UM SISTEMA DE PROGRAMAÇÃO E DEPURAÇÃO CONVERSACIONAL PARA LINGUAGEM TIPO MONTADOR

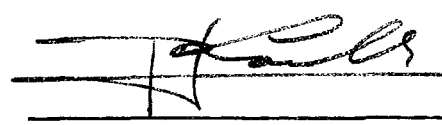
PARTE II

SUPERVISOR E INTERPRETADOR

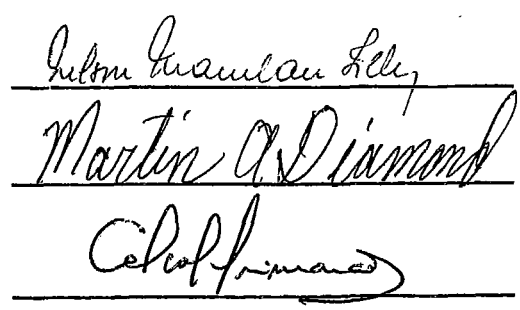
Jaime Frederico Gutbrod Caruso

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 CIÊNCIA (M.Sc.)

Aprovada por:



Presidente



RIO DE JANEIRO
ESTADO DA GUANABARA - BRASIL
AGOSTO DE 1974

A minha esposa,

por seu amor e compreensão.

AGRADECIMENTOS

Quero expressar a minha gratidão a todos aqueles que, de algum modo, me auxiliaram na elaboração deste trabalho.

Agradeço à CAPES, BNDE e à COPPE, pelo apoio financeiro dado durante a época em que obtive meus créditos.

Agradeço, especialmente, ao meu amigo Otacílio José Carollo de Souza pelo incentivo dado para que eu me empenhasse em mais rapidamente apresentar meu trabalho; ao meu orientador, Prof. Pierre Jean Lavelle; Professores, Funcionários e Colegas da COPPE; aos meus pais e aos meus sogros.

Agradeço às Srtas. Beatriz Moojen e Cristina Raymundo, pela dedicação com que datilografaram este trabalho.

RESUMO

Um Sistema de Programação e Depuração Conversacional para Linguagem Tipo Montador, foi projetado para o minicomputador MITRA 15, sem memória auxiliar. Ele é composto de um Supervisor que controla o sistema, um Analisador e um Interpretador.

A Parte I: Analisador e Montador é a parte que se preocupa com análise sintática da linguagem e com a montagem do programa para a execução pelo Interpretador.

A estrutura interna desenvolvida para o programa é tal que permite a decompilação do programa antes de sua entrada na fase de interpretação. A linguagem conversacional interpretada é um subconjunto da linguagem assembler do MITRA 15.

A Parte II divide-se em Interpretador e Supervisor.

O Interpretador trabalha sobre a estrutura montada pelo Analisador, executando o programa.

O Supervisor é quem controla o sistema e por sua estrutura possibilita o partilhamento do tempo entre os programas.

ABSTRACT

A Conversational Programming and Debugging System for Assembler-Like Language is designed for the MITRA 15 mini-computer without auxiliary memory. It is made up of an Analyser and Assembler, and a Supervisor and an Interpreter.

The first part, the Analyser and Assembler, does the syntactical analysis and assembly of programs to be executed by the Interpreter. The internal structure of programs is such that a de-compilation can be performed before starting the interpretation. The interpreted conversational language is a sub-set of the MITRA 15 assembly language.

The second part is divided into the Interpreter and Supervisor.

The Interpreter works on the analyser-built structure, executing the program.

The Supervisor controls the system allocating the Analyser, Assembler and Interpreter creating a time-sharing environment.

ÍNDICE

C A P Í T U L O I

INTRODUÇÃO

Complementando o projeto do Sistema definido na Tese "Um Sistema de Programação e Depuração Conversacional para Linguagem do Tipo Montador Parte I Analisador e Montador", a Parte II apresenta o Interpretador e o Supervisor do mesmo.

Observe-se que em primeiro lugar apresenta-se o Interpretador, uma vez que procurou-se apresentar primeiro as partes do Sistema não diretamente envolvidas com o TIME-SHARING (Analisador e Interpretador), e por fim o Supervisor.

Devido a ordem de apresentação escolhida, vários nomes referenciados no Interpretador não estão definidos. As definições destes nomes encontram-se na seção "Descrição do Contexto".

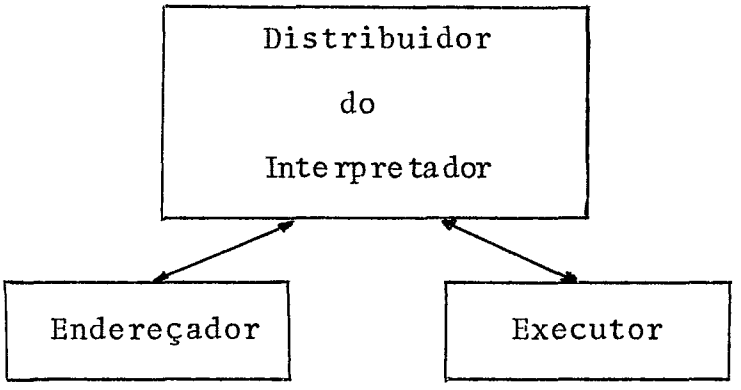
C A P Í T U L O II

INTERPRETADOR

O Interpretador é a parte do Sistema responsável pela execução dos programas corretamente montados. A execução de uma instrução, pelo Interpretador, é decomposta em três fases distintas:

- Fetch;
- Endereçamento;
- Execução propriamente dita.

O Interpretador é composto de três seções correspondentes a cada uma das fases acima mencionadas, como vemos na figura.



DISTRIBUIDOR

O Distribuidor é responsável pelo controle da execução do Interpretador. Através do contador de programa (#CTPROG) obtém o conteúdo da próxima instrução a executar; armazena-o na dupla palavra # INST, e chama o Endereçador.

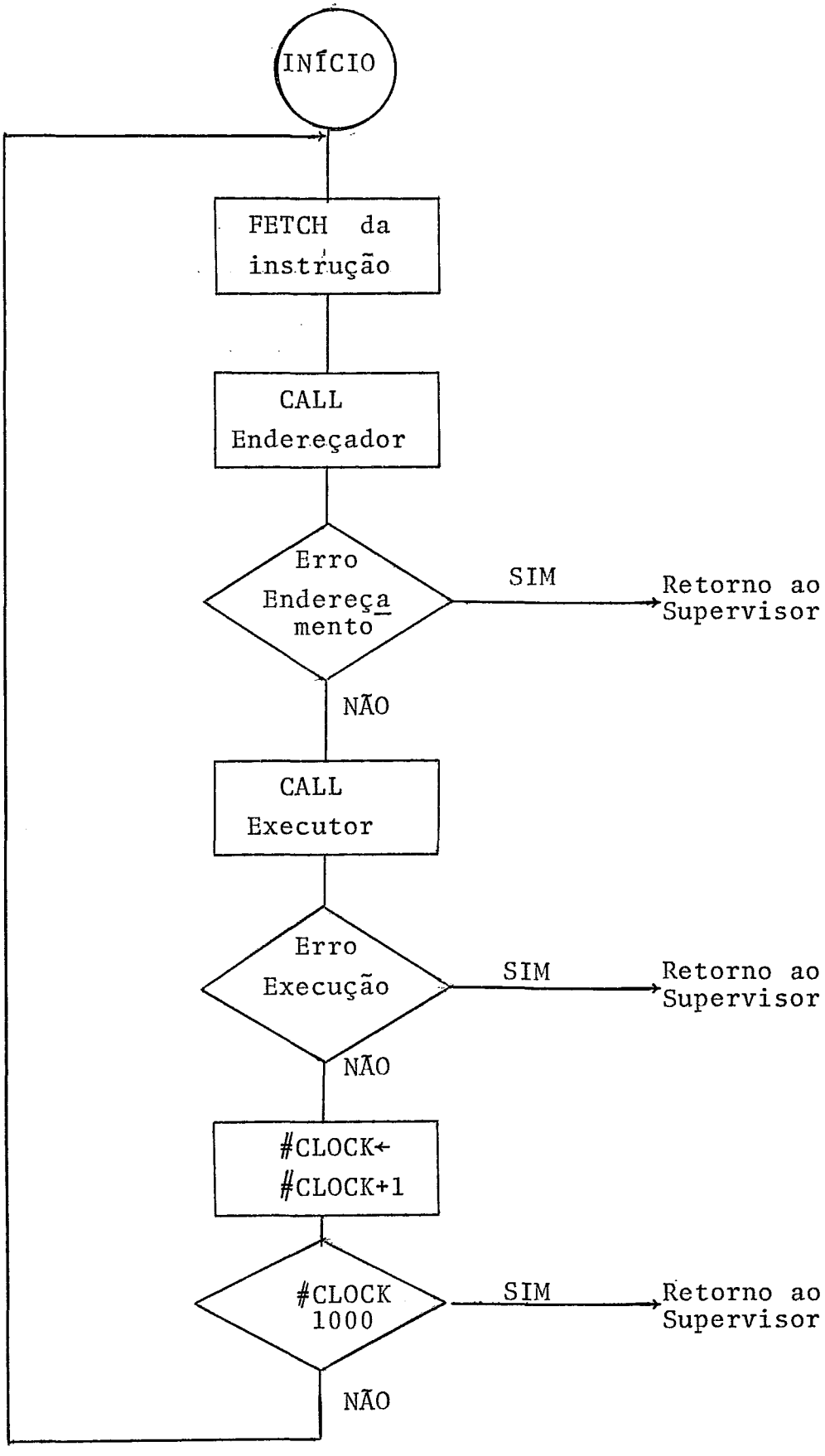
O Endereçador retorna ao Distribuidor caracterizando, no Indexador, se o endereço obtido situa-se dentro dos limites corretos ou não, caso em que a execução do processo é interrompida com retorno ao Supervisor e # STATUS igual a &F000, caso contrário a execução prossegue normalmente com a chamada do Executor. O Executor retorna ao Distribuidor, caracterizando no Indexador se a paridade do endereço do operando está correta ou não, caso em que a execução do programa é interrompida com retorno ao Supervisor e # STATUS igual a &F000.

Após a execução normal de uma instrução, o conteúdo da palavra # CLOCK é incrementado de um. Esta palavra é utilizada como controlador do tempo concedido para a execução do programa e seu valor máximo é 1000 (mil). Atingido este valor, a execução do programa é suspensa.

O Distribuidor devolve o controle ao Supervisor com o # STATUS modificado para IDLE e o usuário é forçado a enviar um comando !ØGO ou !ØBYE (ver descrição dos Comandos do Sistema). Particularmente a instrução CSV M:WAIT, ao ser executada, força o conteúdo do # CLOCK para 999 de modo a possibilitar o retorno ao Supervisor.

Após completar a execução normal de uma instrução e sempre que o conteúdo da palavra # CLOCK for menor que 1000, o Distribuidor vai fazer o FETCH de uma nova instrução.

Na página ao lado está apresentado o MACRO-FLUXO do Distribuidor.


INÍCIO
FETCH da instrução
CALL Endereçador
Erro Endereçamento
SIM
Retorno ao Supervisor
NÃO
CALL Executor
Erro Execução
SIM
Retorno ao Supervisor
NÃO
#CLOCK← #CLOCK+1
#CLOCK 1000
SIM
Retorno ao Supervisor
NÃO

ENDEREÇADOR.

O Endereçador determina o número de operandos(*) da instrução, obtém e verifica a validade do endereço de cada operando.

Neste parágrafo são considerados quatro tipos básicos de instrução:

- Instruções normais - São as instruções de um operando, cujo endereço é caracterizado pelo tipo de endereço e argumento.

- Instruções de endereço em registrador - É o subconjunto das instruções de CSV simuladas, menos CSV M:KEY e CSV M:EXIT. O endereço absoluto do operando está em #ACUMUL.

- Instruções de Manipulação do STRING - São as instruções de três operandos. O endereço do primeiro operando é caracterizado pelo tipo de endereço e argumento. O endereço absoluto do segundo operando está em #ACUMUL. O endereço absoluto do terceiro operando é obtido pela expressão:

Conteúdo de #ACUMUL+Conteúdo de #EXTAC - 1.

