PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

TÉCNICAS DE OTIMIZAÇÃO DE CÓDIGO OBJETO E SUAS APLICAÇÕES EM UM COMPILADOR PARA A LINGUAGEM BASIC, USANDO UM COMPILADOR DE COMPILADORES

por

PAULO ALBERTO DE AZEREDO

Tese submetida como requisito parcial
para a obtenção do grau de

MESTRE EM CIÊNCIAS

EM

INFORMÁTICA

Assinatura do Orientador da Tese

Rio de Janeiro, GB , Janeiro de 1972

TÉCNICAS DE OTIMIZAÇÃO DE CÓDIGO OBJETO E SUAS APLICAÇÕES EM UM COMPILADOR PARA A LINGUAGEM "BASIC", USANDO UM COMPILADOR DE COMPILADORES.

SUMARIO

ABSTRATO

O objetivo dêste trabalho é criar um compilador para a linguagem BASIC (Beginner's All-purpose Symbolic Instruction Code) , para o sistema IBM-7044.

Paralelamente, são apresentadas técnicas de otimização de código objeto. O compilador, gerado pelo sistema COMCOM (COMpiler COMpiler), utiliza algumas destas técnicas, visando produzir um código eficiente.

ABSTRACT

The purpose of this work is to develop a compiler for the BASIC (Beginner's All-purpose Symbolic Instruction Code) language, under the IBM-7044 system.

At the same time, some techniques of object code optimization are presented. The compiler, generated by the COMCOM (COMpiler-COMpiler) system, uses some of these techniques, in order to produce an efficient code.

INTRODUÇÃO

Este trabalho surgiu da necessidade de ser implantado no sistema IBM-7044 da PUC/RJ, uma linguagem de alto nível, que fosse, ao mesmo tempo, precisa, simples e fácil de ser aprendida, para ser utilizada em cursos de iniciação à ciência dos computadores. A linguagem BASIC foi escolhida, por satisfazer todos estes requisitos. Além disto, esta linguagem deve facilitar o acesso ao computador a usuários ocasionais, com pequenos programas de tempo de execução reduzido.

O compilador produz código MAP-7044, que é gravado em fita magnética para posterior carga e edição, permitindo, desta maneira, o uso de rotinas da "library" do 7044.

São aceitas todas as instruções da linguagem BASIC original desenvolvida no DARTMOUTH COLLEGE. As extensões da linguagem (9) não foram introduzidas neste compilador, embora isto possa ser feito com relativa facilidade.

A linguagem BASIC foi desenvolvida em 1965 no DARTMOUTH COLLEGE, sob a orientação de J. KEMENY e T. KURTZ. Foi implantada no computador GE 255, e atualmente é disponível em quase todos os tipos de computadores. O objetivo dos autores era atrair, através de uma linguagem simples de ser aprendida e usada, usuários para o computador. Na realidade, o BASIC agia como uma iniciação para outras linguagens mais poderosas, tais como o FORTRAN ou ALGOL.

Outra vantagem da linguagem BASIC, é o fato da mesma ser recomendada para "TIME-SHARING", tornando, assim, o usuário apto para, no futuro utilizar terminais remotos com facilidade.

4. OTIMIZAÇÃO DE CÓDIGO OBJETO

4.1 - Introdução

Desde o desenvolvimento do primeiro compilador FORTRAN, a maior preocupação daqueles que o desenvolveram, era a produção de código objeto eficiente. Entretanto, muitas vezes, a necessidade de ter-se um compilador rápido tem afastado as possibilidades de otimização. Isto torna-se mais evidente ainda, pelos numerosos compiladores de um só passe que tem surgido ultimamente (PUFT, WATFOR, etc).

A necessidade de compiladores onde a otimização do código gerado é preterida à velocidade é perfeitamente justificada, especialmente no domínio universitário, onde a maioria dos programas submetidos ao computador tem pouco tempo de execução, após o processo de depuração. Também durante esta fase de depuração, um compilador rápido é mais conveniente, embora alguns destes otimizam trechos do código gerado, como veremos adiante.

Entretanto, programas com finalidades diversas daquela acima mencionada, requerem um tempo de execução prolongado ou uso frequente após a fase de depuração. Dentro desta classe de programas, estão incluidos todos aqueles que ficam catalogados na biblioteca do computador à disposição dos usuários. É conveniente, pois, que estes programas, após a depuração sejam compilados por um compilador que otimiza o código gerado.

A grande diferença, no que diz respeito à otimização, entre estes dois tipos de compiladores, reside no fato de que os primeiros, se otimizam, somente o fazem ao nível de comando, enquanto que os outros o fazem ao nível de programa, isto é, otimizam o código escrito pelo programador.

4.2 - Técnicas de Otimização

Bàsicamente, pode-se considerar a otimização em duas formas: local e global.

A otimização local envolve o exame dêsde pequenas partes de um comando, até um comando inteiro.

Para melhor caracterização, pode-se dividir a otimização local em doi- sub-tipos: em expressões aritméticas e em comandos.

4.2.1 - Otimização local. Expressões Aritméticas e Lógicas.

4.2.1.1 - Geração de código objeto para funções internas simples.

Para referências a funções internas, tais como ABS, SIGN,IFIX etc, ao invés de ser geradas chamadas as mesmas, o código para executar estas funções é desenvolvido, tal como uma macro-instrução de uma linguagem assembler. Esta técnica representa economia de código gerado, dêsde que a quantidade de instruções que compõem a função não seja maior que o número de instruções necessárias para executarem o desvio para a rotina apropriada.

4.2.1.2 - Redisposição do menos unário.

Durante a compilação de uma expressão, todos os sinais menos unários podem ser movidos para a posição mais prioritária da expressão, ou trecho de expressão. Por exemplo, a expressão

(-A-B)

é transformada para

-(A+B)

Isto é feito já visando outros tipos de otimizações.

Da mesma forma, as expressões do tipo abaixo, sofrem as seguintes transformações:

A+(-B)	para	(A-B)
-(B-C)	para	(C-B)

4.2.1.3 - Avaliação prévia de operações com constantes.

Quando um operador combina duas constantes, esta operação pode ser substituida pelo resultado da operação, que é feito em tempo de compilação.

Exemplo:

X = 2 + 7 é substituido por X = 9

Teõricamente, tôdas as operações feitas sôbre constantes, e que não envolva variáveis do programa, podem ser avaliadas ao tempo de compilação.

Exemplo:

X = SQRT(2.0) é substituido por X = 1.41...

4.2.1.4 - Conversão de constantes

Se uma constante aparece em uma expressão mista, a conversão' pode ser feita durante a compilação.

Exemplo:

A = A+1 é substituido por A = A+1.

4.2.1.5 - Substituição de constantes que aparecem como multiplicadôres, divisores ou expoentes.

Em uma expressão, nas operações de multiplicação, ou potenciação, em que o 2º operando é a constante 2, as seguintes simplificações podem ser feitas:

A**2	é substituido por	A+A
A**2	é substituido por	A*A

que são operações mais rápidas que as originais.

Da mesma forma, quando em operações de multiplicação e/ou divisão o 2º operando for da forma

$B = 2^k$, k inteiro,

então a operação pode ser substituida da seguinte forma:
multiplicação é substituida por k "shifts" à esquerda, de um bit cada.
divisão é substituida por k "shifts" à direita, de um bit cada.

Naturalmente, o 1º operando é do tipo inteiro.

4.2.1.6 - Fatoração de expressões aritméticas e lógicas.

Este tipo de otimização procura transformar expressões em formas mais simplificadas, através de fatoração.

Exemplo:

A*B+A*C é substituido por A*(B+C)

A fatoração de expressões, como veremos adiante, também pode ser feita ao nível global. Na realidade, a primeira é um caso particular' da segunda.

4.2.1.7 - Eliminação de sub-expressões comuns.

Quando uma sub-expressão é repetida duas ou mais vêzes em uma expressão, como por exemplo:

A*B+(A*B)**2

a sub-expressão (A*B) pode ser avaliada sòmente uma vêz, sendo seu resultado usado em tôdas as referências à ela.

Também como no caso anterior, êste tipo de otimização é um caso particular de eliminação de sub-expressões comuns em nível global.

4.2.1.8 - Desenvolvimento de expressões lógicas.

Além dos tipos acima citados de otimização de expressões lógicas, podem ser apontadas as seguintes otimizações:

Se uma expressão lógica contém sòmente operadores OR ou AND, o compilador pode desmembrar o comando em vários outros, funcionalmente equivalentes ao original, de tal forma que os operadores não mais apareçam, com o objetivo de acelerar a execução. Exemplificando a linguagem FORTRAN, seja o comando de desvio condicional:

```
IF(A.LT.B.OR.C.GT.D) GO TO 10
```

é equivalente a:

```
IF(A.LT.B) GO TO 10
IF(C.GT.D) GO TO 10
```

Desta maneira, dentro da expressão, a primeira ocorrência de .TRUE. executará o desvio.

4.2.1.9 - Escolha conveniente de ordem de compilação em determinados tipos de expressões aritméticas.

Dependendo da maneira de compilar determinadas expressões aritméticas, pode-se fazer com que, além de se obter menor número de instruções no código gerado, sejam utilizados menos variáveis temporárias . Esta técnica visa deixar no acumulador, o resultado de uma sub-expressão que deva ser operado em seguida, evitando-se, desta forma, a criação de uma variável temporária para salvar seu resultado.

Exemplo:

(expr1)/(expr2)

Se a primeira expressão compilada for expr1, então teremos de criar uma variável temporária para o resultado. O resultado de expr2 também deverá ir para uma variável temporária para, finalmente, ser efetuada a divisão.

Entretanto, se expr2 fôsse avaliada inicialmente, só haveria necessidade de uma variável temporária, ao invés de duas, pois o resultado de expr1 reside no acumulador. Com isto, ganha-se memória e tempo de execução, dado o menor número de instruções geradas.

4.2.1.10 - Otimização em outros comandos.

Para aqueles comandos que podem ser escritos em uma forma particular reduzida da sua forma geral, é claro que o código gerado para este caso deve ser mais simples que o caso geral.

Exemplo:

Suponha-se a forma geral de um comando como:

DO <variable> = <expr1> TO <expr2> BY <expr3>

mas que possa aparecer sob a forma:

DO <variable> = <expr1> TO <expr2>

onde o passo é considerado 1, "by default".

Neste caso, o código gerado é mais simples que aquele gerado para o caso anterior.

Da mesma forma, as instruções de desvio geradas por um comando IF podem ser arranjadas de forma a maximizar a eficiência do código.

Por exemplo o comando:

IF < condição> THEN < comando>

é definido no PL/I como sendo um teste da condição, seguido de / um desvio se falsa, ao comando seguinte. Entretanto, quando o comando após o THEN é um GO TO, o comando IF pode ser compilado como um desvio se verdadeira, para o label especificado no GO TO.

4.2.3 - Otimização Global

Na otimização global, são examinados todos os comandos que compõem um bloco de programa. Entende-se por bloco de programa, uma sequência de comandos, na qual só se pode dar entrada pelo primeiro comando e só se pode sair do bloco pelo último comando do mesmo. O programa todo, também pode ser considerado como um bloco, sendo que neste caso, o bloco não é considerado para fins de otimização.

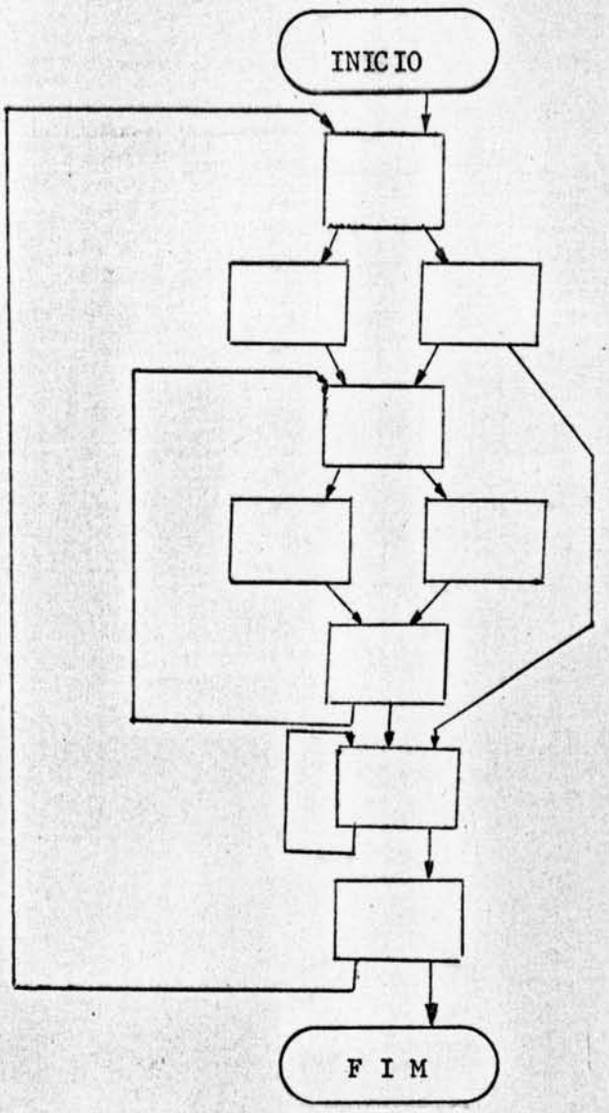


Fig. 1 Programa blocado.

Internamente a cada bloco, podem ser feitas as seguintes otimizações:

4.2.3.1 - Eliminação de expressões comuns.

Uma expressão pode ocorrer duas ou mais vêzes dentro de um bloco, de tal forma que as variáveis envolvidas na expressão não trocam de valor entre as suas ocorrências.

O compilador pode eliminar expressões comuns, salvando o resultado da primeira avaliação em uma variável temporária (gerado pelo compilador), ou na própria variável onde o valor da expressão é atribuido.

Por exemplo:

```
X1 = A1*B1
     .
     .
     .
Y1 = A1*B1
```

Assumindo-se que os valôres de A1, B1, e X1 não se alterem entre êstes comandos, pode ser feita uma otimização da seguinte forma:

```
X1 = A1*B1
     .
     .
     .
Y1 = X1
```

Se a variável à qual é atribuido o valor resultante da avaliação da primeira expressão comum é alterado antes da ocorrência da última expressão, então o resultado é atribuido a uma variável temporária, da seguinte forma:

```
T1 = A1*B1
X1 = T1
   :
X1 =
   :
Y1 = T1
```

Se uma expressão comum ocorrer sob a forma de uma sub-expressão em uma expressão, uma variável temporária é criada para preservar o valor da sub-expressão comum. Por exemplo, na expressão C1+A1*B1, uma variável temporária é criada para guardar o valor da sub-expressão A1*B1 , se esta for a sub-expressão comum.

