-D 01 CONSTANT CTL.TAB.EXT = 1 ;
0045 0000-D 01 CONSTANT CTL.TAB.ROT = 2 ;
```

```
0047 0000-D 01 &
0048 0000-D 01 & CONSTANT PARA OS ERROS :
0049 0000-D 01 &
0050 0000-D 01 CONSTANT ERRO.TRUNC.PAL          = 0 ;
0051 0000-D 01 CONSTANT ERRO.TRUNC.BIN          = 1 ;
0052 0000-D 01 CONSTANT ERRO.TRUNC.DEC          = 2 ;
0053 0000-D 01 CONSTANT ERRO.TRUNC.HEX          = 3 ;
0054 0000-D 01 CONSTANT ERRO.CAR.INESP          = 4 ;
0055 0000-D 01 CONSTANT ERRO.FIM.INESPERADO     = 5 ;
0056 0000-D 01 CONSTANT ERRO.EST.TAB.SIMB       = 6 ;
0057 0000-D 01 CONSTANT ERRO.MAQ.NVAL           = 7 ;
0058 0000-D 01 CONSTANT ERRO.RDEF.DESL          = 8 ;
0059 0000-D 01 CONSTANT ERRO.IL01.JALOC         = 9 ;
0060 0000-D 01 CONSTANT ERRO.OBJ.CH             = 10;
0061 0000-D 01 CONSTANT ERRO.EST.PILHA.SIMB     = 11;
0062 0000-D 01 CONSTANT ERRO.SINTATICO          = 12;
0063 0000-D 01 CONSTANT ERRO.ID.NAO.DECL        = 13;
0064 0000-D 01 CONSTANT ERRO.MNEM.INV           = 14;
0065 0000-D 01 CONSTANT ERRO.EXC.CPO            = 15;
0066 0000-D 01 CONSTANT ERRO.ID.JA.DECL         = 16;
0067 0000-D 01 CONSTANT ERRO.ROT.DUP            = 17;
0068 0000-D 01 CONSTANT ERRO.CPO.FALT           = 18;
0069 0000-D 01 CONSTANT ERRO.EST.TAB.USO.EXT    = 19;
0070 0000-D 01 CONSTANT ERRO.EST.TAB.USO.ROT    = 20;
0071 0000-D 01 CONSTANT ERRO.EST.TAB.EXT        = 21;
0072 0000-D 01 CONSTANT ERRO.EXT.HARD           = 22;
0073 0000-D 01 &
0074 0000-D 01 & CONTROLES DE CAMPO NA TABELA DE ENTRADA :
0075 0000-D 01 &
0076 0000-D 01 CONSTANT FIX     =  0 ;
0077 0000-D 01 CONSTANT COND    =  1 ;
0078 0000-D 01 CONSTANT LIT     =  2 ;
0079 0000-D 01 CONSTANT VAR     =  4 ;
0080 0000-D 01 CONSTANT LIT.VAR = 6 ;
0081 0000-D 01 &
0082 0000-D 01 CONSTANT TERM = 800 ;
0083 0000-D 01 CONSTANT HEX = TERM OR 4 ;
0084 0000-D 01 CONSTANT DEC = TERM OR 5 ;
```

```
0086 0000-D 01 &
0087 0000-D 01 & VARIAVEIS GLOBAIS :
0088 0000-D 01 &
0089 0000-D 01 & TABELA DE SIMBOLOS :
0090 0000-D 01 GLOBAL BYTE(*) TSEQ.SIMB = TAM.TSIMB : BYTE.VAZIO ;
0091 3800-D 01 &
0092 3800-D 01 GLOBAL BYTE AT.CPO = 1 ;
0093 3801-D 01 GLOBAL WORD D.CPO ;
0094 3803-D 01 GLOBAL WORD EBUF.LIT;
0095 3805-D 01 GLOBAL WORD ECOD.GERADO;
0096 3807-D 01 GLOBAL WORD EMEM.REC = 0 ;
0097 3809-D 01 GLOBAL WORD ETAB.SIMB ;
0098 380B-D 01 GLOBAL WORD IND = -1;
0099 380D-D 01 GLOBAL WORD IND.PROT = -1 ;
0100 380F-D 01 GLOBAL WORD N.ENTR = 0 ;
0101 3811-D 01 GLOBAL WORD NB;
0102 3813-D 01 GLOBAL WORD NMI ;
0103 3815-D 01 GLOBAL WORD NPAL.OBJ = 0 ;
0104 3817-D 01 GLOBAL WORD NUM.ORDEM = -1 ;
0105 3819-D 01 GLOBAL WORD(*) PIL.END.EXT.TAB = TEXT.TAB : VAZIO;
0106 3C19-D 01 GLOBAL WORD(*) PIL.END.EXT.MIC = TEXT.MIC : 07FFF;
0107 5C19-D 01 GLOBAL WORD(TPILROT) PIL.ROT ;
0108 6C19-D 01 GLOBAL WORD VAL.EMEM = 0 ;
```

```
0110 6C1B-D 01 GLOBAL PROCEDURE ACAO.SEMANT(BYTE ACAO);
0111 6C1B-D 01 &
0112 6C1B-D 01 & FUNCAO : PROVER A ANALISE SEMANTICA NECESSARIA.
0113 6C1B-D 01 &
0114 6C1B-D 01 BEGIN
0115 0008-P 02 &
0116 0008-P 02 & VARIAVEIS EXTERNAS :
0117 0008-P 02 &
0118 0008-P 02 & AREA ADICIONAL
0119 0008-P 02 EXTERNAL WORD EADIC;
0120 6C1C-D 02 EXTERNAL WORD TADIC;
0121 6C1C-D 02 WORD(*) ECPO.MEM REF EADIC;
0122 6C1C-D 02 EXTERNAL BYTE(*) CAD.INFO.ARQS;
0123 6C1C-D 02 EXTERNAL WORD CONTA.LINHAS.PAG;
0124 6C1C-D 02 WORD ESV.SE.EXISTE POS CAD.INFO.ARQS + 60;
0125 6C1C-D 02 BYTE SOFI POS CAD.INFO.ARQS + 62;
0126 6C1C-D 02 EXTERNAL WORD EBUF.CPO ;
0127 6C1C-D 02 BYTE(*) DESCR.CPO REF EBUF.CPO ;
0128 6C1C-D 02 WORD POS.CPO POS DESCR.CPO ;
0129 6C1C-D 02 BYTE CTL.CPO POS POS.CPO + 2 ;
0130 6C1C-D 02 BYTE TAM.CPO POS CTL.CPO + 1 ;
0131 6C1C-D 02 BYTE ENC.CPO POS TAM.CPO + 1 ;
0132 6C1C-D 02 & SE CAMPO FIXO OU CONDICIONAL :
0133 6C1C-D 02 BYTE(*) CPO.FIX POS ENC.CPO + 1 ;
0134 6C1C-D 02 & SE CAMPO VARIAVEL (LITERAL OU NAO) :
0135 6C1C-D 02 WORD MNEM.CAR POS ENC.CPO + 1 ;
0136 6C1C-D 02 BYTE(*) CPO.VAR POS MNEM.CAR + 2 ; & VARIAVEL NAO LITERAL
0137 6C1C-D 02 & PRIMEIRO REGISTRO DO ARQUIVO-TABELA
0138 6C1C-D 02 EXTERNAL WORD EPRIM.REG ;
0139 6C1C-D 02 BYTE(*) PRIM.REG REF EPRIM.REG ;
0140 6C1C-D 02 BYTE(9) NMAQ POS PRIM.REG ; & NOME DA MAQUINA
0141 6C1C-D 02 WORD    TPAL POS NMAQ + 9 ; & TAM DA PALAVRA A SER GERADA
0142 6C1C-D 02 BYTE    ENC.OBJ POS TPAL + 2 ; & DEFAULT DE PREENCHIMENTO DO OBJETO
0143 6C1C-D 02 BYTE(*) NCPO POS ENC.OBJ + 1 ; & VETOR DE NUM CAMPOS POSSIVEIS
0144 6C1C-D 02 EXTERNAL WORD INIC.ID;
0145 6C1C-D 02 EXTERNAL WORD INIC.TOKEN;
0146 6C1C-D 02 EXTERNAL BYTE NB.PAL;
0147 6C1C-D 02 EXTERNAL WORD NMI.REG ;
0148 6C1C-D 02 EXTERNAL BYTE QUAL.SIMB;
0149 6C1C-D 02 EXTERNAL BYTE(*) REG.OBJ ;
0150 6C1C-D 02 EXTERNAL BYTE TAM.ID;
0151 6C1C-D 02 EXTERNAL BYTE TAM.TOKEN;
0152 6C1C-D 02 EXTERNAL WORD TREG.OBJ ;
0153 6C1C-D 02 EXTERNAL WORD UL.OBJ;
0154 6C1C-D 02 EXTERNAL WORD VAL.DEC;
```

```
0156 6C1C-D 02 &
0157 6C1C-D 02 & VARIAVEIS LOCAIS :
0158 6C1C-D 02 &
0159 6C1C-D 02 BYTE(*) ENTRADA REF LTAB.SIMB ;
0160 6C1C-D 02 BYTE(9) NID POS ENTRADA ;
0161 6C1C-D 02 BYTE CTL.TAB POS NID + 9 ;
0162 6C1C-D 02 WORD ATRIB.TAB POS CTL.TAB + 1 ;
0163 6C1C-D 02 WORD PROX.ENDER POS ATRIB.TAB + 2 ;
0164 6C1C-D 02 & TAB HASH DE ENDERECOS DE ENTRADA NA TAB SEQ DE SIMB :
0165 6C1C-D 02 WORD(*) THASH.END = NUM.ENTR.TSIMB ; VAZIO ;
0166 741C-D 02 & CODIGO GERADO
0167 741C-D 02 BYTE(*) COD.GERADO REF ECOD.GERADO ;
0168 741C-D 02 & REGISTRO DE SAIDA PARA ARQUIVO OBJETO :
0169 741C-D 02 & LINHA DE IMPRESSAO PARA CODIGO GERADO :
0170 741C-D 02 BYTE(*) LIN.CGER = (" CODIGO GERADO : Ö",63:BRANCO) ;
0171 746D-D 02 BYTE(*) CGER POS LIN.CGER + 17 ;
0172 746D-D 02 BYTE(*) CAR POS 0;
0173 746D-D 02 &
0174 746D-D 02 WORD AUX,X,Y;
0175 7473-D 02 & HASH
0176 7473-D 02 WORD CONTA.COLISAO = 0 ;
0177 7475-D 02 BYTE(9) GID ;
0178 747E-D 02 BYTE ID.JA.DECL = FALSO ;
0179 747F-D 02 WORD INCREM ;
0180 7481-D 02 BYTE PERC.TSIMB = 0 ;
0181 7482-D 02 WORD PT.GID;
0182 7484-D 02 BYTE(*) CAR.ID REF PT.GID ;
0183 7484-D 02 WORD REC.XOR ;
0184 7486-D 02 & ANAL. SEMANT. ORG
0185 7486-D 02 WORD DIF.ENDER ;
0186 7488-D 02 WORD EMEM.ORG ;
0187 748A-D 02 WORD ID.REG = 1 ;
0188 748C-D 02 WORD NMI.MEM ;
0189 748E-D 02 WORD NMI.PULA ;
0190 7490-D 02 WORD NREG.PULA ;
```

