UM SISTEMA DE PROGRAMAÇÃO E DEPURAÇÃO CONVERSACIONAL PARA LINGUAGEM TIPO MONTADOR

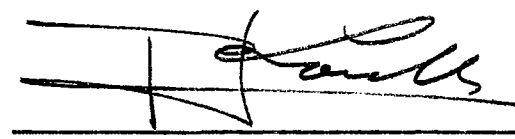
PARTE I

ANALISADOR E MONTADOR

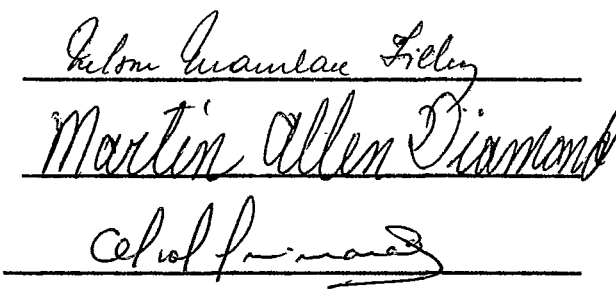
Otacilio José Carollo de Souza

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 À OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA (M.Sc.).

Aprovado por:



Presidente



RIO DE JANEIRO

ESTADO DA GUANABARA - BRASIL

AGOSTO DE 1974

Ã minha esposa por seu

amor e dedicação

AGRADECIMENTOS

Na elaboração deste trabalho, muitos foram os que me auxiliaram; a eles quero expressar meus agradecimentos.

Agradeço ao CNPq, COPPE, BNDE e a UFRGS, pelo apoio financeiro dado durante a realização deste trabalho.

Agradeço, especialmente, ao Prof. Ivo Wolff, Reitor da Universidade Federal do Rio Grande do Sul; ao Prof. Walter Otto Cybis, Superintendente Acadêmico da mesma Universidade; ao Prof. Manoel Luiz Leão, Diretor do Centro de Processamento de Dados da Universidade Federal do Rio Grande do Sul; ao meu orientador , Prof. Pierre Jean Lavelle; Professores, Funcionários e Colegas da COPPE ; aos meus pais, aos meus sogros e aos meus filhos.

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 Languages 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 decompilation 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

Muito se tem feito em termos de linguagem conversacionais de alto nível, para sistemas de grande, médio e pequeno porte, mas nada ou quase nada se fez em termos de linguagems conversacionais de baixo nível.

Sentimos essa necessidade pela existência, na COPPE, do Minicomputador MITRA 15, que possui apenas teletypes como dispositivos de entrada e saída, não possuindo memória auxiliar. Isto torna a tarefa do usuário maçante e cansativa, pois o mesmo gasta muito tempo preparando o computador para executar o seu programa, tempo este perdido, em sua maioria, na leitura de uma fita de papel.

Nosso projeto tem como objetivo dar ao usuário maiores facilidades na depuração dos erros de seu programa e possibilitar a utilização do computador por um maior número de pessoas. Para isso procuramos ter em vista as seguintes características:

a) Ser a linguagem conversacional compatível com a linguagem assembler da máquina;

b) ter a possibilidade de trabalhar com tempo partilhado ("TIME-SHARING");

c) dar, ao usuário, a possibilidade de inserir ou retirar instruções e executar o programa;

d) dar, ao usuário, a possibilidade de listar ou perfurar o seu programa ou parte do mesmo (linguagem fonte);

e) simplicidade de operação;

f) segurança.

CARACTERÍSTICAS

a) Ser a linguagem conversacional compatível com a linguagem assembler da máquina:

Esta característica é a razão de ser do projeto, pois é ela que vai permitir o uso mais intenso da máquina pela maior facilidade que dará, ao usuário, na depuração de seus programas. Para isto foi escolhido um grupo de instruções que apresentaremos quando falarmos sobre a linguagem assembler do MITRA 15.

b) Ter a possibilidade de trabalhar com tempo-partilhado ("TIME-SHARING").

Como o MITRA tem quatro teletypes, o projeto prevê a utilização das mesmas pelos usuários, aproveitando o "OVERLAP" entre o processamento e a entrada e saída. Fica disponível para cada teletype, cinco Kbytes de memória, num total de vinte Kbytes. Dos doze Kbytes restantes, dez pertencem ao sistema projetado e dois ao sistema do MITRA (MONITOR DE BASE. MOB).

Quanto ao tempo disponível para cada usuário, temos duas situações:

1º) quando em tempo de montagem o tempo disponível será o da análise e montagem de uma instrução.

2º) quando em tempo de execução tempo disponível será o da execução de mil instruções do seu programa.

c) Dar, ao usuário, a possibilidade de inserir ou eliminar instruções e executar o programa.

É permitido ao usuário inserir ou eliminar instruções, para lhe dar maior flexibilidade, mas o usuário deve obedecer as regras estabelecidas para isso, e observar os detalhes em relação à compatibilidade do seu programa com relação à execução fora do nosso sistema.

d) Dar, ao usuário, a possibilidade de listar ou perfurar o seu programa ou parte do mesmo.

Isto só será possível em tempo de análise e montagem do programa, pois como não temos uma memória auxiliar (disco) não poderemos guardar o programa fonte, quando do início da execução.

e) Simplicidade de operação.

A operação do sistema é muito simples, tendo apenas os comandos de comunicação do usuário com o sistema e mais o grupo de instruções da linguagem de baixo nível, que foram escolhidas para compor a linguagem conversacional.

f) Segurança.

O sistema toma para si a tarefa de deixar, ao usuário, apenas a área que lhe foi reservada não possibilitando a invasão de outras áreas e nem a modificação do processo em tempo de execução.

C A P Í T U L O II

A LINGUAGEM ASSEMBLER DO MITRA 15

ESTRUTURA MODULAR

A estrutura da linguagem assembler do MITRA 15 é bem diferente das linguagem assembler de outras máquinas, apresentando-se de forma modular, como veremos a seguir. A modularidade com que ela se apresenta nos traz vantagens como:

- Facilitar a especificação de um sistema de maneira modular;
- facilitar a programação do sistema de maneira paralela por vários programadores;
- permitir a utilização de seções idênticas de um sistema para outro;
- facilitar a depuração.

Um programa, na linguagem assembler do MITRA, é composto de várias seções, sendo que é a seção que dá à linguagem, a estrutura modular.

Temos dois tipos de seções num programa:

a) Seção de dados comuns (CDS: Common Data Section), que tem como função, a reserva de área e definição de dados que podem ser usados por todas as outras seções. Existe apenas uma seção de dados comuns para cada programa.

b) Seção de Programa. Este tipo de seção tem por função a reserva de áreas e a definição de dados locais, isto é, os dados da própria seção. E a função de execução do programa, que é feita por uma série de instruções que tem, por objeto, o tratamento dos dados, tanto locais como comuns.

A seção de programa é composta por um segmento de dados locais (LDS: Local Data Segment) e um segmento de programa executável (LPS: Local Program Segment).

Tem-se acesso à CDS a partir de todo o programa. Em particular, tem-se acesso à CDS a partir de uma LPS no modo geral (Direto, Indireto, ou Indireto Indexado). Os nomes de dados e rótulos definidos numa LDS são locais à seção. Eles podem, entretanto, ser referenciados na CDS.

BASES DAS SEÇÕES E SEGMENTOS

Base Geral G

A Base Geral G está associada de maneira biunívoca ao programa; é a base implícita para todos aqueles endereços referenciados pelo programa. Ela é adicionada automaticamente pela micromáquina ao endereço definido nas instruções.

Base L

A Base L é a base local implícita dos dados locais, associada a um segmento de dados locais.

Base P.

A Base P é a base do programa, associada a um segmento de programa. A Base P contém, inicialmente, o endereço da primeira instrução executável da seção, pois constitui o contador em curso da execução da seção. O valor efetivo das Bases L e P de uma seção podem ser ignorados pelo programador, no momento da programação. Eles são definidos automaticamente pelo editor de linhas em valor relativo à Base Geral do programa e guardados na PRT (Program Relocation Table) do programa.

CONSEQUÊNCIA DA MODULARIDADE DOS PROGRAMAS SOBRE O MITRA 15

Do ponto de vista do HARDWARE, a modularidade implica na existência de instruções especiais de chamada e de retorno da seção. Para que isto fosse possível criou-se, na linguagem, as diretivas ditas de seccionamento.

CDS : Common Data Section

LDS : Local Data Segment

LPS : Local Program Segment

FIN : Fin de Segment ou de Section

IDS : Indirect Data Segment

Cada segmento termina por uma diretiva FIN. Designamos por "Módulo de Programa" (ou Módulo D'Assemblage"), o resultado de uma montagem. Cada módulo deve terminar, obrigatoriamente, por uma diretiva END.

Exemplo 1 : ORGANIZAÇÃO USUAL DE UM PROGRAMA

```
COMMON      CDS
TWB         RES     16
C1          DATA    1
            FIN

LOCAL       LDS
            RES     2
C2          DATA    &F0
            FIN

SPROG       LPS     LOCAL
DEB         LDA     =3
            AND     C2
            RTS
            FIN     DEB

LOCP        LDS
            RES     2
U           DATA,1  28
V           DATA,1  31
TAB         DATA    ATAB
ATAB        RES     1024
            FIN

PRINC       LPS     LOCP
INIT        LDA     U
            ADD     #C1
            STA     TAB
            CLS     SPROG
            CSV     M:EXIT
            FIN     INIT
            END     PRINC
```

Seção de dados comuns

Segmento de dados locais

Segmento de programa

SEÇÃO 1

Segmento de dados locais

Segmento de programa

SEÇÃO 2

Marca de fim de arquivo (% EOD sobre o cartão ou fita perfurada).

LINGUAGEM CONVERSACIONAL.

Vamos, agora, apresentar o Sub-Conjunto da linguagem assembler do MITRA, que nos propomos a interpretar. Alertamos que daremos apenas o conjunto a interpretar e que as características da linguagem assembler constam no Manual de Apresentação do MITRA 15, e seria maçante colocarmos, aqui, a sua descrição completa, uma vez que no Analisador iremos mostrar em detalhe cada uma das diretivas e instruções interpretadas e o que foi preciso restringir para que o projeto fosse exequível.