Uma aplicação importante desta técnica ocorre em comandos que contém variáveis subscritas, onde o índice das variáveis são expressões ' comuns.

Exemplo:

```
PAYROLLTAX(A1*B1+C1)=PAYCODE(A1*B1+C1)*WEEKPMNT(A1*B1+C1)
```

4.2.3.2 - Retirada de expressões invariantes do interior de "loops".

Uma expressão é dita invariante, se o compilador puder decidir se o valor desta expressão será idêntico para tôdas iterações do "loop".

O "loop" pode ser ou um "DO loop" ou um "loop" que possa ser detetado pela análise do fluxo do programa.

Exemplo:

```
DO  I = 1  TO N;
B(I) = C(I)*SQRT(N);
P = N*J;
END;
```

Êste "loop" pode ser otimizado a produzir código eficiente, se for transformado em:

```
TEMP = SQRT(N);
P = N*J;
DO I = 1 TO N;
B(I) = C(I)*TEMP;
END;
```

A retirada de expressões do interior de "loops" é feita, assumindo-se que tôdas as expressões no "loop" são executadas mais frequentemente do que aquelas que estão imediatamente fora do "loop". Ocasionalmente isto falha.

Por exemplo:

```
DO I = 1 TO N;
  :
        DO J=I TO N/3;
        X(J)=Y(J)*SQRT(N);
        END;
  :
END;
```

EXECUTAV. NÃO-E

I= 1 J= 1,2,3
I= 2 J= 2,3
I= 3 J= 3
I= 4 –
:
I= 10 –

Eu vi 6 vêzes e não 5! Onde está o erro? Paulo Freire 29/NOV/80

Supondo que N=10, a função SQRT(N) será executada 5 vêzes; entretanto, se esta for retirada do "DO loop" menor, será executada 10 vêzes!

Por outro lado, nem sempre é possível o reconhecimento de "loops" em um programa, principalmente quando são usados "labels" variáveis, que podem, inadvertidamente inibirem o seu reconhecimento.

4.2.3.3 - Contrôle de comando "DO loop".

Sempre que for possível, os valôres das variáveis de contrôle dos comandos "DO loop" devem ser conservados em indexadores, unidades

de acesso mais rápido, durante a execução do "loop". Por exemplo,o compilador deve fazer manter em indexadores os valores de I, K e L no seguinte' comando:

DO I = A TO K BY L;

4.2.3.4 - São possíveis ainda outras otimizações, sendo que a maioria delas são dependentes do "hardware" da máquina em questão, e da linguagem que é compilada.

4.3 - FORTRAN H. [15]

O exemplo mais característico de otimização de código objeto é dado no compilador FORTRAN H.

Êste compilador deteta, inicialmente, todos os blocos do programa, e identifica, dentro de cada um, tôdas as ocorrências de "loops". ' São também identificadas as 128 variáveis mais comuns do programa, bem como aonde elas são usadas e onde à elas são atribuidos valores. A partir ' daí, são feitas as otimizações citadas em 4.2.3.1, 4.2.3.2, 4.2.3.3 e 4.2.3.4. Também são executadas otimizações locais.

Os autôres do FORTRAN H afirmam que para pequenos programas , como uma estrutura simples, o compilador gera código objeto perfeito, de tal forma que não pode ser melhorado por um programador. [15]

A relação de velocidade entre os programas objetos gerados pelo FORTRAN H e o WATFIV varia de acôrdo com o programa compilado, mas é aproximadamente de 20 para 1 [15].

O FORTRAN H aceita tôda a linguagem FORTRAN e foi escrito em FORTRAN. Posteriormente foi usado para compilar a si mesmo, produzindo uma "performance" satisfatória. [15]

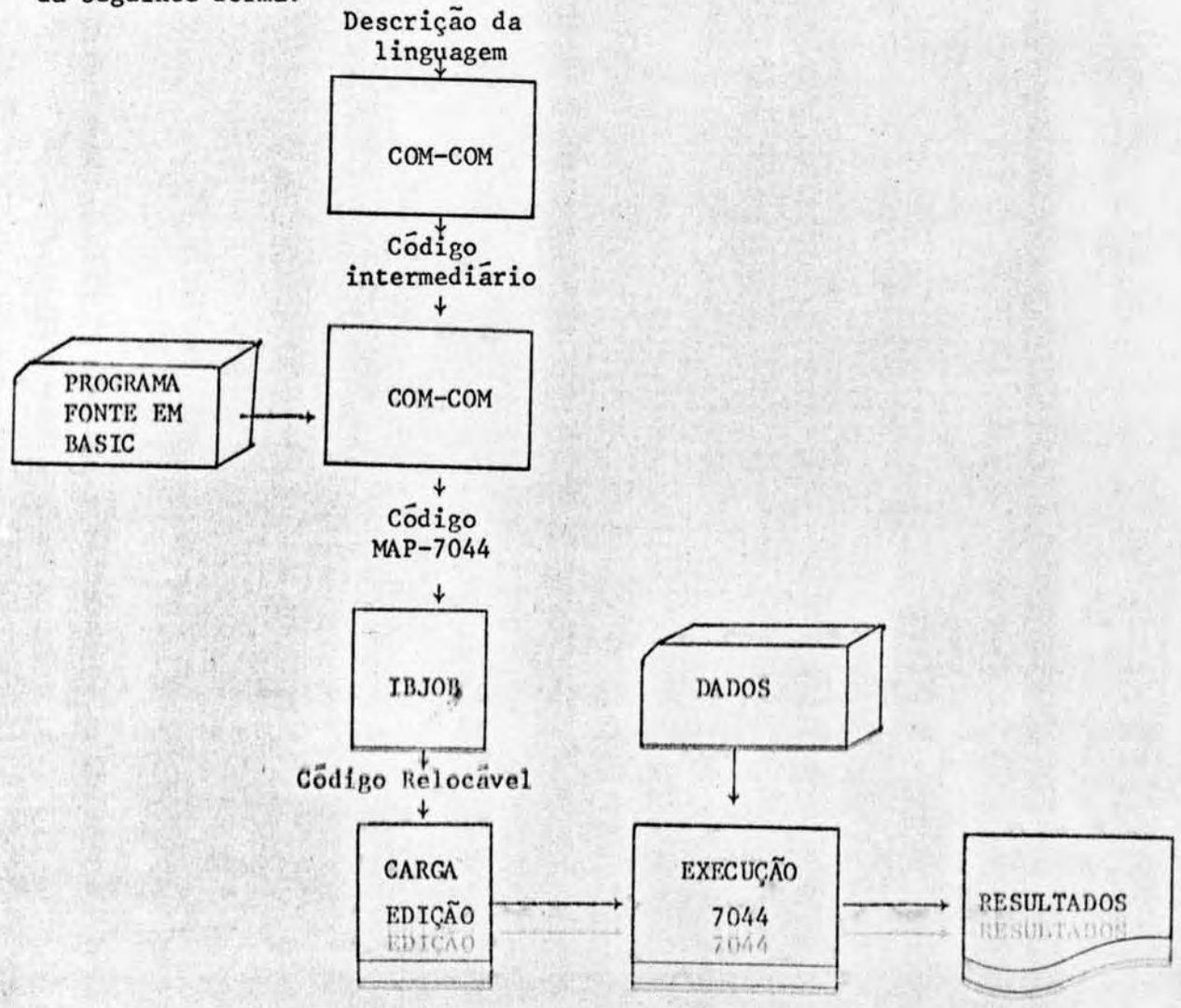
4.4 - Outros compiladores otimizadores.

Atualmente, outros compiladores que otimizam o código objeto gerado estão disponíveis. Entre êles, podem ser citados o PL/I [21] e o ALGOL [9].

5. ESTRUTURA DO SISTEMA

A descrição sintática e semântica do Compilador BASIC, objeto dêste trabalho, é feito na linguagem COMCOM, que produz um código intermediário. A Compilação de programas escritos na linguagem BASIC é feita por êste intermediário e pelo sistema COMCOM. O código resultante desta operação é MAP-7044.

Esquemáticamente, podemos representar a estrutura do sistema da seguinte forma:

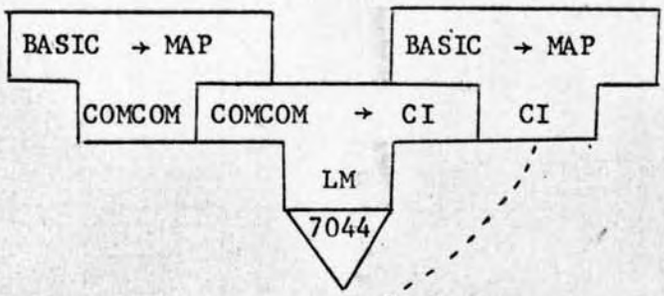


Após implantado, o sistema não mais exigirá a parte inicial, onde é obtido o código intermediário.

Segundo a notação de Earley, a estrutura do sistema é descrita da seguinte forma:

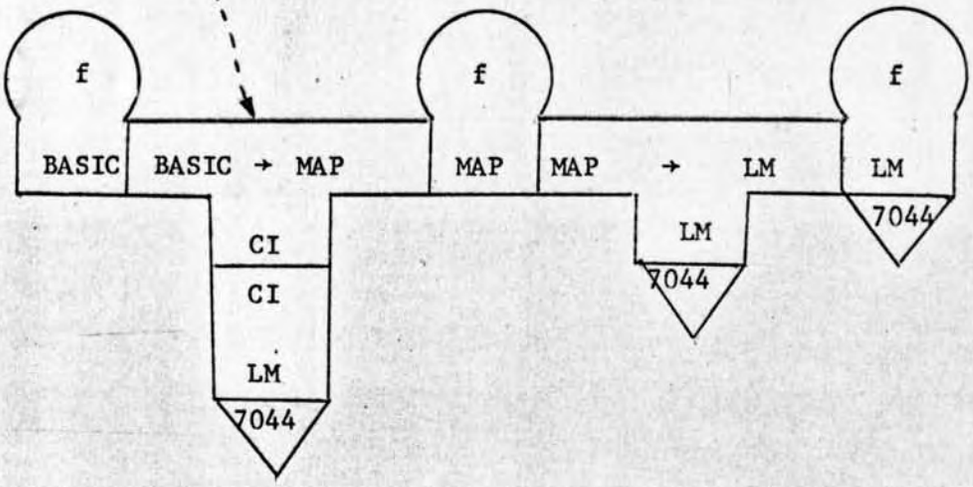
1º FASE

OBTENÇÃO DO CÓDIGO INTERMEDIÁRIO



2º FASE

COMPILAÇÃO DE PROGRAMAS EM BASIC E EXECUÇÃO



6. DESCRIÇÃO SINTÁTICA DA LINGUAGEM BASIC

Passaremos agora a descrever a sintaxe da linguagem BASIC, utilizando a Forma Normal de Backus (BNF). Êste tipo de definição sintática é muito conveniente quando se trata de uma linguagem algorítmica.

6.1 - Segundo a notação de Backus. [6]

Em adição aos símbolos admitidos na linguagem BASIC, são introduzido metasímbolos, necessários a definição da linguagem. Êstes metasímbolos são:

<, >, ::=, | , { e }

<, > são usados como delimitadores para conter o nome das classes.

::= é usada para indicar definição. Deve ser entendido como "é definido' como" ou "consiste de".

| é usado para indicar alternativa.

{ } são usados para indicar que a parte que é contida dentro destes delimitadores podem ou não ocorrer.

<nulo> ::=

(isto é, a sequência vazia de símbolos)

<caracter> ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|X|Y|Z

<dígito> ::= 0|1|2|3|4|5|6|7|8|9

<caracter especial> ::= +|-|*|/|Ø|=|(|)|.|,

<inteiro> ::= <digito> | <inteiro> <digito>

<número decimal> ::= <inteiro> . <inteiro>|<inteiro>.|.<inteiro>

<sinal> ::= <nulo>|-|+

<expoente> ::= E <sinal> <digito>{<digito>}

<parte de número> ::= <inteiro> | <número decimal>

<número> ::= <parte de número>|<parte de número><expoente>

<número com sinal> ::= <sinal><número>

<variável simples> ::= <caracter>|<caracter><digito>

<variável subscrita> ::= <variavel simples>(<expressão>{,< expressão>}

<variável> ::= <variável simples>|<variável subscrita>

<nome de função> ::= SEN|COS|TAN|ATN|EXP|LOG|RQU|INT|ABS|FN<caracter>

<termo função> ::= <nome de função>(<expressão>)

<termo> ::= <número>|<variável>|<termo função>|(<expressão>)

<fatôr involução> ::= <termo>|<termo>**<fator involução>

<fator multiplicação> ::= <fator involução>|<fator involução>
<operador de multiplicação> <fator multiplicação>

<operador de multiplicação> ::= *|/

<expressão> ::= <fator multiplicação>|<sinal> <expressão>|<fator de multiplicação>
<operador de adição> <expressão>

<operador de adição> ::= +|-

<comando de atribuição> ::= ARIT <variável> = <expressão>

<comando LEIA> ::= LEIA <lista de leitura>

<lista de leitura> ::= <variável>|<lista de leitura>, <variável>

<comando IMPRIMA> ::= IMPRIMA <lista de impressão>

<lista de impressão> ::= <expressão>|'<mensagem>'| <nulo>

<mensagem> ::= <qualquer sequência de símbolos, que não formem palavras reservadas>

<símbolos> ::= <caracter>|<digito>|<caracter especial>

<palavras reservadas> ::= <nome de função>|<operador de relação>|
DIM|VA|VAPARA|SE|PARA|PROXIMO|FIM|PARE|
LEIA|IMPRIMA|VASUB|RETORNE|DEF|ARIT|COM|
ENTAO|ATE|DEGRAV|SUB

<comando VA PARA> ::= VA PARA <número de comando>

<número de comando> ::= <inteiro>

<comentário> ::= COM <mensagem>

<operador de relação> ::= MEN|MEG|MAI|MAG|IGU|DIF

<comando SE> ::= SE <expressão> <operador de relação> <expressão>
ENTAO <número de comando>

<comando PARA> ::= PARA <variável simples> = <expressão> ATE
<expressão> <parte final>

<parte final> ::= <nulo>| DEGRAU <expressão>

<comando PROXIMO> ::= PROXIMO <variável simples>

<comando FIM> ::= FIM

<comando PARE> ::= PARE

<comando DIM> ::= DIM <lista de dimensão>

<lista de dimensão> ::= <variável dimensão>|<variável dimensão><lista dimensão>

<variável dimensão> ::= <variável simples>(<inteiro>{,<inteiro>})

<comando DEF> ::= DEF FN <caracter>(<variável simples>)=<expressão>

<comando VA SUB> ::= VA SUB <nº de comando>

<comando RETORNE> ::= RETORNE

<comando> ::= <comando de atribuição>|<comando LEIA>
<comando IMPRIMA>|<comando VA PARA>|
<comando SE>|<comando DEF>|
<comando DIM>|<comando PARA>|
<comando VA SUB>|<comando RETORNE>
<comentário>|<comando PARE>

<comando BASIC> ::= <comando>|<número de comando> <comando>

<programa BASIC> ::= <programa> <comando FIM>

<programa> ::= <comando BASIC>|<comando BASIC> <programa>

7. DESCRIÇÃO SEMÂNTICA DA LINGUAGEM BASIC. [9]

A semântica da linguagem BASIC é tão expontânea que o simples exame de um programa exemplo pode servir para descrever a semântica da lin - guagem. Sòmente poucas partes não são tão fácilmente compreendidas.