```
0192 7492-D 02 & ANALISE SEMANTICA ;
0193 7492-D 02 WORD DC.GER;
0194 7494-D 02 BYTE EH.ORG = FALSO ;
0195 7495-D 02 BYTE ERRO.TRUNC ;
0196 7496-D 02 BYTE MAX.CTE ;
0197 7497-D 02 WORD MNEM.ROT = 0 ;
0198 7499-D 02 WORD PALVRA ;
0199 749B-D 02 BYTE P.CPO = 0 ;
0200 749C-D 02 WORD PIL.ROT.FFFF = 0 ;
0201 749E-D 02 WORD POS.BIT ;
0202 74A0-D 02 WORD POS.ROT ;
0203 74A2-D 02 WORD REG.ATUAL ;
0204 74A4-D 02 BYTE ROT.INDEF = FALSO ;
0205 74A5-D 02 BYTE TAM.ROT ;
0206 74A6-D 02 WORD ATREG;
0207 74A8-D 02 WORD TOTREG;
0208 74AA-D 02 &
0209 74AA-D 02 & PROCEDIMENTOS EXTERNOS :
0210 74AA-D 02 &
0211 74AA-D 02 PROCEDURE ERRO(WORD NUM.ERRO);
0212 74AA-D 02   EXTERNAL;
0213 74AA-D 02 PROCEDURE POP.TOPO;
0214 74AA-D 02   EXTERNAL;
```

```
0216 74AA-D 02 PROCEDURE RW.AT;
0217 74AA-D 02 & ROTINAS QUE A CHAMAM : GRAVA.OU.REGRAVA
0218 74AA-D 02 &                        ACAO 7
0219 74AA-D 02 BEGIN
0220 000C-P 03 IF NPAL.OBJ = 0
0221 000C-P 03 THEN
0222 0012-P 03       RETURN;
0223 0014-P 03 REWRITE(UL.OBJ,AREG.OBJ,TREG.OBJ);
0224 0026-P 03 FILL(AREG.OBJ,TREG.OBJ,ENC.OBJ);
0225 003C-P 03 END RW.AT;
0226 74AA-D 02 GLOBAL PROCEDURE GRAVA.OU.REGRAVA(WORD NB.GRAV);
0227 74AA-D 02 & ROTINAS QUE A CHAMAM : ACAO 7
0228 74AA-D 02 &                        GR.REG.CH
0229 74AA-D 02 &                        FIM.DE.GERACAO
0230 74AA-D 02 BEGIN
0231 0046-P 03 IF (ATREG:=CURRENT(UL.OBJ)) < (TOTREG:=TOTREC(UL.OBJ))
0232 0064-P 03 THEN
0233 006A-P 03       BEGIN
0234 006A-P 04       RW.AT;
0235 006E-P 04       RETURN;
0236 0070-P 04       END;
0237 0070-P 03 IF ID.REG = TOTREG
0238 0070-P 03 THEN
0239 007A-P 03       REWRITE(UL.OBJ,AREG.OBJ,NB.GRAV)
0240 008C-P 03 ELSE
0241 008E-P 03       BEGIN
0242 008E-P 04       WRITE(UL.OBJ,AREG.OBJ,NB.GRAV);
0243 00A0-P 04       ID.REG:=1+CURRENT(UL.OBJ);
0244 00B0-P 04       END;
0245 00B0-P 03 FILL(AREG.OBJ,TREG.OBJ,ENC.OBJ);
0246 00C6-P 03 END GRAVA.OU.REGRAVA;
0247 74AC-D 02 PROCEDURE GR.REG.CH;
0248 74AC-D 02 & ROTINAS QUE A CHAMAM : ACAO 4
0249 74AC-D 02 &                        ACAO 7
0250 74AC-D 02 BEGIN
0251 00C8-P 03 GRAVA.OU.REGRAVA(TREG.OBJ);
0252 00D2-P 03 NPAL.OBJ:=1-TREG.OBJ;
0253 00DA-P 03 IF ATREG < TOTREG
0254 00DA-P 03 THEN
0255 00E4-P 03       BEGIN
0256 00E4-P 04       READ(UL.OBJ,AREG.OBJ,ANB);
0257 00F6-P 04       ID.REG:=ATREG+1;
0258 0100-P 04       END;
0259 0100-P 03 END GR.REG.CH;
```

```
0261 74AC-D 02 WORD PROCEDURE BITBIN(WORD ECAD,TCAD);
0262 74AC-D 02 BEGIN
0263 010E-P 03 PALVRA:=0;
0264 0112-P 03 WHILE TCAD > 0
0265 0112-P 03 DO
0266 0118-P 03     BEGIN
0267 0118-P 04       IF CAR(ECAD)
0268 011C-P 04       THEN
0269 0122-P 04             & CARACTER F' 1
0270 0122-P 04             SET(&PALVRA,16-TCAD);
0271 0132-P 04       ECAD:=!+1;
0272 0136-P 04       TCAD:=!-1;
0273 013A-P 04       END;
0274 013C-P 03 BITBIN:=PALVRA;
0275 0144-P 03 END BITBIN;
```

```
0277 74B2-D 02 GLOBAL PROCEDURE GERA.CODIGO;
0278 74B2-D 02 &
0279 74B2-D 02 & FUNCAO : TRANSCREVER A MICRO-ORDEM PARA A MICRO-INSTRUCAO.
0280 74B2-D 02 &
0281 74B2-D 02 & ROTINAS QUE A CHAMAM : TRATA.CPO.LIT
0282 74B2-D 02 &                        PROC.MNEM
0283 74B2-D 02 &
0284 74B2-D 02 BEGIN
0285 014A-P 03 DC.GER:=POS.CPO+TAM.CPO;
0286 015A-P 03 FOR X:=1 TO TAM.CPO
0287 0164-P 03 DO
0288 0170-P 03     IF BIT(@D.CPO,16-X)
0289 0180-P 03     THEN & BIT LIGADO
0290 0184-P 03          SET(@COD.GERADO,DC.GER-X)
0291 0196-P 03     ELSE & BIT DESLIGADO
0292 0198-P 03          CLEAR(@COD.GERADO,DC.GER-X);
0293 01B6-P 03 IF AT.CPO > NCPO(MNEM.ROT)
0294 01C2-P 03 THEN
0295 01CA-P 03       & EXCESSO DE CAMPOS
0296 01CA-P 03       ERRO(ERRO.EXC.CPO);
0297 01D2-P 03 AT.CPO:=I+1;
0298 01D6-P 03 END GERA.CODIGO;
```

```
0300 7484-D 02 PROCEDURE PREENCHE.TSEQ.SIMB(WORD ENDER);
0301 7484-D 02 BEGIN
0302 01E0-P 03 WORD CONTEUDO REF ENDER;
0303 7486-D 03 MBT(INIC.ID,CONTEUDO:=ATSEQ.SIMB+TAM.ENTR.TSIMBAN.ENTR,TAM.ID);
0304 0200-P 03 IF (N.ENTR:=:+1) = NUM.ENTR.TSIMB
0305 0204-P 03 THEN
0306 020F-P 03       % ESTOURO DA TABELA DE SIMBOLOS
0307 020F-P 03       ERRO(ERRO.EST.TAB.SIMB);
0308 0216-P 03 ID.JA.DECL:=FALSO;
0309 021A-P 03 END PREENCHE.TSEQ.SIMB;
```

```
0311 74B6-D 02 PROCEDURE HASH;
0312 74B6-D 02 &
0313 74B6-D 02 & FUNCAO : DETERMINAR O ENDERECO DE ENTRADA NA TABELA DE SIMBOLOS, QUE
0314 74B6-D 02 &          SERA' PREENCHIDA COM ID'S DE ROTULO, CONSTANTE E VARIAVEL EXT
0315 74B6-D 02 &          A FUNCAO DE HASH PASSA POR TRES FASES :
0316 74B6-D 02 &            - FASE DE OU-EXCLUSIVO - E' FEITO OU-EXCLUIVO ENTRE WORD'S
0317 74B6-D 02 &              COMPOSTAS PELA ASSOCIACAO DE CADA DOIS CARACTERES DO ID,
0318 74B6-D 02 &              SENDO QUE DE CADA CARACTER E' RETIRADO O 1. E S. BIT MAI
0319 74B6-D 02 &              SIGNIFICATIVO PARA ENTAO SER COMPACTADO EM 6 BITS E TRANS
0320 74B6-D 02 &              NA WORD.
0321 74B6-D 02 &            - FASE DE POTENCIA - A WORD RESULTANTE DA FASE ANTERIOR E'
0322 74B6-D 02 &              ELEVADA AO QUADRADO E ENTAO SAO TOMADOS OS 10 BITS MENOS
0323 74B6-D 02 &              SIGNIFICATIVOS PARA ENTRADA NA TABELA DE SIMBOLOS.
0324 74B6-D 02 &            - FASE DE COLISAO - EM CASO DE COLISAO E' USADO UM INCREME
0325 74B6-D 02 &              QUE DEPENDE DOS DOIS PRIMEIROS CARACTERES DO ID.
0326 74B6-D 02 &
0327 74B6-D 02 BEGIN
0328 021C-P 03 PT.GID:=AGID;
0329 0224-P 03 FILL(PT.GID,9,BYTE.VAZIO);
0330 0232-P 03 MBT(INIC.ID,PT.GID,TAM.ID);
0331 0242-P 03 REC.XOR:=BRANCO;
0332 0248-P 03 FOR AUX:=0 TO 3
0333 024C-P 03 DO
0334 0258-P 03     & FASE OU-EXCLUSIVO :
0335 0258-P 03     BEGIN
0336 0258-P 04     REC.XOR:=! XOR (((CAR.ID(0) AND 80007) OR ((CAR.ID(0) SHR 1) AND 8003
0337 026F-P 04                        OR
0338 026F-P 04                         (((EXPS CAR.ID(1)) RTL 6) AND 801C0)
0339 027A-P 04                         OR
0340 027A-P 04                         ((EXPS(CAR.ID(1) SHR 1) RTL 6) AND 80E00));
0341 0296-P 04     PT.GID:=!+2;
0342 029A-P 04     END;
0343 02A6-P 03 REC.XOR:=((REC.XOR*REC.XOR) AND 803FF);&FASE DE POTENCIA
```