Diretivas de Seccionamento.

CDS : Seção de Dados Comuns.

LDS : Segmento de Dados Locais.

LPS : Segmento de Programa Executável.

FIN : Fim de Segmento.

END : Fim de Módulo.

IDS : Não Interpretada.

Diretivas do Montador.

As diretivas do montador só poderão ocorrer na seção de dados comuns ou no segmento de dados locais. São elas :

RES : Reserva uma zona de memória em palavras e em endereço de palavra (PAR).

RES,1 : Reserva uma zona de memória em bytes e em endereço de byte.

DATA : Definição de dados em palavras e em endereço de palavras.

DATA,1 : Definição de dados em bytes e em endereço de byte.

TEXT : Definição de uma cadeia de caracteres em endereço de bytes.

As diretivas do montador não interpretadas são BND, BASE, EQU, GOTO, DO, PAGE, GEN, DEF, REF.

Instruções do Montador.

As instruções só poderão ocorrer no segmento de programa. Nos limitaremos, aqui, a apresentar os mnemônicos com a sua função. Quanto aos tipos de endereçamentos de cada um, os descreveremos no Analisador.

Carregamento e Armazenamento.

LBL	:	(LOAD BYTE LEFT)	Byte
SBL	:	(STORE BYTE LEFT)	
LBR	:	(LOAD BYTE RIGHT)	
SBR	:	(STORE BYTE RIGHT)	
LBX	:	(LOAD BYTE X)	
LDA	:	(LOAD A)	Palavra
STA	:	(STORE A)	
LDE	:	(LOAD E)	
STE	:	(STORE E)	
LDX	:	(LOAD X)	
STX	:	(STORE X)	
LDR	:	(NÃO INTERPRETADA)	
STR	:	(NÃO INTERPRETADA)	

LEA : (LOAD EFFECTIVE ADRESS)
SPA : (NÃO INTERPRETADA)
STS : (STORE SELECTIVE)
} Palavra

DLD : (DOUBLE LOAD)
DST : (DOUBLE STORE)
} Dupla Palavra

Aritmética de Vírgula Fixa.

ADD : (ADDITION)
ADM : (ADDITION MEMORY)
SUB : (SUBSTRACTION)
MUL : (MULTIPLICATION)
DIV : (DIVISION)

Operações Lógicas.

IOR : (INCLUSIVE OR)
EOR : (EXCLUSIVE OR)
AND : (AND)
CMP : (COMPARE)

Incrementação e Decrementação de Registros.

ICX : (INCREMENT X)
DCX : (DECREMENT X)
ICL : (NÃO INTERPRETADA)
DCL : (NÃO INTERPRETADA)

Shifts.

SHR : (NÃO INTERPRETADA, MAS TODAS INSTRUÇÕES DE-

LA DERIVADAS, O SÃO)

SLLS : (SHIFT LEFT LOGICAL SIMPLE)

SRCS : (SHIFT RIGHT CIRCULAR SIMPLE)

SAD : (SHIFT ARITHMETIC DOUBLE)

SLCD : (SHIFT LEFT CIRCULAR DOUBLE)

SLCS : (SHIFT LEFT CIRCULAR SIMPLE)

SAS : (SHIFT ARITHMETIC SIMPLE)

SRLS : (SHIFT RIGHT LOGICAL SIMPLE)

SRCD : (SHIFT RIGHT CIRCULAR DOUBLE)

SHC : (NÃO INTERPRETADA, MAS TODAS INSTRUÇÕES DE-
LA DERIVADA, O SÃO)

SLLD : (SHIFT LEFT LOGICAL DOBLE)

SRLD : (SHIFT RIGHT LOGICAL DOUBLE)

PTY : (PARITY)

NLZ : (NORMALIZATION)

Operação entre Registros.

SRG : (NÃO INTERPRETADA, MAS TODAS INSTRUÇÕES DE-
LA DERIVADAS, O SÃO)

XAE : (EXCHANGE A AND E)

XAX : (EXCHANGE A AND X)

XEX : (EXCAHNGE E AND X)

XAA : (EXCHANGE LEFT BYTE OF A AND RIGHT BYTE OF A)

CCE : (COPY COMPLEMENT E)

ACE : (ADD CARRY AND E)

CCA : (COPY COMPLEMENT A)

AEE : (A EXCLUSIVE OR WITH E)

CNX : (COPY NEGATIVE X)

AIE : (A INCLUSIVE OR WITH E)

AEE : (A AND E)

LNE : (LOAD NEGATIVE E)

CNA : (COPY NEGATIVE A)

CHX : (COPY HALF X)

Aritmética de Vírgula Flutuante.

FAD : (NÃO INTERPRETADA)

FSU : (NÃO INTERPRETADA)

FMU : (NÃO INTERPRETADA)

FDU : (NÃO INTERPRETADA)

Tratamento de Cadeias de Bytes.

MVS : (MOVE BYTE STRING)

CPS : (COMPARE STRING)

TRS : (TRANSLATE STRING)

Desvios (Branch).

BRU : (BRANCH UNCONDITIONAL)

BRX : (BRANCH WITH INDEX)

BCT : (BRANCH IF CARRY TRUE)

BOT : (BRANCH IF OVERFLOW TRUE)

BCF : (BRANCH IF CARRY FALSE)

BOF : (BRANCH IF OVERFLOW FALSE)

BAZ : (BRANCH IF A EQUAL TO ZERO)

BAN : (BRANCH IF A NEGATIVE)

BE : (BRANCH IF EQUAL TO)

BZ : (BRANCH IF EQUAL TO ZERO)

BL : (BRANCH IF LESS THAN)

BLZ : (BRANCH IF LESS THAN ZERO)

BNE : (BRANCH ON NOT EQUAL TO)

BNZ : (BRANCH IF NOT EQUAL TO ZERO)

BGE : (BRANCH IF GREATER THAN OR EQUAL TO)

BPZ : (BRANCH IF POSITIVE OR EQUAL TO ZERO)

Desvios do Sistema.

CLS : (CALL SECTION)

RTS : (RETURN SECTION)

CSV : (CALL SUPERVISOR)

RSV : (NÃO INTERPRETADA)

DIT : (NÃO INTERPRETADA)

DITR : (NÃO INTERPRETADA)

Instruções de Comando.

TES : (NÃO INTERPRETADA)

STM : (NÃO INTERPRETADA)

CLM : (NÃO INTERPRETADA)

RD : (NÃO INTERPRETADA)

WD : (NÃO INTERPRETADA)

LDP : (NÃO INTERPRETADA)

FORMATO DE UMA LINHA NA LINGUAGEM FONTE.

Uma linha da linguagem fonte comporta no máximo qua-

tro zonas:

A Zona de Rótulo:

Começa, obrigatoriamente, na coluna 1, contendo um nome de 1 até 6 caracteres alfanuméricos, sendo que o primeiro deve ser uma letra e termina por uma coluna branca.

A Zona de Comando:

Começa sobre a primeira coluna não branca, que segue a zona de rótulo (ou a primeira coluna não branca, que não a coluna 1 se a zona de rótulo não é utilizada) e termina por uma coluna branca. Essa zona contém uma diretiva ou uma instrução.

A Zona de Argumento:

Começa sobre a primeira coluna não branca, que segue a zona de comando e termina por uma coluna branca, se a primeira coluna, após a zona de comando, contém o caráter especial "*" a zona de argumento é considerada como vazia.

A Zona de argumento não pode exceder a coluna 72 (MITRA 2).

A Zona de Comentário:

Começa sobre a primeira coluna, que segue o caráter especial "*".

Linha de Comentário.

Uma linha de comentário é uma linha que tem o caráter especial "*" na coluna 1.

Tanto as linhas de comentários como as zonas de comentário serão ignoradas e não serão reproduzidas na listagem ou perfuração do programa fonte.

Linhas Virgens.

As linhas virgens não serão aceitas.

C A P Í T U L O III

ESTRUTURA DE UM PROGRAMA NO SISTEMA PROJETADO.

No nosso sistema, a única organização permitida para um programa, é a apresentada no Exemplo 1, isto é, cada programa terá uma seção de dados comuns e cada segmento de programa terá o seu segmento de dados locais (Organização Usual). As outras organizações apresentadas, na página 3-4 do Manual de Apresentação do MITRA 15, não serão aceitas. Pelo motivo de poder ter várias seções chamando umas às outras, é que é necessário a PRT (Program Relocation Table) para poder, a execução, passar de uma seção para a outra.

O grande problema que enfrentamos, foi o de projetar uma estrutura que permita a decompilação do programa, quando em tempo de montagem, pois o MITRA não possui memória auxiliar (disco). Falamos em decompilação porque para que seja possível listar ou perfurar o programa fonte, é necessário ter uma estrutura que nos dê todas as informações para esta volta, já que cada instrução, ao entrar no sistema, é colocada em termos da linguagem do interpretador.

Passaremos, agora, a descrever como é a estrutura de um programa. Para a clareza da explanação, colocamos um esquema da estrutura no fim do capítulo.

DESCRIÇÃO DA PRT.

A PRT é uma tabela que nos dá informações sobre a

CDS e a diversas seções que o programa pode ter, além de algumas informações gerais. Ela deve começar sempre em fronteira de palavra, e seu tamanho é igual a (nº de duplas LDS/LPS previstas + 1 para CDS + 1 para o sistema) x 20 bytes. Cada vinte bytes da PRT contém informações sobre uma seção ou a CDS.

Os primeiros vinte bytes correspondem às informações sobre a CDS e gerais, conforme o esquema.

BYTES 0 - 5 : Nome da CDS definido na diretiva CDS.

BYTES 6 - 7 : Apontador da última seção do programa inicializado pela diretiva END.

BYTES 8 - 9 : Usados pelo Interpretador.

BYTES 10 - 11 : Caracteriza se o programa está sendo alterado ou listado.

BYTES 12 - 13 : Apontador da CDS. Contém o endereço inicial da tabela de nomes da CDS.

BYTES 14 - 15 : Não usados.

BYTES 16 - 17 : Apontador da 1ª seção. Contém o endereço da 1ª seção executável do programa na PRT. Aponta para o nome da LDS.

BYTES 18 - 19 : Apontador de dados da CDS. Contém o endereço inicial da área de dados da CDS.