Tendo em vista esta simplicidade, será dada apenas uma descri ção sumária da semântica da linguagem BASIC.

VARIÁVEIS

Variável simples é a designação dada a um valor simples. Êste valor pode ser usado em expressões para formar outros valôres e podem ser al terados através de comando de atribuição ou leitura.

Variável subscrita designa um conjunto de valôres, cada um com o mesmo significado de uma variável simples. Cada expressão aritmética ' da lista de subscritos corresponde a um subscrito da variável. O elemento do conjunto a que é feita a referência é especificado pelo valor numérico atual dos subscritos. Quando se trata de variável com um subscrito, o valor numéri co atual do mesmo deverá estar contido entre zero e o tamanho máximo, especificado no comando DIM, pois ao primeiro elemento corresponde a ordem zero.

Nas variáveis de dois subscritos, o primeiro corresponde à or dem da linha da matriz e o segundo à ordem da coluna. O valor numérico de ca da subscrito deverá estar contido entre zero e o tamanho máximo especificado para cada subscrito.

No compilador descrito neste trabalho as variáveis, subscritos ou não, sòmente representam um tipo de dado numérico, que corresponde a representação de ponto flutuante do computador IBM-7044.

NÚMEROS

Números tem o seu significado convencional. A parte de expoente é um fator de escala expressada como uma potência inteira de 10.

EXPRESSÕES

As expressões aritméticas exprimem a maneira pela qual devam ser operadas os seus componentes. O valor de uma expressão é obtido, pela execução das operações indicadas sôbre o valor corrente dos elementos primários da expressão.

COMANDOS

A unidade de operação da linguagem é o comando. Cada comando' normalmente é executado na ordem em que êle comparece dentro do programa, exceto quando houver um comando que especifique incondicional ou condicional - mente que esta ordem natural seja desfeita. Uma transferência incondicional' pode ser feito por um comando VA PARA. O comando SE permite comparar o valor de duas expressões aritméticas através de um dos operadores de relação, e se o resultado da comparação for verdadeiro, o comando seguinte a ser executado será aquele cujo número comparece à direita da palavra ENTAO do comando SE.

O comando PROXIMO, é usado para indicar o término do laço iniciado pelo comando PARA, e precisa usar exatamente a mesma variável especificada no PARA, neste comando, se a terceira expressão for omitida, será suposto o valor 1.

O comando LEIA atribuirá às variáveis da sua lista, os valores obtidos da leitura de cartões perfurados. Os valores que devam ser atribuidos às variáveis da lista, deverão estar perfurados em cartão, e separados, um do outro, de pelo menos uma coluna em branco. As formas permitidas de representar números em cartões são aquelas reconhecidas pela rotina CNVCNS. A cada comando LEIA não é necessariamente acionado um cartão. Isto somente será feito se já tiverem sido lidos todos os valores perfurados do cartão anterior.

Para a impressão, podem ser especificados expressões aritméticas e/ou literais. No caso de expressões, esta será previamente avaliada.

Surotinas terminam com o comando RETORNE, mas não existe uma maneira especial de indicar seu início. Qualquer comando do programa pode ser início da mesma. O usuário deve indicar o seu início, através do comando VA SUB.

As funções são definidas pelo comando DEF; o nome das funções é composto das letras FN seguidas por uma letra qualquer.

O comando DIM é usado para criar variáveis subscritas; entretanto, este comando não é necessário para dimensionar variáveis subscritas cuja dimensão ou dimensões não ultrapassem a 10 elementos.

COM é usado para inserir comentários no interior do programa.

O comando PARE faz com que a execução do programa seja encerrada.

Embora a linguagem BASIC original exija que todos os comandos sejam numerados, e que os números estejam em ordem crescente, êste compilador não o exige, sendo interessante, inclusive, que sòmente sejam numerados os comandos que necessitam de numeração. Com isto, evita-se a utilização desnecessária da tabela de símbolos, ficando esta com maior disponibilidade.

8. TÉCNICAS DE OTIMIZAÇÃO UTILIZADAS

As otimizações de código objeto utilizadas no compilador, reduzem-se ao nível local, tanto pela própria maneira / de ser gerado o compilador, como pela sua finalidade. A maioria delas foi feita em expressões aritméticas. Os comandos PARA, SE e LEIA também sofreram otimizações.

8.1 - Otimizações em Expressões Aritméticas

8.1.1 - Variáveis Temporárias.

Durante a compilação das expressões aritméticas, é evitado, quando possível, a criação de variáveis temporárias desnecessárias. Isto é conseguido, fazendo-se com que após ser / compilado uma operação qualquer, seja verificada a expressão / seguinte: se esta exigir o resultado da primeira, que está em um dos registradores (AC/MQ), então não é criada uma variável temporária. Com esta técnica, consegue-se evitar a criação de aproximadamente 50% dos temporários que seriam criados, dependendo da expressão.

8.1.2 - Constantes.

Pelo fato de a linguagem BASIC só admitir aritmética de ponto flutuante, as constantes que não são escritas com ponto decimal, são imediatamente convertidas durante a sua compilação.

Também as constantes utilizadas como índice de variáveis indexadas são convertidas para o ponto durante a compilação, se as mesmas forem escritas como ponto flutuante.

8.1.3 - Função ABS.

A função interna ABS é uma exceção dentre as funções internas: sua compilação não gera chamada para uma rotina externa, e sim os comandos que executam a sua função.

8.1.4 - Teste de Subscritos.

O teste de validade de subscritos de variáveis durante a execução de um programa BASIC, é feito através da rotina TINDEX. Entretanto, quando o subscrito é uma constante, este teste é feito durante a compilação do mesmo.

8.2 - Otimização no comando "PARA".

Se qualquer uma das expressões do comando se reduzirem a uma variável simples ou uma constante, então o código gerado para o comando é reduzido, no sentido de não serem criados temporários para armazenarem o seu resultado, como seria, se fossem considerados genericamente como expressões.

8.3 - Otimização no comando "SE"

Da mesma forma que no caso do comando "PARA", nenhum temporário será gerado, se a primeira expressão for reduzida a uma variável simples ou constante.

8.4 - Otimização no comando "LEIA"

Geralmente, os compiladores de linguagens de alto nível, geram, nos comandos de leitura, instruções estanques para a leitura de cada variável da lista de variáveis, tal como se cada uma estivesse em um comando de leitura individual. A otimização feita nêste compilador, faz com que haja sòmente um conjunto de instruções de leitura de dados, que é executado tantas vêzes quantas forem as variáveis da lista de variáveis.

Esta técnica é possível aplicar, quando as variáveis da lista de variáveis são do mesmo tipo, que é o caso da linguagem BASIC.

9. TABELA DE SÍMBOLOS, CÓDIGO GERADO.

Nesta seção serão apresentados o uso da tabela de símbolos e os códigos gerados pelo compilador, para cada tipo de comando da linguagem.

TABELA DE SÍMBOLOS

O compilador utiliza para armazenar e manipular os símbolos do programa a própria tabela disponível no COMCOM, que pode ser usada através das funções $IHASH (para instalar) e / $JHASH (para consulta).

Qualquer variável ou função definida no programa, é instalado na tabela, juntamente com seus atributos, que descrevem a unidade. Também os nºs de comandos são instalados na tabela. Esta descrição é feita no vetor ATRIB, da seguinte forma:

SIMBOLO (tipo)	ATRIBUTO	
VARIAVEL SIMPLES	0	
" c/ 1 SUBSCRITO	N	(N=nº de elementos declarados)
" c/ 2 SUBSCRITOS	K	(K=nº de linhas declaradas x 100000 +nº de colunas declaradas. Esta operação visa armazenar em somente um elemento de / ATRIB, as duas dimensões declaradas para a variável).
Nº DE COMANDO(DEFINIDO)	99999	
Nº DE COMANDO(SÓ REFERENCIADO)	10000	
FUNÇÕES DEFINIDAS NO PROGRAMA	90000	
FUNÇÕES INTERNAS	80000	
" " REFERENCIADAS	80001	

10. CÓDIGO GERADO

COMANDO		CÓDIGO
DIM	um subscrito (subs 1)	var BSS subs 1
	dois subscritos (subs1, subs2)	var BSS subs1 x subs2
VA PARA nc		TRA Tnc
PARE		TRA S.JxIT
VA SUB nc		TSX Tnc,4
RETORNE		TRA 1.4
número de comando (nc)		Tnc BSS o

Nas tabelas seguintes, cada comando é apresentado nas diversas formas com que êle pode aparecer. Assim, cada coluna da tabela representa uma forma com que o comando pode se apresentar. Ao longo da coluna, é apresentado o código gerado para aquela forma. Os quadros vazios indicam que nenhum código é gerado para executar a função descrita para a linha em que está o quadro.

Entende-se por "Forma Geral", a forma de um comando no qual nenhuma expressão aritmética do mesmo se apresenta sob a forma reduzida, isto é, não é uma variável simples ou constante.

Nos quadros onde se lê "IDEM" , significa que o código que está no quadro à sua esquerda se repete no quadro em questão.

As rotinas TOFIX, TINDEX e BSCRD foram criadas especialmente para êste sistema, e estão descritas na seção 12.2.

CÓDIGO GERADO PARA O COMANDO "LEIA"

linha 1: Inicialização do contrôle do "loop" de leitura.

linha 2: Avaliação da expressão de compõe o índice da variável subscrita.

linha 3: Teste do índice e obtenção do endereço efetivo.

linha 4: Endereços bases das variáveis de leitura e "loop" de leitura.

linha 5: Avaliação da expressão que compõe o primeiro índice da variável ' subscrita.

linha 6: Teste do primeiro índice.

linha 7: Avaliação da expressão que compõe o segundo índice da variável ' subscrita.

linha 8: Teste do segundo índice e obtenção do endereço efetivo.

linha 9: Idem linha 4.

COMANDO LEIA

	TIPO DE VARIÁVEL		
	SIMPLES	UM SUBSCRITO var(ind)	
		ind = expressão	ind=constante ou variável
1	LXA n,4	LXA n,4	LXA n,4
2		Código para avaliar a expressão. Resultado em T1.	
3		CALL TOFIX(T1,T1) CALL TINDEX(T1,MAX) CLA T1 ADD STARTn-N	CALL TINDEX(ind,MAX) CLA ind ADD STARTn-N
4	Endereço das variáveis da lista, na ordem de compa recimento. STARTn CALL BSCRD(R) CLA R STO* STARTn,4 TX1 *+1,4,-1 TX4 STARTn,4,0	idem	idem

(continua).

COMANDO LEIA (continuação)

	DOIS SUBSCRITOS var(ind 1, ind 2)			
	ind 1 = expressão 1 ind 2 = expressão 2	ind 1 = expressão 1 ind 2 = const./variável	ind 1 = const.ou variável ind 2 = expressão 2	ind 1 = const.ou variável ind 2 = const.ou variável
5	Código para avaliar expressão 1. Resultado em T1	idem		
6	CALL TOFIX(T1,T1) CALL TINDEX(T1,MAXL) CLA T1 STO TREAD	idem	CALL TOFIX(ind1,T1) CALL TINDEX(T1,MAXL) CLA T1 STO TREAD	idem
7	Código para avaliar expressão 2, Resultado em T1.		Código para avaliar expressão 2.Resultado em T1.	
8	CALL TOFIX(T1,T1) CALL TINDEX(T1,MAXC) CLA T1 SUB =1 ARS 35 MPY MAXL ADD TREAD ADD STARTn-N	idem	idem	CALL TOFIX(ind2,T1) CALL TINDEX(T1,MAXC) CLA T1 SUB =1 ARS 35 MPY MAXL ADD TREAD ADD STARTn-N
9	IDEM A VARIAVEL SIMPLES	IDEM A VARIAVEL SIMPLES	IDEM A VARIAVEL SIMPLES	IDEM A VARIAVEL SIMPLES

Nos quadros anteriores, os símbolos usados têm o seguinte significado:

N = número de elementos da lista.

n = número de ordem do comando em questão, dentre todos os comandos LEIA do programa.

MAX = número de elementos com o qual foi dimensionado um vetor.

MAXL = número de linhas com o qual foi dimensionada uma matriz.

MAXC = número de colunas com o qual foi dimensionada uma matriz.

CÓDIGO GERADO PARA O COMANDO "SE"

linha 1: Avaliação da primeira expressão.

linha 2: Avaliação da segunda expressão.

linha 3: Comparação das duas expressões

linha 4: Transformação do resultado da comparação para ponto fixo.

linha 5: Teste do resultado da comparação.

COMANDO SE

SE <upl> <operador de relação> <expr2> ENTAO <nº de comando>

	FORMA GERAL	exp=const. ou variável	exp2=const. ou variável
1	Código para avaliar expl. Resultado em T1.		Conforme expressão 1
2	Código para avaliar exp2. Resultado no AC.	Conforme expressão 2	CIA exp2
3	FSB T1	FSB expl	Conforme expressão 1
4	UFA =0230000000000 ALS 10 ARS 10	idem	idem
5	Código para testar AC VEJA ABAIXO	idem	idem

CODIGOS TESTAR ACUMULADOR

TESTE	CODIGO
IGU	TZE
MAI	TMI
MEN	TPL
MEG	TPL TZE
MAG	TMI TZE
DIF	TNZ

CÓDIGO GERADO PARA OS COMANDOS "PARA" E "PROXIMO"

linha 1: Avaliação da primeira expressão.

linha 2: Avaliação da segunda expressão.

linha 3: Avaliação da terceira expressão.

linha 4: Teste de incremento negativo.

linha 5: Teste de fim de condição.

linha 6: Incremento da variável de contrôle e retorno ao início do "loop".