```
0345 02B6-P 03 IF THASH.END(REC.XOR) = VAZIO
0346 02BC-P 03 THEN
0347 02C6-P 03       & ENTRADA VAZIA NA TAB HASH ==> ID NAO DECL
0348 02C6-P 03       BEGIN
0349 02C6-P 04       PREENCHE.TSEQ.SIMB(@ETAB.SIMB);
0350 02D0-P 04       THASH.END(REC.XOR):=ETAB.SIMB;
0351 02DF-P 04       RETURN;
0352 02E0-P 04       END;
0353 02E0-P 03 ETAB.SIMB:=THASH.END(REC.XOR);
0354 02EE-P 03 IF COMPARE(@GID,9,ETAB.SIMB,9) = 0
0355 02FE-P 03 THEN
0356 0302-P 03       & ID JA DECL
0357 0302-P 03       BEGIN
0358 0302-P 04       ID.JA.DECL:=VERO;
0359 0306-P 04       RETURN;
0360 0308-P 04       END;
0361 0308-P 03 & COLISAO :
0362 0308-P 03 CONTA.COLISAO:=!+1;
0363 030C-P 03 WHILE PROX.ENDER <> VAZIO
0364 0312-P 03 DO
0365 031A-P 03     & PROCURA ENTRADA LIVRE
0366 031A-P 03     BEGIN
0367 031A-P 04     IF COMPARE(@GID,9,ETAB.SIMB:=PROX.ENDER,9) = 0
0368 0332-P 04     THEN
0369 0336-P 04           & ID JA' DECLARADO
0370 0336-P 04           BEGIN
0371 0336-P 05           ID.JA.DECL:=VERO;
0372 033A-P 05           RETURN;
0373 033C-P 05           END;
0374 033C-P 04     END;
0375 033E-P 03 PREENCHE.TSEQ.SIMB(@PROX.ENDER);
0376 034A-P 03 ETAB.SIMB:=PROX.ENDER;
0377 0354-P 03 END HASH;
```

```
0379 7488-D 02 PROCEDURE TRATA.CPO.LIT;
0380 7488-D 02 &
0381 7488-D 02 & FUNCAO : TRATAR CAMPOS DA MICRO-INSTRUCAO QUE SEJAM LITERAIS.
0382 7488-D 02 &
0383 7488-D 02 BEGIN
0384 0356-P 03 ERUF.LIT:=ERUF.CPO;
0385 035E-P 03 IF INIC.ID = VAZIO
0386 035E-P 03 THEN
0387 0368-P 03       & DEFAULT DE ENCHIMENTO DEVE SER ASSUMIDO
0388 0368-P 03       BEGIN
0389 0368-P 04       D.CPO:=ENC.CPO;
0390 0374-P 04       GERA.CODIGO;
0391 0378-P 04       RETURN;
0392 037A-P 04       END;
0393 037A-P 03 HASH;
0394 037E-P 03 D.CPO:=ATRIB.TAB;
0395 0388-P 03 IF (CTL.TAB AND 07F) < CTL.TAB.ROT
0396 0392-P 03 THEN
0397 0396-P 03       & ENTRADA E' DECL DE CONSTANTE OU VARIAVEL EXTERNA
0398 0396-P 03       BEGIN
0399 0396-P 04       IF NOT ID.JA.DECL
0400 0396-P 04       THEN
0401 039F-P 04            BEGIN
0402 039F-P 05            ERRO(ERRO.ID.NAO.DECL);
0403 03A6-P 05            AT.CPO:=I+1;
0404 03AA-P 05            END;
0405 03AA-P 04       IF CTL.TAB = CTL.TAB.EXT
0406 03B0-P 04       THEN
0407 03B6-P 04            BEGIN
0408 03B6-P 05            IF (IND:=I+1) > TEXT.MIC
0409 03BA-P 05            THEN
0410 03C4-P 05                 BEGIN
0411 03C4-P 06                 ERRO(ERRO.EST.TAB.USO.EXT);
0412 03CC-P 06                 IND:=0;
0413 03D0-P 06                 END;
0414 03D0-P 05            PIL.END.EXT.MIC(IND):=VAL.EMEM;
0415 03DE-P 05            END;
0416 03DE-P 04       END
0417 03DE-P 03 ELSE
0418 03E0-P 03       & ENTRADA E' DECL DE ROTULO
0419 03E0-P 03       IF (ID.JA.DECL AND ATRIB.TAB=VAZIO)
0420 03F2-P 03          OR
0421 03F2-P 03          (NOT ID.JA.DECL)
0422 03F8-P 03       THEN
0423 03FC-P 03            BEGIN
0424 03FC-P 04            ROT.INDEF:=VERO;
0425 0400-P 04            POS.ROT:=POS.CPO/4;
0426 040C-P 04            TAM.ROT:=TAM.CPO/4 + (IF (TAM.CPO MOD 4) = 0 THEN 0 ELSE 1) ;
0427 0432-P 04            D.CPO:=ETAB.SIMB;
0428 043A-P 04            PIL.ROT(IND.PROT:=I+1):=VAL.EMEM; & EMPILHA NUMERO DA MICRO-IN
```

```
0429 044C-P 04          PIL.ROT(IND.PROT:=I+1):=ETAB.SIMB; & EMPILHA ENDER DE ROT EM T
0430 045E-P 04          IF (IND.PROT+1) >= TPILROT
0431 0464-P 04          THEN
0432 046A-P 04              BEGIN
0433 046A-P 05              ERRO(ERRO.EST.TAB.USO.ROT);
0434 0472-P 05              IND.PROT:=-1;
0435 0476-P 05              END;
0436 0476-P 04          END;
0437 0476-P 03 GERA.CODIGO;
0438 047A-P 03 END TRATA.CPO.LIT;
```

```
0440 7488-D 02 PROCEDURE PROC.MNEM(WORD ECPO);
0441 7488-D 02 &
0442 7488-D 02 & FUNCAO : PROCURAR ENTRE OS MNEMONICOS DA ATUAL MICRO-ORDEM O EQUIVALEN
0443 7488-D 02 &          AO DO CAMPO DE ENTRADA.
0444 7488-D 02 &
0445 7488-D 02 BEGIN
0446 0484-P 03 IF INIC.ID = VAZIO
0447 0484-P 03 THEN
0448 048E-P 03       & DEFAULT DE PREENCHIMENTO DEVE SER ASSUMIDO
0449 048E-P 03       BEGIN
0450 048E-P 04       D.CPO:=ENC.CPO;
0451 049A-P 04       GERA.CODIGO;
0452 049E-P 04       RETURN;
0453 04A0-P 04       END;
0454 04A0-P 03 & MOVIMENTA MNEM DE ENTRADA PARA PESQ SEQUENCIAL :
0455 04A0-P 03 MBT(INIC.ID,X:=ECPO.MEM(CAT.CPO+1)-9,TAM.ID);
0456 04C2-P 03 FILL(X+TAM.ID,9-TAM.ID,BRANCO);
0457 04DA-P 03 D.CPO:=0;
0458 04DE-P 03 & PESQUISA SEQUENCIAL :
0459 04DE-P 03 WHILE COMPARE(X,9,ECPO+D.CPO,9) <> 0
0460 04F2-P 03 DO
0461 04F6-P 03       D.CPO:=!+9;
0462 04FC-P 03 IF X=ECPO+D.CPO
0463 04FC-P 03 THEN & ID NAO CONSTA DA LISTA DE MNEMONICOS DESSE CAMPO
0464 050A-P 03       BEGIN
0465 050A-P 04       HASH;
0466 050E-P 04       IF NOT (ID.JA.DECL AND (CTL.TAB = CTL.TAB.CTE OR CTL.TAB = CTL.TAB.
0467 052A-P 04       THEN
0468 052E-P 04            BEGIN
0469 052E-P 05            ERRO(ERRO.MNEM.INV);
0470 0536-P 05            D.CPO:=0;
0471 053A-P 05            END
0472 053A-P 04       ELSE
0473 053C-P 04            BEGIN
0474 053C-P 05            Y:=TAM.CPO;
0475 0548-P 05            AUX:=POS.CPO;
0476 0550-P 05            TAM.CPO:=16;
0477 055A-P 05            D.CPO:=ATRIB.TAB;
0478 0564-P 05            POS.CPO:=TPAL-16;
0479 0574-P 05            AT.CPO:=NCPO(MNEM.ROT);
0480 0584-P 05            GERA.CODIGO;
0481 0588-P 05            IF CTL.TAB = CTL.TAB.EXT
0482 058E-P 05            THEN
0483 0594-P 05                 BEGIN
0484 0594-P 06                 IF (IND:=!+1) > TEXT.MIC
0485 0598-P 06                 THEN
0486 05A2-P 06                      BEGIN
0487 05A2-P 07                      ERRO(ERRO.EST.TAB.USO.EXT);
0488 05AA-P 07                      IND:=0;
0489 05AE-P 07                      END;
```

```
0490 05AE-P 06                    PIL.END.EXT.MIC(IND):=VAL.EMEM;
0491 05BC-P 06                    END;
0492 05BC-P 05                TAM.CPO:=Y;
0493 05C6-P 05                POS.CPO:=AUX;
0494 05CE-P 05                RETURN;
0495 05D0-P 05                END;
0496 05D0-P 04          END;
0497 05D0-P 03 D.CPO:=179;
0498 05DC-P 03 GERA.CODIGO;
0499 05E0-P 03 END PROC.MNEM;
```

```
0501 74BA-D 02 &
0502 74BA-D 02 & INICIO DOS COMANDOS ;
0503 74BA-D 02 &
0504 74BA-D 02 POP.TOPO;
0505 05E6-P 02 CASE ACAO OF
0506 05F0-P 02 BEGIN
0507 05F0-P 03         &
0508 05F0-P 03         & ACAO 0 : D ~=> = [0] N
0509 05F0-P 03         &
0510 05F0-P 03         & ID E' DECLARACAO DE CONSTANTE
0511 05F0-P 03         &
0512 05F0-P 03         BEGIN
0513 05F0-P 04         HASH;                & INSERCAO DE ID NA TAB SIMB
0514 05F4-P 04         IF ID.JA.DECL
0515 05F4-P 04         THEN
0516 05FA-P 04              & ID NAO PODE TER SIDO USADO ANTERIORMENTE
0517 05FA-P 04              ERRO(ERRO.ID.JA.DECL);
0518 0602-P 04         CTL.TAB:=CTL.TAB.CTE;
0519 060C-P 04         IF QUAL.SIMB = DEC
0520 060C-P 04         THEN
0521 0614-P 04              ATRIB.TAB:=VAL.DEC
0522 061A-P 04         ELSE
0523 0620-P 04              IF QUAL.SIMB=HEX
0524 0620-P 04              THEN
0525 0628-P 04                   BEGIN
0526 0628-P 05                   WHILE TAM.TOKEN > 4
0527 0628-P 05                   DO
0528 0630-P 05                        BEGIN
0529 0630-P 06                        ERRO(ERRO.TRUNC.HEX);
0530 0638-P 06                        INIC.TOKEN:=I+1;
0531 063C-P 06                        TAM.TOKEN:=I-1;
0532 0640-P 06                        END;
0533 0642-P 05                   ATRIB.TAB:=HEXBIN(INIC.TOKEN,TAM.TOKEN);
0534 065A-P 05                   END
0535 065A-P 04              ELSE
0536 065C-P 04                   ERRO(ERRO.SINTATICO);
0537 0664-P 04         END;
```