Os vinte bytes correspondentes a cada seção, contém as seguintes informações:

BYTES 0 - 5 : Nome da LDS definido na diretiva LDS.

BYTES 6 - 11 : Nome da LPS definido na diretiva LPS.

BYTES 12 - 13 : Apontador da LDS. Contém o endereço inicial da

tabela de nomes da LDS.

BYTES 14 - 15 : Apontador da LPS. Contém o endereço inicial da LPS.

BYTES 16 - 17 : Primeira instrução. Contém o endereço da primeira instrução executável da LPS.

BYTES 18 - 19 : Apontador de dados. Contém o endereço inicial da área de dados da LDS.

OBSERVAÇÃO : Dos últimos vinte bytes da PRT, apenas os bytes 12 - 13 serão usados, contendo o apontador da tabela de nomes da LDS, que existiria se tivéssemos mais uma seção. Isto é feito para possibilitar o cálculo do tamanho da LPS da última seção, usando o mesmo procedimento das LPS anteriores.

DESCRIÇÃO DA TABELA DE NOMES.

A tabela de nomes contém informações sobre a CDS ou LDS e a LPS de uma seção.

Está dividida em três partes distintas:

1) Área de nomes de dados definidos.
2) Área de nomes de dados ainda não definidos (quando da montagem), rótulos de instrução e nomes de seções.
3) Área de dados.

Os primeiros oito bytes da tabela de nomes, têm uma utilização especial, pois são apontadores que indicam a próxima po

sição disponível nas três áreas e na tabela de LPS, se estivermos numa seção.

BYTE 0 - 1 : Bytes CDS (ou bytes LDS). Contém o endereço da próxima posição disponível na área de dados.

BYTE 2 - 3 : Varaux da seção (CDS ou LDS). Contém o endereço da próxima posição disponível na área de nomes de dados definidos.

BYTE 4 - 5 : Apontador de rótulo (CDS ou LDS). Contém o endereço da próxima posição disponível na área de nomes de dados ainda não definidos (quando da montagem), rótulos de instrução e nomes de seções.

BYTE 6 - 7 : Apontador da LPS. Contém o endereço da próxima posição disponível na LPS (este apontador não existe para a tabela de nomes da CDS).

As posições seguintes da tabela de nomes, que não as da área de dados compõem-se de oito bytes assim dispostos:

BYTES 0 - 5 : Nome do dado. Contém o nome do dado ou o nome do rótulo ou o nome de uma seção chamada nessa seção.

BYTE 6 : Contém, nos quatro bits de mais alta ordem, o tipo de dado. Os quatro bits de mais baixa ordem contém a parte mais significativa do deslocamento do dado (definiremos abaixo). Os valores que o tipo de dado pode tomar, estão definidos no Analisador, na parte referente à tabela das diretivas.

BYTE 7 : Deslocamento. Contém a parte menos significativa do deslocamento do dado. O mesmo é definido como

a distância a que se encontra o dado ligado aquele nome de dados, em relação ao apontador da área de dados.

OBSERVAÇÃO : A área de nomes de dados definidos inicia no endereço indicado pelo apontador da LDS (ou CDS) mais oito bytes e se desenvolve no sentido crescente dos endereços.

A área de nome de dados, ainda não definidos (quando da montagem), rótulos de instrução e nomes de seções inicia no endereço indicado pelo apontador de dados menos oito bytes e se desenvolve no sentido decrescente dos endereços. NOTE que o limite extremo das duas áreas podem se encontrar. No deslocamento (área de rótulos e seções), teremos o número do rótulo ou o número da seção.

A tabela de nomes possui, ainda, a área de dados , onde ficam os dados propriamente dito, sem formato fixo.

A área de dados cresce com o endereço da memória e inicia no endereço indicado pelo apontador de dados da LDS (ou CDS).

DESCRIÇÃO DA LPS.

A LPS (começa em endereço PAR), é a tabela que vai conter as instruções para o Interpretador. Cada posição da tabela tem 4 bytes. Os quatro primeiros têm uma utilização especial ,

pois não contém uma instrução, mas as seguintes informações:

BYTE 0 : Contém o número de seções chamadas nesta seção.

BYTE 1 : Contém o número de rótulos usados na LPS desta seção.

BYTE 2-3 : Não são usados.

As próximas posições da LPS correspondem às instruções e contém as seguintes informações:

BYTE 0 : NUM ROT - o número do rótulo contém um número de 0 (zero) a 255.

O zero caracteriza a não existência de rótulo para a instrução.

Um número de 1(um) a 255 caracteriza a existência de rótulo para a instrução. Este número será igual ao número que estiver no byte 7 (deslocamento) do rótulo (nome do dado) correspondente na área de nomes ainda não definidos, rótulos e nomes de seção.

BYTE 1 : COD INST - código da instrução é um número de 2 a 152, que indica a posição na tabela de mnemônicos da instrução a interpretar.

BYTE 2 : TIPO END - tipo de endereçamento. Contém um dos códigos de modo de endereçamento.

BYTE 3 : NUM ARG - número do argumento. Depende do tipo de endereçamento e da instrução (mnemônico).

Será zero se a instrução não tiver argumento. Será um dado imediato se o tipo de endereçamento assim o exigir ou será o número (posição na tabela de nome de dados ou rótulo.

LAY-OUT DE UM PROGRAMA

P R T						
NOME DA CDS	APONT. ULT. SEC.		APONT. da CDS		APONT. 1ª SEC.	APONT. DADOS
NOME DA LDS1	NOME DA LPS1		APONT. LDS1	APONT. LPS1	PRIMEIRA INST.	APONT. DADOS
NOME DA LDSN	NOME DA LPSN		APONT. LDSN	APONT. LPSN	PRIMEIRA INST.	APONT. DADOS
			APONT. LDSN+1			

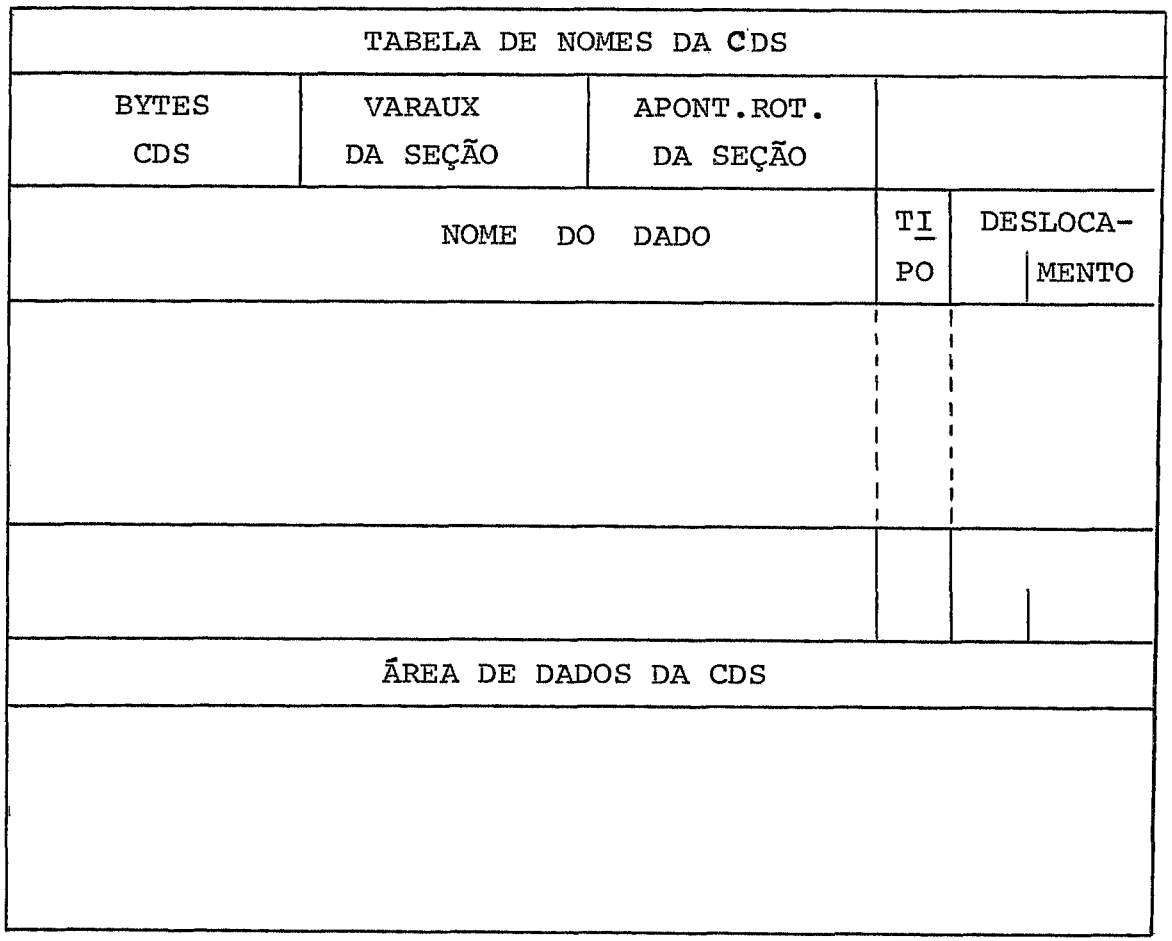


TABELA DE NOMES DA CDS
BYTES CDS
VARAUX DA SEÇÃO
APONT.ROT. DA SEÇÃO
NOME DO DADO
TIPO
DESLOCA-MENTO
ÁREA DE DADOS DA CDS