COMANDO PARA E PROXIMO

PARA <variável simples> = <expressão 1> ATE <expressão 2> {DEGRAU <expressão 3>}

	CASO GERAL	expressão 1 = constante ou variável simples	expressão 2 =constante ou variável simples	expressão 3 =constante ou variável simples	expressão 3 nao esta presente
1	Código para avaliar expressão 1. Resultado em VR.	CLA exprl STO VR	Conforme expressão 1	Conforme expressão 1	Conforme expressão 1
2	Código para avaliar expressão 2. Resultado em P1.	Conforme expressão 2		Conforme expressão 2	Conforme expressão 2
3	Código para avaliar expressão 3. Resultado em P2.	Conforme expressão 3	Conforme expressão 3		
4	TPL *+4 CLA *+5 SSP STO *+3	Conforme expressão 3	Conforme expressão 3		
5	D1 CLA P1 FSB VR TMI E1+1	Conforme expressão 2	D1 CLA exp2 FSB VR TMI E1+1	Conforme expressão 2	Conforme expressão 2
6	CLA VR FAD P2 STO VR E1 TRA D1	idem	idem	CLA VR FAD exp3 STO VR E1 TRA D1	CLA VR FAD =1.0 STO VR E1 TRA D1

CÓDIGO GERADO POR EXPRESSÕES ARITMÉTICAS

POTÊNCIAS

linha 1: avaliação da primeira expressão.

linha 2: " " segunda " .

linha 3: execução da operação.

SOMAS / SUBTRAÇÕES

linha 1: avaliação da primeira expressão.

linha 2: " " segunda " .

linha 3: carregar o primeiro operando.

linha 4 e 5: execução da operação correspondente.

PRODUTOS / DIVISÕES

linha 1: avaliação da primeira expressão.

linha 2: " " segunda " .

linha 3 e 4: execução da operação.

EXPRESSÕES ARITMÉTICAS

POTÊNCIAS

	FORMA GERAL	FORMAS PARTICULARES		
	(exp1)**(exp2)	exp1 = constante ou variável simples	exp2 = constante ou variável simples.	exp1 = const./var.simples exp2 = const./var. simples
1	Código para avaliar exp1. Resultado em T1.		Código para avaliar exp1. Resultado em T1.	
2	Código para avaliar exp2. Resultado em T2.	Código para avaliar exp2. Resultado em T1.		
3	CLA T1 STQ T2 TSL .EXP3.	CLA exp1 STQ T1 TSL .EXP3.	CLA T1 STQ exp2 TSL .EXP3.	CLA exp1 STQ exp2 TSL .EXP3.

TÊRMOS

VARIÁVEIS SUBSCRITADAS	FUNÇÕES INTERNAS	FUNÇÕES COMANDO
LAC ind,4	CALL name(arg)	TSX FNa,4 PZE arg

ind = variável que conterá o resultado da avaliação do índice.
arg = variável que conterá o resultado da avaliação do argumento.
name = nome da função chamada.
FNa = nome da função chamada.

SOMAS / SUBTRAÇÕES

	FORMA GERAL		FORMAS PARTICULARES		
	(expl) ± (exp2)		expl = const. ou variável.	exp2 = const.ou variável	expl = const.ou variavel exp2 = " " "
1	Código para avaliar expl. Resultado em T1.			Código para avaliar expl. Resultado em T1.	
2	Código para avaliar exp2. Resultado em T2.		Código para avaliar exp2. Resultado em T2.		
3	CLA T1		CLA expl	CIA T1	CIA expl
4	SOMA	FAD T2	FAD T2	FAD exp2	FAD exp2
5	SUBTRAÇÃO	FSB T2	FSB T2	FSB exp2	FSB exp2

PRODUTOS / DIVISÕES

		FORMA GERAL	FORMAS PARTICULARES		
		(exp1) * (exp2)	exp1 = const.ou variável	exp2 = const.ou variável	exp1 = const. ou variável exp2 = " " "
1		Código para avaliar exp1. Resultado em T1.		Código para avaliar exp1. Resultado em T1.	
2		Código para avaliar exp2. Resultado em T2.	idem		
3	PRODUTOS	LQD T1 FPM T2	LQD exp1 FPM T2	LDQ T1 FPM exp2	LDQ exp1 FPM exp2
4	DIVISÕES	LDQ T1 FPD T2	LDQ exp1 FPD T2	LDQ T1 FPD exp2	LDQ exp1 FPD exp2

CÓDIGO GERADO PELO COMANDO "IMPRIMA"

linha 1: espacejamento horizontal entre os elementos da lista de impressão.

linha 2: instalação do elemento no "buffer" de impressão.

CÓDIGO GERADO POR COMANDO DE ATRIBUIÇÃO

lado esquerdo do sinal "=".

linha 1: avaliação do subscrito de variável subscritada.

linha 2: não se aplica ao caso.

linha 3: atribuição pròpriamente dita.

lado direito do sinal "=".

linha 1: avaliação da expressão aritmética.

linha 2: carregar resultado da expressão no acumulador.

linha 3: não se aplica ao caso.

COMANDO IMPRIMA

	Constantes sem parte decimal	Constantes com parte decimal	Variáveis simples	Variáveis subscritadas	Expressões	Literais
1	CALL PORLIN (BRACK)					
2	CALL PORINT(const.)	CALL POREAL(const)	CALL PORNUM(VR)	Código para avaliar expressão. Resultado em T1. LAC T1.4 CIA VR.4 STO T1 CALL PORNUM(T1)	Código para avaliar expressão. Resultado em T1. CALL PORNUM(T1)	È usada a rotina PORLIN, para colocar cada seis caracteres no "buffer" da impressora.

ATRIBUIÇÃO

	LADO ESQUERDO DO SINAL '='		LADO DIREITO DO SINAL '='	
	variável simples	variável indexada	constante ou variável	expressão
1		Código para avaliar o índice. Resultado em T1. LAC T1.1		Código para avaliar a expressão. Resultado em T1.
2			CLA var	CIA T1
3	STO var	STO var,1		

var = nome da variável que comparece do lado esquerdo do sinal '='.

11. MENSAGENS DE ÊRRO

Existem dois tipos de mensagens de êrro que podem ocorrer: durante a compilação, para êrros de sintaxe ou organização do programa, e durante a execução do programa.

Durante a execução do programa BASIC, podem ocorrer mensagens devidas a testes gerados pelo compilador, como também do sistema operacional do 7044.

11.1 - Durante a Compilação

11.1.1 - *** COMANDO INVALIDO OU CONTINUAÇÃO INVALIDA DE COMANDO.

Foi detetado uma unidade sintática que não pertence a linguagem BASIC.

11.1.2 - *** FALTA COMANDO 'FIM'. INCLUIDO.

Durante a leitura do programa foi encontrado um fim de fluxo.

11.1.3 - *** NUMERO DE COMANDO NÃO DEFINIDO. NNN (I)

Número de comando referenciado em comando VA PARA ou SE ou VA SUB não está definido.

11.1.4 - *** COMANDO FUNÇÃO COM NOME REPETIDO. IGNORADO.

Foi tentada a definição de uma função com o mesmo nome de outra, anteriormente definida.

11.1.5 - *** COMANDO FUNÇÃO MAL POSICIONADO. IGNORADO.

Foi tentada a definição de um comando função após um comando executável do programa.

11.1.6 - *** NNN DEVE SER NADA COM SUBSCRITO(S). (I)

Foi feita referência a uma variável definida como indexada, sem o uso de índices.

11.1.7 - *** NNN NÃO DEVE SER SUBSCRITADA. (I)

Foi feita referência a uma variável simples, como se esta fôsse indexada com um índice.

11.1.8 - *** AGRUPAMENTO INCORRETO DE COMANDOS 'PARA' E 'PROXIMO'.IGNORADO.

A variável do comando 'PROXIMO' não é a mesma definida no comando 'PARA' correspondente.

11.1.9 - *** NÃO EXISTE O COMANDO 'PARA' PARA ESTE 'PROXIMO'. IGNORADO.

Foi encontrado um comando 'PROXIMO', sem que tenha havido um comando 'PARA' anteriormente.

11.1.10 - *** VARIAVEL DO COMANDO 'PARA' NÃO PODE SER INDEXADA. (I)

Auto explicativo.

11.1.11 - *** NUMERO DE COMANDO REPETIDO. (I)

Foi atribuido à um comando um número anteriormente usado como tal.

11.1.12 - *** COMANDO 'PARA' SEM O CORRESPONDENTE 'PROXIMO'. (I)

Não foi encontrado dentro do programa, o encerramento do laço iniciado em um comando 'PARA'.

11.1.13 - *** FALTA ')' APÓS UUUUUUU. INCLUIDO.

Faltou o fecha parênteses após uma expressão, para a qual tinha sido aberto em parênteses.

11.1.14 - *** NNN NÃO DEVE SER USADA COM DOIS SUBSCRITOS. (I)

Foi feita referência à uma variável declarada com um subscrito, como se ela tivesse dois subscritos.

11.1.15 - *** NNN NÃO DEVE SER USADA COM UM SUBSCRITO. (I)

Foi feita referência à uma variável declarada com dois subscritos, como se ela tivesse um subscrito.

11.1.16 - *** SUBSCRITO FORA DOS LIMITES ESPECIFICADOS. (I)

Um índice de uma variável indexada, escrito como uma constante, ultrapassa o limite previsto para o índice, ou é negativo, Esta mensagem também pode ocorrer ao tempo de execução do programa.

A mensagem abaixo será impressa, se durante a compilação, houver ocorrido um dos êrros assinalados com (I).

11.1.17 - *** EXECUÇÃO INIBIDA.

8.2 - Durante a Execução

11.2.1 - *** SUBSCRITO FORA DOS LIMITES ESPECIFICADOS.

O subscrito de uma variável com um índice, ou um dos subscritos de uma com dois, está fora do limite especificado no comando DIM. Se a variável não foi definida no comando DIM, o limite é de 0(zero)' a 10. A execução é interrompida.

11.2.2 - *** UUUUUU CARATER NÃO NUMÉRICO NO CARTÃO DE DADOS.

Durante a conversão de um dado numérico de um cartão de dados. Foi encontrado um carater não numérico. A execução é encerrada.

11.2.3 - Durante a execução, podem ocorrer as mensagens previstas para as rotinas internas SIN, COS, ATAN, TAN, AINT, ALOG, EXP, SQRT e .EXP3.' do FORTRAN - 7044, que correspondem, respectivamente as rotinas SEN COS, ATN, TAN, INT, LOG, EXP, RQU e a operação de potenciação do BASIC.

12. CONCLUSÕES E SUGESTÕES

Como tinhamos visto anteriormente, sempre que um compilador gera código otimizado, ele perde em eficiência. A medida usual, e adotada em muitas Universidades e centros de Computação, é manter catalogados na biblioteca do computador, dois compiladores da mesma linguagem: um que otimiza e código gerado e outro que não otimiza. Este é usado na fase de "delurgging" de programas e aquele na fase de implementação.

De qualquer forma, a otimização de código objeto ainda é assunto de discussão, não só no otimizar como principalmente como otimizar.

Quanto ao que otimizar, este está sendo encarado sob outro ponto de vista, desde o surgimento dos "compiler-compiler", que por sua própria estrutura, permitem somente um passe no programa, o que restringe, substancialmente, as possibilidades de otimizações globais.

As dúvidas em como otimizar podem ser resumidas nas seguintes questões:

a. Quais os métodos a utilizar?

b. Podem eles serem adaptados a menores e mais rápidos compiladores?

c. São estes métodos aplicáveis a tipos variados de linguagens?

A tendência geral é admitir que a análise de um programa, necessária a otimização do código objeto gerado, tem contribuido em têrmos de extensões potenciais, a saber:

a. Ela pode ser estendida para detetar irregularidades no fluxo do programa;

b. A análise do fluxo de um programa, tem contribuido mais no sentido teórico que prático, isto é, tem conduzido à compreensão da estrutura da linguagem; isso poderá, eventualmente, ser utilizada no desenvolvimento de futuras linguagens de programação.

Na maioria dos casos, os programas de tempo de execução prolongado ou uso frequente, são escritos por programadores experimentados, aos quais deve ser deixada a recomendação no sentido de que procurem otimizar o código dos programas [22], evitando, assim, a execução desnecessária de determinados comandos, a repetição de sub-expressões comuns, etc. Estas medidas, evitariam a necessidade de que se tenha um compilador que execute otimizações globais (ou pelo menos aquelas que podem ser feitas pelo programador).

Entretanto, esta situação pode-se modificar, se as pesquisas para desenvolver algoritmos eficientes de otimização global produzirem resultados melhores do que os obtidos até agora.

Esta é a nosso ver, uma área de pesquisas em que resultados positivos seriam de grande proveito prático.

REFERÊNCIAS

1. MANUAL BASIC
Third edition. Dartmouth College Computation Center - 1966

2. MANUAL "O DIA DO COMPUTADOR". LINGUAGEM BÁSICA.
RDC/PUC - 1970

3. LINGUAGEM BASICA ("BASIC") PARA O COMPUTADOR IBM-1130
Manual de uso. Marcos Glandoski, Pedro Salunbanch. Publicação interna do ITA.

4. THE COMCOM SOFTWARE WRITING SYSTEM.
Arndt Von Staa. Computer Science Department. RDC/PUC.

5. COMCOM (COMPILADOR DE COMPILADORES).
Arndt Von Staa. Grupo de Aplicações - RDC/PUC.

6. THE ANATOMY OF A COMPILER.
John A.N. Lee; Reinhocd. Computer Science Series.

7. PROGRAMMING SYSTEMS AND LANGUAGES.
Saul Rosen; Mc Graw Hill.

8. IBM 7040 AND 7044 DATA PROCESSING SYSTEMS. STUDENT TEXT.

9. PROGRAMMING LANGUAGES: HISTORY AND FUNDAMENTALS.
Jean E. Sammet; Prentice Hall.

10. MANUAL SCAN

Arndt Von Staa - PUC/RJ

11. MANUAL SUBROTINAS PORLIN, PORINT, POREAL, IMPRIME.

Arndt Von Staa - PUC/RJ

12. MANUAL CNV CNs.

Arndt Von Staa - PUC/RJ

13. GENERATION OF OPTIMAL CODE FOR EXPRESSION VIA FACTORIZATION

Nelvin A. Breuer - CACOS Volume 12/Number 6/June, 1969.