```
0539 0668-P 03          &
0540 0668-P 03          & ACAO 1 : D --> EXTERNA [1]
0541 0668-P 03          &
0542 0668-P 03          & ID E' DECLARACAO DE VARIAVEL EXTERNA
0543 0668-P 03          &
0544 0668-P 03          BEGIN
0545 0668-P 04          HASH;                   & INSERCAO DE ID NA TAB SIMB
0546 066C-P 04          IF ID.JA.DECL
0547 066C-P 04          THEN
0548 0672-P 04              ERRO(ERRO.ID.JA.DECL);
0549 067A-P 04          CTL.TAB:=CTL.TAB.EXT;
0550 0680-P 04          IF (ATRIB.TAB:=NUM.ORDEM:=I+1) >= IEXT.TAB
0551 0692-P 04          THEN
0552 069A-P 04              BEGIN
0553 069A-P 05                  ERRO(ERRO.EST.TAB.EXT);
0554 06A2-P 05                  NUM.ORDEM:=-1;
0555 06A6-P 05              END;
0556 06A6-P 04          IF NOT SOFT
0557 06A6-P 04          THEN
0558 06AE-P 04              ERRO(ERRO.EXT.HARD);
0559 06B6-P 04          PIL.END.EXT.TAB(NUM.ORDEM):=ETAB.SIMB;
0560 06C4-P 04          END;
0561 06C6-P 03          &
0562 06C6-P 03          & ACAO 2 : MIS --> [2] : LC
0563 06C6-P 03          &
0564 06C6-P 03          & ID E' DECLARACAO DE ROTULO
0565 06C6-P 03          &
0566 06C6-P 03          BEGIN
0567 06C6-P 04          HASH;
0568 06CA-P 04          IF ID.JA.DECL
0569 06CA-P 04             AND
0570 06CA-P 04             ATRIB.TAB <> VAZIO
0571 06D0-P 04          THEN
0572 06DE-P 04              & ESSE MESMO ROTULO JA' PRECEDE OUTRA MICRO-INSTRUCAO
0573 06DE-P 04              ERRO(ERRO.ROT.DUP);
0574 06E6-P 04          CTL.TAB:=CTL.TAB.ROT;
0575 06F0-P 04          ATRIB.TAB:=VAL.EMEM;
0576 06FA-P 04          END;
```

```
0578 06FC-P 03          &
0579 06FC-P 03          & ACAO 3 : C --> ID [3]
0580 06FC-P 03          &
0581 06FC-P 03          &          C --> [3] EPSILON
0582 06FC-P 03          &
0583 06FC-P 03          &          B --> [3] LC1
0584 06FC-P 03          &
0585 06FC-P 03          &          MIS --> [3] LC1
0586 06FC-P 03          &
0587 06FC-P 03          & ID E' DECLARACAO DE MICRO-ORDEM
0588 06FC-P 03          &
0589 06FC-P 03          BEGIN
0590 06FC-P 04          LABEL PROX.CPO;
0591 7A0A-D 04 PROX.CPO:
0592 06FC-P 04          EBUF.CPO:=ECPO.MEM(AT.CPO+P.CPO);
0593 0710-P 04          CASE CTL.CPO OF
0594 071F-P 04          BEGIN
0595 071F-P 05                  & CAMPO E' FIXO :
0596 071F-P 05                  PROC.MNEM(ACPO,FIX);
0597 072F-P 05                  & CAMPO E' CONDICIONAL :
0598 072F-P 05                  BEGIN
0599 072F-P 06                  PROC.MNEM(ACPO,FIX);
0600 073A-P 06                  MNEM.ROT:=D.CPO;
0601 0742-P 06                  END;
0602 0744-P 05                  & CAMPO E' LITERAL NAO VARIAVEL :
0603 0744-P 05                  TRATA.CPO.LIT;
0604 074A-P 05                  & CTL.CPO=3 : VAZIO
0605 074A-P 05                  BEGIN
0606 074A-P 06                  END;
0607 074C-P 05                  & CAMPO E' VARIAVEL NAO LITERAL :
0608 074C-P 05                  BEGIN
0609 074C-P 06                  IF MNEM.CAR <> MNEM.ROT
0610 0752-P 06                  THEN
0611 075A-P 06                          & ESSE CAMPO NAO CORRESPONDE AO MNEMONICO CONDICIONAL
0612 075A-P 06                          BEGIN
0613 075A-P 07                          P.CPO:=I+1;
0614 075E-P 07                          GOTO PROX.CPO;
0615 0760-P 07                          END;
0616 0760-P 06                  PROC.MNEM(ACPO,VAR);
0617 076C-P 06                  END;
0618 076E-P 05                  & CTL.CPO=5 : VAZIO
0619 076E-P 05                  BEGIN
0620 076E-P 06                  END;
0621 0770-P 05                  & CAMPO E' VARIAVEL LITERAL :
0622 0770-P 05                  BEGIN
0623 0770-P 06                  IF MNEM.CAR <> MNEM.ROT
0624 0776-P 06                  THEN
0625 077E-P 06                          BEGIN
0626 077E-P 07                          P.CPO:=I+1;
0627 0782-P 07                          GOTO PROX.CPO;
```

```
0628 0784-P 07                    END;
0629 0784-P 06                 TRATA.CPO.LTI;
0630 0788-P 06                  END;
0631 078A-P 05              END;
0632 079C-P 04           END;
0633 079E-P 03           &
0634 079F-P 03           & ACAO 4 : LI --> [4] ; I LI
0635 079F-P 03           &
0636 079F-P 03           &              P --> ... A LI [4] .
0637 079F-P 03           &
0638 079F-P 03           & FIM DE MICRO-INSTRUCAO
0639 079F-P 03           &
0640 079F-P 03           IF EH.ORG
0641 079F-P 03           THEN
0642 07A4-P 03                 & ULTIMO CARTAO FOI DE ORG
0643 07A4-P 03                 BEGIN
0644 07A4-P 04                 EH.ORG:=FALSO;
0645 07A8-P 04                 VAL.EMEM:=EMEM.ORG;
0646 07B0-P 04                 END
0647 07B0-P 03           ELSE
0648 07B2-P 03                 BEGIN
0649 07B2-P 04                 IF AT.CPO <> NCPO(MNEM.ROT) + 1
0650 07BE-P 04                 THEN
0651 07C8-P 04                       & FALTAM AINDA MICRO-ORDENS PARA COMPOR A MICRO-INSTRUCA
0652 07C8-P 04                       ERRO(ERRO.CPO.FALT);
0653 07D0-P 04                 & INICIALIZACOES PARA INICIO DE MICRO-INSTRUCAO :
0654 07D0-P 04                 AT.CPO:=1;
0655 07D4-P 04                 VAL.EMEM:=I+1;
0656 07D8-P 04                 MNEM.ROT:=P.CPO:=0;
0657 07E0-P 04                 &TRANSFORMA BINARIO EM HEXADECIMAL PARA IMPRESSAO :
0658 07E0-P 04                 AUX:=NB.PAL-1;
0659 07EC-P 04                 FOR X:=0 TO AUX
0660 07F0-P 04                 DO
0661 07FE-P 04                       BINHEX(EXPZ COD.GERADO(X),ACGER+X+X,2,0);
0662 082A-P 04                 IF ROT.INDEF
0663 082A-P 04                 THEN
0664 0830-P 04                       & ROTULO INDEFINIDO : COLOCA * NO COD GERADO
0665 0830-P 04                       BEGIN
0666 0830-P 05                       ROT.INDEF:=FALSO;
0667 0834-P 05                       MBT(A"****",ACGER+POS.ROT,TAM.ROT);
0668 0848-P 05                       END;
0669 0848-P 04                 CONTA.LINHAS.PAG:=I+1;
0670 084C-P 04                 WRITE(UL.RELAT,ALIN.CGER,81);
0671 085A-P 04                 & PREENCHE OBJETO :
0672 085A-P 04                 MBT(ACOD.GERADO,AREG.OBJ+NPAL.OBJ,X);
0673 086E-P 04                 FILL(ACOD.GERADO,X,ENC.OBJ);
0674 0884-P 04                 IF (NPAL.OBJ:=I+X) = TREG.OBJ
0675 088C-P 04                 THEN
0676 0894-P 04                       & NUMERO DE PALAVRAS GERADAS COMPLETA TAM REG OBJETO
0677 0894-P 04                       BEGIN
```

```
0678 0894-P 05                    GR.REG.CH;
0679 0898-P 05                    END;
0680 0898-P 04                END;
```

```
0682 089A-P 03          &
0683 089A-P 03          & ACAO 5 : P --> MAQUINA ID [5] ; ...
0684 089A-P 03          &
0685 089A-P 03          & INICIO DO MICRO-PROGRAMA :
0686 089A-P 03          &
0687 089A-P 03          BEGIN
0688 089A-P 04          MBT(INIC.ID,AGID,TAM.ID);
0689 08AA-P 04          FILL(AGID+TAM.ID,9-TAM.ID,BRANCO);
0690 08C8-P 04          IF COMPARE(ANMAQ,9,AGID,9) <> 0
0691 08D8-P 04          THEN
0692 08DC-P 04              ERRO(ERRO.MAQ.NVAL);
0693 08E4-P 04          FILL(XCOD,GERADO,NB.PAL-1,ENC.OBJ);
0694 08FE-P 04          END;
0695 0900-P 03          &
0696 0900-P 03          & ACAO 6 : CT --> [6] HEX
0697 0900-P 03          &
0698 0900-P 03          &              CT --> [6] BIN
0699 0900-P 03          &
0700 0900-P 03          & CAMPO E' CONSTANTE
0701 0900-P 03          &
0702 0900-P 03          BEGIN
0703 0900-P 04          LBUF.CPO:=ECPO.MEM(AT.CPO+P.CPO);
0704 0914-P 04          IF QUAL.SIMB = HEX
0705 0914-P 04          THEN
0706 091C-P 04              & CONSTANTE HEXADECIMAL
0707 091C-P 04              BEGIN
0708 091C-P 05              MAX.CTE:=AUX:=4;
0709 0924-P 05              ERRO.TRUNC:=ERRO.TRUNC.HEX;
0710 0928-P 05              END
0711 0928-P 04          ELSE
0712 092A-P 04              & CONSTANTE BINARIA
0713 092A-P 04              BEGIN
0714 092A-P 05              MAX.CTE:=16;
0715 0930-P 05              AUX:=1;
0716 0934-P 05              ERRO.TRUNC:=ERRO.TRUNC.BIN;
0717 0938-P 05              END;
0718 0938-P 04          X:=POS.CPO+TAM.TOKEN*AUX;
0719 0948-P 04          IF X < TPAL
0720 094E-P 04          THEN
0721 0956-P 04              & CONSTANTE REPRESENTA APENAS UMA MICRO-ORDEM
0722 0956-P 04              BEGIN
0723 0956-P 05              WHILE TAM.TOKEN > MAX.CTE
0724 0956-P 05              DO
0725 0960-P 05                  & EXCESSO DE CARAC HEXA
0726 0960-P 05                  BEGIN
0727 0960-P 06                  ERRO(ERRO.TRUNC);
0728 096A-P 06                  TAM.TOKEN:=!-1;
0729 096E-P 06                  INIC.TOKEN:=!+1;
0730 0972-P 06                  END;
0731 0974-P 05              D.CPO:=IF QUAL.SIMB=HEX THEN HEXBIN(INIC.TOKEN,TAM.TOKEN)
```