- Instruções de zero endereço - Explicadas mais a frente.

A obtenção do endereço absoluto de um operando caracterizado por tipo de endereço e argumento é função do tipo de endereço, confor-

---

(*) Considera-se operando, o dado ou o rótulo referenciado pela instrução que está sendo interpretada, ou o conteúdo de um registrador.

me descrição abaixo:

— Endereçamento tipo Direto Geral/Direto Local —

Esses tipos de endereçamentos referem-se somente à instruções que operam sobre dados.

O dado se encontra na zona diretamente acessível da CDS ou da LDS e, a obtenção do seu endereço se processa da seguinte maneira:

1. Endereço do nome de dados - É obtido através da expressão:

#APTCDS + ARGUMENTO x 8

ou

#APTLDS + ARGUMENTO x 8

2. Endereço do dado propriamente dito - É obtido através da expressão :

#APDCDS + DESLOCAMENTO DO NOME DE DADOS COMUM

ou

#APDLDS + DESLOCAMENTO DO NOME DE DADOS LOCAL

— Endereçamento tipo Indireto Local -

O endereçamento tipo Indireto Local se refere somente à instruções que operam sobre dados.

A obtenção do endereço se processa, em parte, de forma idêntica à exposta acima, para o endereçamento tipo Direto Local, sendo que o passo 2 deve ter o seu significado mudado, ou seja:

2. Endereço do Endereço do Dado - É obtido através da

expressão:

#APDLDS + DESLOCAMENTO DO NOME DE DADOS LOCAL

O endereço do dado é obtido através de mais um passo.

3. Endereço do Dado - É obtido do conteúdo do endereço determinado em 2.

O endereço obtido em 3 é verificado contra os limites das áreas utilizadas da CDS e de cada LDS.

Se o endereço obtido não se situa entre os limites da área utilizada de uma das seções do programa, o controle é devolvido ao Distribuidor com indicação de erro no Indexador.

– Endereçamento tipo Indireto Geral Indexado/Indireto Local Indexado.–

Estes tipos de endereçamento se referem somente à instruções que operam sobre dados.

A obtenção do endereço se processa, em parte, de forma idêntica à exposta acima, para endereçamento tipo Direto Geral/Direto Local, sendo que o passo 2 deve ter o seu significado mudado, ou seja:

2. Endereço do endereço do início da área de dados - É obtido através da expressão:

#APDCDS + DESLOCAMENTO DO NOME DE DADOS COMUM

ou

#APDLDS + DESLOCAMENTO DO NOME DE DADOS LOCAL

O endereço do dado é obtido através de mais dois pas

sos.

3. Endereço do início da Área de Dados - é obtido do conteúdo do endereço determinado em 2.

4. Endereço do Dado - é obtido através da expressão:

ENDEREÇO INICIAL DA ÁREA DE DADOS + #INDEX

As mesmas verificações, quanto à validade do endereço, feitas para endereçamento tipo Indireto Local , devem ser procedidas aqui.

— Endereçamento tipo LABEL —

O endereçamento tipo LABEL se refere somente à instruções de ramificação para uma outra instrução dentro da mesma LPS. Este tipo de endereçamento é uma adaptação dos tipos RELATIF PLUS e RELATIF MOINS do MITRA 15 (ver MITRA 15 - Manuel de Présentation).

O argumento fornece o número do rótulo da instrução para a qual deve ser feita a ramificação. Este rótulo é pesquisado entre os limites da LPS atual e quando encontrado, fornece o endereço da ramificação.

Se a instrução simulada é um BRX, após ter sido encontrado o rótulo, somamos ou subtraimos do endereço da ramificação o valor do produto 4 x conteúdo do #INDEX, dependendo se o endereço da ramificação é maior ou menor que #CTPROG.

O endereço assim calculado, endereço da ramificação para a instrução de BRX, é verificado contra os limites da área

utilizada da LPS atual, e se estiver fora deles, o controle é devolvido ao Distribuidor com indicação de erro no Indexador.

– Endereçamento tipo Indireto Geral/Indireto Local LABEL –

Estes tipos de endereçamentos referem-se somente à instruções de ramificação.

O endereçamento tipo Indireto Local LABEL é uma adaptação do endereçamento tipo Indireto Local nas instruções de ramificação.

A obtenção do endereço se processa, em parte, de forma idêntica à exposta acima para o endereçamento tipo Direto Geral/Direto Local, sendo que o passo 2 deve ter o seu significado mudado, ou seja:

2. Endereço do Endereço da Instrução - É obtido através da expressão:

#APDCDS + DESLOCAMENTO DO NOME DE DADOS COMUM

ou

#APDLDS + DESLOCAMENTO DO NOME DE DADOS LOCAL

O endereço da instrução é obtido através de mais um passo:

3. Endereço da Instrução - É obtido do conteúdo do endereço determinado em 2.

Se a instrução simulada é um BRX, o passo 2 é ligeiramente modificado, ou seja:

2. Endereço do endereço da Instrução - É obtido através da expressão:

#APDCDS + DESLOCAMENTO DO NOME DE DADOS COMUM + #INDEX

ou

#APDLDS + DESLOCAMENTO DO NOME DE DADOS LOCAL + #INDEX

O endereço obtido em 3 é verificado contra os limi - tes das áreas utilizadas de cada LPS.

Se o endereço obtido não se situa entre os limites da área utilizada de uma das seções do programa, o controle é devolvido ao Distribuidor, com indicação de erro no Indexador.

— Endereçamento tipo Seção —

O endereçamento tipo Seção se refere somente à instrução CLS. O argumento fornece o número da seção dentro da LPS atual. Este número é utilizado na pesquisa da posição correspon - dente a esta seção na Tabela de Nomes da LDS atual. Nesta posição está armazenado o endereço da entrada correspondente a esta seção, na PRT do programa, e este é o endereço do operando.

— Endereçamento tipo Paramétrico e Paramétrico Indexado —

No endereçamento tipo Paramétrico, o operando é o próprio argumento, enquanto no endereçamento tipo Paramétrico Indexado, o operando é obtido somando-se o argumento e o conteúdo de #INDEX. Em ambos os casos, o operando é armazenado na palavra de endereço 8 (PRT) do programa, e este passa a ser o endereço dado.

O endereço obtido por um dos métodos acima descritos é armazenado em #APPESQ, e o controle é devolvido ao Distribui - dor com indicação de normalidade no Indexador.

Os endereços absolutos de operandos armazenados em

#ACUMUL ou obtidos através de #ACUMUL e #EXTAC (instruções de endereço em registrador e instruções de manipulaão de STRING) são verificados contra os limites das áreas utilizadas da CDS e de cada LDS.

Se o endereço não se situa entre os limites de uma das seções do programa, o controle é devolvido ao Distribuidor , com indicação de erro no Indexador.

As instruções de zero endereço são consideradas as operações entre registradores (ver MITRA 15 - Manuel de Présentation) e as instruções CSV M:KEY e CSV M:EXIT. Em ambos os casos, nenhuma ação é tomada no sentido da obtenção do endereço. A passagem deste tipo de instruções, pelo Endereçador, visa manter um fluxo único e simples para a execução.

EXECUTOR.

O Executor é responsável pela execução propriamente dita das instruções.

Algumas funções básicas são realizadas pelo Executor antes de executar uma instrução:

- O contador de programa (#CTPROG) é incrementado e passa a apontar a próxima instrução.

  Este valor, evidentemente, poderá ser alterado pela execução de alguma das instruções de ramificação.

- Verifica se o endereço do operando (conteúdo de #APPESQ) é válido para a instrução, ou seja, se a ins

rução aceita endereço de BYTE ou endereço de palavra (ver MITRA 15 - Manuel de Présentation). A verificação é feita através da Tabela de Apontadores para a Tabela de Classes de Endereços. Os quatro BITS da esquerda do elemento da tabela correspondente à instrução, descrevem se esta pode aceitar endereço de BYTE (valor dos quatro BITS é zero) ou endereço de palavra (valor dos quatro BITS é um). Se o módulo do endereço não for compatível com a instrução, o controle é devolvido ao Distribuidor com indicação de erro no Indexador.

Se a instrução a executar é um CSV M:IO, o Interpretador prepara um CONTROL BOLCK e um BUFFER para o programa na Zona de Dados Comum do Sistema e executa um CSV M:ZIO para este CONTROL BLOCK. O controle das operação de entrada e saída é feita pelo Supervisor, e será detalhado no Capítulo referente a este.

O programa pode ser interpretado com OVERLAP de I/O e processamento interno enquanto não aparecer uma instrução CSV M:WAIT.

A execução de um CSV M:WAIT força a devolução de controle ao Supervisor (ver Distribuidor) e modificação do #STATUS para "aguardando fim de I/O". O programa só voltará a ser executado ao fim da operação de E/S. A execução de CSV M:EXIT provoca a devolução do controle ao Supervisor com #STATUS igual a &0000 e fim de execução do programa.

As demais instruções, inclusive os demais CSV, são interpretados diretamente.

Após a execução de cada instrução, o Executor salva os conteúdos dos registradores modificados e dos indicadores CARRY ou OVERFLOW, quando a instrução os afeta (ver MITRA 15 - Manuel de Présentation). Antes de devolver o controle ao Distribuidor, o Executor verifica se o novo valor de #CTPROG está dentro dos limites da área utilizada da LPS atual, e se não estiver, coloca uma indicação de erro no Indexador.

C A P Í T U L O III

SUPERVISOR

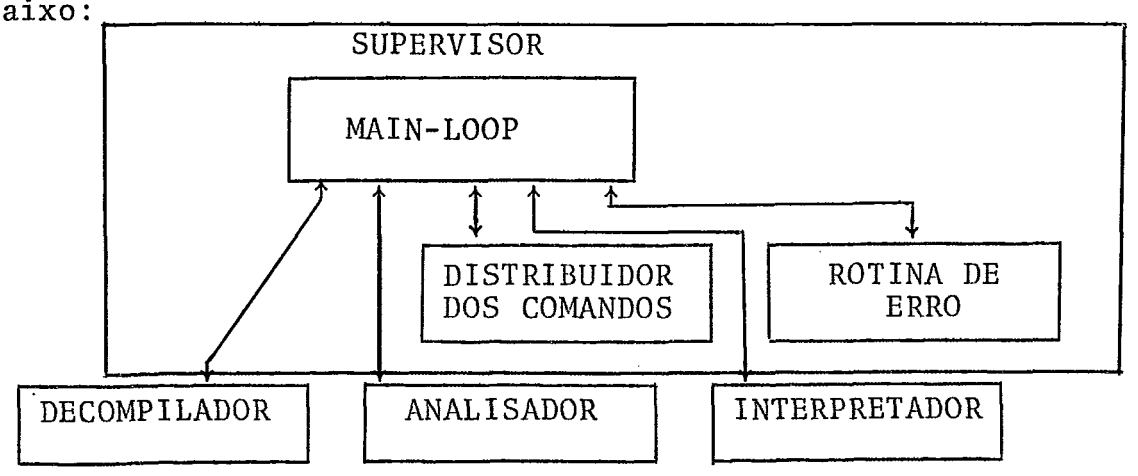
O Supervisor é a parte do Sistema responsável pelo seu controle. Como controlador o Supervisor deve determinar sobre que programa o Sistema passará a atuar, e de que maneira será exercida esta ação.

O programa é determinado através do "MAIN-LOOP", enquanto a ação a ser executada sobre o programa é determinada pelo seu contexto, mais especificamente, pelo seu STATUS.

Além das funções acima mencionadas o Supervisor controla as entradas e saidas de cada programa, assim como inicia e controla as entradas e saidas do próprio Sistema.

Finalmente, o Supervisor analisa e executa os comandos do Sistema, alterando o contexto do programa quando for necessário.

Esquematicamente o Supervisor apresenta-se como na Figura abaixo:



CONTEXTO DE UM PROGRAMA

O contexto de um programa é o conjunto de apontadores e campos de trabalho, que o descrevem em um intervalo de tempo.