14. OPTIMAL CODE FOR SERIAL AND PARALLEL COMPUTATION

Richard J. Fateman - CACM Volume 12/Number 12/December,1969.

15. INTRODUCTION TO NON-NUMERIC COMPUTATION

Faculty of Mathematics - University of Waterloo - 1970.

16. OBJECT CODE OPTIMIZATION

Edward S. Lowry and C.W. Medlock - CACM Volume 12/Number 12/January, 1969.

17. NOTES ON THE ACM COMPUTER OPTIMIZATION SYMPOSIUM,URBANA.

Peter Wegner. - CACM Volume 13/Number 10/October, 1970.

18. OPTIMIZATION OF EXPRESSIONS IN FORTRAN

Vicent A. Busam and Donald E. Englund - CACM Volume 12/Number 12/December, 1969.

19. COMPILING TECHNIQUES
F.R.A. Hopgood - MacDonald Computer Monographs.

20. A FORMALISM FOR TRANSLATOR INTERACTIONS.
J. Earley and H. Sturgis. CACM Vol.13 - October 1970.

21. PL/I OPTIMIZING COMPILER
IBM System/360 Operating System.
Program Number 5734-PL1
File No. S360-29
Order No. GC33-0001-0

22. BOLETIM TÉCNICO Nº 1 RDC-PUC/RJ.

23. MANUAL DE FORTRAN-IV. SISTEMA IBM-7044.
RIO DATACENTRO.

24. COMPILER CONSTRUCTION FOR DIGITAL COMPUTERS.
David Gries. Cornell University - John Wiley - 1971.

25. PROGRAM OPTIMIZATION. ANNUAL REVIEW IN AUTOMATIC PROGRAMMING. VOL.5.
Pergamon, New York.

APÊNDICE

LISTAGEM DO COMPILADOR

```
DECLARE UNARY(15) FIXED, FUNCAO CHAR(144),V FIXED.,
DECLARE FINT FIXED, INTG FIXED, AST CHAR(1).,
DECLARE M CHAR(6), ARG CHAR(12), IN FIXED, NLI FIXED, NC FIXED.,
DECLARE A2 CHAR(12), ZZ CHAR(6), TEMP1 CHAR(12), E1 FIXED.,
DECLARE  XY FIXED, PAA CHAR(4).,
DECLARE NOGO FIXED, NREAD FIXED,LREAD FIXED, XREAD FIXED.,
DECLARE FIXO FIXED , TIND FIXED.,
DECLARE FUNC(16) CHAR(4),PEXP3P FIXED, PODE FIXED, VFN CHAR(12),T FIXED
      , NDO FIXED, TOTDO FIXED, INP FIXED, OUT FIXED, NITEM FIXED.,
DECLARE POWER(40) FIXED, POT(40) FIXED.,
DECLARE                    ATRIB(100) FIXED, TBEND FIXED, I FIXED.,
 ECLARE TEMP FIXED, OPN(40) CHAR(12), OPR(40) CHAR(6),
        LEV FIXED, OPL(40) CHAR(6), TMP(20) FIXED, JCN(20) FIXED,
        KCN(20) FIXED, J1 FIXED, K1 FIXED,
        WHERE FIXED, OPND CHAR(12), VAR CHAR(6), X FIXED,
       A3(15) CHAR(10), ENDO(15) FIXED, PMAX FIXED, TNOME(15) CHAR(6),
        NTEMP FIXED, IFTEMP FIXED,X1 FIXED, MASK FIXED,
        Z CHAR(2), XMAX FIXED, J FIXED, K FIXED.,
/* ------------------------------------------------------------------ */
/* KEYS DEFINITION */
/* ------------------------------------------------------------------ */
/* DECLARACAO DE SIMBOLOS RESERVADOS */
KEY RSVD = (LOP OR ''ENTAO'' OR ''ATE'' OR ''DEGRAU'' OR FNAME  OR
          ''SUB'' OR FST OR START).,
KEY START = (''DIM'' OR ''VA'' OR ''VAPARA'' OR ''SE'' OR ''PARA'' OR
           ''PROXIMO'' OR ''FIM'' OR ''PARE'' OR ''LEIA'' OR
           ''IMPRIMA'' OR ''VASUB'' OR ''RETORNE'' OR ''DEF'' OR
           ''ARIT'' OR ''COM'').,
/* ------------------------------------------------------------------ */
/* ESTA KEY RECONHECE COMANDOS INVALIDOS OU CONTINUACOES INVALIDAS.  */
KEY INVLD = 0-(PART).,
KEY PART = (START OR $U) BEGIN.,
                         IF START NE '' THEN CALL $FAIL., END., .,
KEY FNAME = (''SEN'' OR ''COS'' OR ''TAN'' OR ''ATN'' OR ''EXP'' OR
           ''ABS'' OR ''LOG'' OR ''RQU'' OR ''INT'' ).,
KEY VR = (RSVD OR $I) BEGIN., IF RSVD NE '' THEN CALL $FAIL., END., .,
KEY INDV1 = VR ''('' AEXP '')''.,
KEY INDV2 = VR ''('' AEXP1 '','' AEXP2 '')''.,
KEY BUILTIN = FNAME ''('' AEXP '')''.,
KEY FUNCT = FST ''('' AEXP '')''.,
KEY LOP = (''MEN'' OR ''MEG'' OR ''MAI'' OR ''MAG'' OR ''IGU'' OR
''DIF'').,
/* ------------------------------------------------------------------ */
/* RECONHECEDOR DE COMANTARIOS */
KEY COM = ''COM'' 0-((RSVD OR $U) BEGIN., IF RSVD NE '' THEN CALL $FAIL
                                  ., END.,) .,
 * ------------------------------------------------------------------ */
 * RECONHECEDOR PARA O COMANDO 'PARA' */
 EY FOR = (LAB OR) ''PARA'' VR ''='' AEXP1 ''ATE'' AEXP2
          0-1(''DEGRAU'' AEXP3).,
  Y LAB = $C.,
  Y AEXP1 = AEXP.,
  Y AEXP2 = AEXP.,
  Y AEXP3 = AEXP.,
```

```
/* ------------------------------------------------------------------ */
/* RECONHECEDOR DO COMANDO 'PROXIMO' */
KEY NEXT = 0-1(LAB) ''PROXIMO'' VR.,
/* ------------------------------------------------------------------ */
/* RECONHECEDOR DO COMANDO 'LEIA' */
KEY READ = (LAB OR) ''LEIA'' READLIST.,
KEY READLIST = VARY 0-(SET INDEX TO NREAD '','' VARY).,
KEY VARY =  (IND1 OR IND2 OR VR).,
KEY IND1= VR ''('' AEXP ''),
KEY IND2 = VR ''('' AEXP1 '','' AEXP2 '')''.,
/* ------------------------------------------------------------------ */
/* RECONHECEDOR DO COMANDO 'FIM' */
KEY FIM = 0-1(LAB) ''FIM''.,
/* ------------------------------------------------------------------ */
/* RECONHECEDOR DO COMANDO 'RETORNE' */
KEY RET = 0-1(LAB) ''RETORNE''.,
/* ------------------------------------------------------------------ */
/* RECONHECEDOR DO COMANDO 'VA SUB' */
KEY GOSUB = 0-1(LAB) (''VA'' ''SUB'' OR ''VASUB'') GTO.,
/* ------------------------------------------------------------------ */
/* RECONHECEDOR DO COMANDO 'SE ENTAO' */
KEY IFTHEN = 0-1(LAB) ''SE'' AEXP1 LOP AEXP2 ''ENTAO'' GTO.,
KEY GTO = $C.,
/* ------------------------------------------------------------------ */
/* RECONHECEDOR DO COMANDO 'VA PARA' */
KEY GOTO = 0-1(LAB) (''VA'' ''PARA'' OR ''VAPARA'') GTO.,
/* ------------------------------------------------------------------ */
/* THIS KEY RECOGNIZES THE 'IMPRIMA' STATEMENT */
KEY PRINT = (LAB OR) ''IMPRIMA'' PRINTLIST.,
KEY PRINTLIST = (ITEM 0-(SET INDEX TO NITEM '','' ITEM) OR).,
KEY ITEM = (AEXP OR $L).,
/* ------------------------------------------------------------------ */
/* THIS KEY RECOGNIZES THE 'DIM' DECLARATION */
KEY DIM = (LAB OR) ''DIM'' DVAR 0-('','' DVAR).,
KEY DVAR = (DIMVAR1 OR DIMVAR2).,
KEY DIMVAR1 = VR ''('' $C ''),
KEY DIMVAR2 = VR ''('' C1 '','' C2 '')''.,
KEY C1 = $C.,
KEY C2 = $C.,
/* ------------------------------------------------------------------ */
/* THIS KEY RECOGNIZES FUNCTION STATEMENT DECLARATION */
KEY DEFN = (LAB OR) ''DEF'' FST ''('' VR '')'' ''='' AEXP.,
KEY FST = (''FNA'' OR ''FNB'' OR ''FNC'' OR ''FND'' OR ''FNE'' OR
           ''FNF'' OR ''FNG'' OR ''FNH'' OR ''FNI'' OR ''FNJ'' OR
           ''FNK'' OR ''FNL'' OR ''FNM'' OR ''FNN'' OR ''FNO'' OR
           ''FNP'' OR ''FNQ'' OR ''FNR'' OR ''FNS'' OR ''FNT'' OR
           ''FNU'' OR ''FNV'' OR ''FNW'' OR ''FNX'' OR ''FNY'' OR
           ''FNZ'').,
/* ------------------------------------------------------------------ */
/* THIS KEY RECOGNIZES ARITHMETICS EXPRESSIONS */
KEY AEXP = (''-'' OR ''+'' OR) TERM 0-(AOP TERM).,
KEY TERM= FACT 0-( MOP FACT ).,
KEY FACT = PRIM 0-(EOP PRIM).,
KEY AOP= (''+'' OR ''-'').,
KEY MOP= (''*'' OR ''/'').,
```

```
KEY EOP = ''**''.,
KEY FPA= '')'',
KEY PRIM =(CT OR BUILTIN OR FUNCT OR INDV1 OR INDV2 OR VR OR ''('' AEXP
          (FPA OR BEGIN., OUTPUT ' *** FALTA '')'' APOS ', AEXP.,
                          END., FOUND FPA '')'' )).,
KEY CT = (CR OR CI).,
KEY CI = (''-'' OR ''+'' OR) $C.,
KEY CR = (''-'' OR ''+'' OR) ($C ''.'' $C ''E'' (''-'' OR ''+'' OR) $C
        OR $C ''.'' $C OR ''.'' $C ''E'' (''-'' OR ''+'' OR) $C OR
        ''.'' $C OR $C ''.'' ''E'' (''-'' OR ''+'' OR) $C OR
        $C ''.'' ).,
KEY VARV = (INDV1 OR INDV2 OR VR).,
/* ------------------------------------------------------------------ */
/* FUNCTIONS DECLARATIONS */
/* ------------------------------------------------------------------ */
/* THESE FUNCTIONS PRINTS ERRORS MESSAGES FOR SUBSCRIPTED VARIABLES */
ER1.. FUNCTION ERRO1(A KEY).,
      OUTPUT' *** ',A,' NAO DEVE SER USADA COM DOIS SUBSCRITOS.'.,
END ER1.,
ER2.. FUNCTION ERRO2(A KEY).,
      OUTPUT' *** ',A,' NAO DEVE SER SUBSCRITADA.'.,
END ER2.,
ER3.. FUNCTION ERRO3(A KEY).,
      OUTPUT' *** ',A,' DEVE SER USADA COM SUBSCRITO(S).'.,
END ER3.,
ER4.. FUNCTION ERRO4(A KEY).,
      OUTPUT' *** ',A,' NAO DEVE SER USADA COM UM SUBSCRITO.'.,
END ER4.,
/* ------------------------------------------------------------------ */
FTEMPOR.. FUNCTION TEMPOR(A CHAR,NUM FIXED) CHAR .,
          RETURN A CAT $SUBST($TOCHA(NUM),8,5).,
END FTEMPOR.,
/* ------------------------------------------------------------------ */
/* FUNCAO PARA PROCES SAR NUMERO D E COMANDO */
F1.. FUNCTION INSTLAB (A  FIXED ).,
     M = TEMPOR('T',A).,
     T = $IHASH(M).,
     IF T LT 1 THEN DO.,
                    ATRIB(-T) = 99999.,
                    OUTCOD M, ' BSS     0'.,
                    RETURN.,
                    END.,
     IF ATRIB(T) EQ 100000 THEN DO.,
     IF ATRIB(T) EQ 100000 THEN DO.,ATRIB(T) = 99999.,
                                ATRIB(T) = 99999.,
                                OUTCOD M, ' BSS     0'.,
                                RETURN.,
                                END.,
     IF ATRIB(T) EQ 99999 THEN DO.,
        OUTPUT' *** NUMERO DE COMANDO REPETIDO.',A.,
        NOGO = 1.,   END.,
END F1.,
/* ------------------ ------------------------------------------------ */
/* THIS FUNCTION CRIATES THE TEMPORARIES */
FE1.. FUNCTION INCREM CHAR.,
        Z=$SUBST(X,11,2).,  X=X+1.,
```

```
        X1 = $IHASH('T.' CAT Z).;
        IF X1 LT 1 THEN ATRIB(-X1)=0.;
        RETURN Z.;
END FF1.;
/* ------------------------------------------------------------------ */
/* THIS FUNCTION STORES THE RESULT OF OPERATIONS WITH FUNCTIONS
   IN A TEMPORARY VARIABLE */
F100.. FUNCTION FUNF.;
      IF WHERE = 1 THEN DO.;
                        OPND = 'T.' CAT INCREM.;
                        OUTCOD '       STO     ', OPND.;
                        END.;
      IF WHERE = 2 THEN DO.;
                        OPND = 'T.' CAT INCREM.;
                        OUTCOD '       STO     ',OPND.;
                        END.;
      END F100.;
/* ------------------------------------------------------------------ */
/* ESTA FUNCAO INSTALA NA TABELA DE SIMBOLOS, OS NUMEROS DE COMANDO
  DEFINIDOS NOS COMANDOS 'VA PARA' E 'VA SUB' E 'SE ENTAO' */
FGTO.. FUNCTION INSTGTO(PGTO FIXED).;
      M = TEMPOR('T',PGTO).;
      T = $IHASH(M).;
      IF T LT 1 THEN ATRIB(-T) = 100000.;
END FGTO.;
/* ------------------------------------------------------------------ */
/* ------------------------------------------------------------------ */
FT.. FUNCTION TEST(A KEY).;
IF (A.AOP NE '' OR
   A.MOP NE '' OR
   A.EOP NE '' OR
   '('   EQ A  OR
   A.INDV1 NE '' OR
   A.INDV2 NE '' OR