```
0732 098A-P 05                                          ELSE BITBIN(INIC.TOKEN,TAM.TOKEN);
0733 099E-P 05                  GERA.CODIGO;
0734 09A2-P 05                  IF CTL.CPO = COND
0735 09A8-P 05                  THEN
0736 09AE-P 05                      & CAMPO E' CONDICIONAL
0737 09AE-P 05                      MNEM.ROT:=D.CPO AND (0FFFF SHR (16-TAM.CPO));
0738 09CA-P 05                  END
0739 09CA-P 04              ELSE
0740 09CC-P 04                  & CONSTANTE REPRESENTA O RESTO DA MICRO-INSTRUCAO
0741 09CC-P 04                  BEGIN
0742 09CC-P 05                  D.CPO:=HEXBIN(INIC.TOKEN,1);
0743 09DC-P 05                  IF X > TPAL
0744 09E2-P 05                  THEN
0745 09EA-P 05                      & CONSTANTE POSSUI MAIS BITS QUE OS QUE FALTAM
0746 09EA-P 05                      BEGIN
0747 09EA-P 06                      X:=!-TPAL; & NUM DE BITS A SEREM TRUNCADOS
0748 09F8-P 06                      IF X >= MAX.CTE
0749 09F8-P 06                      THEN
0750 0A02-P 06                          ERRO(ERRO.TRUNC);
0751 0A0C-P 06                      D.CPO:=HEXBIN(INIC.TOKEN:=!+X/AUX,1);
0752 0A28-P 06                      AUX:=!-X MOD AUX;
0753 0A3A-P 06                      END;
0754 0A3A-P 05                  POS.BIT:=POS.CPO;
0755 0A42-P 05                  REPEAT
0756 0A42-P 05                      BEGIN
0757 0A42-P 06                      FOR X:=1 TO AUX
0758 0A46-P 06                      DO
0759 0A54-P 06                          IF BIT(AD.CPO,16-X)
0760 0A64-P 06                          THEN
0761 0A68-P 06                              SET(ACOD.GERADO,POS.BIT+AUX-X)
0762 0A7E-P 06                      ELSE
0763 0A80-P 06                              CLEAR(ACOD.GERADO,POS.BIT+AUX-X);
0764 0AA2-P 06                      POS.BIT:=!+X-1;
0765 0AAE-P 06                      AUX:=IF QUAL.SIMB = HEX THEN 4
0766 0AB6-P 06                                              ELSE 1;
0767 0AC0-P 06                      D.CPO:=HEXBIN(INIC.TOKEN:=!+1,1);
0768 0ADA-P 06                      END
0769 0ADA-P 05                  UNTIL POS.BIT >= TPAL;
0770 0AE8-P 05                  AT.CPO:=NCPO(MNEM.ROT)+1;
0771 0AFC-P 05                  END;
0772 0AFC-P 04              END;
```

```
0774 0B00-P 03          &
0775 0B00-P 03          & ACAO 7 : B --> ORG [7] HEX
0776 0B00-P 03          &
0777 0B00-P 03          & MUDANCA DE ENDERECO DE MICRO-MEMORIA
0778 0B00-P 03          &
0779 0B00-P 03          BEGIN
0780 0B00-P 04          EH.ORG:=VERO;
0781 0B04-P 04          WHILE TAM.TOKEN > 4
0782 0B04-P 04          DO
0783 0B0C-P 04              BEGIN
0784 0B0C-P 05              ERRO(ERRO.TRUNC.HEX);
0785 0B14-P 05              TAM.TOKEN:=!-1;
0786 0B18-P 05              INIC.TOKEN:=!+1;
0787 0B1C-P 05              END;
0788 0B1E-P 04          EMEM.ORG:=HEXBIN(INIC.TOKEN,TAM.TOKEN);
0789 0B30-P 04          & NMI.REG:=TREG.OBJ/(TPAL/8);
0790 0B30-P 04          NMI.MEM:=VAL.EMEM MOD NMI.REG;
0791 0B3C-P 04          IF VAL.EMEM <= EMEM.ORG
0792 0B3C-P 04          THEN
0793 0B46-P 04              & ORG AVANCA ENDERECO
0794 0B46-P 04              BEGIN
0795 0B46-P 05              DIF.ENDER:=EMEM.ORG-VAL.EMEM;
0796 0B52-P 05              NREG.PULA:=DIF.ENDER/NMI.REG;
0797 0B5E-P 05              NMI.PULA:=DIF.ENDER MOD NMI.REG;
0798 0B6A-P 05              IF NREG.PULA > 0
0799 0B6A-P 05              THEN
0800 0B70-P 05                  & AVANCA NREG.PULA REGISTROS NO OBJETO
0801 0B70-P 05                  BEGIN
0802 0B70-P 06                  IF (ATREG:=CURRENT(UL.OBJ)) < (TOTREG:=TOTREC(UL.OBJ))
0803 0B8E-P 06                  THEN
0804 0B94-P 06                      & REG CORRENTE MENOR QUE TOTAL DE REGS
0805 0B94-P 06                      BEGIN
0806 0B94-P 07                      RW.AT; & REESCREVE O ATUAL
0807 0B98-P 07                      IF (ID.REG:=NREG.PULA+ATREG) > TOTREG
0808 0BA4-P 07                      THEN
0809 0BAA-P 07                          & PROX REG E' MAIOR QUE TOTAL ATUAL
0810 0BAA-P 07                          ID.REG:=TOTREG;
0811 0BB2-P 07                      READANY(UL.OBJ,AREG.OBJ,ANR,ID.REG);
0812 0BC8-P 07                      NREG.PULA:=!+ATREG-ID.REG;
0813 0BD4-P 07                      END;
0814 0BD4-P 06                  WHILE NREG.PULA > 0
0815 0BD4-P 06                  DO
0816 0BDA-P 06                      & CRIA NOVOS REGS NO OBJETO
0817 0BDA-P 06                      BEGIN
0818 0BDA-P 07                      GRAVA.OU.REGRAVA(TREG.OBJ);
0819 0BE4-P 07                      NREG.PULA:=!-1;
0820 0BE8-P 07                      END;
0821 0BEA-P 06                  END;
0822 0BEA-P 05              IF (NPAL.OBJ:=(NMI.MEM+NMI.PULA)*NB.PAL) >= TREG.OBJ
0823 0BFC-P 05              THEN
```

```
0824 0C02-P 05                    & PROX ENDER ULTRAPASSA ATUAL REG
0825 0C02-P 05                    BEGIN
0826 0C02-P 06                    GR.REG.CH;
0827 0C06-P 06                    END;
0828 0C06-P 05                END
0829 0C06-P 04          ELSE
0830 0C08-P 04                & ORG RECUA ENDERECO
0831 0C08-P 04                BEGIN
0832 0C08-P 05                EMEM.REC:=IF EMEM.REC > VAL.EMEM
0833 0C08-P 05                              THEN
0834 0C12-P 05                                   EMEM.REC
0835 0C12-P 05                              ELSE
0836 0C18-P 05                                   VAL.EMEM;
0837 0C20-P 05                IF (ID.REG:=VAL.EMEM/NMI.REG+1) <> (X:=EMEM.ORG/NMI.REG+1)
0838 0C3C-P 05                THEN
0839 0C40-P 05                    & RECUA REG
0840 0C40-P 05                    BEGIN
0841 0C40-P 06                    IF NPAL.OBJ <> 0
0842 0C40-P 06                    THEN
0843 0C46-P 06                        GRAVA.OU.REGRAVA(NB.PAL*(IF (AUX:=EMEM.REC MOD NMI.
0844 0C52-P 06                                                    THEN NMI.REG
0845 0C54-P 06                                                    ELSE AUX));
0846 0C6A-P 06                    READANY(UL.OBJ,AREG.OBJ,ANB,X);
0847 0C80-P 06                    END;
0848 0C80-P 05                NPAL.OBJ:=(EMEM.ORG MOD NMI.REG)*NB.PAL;
0849 0C92-P 05                END;
0850 0C92-P 04          END;
0851 0C96-P 03 END;
0852 0CAA-P 02 INIC.ID:=VAZIO;
0853 0CAE-P 02 END ACAO.SEMANT;
0854 74C2-D 01 END;

OBJETO:RSEMANT IDP08 V=10101 J VERSAO=V.01   COD=MICROA   0069 REGISTROS
** FIM LPS **  ADVERTENCIAS:00      ERROS:00  MEMORIA: 80CB0-P + 874C2-D
```

```
LPS/500 X.06    GRAVACAO DE MODULO-OBJETO - MONTAMIC    E:FGROBJ    23/09/82 12:08 PAG:00


ARQUIVO FONTE: FGROBJ   (DP08)    VOLUME=10101     VERSAO=V.04   CODIGO=MICRDA
EQUIPAMENTO:   C500
PARAMETROS:    E:FGROBJ;S:RGROBJ;ES
OPCOES INICIAIS: LE NLC LA ES S D0 01 02 D3 D4 D5 D6 D7 PFS


0002 0000-D 00 ?LPS,S,D,"GRAVOBJ"
0003 0000-D 00 BEGIN
0004 0000-P 01 &
0005 0000-P 01 & CONSTANTES :
0006 0000-P 01 &
0007 0000-P 01 CONSTANT BRANCO = " " ;
0008 0000-D 01 CONSTANT BYTE.VAZIO = 0FF ;
0009 0000-D 01 CONSTANT DEF.E.S = 0FFFF ;
0010 0000-D 01 CONSTANT FIM.ARQ = 1 ;
0011 0000-D 01 CONSTANT NAO.CANC = 08000 ;
0012 0000-D 01 CONSTANT SEQL.OBJ = 00103 ;
0013 0000-D 01 CONSTANT TEXT.TAB = 512 ;
0014 0000-D 01 CONSTANT TEXT.MIC = 8*TEXT.TAB ;
0015 0000-D 01 CONSTANT UL.MOD.OBJ = 242 ;
0016 0000-D 01 CONSTANT VAZIO = 0FFFF ;
0017 0000-D 01 GLOBAL PROCEDURE GRAVA.MOD.OBJ;
0018 0000-D 01 &
0019 0000-D 01 & FUNCAO : TRANSFORMAR A IMAGEM DA MEMORIA EM UL.OBJ PARA MODULO-OBJETO.
0020 0000-D 01 &
0021 0000-D 01 BEGIN
0022 0000-P 02 &
0023 0000-P 02 & VARIAVEIS EXTERNAS :
0024 0000-P 02 &
0025 0000-P 02 EXTERNAL WORD EPRIM.REG;
0026 0000-D 02 EXTERNAL WORD IND;
0027 0000-D 02 EXTERNAL WORD NB;
0028 0000-D 02 EXTERNAL BYTE NB.PAL;
0029 0000-D 02 EXTERNAL WORD NMI;
0030 0000-D 02 EXTERNAL WORD NMI.PROG;
0031 0000-D 02 EXTERNAL WORD NUM.ORDEM;
0032 0000-D 02 EXTERNAL WORD(*) PIL.END.EXT.MIC;
0033 0000-D 02 EXTERNAL WORD(*) PIL.END.EXT.TAB;
0034 0000-D 02 EXTERNAL BYTE(*) REG.OBJ;
0035 0000-D 02 EXTERNAL BYTE(*) REG.FIM.GERA;
0036 0000-D 02 EXTERNAL WORD TREG.OBJ;
0037 0000-D 02 EXTERNAL WORD UL.OBJ;
0038 0000-D 02 EXTERNAL BYTE(*) VER.OBJ;
```