O Sistema considera, que os programas podem assumir dois estados básicos distintos:

- INTERPRETAÇÃO e
- NÃO-INTERPRETAÇÃO

O contexto de um programa no estado de não-interpretação (montagem, alteração ou decompilação) descreva a seção comum (CDS), e a seção local (LDS e LPS) sobre a qual o Sistema está atuando. No estado de interpretação, além de descrever a seção comum(CDS) e a seção local (LDS e LPS) que está sendo interpretada, o contexto armazena o conteudo dos registradores e o estado dos indicadores do programa.

O contexto de um programa pode ser modificado, de modo a refletir a nova situação do programa, após o Sistema ter atuado sobre o mesmo.

Observe-se que o intervalo de tempo acima mencionado é diretamente dependente da ação sendo exercida sobre o programa. Este intervalo pode ser o tempo decorrido para:

- Análise e montagem de uma diretiva ou instrução;
- Preparação de um Buffer para a decompilação;
- Alteração de uma seção;
- Interpretação de uma instrução;

sendo portanto, extremamente variável.

ORGANIZAÇÃO E MANIPULAÇÃO DOS CONTEXTOS

Os contextos são manipulados pelo Supervisor através de um conjunto de apontadores e indicadores, e estão organizados de modo a que, apenas o contexto do programa sobre o qual o Sistema passará a atuar, é copiado na zona diretamente acessível da CDS. Na zona indiretamente acessível da CDS encontra-se o contexto de cada um dos programas capazes de existir no Sistema. Antes de abandonar um programa o Supervisor copia o seu contexto (diretamente acessível) na posição correspondente da zona indiretamente acessível da CDS.

O contexto de cada programa está dividido em duas partes:

- contexto atual
- contexto anterior

endereçaveis por um par de apontadores armazenados em uma Tabela (TABCTX).

O contexto atual é aquele que será copiado na zona diretamente acessível da CDS quando o Sistema passar a atuar sobre o programa.

O contexto anterior poderá existir apenas no estado de não-interpretação quando alguma alteração ou decompilação estiver em curso. Neste caso as duas palavras que compõe o contexto anterior armazenam #APQPRT e #STATUS da seção para onde o sistema deve retornar finda a alteração ou decompilação. Quando nenhuma alteração está em curso o conteudo de ambas as palavras é nulo.

A Figura apresenta uma visão global do contexto cuja manipulação passamos a descrever:

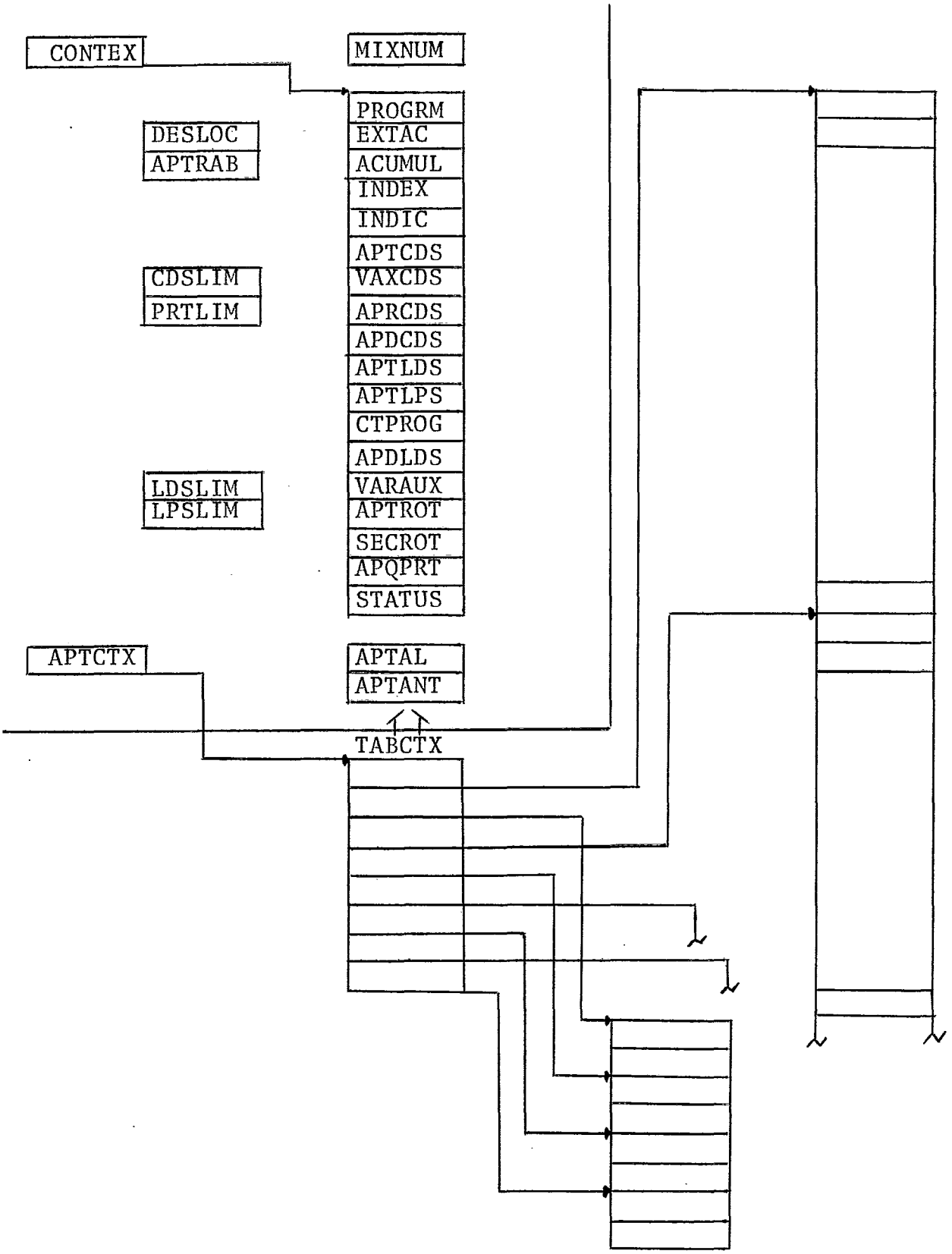
O indicador MIXNUM assume um dos valores 0,4,8,12 e serve de indexador para através APTCTX acessarmos a tabela TABCTX da qual obtemos os apontadores de contexto atual (APTAL) e contexto anterior (APTANT) do programa.

Através APTAL podemos trazer o contexto do programa da zona indiretamente acessível da CDS para a zona diretamente acessível da CDS e vice-versa.

Através APTANT podemos fazer o programa retornar a seção que estava sendo montada quando o sistema recebeu um comando de alteração ou decompilação.

O Analisador armazena a parte referente a cada seção (geral ou local) do programa na entrada correspondente da PRT, e em palavras reservadas da própria seção (ver Analisador - Parte I). Devido a isto, no estado de não-interpretação o Supervisor é capaz de recriar o contexto de qualquer seção, a partir dos valores determinados para #APQPRT e #STATUS.

A mudança de seção durante a interpretação do programa é feita pelo próprio Interpretador a partir dos valores determinados para #APQPRT e #CTPROG.


CONTEX
MIXNUM
PROGRM
EXTAC
ACUMUL
INDEX
INDIC
APTCDS
VAXCDS
APRCDS
APDCDS
APTLDS
APTLPS
CTPROG
APDLDS
VARAUX
APTROT
SECROT
APQPRT
STATUS
DESLOC
APTRAB
CDSLIM
PRTLIM
LDSLIM
LPSLIM
APTCTX
APTAL
APTANT
TABCTX

DESCRIÇÃO DO CONTEXTO

O contexto de um programa é composto de 18 palavras consecutivas da memória. Algumas palavras do contexto assumem significados distintos dependendo do estado do programa, conforme a Tabela abaixo:

PALAVRA	NOME	DESCRIÇÃO
00	PROGRM	Apontador para o endereço zero do programa.
02	EXTAC	Armazena o conteudo da extensão do acumulador durante a interpretação do programa.
	DESLOC	Armazena o valor do deslocamento relativo ao inicio da área de dados da seção (geral ou local), que está sendo montada.
04	ACUMUL	Aramzena o conteudo do acumulador durante a interpretação do programa.
	APTRAB	Apontador para o próximo endereço disponível da área de dados da seção (geral ou local) que está sendo montada.
06	INDEX	Armazena o conteudo do indexador durante a interpretação do programa.
08	INDIC	Armazena o conteudo dos indicadores CARRY e OVERFLOW (BYTES 0 e 1 da palavra, respectivamente) durante a interpretação do processo.

PALAVRA	NOME	DESCRIÇÃO
10	APTCDS	Apontador para o endereço zero da Tabela de Nomes da CDS do programa.
12	VAXCDS	Apontador para o próximo endereço disponível para um nome definido na Tabela de Nomes da CDS do programa, durante a não-interpretação.
	CDSLIM	Apontador para o endereço limite utilizado da área de dados comuns do programa, durante a interpretação.
14	APRCDS	Apontador para o próximo endereço disponível para um nome não-definido na Tabela de Nomes da CDS do programa, durante a não-interpretação.
	PRTLIM	Apontador para o endereço limite utilizado da PRT do programa, durante a interpretação.
16	APDCDS	Apontador para o endereço zero da área de dados comuns do programa.
18	APTLDS	Apontador para o endereço zero da Tabela de Nomes da LDS atual do programa.
20	APTLPS	Apontador para o endereço zero da LPS atual do programa.
22	CTPROG	Apontador para o endereço da próxima instrução a montar, decompilar ou interpretar da LPS atual do programa.

PALAVRA	NOME	DESCRIÇÃO
24	APDLDS	Apontador para o endereço zero da área de dados da LDS atual do programa.
26	VARAUX	Apontador para o próximo endereço disponível para um nome definido na Tabela de Nomes da LDS atual do programa, durante a não-interpretação.
	LDSLIM	Apontador para o endereço limite utilizado da área de dados da LDS atual do programa, durante a interpretação.
28	APTROT	Apontador para o próximo endereço disponível para um nome não-definido, nome externo geral, nome de seção ou rótulo na Tabela de Nomes da LDS atual do programa, durante a não-interpretação.
	LPSLIM	Apontador para o endereço limite utilizado na LPS atual do programa, durante a interpretação.
30	SECROT	Armazena o contador de seções chamadas na LPS atual (BYTE 0 da palavra) e o contador de rótulos referenciados na LPS atual (BYTE 1 da palavra).
32	APQPRT	Apontador para o endereço da PRT correspondente a seção atual do programa.

PALAVRA	NOME	DESCRIÇÃO
34	STATUS	Armazena o estado atual do programa. O Anexo I descreve detalhadamente os sub-estados possíveis dentro de cada estado.

COMANDOS DO SISTEMA

Visando permitir uma comunicação fácil entre os usuários e o Sistema existe um conjunto de comandos cuja sintaxe e semântica serão apresentadas a seguir.

SINTAXE

Os comandos do Sistema são de formato fixo e caracterizados pelo carater "!" na coluna 0. O nome do comando começa na coluna 2 e os parâmetros (quando houver) iniciam na coluna 6.
NOTA: O carater ß representa um espaço.

```
<comando-do-sistema>::=!ß<comando-simples>|
                       !ß<comando-alterador>|
                       !ß<comando-decompilador>|
<comando-simples>::= ALO/BYE/GO/RED/RTN/RUN
<comando-alterador>::=<alterador-geral>ß<parâmetro-geral>|
                      <alterador-local>ß<parâmetro-local>|
                      <alterador-programa>ß<parâmetro-programa>|
```

<alterador-geral>::= CDS

<parâmetro-geral>::=<expressão-delimitadora>

<alterador-local>::= LDS

<parâmetro-local>::=<nome-seção-dados><expressão-delimitadora>

<alterador-programa>::= LPS

<parâmetro-programa>::=<nome-seção-programa><expressão-delimitadora>

<nome-seção-dados>::={nome de LDS com 6 caracteres ajustados a esquerda}

<nome-seção-programa>::={nome de LPS com 6 caracteres ajustados a esquerda}

<expressão-delimitadora>::=<delimitador><expressão>

<delimitador>::=<inteiro-256>

<expressão>::=+<inteiro-256>|-<inteiro-256>

<inteiro-256>::={inteiro maior ou igual a zero e menor que 256}

<comando-decompilador>::=<decompilador>|
<decompilador><parâmetro>

<decompilador>::= LST|PCH

<parâmetro>::= CDS|LDSb<nome-seção-dados>|
LPSb<nome-seção-programa>

SEMÂNTICA

ALO - Este comando permite ao usuário criar um programa associando-o ao terminal. O programa criado passa a aguardar a diretiva CDS, pelo teclado ou por fita de papel (ver o comando RED).