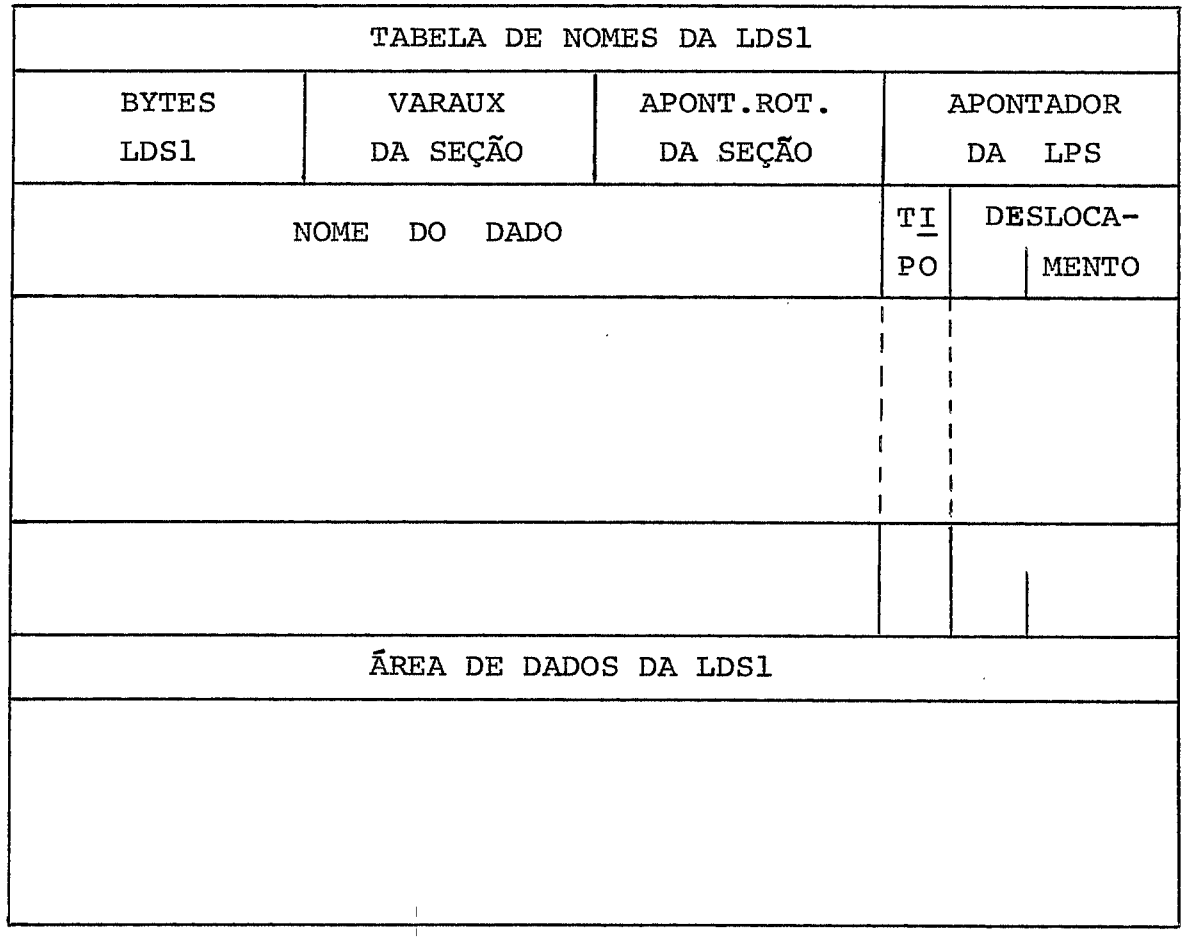
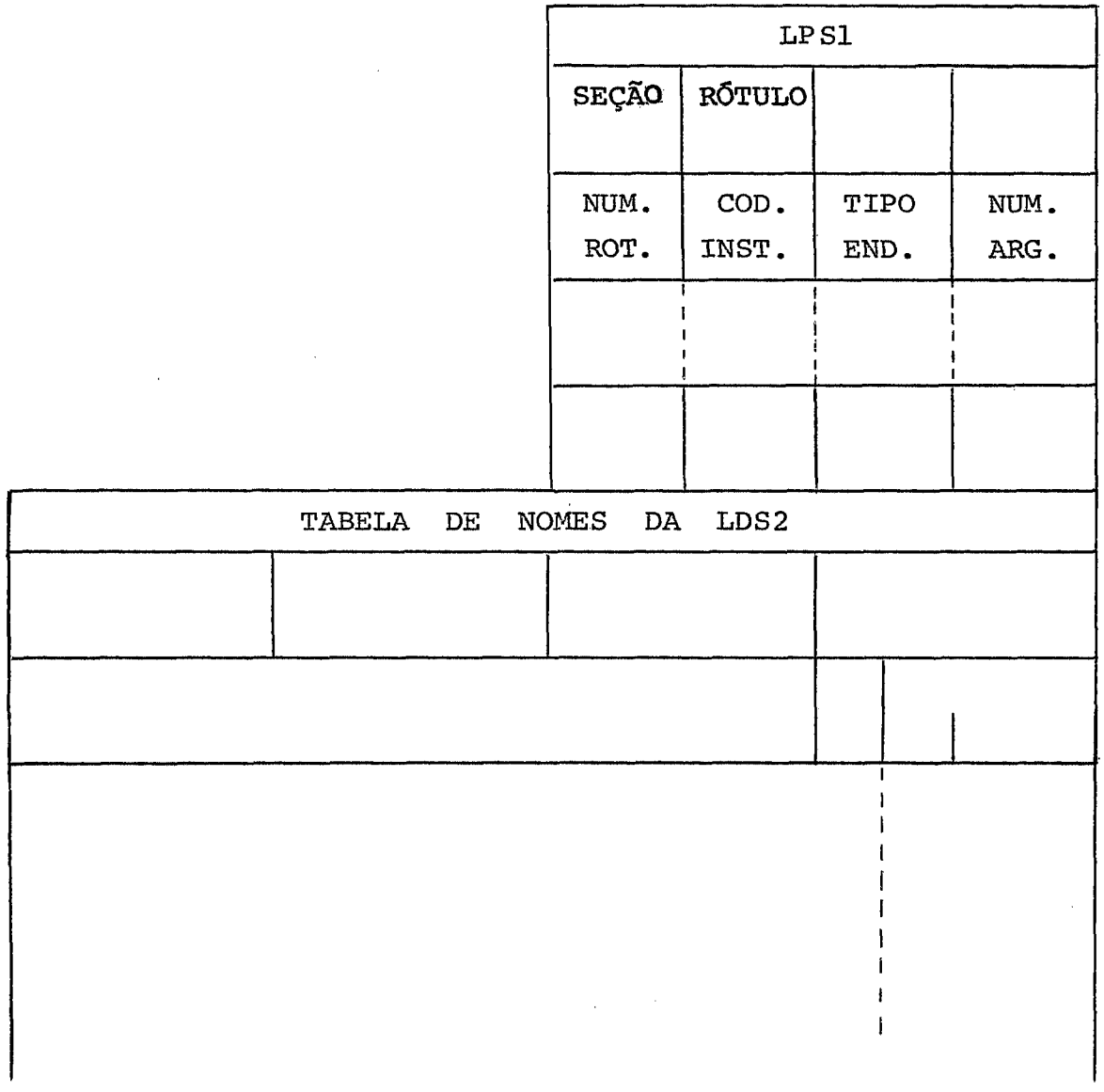


TABELA DE NOMES DA LDS1
BYTES LDS1
VARAUX DA SEÇÃO
APONT.ROT. DA SEÇÃO
APONTADOR DA LPS
NOME DO DADO
TIPO
DESLOCA-MENTO
ÁREA DE DADOS DA LDS1


LPS1
SEÇÃO
RÓTULO
NUM. ROT.
COD. INST.
TIPO END.
NUM. ARG.
TABELA DE NOMES DA LDS2

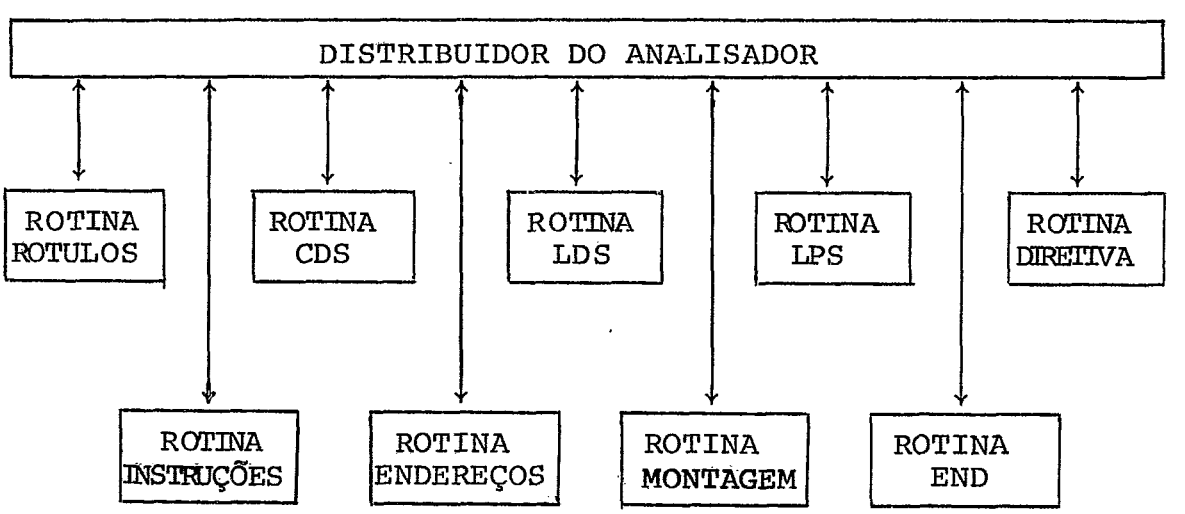
C A P Í T U L O IV

ANALISADOR

O Analisador é uma subrotina do Supervisor. Ele recebe uma linha do Supervisor, que passa a ser analisada como uma diretiva de seccionamento, ou uma diretiva do montador, ou uma instrução, conforme descrição da "Variável do Contexto STATUS" (vide tese Um Sistema de Programação e Depuração Conversacional para Linguagem Tipo Montador PARTE II Supervisor e Interpretador). Se a diretiva ou instrução estiver correta, passa então para a fase de montagem.

É importante anotar, aqui, que o Analisador somente executa o que o Supervisor ordenar, através do STATUS, e que para o Analisador, é transparente o problema de saber sobre qual o programa ele está trabalhando.

O Analisador é composto de várias partes (seções), como vemos na figura abaixo, que descreveremos a seguir:



DISTRIBUIDOR DO ANALISADOR

O distribuidor do Analisador tem como função, a comunicação com o Supervisor e a distribuição das tarefas para as rotinas a ele ligadas.