```
0040 0000-D 02 &
0041 0000-D 02 & VARIAVEIS LOCAIS :
0042 0000-D 02 &
0043 0000-D 02 BYTE(20) DIRETORIO = (1,2:0,"C",2:0,10:BRANCO,0,2,2:0);
0044 0014-D 02 WORD PONT.REG.LIVRE POS DIRETORIO + 4;
0045 0014-D 02 BYTE(*) NOME.MOD POS DIRETORIO + 6 ;
0046 0014-D 02 WORD FIM.NOME.BLC;
0047 0016-D 02 BYTE(*) RESTO.BLC REF FIM.NOME.BLC;
0048 0016-D 02 & USO PARA BLC NOME :
0049 0016-D 02 WORD NUM.SIMB.EXT POS RESTO.BLC; & NUMERO DE VARIAVEIS EXTERNAS
0050 0016-D 02 WORD TAM.PROG POS NUM.SIMB.EXT + 2; & TAMANHO DA AREA DE PROGRAMA
0051 0016-D 02 & USO PARA BLC GLOBAL :
0052 0016-D 02 WORD DESLOC POS RESTO.BLC;
0053 0016-D 02 BYTE C.S POS DESLOC + 2;
0054 0016-D 02 BYTE(197) FORM.BLC;
0055 00DB-D 02 WORD TAM.BLC POS FORM.BLC;
0056 00DB-D 02 BYTE TIPO.BLC POS TAM.BLC + 2;
0057 00DB-D 02 BYTE(*) NOME.BLC POS TIPO.BLC + 1;
0058 00DB-D 02 WORD END.REL POS TIPO.BLC + 1;
0059 00DB-D 02 BYTE NBB POS END.REL + 2;
0060 00DB-D 02 BYTE(*) BIT.RELOC POS NBB + 1;
0061 00DB-D 02 BYTE(577) REG.TEXTO ;
0062 031C-D 02 WORD MAIS.BII;
0063 031E-D 02 WORD TAM.TEXTO;
0064 0320-D 02 WORD NCAR.NOME;
0065 0322-D 02 WORD NMI.REL ;
0066 0324-D 02 WORD Y,AUX,X;
0067 032A-D 02 WORD ENDER.AUX1,ENDER.AUX2;
0068 032E-D 02 BYTE(*) PRIM.REG REF EPRIM.REG ;
0069 032E-D 02 BYTE(9) NMAQ POS PRIM.REG ; & NOME DA MAQUINA
0070 032E-D 02 WORD    TPAL POS NMAQ + 9 ; & TAM DA PALAVRA A SER GERADA
0071 032E-D 02 BYTE    ENC.OBJ POS TPAL + 2 ; & DEFAULT DE PREENCHIMENTO DO OBJETO
0072 032E-D 02 BYTE(*) NCPO POS ENC.OBJ + 1 ; & VETOR DE NUM CAMPOS POSSIVEIS
0073 032E-D 02 BYTE(*) NOM.OBJ POS REG.FIM.GERA + 9 ;
0074 032E-D 02 BYTE(*) CAR POS 0;
0075 032E-D 02 &
0076 032E-D 02 & PROCEDIMENTOS EXTERNOS :
0077 032E-D 02 &
0078 032E-D 02 WORD PROCEDURE CATALOG(WORD CONTR,ENOME);
0079 032E-D 02   EXTERNAL;
0080 032E-D 02 WORD PROCEDURE FILENAME(WORD CONTR,ENOME);
0081 032E-D 02   EXTERNAL;
```

```
0083 032E-D 02 PROCEDURE GRAVA(WORD ENDER,TAM);
0084 032E-D 02 BEGIN
0085 0010-P 03 IF TAM > 512
0086 0010-P 03 THEN
0087 001A-P 03       BEGIN
0088 001A-P 04       WRITE(UL.MOD.OBJ,ENDER,512);
0089 002C-P 04       TAM:=TAM-512;
0090 0038-P 04       ENDER:=ENDER+512;
0091 0044-P 04       END;
0092 0044-P 03 WRITE(UL.MOD.OBJ,ENDER,TAM);
0093 0056-P 03 END GRAVA;
0094 0332-D 02 PROCEDURE GRAVA.CABEC.TEXTO;
0095 0332-D 02 &
0096 0332-D 02 & FUNCAO : PREENCHER A PRIMEIRA PARTE DO BLOCO TEXTO.
0097 0332-D 02 &
0098 0332-D 02 BEGIN
0099 0058-P 03 CONSTANT TEXTO.DE.PRG = 031;
0100 0332-D 03 NBB:=TAM+MAIS.BIT/8+(IF MAIS.BIT MOD 8 = 0
0101 0070-P 03                       THEN
0102 0072-P 03                            0
0103 0072-P 03                       ELSE
0104 0076-P 03                            1);
0105 007C-P 03 TAM.BLC:=NBB+5+TAM.TEXTO;
0106 008C-P 03 TIPO.BLC:=TEXTO.DE.PRG;
0107 0092-P 03 GRAVA(ADFORM.BLC,NBB+6);
0108 00A4-P 03 END GRAVA.CABEC.TEXTO;
0109 0332-D 02 PROCEDURE GRAVA.DIRETORIO;
0110 0332-D 02 &
0111 0332-D 02 & FUNCAO : PREENCHER O DIRETORIO DO MODULO-OBJETO.
0112 0332-D 02 &
0113 0332-D 02 & DESCRICAO DO DIRETORIO :
0114 0332-D 02 &
0115 0332-D 02 & **CAMPOS**      * **TAM** * **VALOR** *
0116 0332-D 02 &                   (BYTE)
0117 0332-D 02 & NUM ENTR        *    1    *     1     *
0118 0332-D 02 &
0119 0332-D 02 & PONT PROX       *    2    *     0     *
0120 0332-D 02 &
0121 0332-D 02 & TIPO            *    1    *    "C"    *
0122 0332-D 02 &
0123 0332-D 02 & PONT REG LIVRE  *    2    *  NREG+1   *
0124 0332-D 02 &
0125 0332-D 02 & NOME DA ENTR    *    10   *  UL.OBJ   *
0126 0332-D 02 &
0127 0332-D 02 & NUM REG INICIO  *    2    *     2     *
0128 0332-D 02 &
0129 0332-D 02 & ATRIB           *    2    *     0     *
0130 0332-D 02 &
0131 0332-D 02 BEGIN
0132 00A6-P 03 FILENAME(UL.OBJ,ANOME.MOD);
```

```
0133 0084-P 03 GRAVA(ADIRETORIO,20);
0134 0000-P 03 END GRAVA,DIRETORIO;
```

```
0136 0332-D 02 PROCEDURE GRAVA.BLC.NOME;
0137 0332-D 02 &
0138 0332-D 02 & FUNCAO : PREENCHER O BLOCO NOME DO MODULO-OBJETO.
0139 0332-D 02 &
0140 0332-D 02 & DESCRICAO DO BLOCO NOME :
0141 0332-D 02 &
0142 0332-D 02 & **CAMPOS** * **TAM** * **VALOR**
0143 0332-D 02 &                (BYTE)
0144 0332-D 02 & TAMANHO     *     2     *   VAR+26
0145 0332-D 02 &
0146 0332-D 02 & TIPO        *     1     *     001
0147 0332-D 02 &
0148 0332-D 02 & NOME        *   VAR     *      -
0149 0332-D 02 &
0150 0332-D 02 & NUM SIMB    *     2     *   NUM.ORDEM+1
0151 0332-D 02 &
0152 0332-D 02 & TAM PROG    *     2     *   NMI.PROG*NB.PAL
0153 0332-D 02 &
0154 0332-D 02 & TAM DADOS   *     2     *     0
0155 0332-D 02 &
0156 0332-D 02 & END EXEC    *     2     *     0
0157 0332-D 02 &
0158 0332-D 02 & RESERVADO   *    16     *      -
0159 0332-D 02 &
0160 0332-D 02 & CHECK-SUM   *     1     *      -
0161 0332-D 02 &
0162 0332-D 02 BEGIN
0163 00C2-P 03 CONSTANT SUB.ROTINA = 001;
0164 0332-D 03 NCAR.NOME:=SCAN(ANOME.MOD,10,BRANCO); & TAMANHO DO IDENTIFICADOR
0165 00D4-P 03 TAM.BLC:=26+NCAR.NOME;
0166 00DE-P 03 TIPO.BLC:=SUB.ROTINA;
0167 00E2-P 03 MBT(ANOME.MOD,ANOME.BLC,NCAR.NOME);
0168 00F2-P 03 FIM.NOME.BLC:=ANOME.BLC+NCAR.NOME;
0169 00FE-P 03 NUM.SIMB.EXT:=NUM.ORDEM+(IF NUM.ORDEM==1
0170 00FE-P 03                              THEN 1
0171 0106-P 03                              ELSE 2);
0172 0114-P 03 TAM.PROG:=NMI.PROG*NB.PAL;
0173 0124-P 03 FILL(ATAM.PROG+2,21,0); & PREENCHE A PARTIR DE TAM DADOS COM 0
0174 0136-P 03 GRAVA(AFORM.BLC,TAM.BLC+2);
0175 0146-P 03 END GRAVA.BLC.NOME;
```

```
0177 0332-D 02 PROCEDURE GRAVA.BLCS.EXT;
0178 0332-D 02 &
0179 0332-D 02 & FUNCAO : PREENCHER BLOCOS DE VARIAVEIS EXTERNAS, CADA VARIAVEL POSSUI
0180 0332-D 02 &          UM BLOCO.
0181 0332-D 02 &
0182 0332-D 02 & DESCRICAO DO BLOCO :
0183 0332-D 02 &
0184 0332-D 02 & **CAMPOS** * **TAM** * **VALOR**
0185 0332-D 02 &                (BYTE)
0186 0332-D 02 & TAMANHO    *     2     *    VAR+2
0187 0332-D 02 &
0188 0332-D 02 & TIPO       *     1     *    010
0189 0332-D 02 &
0190 0332-D 02 & NOME       *    VAR    *     -
0191 0332-D 02 &
0192 0332-D 02 & CHECK-SUM  *     1     *     -
0193 0332-D 02 &
0194 0332-D 02 BEGIN
0195 0148-P 03 CONSTANT DADOS = 010;
0196 0332-D 03 FOR AUX:=0 TO NUM.ORDEM
0197 014C-P 03 DO
0198 015A-P 03     & A CADA VARIAVEL EXTERNA DECLARADA SERA' CRIADO UM BLOCO
0199 015A-P 03     BEGIN
0200 015A-P 04     IF (NCAR.NOME:=SCAN(CX:=PIL.END.EXT.TAB(AUX)),9,BYTE.VAZIO)) = 0
0201 0176-P 04     THEN
0202 017A-P 04          NCAR.NOME:=9;
0203 017E-P 04     TAM.BLC:=NCAR.NOME+2;
0204 0188-P 04     TIPO.BLC:=DADOS;
0205 018E-P 04     MBT(X,ANOME.BLC,NCAR.NOME);
0206 019E-P 04     FORM.BLC(3+NCAR.NOME):=0; & CHECK-SUM
0207 01A8-P 04     GRAVA(AFORM.BLC,TAM.BLC+2);
0208 01B8-P 04     END;
0209 01C4-P 03 END GRAVA.BLCS.EXT;
```