   A.BUILTIN NE '' OR
   A.FUNCT NE '' ) THEN RETURN 0.;
IF A.VR NE '' THEN RETURN 1.;
IF A.CI NE '' THEN RETURN 2.;
RETURN 3.;
END FT.;
/* ------------------------------------------------------------------ */
FIX.. FUNCTION TOFIXED(MAX FIXED).;
      TIND = 1.;
      /* INDEX IS AN EXPRESSION */
      IF WHERE LE 2 THEN D1.. DO.;
         OPND = 'T.' CAT INCREM.;
         IF WHERE = 1 THEN OUTCOD '       STO     ',OPND.;
                      ELSE OUTCOD '       STQ     ',OPND.;
         OUTCOD '        CALL    TOFIX(',$CMPCT(OPND),',',$CMPCT(OPND)
                        ,')'.;
         OUTCOD '        CALL    TINDEX(',$CMPCT(OPND),',=',MAX  ,')'.;
         FIXO=1.;
         RETURN.;
      END D1.;
      /* INDEX IS A VARIABLE */
```

```
      IF '=' NE $SUBST(OPND,1,1) THEN D2.. DO.,
         ARG=OPND.,
         OPND='T.' CAT INCREM.,
         OUTCOD '       CALL    TOFIX(',$CMPCT(ARG),',',$CMPCT(OPND),
                        ')'.,
         OUTCOD '       CALL    TINDEX(',$CMPCT(OPND),',=',MAX  ,')'.,
         FIXO=1.,
         RETURN.,
      END D2.,
      /* INDEX IS A CONSTANT */
      IN = $INDEX(OPND,'.') - 1.,
      OPND=$SUBST(OPND,1,IN).,
      IF FINT = 1 THEN GO TO INTEST.,
      INTG = $TOFIX($SUBST(OPND,2,IN-1)).,
      INTEST.. IF INTG GT MAX OR INTG LT 0 THEN
            OUTPUT' *** SUBSCRITO FORA DOS LIMITES ESPECIFICADOS.'.,
      RETURN.,
END TFIX.,
/* ------------------------------------------------------------------ */
F2.. FUNCTION RESULT.,
        TEMP=TEMP+1., OPR(TEMP)=OPL(LEV).,
        POWER(TEMP)=POT(LEV)., POT(LEV) = 0.,
        IF WHERE LE 2 THEN DO.,
             OPND='T.' CAT INCREM.,
             IF WHERE LE 1 THEN OUTCOD '       STO     ', OPND.,
                           ELSE OUTCOD '       STQ     ', OPND.,
             END.,
        OPN(TEMP)=OPND.,
END F2.,
/* ------------------------------------------------------------------ */
FR.. FUNCTION ANSWER.,
     CALL RESULT.,
     DO XY=TMP(LEV)+1 TO TEMP.,
        IF OPN(XY) = VFN THEN DO.,
          IF (OPR(XY) = 'FPM   ' OR OPR(XY) = 'FPD   '   )
             THEN AST = 'I'.,
             ELSE AST = '*'.,
          OPR(XY)=$SUBST(OPR(XY),1,3) CAT AST.,
          OPN(XY)='1,4   '.,
          END.,
        OUTCOD '       ',OPR(XY),'  ',OPN(XY).,
     IF POWER(XY) = 1 THEN DO.,
        OUTCOD '       TSL     .EXP3.'.,
        PEXP3P = 1 .,
        POWER(XY)=0.,
        END.,
ND FR.,
* ------------------------------------------------------------------ */
ARITHMETICS.. FUNCTION'ARITEXPR(A KEY).,
* ------------------------------------------------------------------ */
6.. FUNCTION PRIMEXPR(A KEY).,
        IF '('=A THEN D2.. DO.,
             CALL ARITEXPR(A.AEXP)., GO TO UNT.,
        END D2.,
        PAA=$SUBST('*' CAT $TOCHA(A) CAT '**',1,4).,
```

```
      I = $INDEX(FUNCAO,PAA).,
      IF I = 0 THEN GO TO BLACK.,
/* COMPILE A BUILT-IN FUNCTION CALL */
      IF I LE 36 THEN D4.. DO.,
     I=$JHASH(A.FNAME).,
     IF I LT 1 THEN OP1.. DO.,
          LEV = LEV +1.,  OPL(LEV) = 'ABS'.,
          CALL ARITEXPR(A.AEXP).,
      CALL FUNF.,
          OUTCOD '        CLA     ',OPND.,
          OUTCOD '        SSP'.,
          GO TO UT.,
     END OP1.,
     T = I.,
     ATRIB(T)=80001.,
      LEV = LEV +1., OPL(LEV) = FUNC(T).,
     CALL ARITEXPR(A.AEXP).,
      CALL FUNF.,
     OUTCOD'        CALL    ',$CMPCT(OPL(LEV)) CAT '(' CAT
     $CMPCT(OPND) CAT ')'.,
     UT.. OPND = 'T.' CAT INCREM.,
     OUTCOD '        STO     ',OPND.,
     LEV = LEV - 1.,   WHERE = 3.,
     GO TO UNT.,
     END D4.,
/* COMPILE A FUNCTION STATEMENT CALL */
     ELSE D5.. DO.,
     I=$JHASH(A.FST).,
     IF I LT 1 THEN DO.,
               OUTPUT' *** FUNCAO COMANDO NAO DEFINIDA.'.,
               NOGO=1.,   RETURN.,   END.,
               ELSE  D6.. DO.,
               LEV = LEV +1., OPL(LEV) = A.FST.,
               CALL ARITEXPR(A.AEXP).,
      CALL FUNF.,
               OUTCOD '        TSX     ',$CMPCT(A.FST) CAT ',4'.,
               OUTCOD '        PZE     ',OPND.,
               GO TO UT.,
               END D6.,
     END D5.,
/* COMPILE A ONE SUBSCRIPTED VARIABLE */
LACK..  IF A.INDV1 NE '' THEN D1.. DO.,
          I=$IHASH(A.INDV1.VR)., IF I LT 1 THEN DO.,
               ATRIB(-I)=10., GO TO O1., END.,
          IF ATRIB(I)=0 THEN DO.,
             CALL ERRO2(A.VR).,
             NOGO=1., RETURN.,
             END.,
     O1..  LEV=LEV+1., OPL(LEV)=A.VR.,
     CALL ARITEXPR(A.AEXP).,
     CALL TOFIXED(ATRIB($ABS($JHASH(A.VR)))).,
     TEMP=TEMP+1.,
     OPR(TEMP)='LAC'.,
     OPN(TEMP)=$CMPCT(OPND CAT ',1').,
          OPND=$CMPCT(OPL(LEV) CAT   ',1').,
```

```
            LEV=LEV-1., WHERE=4., GO TO UNT.,
        END D1.,
* COMPILE A TWO SUBSCRIPTED VARIABLE */
    IF A.INDV2 NE '' THEN D7.. DO.,
       I=$IHASH(A.INDV2.VR).,
       IF I LT 1 THEN DO.,
          ATRIB(-I) = 1000010.,
          GO TO Z1.,
          END.,
       IF ATRIB(I)=0 THEN DO.,
           CALL ERRO2(A.VR).,
          RETURN.,
          END.,
       IF ATRIB(I) LE 99999 THEN DO.,
       CALL ERRO1(A.VR).,
          RETURN.,   END.,
* EVALUATE THE 1ST INDEX */
 Z1.. LEV=LEV+1.,   OPL(LEV)=A.VR.,
    I=$ABS(I).,
    NLI=ATRIB(I)/100000.,
    NC=$MOD(ATRIB(I),100000).,
    CALL ARITEXPR(A.INDV2.AEXP1).,
    CALL TOFIXED(NLI).,
    XREAD=1.,
    OUTCOD '       CLA     ',$CMPCT(OPND).,
    OUTCOD '       STO     TREAD'.,
* OK. EVALUATE THE 2ND INDEX */
    CALL ARITEXPR(A.INDV2.AEXP2).,
    CALL TOFIXED(NC).,
    OUTCOD '       STQ     =',NLI.,
    OUTCOD '       MPY     ',OPND.,
    OUTCOD '       LLS     35'.,
    OUTCOD '       ADD     TREAD'.,
    OUTCOD '       STO     TREAD'.,
    TEMP=TEMP+1.,
    OPR(TEMP)='LAC'.,   OPN(TEMP)='TREAD,1'.,
    OPND=$CMPCT(OPL(LEV) CAT ',1').,
    LEV=LEV-1.,   WHERE=4.,   GO TO UNT.,
 END D7.,
  /* COMPILE A SIMPLE VARIABLE */
       IF A.VR NE '' THEN D3.. DO.,
          I=$IHASH(A.VR)., IF I LT 1 THEN ATRIB(-I)=0.,
               ELSE IF ATRIB(I) NE 0 THEN   D8.. DO.,
              CALL ERRO3(A.VR).,
              NOGO=1.,   RETURN.,            END D8.,
          OPND=A.VR., WHERE=3., GO TO UNT.,
       END D3.,
  /* COMPILE A CONSTANT */
 FINT = 0.,
 IF A.CT.CI  NE  '' THEN DO.,
                     OPND='=0' CAT $AJUST(A.CI,'0') CAT '.0' .,
                     INTG = $TOFIX(A.CT.CI).,
                     FINT = 1.,
                     END.,
                     ELSE OPND='=0' CAT $AJUST(A.CR,'0').,
```

```
        WHERE =3.,
     /* COMPILATION OF UNARY MINUS */
NT..    IF UNARY(V) = 1 THEN D10.. DO.,
            IF WHERE GT 2 THEN D9.. DO.,
                OUTCOD'       CLS     ',OPND.,
                OPND = 'T.' CAT INCREM.,
                OUTCOD'       STO     ',OPND.,
                UNARY(V) = 0.,
                WHERE=3.,
                RETURN.,
                END D9.,
            IF WHERE = 2 THEN DO.,
                OUTCOD'       LLS     35'.,
                              END.,
                OUTCOD'       CHS'.,
            UNARY(V)=0.,
            END D10.,
            RETURN.,
ND F6.,
* ------------------------------------------------------------- */
* THIS FUNCTION COMPILES FACTORS */
7.. FUNCTION FACTEXPR(A KEY).,
EV=LEV+1., TMP(LEV)=TEMP.,JCN(LEV)=L.,OPL(LEV)='CLA'.,KCN(LEV)=L1.,
.,L1=0.,
71.. L=L+1.,L1=L1+1.,CALL PRIMEXPR(A.PRIM(L)).,
/* FIM DA SEQUENCIA DE POTENCIAS */
IF A.EOP(L1) = '' THEN D1.. DO.,
IF TMP(LEV) NE TEMP THEN D2.. DO.,
IF TMP(LEV)=TEMP-1 AND OPR(TEMP)='LAC'  THEN GO TO F72.,
      CALL ANSWER., WHERE=1.,
END D2., TEMP=TMP(LEV).,
F72.. L=JCN(LEV)+L., L1=KCN(LEV)+L1-1., LEV=LEV-1.,
RETURN., END D1.,
 * PROXIMO PRIMARY */
CALL RESULT., OPL(LEV)='LDQ'., POT(LEV) = 1., GO TO F71.,
END F7.,
/* ------------------------------------------------------------- */
/* THIS FUNCTION COMPILES TERMS */
5.. FUNCTION TERMEXPR(A KEY).,
      LEV=LEV+1., TMP(LEV)=TEMP., JCN(LEV)=K., OPL(LEV)='LDQ'.,
      KCN(LEV)=K1., K,K1=0.,
1..   K=K+1., K1=K1+1., CALL FACTEXPR(A.FACT(K)).,
   /* END OF A PRODUCTORY */
      IF A.MOP(K1)= '' THEN D1.. DO.,
          IF TMP(LEV) NE TEMP THEN D2.. DO.,
              IF TMP(LEV)=TEMP-1 AND OPR(TEMP)='LAC   '
                 THEN GO TO F52.,
              CALL ANSWER., WHERE=2.,
          END D2., TEMP=TMP(LEV).,
   F52..  K=JCN(LEV)+K., K1=KCN(LEV)+K1-1., LEV=LEV-1.,
          RETURN.,
      END D1.,
   /* NEXT FACTOR OF A PRODUCTORY */
      CALL RESULT.,
      IF '*' EQ A.MOP(K1) THEN OPL(LEV)='FPM'.,
```

```
                        ELSE D3.. DO., OPL(LEV)='FPD'.,
                    IF OPR(TEMP)='FPM   ' THEN GO TO F51.,
                END D3.,
            GO TO F51.,
END F5.,
/* ------------------------------------------------------------------ */
/* MAIN FUNCTION, COMPILATION OF SUMS OF PRODUCTS */
            LEV=LEV+1., TMP(LEV)=TEMP., JCN(LEV)=J., OPL(LEV)='CLA'.,
            KCN(LEV)=J1., J,J1=0.,
            V = V+1.,
            IF '-' = A THEN UNARY(V) = 1.,
                       ELSE UNARY(V) = 0.,
41..        J=J+1., J1=J1+1., CALL TERMEXPR(A.TERM(J)).,
            V = V-1.,
    /* END OF A SUMMATORY */
            IF A.AOP(J1) EQ '' THEN D1.. DO.,
                IF TMP(LEV) NE TEMP THEN D2.. DO.,
                    CALL ANSWER., WHERE=1.,
                END D2., TEMP=TMP(LEV).,
          J=JCN(LEV)+J., J1=KCN(LEV)+J1-1., LEV=LEV-1.,
                OUTCOD '*** ', A.,
                RETURN.,
            END D1.,
    /* NEXT TERM OF A SUMMATORY */
            CALL RESULT.,
            IF '+'=A.AOP(J1) THEN OPL(LEV)='FAD'.,
                             ELSE OPL(LEV)='FSB'.,
            GO TO F41.,
END ARITHMETICS.,
/* ------------------------------------------------------------------ */
/* MACRO DECLARATIONS */
/* ------------------------------------------------------------------ */
    /* THIS MACRO COMPILES THE END STATEMENT */
AND..       MACRO (FIM OR '' '''EOF''' '')).,
            IF FIM EQ '' THEN OUTPUT ' FALTA COMANDO ''FIM''. INCLUIDO.',
            OUTCOD '        TRA     S.JXIT'.,
    /* OUTPUT OF THE SYMBOL TABLE */
           E1 = $LMLOC.,
           X = 0.,
           D1.. X = X + 1.,
            VAR=$SYMBO(X).,
           VALUE=ATRIB(X).,
                IF VALUE=99999 OR VALUE=90000 OR VALUE=80000 THEN
                        GO TO FD1.,
                IF VALUE=80001 THEN DO.,