A variável do contexto STATUS e quem transmite as informações para esta distribuição.

É o distribuidor do Analisador que verifica a existência de rótulo analisando a primeira coluna da linha.

ROTINA DE RÓTULOS

Definimos como rótulo, todos os nomes que aparecem na zona de rótulo.

A função da rotina de rótulos é verificar a validade do rótulo para as diretivas de segmentação, diretivas do montador e instruções.

Além disso, o rótulo é preparado para a fase de montagem se estiver correto, e dentro das normas para cada um dos casos.

ROTINA DA CDS

Para que possamos dar a função da rotina da CDS é necessário apresentarmos a diretiva CDS.

RÓTULO	COMANDO	ARGUMENTO	COMENTÁRIO
<NOME>	CDS		<COMENT>

<COMENT> é constituido de informações para a criação da PRT e da tabela de nomes da seção de dados comuns (CDS). As informações devem vir logo após a diretiva CDS, conforme o esquema a baixo:(*)

RÓTULO CDSbb*bbXXX,YYY,ZZZZ

XXX - é um número que define o número máximo de seções do programa, até um máximo de 127 seções (inclusive a CDS).

YYY - é um número que define o número máximo de rótulos da seção de dados comuns do programa, até um máximo de 255 rótulos. Ele é igual ao número de rótulos à esquerda de cada diretiva, mais o número de rótulos à direita das diretivas DATA da seção.

ZZZZ - é um número que define o tamanho, em bytes, da área de dados da seção de dados comuns do programa, até um máximo de 4095 bytes.

Exemplos:

```
ROT   CDSbb*bb10,30,200
X CDSbb*bb25,100,1000
CDS  CDSbb*bb3,90,810
```

A função da rotina da CDS é analisar, sintaticamente, a diretiva e reservar a área para a PRT e a tabela de nomes da seção de dados comuns através da criação do apontador da CDS, do a-

(*) O <COMENT> não existe na linguagem assembler do MITRA 15 e foi desenvolvido de modo a ser compatível com ela e fornecendo informações necessárias ao nosso sistema.

pontador de dados, do apontador da LDS1, com as informações dadas pelo <COMENT> (XXX,YYY,ZZZZ), e colocar, na primeira posição da PRT o <NOME> da diretiva, o STATUS é posicionado para receber a seguir, as diretivas do montador.

ROTINA DA LDS.

Para que possamos dar a função da rotina da LDS é necessário apresentarmos a diretiva LDS.

RÓTULO	COMANDO	ARGUMENTO	COMENTÁRIO
<NOME>	LDS		<COMENT>

<COMENT> é constituido de informações para a criação da tabela de nomes da seção. As informações devem vir logo após a diretiva LDS, conforme o esquema abaixo:

RÓTULO LDS␣*␣YYY,ZZZZ

YYY - é um número que define o número máximo de rótulos da seção, até um máximo de 255 rótulos. Ele é igual ao número de rótulos à esquerda das diretivas da seção, mais o número de rótulos à esquerda das instruções desta seção, mais o número de rótulos à direita das diretivas DATA desta seção, mais o número de seções distintas referenciadas pela instrução CLS na seção.

ZZZZ - é um número que define o tamanho, em bytes, da área de dados da seção até um máximo de 4095 bytes. Os ró

tulos à direita da diretiva DATA, correspondem a **uma palavra.**

Exemplos:

```
ROT1   LDSØ*Ø35,200
X1 LDSØ*Ø100,1000
LDS1   LDSØ*Ø100,1000
```

A função da rotina da LDS é analisar sintaticamente a diretiva e reservar a área para a tabela de nomes da seção, através da criação do apontador da LPS (ajustado a endereço de palavra) do apontador de dados e do apontador da LDS seguinte, com as informações dadas pelo <COMENT> (YYY,ZZZZ), e colocar, na PRT, o <NOME> da diretiva.

O STATUS é posicionado para receber a seguir, as diretivas do montador.

ROTINA DA LPS.

Para que possamos dar a função da rotina LDS, é necessário apresentarmos a diretiva LPS.

RÓTULO	COMANDO	ARGUMENTO	COMENTÁRIO
<NOME1>	LPS	<NOME2>	<COMENT>

<NOME1> constitui o nome da LPS e da seção; é uma definição externa implícita. O nome referenciado, dentro de um CALL SECTION, constitui a referência externa implícita correspondente.

<NOME2> é o nome da LDS associada.

<COMENT> é constituido de informações para a criação do segmento de programa executável. As informações devem vir logo após a diretiva LDS conforme o esquema abaixo:

RÓTULO LPS ROTLDS/0*/0YYY

YYY - é um número que define o número máximo de instruções da seção até um máximo de 255. As diretivas de segmentação FIN e END também são contadas.

Exemplos:

SEÇÃO1 LPS ROT1/0*/0100

SEC LPS X1/0*/030

LPS1 LPS LDS1/0*/050

A função da rotina da LPS é analisar, sintaticamente, a diretiva e reservar a área para a LPS (segmento de programa executável), através da criação do apontador da tabela de nomes da próxima seção, com as informações dadas pelo <COMENT> (YYY) e colocar, na PRT, o <NOME1> e verificar se <NOME2> é o nome da LDS definida anteriormente (LDS associada).

O STATUS é posicionado para receber, a seguir, as instruções ou a diretiva FIN.

ROTINA DAS DIRETIVAS.

Para que possamos dar a função da rotina das direti - vas, é necessário apresentarmos cada uma delas.

Diretiva RES

RÓTULO	COMANDO	ARGUMENTO
<RÓTULO>	RES	<CONSTANTE DECIMAL INTEIRA>
TOTO	RES	3
	RES	5

Diretiva TEXT

RÓTULO	COMANDO	ARGUMENTO
<RÓTULO>	TEXT	"<CADEIA DE CARACTERES>"
TOTO	TEXT	" CADEIA DE CARACTERES "

OBSERVAÇÃO: Os caracteres ", LF e RC não devem entrar na cadeia de caracteres.

Diretiva DATA,1

RÓTULO	COMANDO	ARGUMENTO
<RÓTULO>	DATA,1	<CONSTANTE>[,<CONSTANTE>]...
TOTO	DATA,1	7,&8E,5
	DATA,1	&FF,15

Diretiva DATA

A diretiva DATA pode ter dois tipos de argumentos

que não podem aparecer juntos. Apresentaremos os dois (TIPO DATA e TIPO ETIQUETA):

TIPO DATA

RÓTULO	COMANDO	ARGUMENTO
<RÓTULO>	DATA	[-]<CONST.>[,[-]<CONST.>]...
TOTO	DATA	8,-9,&FFA2
	DATA	-1

TIPO ETIQUETA

RÓTULO	COMANDO	ARGUMENTO
<RÓTULO>	DATA	[#]<ETIQUETA>
TOTO	DATA	TATA
TOCA	DATA	#ACC

OBSERVAÇÃO: Não pode ocorrer o mesmo nome de dado mais de uma vez no lado direito de uma diretiva **TIPO** ETIQUETA.

Diretiva RES,1

RÓTULO	COMANDO	ARGUMENTO
<RÓTULO>	RES,1	<CONSTANTE **DECIMAL** INTEIRA>
TATA	RES,1	6
	RES,1	9

Diretiva FIN

RÓTULO	COMANDO	ARGUMENTO
<RÓTULO>	FIN	
FIM1	FIN	

OBSERVAÇÕES: O <RÓTULO> só não é necessário se a diretiva do montador, anterior, era igual, desde que não fosse a diretiva DATA TIPO ETIQUETA. Sempre que houver mudança de tipo de diretivas (inclusive da diretiva DATA TIPO DATA; para DATA TIPO ETIQUETA), a diretiva entrante deve ter rótulo.

A <ETIQUETA> pode ser qualquer <RÓTULO> definido numa diretiva do montador ou <RÓTULO> das instruções.

A função da rotina das diretivas é analisar, sintaticamente as diretivas e montá-las na tabela de nomes da LDS (ou CDS), controlar para que não aconteçam invasões (estouros) das áreas da tabela de nomes e da LPS (ou da primeira LDS se estivermos numa CDS). Posicionar o STATUS para receber a diretiva de segmentação LDS se estivermos na diretiva FIN da seção de dados comuns, ou à diretiva de segmentação LPS se estivermos na diretiva FIN do segmento de dados locais.

ROTINA DAS INSTRUÇÕES

A rotina das instruções tem duas fases distintas:

PRIMEIRA: Verifica se o mnemônico da instrução apresentada corresponde a algum da tabela de mnemônicos (menos o FIN e o END). Se isso ocorre, ela coloca o número de entrada na tabela de mnemônicos no byte correspondente ao código da instrução, que está sendo montada no campo auxiliar. Após, volta ao distribuidor do Analisador,que entrega o controle para a rotina de endereços. Se o mnemônico não foi encontrado na tabela de mnemônicos, passa para a segunda fase .

SEGUNDA: Verifica se o mnemônico é o FIN (de LPS) que, só pode ter o formato abaixo:

RÓTULO	COMANDO	ARGUMENTO
	FIN	<ETIQUETA2>
	FIN	INÍCIO

<ETIQUETA2> é o <RÓTULO> da primeira instrução a ser executada da seção. Se for FIN, a diretiva é analisada sob os aspectos mostrados acima e é colocado na PRT , na palavra "PRIMEIRA INSTRUÇÃO" da seção correspondente, o endereço da primeira instrução executável. O# STATUS é posicionado para receber a diretiva END, e volta a distribuidor do Analisador com informação para que o controle passe para a rotina de montagem.

ROTINA DOS ENDEREÇOS.

A função da rotina dos endereços é verificar se o ti

po de endereçamento usado pelo programador é válido para aquele mnemônico.

Analisando a zona de argumento da instrução, obtém-se o tipo de endereçamento usado pelo programador. Como o mnemônico já foi analisado, seu código serve de entrada na "Tabela de Apontadores para a Tabela de Classe de Endereço", que nos dá a entrada na "Tabela de Classes de Endereços". Com isto isolamos uma palavra da "Tabela de Classe de Endereços", que usaremos para, através de um "AND" com uma palavra da "Tabela de Tipos de Endereços", isolada através do tipo de endereçamento usado pelo programador, dizer se a instrução é válida ou não. Se o resultado do "AND" for zero, a instrução não é válida.