É necessário que nenhum programa esteja associado ao ter

minal quando o comando for enviado, caso contrário será erro e o comando ignorado.

Exemplo:! ALO

BYE - Este comando permite ao usuário destruir o programa associado ao terminal. O comando pode ser enviado a qualquer momento em que o teclado estiver habilitado e não for entrada causada pelo programa (interpretação).

Exemplo:! BYE

GO - Este comando permite ao usuário informar ao sistema que deseja continuar a execução do programa (ver Interpretador).

Exemplo:! GO

RED - Este comando permite ao usuário construir o programa ou parte do programa, através entrada por fita de papel. O comando só pode ser recebido se o programa estiver pronto para receber uma diretiva de segmentação (CDS, LDS, LPS). A fita será lida até que seja encontrado o registro de fim de fita ou a diretiva END. Em ambos os casos o usuário recebe em seu terminal uma mensagem reabilitando entrada pelo teclado.

Exemplo:! RED

RTN - Este comando permite ao usuário informar que concluiu as alterações que fazia no programa e quer retornar ao ponto em que se encontrava quando enviou o primeiro de uma

série de <comando-alterador> ao sistema.

Exemplo:! RTN

RUN - Este comando permite ao usuário executar o programa. O comando só é aceito se o programa tiver passado pela ROTINA DO SEGUNDO PASSO, sem erros.

Exemplo:! RUN

CDS - Este comando permite ao usuário alterar a área de dados comuns do programa através um dos procedimentos a baixo relacionados, onde o <delimitador> indica o nome de dados após o qual serão feitas as alterações:

-supressão-os dados logicamente associados com os <inteiro-256> nome de dados após o < delimitador> serão suprimidos.

Exemplo:! CDS 3-2

Neste caso, o quarto e quinto nomes de dados e os dados a eles associados serão suprimidos.

-adição- o usuário deseja acrescentar dados associados com inteiro-256 nomes de dados (0 delimitador é irrelevante neste caso).

Exemplo:! CDS 5+3

Neste caso, três nomes de dados poderão ser inseridos entre o último nome de dados da seção e a diretiva FIN.

LDS - Este comando permite ao usuário alterar a área de dados locais da seção referenciada no comando através um dos procedimentos abaixo relacionados, onde o <delimitador> indica o nome de dados a partir do qual serão feitas as alterações:

-supressão-os dados logicamente associados com os <inteiro-256> nomes de dados após o delimitador serão suprimidos.

Exemplo: ! LDS SEÇÃO 8-2

Neste caso o nono e décimo nomes de dados e os dados a eles associados serão suprimidos.

-adição- o usuário deseja acrescentar dados associados com <inteiro-256> nomes de dados (O <delimitador> é irrelevante neste caso).

Exemplo: ! LDS SEÇÃO 5+3

Neste caso, três nomes de dados poderão ser inseridos entre o último nome de dados da seção e a diretiva FIN.

LPS - Este comando permite ao usuário alterar a área de programa da seção referenciada no comando através um dos procedimentos abaixo relacionados, onde o <delimitador> indica a instrução a partir da qual serão feitas as alterações:

-supressão-<inteiro-256> instruções após o<delimitador> serão suprimidas.

Exemplo: ! LPS PROGRA20-3

Neste caso as instruções 21, 22 e 23 serão suprimidas.

-adição- <inteiro-256> instruções serão inseridas após o <delimitador>.

Exemplo: ! LPS PROGRA9+3

Neste caso três instruções serão inseridas entre as instruções nove e dez.

LST - Este comando permite ao usuário listar o programa ou parte do programa.

Exemplos: ! LST
! LST CDS
! LST LDS DADOS
! LST LPS PROGRA

PCH - Este comando permite ao usuário perfurar o programa ou parte do programa.

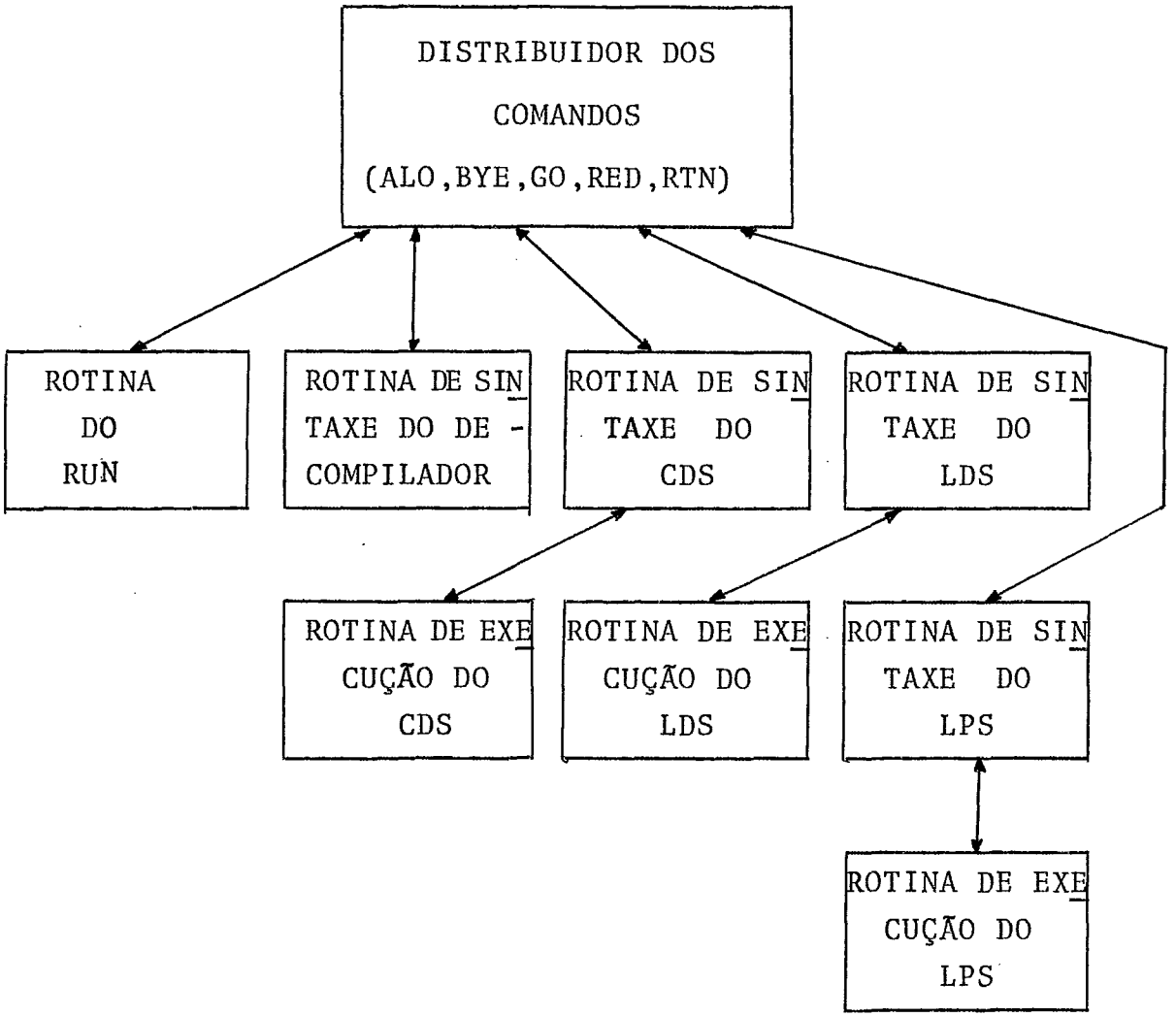
Exemplos: ! PCH
! PCH CDS
! PCH LDS DATUM
! PCH LPS INSTRU

DISTRIBUIDOR DOS COMANDOS

A função do distribuidor dos comandos é verificar a validade do comando recebido e passar o controle para a rotina que especificamente trata deste comando. Se o comando for inválido o

controle é devolvido ao "MAIN-LOOP" com o aviso de erro de sintaxe.

Abaixo está apresentado um esquema do Distribuidor de Comandos e as rotinas que ele pode chamar.



A seguir apresentamos uma descrição de cada rotina.

ROTINA DE ! ALO

Esta rotina é executada pelo próprio Distribuidor de Comandos e pode ser dividida em dois passos:

- SINTAXE
- EXECUÇÃO

SINTAXE - Se o #STATUS do programa é diferente de &F000 o controle é devolvido ao "MAIN-LOOP" com aviso de erro de sintaxe.

EXECUÇÃO - A área de trabalho do programa é preenchida com o carater "NULL" (00), o #STATUS passa a ser &0001 e #APQPRT recebe o valor de #PROGRM.
O controle é devolvido ao "MAIN-LOOP" de modo a que, a Rotina de Mensagens do Sistema envie um aviso de início de trabalho para o usuário.

ROTINA DE ! BYE

Esta rotina é executada pelo próprio Distribuidor de Comandos.

O #STATUS do programa é feito igual a &F000 e o controle é devolvido ao "MAIN-LOOP" de modo a que, a Rotina de Mensagens do Sistema envie um aviso de fim de trabalho para o usuário.

ROTINA DE ! GO

Esta rotina é executada pelo próprio Distribuidor de Comandos e pode ser dividida em dois passos:

- SINTAXE
- EXECUÇÃO

SINTAXE - Se o #STATUS for diferente de &F003 o controle é devolvido ao "MAIN-LOOP" com aviso de erro de sintaxe.

EXECUÇÃO - O #STATUS é modificado para &2000, o programa entra na lista de interpretável.
O controle é devolvido ao "MAIN-LOOP".

ROTINA DE ! RED

Esta rotina é executada pelo próprio Distribuidor de Comandos e pode ser dividida em dois passos:

- SINTAXE
- EXECUÇÃO

SINTAXE - Se os requisitos de #STATUS não forem atendidos o controle é devolvido ao "MAIN-LOOP" com aviso de erro de sintaxe.

NOTA: No momento atual o sistema MITRA 15 da COPPE dispõe de leitora/perfuradora de fita de papel apenas na console. Deste modo, se a leitora/perfuradora esti

ver ocupada, o controle é devolvido ao "MAIN-LOOP" com aviso de ocupação da leitora/perfuradora e o comando é ignorado.

EXECUÇÃO - O programa é logicamente associado ao console do Sistema ao ser modificado o BYTE zero do #STATUS para &80. O teclado do terminal do usuário é desabilitado. O Sistema é marcado como ocupando a console.

ROTINA DE ! RUN

A Rotina de ! RUN está dividida em dois passos:

- SINTAXE
- EXECUÇÃO

SINTAXE - É verificada pelo próprio Distribuidor de Comandos. Se o #STATUS do programa é diferente de &F002 o controle é devolvido ao "MAIN-LOOP" com aviso de erro de sintaxe.

EXECUÇÃO - É realizada por uma seção (ROTINA DE LINK-EDIÇÃO) , chamada ao fim da análise sintática do comando.
A LINK-EDIÇÃO consiste em transformar uma estrutura adequada a decompilação e alteração, numa estrutura adequada a interpretação.
A ação da LINK-EDIÇÃO é exercida diretamente sobre as diretivas DATA tipo etiqueta da CDS e de cada LDS.
As referências a nomes de dados ou rótulos são trans

formadas em referência aos próprios dados ou instruções.

Finalmente, é criado o contexto de interpretação do programa, ou seja: a parte do contexto referente a primeira seção a ser interpretada é construida, #CT PROG é posicionado na primeira instrução a interpretar, os campos de armazenamento dos registradores e indicadores é zerado e o #STATUS do programa é mudado para &2000.

ROTINA DE ! LST E ! PCH

A Rotina de ! LST e ! PCH está dividida em dois passos:

- SINTAXE
- EXECUÇÃO

SINTAXE - É verificada por uma seção(Rotina de Sintaxe do Decompilador) chamada pelo Distribuidor dos Comandos. Se o programa está marcado como "sendo alterado"(palavra 10 do programa - ver Analisador-Parte I)o controle é devolvido ao Distribuidor de Comandos com aviso de erro de sintaxe.