                        OUTCOD '        EXTERN  ',FUNC(X).,
                        GO TO FD1.,
                        END.,
                IF VALUE=100000 THEN DO.,
                         OUTPUT ' *** NUMERO DE COMANDO NAO DEFINIDO ',
                                $SYMBO(X).,
                         GO TO FD1.,
                         END.,
                IF VALUE = 0 THEN DO.,
                         OUTCOD VAR,' BSS     1'.,
```

```
                    GO TO FD1.,
                    END.,
          IF VALUE GT 100000 THEN DO.,
                    SIZE=VALUE/100000*$MOD(VALUE,100000).,
                    OUTCOD VAR,' BSS      ',SIZE.,
                    GO TO FD1.,
          END.,
           OUTCOD VAR,' BSS      ',VALUE.,
      FD1.. IF X LT E1 THEN GO TO D1.,
  /* OUTPUT OF TEMPORARIES AND INTERNAL CONSTANTS */
IF OUT = 1 THEN DO.,
          OUTCOD'       EXTERN  IMPRME,PORLIN,POREAL,PORINT'.,
          OUTCOD 'BRACK  OCT     363660606060'.,
          END.,
OUTCOD '        EXTERN  LRESET'.,
IF INP = 1 THEN DO.,
          OUTCOD'       EXTERN  BSCRD'.,
          OUTCOD'R      BSS     1'.,
          OUTCOD '       PZE     **,,SAVE.'.,
          OUTCOD '       PZE     SAVE.,,**'.,
          OUTCOD 'SAVE.  BSS     20'.,
          END.,
IF IFTEMP=1 THEN OUTCOD 'K00    BSS     1'.,
IF XREAD=1 THEN OUTCOD'TREAD  BSS     1'.,
IF FIXO=1 THEN DO.,
               OUTCOD '       EXTERN  TOFIX'.,
               OUTCOD '       EXTERN  TINDEX'.,
               END.,
IF PEXP3P = 1 THEN OUTCOD'       EXTERN  .EXP3.'.,
   OUTCOD '       END'., STOP.,
IF TOTDO NE 0 THEN OUTPUT' *** FALTA(M)  ',TOTDO,'  COMANDO(S) ''PROXIM
O''.'.,
IF NOGO = 1 THEN OUTPUT' *** EXECUCAO INIBIDA.'.,
END MAND.,
/* ------------------------------------------------------------------ */
/* THIS MACRO COMPILES THE 'PARA' STATEMENT */
FOR.. MACRO FOR.,
      PODE = 1.,
      IF LAB NE '' THEN CALL INSTLAB(LAB).,
      NDO=NDO+1., TOTDO=TOTDO+1.,
      /* COMPILATION OF THE FIRST EXPRESSION */
      TEMP,LEV=0., X=1.,
      CALL ARITEXPR(AEXP1.AEXP).,
      IF TEMP NE 0 THEN OUTCOD '       ',OPR(1),'  ',OPN(1).,
      IF WHERE GT 2 THEN D1.. DO.,
                            OUTCOD '       CLA     ',OPND., GO TO
                            D2.,
                       END D1.,
      /* COMPILE THE ASSIGNMENT */
      IF WHERE LE 1 THEN
                  D2.. OUTCOD '       STO     ',VR.,
                  ELSE OUTCOD '       STQ     ',VR.,
      I=$IHASH(VR).,
      IF I LT 1 THEN ATRIB(-I)=0.,
                ELSE IF ATRIB(I) NE 0 THEN DO.,
```

```
                    OUTPUT ' *** VARIAVEL DO COMANDO ''PARA'' E'' IND'
                                ,'EXADA.'., GO TO FMFOR.,
                                           END.,
      TNOME(TOTDO)=VR., ENDO(TOTDO)=NDO.,
      /* COMPILATION OF THE SECOND EXPRESSION */
      I=TEST(AEXP2).,
      IF I GE 1 THEN DO.,
                IF I=1 THEN A2=$CMPCT(AEXP2).,
           IF I=2 THEN A2='=' CAT '0' CAT $AJUST(AEXP2,'0') CAT '.0'.,
                ELSE A2='=' CAT '0' CAT $AJUST(AEXP2,'0').,
         GO TO C3.,
         END.,
      NTEMP=NTEMP+1.,
      X1=$IHASH(TEMPOR('P',NTEMP)).,
      IF X1 LT 1 THEN ATRIB(-X1)=0.,
      TEMP,LEV=0., X=1.,
      CALL ARITEXPR(AEXP2.AEXP).,
      IF TEMP NE 0 THEN OUTCOD '        ',OPR(1),'  ',OPN(1).,
      /* STORE THE RESULT IN A TEMPORARY */
      IF WHERE LE 1 THEN
                        OUTCOD '        STO     ',TEMPOR('P',NTEMP).,
                     ELSE OUTCOD '        STQ     ',TEMPOR('P',NTEMP).,
      A2=TEMPOR('P',NTEMP).,
      /* COMPILATION OF THE THIRD EXPRESSION, IF ANY */
C3..  IF AEXP3 NE '' THEN P1.. DO.,
      I=TEST(AEXP3).,
      IF I GE 1 THEN DO.,
      IF I = 1 THEN DO., A3(TOTDO)=$CMPCT(AEXP3)., GO TO TN., END.,
      IF I = 2 THEN A3(TOTDO)= '=0' CAT $AJUST(AEXP3,'0') CAT '.0'.,
               ELSE A3(TOTDO)= '=0' CAT $AJUST(AEXP3,'0') .,
      GO TO NS.,
      END.,
                    TEMP,LEV=0., X=1.,
                    CALL ARITEXPR(AEXP3.AEXP).,
      NTEMP=NTEMP+1.,
      X1=$IHASH(TEMPOR('P',NTEMP)).,
      IF X1 LT 1 THEN ATRIB(-X1)=0.,
                    IF TEMP NE 0 THEN
                                  OUTCOD '        ',OPR(1),'  ',OPN(1).,
      /* STORE THE RESULT IN A TEMPORARY */
      IF WHERE LE 1 THEN
                        OUTCOD '        STO     ',TEMPOR('P',NTEMP).,
                     ELSE OUTCOD '        STQ     ',TEMPOR('P',NTEMP).,
      A3(TOTDO)=TEMPOR('P',NTEMP).,
      /* GENERATE TEST FOR NEGATIVE INCREMENT */
TN..  OUTCOD '        TPL     *+4'.,
      OUTCOD '        CLA     *+5'.,
      OUTCOD '        SSP     '.,
      OUTCOD '        STO     *+3'.,
                       END P1.,
      /* INCREMENT NOT SPECIFIED.  1.0 TAKEN. */
                    ELSE A3(TOTDO)='=1.0'.,
NS..  OUTCOD TEMPOR('D',NDO),      ' BSS     0'.,
       OUTCOD '        CLA     ',A2.,
      OUTCOD '        FSB     ',VR.,
```

```
      OUTCOD '       UFA     =0233000000000'.,
      OUTCOD '       ALS     10'.,
      OUTCOD '       ARS     10'.,
      OUTCOD '       TMI     ',TEMPOR('E',NDO) CAT '+1'.,
FMFOR.. END MFOR.,
/* ------------------------------------------------------------------ */
/* ESTA MACRO COMPILA O COMANDO 'PROXIMO' */
MEND.. MACRO NEXT.,
      PODE = 1.,
      IF TOTDO LE 0 THEN DO.,
                  OUTPUT' *** NAO EXISTE O COMANDO ''PARA'' PARA',
                      ' ESTE ''PROXIMO''. IGNORADO.'.,
                      GO TO FMEND.; END.;;;
      IF LAB NE '' THEN CALL INSTLAB(LAB).,
      ZZ = VR.,
      IF TNOME(TOTDO) NE ZZ THEN DO.,
                           OUTPUT' ***  AGRUPAMENTO INCORRETO DE',
                               ' COMANDOS ''PARA E ''PROXIMO''.'.,
                               GO TO FMEND., END., .,
      OUTCOD '       CLA     ', ZZ.,
      OUTCOD '       FAD     ', A3(TOTDO).,
      OUTCOD '       STO     ', ZZ.,
      OUTCOD TEMPOR('E',ENDO(TOTDO)), ' TRA     ',
             TEMPOR('D',ENDO(TOTDO)).,
      IF A3(TOTDO) = '=1.0' THEN NTEMP=NTEMP-1.,
                            ELSE NTEMP=NTEMP-2.,
      TOTDO = TOTDO -1 .,
FMEND.. END MEND.,
/* ------------------------------------------------------------------ */
/* ESTA MACRO COMPILA O COMANDO 'SE ENTAO' */
MIFTHEN.. MACRO IFTHEN.,
         DECLARE COMPAR(6) LABEL INITIAL, CODCMP CHAR(21).,
      PODE = 1.,
         CODCMP = 'IGUMAIMENMEGMAGDIF   '.,
         CALL INSTGTO(GTO).,
         IF LAB NE '' THEN CALL INSTLAB(LAB).,
         E1=0.,
         TEMP,LEV=0., X=1.,
         CALL ARITEXPR(AEXP1).,
         IF TEMP NE 0 THEN
         OUTCOD '       ',OPR(1),'  ',OPN(1).,
         IF WHERE GT 2 THEN D1.. DO.,
         TEMP1 = OPND.,
         E1=1.,
         GO TO T2.,
         END D1.,
         IFTEMP=1.,
         IF WHERE LE 1 THEN
         OUTCOD '       STO     K00'.,
         ELSE OUTCOD '       STQ     K00'.,
  T2..   TEMP,LEV=0., X=1.,
         CALL ARITEXPR(AEXP2).,
         IF TEMP NE 0 THEN
           OUTCOD '       ',OPR(1),'  ',OPN(1).,
         IF WHERE GT 2 THEN
```

```
          OUTCOD '        CLA      ',OPND.,
        IF WHERE=2 THEN
          OUTCOD '        LLS      35'.,
        IF E1=1 THEN
          OUTCOD '        FSB      ',TEMP1.,
              ELSE
          OUTCOD '         FSB     K00'.,
        OUTCOD'         UFA      =0233000000000'.,
        OUTCOD'         ALS      10'.,
        OUTCOD'         ARS      10'.,
        GO TO COMPAR($INDEX(CODCMP,$CMPCT(IFTHEN.LOP))/3+1).,
        COMPAR(1).. OUTCOD'         TZE      ',TEMPOR('T',GTO).,
                    GO TO FMIFTHEN.,
        COMPAR(2).. OUTCOD'         TMI      ', TEMPOR('T',GTO).,
                    GO TO FMIFTHEN.,
        COMPAR(3).. OUTCOD'         TPL      ', TEMPOR('T',GTO).,
                    GO TO FMIFTHEN.,
        COMPAR(4).. OUTCOD'         TPL      ', TEMPOR('T',GTO).,
                    OUTCOD'         TZE      ', TEMPOR('T',GTO).,
                    GO TO FMIFTHEN.,
        COMPAR(5).. OUTCOD'         TMI      ', TEMPOR('T',GTO).,
                    OUTCOD'         TZE      ', TEMPOR('T',GTO).,
                    GO TO FMIFTHEN.,
        COMPAR(6).. OUTCOD'         TNZ      ', TEMPOR('T',GTO).,
FMIFTHEN.. END MIFTHEN.,
/* ------------------------------------------------------------------ */
/* ESTA MACRO COMPILA O COMANDO 'VA PARA' */
MGOTO.. MACRO GOTO.,
        PODE = 0.,
        IF LAB NE '' THEN CALL INSTLAB(LAB).,
        CALL INSTGTO(GTO).,
        OUTCOD '        TRA      ',TEMPOR('T',GTO).,
END MGOTO.,
/* ------------------------------------------------------------------ */
/* ESTA MACRO COMPILA O COMANDO 'VA SUB' */
MGOSUB.. MACRO GOSUB.,
       PODE = 1.,
         IF LAB NE '' THEN CALL INSTLAB(LAB).,
         CALL INSTGTO(GTO).,
         OUTCOD '        TSX      ',$TOCHA(GTO),',4'.,
END MGOSUB.,
/* ------------------------------------------------------------------ */
/* ESTA MACRO COMPILA O COMANDO 'RETORNE' */
MRET.. MACRO RET.,
       PODE = 1.,
       IF LAB NE '' THEN CALL INSTLAB(LAB).,
       OUTCOD'         TRA      1,4'.,
END MRET.,
* ------------------------------------------------------------------ */
* ESTA MACRO FAZ BY-PASS EM COMENTARIOS */
COM.. MACRO (LAB OR) COM.,
END MCOM.,
* ------------------------------------------------------------------ */
* ESTA MACRO COMPILA O COMANDO 'PARE' */
STOP.. MACRO 0-1(LAB) ''PARE''.,
```

```
      PODE = 0.,
      IF LAB NE '' THEN CALL INSTLAB(LAB).,
      OUTCOD'       TRA     S.JXIT'.,
END MSTOP.,
/* ------------------------------------------------------------------ */
   /* THIS MACRO COMPILES ASSIGNEMENT STATEMENTS */
MAC1.. MACRO (LAB OR) ''ARIT'' VARV ''='' EXPRESSION.,
       KEY EXPRESSION=AEXP!!
       DCL TEMPR CHAR(6).,
      IF LAB NE '' THEN CALL INSTLAB(LAB).,
     PODE = 1.,
       TEMPR=VARV.$I.,                          TEMP,LEV=0., X=1.,
   /* LEFT HAND OF '=' IS SUBSCRIPTED */
       IF VARV.INDV1 NE '' THEN D1.. DO.,
            I=$IHASH(VARV.INDV1.VR).,
            IF I LT 1 THEN DO., ATRIB(-I)=11., GO TO OU1., END.,
            IF ATRIB(I) EQ 0 THEN    DO.,
                     CALL ERRO2(VARV.$I).,
                NOGO=1.,
                     RETURN.,      END.,
 OU1.. CALL ARITEXPR(VARV.AEXP).,
      CALL TOFIXED(ATRIB($JHASH(VARV.$I))).,
      TEMP=1.,