ROTINA DE MONTAGEM.

A rotina de montagem tem a função de montar uma instrução ou as diretivas de segmentação FIN de LPS e END. Ela move o campo auxiliar para a posição apontada pelo contador de programa, e o incrementa de 4 (bytes).

Se a instrução tiver rótulo, o coloca na área de rótulos.

ROTINA DO END.

A rotina do END tem a função de examinar a diretiva e se ela não for um END, posicionar o # STATUS para receber a diretiva de segmentação LDS, passando o controle para o distribuidor do Analisador, que passará o controle para a rotina da LDS.

Se a diretiva for END, deverá ter o seguinte formato:

RÓTULO	COMANDO	ARGUMENTO
	END	<NOME DE SEÇÃO>
	END	SEC1

A diretiva END será analisada e se estiver correta, o apontador da 1ª seção (na PRT) será inicializado com o endereço da seção na PRT (posição na PRT). O # STATUS será posicionado para executar a rotina do segundo passo.

ROTINA DO SEGUNDO PASSO.

A rotina do segundo passo é chamada diretamente pelo Supervisor e sua função é completar a análise dos processos.

A rotina verifica nas tabelas de nomes da CDS e de cada LDS, a existência de nomes definidos como EXTERNO LOCAL, RÓTULO EXTERNO ou SEÇÃO EXTERNA (vide Tabela do Tipo do Dado) não definida na PRT. A ocorrência de algum destes casos, provoca interrupção da rotina com #STATUS igual a &F000.

O# STATUS &F000 obriga o usuário, a entrar com um comando de alteração (ver descrição dos comandos e rotinas de alteração).

Se nenhum dos casos acima ocorrer o# STATUS de processo é feito igual a &9000, ou seja, pronto para a execução (ver descrição dos #STATUS).

O usuário trabalha com a rotina do segundo passo, de forma iterativa, fazendo correções no seu processo e submetendo-o novamente à execução até que este esteja correto.

TABELAS.

Tabela das Diretivas.

A tabela das diretivas é composta pelas diretivas do montador e pela diretiva de seccionamento FIN. Cada posição contém seis caracteres conforme o quadro abaixo:

ENTRADA	DIRETIVA
0	DATA
6	DATA,1
12	FIN
18	RES
24	RES,1
30	TEXT

Esta tabela é usada no Analisador, pela rotina das diretivas, quando da análise da zona de comando, para fazer a distribuição para a parte da rotina correspondente à diretiva em análise.

O tipo do dado (bytes) da tabela de nomes, toma os seguintes valores dado pela rotina das diretivas.

Tipo do Dado.

VALOR	CARACTERÍSTICA
0 -	ETIQ : DIRETIVA DATA DO TIPO ETIQUETA A DATA D
1 -	DATA : DIRETIVA DATA DO TIPO DATA A DATA 2,-1,&15
3 -	RES : DIRETIVA RES A RES 11
5 -	DATA,1: DIRETIVA DATA,1 A DATA,1 &FA,3
6 -	RES,1 : DIRETIVA RES,1 A RES,1 15
7 -	TEXT : DIRETIVA TEXT A TEXT "NÃO TEM TEXTO"
8 -	EXTL : EXTERNO LOCAL é um nome de dado ou rótulo de instrução, que aparece no lado direito de uma diretiva DATA TIPO ETIQUETA, e ainda não definido. A DATA B B Ainda não definido.

VALOR	CARACTERÍSTICA
9 -	EXTG : EXTERNO GERAL é um nome de dado da seção de dados gerais, que aparece no lado direito de uma diretiva DATA TIPO ETIQUETA de um segmento de dados locais. A DATA # B B pertence à seção de dados comuns.
10 -	SECEXT: SEÇÃO EXTERNA é uma seção que foi chamada através de um CLS e que ainda não foi definida.
11 -	SECDEF: SEÇÃO DEFINIDA é uma seção que foi chamada através de um CLS e que já existe na PRT.
12 -	ROTEXT: RÓTULO EXTERNO é um rótulo de instrução, que ainda não foi definido (Referência para frente).
13 -	ROTDEF: RÓTULO DEFINIDO é um rótulo de instrução, que já foi definido.
14 -	FIN : DIRETIVA FIN da seção de dados gerais (CDS) ou FIN do segmento de dados locais (LDS).

Tabela das Rotinas dos Utilitários do MOB.

Esta tabela é composta pelas rotinas, que se achou mais necessárias e de imediata facilidade de implementação, para o Interpretador.

Cada posição é composta de quatro caracteres (bytes) (nome da rotina para o MOB) estando, as rotinas, na ordem alfabética como abaixo :

ENTRADA	ROTINA
0	ASEB
4	BNDC
8	BNHX
12	DCBN
16	EBAS

ENTRADA	ROTINA
20	EXIT
24	HXBN
28	IO
32	KEY
36	WAIT

O Analisador, ao gerar o código para o argumento da tabela da LPS, o faz igual ao valor da entrada da tabela (das rotinas dos utilitários do MOB) dividido por dois. Isso facilita a execução pelo Interpretador da instrução (CSV).

Note que o M: do nome dos utilitários do MOB foi deixado de fora, mas deverá ser usado quando da chamada da rotina.

OBSERVAÇÃO : O caráter ":" não é considerado por nós como alfabético e só será aceito no nome das rotinas dos utilitários do MOB.

Tabela de Mnemônicos.

Compõe-se de 3 tabelas:

- Tabela da primeira parte do mnemônico (uma palavra).
- Tabela da segunda parte do mnemônico (uma palavra).
- Tabela de apontadores para a tabela de classe de endereço (um byte).

ENTRADA CÓDIGO	MNEMÔNICOS			
	PARTE 1		PARTE 2	
0	0	0	0	0
2	A	A	E	
4	A	C	E	
6	A	D	D	
8	A	D	M	
10	A	E	E	
12	A	I	E	
14	A	N	D	
16	B	A	N	
18	B	A	Z	
20	B	C	F	
22	B	C	T	
24	B	E		
26	B	G	E	
28	B	L		
30	B	L	Z	

ENTRADA	APONTADOR	
	PARIDADE	CLASSE
0	0	0
1	0	0
2	0	0
3	1	1
4	1	2
5	0	0
6	0	0
7	1	1
8	1	3
9	1	3
10	1	3
11	1	3
12	1	3
13	1	3
14	1	3
15	1	3

ENTRADA CÓDIGO	MNEMÔNICOS			
	PARTE 1		PARTE 2	
32	B	N	E	
34	B	N	Z	
36	B	O	F	
38	B	O	T	
40	B	P	Z	
42	B	R	U	
44	B	R	X	
46	B	Z		
48	C	C	A	
50	C	C	E	
52	C	H	X	
54	C	L	S	
56	C	M	P	
58	C	N	A	
60	C	N	X	
62	C	P	S	
64	C	S	V	
66	D	C	X	
68	D	I	V	
70	D	L	D	
72	D	S	T	
74	E	O	R	
76	I	C	X	
78	I	O	R	

ENTRADA	APONTADOR	
	PARI-DADE	CLASSE
16	1	3
17	1	3
18	1	3
19	1	3
20	1	3
21	1	3
22	1	3
23	1	3
24	0	0
25	0	0
26	0	0
27	0	7
28	1	1
29	0	0
30	0	0
31	0	1
32	0	6
33	1	4
34	1	1
35	1	2
36	1	2
37	1	1
38	1	4
39	1	1

ENTRADA CÓDIGO	MNEMÔNICOS			
	PARTE 1		PARTE 2	
80	L	B	L	
82	L	B	R	
84	L	B	X	
86	L	D	A	
88	L	D	E	
90	L	D	X	
92	L	E	A	
94	L	N	E	
96	M	U	L	
98	M	V	S	
100	N	L	Z	
102	P	T	Y	
104	R	T	S	
106	S	A	D	
108	S	A	S	
110	S	B	L	
112	S	B	R	
114	S	L	C	D
116	S	L	C	S
118	S	L	L	D
120	S	L	L	S
122	S	R	C	D
124	S	R	C	S
126	S	R	L	D

ENTRADA	APONTADOR	
	PARIDADE	CLASSE
40	0	1
41	0	1
42	0	1
43	1	1
44	1	1
45	1	1
46	0	2
47	0	0
48	1	1
49	0	2
50	0	5
51	0	5
52	0	0
53	0	5
54	0	5
55	0	2
56	0	2
57	0	5
58	0	5
59	0	5
60	0	5
61	0	5
62	0	5
63	0	5

ENTRADA CÓDIGO	MNEMÔNICOS			
	PARTE 1		PARTE 2	
128	S	R	L	S
130	S	T	A	
132	S	T	E	
134	S	T	S	
136	S	T	X	
138	S	U	B	
140	T	R	S	
142	X	A	A	
144	X	A	E	
146	X	A	X	
148	X	E	X	
150	F	I	N	
152	E	N	D	

ENTRADA	APONTADOR	
	PARIDADE	CLASSE
64	0	5
65	1	2
66	1	2
67	1	2
68	1	2
69	1	1
70	0	2
71	0	0
72	0	0
73	0	0
74	0	0

A tabela de Mnemônicos é usada pela rotina das instruções.

Com as tabelas da primeira e segunda parte do mnemônico, ela verifica a validade do mesmo e o valor da entrada na tabela é colocado na instrução para o Interpretador, no segundo byte (mnemônico).

A tabela de apontadores para a tabela das classes de endereço, está ligada às anteriores, pelo código do mnemônico dividido por dois. O byte de informação da tabela está dividido em duas partes.

a) PARIDADE: Os quatro bits, de mais alta ordem, nos dizem qual é o tipo de endereço que o mnemônico aceita. Se endereço de palavra, o código será um(1). Se endereço de byte, o código será zero(0). Esta informação é usada pelo interpretador.

b) CLASSE : Os quatro bits, de mais baixa ordem, nos dizem qual a classe a que pertence o mnemônico. A classe é a entrada para a tabela das classes de endereço que contém os modos de endereçamento válidos para o mnemônico analisado.

Modo de Endereçamento.

Os modos de endereçamento, que aqui apresentaremos , foram organizados de forma a podermos decompilar o programa.