```
0211 0334-D 02 PROCEDURE GRAVA.BLC.GL;
0212 0334-D 02 &
0213 0334-D 02 & FUNCAO : PREENCHER O BLOCO GLOBAL PARA FUNCIONAR COMO ENTRY-POINT.
0214 0334-D 02 &
0215 0334-D 02 & DESCRICAO DO BLOCO :
0216 0334-D 02 &
0217 0334-D 02 & **CAMPOS** * **TAM** * **VALOR**
0218 0334-D 02 &                 (BYTE)
0219 0334-D 02 & TAMANHO     *     2     *  VAR+4
0220 0334-D 02 &
0221 0334-D 02 & TIPO        *     1     *  821
0222 0334-D 02 &
0223 0334-D 02 & NOME        *    VAR    *  -
0224 0334-D 02 &
0225 0334-D 02 & DESLOC      *     2     *  0
0226 0334-D 02 &
0227 0334-D 02 & CHECK-SUM   *     1     *  0
0228 0334-D 02 &
0229 0334-D 02 BEGIN
0230 01C6-P 03 CONSTANT TIPO.PROG = 821;
0231 0334-D 03 NCAR.NOME:=SCAN(ANOME.MOD,10,BRANCO);
0232 01D8-P 03 TAM.BLC:=NCAR.NOME+4;
0233 01E2-P 03 TIPO.BLC:=TIPO.PROG;
0234 01E8-P 03 MBTCANOME.MOD,ANOME.BLC,NCAR.NOME);
0235 01F8-P 03 FIM.NOME.BLC:=ANOME.BLC+NCAR.NOME;
0236 0204-P 03 DESLOC:=0;
0237 0208-P 03 C.S:=0;
0238 0212-P 03 GRAVACAFORM.BLC,TAM.BLC+2);
0239 0222-P 03 END GRAVA.BLC.GL;
```

```
0241 0334-D 02 PROCEDURE GRAVA.BLCS.TEXTO;
0242 0334-D 02 &
0243 0334-D 02 & FUNCAO : PREENCHER BLOCOS DE TEXTO. CADA REG SERA' DIVIDIDO EM DOIS BL
0244 0334-D 02 &          DE TEXTO. CADA BLOCO DE TEXTO SERA' GRAVADO EM DOIS REGISTROS
0245 0334-D 02 &          MODULO-OBJETO.
0246 0334-D 02 &
0247 0334-D 02 & DESCRICAO DO BLOCO TEXTO :
0248 0334-D 02 &
0249 0334-D 02 & **CAMPOS**        * **TAM**      * **VALOR**
0250 0334-D 02 &                     (BYTE)
0251 0334-D 02 & TAMANHO           *    2         *   NBB+5+TREG.OBJ/2
0252 0334-D 02 &
0253 0334-D 02 & TIPO              *    1         *   031
0254 0334-D 02 &
0255 0334-D 02 & ENDER REL         *    2         *   MULTIPLO DE NB/2
0256 0334-D 02 &
0257 0334-D 02 & NBYTES.DE.BITS    *    1         *   -
0258 0334-D 02 &
0259 0334-D 02 & BITS DE RELOC     *   NBB        * C0,SE ABS. C110,SE RELOC EXTERNA
0260 0334-D 02 &
0261 0334-D 02 & TEXTO             *   TREG.OBJ/2 *   -
0262 0334-D 02 &
0263 0334-D 02 & CHECK-SUM         *    1         *   -
0264 0334-D 02 &
0265 0334-D 02 BEGIN
0266 0224-P 03 END.REL:=0;
0267 0228-P 03 NMI.REL:=0;
0268 022C-P 03 IND:=0;
0269 0230-P 03 REWIND(UL.OBJ);
0270 0234-P 03 WHILE READ(UL.OBJ OR NAO.CANC,AREG.OBJ,ANB) <> FIM.ARQ
0271 0250-P 03 DO
0272 0256-P 03    & TODOS OS REGS SERAO DIVIDIDOS AO MEIO E REGRAVADOS COMO TEXTO
0273 0256-P 03    FOR AUX:=0 TO 1
0274 025A-P 03    DO
0275 0266-P 03       BEGIN
0276 0266-P 04       TAM.TEXTO:=NB/2;
0277 0272-P 04       & CASO NAO HAJA VAR EXTERNA NESSE SEGMENTO DE TEXTO:
0278 0272-P 04       NBB:=TAM.TEXTO/NB.PAL + (IF TAM.TEXTO MOD NB.PAL = 0
0279 0286-P 04                                   THEN 0
0280 0288-P 04                                   ELSE 1);
0281 0294-P 04       FILL(XBIT.RELOC,191,0); & PREENCHE BITS DE RELOC
0282 02A4-P 04       MBT(ENDER.AUX1:=AREG.OBJ+AUX*TAM.TEXTO,AREG.TEXTO,TAM.TEXTO);
0283 02C6-P 04       NMI.REL:=!+(NB/NB.PAL)/2;
0284 02DC-P 04       MAIS.BIT:=0;
0285 02E0-P 04       WHILE NUM.ORDEM <> -1 & NAO EXISTE VAR EXTERNA
0286 02E0-P 04             AND
0287 02E6-P 04             (NMI:=PIL.END.EXT.MIC(IND))< NMI.REL & NAO EXISTE NESSE SEGM
0288 02F6-P 04       DO
0289 0300-P 04          & ABRE ESPACO PARA BYTE COM NUMERO DE ORDEM
0290 0300-P 04          & E
```

```
0291 0300-P 04              & MODIFICA BIT DE RELOC DE ABSOLUTO PARA RELOC EXTERNA
0292 0300-P 04              BEGIN
0293 0300-P 05              IF (X:=(((NMT+1)*NB.PAL) MOD TREG.OBJ) MOD (NB/2)) = 0 & DESL N
0294 031E-P 05              THEN
0295 0320-P 05                   X:=NB/2;
0296 032C-P 05              MBT(ENDER.AUX1+X,(ENDER.AUX2:=AREG.TEXTO+X+MAIS.BIT)+1,TAM.TEXT
0297 0358-P 05              TAM.TEXTO:=I+1;
0298 035C-P 05              CAR(ENDER.AUX2):=CAR(ENDER.AUX2-1);
0299 0374-P 05              CAR(ENDER.AUX2-1):=0;
0300 037E-P 05              SET(ABIT.RELOC,(X:=I+MAIS.BIT)-2);
0301 0396-P 05              SET(ABIT.RELOC,X-1);
0302 03A6-P 05              MAIS.BIT:=I+1;
0303 03AA-P 05              IND    :=I+1;
0304 03AE-P 05              END;
0305 03B0-P 04         GRAVA.CABEC.TEXTO;
0306 03B4-P 04         END.REL:=I+NB/2;
0307 03C4-P 04         GRAVA(AREG.TEXTO,TAM.TEXTO);
0308 03D2-P 04         GRAVACA(600),1); & CHECK-SUM
0309 03DE-P 04         END;
0310 03F0-P 03 END GRAVA.BLCS.TEXTO;
0311 0337-D 02 PROCEDURE GRAVA.BLC.FIM;
0312 0337-D 02 &
0313 0337-D 02 & FUNCAO : PREENCHER BLOCO FIM E CORRIGIR INFO DE DIRETORIO
0314 0337-D 02 &
0315 0337-D 02 & DESCRICAO DO BLOCO FIM :
0316 0337-D 02 &
0317 0337-D 02 & **CAMPOS**   * **TAM** * **VALOR**
0318 0337-D 02 &                  (BYTE)
0319 0337-D 02 & TAMANHO      *     2     *      2
0320 0337-D 02 &
0321 0337-D 02 & TIPO         *     1     *     ÕF0
0322 0337-D 02 &
0323 0337-D 02 & CHECK-SUM    *     1     *      -
0324 0337-D 02 &
0325 0337-D 02 BEGIN
0326 03F2-P 03 TAM.BLC:=2;
0327 03F6-P 03 TIPO.BLC:=ÕF0;
0328 03FC-P 03 FORM.BLC(3):=0; & CHECK-SUM
0329 0400-P 03 GRAVA(AFORM.BLC,4);
0330 040C-P 03 NB:=20;
0331 0412-P 03 READANY(UL.MOD.OBJ,ADIRETORIO,ANB,1);
0332 0426-P 03 PONT.REG.LIVRE:=TOTREC(UL.MOD.OBJ)+1;
0333 0436-P 03 REWRITE(UL.MOD.OBJ,ADIRETORIO,20);
0334 0446-P 03 END GRAVA.BLC.FIM;
```

```
LPS/500 X.06    GRAVACAO DE MODULO-OBJETO - MONTAMTC    E:FGROBJ    23/09/82 12:09 PAG:01

0336 0337-D 02 &
0337 0337-D 02 & INICIO DOS COMANDOS :
0338 0337-D 02 &
0339 0337-D 02 CREATE(UL.MOD.OBJ,DEF.E.S,SEUL.OBJ);
0340 045A-P 02 GRAVA.DIRETORIO;
0341 045F-P 02 GRAVA.BLC.NOME;
0342 0462-P 02 GRAVA.BLCS.EXT;
0343 0466-P 02 IF NUM.ORDEM <> -1
0344 0466-P 02 THEN
0345 046F-P 02       & EXISTE VARIAVEL EXTERNA
0346 046F-P 02       GRAVA.BLC.GL;
0347 0472-P 02 GRAVA.BLCS.TEXTO;
0348 0476-P 02 GRAVA.BLC.FIM;
0349 047A-P 02 PURGE(UL.OBJ);
0350 0484-P 02 CATALOG(UL.MOD.OBJ,ÃNOM.OBJ);
0351 0492-P 02 UL.OBJ:=UL.MOD.OBJ;
0352 0498-P 02 END GRAVA.MOD.OBJ;
0353 0337-D 01 END;

OBJETO:RGROBJ   IDP08 V=10101 1 VERSAO=V.04    COD=MICROA   0005 REGISTROS
** FIM LPS **   ADVERTENCIAS:00      ERROS:00   MEMORIA: 8049A-P + 80337-D
```

```
LIGADOR C500  V:V.24

EQUIPAMENTO : C500

PARAMETROS E/OU DIRETIVAS :
F:5,7,1,2,3,4,9,10;S:8;ES;LN;LM;M:MICROA;P:.PARMS,80;A:2600;V:V00;