Se a decompilação é parcial (de uma LDS ou LPS) e a seção referenciada não existe o controle é devolvido ao Distribuidor de Comandos com aviso de erro de

sintaxe.

A sintaxe estando correta os valores de #APQPRT e #STATUS são armazenados no contexto anterior do programa. É criado um contexto atual para o programa e o #STATUS é posicionado como indicado no Apêndice 1. O teclado é desabilitado.

NOTA: As observações feitas quanto a leitora/perfuradora de fita de papel no comando ! RED também são válidas para ! PCH.

EXECUÇÃO - É realizada por uma seção (Distribuidor) chamada pelo "MAIN-LOOP" ao fim de cada impressão ou perfuração.

A decompilação de um processo anda em paralelo com o Analisador e Interpretador.

O fim de uma decompilação é determinada pela combinação do #STATUS com o apontador de limite da seção, p.ex. #VARAUX.

Se a decompilação terminada é por perfuração, um último registro é perfurado como indicação de fim de arquivo para o monitor do MITRA 15 (ver MITRA 15 - Manuel de Presentation).

Ao fim da decompilação o contexto anterior é utilizado para recriar o contexto atual ao tempo do comando decompilador. O teclado é reabilitado.

ROTINAS DE ALTERAÇÃO (! CDS, ! LDS, ! LPS)

As Rotinas de Alteração seguem todas o mesmo flow básico. Existe sempre uma Rotina de Sintaxe do Comando CDS (LDS ou LPS), encarregada de uma sintaxe prévia do comando, a qual é chamada pelo Distribuidor de Comandos, e uma Rotina de Execução do Comando CDS(LDS ou LPS) chamada pela Rotina de Sintaxe correspondente que examina, sobre a própria seção, a alteração a fazer,e a executa ou não.

SINTAXE - Se o comando alterador não é ! CDS, e a seção referenciada não existe, o controle é devolvido ao"MAIN-LOOP" com o aviso de erro de sintaxe.

A <expressão-delimitadora> é verificada contra os limites da seção a alterar. Se estes limites não forem respeitados o controle é devolvido ao "MAIN-LOOP" com o aviso de erro de sintaxe.

Se o programa não estiver marcado como "sendo alterado" (palavra 10 do programa - ver Analisador-Parte II), os valores de #APQPRT e #STATUS são armazenados no seu contexto anterior, antes de ser criado um novo contexto atual. O programa é marcado como "sendo alterado".

EXECUÇÃO - A Rotina de Execução trata de dois casos distintos:

- INSERÇÃO
- SUPRESSÃO

INSERÇÃO - O processamento de inserção é bastante simples.
A rotina abre espaço para as novas diretivas através supressão da diretiva FIN (comandos ! CDS,! LDS) e para novas instruções através um deslocamento das que vem após aquela indicada pelo <delimitador>.
A área onde serão feitas as inserções é preenchida com o carater "NULL" (00) e o #STATUS posicionado para a fase de montagem correspondente ao comando.
O controle é devolvido ao "MAIN-LOOP" e as inserções serão feitas pelo sistema de montagem normal, através o Analisador.

SUPRESSÕES - O processamento de supressão é, por outro lado, bastante complexo.
Consideremos as supressões numa LDS. A supressão de nomes definidos numa LDS implica em verificar se existe referência a estes nomes na LPS correspondente. A referência a algum dos nomes implica em devolução do controle a Rotina de Sintaxe correspondente com o aviso de supressão não realizada.
Cada um dos nomes suprimíveis devem ser pesquisados na Tabela de Nomes da CDS e a referência a algum deles deve ter o seu tipo trocado para externo local.
Cada nome suprimivel é verificado quanto a referência a ele em diretiva DATA tipo etiqueta. Se houver o nome é tornado não definido e a referência modificada para sua nova posição.
Após a supressão física dos nomes e dos dados a eles

associados, a Tabela de Nomes da LDS é realocada e o argumento de todas as instruções com referência aos nomes realocados são ajustados a nova situação. Da mesma forma as referências aos nomes realocados dentro de diretivas DATA tipo etiqueta devem ser ajustadas para as novas posições.

A supressão de nomes definidos da CDS processa-se de forma idêntica a exposta acima, porém a verificação dos nomes é mais trabalhosa, uma vez que deve ser feita em cada LDS (em diretivas DATA tipo ETIQUETA) e em cada LPS.

A supressão de instruções é precedida pela modificação do rótulo de cada instrução suprimível (com rótulo) de rótulo definido para rótulo não definido.

Finalizando uma rotina de supressão os nomes não definidos são verificados quanto a referência em alguma diretiva DATA tipo etiqueta sendo que os rótulos são verificados quanto a referência em alguma instrução. A não existência de qualquer referência a um nome não definido implica em sua supressão.

O #STATUS do programa é sempre colocado em &F004 numa supressão.

ROTINA DE ! RTN

Esta rotina é executada pelo próprio Distribuidor de

Comandos e pode ser dividida em dois passos:

- SINTAXE
- EXECUÇÃO

SINTAXE - Se o programa não está marcado como "sendo alterado" o controle é devolvido ao "MAIN-LOOP" com o aviso de erro de sintaxe.

EXECUÇÃO - O contexto anterior é utilizado para construir o contexto atual para a seção sendo montada ao tempo do primeiro comando alterador.

COMENTÁRIOS - Ao ser recebido um novo comando alterador ou o comando ! RTN é feito um trabalho direto sobre a seção (comum ou local(LDS e LPS)) alterada de modo a não haver buracos na estrutura do programa. Os campos reservados da seção utilizados para recriar seu contexto são todos ajustados a nova situação.

Se uma diretiva FIN tiver sido suprimida é forçado o #STATUS &F005 do qual o programa sai somente quando a diretiva FIN suprimida é recriada.

Um programa pode ser alterável em qualquer ponto da montagem, sempre que o programa estiver entrando pelo teclado.

ROTINA DE MENSAGENS DO SISTEMA

A Rotina de Mensagens está subordinada ao Supervisor, que a utiliza quando tem que dar ao usuário um aviso sobre uma ocorrência em seu programa. Uma ocorrência pode ser um aviso de início ou fim de trabalho ou uma pergunta se o usuário quer continuar a interpretar seu programa, ou um erro ocorrido no programa. Para os erros o Supervisor recebe do Analisador e do Interpretador a informação de que ocorreu um erro e já no indexador a entrada para o tipo de erro ocorrido.

As mensagens a ser dadas ao usuário vem precedida de dois asteriscos e se encontram na Tabela de Mensagens do Sistema,ocupando uma palavra para cada mensagem.

As mensagens EI (Endereço Inválido), NI (Nome Indefinido) e OI (Operando Inválido) são seguidas do número da seção (LDS e LPS) onde ocorreu o erro.

A seguir apresentaremos a Tabela de Mensagens com sua descrição.

TABELA DE MENSAGENS DO SISTEMA		
ENTRADA	MENSAGEM	DESCRIÇÃO
0	CE	(Continuar Execução)-Pergunta se o usuário quer continuar a interpretar o programa SIM ! GO NÃO ! BYE

ENTRADA	MENSAGEM	DESCRIÇÃO
2	CO	(Console Ocupado)-Avisa que o console está ocupado por outro programa.
4	ED	(Estouro da Área de Dados)-Diretiva não aceita, não tem espaço para os dados.
6	EI	(Endereço Inválido)-Tentativa de modificar instrução ou endereço fora de limites.
8	EN	(Estouro na Área de Nomes de Dados)-Diretiva não aceita, não tem espaço para nome de dados.
10	EP	(Estouro da Área do Programa)-Instrução não aceita, não tem espaço para instrução.
12	ER	(Estouro da Área da PRT)-Diretiva de Segmentação não aceita, não tem espaço para outra seção.
14	ES	(Erro de Sintaxe)-erro de sintaxe da diretiva ou instrução ou comando do sistema.
16	FT	(Fim de Trabalho)-Terminou a execução do programa ou foi recebido o comando ! BYE.
18	IT	(Início de Trabalho)-Início de um programa,foi recebido o comando ! ALO.
20	NI	(Nome Indefinido)-Nomes de dados ou rótulos de programas ou nomes de seções não definidos.

ENTRADA	MENSAGEM	DESCRIÇÃO
22	OI	(Operando Inválido)-Modo de endereçamento inválido, instrução só aceita operando de endereço PAR.
24	SN	(Supressão Não Realizada)-Um comando alterador tentou suprimir um nome referenciado na LPS. O programa está na seção referenciada neste comando. Nenhum nome foi suprimido.
26	TS	(Termine a Seção)-Uma seção foi alterada com supressão da diretiva FIN. O usuário é forçado a recolocar a diretiva FIN.

OBSERVAÇÃO: As mensagens aqui apresentadas são as que ficam,quando tivermos a plena certeza que o Sistema implantado não possuir mais erro. Frisamos que, no momento da implantação, deverá ocorrer erros do próprio Sistema e será necessário criar mensagens que nos ajudem a depurá-lo.

MAIN-LOOP

O "MAIN-LOOP" é a parte responsável pelo "TIME-SHARING" do Sistema. Podemos considerá-lo como dividido em duas partes principais. A primeira inicializa o Sistema, dando valores ao #PROGRM de cada programa, zerando todos os #STATUS e iniciando uma leitura

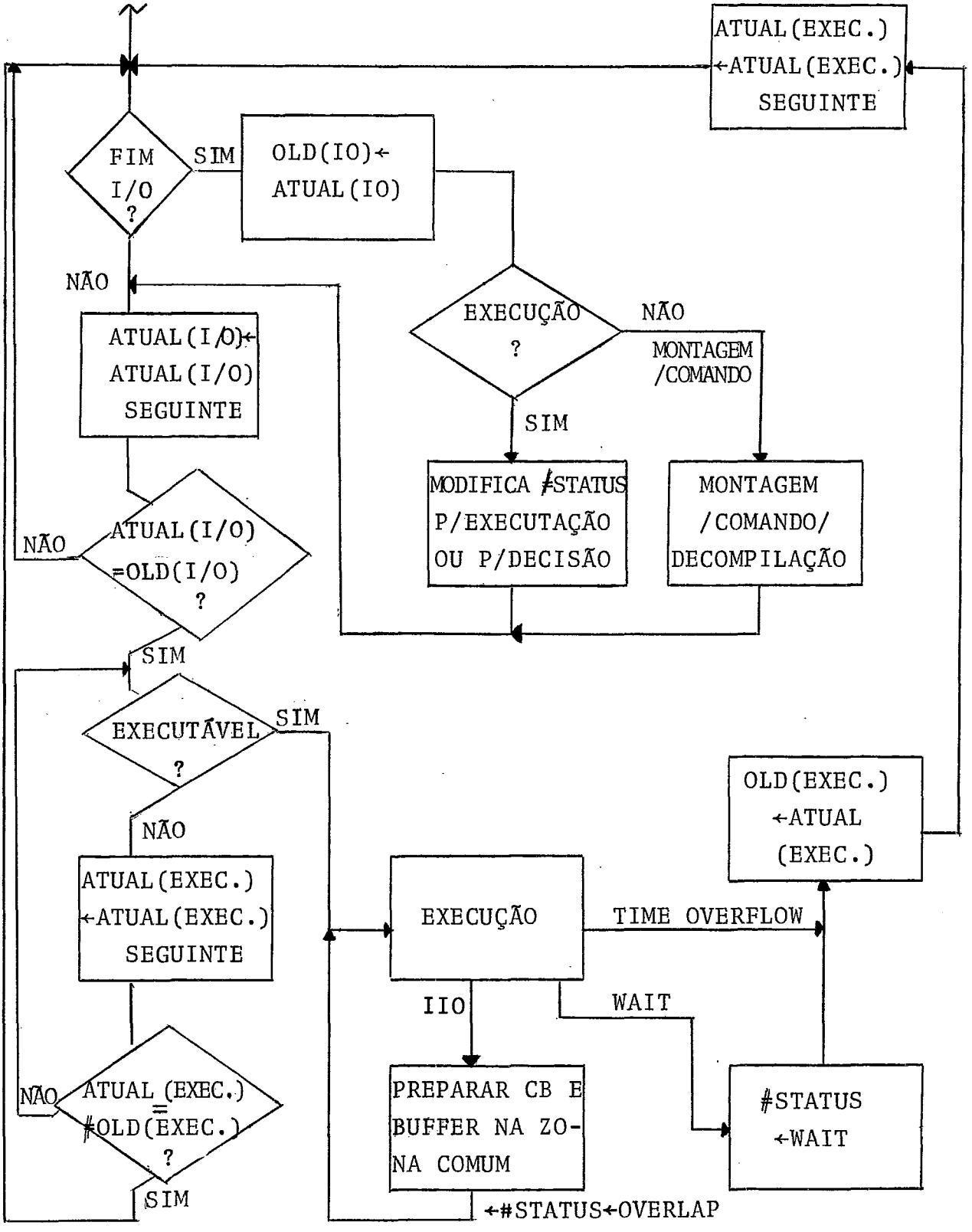
para todos os terminais. A segunda é efetivamente o "MAIN-LOOP" , pois é a que controla o Sistema. Este controle é exercido com o auxílio de duas listas:

a) Lista das Entradas e Saídas

b) Lista dos Programas

O esquema anexo representa o controle destas listas. Observa-se que os programas prontos para interpretação só serão executados quando todas as entradas e saídas estiverem pendentes.