CÓDIGO			DESCRIÇÃO	
01	DL	:	DIRETO LOCAL	Definido
02	IL	:	INDIRETO LOCAL	
03	ILX	:	INDIRETO LOCAL INDEXADO	
04	DG	:	DIRETO GERAL	
05	IG	:	INDIRETO GERAL	
06	IGX	:	INDIRETO GERAL INDEXADO	
07	P	:	PARAMÉTRICO DECIMAL	Subdividido
08	PX	:	PARAMÉTRICO DECIMAL INDEXADO	
09	PH	:	PARAMÉTRICO HEXADECIMAL	
10	PHX	:	PARAMÉTRICO HEXADECIMAL INDEXADO	

CÓDIGO		DESCRIÇÃO	
11	SE :	SEM ENDEREÇO	} Criados
12	SUP :	ROTINA DO SUPERVISOR	
	RP :	RELATIVO MAIS	} Suprimidos
	RM :	RELATIVO MENOS	

Os modos de endereçamento (CÓDIGOS) serão as entradas para a tabela de tipos de endereços.

O grupo de modos de endereçamento, que forma uma classe, serão apresentados na tabela das classes de endereços.

Tabela das Classes de Endereços.

A tabela das classes de endereços contém oito classes, tendo para cada classe, uma palavra para informar quais os modos de endereçamentos são aceitos para a mesma.

Essa tabela é usada pela rotina de endereços para verificar a validade do mesmo.

As classes são compostas dos modos de endereçamento que tem os bits ligados (1) na tabela das classes de endereços, conforme o esquema a seguir:

ENTRADA	PALAVRA EM BINÁRIO																PALAVRA EM HEXADE-CIMAL
CLASSE			DL	IL	ILX	DG	IG	IGX	P	PX	PH	PHX	SE	SUP			
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0008
1	0	0	1	1	1	1	0	1	1	0	1	0	0	0	0	0	3DA0
2	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	3D00
3	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	3200
4	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	0	20F0
5	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	00F0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0004
7	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000

Tabela de Tipos de Endereços.

A tabela é composta de 12 palavras, uma para cada modo de endereçamento. A tabela de tipos de endereços tem, por função, possibilitar uma maior facilidade de identificação se o modo de endereçamento é válido para o mnemônico analisado. Isto é feito através de um "AND" entre a posição da tabela das classes de endereços apontada pela classe a que o mnemônico pertence e a posição da tabela de tipos de endereços, apontada pelo código de endereços.

Se o resultado do "AND" for zero, o mnemônico não pode ter aquele modo de endereçamento.

OBSERVAÇÃO : Nas instruções de desvio (BRANCH), onde o modo de endereçamento for **DIRETO** LOCAL (Código de Endereços = 01) **trocamos para LABEL (Código de Endereço = 00).**

Na instrução CLS trocamos o código de endereço de &01 para &0D nas instruções de desvio (BRANCH INDIRETO LOCAL) modo de endereçamento INDIRETO LOCAL (Código de endereço igual a 02), nós trocamos para código de endereço &0E. Todas essas modificações visam facilitar a tarefa do Interpretador, quando da execução do programa.

ENTRADA = (CÕD.END-1)2	PALAVRA EM BINÁRIO																PALAVRA EM HEXADE-CIMAL
MODO DE ENDEREÇAMENTO			DL	IL	ILX	DG	IG	IGX	P	PX	PH	PHX	SE	SUP			
01	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
02	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000
03	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0800
04	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0400
05	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0200
06	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0100
07	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0080
08	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0040
09	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0020
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0010
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0008
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0004

C A P Í T U L O V

EXEMPLO DE UM PROGRAMA

O programa que nós apresentaremos a seguir como exemplo, recebe uma linha com números em formato livre e separados por qualquer caracteres (alfabéticos ou especiais) e verifica qual é o maior deles.

Linha recebida.

1 234 45 12 789 345 567 98

Resultado.

O MÁXIMO É O NÚMERO 789

PROGRAMA NA LINGUAGEM FONTE

```
CDS         CDS * 4,7,180
X1          RES 16
ENDER       DATA STRING
STRING      RES 36
LINHA       RES,1 72
MAXIMO      RES 1
X2          FIN

LDS1        LDS * 14,16
A1          RES 2
CB          DATA 0
```

```
A2          DATA,1 0
            DATA,1 4
END         DATA#LINHA
A4          DATA 72
CONTA       DATA -2
A6          RES 1
A7          FIN

LPS1        LPS LDS1 * 27
ROT0        LEA CB
            CSV M:IO
            CSV M:WAIT
            LDA END
ROT3        CSV M:DCBN
            BAZ ROT1
            BAN ROT1
            LDA =2
            ADM CONTA
            XAX
            STE @#ENDER,X
            STA A6
            LBR @A6
            CMP =&0D
            BE  ROT4
ROT2        LDA A6
            ADD =1
            BRU ROT3
```

```
ROT1        XAX
            STA A6
            LBR @A6
            CMP =&0D
            BE  ROT4
            BRU ROT2
ROT4        LDA CONTA
            RTS
            FIN ROT0

LDS2        LDS * 10,6
B0          RES 2
B1          RES 1
B2          FIN

LPS2        LPS LDS2 * 20
INICIO      CLS LPS1
            BAN ROT5
            STA B1
            LDA #STRING
            STA# MAXIMO
            LDX =0
ROT3        LDA @#ENDER,X
            CMP #MAXIMO
            BGE ROT1
ROT4        XAX
            CMP B1
```

```
                BGE ROT5
                XAX
                ICX =2
                BRU ROT3
ROT1            STA #MAXIMO
                BRU ROT4
ROT5            CLS LPS3
                CSV M:EXIT
                FIN INICIO

LDS3            LDS * 12,58
C1              RES 2
CB1             DATA 0
C2              DATA,1 &80,7
END             DATA #LINHA
C3              DATA 25
C4              TEXT "O MAXIMO E O NUMERO "
C5              TEXT "NAO ENTROU NUMERO ALGUM    "
C6              FIN

LPS3            LPS LDS3 * 18
INIC            BAN ROT1
                LDE =20
                LEA C4
                MVS #LINHA
                LEA #LINHA
                ADD =20
```

```
            LDE #MAXIMO
            CSV M:BNDC
ROT2        LEA CB1
            CSV M:IO
            CSV M:WAIT
            RTS
ROT1        LDE =25
            LEA C5
            MVS #LINHA
            BRU ROT2
            FIN INIC
            END LPS2
```

DIMENSIONAMENTO DA ÁREA PARA O PROGRAMA

As áreas para as tabelas (PRT, NOMES, LPS), que vamos definir são as mínimas. Nós podemos aumentá-las quando do dimensionamento, prevendo prováveis modificações no programa.

a) Diretiva CDS CDS

XXX = 4, pois o programa tem 3 seções e a CDS.

YYY = 7, pois temos na CDS 6 nomes de dado a esquerda e 1 a direita.

ZZZZ = 180, isto é 16x2+2+36x2+72+1x2. Note que o nome STRING a direita da diretiva DATA tipo etiqueta ocupa uma palavra.

b) Diretiva LDS1 LDS

YYY = 14, pois a seção possui 8 nomes de dados na LDS, 5 rótulos de instrução e 1 nome de dados na LDS a direita da diretiva DATA tipo etiqueta (LINHA)

ZZZZ = 16, isto é 2x2+2+1+1+2+2+2+1x2

c) Diretiva LPS1 LPS LDS1

XXX = 27, pois temos 26 instruções e mais a diretiva FIN de LPS.

d) Diretiva LDS2 LDS

YYY = 10, pois a seção possui 3 nomes de dados, 5 rótulos e 2 seções chamadas na LPS.

ZZZZ = 6, isto é 2x2+1x2

e) Diretiva LPS2 LPS LDS2

XXX = 20, pois temos 19 instruções e mais a diretiva FIN de LPS.

f) Diretiva LDS3 LDS

YYY = 12, pois a seção possui 8 nomes de dados na LDS, 3 rótulos na LPS e 1 nome de dados na LDS a direita da diretiva DATA tipo etiqueta (LINHA).

ZZZZ = 58, isto é 2x2+2+1+1+2+2+20+26

g) Diretiva LPS3 LPS LDS3

XXX = 18, pois temos 16 instruções e mais as diretivas FIN de LPS e a diretiva END.

OBSERVAÇÃO: O nome da diretiva FIN de LDS (ou CDS) entrou como sendo um nome de dados apenas porque ele vai entrar na tabela de nome na área de dados defindos , mas não será aceita referência a ele, pois ali esta apenas para podermos fazer a decompilação.

PROGRAMA NA LINGUAGEM OBJETO

Vamos apresentar agora como ficará o programa na memória, depois de executada a rotina do segundo passo.

As tabelas serão dispostas na ordem abaixo para a melhor visualização pelo leitor.

a) PRT do programa
b) Tabela de nomes da CDS
c) Tabela de nomes da LDS1
d) LPS1
e) Tabela de nomes da LDS2
f) LPS2
g) Tabela de nomes da LDS3
h) LPS3.

OBSERVAÇÃO: A letra B minuscula com um traço "$\not{b}$", representa o carater branco. Se não há nada escrito numa célula o conteudo é irrelevante e inalterado pela montagem. Este conteudo será o gerado pelo Supervisor.