BIBLIOTECAS DE MODULOS OBJETO :
 005 RRSINT    [DP08]  VOLUME=10101  CO=MICROA  VERSAO=V.02
 007 RAIRAR    [DP08]  VOLUME=10101  CO=MICROA
 001 RLEX      [DP08]  VOLUME=10101  CO=MICROA  VERSAO=V.00
 002 RPARMS    [DP08]  VOLUME=10101  CO=MICROA  VERSAO=V.01
 003 RSEMANT   [DP08]  VOLUME=10101  CO=MICROA  VERSAO=V.01
 004 RGROBJ    [DP08]  VOLUME=10101  CO=MICROA  VERSAO=V.04
 009 RLPS      [DP00]  VOLUME=00000  CO=SISTEM  VERSAO=V.01
 010 ENTSAI    [DP00]  VOLUME=00000  CO=SISTEM  VERSAO=V.07

OPCOES EM CURSO :  LM;  LN; NLX;  LA;  ES; NOM; NRE; NCR;

ARQUIVO DE SAIDA :
 008 MONTAMIC  [DP08]  VOLUME=10101  CO=MICROA  VERSAO=V.00
```

LIGADOR C500 V:V.24

LISTA DE REFERENCIAS CRUZADAS

.ABRE	GLBP	0000	1084	.ABRE
.ACESÇALC6	GLBP	0000	0B1E	.PREPARAES
.ACESÇALCW	GLBP	0000	0B36	.PREPARAES
.AREAÇCORR	GLBD	0000	0A06	.AREANALC
.AREAMENS	GLBD	0000	09C8	.PREPARAES
.AREANALC	GLBD	0000	0A16	.AREANALC
.AREARESP	GLBD	0000	09F2	.PREPARAES
.ECOT'VSAI	GLBP	0000	0C94	.PREPARAES
.ENDERNALC	GLBP	0000	0B5A	.PREPARAES
.ENDNALCAI	GLBD	0000	0B18	.AREANALC
.ENVIXAGEM	GLBP	0000	0BEE	.PREPARAES
.ERROGRAVE	GLBP	0000	0C84	.PREPARAES
.FECHA	GLBP	0000	0A60	CLOSE
.FILEID	GLBP	0000	19F2	.FILEID
.LF	GLBP	0000	0DEE	.LE
.LIBERANL	GLBP	0000	0AE8	.LIBERANL
.MOVEPARM	GLBP	0000	0B4E	.PREPARAES
.NIVELPREP	GLBD	0000	0B17	.AREANALC
.PARMS	GLBD	0000	849C	ANPARMS
.PREPARAES	GLBP	0000	0CFA	.PREPARAES
.PTARÖPRIM	GLBD	0000	0B16	.AREANALC
.SAIDA	GLBP	0000	0F54	.SAIDA
.SAIMSG	GLBP	0000	1204	.SAIMSG
ACAO.ÖMANT	GLBP	0000	20AE	SEMANT
ALLOCATE	GLBP	0000	1C38	ALLOCATE
ANALIXARMS	GLBP	0000	2DF6	ANPARMS
AT.CPO	GLBD	0000	47DA	SEMANT
ATRAB	GLBP	0000	1D2C	ATRAB
AVANCA.PT	GLBP	0000	1D9C	ANLEX
BINDEC	GLBP	0000	1A84	BINDEC
BINHEX	GLBP	0000	09BC	BINHEX
BIT	GLBP	0000	2DAE	BIT
CARECÖARMS	GLBD	0000	0000	MICROA
CAD.IXAROS	GLBD	0000	84EC	ANPARMS
CATALOG	GLBP	0000	1910	CATALOG
CLEAR	GLBP	0000	2D94	CLEAR
CLOSE	GLBP	0000	0A5A	CLOSE
COME.CNCOS	GLBP	0000	1DAA	ANLEX
CONTAC.PAG	GLBD	0000	005D	MICROA
CONTA.CAR	GLBD	0000	0F1A	ANLEX
CREATE	GLBP	0000	174E	CREATE
CREATEDISK	GLBP	0000	1AEA	CREATEDISK
CURRENT	GLBP	0000	2DC8	CURRENT
D.CPO	GLBD	0000	47DB	SEMANT
DIGITO	GLBP	0000	1D76	ANLEX
DISPLAY	GLBP	0000	11EA	DISPLAY
EADIC	GLBD	0000	0F18	ATRAB
EBUF.CPO	GLBD	0000	005F	MICROA
EBUF.LIT	GLBD	0000	47DD	SEMANT
ECOD.ORADO	GLBD	0000	47DF	SEMANT
EMEM.REC	GLBD	0000	47E1	SEMANT
EMPTY	GLBP	0000	1BFC	EMPTY
END.CÇAROS	GLBD	0000	852B	ANPARMS
ERRIM.REG	GLBD	0000	0061	MICROA
ERRO	GLBP	0000	0036	MICROA
ETAB.SIMP	GLBD	0000	47E3	SEMANT

FILEFCB	GLBP	0000	19E4	FILEFCB
FILENAME	GLBP	0000	1ADA	FILENAME
FILEUNIT	GLBP	0000	1ACA	FILEUNIT
FILEVOL	GLBP	0000	1A4A	FILEVOL
FILL	GLBP	0000	12A2	FILL
FLAG	GLBD	0000	09BE	FLAG
FORWARD	GLBP	0000	0AC8	FORWARD
FREE	GLBP	0000	1998	FREE
GERA.CODIGO	GLBP	0000	21F8	SEMANT
GRAVA%.OBJ	GLBP	0000	12B4	GRAVOBJ
GRAVA(RAVA	GLBP	0000	20EC	SEMANT
HEXBIN	GLBP	0000	2D5E	HEXBIN
IMPR.CABEC	GLBP	0000	0144	MICROA
IND	GLBD	0000	47E5	SEMANT
IND.PROT	GLBD	0000	47E7	SEMANT
INIC.ID	GLBD	0000	0F1C	ANLEX
INIC.TOKEN	GLBD	0000	0F1E	ANLEX
INICIO	GLBD	0000	0063	MICROA
LETRA	GLBP	0000	1D52	ANLEX
N.ENTR	GLBD	0000	47E9	SEMANT
NB	GLBD	0000	47EB	SEMANT
NB.LIDOS	GLBD	0000	0F20	ANLEX
NB.PAL	GLBD	0000	0064	MICROA
NMT	GLBD	0000	47ED	SEMANT
NMT.PROG	GLBD	0000	0065	MICROA
NMT.REG	GLBD	0000	0067	MICROA
NPAL.OBJ	GLBD	0000	47EF	SEMANT
NUM.LINHA	GLBD	0000	0069	MICROA
NUM.ORDEM	GLBD	0000	47F1	SEMANT
OPEN	GLBP	0000	1068	OPEN
OPENINPUT	GLBP	0000	11C4	OPENINPUT
OPENUNIT	GLBP	0000	1286	OPENUNIT
PEGA.'OKEN	GLBP	0000	1D4E	ANLEX
PIL.E'.MIC	GLBD	0000	4BF3	SEMANT
PIL.E'.TAB	GLBD	0000	47F3	SEMANT
PIL.ROT	GLBD	0000	6BF3	SEMANT
POP.TOPO	GLBP	0000	0000	MICROA
POS.CC.ARG	GLBD	0000	852D	ANPARMS
PT.REGLIDO	GLBD	0000	0F22	ANLEX
PURGE	GLBP	0000	0A0E	PURGE
PUSH.%.DIR	GLBP	0000	0178	MICROA
QUAL.SIMB	GLBD	0000	0F24	ANLEX
READ	GLBP	0000	1186	READ
READANY	GLBP	0000	0DE2	READANY
REG.FCGERA	GLBD	0000	006B	MICROA
REG.OBJ	GLBD	0000	00B5	MICROA
REG.RELAT	GLBD	0000	04B5	MICROA
REWIND	GLBP	0000	1780	REWIND
REWRITE	GLBP	0000	0F46	REWRITE
SCAN	GLBP	0000	1058	SCAN
SET	GLBP	0000	1766	SET
SETIAOFILE	GLBP	0000	1C3C	ALLOCATE
TADIC	GLBD	0000	0F16	ATRAB
TAM.ID	GLBD	0000	0F25	ANLEX
TAM.TOKEN	GLBD	0000	0F26	ANLEX
TEM.AO.ROT	GLBD	0000	8535	ANPARMS
TOTREC	GLBP	0000	195F	TOTREC
TREG.OBJ	GLBD	0000	0513	MICROA

LIGADOR-C500 V:V.24

TSEQ.SIMB	GLBD	0000	0FDA	SEMANT
UL.FONTE	GLBD	0000	0515	MICROA
UL.NPÇALOC	GLBD	0000	8536	ANPARMS
UL.OBJ	GLBD	0000	0517	MICROA
UL.PRÖALOC	GLBD	0000	853C	ANPARMS
VAL.DEC	GLBD	0000	0F27	ANLEX
VAL.FMEM	GLBD	0000	7BF3	SEMANT
VER.OBJ	GLBD	0000	0519	MICROA
VERSION	GLBP	0000	186E	VERSION
VERSIONED	GLBP	0000	17A0	VERSIONED
WRITE	GLBP	0000	1294	WRITE

```
LIGADOR C500  V:V.24

MAPA DE MEMORIA

RAIZ

MICROA          0000   0000
BINHEX          09BC   09BF
FLAG            0A0E   09BE
PURGE           0A0E   09C0
CLOSE           0A5A   09C0
FORWARD         0AC8   09C8
.LIBERANL       0AE8   09C8
.PREPARAES      0B1E   09C8
.AREANALC       0DE2   0A06
READANY         0DE2   0B1A
.LE             0DEE   0B1A
REWRITE         0E46   0B32
.SAIDA          0F54   0B32
SCAN            1058   0B46
OPEN            1068   0B46
.ABRE           1084   0B46
READ            11B6   0B62
OPENINPUT       11C4   0B62
DISPLAY         11EA   0B62
.SAIMSG         1204   0B64
OPENUNIT        1286   0B76
WRITE           1294   0B76
FILL            12A2   0B76
GRAVOBJ         12B4   0B76
CREATE          174F   0EAE
SET             1766   0EAE
REWIND          1780   0EAE
VERSIONED       17A0   0EAE
VERSION         186E   0EC0
CATALOG         1910   0ED2
TOTREC          195E   0ED2
FREE            1998   0ED2
FILEFCB         19E4   0ED2
.FILEID         19F2   0ED2
FILEVOL         1A4A   0ED2
BINDEC          1A84   0ED2
FILEUNIT        1ACA   0ED2
FILENAME        1ADA   0ED2
CREATEDISK      1AEA   0ED2
EMPTY           1BEC   0EF4
ALLOCATE        1C38   0EF4
ATRAB           1D2C   0F16
ANLEX           1D4F   0F1A
SEMANT          20AF   0FDA
HEXBIN          2D5E   849C
CLEAR           2D94   849C
BIT             2DAE   849C
CURRENT         2DC8   849C
ANPARMS         2DF6   849C

END FINAL DA SECAO  :  PROG=310A DADOS=857E

AREA ADICIONAL  :  0A28
```

LIGADOR C500 V:V.24

*** CONDICAO DE TERMINO : 000 ***