Para cada lista existe um par de apontadores. O apontador denominado de ATUAL, aponta o próximo programa a ser analisado em termos de entrada e saída ou execução. O apontador denominado de OLD, aponta o último programa atendido em termos de entrada e saída ou execução.


ATUAL(EXEC.)
←ATUAL(EXEC.)
SEGUINTE
FIM
I/O
?
SIM
OLD(IO)←
ATUAL(IO)
NÃO
ATUAL(I/O)←
ATUAL(I/O)
SEGUINTE
EXECUÇÃO
?
NÃO
MONTAGEM
/COMANDO
SIM
MODIFICA #STATUS
P/EXECUTAÇÃO
OU P/DECISÃO
MONTAGEM
/COMANDO/
DECOMPILAÇÃO
NÃO
ATUAL(I/O)
=OLD(I/O)
?
SIM
EXECUTÁVEL
?
SIM
NÃO
ATUAL(EXEC.)
←ATUAL(EXEC.)
SEGUINTE
NÃO
ATUAL(EXEC.)
≠OLD(EXEC.)
?
SIM
EXECUÇÃO
TIME OVERFLOW
OLD(EXEC.)
←ATUAL
(EXEC.)
IIO
WAIT
PREPARAR CB E
BUFFER NA ZO-
NA COMUM
#STATUS
←WAIT
←#STATUS←OVERLAP

C A P Í T U L O IV

CONCLUSÃO

Como foi dito na Introdução (Tese "Um Sistema de Pro - gramação e Depuração Conversacional para Linguagem Tipo Montador Parte I Analisador e Montador") o sistema projetado vem possibilitar ao usuário do MITRA 15 uma maior facilidade tanto do ponto de vista de sua maior utilização, por poder ser usado por um maior número de pessoas de maneira mais simples, como pela maior facilidade na depuração de programas.

Outra vantagem é que, sendo o Sistema projetado, protegido contra invasões de áreas proibidas, contra modificação de instruções do programa e contra endereço inválido para as instruções, todo o programa que passar pelo Sistema e que posteriormente for executado fora dele, permitirá a utilização de um montador enormemente simplificado, já que o programa está correto.

A garantia da proteção dada pelo Sistema, permite que a máquina seja mais simples e de custo mais baixo, uma vez que elimina a necessidade de sistemas de proteção (BITS ou chaves de acesso, registrador limite, etc).

Quanto a implementação do Sistema recomendamos que seja gradativa, isto é, primeiro implementar o Sistema sem fazer TIME-SHARING, fazendo um Supervisor simplificado, trabalhando com um só programa. Somente depois que Analisador, Interpretador e Rotinas de Comandos estiverem funcionando é que se deve implantar um Super-

visor definitivo.

O uso que se fará do Sistema é que permitirá uma avaliação de sua estrutura e apontará as modificações necessárias, em termos de estrutura e do grupo de diretivas e instruções a simular.

Podemos ver que o Sistema foi previsto para uma interação muito grande com o usuário, aproveitando as características modulares da linguagem simbólica do MITRA 15, e tem como objetivo maior permitir a depuração de programas, os quais podem ser cindidos e testados em módulos de poucas seções.

Sistemas semelhantes podem ser desenvolvidos para minicomputadores de outros fabricantes e portanto, o trabalho tem aplicação ampla e representa uma contribuição significativa para o melhor aproveitamento dos mesmos.

BIBLIOGRAFIA

1. BARRON, D.W. : "Assembler and Loaders", MacDonald: London and American Elsevier Inc.: New York,pp. 1-47,1972, 2nd. Edition.

2. CII: MITRA 15 - "Manuel de Presentation", 1973.

3. CII: MITRA 15 - "Manuel de Reference", Juin 1972.

4. CII: MITRA 15 - "Manuel d'Utilisation", Moniteur de Base MOB , Juillet 1973.

5. GEAR : "Computer Organization and Programming",McGraw-Hill, pp. 59-122, 1969.

6. KNUTH, D.E. : "The Art of Computer Programming",Addison Wesley, Vol. 1, pp. 235-295, 1968.

7. WATSON, R.W. : "Timesharing System Design Concepts", McGraw-Hill, pp. 3-33, 1970.

A N E X O I

DESCRIÇÃO DO STATUS

O #STATUS é a palavra do contexto utilizada pelo Sistema para determinar o seu fluxo de ação. O BYTE 0 do #STATUS determina o primeiro ramo deste fluxo, ou seja :

Do MAIN-LOOP para o Analisador, ou

Do MAIN-LOOP para o Interpretador, ou

Do MAIN-LOOP para o Distribuidor de Comandos, ou

Do MAIN-LOOP para o Decompilador, ou

Do MAIN-LOOP para o Segundo Passo,

enquanto o BYTE 1 serve para determinar o fluxo intermamente a cada rotina chamada do MAIN-LOOP.

Anexo, é apresentada a interpretação de cada valor do #STATUS.

STATUS	DESCRIÇÃO
F000	A única entrada válida pelo terminal é o comando !.ALO
0001	O programa está pronto para receber a diretiva CDS pelo teclado.
0002	O programa está pronto para receber as diretivas da CDS pelo teclado.
0003	O programa recebeu a diretiva FIN de CDS e está pronto para receber a diretiva LDS pelo teclado.
0004	O programa está pronto para receber as diretivas da LDS pelo teclado.
0005	O programa recebeu a diretiva FIN de LDS e está pronto para receber a diretiva LPS pelo teclado.
0006	O programa está pronto para receber as instruções da LPS pelo teclado.
0007	O programa recebeu a diretiva FIN de LPS e está pronto para receber a diretiva LDS ou a diretiva END pelo teclado.
8001 8002 8003 8004 8005 8006 8007	NOTA : A interpretação destes #STATUS é dada pelo BYTE1, e corresponde a dos #STATUS de BYTE 0 igual a &00. O BYTE0, igual a &80 indica que a entrada está sendo feita por fita de papel.

STATUS	DESCRIÇÃO
7000	O programa está pronto para entrar no Segundo Passo, ou está sendo tratado pelo Segundo Passo.
F001	O programa foi dado como não-interpretável, pelo Segundo Passo. O usuário é obrigado a enviar um comando alterador, ou o comando ! BYE.
F002	O programa foi dado como interpretável pelo Segundo Passo. O usuário só pode enviar um dos comandos ! LST, ! PCH, ! RUN ou ! BYE.
2000	O programa está pronto para ser executado ou está em execução.
2001	O programa está sendo executado com OVERLAP de I/O.
A000	O programa está em WAIT I/O. Ao término do I/O o seu #STATUS deve passar a 2000.
F003	O programa está aguardando decisão do usuário, para continuar a ser executado. O usuário deve enviar um comando ! GO ou BYE.
B000	O programa estava com #STATUS 2001 e está aguardando o FIM do I/O de dados para mudar o seu #STATUS para F003.

STATUS	DESCRIÇÃO
	NOTA : Os #STATUS para o Decompilador devem refletir se o processo está sendo listado (BYTE0 igual a &40) ou perfurado (BYTE0 igual a &C0). Além disto, precisam refletir se a decompilação é total ou apenas de uma seção. O BYTE1 será dividido em duas partes de 4 BITS cada uma. Os 4 BITS de mais alta ordem indicam que a decompilação é total, quando o seu valor é zero. Se apenas uma seção está sendo decompilada, o seu valor é um. Os 4 BITS de mais baixa ordem, indicam o que será decompilado. EXEMPLO: 4001 - Impressão da diretiva CDS numa decompilação total. 4011 - Impressão da diretiva CDS numa decompilação da CDS.
4001	O programa está pronto para ter a sua diretiva CDS impressa numa decompilação total.
4002	O programa está pronto para ter uma diretiva da CDS impressa numa decompilação total.
4003	O programa está pronto para ter uma diretiva LDS atual impressa numa decompilação total.

STATUS	DESCRIÇÃO
4004	O programa está pronto para ter uma diretiva da LDS atual impressa numa decompilação total.
4005	O programa está pronto para ter a diretiva LPS atual impressa numa decompilação total.
4006	O programa está pronto para ter impressa uma instrução da LPS atual numa decompilação total.
F004	O programa sofreu uma alteração de supressão. O usuário só pode entrar com um outro comando alterador ou ! RTN ou ! BYE.
F005	O programa teve uma diretiva FIN suprimida durante uma alteração. O usuário é obrigado a enviar um comando alterador para esta seção e incluir o comando FIN ou o programa permanecerá neste ≠STATUS. O outro comando aceito é ! BYE.

A N E X O II

ALGUNS ASPECTOS DA DIRETIVA DATA TIPO ETIQUETA

A pergunta do porque na diretiva DATA tipo etiqueta, o nome de dados (ou rótulo) à direita da diretiva deve aparecer apenas uma vez na LDS (ou CDS) da seção, é respondida claramente neste apêndice, que mostra um estudo de uma estrutura para o caso de se dar a diretiva DATA tipo etiqueta uma maior flexibilidade, isto é, poder ter a seguinte composição.

RÓTULO	COMANDO	ARGUMENTO
<NOME1>	DATA	[#]<NOME2>[,[#]<NOME3>]...
ALFA	DATA	BETA, #CALA, DADO, BETA
	DATA	# X, Z, BETA

Realmente esta composição, para a diretiva DATA tipo etiqueta é a que maiores dificuldades apresenta, devido à flexibilidade que deve ter a estrutura de nossos programas, ou seja, uma estrutura que permita:

- LISTAGEM (DECOMPILAÇÃO)
- ALTERAÇÃO
- EXECUÇÃO.

Passaremos a chamar, aqui, a diretiva DATA tipo etiqueta de diretiva ETIQ.

Através de alguns "snapshots" apresentaremos a diretiva ETIQ durante diferentes fases da montagem, listagem e alte

rações de uma seção de dados local ou comum.

Os algoritmos aplicáveis a cada um dos casos, também serão apresentados.

MONTAGEM.

A MONTAGEM é considerada a fase seguinte à análise sintática. Através de 4 "snapshots" procuraremos enfatizar duas situações distintas:

- rótulos não definidos são referenciados dentro da diretiva ETIQ.
- rótulos não definidos referenciados dentro da diretiva ETIQ passam a ser definidos.

Seja parte de uma seção:

```
LDS1    LDS
A1      RES 2
A2      TEXT "ERRO"
COCA    DATA A5, OCA, A5
A4      RES,1 4
```

Snapshot 1 - observemos que as referências a A5 formam uma lista encadeada circular.