PRT DO PROCESSO

0	C | D	S | ∅	∅ | ∅	60			100		40	172
20	L | D	S | 1	∅ | ∅	L | P	S | 1	∅ | ∅	352	496	500	480
40	L | D	S | 2	∅ | ∅	L | P	S | 2	∅ | ∅	608	710	714	704
60	L | D	S | 3	∅ | ∅	L | P	S | 3	∅ | ∅	794	964	968	906
80	|	|	|	|	|	|	1040			

TABELA DE NOMES DA CDS

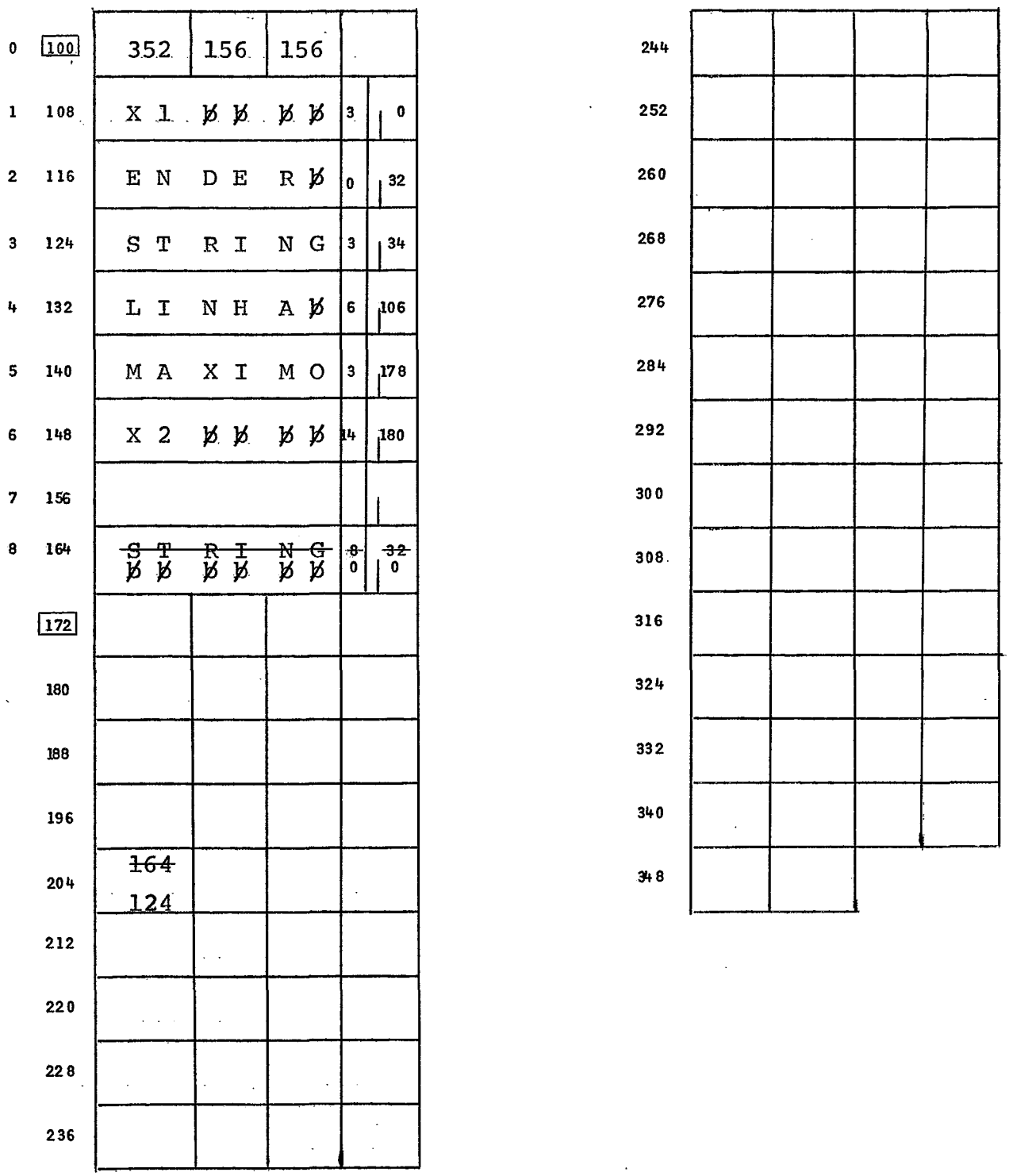


TABELA DE NOMES DA LDS1

0	352	496	424	424	608		
1	360	A 1 ∅ ∅	∅ ∅	3	0		
2	368	C B ∅ ∅	∅ ∅	1	4		
3	376	A 2 ∅ ∅	∅ ∅	5	6		
4	384	E N D ∅	∅ ∅	0	8		
5	392	A 4 ∅ ∅	∅ ∅	1	10		
6	400	C O N T	A ∅	1	12		
7	408	A 6 ∅ ∅	∅ ∅	3	14		
8	416	A 7 ∅ ∅	∅ ∅	14	16		
9	424						
10	432	R O T 2	∅ ∅	13	5		
11	440	R O T 4	∅ ∅	~~12~~ 13	4		
12	448	R O T 1	∅ ∅	~~12~~ 13	3		
13	456	R O T 3	∅ ∅	13	2		
14	464	R O T 0	∅ ∅	13	1		
15	472	L I N H	A ∅	9	4		
	480			0	0	4	
	488	472	72	-2			

LPS1

496	0 | 5	
500	1 | 92	1 | 2
504	0 | 64	11 | 14
508	0 | 64	11 | 18
512	0 | 86	1 | 4
516	2 | 64	11 | 6
520	0 | 18	0 | 3
524	0 | 16	0 | 3
528	0 | 86	7 | 2
532	0 | 8	1 | 6
536	0 | 146	11 | 0
540	0 | 132	6 | 2
544	0 | 130	1 | 7
548	0 | 82	2 | 7
552	0 | 56	9 | 13
556	0 | 24	0 | 4
560	5 | 86	1 | 7

564	0 | 6	7 | 1
568	0 | 42	0 | 2
572	3 | 146	11 | 0
576	0 | 130	1 | 7
580	0 | 82	2 | 7
584	0 | 56	9 | 13
588	0 | 24	0 | 4
592	0 | 42	0 | 5
596	4 | 86	1 | 6
600	0 | 104	11 | 0
604	0 | 150	0 | 1

TABELA DE NOMES DA LDS2

0	608	710	640	640	794		
1	616	B 0 ⌀ ⌀	⌀ ⌀		3	0	
2	624	B 1 ⌀ ⌀	⌀ ⌀		3	4	
3	632	B 2 ⌀ ⌀	⌀ ⌀		14	6	
4	640						
5	648	L P S 3 ⌀ ⌀			10	2	
6	656	R O T 4 ⌀ ⌀			13	5	
7	664	R O T 1 ⌀ ⌀			~~12~~ 13	4	
8	672	R O T 3 ⌀ ⌀			13	3	
9	680	R O T 5 ⌀ ⌀			~~12~~ 13	2	
10	688	L P S 1 ⌀ ⌀			11	1	
11	696	I N I C	I O		13	1	
	704						

LPS2

710	2	5		
714	1	54	13	1
718	0	16	0	2
722	0	130	1	2
726	0	86	4	3
730	0	130	4	5
734	0	90	7	0
738	3	86	6	2
742	0	56	4	5
746	0	26	0	4
750	5	146	11	0

754	0	56	1	2
758	0	26	0	2
762	0	146	11	0
766	0	76	7	2
770	0	42	0	3
774	4	130	4	5
778	0	42	0	5
782	2	54	1	2
786	0	64	1	10
790	0	150	0	1

TABELA DE NOMES DA LDS3

0	[794]	964	866	866	1040		
1	802	C 1	∅ ∅	∅ ∅	3	0	
2	810	C B	1 ∅	∅ ∅	1	4	
3	818	C 2	∅ ∅	∅ ∅	5	6	
4	826	E N	D ∅	∅ ∅	0	8	
5	834	C 3	∅ ∅	∅ ∅	1	10	
6	842	C 4	∅ ∅	∅ ∅	7	12	
7	850	C 5	∅ ∅	∅ ∅	7	32	
8	858	C 6	∅ ∅	∅ ∅	14	58	
9	866						
10	874	R O	T 2	∅ ∅	13	3	
11	882	R O	T 1	∅ ∅	~~12~~ 13	2	
12	890	I N	I C	∅ ∅	13	1	
13	898	L I	N H	A ∅	9	4	
	[906]			0	64	7	
	914	898	25	0	M	A	
	922	X I	M O	E		O	
	930	N	U M	E R	O		
	938	N A	O	E N	T	R	
	946	O U	N	U M	E	R	
	954	O	A L	G U	M		
	962						

LPS3

964	0 3	
968	1 16	0 2
972	0 88	7 20
976	0 92	1 6
980	0 98	4 4
984	0 92	4 4
988	0 6	7 20
992	0 88	4 5
996	0 64	12 2
1000	3 92	1 2
1004	0 64	12 14
1008	0 64	12 18
1012	0 104	11 0
1016	2 88	7 25
1020	0 92	1 7
1024	0 98	4 4
1028	0 42	0 3
1032	0 150	0 1
1036	0 152	0 2

C A P Í T U L O VI

CONCLUSÃO

Como podemos ver, através do exemplo, na estrutura do programa apresentado, a parte referente à PRT é a que consome mais memória, seguida da Tabela de Nomes. Por este motivo recomendamos , aos usuários, que ao dimensionarem seus programas , tomem o cuidado de não reservarem espaço demasiado para possíveis adições de novas seções, assim como para possíveis inclusões de Nomes de Dados ou Rótulos.

Note que isto não quer dizer que, para a Tabela de LPS, possamos reservar espaço à vontade, mas que o dimensionamento deve ser moderado.

Podemos verificar que, pelo dito acima, este é um Sistema voltado para facilitar a depuração de pequenos módulos de programas, aliado às características apresentadas pela linguagem assembler do MITRA 15.

Outro ponto importante de se notar é o sub-conjunto da linguagem assembler do MITRA 15, selecionado para fazer parte da linguagem conversacional do Sistema; não tirou as características da primeira. Simplesmente a restringiu um pouco.

A estrutura de um programa, até a execução da rotina do segundo passo inclusive, permite a decompilação, sem se preocupar com a existência de uma memória auxiliar, o que é muito interessante para que os minicomputadores, que não possuem me

mória auxiliar, ou mesmo possuindo, possam desenvolver estrutura semelhante para aumentar a sua utilização.

A organização do próprio Analisador , não se preocupando se o Sistema está executando um ou mais programas possibilita a existência de vários tipos de Supervisores, facilitando a implementação do Sistema por partes.

Como vemos, o trabalho apresentado pelo menos nesta primeira parte, não é conclusivo; é sim, um passo dado para a obtenção de um bom Sistema de Programação e Depuração Conversacional para Linguagens do Tipo Montador.

Uma sequência natural do trabalho seria a sua implementação dentro das características aqui apresentadas e, após a sua utilização, uma ampliação do seu sub-conjunto de instruções, e a reprogramação das rotinas do Analisador que se apresentarem ineficientes com relação ao tempo de execução.

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.