0				
8	A 1		3	0
16	A 2		7	4
24	C O C A		0	8
32	A 4		6	14
40				
48				
56				
64				
72				
80				
88				
96				
104				
112				
120				
128				
136				
144				
152	0 C A		8	10
160	A 5		8	8′ 12

0	0 0	24		48	
2	0 0	26		50	
4	E R	28		52	
6	R 0	30		54	
8	N160	32		56	
10	N152	34		58	
12	D8	36		60	
14	0 0	38		62	
16	0 0	40		64	
18		42		66	
20		44		68	
22		46		70	

Acima, adotamos uma convenção que seguiremos nos próximos "snapshots":

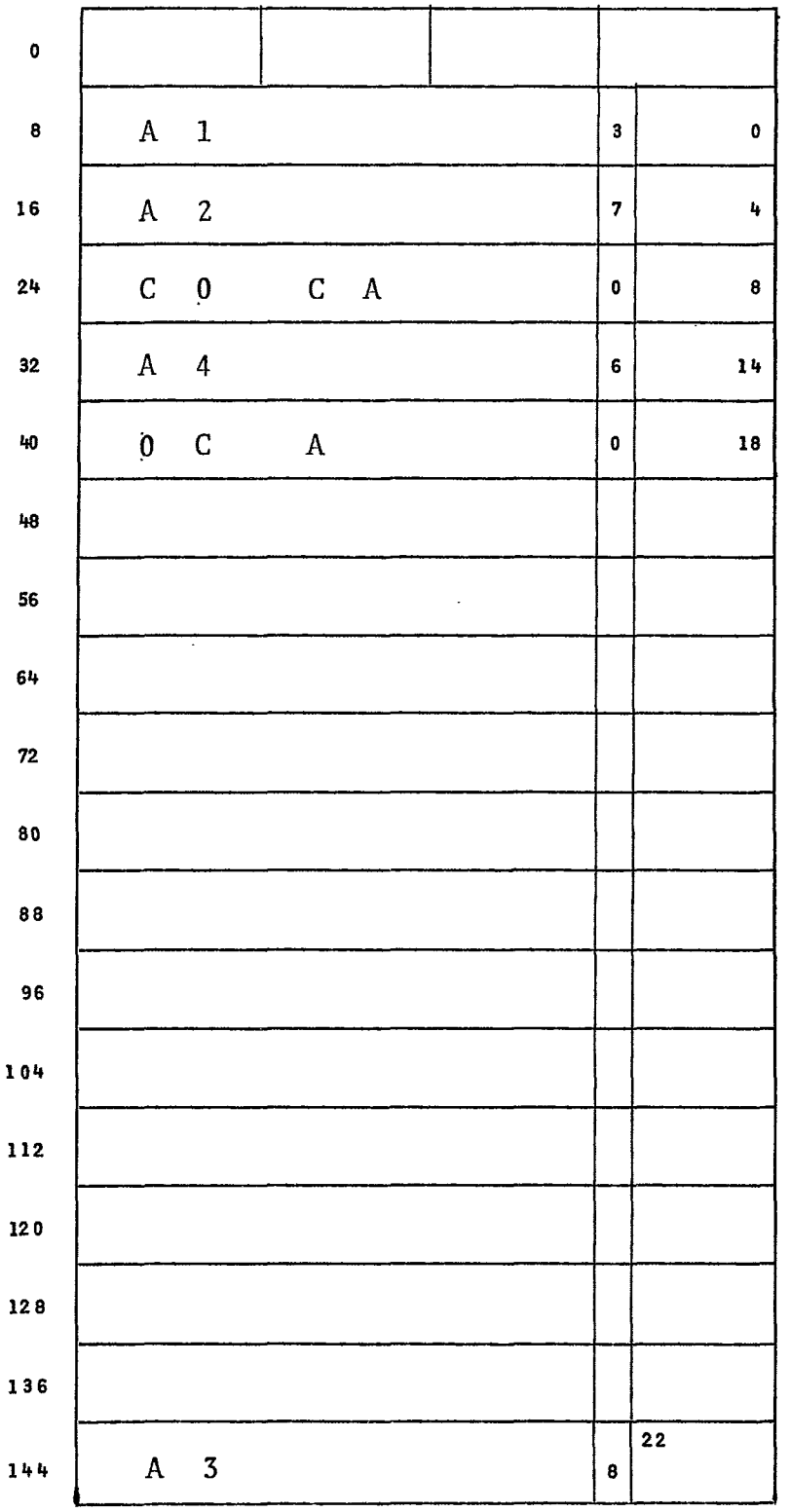
N - indica a área da tabela de rótulos;

D - indica a área de dados.

Acrescentemos uma nova diretiva à seção

```
LDS1   LDS
A1     RES 2
A2     TEXT "ERRO"
COCA   DATA A5, OCA, A5
A4     RES,1 4
OCA    DATA A5, A2, A3
```

Snapshot 2 - Observemos uma nova referência à A5 e a definição de OCA.



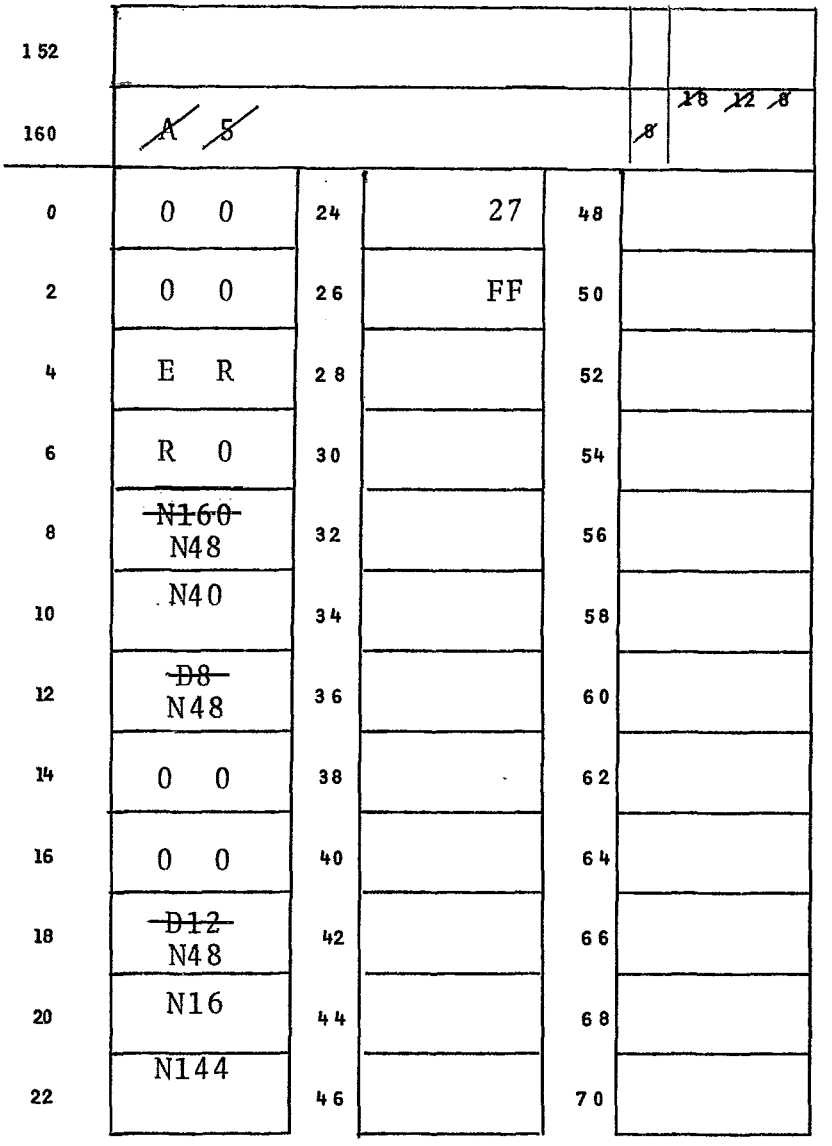
152	~~0 C A~~		~~8~~	~~10~~	
160	A 5		8	8 12 18	
0	0 0	24		48	
2	0 0	26		50	
4	E R	28		52	
6	R 0	30		54	
8	N160	32		56	
10	~~N152~~ N40	34		58	
12	D8	36		60	
14	0 0	38		62	
16	0 0	40		64	
18	D12	42		66	
20	N16	44		68	
22	N144	46		70	

Acrescentemos uma nova diretiva à seção

```
LDS1    LDS
A1      RES 2
A2      TEXT "ERRO"
COCA    DATA A5, 0CA, A5
A4      RES,1 4
0CA     DATA A5, A2, A3
A5      DATA 27, &FF
```

Snapshot 3 - As referências a A5 formavam uma lista encadeada. A lista é desfeita quando A5 é definido.

0			
8	A 1	3	0
16	A 2	7	4
24	C 0 C A	0	8
32	A 4	6	14
40	0 C A	0	18
48	A 5	1	24
56			
64			
72			
80			
88			
196			
104			
112			
120			
128			
136			
144	A 3	8	22



Complementemos a seção:

```
LDS1    LDS

A1      RES 2

A2      TEXT "ERRO"

C0CA    DATA A5, 0CA, A5

A4      RES,1 4

0CA     DATA A5, A2, A3

A5      DATA 27, &FF

A3      DATA,1  0
```

```
        DATA,1  5, 9, 3
A7      RES 3
A8      DATA A2, A3, A9, A5, #LINHA
A9      TEXT "CERTO"
K1      DATA A5, B3
A6      FIN
```

Snapshot 4 - apresenta a história da montagem de A3 até A6.

0			
8	A 1	3	0
16	A 2	7	4
24	C 0 C A	0	8
32	A 4	6	14
40	0 C A	0	18
48	A 5	1	24
56	A 3	5	28
64	A 7	3	32
72	A 8	0	38
80	A 9	7	48
88	K 1	0	54
96	A 6	14	58
104			

112			
120	B 3	8	56
128	L I N H A	9	M
136	~~A 9~~	~~8~~	~~42~~
144	~~A 3~~	~~8~~	~~22~~

0	0 0	24	27	48	C E
2	0 0	26	FF	50	R T
4	E R	28	0 5	52	0 ~~0~~
6	R 0	30	9 3	54	N48
8	N48	32	0 0	56	N120
10	N40	34	0 0	58	
12	N48	36	0 0	60	
14	0 0	38	N16	62	
16	0 0	40	N56	64	
18	N48	42	~~N136~~ N80	66	
20	N16	44	N48	68	
22	~~N144~~ N56	46	N128	70	

ALGORITMO DE MONTAGEM DE UMA DIRETIVA ETIQ.

1. A diretiva existe como EXTERNO LOCAL?
   - Não goto 3.
2. Caminha sobre a lista de referências modificando o endereço em cada nó para referenciar a diretiva. O último nó da lista é aquele que aponta um endereço na zona N.
3. Monta a diretiva na tabela de rótulos.

COMENTÁRIO: Montagem dos nomes referenciados na diretiva

4. Existe algum nome referenciado na diretiva?
   - Não, FIM DO ALGORITMO.
5. O nome referenciado é definido?
   - Sim goto 8
6. O nome referenciado é EXTL?
   - Sim, goto 9.
7. Monta o nome como EXTL na TABELA DE RÓTULOS.
8. Monta endereço de referência ao nome na zona D.
   - Goto 4.
9. Modifica deslocamento do nome para a nova posição de referência na zona D.
10. Monta na zona D o endereço da referência anterior ao nome.
    - Goto 4.

ALTERAÇÃO

A ALTERAÇÃO é considerada como a inserção ou supressão de rótulos. Através de "snapshots" procuraremos enfatizar duas situações distintas:

- Supressão de rótulo referenciado em diretiva ETIQ.
- Supressão de rótulo de diretiva ETIQ.

Seja a seção:

```
LDS1        LDS
A1          RES 2
A2          TEXT "ERRO"
COCA        DATA A5, 0CA, A5
A4          RES, 1 4
0CA         DATA A5, A2, A3
A5          DATA 27, &FF
A3          DATA,1 0
            DATA,1 5, 9, 3
A7          RES 3
A8          DATA A2, A3, A9, A5, #LINHA
A9          TEXT "CERTO"
K1          DATA A5, B3
A6          FIN
```

O Snapshot 5 - mostra a seção antes das alterações.

0					
8	A 1			3	0
16	A 2			7	4
24	C 0	C A		0	8
32	A 4			6	14
40	0 C	A		0	18
48	A 5			1	24
56	A 3			5	28
64	A 7			3	32
72	A 8			0	38
80	A 9			7	48
88	K 1			0	54
96	A 6			14	58
104					
112					
120	B 3			8	56
128	L I	N H	A	9	M
136					
144					
152					
160					

0	0 0	24	27	48	C E
2	0 0	26	FF	50	R T
4	E R	28	0 5	52	0 Ø
6	R 0	30	9 3	54	N48
8	N48	32	0 0	56	N120
10	N40	34	0 0	58	
12	N48	36	0 0	60	
14	0 0	38	N16	62	
16	0 0	40	N56	64	
18	N48	42	N80	66	
20	N16	44	N48	68	
22	N56	46	N128	70	

Verifiquemos o que ocorre ao ser recebido o co - mando:

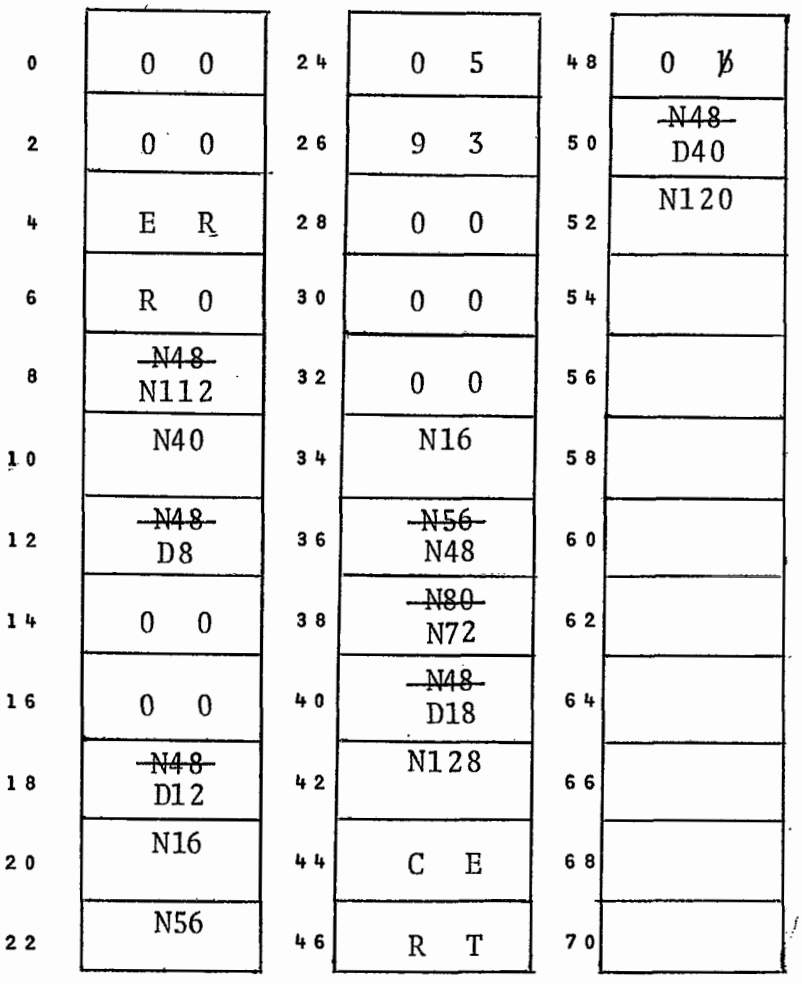
! LDS 4 - 2

Este comando significa suprimir os dois rótulos (0CA e A5) após o rótulo nº 4 (A4) e os dados a eles associados.

Trabalharemos por partes: 1) Supressão de A5, 2) Supressão de 0CA

O Snapshot 6 - mostra a supressão de A5

0			
8	A 1	3	0
16	A 2	7	4
24	C 0 C A	0	8
32	A 4	6	14
40	0 C A	0	18
48	A 3	5	24
56	A 7	3	28
64	A 8	0	34
72	A 9	7	44
80	K 1	0	50
88	A 6	14	54
96			
104			
112	A 5	8	~~8~~ ~~12~~ ~~16~~ ~~40~~ ~~50~~ ~~56~~ ~~52~~
120	B 3	8	
128	L I N H A		
136			
144			
152			
160			



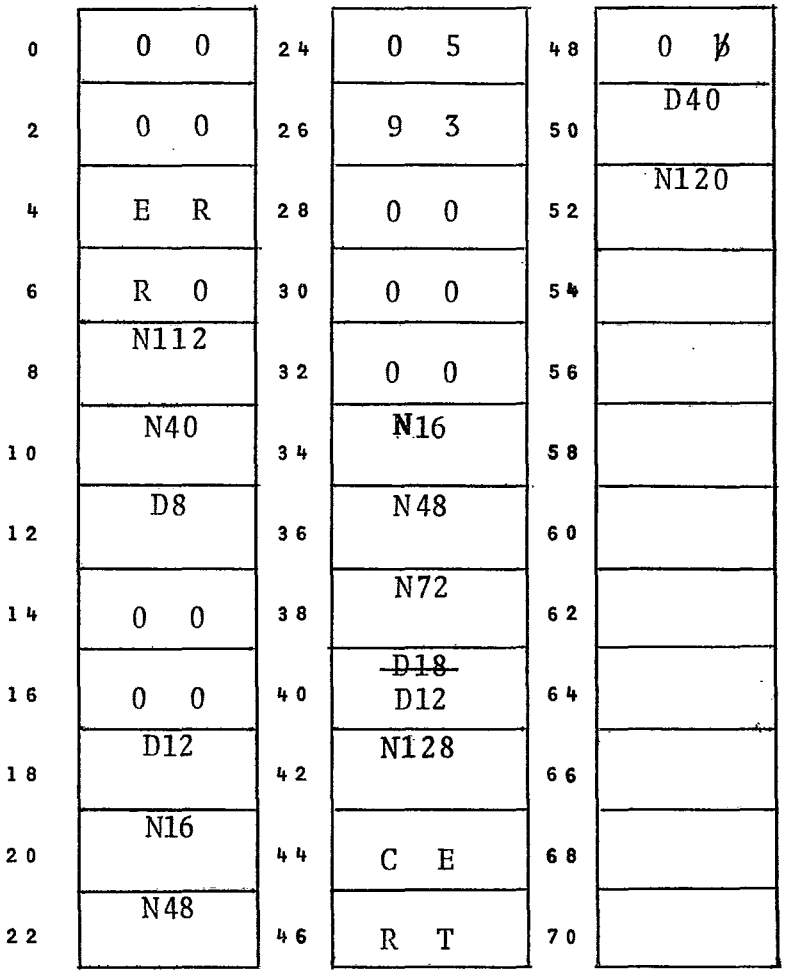
Observemos que a lista de referências a A5 foi reconstruida (A5 voltou a ser EXTL).

O Snapshot 7 mostra a supressão de OCA em 2 estágios:

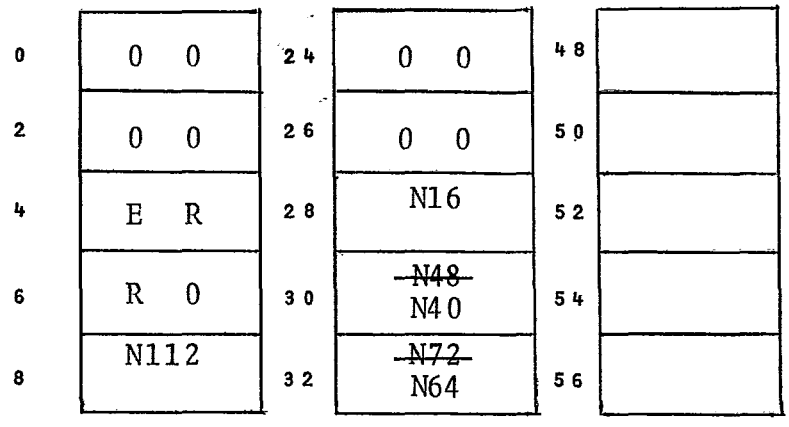
- reorganização dos links de OCA;
- supressão dos dados de OCA.

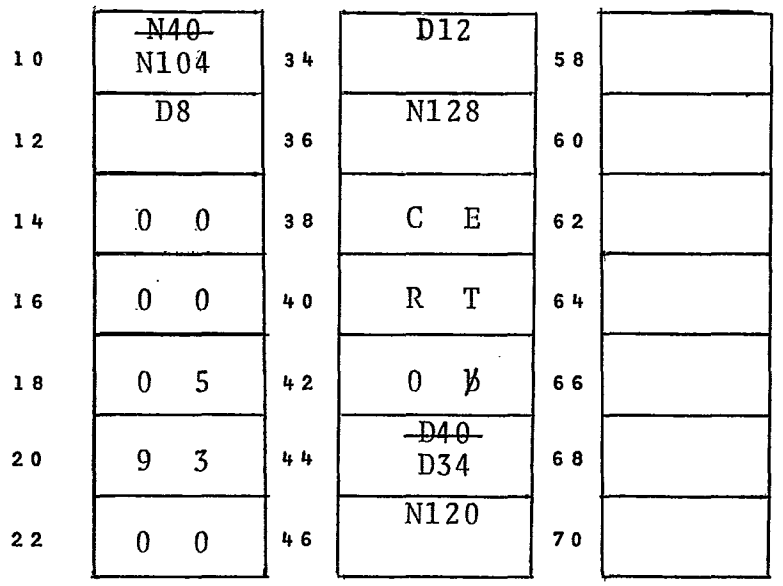
0				
8	A 1		3	0
16	A 2		7	4
24	C 0	C A	0	8
32	A 4		6	14
40	A 3		5	18
48	A 7		3	22
56	A 8		0	28
64	A 9		7	38
72	K 1		0	44
80	A 6		14	48
88				
96				
104	0 C	A	8	10
112	A 5		8	50
120	B 3		8	~~52~~ 46
128	L I	N H A	9	M
136				
144				
152				
160				

REORGANIZAÇÃO DOS LINKS DE OCA



SUPRESSÃO DOS DADOS DE OCA





ALGORITMO DE SUPRESSÃO DE RÓTULOS E DADOS (LOCAL OU COMUM)

1. Se a diretiva a suprimir é ETIQ, verificar, na área de dados de cada diretiva ETIQ após esta, se há referenciais a uma das palavras da área de dados da diretiva a suprimir. Havendo, copiar o conteúdo da palavra referenciada, na palavra que a referencia.

   Executar este passo para cada uma das palavras que serão suprimidas.

2. Compactar a área de dados e a tabela de rótulos (parte definida), alterar o deslocamento de cada rótulo, definido ou EXTL que referencie dados além da área suprimida.

3. Verificar, dentro de cada diretiva ETIQ após a suprimida, se na área de dados existe referência à

área de dados que foi deslocada na supressão. Havendo, subtraí-las do comprimento da área suprimida.

4. Verificar, dentro de todas as diretivas ETIQ, se há referências ao rótulo suprimido. Havendo, torná-lo EXTL e construir a lista de referências ao rótulo.

5. Verificar, dentro de cada diretiva ETIQ, se há referência a rótulos após o rótulo suprimido e que estão entre os rótulos definidos. Havendo, subtrair as referências de 8.

6. Existem mais rótulos para suprimir?
   - Sim goto 1.

7. FIM DO ALGORITMO.

COMENTÁRIO: A supressão é do rótulo de endereço mais alto para o de endereço mais baixo. O passo 1 do algoritmo ocorre na supressão de OCA, como REORGANIZAÇÃO DOS LINKS DE OCA.

LISTAGEM

A LISTAGEM será apresentada apenas quanto aos aspectos da diretiva ETIQ.

A seguinte regra deve ser seguida quando vamos listar uma diretiva ETIQ:

1. Verificar se o conteúdo do endereço, que esta -

mos pesquisando, aponta para a zona N.

- Sim, mover o rótulo para a linha de impressão, goto 1.

2. Avança para o endereço apontado por esta palavra goto 1.

Como exemplo, podemos citar a diretiva A8. Os conteúdos dos endereços 34, 36, 38 e 42 da zona D apontam diretamente um rótulo, enquanto do endereço 40, vamos do endereço 18, deste ao 12 e finalmente ao 8 , que aponta o rótulo desejado (Snapshots).

CONCLUSÃO

O que podemos ver, é que a composição aqui apresentada para a diretiva ETIQ, aumenta em muito a complexidade dos algoritmos de montagem listagem e alteração. No estágio em que estamos, não temos condições de inferir se a complexidade dos algoritmos compensaria a flexibilidade dada pela nova composição da diretiva ETIQ.