      OPR(TEMP)='LAC'.,
      OPN(TEMP)=$CMPCT(OPND CAT ',1').,
       GO TO D6.,
       END D1.,
* LEFT HAND OF '=' IS DOUBLE SUBSCRIPTED */
    IF VARV.INDV2 NE '' THEN D2.. DO.,
      I=$IHASH(VARV.INDV2.VR).,
      E1 = I.,
      IF I LT 1 THEN DO.,
         ATRIB(-I)=1000010.,  GO TO Z1.,  END.,
      IF ATRIB(I)=0 THEN DO.,
      CALL ERRO2(VARV.INDV2.VR).,
         RETURN.,   END.,
      IF ATRIB(I) LE 99999 THEN DO.,
      CALL ERRO1(VARV.$I).,
         RETURN.,      END.,
'* OK. EVALUATE THE 1ST INDEX */
 Z1.. CALL ARITEXPR(VARV.INDV2.AEXP1).,
    I = $ABS(E1).,
    NLI = ATRIB(I)/100000.,
    NC = $MOD(ATRIB(I),100000).,
    XREAD = I.,
    CALL TOFIXED(NLI).,
    OUTCOD '       CLA *    ',$CMPCT(OPND).,
    OUTCOD '       STO      TREAD'.,
* OK. EVALUATE THE 2ND INDEX */
    CALL ARITEXPR(VARV.INDV2.AEXP2).,
    CALL TOFIXED(NC).,
    OUTCOD '       STQ      =',NLI.,
    OUTCOD '       MPY      ',OPND.,
    OUTCOD '       LLS      35'.,
    OUTCOD '       ADD      TREAD'.,
```

```
    OUTCOD '        STO     TREAD'.,
    TEMP = 1.,
    OPR(TEMP) = 'LAC'.,
    OPN (TEMP) = 'TREAD,1'.,
    GO TO D6.,
END D2.,
  /* LEFT HAND OF '=' IS A SIMPLE VARIABLE */
    I = $IHASH(VARV.$I).,
    IF I LT 1 THEN ATRIB(-I) = 0.,
              ELSE IF ATRIB(I) NE 0 THEN
                              OU2.. CALL ERRO3(VARV.$I).,
  /* COMPILATION OF THE ARITHMETIC EXPRESSION */
 D6..  CALL ARITEXPR(EXPRESSION.AEXP).,
       IF TEMP NE 0 THEN
       OUTCOD '        ', OPR(1),'  ', OPN(1).,
       IF WHERE GT 2 THEN D4.. DO.,
            OUTCOD '        CLA     ', OPND., GO TO D5.,
       END D4.,
  /* OUTPUT OF THE ASSIGNEMENT CODE */
       IF WHERE LE 1 THEN
            D5.. IF TEMP NE 0
                 THEN OUTCOD '        STO     ',$CMPCT(TEMPR),',1'.,
                 ELSE OUTCOD '        STO     ', TEMPR.,
            ELSE IF TEMP NE 0
                 THEN OUTCOD '        STQ     ',$CMPCT(TEMPR),',1'.,
                 ELSE OUTCOD '        STQ     ', TEMPR.,
ND MAC1.,
* ------------------------------------------------------------------ */
* THIS MACRO COMPILES THE 'IMPRIMA' STATEMENT */
PRINT.. MACRO PRINT.,
       DECLARE HOL CHAR(6).,
     PODE = 1.,
       OUT = 1.,
       IF LAB NE '' THEN CALL INSTLAB(LAB).,
       IF PRINTLIST EQ '' THEN OUTCOD '        CALL    IMPRME'.,
                          ELSE D1.. DO.,
       I=0.,
       D2.. I=I+1.,
       /* OUTPUT OF A STRING OF CHARACTERS */
        IF ITEM(I).$L NE '' THEN D5.. DO.,
       OUTCOD '        CALL    PORLIN(BRACK)'.,
       DO J=1 TO ($LNGHT(ITEM(I).$L)-1)/6+1.,
       HOL=$SUBST(ITEM(I).$L,(J-1)*6+1,6).,
       OUTCOD '        CALL    PORLIN(=H',HOL,')'.,
       END.,
      GO TO FD2.,
        END D5.,
       X = 1.,
       T=TEST(ITEM(I).AEXP).,
       /* OUTPUT OF AN EXPRESSION */
       IF T=0 THEN D3.. DO.,
       E1 = I.,
       TEMP,LEV=0., X=1.,
       CALL ARITEXPR(ITEM(I).AEXP).,
       I = E1.,
```

```
      IF WHERE GT 2 THEN GO TO XM1.,
      IF WHERE LE 1 THEN OUTCOD '       STO     ','T.',INCREM.,
                    ELSE OUTCOD '       STQ     ','T.',INCREM.,
XM1.. X=X-1.,
      OUTCOD '       CALL    PORLIN(BRACK)'.,
      OUTCOD '       CALL    POREAL(','T.',INCREM,')'.,
      GO TO FD2.,
      END D3.,
      /* OUTPUT OF A SIMPLE VARIABLE */
      IF T=1 THEN D4.. DO.,
      OUTCOD '       CALL    PORLIN(BRACK)'.,
      OUTCOD '       CALL    POREAL(',ITEM(I).AEXP,')'.,
      GO TO FD2.,
      END D4.,
      /* OUTPUT OF A CONSTANT */
      OUTCOD '       CALL    PORLIN(BRACK)'.,
      IF T=2 THEN OUTCOD
                  '       CALL    PORINT(=',ITEM(I).AEXP ,')'.,
                  ELSE OUTCOD
                  '       CALL    POREAL(=',ITEM(I).AEXP ,')'.,
      FD2.. IF ITEM(I+1) NE '' THEN GO TO D2.,
      OUTCOD '       CALL    IMPRME'.,
      END D1.,
ND MPRINT.,
* ------------------------------------------------------------------ */
 THIS MACRO COMPILES THE 'LEIA' STATMENT */
EAD.. MACRO READ.,
      DECLARE PZE(20) CHAR(6).,
      INP=1.,
      IF LAB NE '' THEN CALL INSTLAB(LAB).,
      LREAD = LREAD+1.,
      OUTCOD '       LXA     =',NREAD+1,',4'.,
      OUTCOD '       CLA     ST.',$SUBST(LREAD,11,2),'-',NREAD+2.,
      OUTCOD '       SUB     =',NREAD+1.,
      OUTCOD '       STA     SAVE.-2'.,
      OUTCOD '       ALS     18'.,
      OUTCOD '       STD     SAVE.-1'.,
      OUTCOD '       CLA     SAVE.-1'.,
      OUTCOD '       TMT     ',NREAD+1.,
      I=0.,
 D1.. I=I+1.,
      /* INPUT OF A SUBSCRIPTED VARIABLE */
      /* ONE SUBSCRIPT */
      IF VARY(I).IND1 NE '' THEN D3.. DO.,
           J=$IHASH(VARY(I).VR).,
           IF J LT 1 THEN DO.,
                ATRIB(-J)=10., GO TO R2., END.,
           IF ATRIB(J) = 0 THEN DO.,
                CALL ERRO2(VARY(I).VR).,
                NOGO=1.,
                GO TO FD1., END.,
           IF ATRIB(J) GT 99999 THEN DO.,
                CALL  ERRO4(VARY(I).VR).,
                NOGO = 1.,
                GO TO FD1.,
```

```
        END.,
        /* OK. EVALUATE THE INDEX EXPRESSION */
        R2.. E1 = I.,
             X = 1.,   TEMP,LEV = 0.,
             CALL ARITEXPR(VARY(I).IND1.AEXP).,
             I = E1.,
             CALL TOFIXED(ATRIB($JHASH(VARY.$I))).,
             OUTCOD'       CLA     ',$CMPCT(OPND).,
             OUTCOD'       ADD     ST.',$SUBST(LREAD,11,2),'-',
             NREAD-I+2.,
             OUTCOD'       STO     ST.',$SUBST(LREAD,11,2),'-',
             NREAD-I+2.,
             PZE(I)=$CMPCT(VARY(I).VR).,
             GO TO FD1., END D3.,
    /* TWO SUBSCRIPTS */
    IF VARY(I).IND2 NE '' THEN D4.. DO.,
        J=$IHASH(VARY(I).VR).,
        IF J LT 1 THEN DO.,
             ATRIB(-J) = 1000010.,
             GO TO O2.,  END.,
        IF ATRIB(J) = 0 THEN DO.,
             CALL ERRO2(VARY(I).VR).,
             NOGO=1.,
             GO TO FD1.,
             END.,
        IF ATRIB(J) LE 99999 THEN DO.,
             CALL ERRO1(VARY(I).VR).,
             NOGO=1.,
             GO TO FD1.,   END.,
        /* OK. EVALUATE THE 1ST INDEX */
        O2.. E1 = I.,
             J=$ABS(J).,
             NLI=ATRIB(J)/100000.,
             NC=$MOD(ATRIB(J),100000).,
             X = 1.,   TEMP,LEV = 0.,
             CALL ARITEXPR(VARY(I).IND2.AEXP1).,
             I = E1.,
             XREAD=1.,
             CALL TOFIXED(NLI).,
             OUTCOD'       CLA     ',$CMPCT(OPND).,
             OUTCOD'       STO     TREAD'.,
        /* OK. EVALUATE THE THE 2ND INDEX */
             E1 = I.,
             X = 1.,   TEMP,LEV = 0.,
             CALL ARITEXPR(VARY(I).IND2.AEXP2).,
             I = E1.,
             CALL TOFIXED(NC).,
             OUTCOD '      STQ     =',NLI.,
             OUTCOD '      MPY     ',OPND.,
             OUTCOD '      LLS     35'.,
             OUTCOD'       ADD     TREAD'.,
             OUTCOD'       ADD     ST.',$SUBST(LREAD,11,2),'-',
             NREAD-I+2.,
             NREAD-I+2.,
*** INSTR  1108 UNID      2 CONTA     1 FALTOU ''=''.
```

```
                OUTCOD'       STO     ST.',$SUBST(LREAD,11,2),'-',
                PZE(I) = $CMPCT(VARY(I).VR).,
                GO TO FD1.,
        END D4.,
        /* INPUT OF A SINGLE VARIABLE */
        IF VARY(I).VR NE '' THEN D2.. DO.,
             J=$IHASH(VARY(I).VR).,
             IF J LT 1 THEN DO.,
                  ATRIB(-J)=0., GO TO R1., END.,
             IF ATRIB(J) NE 0 THEN DO.,
                  CALL ERRO3(VARY(I).VR).,
                  NOGO=1.,
                  GO TO FD1.,     END.,
             R1.. PZE(I)=$CMPCT(VARY(I).VR)., GO TO FD1.,
        END D2.,
   FD1.. IF VARY(I+1) NE '' THEN GO TO D1.,
   /* OUTCOD THE VARIABLES ADDRESSES */
        OUTCOD '       TRA     ST.',$SUBST(LREAD,11,2).,
        I = 0.,
   D5.. I = I + 1.,
        OUTCOD'       PZE     ',PZE(I).,
        IF VARY(I+1) NE '' THEN GO TO D5.,
JTCOD 'ST.',$SUBST(LREAD,11,2),'  BSS     0'.,
   OUTCOD'       CALL    BSCRD(R)'.,
   OUTCOD'       CLA     R'.,
   OUTCOD'       STO*    ST.',$SUBST(LREAD,11,2),',4'.,
   OUTCOD'       TXI     *+1,4,-1'.,
   OUTCOD'       TXH     ST.',$SUBST(LREAD,11,2),',4,0'.,
        OUTCOD '       CLA     SAVE.-2'.,
        OUTCOD '       TMT     ',NREAD+1.,
   END MREAD.,
* ------------------------------------------------------------------ */
* THIS MACRO COMPILES THE 'DIM' DECLARATION */
DIM.. MACRO (LAB OR) DIM.,
     D1.. DO I=1 BY 1 WHILE DVAR(I) NE ''.,
          J=$IHASH(DVAR(I)).,
          IF J LT 1 THEN D2.. DO.,
          IF DVAR(I).DIMVAR1 NE '' THEN
                ATRIB(-J)=(DVAR(I).DIMVAR1.$C)  .,
                                        ELSE D3.. DO.,
                NLIN=(DVAR(I).DIMVAR2.C1)  .,
                NCOL=(DVAR(I).DIMVAR2.C2)  .,
                ATRIB(-J)=NLIN*100000+NCOL.,
                                        END D3.,
          END D2.,
             ELSE OUTPUT       '
             '  ***  VARIAVEL REDEFINIDA. ',DVAR(I),' IGNORADA.'.,
        END D1.,
ND MDIM.,
* ------------------------------------------------------------------ */
* EXECUTING PROGRAM */
* ------------------------------------------------------------------ */
          IFTEMP,MASK,PODE,TOTDO,OUT,INP,NTEMP,PMAX,PEXP3P,NDO=0.,
          FIXD,NOGO,LREAD,XREAD,V,NFNC=0.,
          VFN = '    '.,
```

```
        FUNCAO = '*SEN*COS*ATN*EXP*LOG*ROU*TAN'  CAT
                 '*INT*ABS*FNA*FNB*FNC*FND*FNE'  CAT
                 '*FNF*FNG*FNH*FNI*FNJ*FNK*FNL'  CAT
                 '*FNM*FNN*FNO*FNP*FNQ*FNR*FNS'  CAT
                 '*FNT*FNU*FNV*FNW*FNX*FNY*FNZ'  .,
        CALL $IHASH('SEN').,          FUNC( 1) = 'SIN'.,
        CALL $IHASH('COS').,          FUNC( 2) = 'COS'.,
        CALL $IHASH('ATN').,          FUNC( 3) = 'ATAN'.,
        CALL $IHASH('EXP').,          FUNC( 4) = 'EXP'.,
        CALL $IHASH('LOG').,          FUNC( 5) = 'ALOG'.,
        CALL $IHASH('ROU').,          FUNC( 6) = 'SQRT'.,
        CALL $IHASH('TAN').,          FUNC( 7) = 'TAN'.,
        CALL $IHASH('INT').,          FUNC( 8) = 'AINT'.,
        DO I=1 TO $LMLOC.,
        ATRIB(I)=80000.,
        END.,
        TBEND=0., X1=1., OUTCOD '$IBMAP BASIC   NODECK'.,
)O I=1 TO 40., POT(I)=0., END.,
       /* MACRO INSTRUCTIONS DEFINITIONS */
     OUTCOD 'FPM     MACRO   X'.,
     OUTCOD '        FMP     X'.,
     OUTCOD '        LRS     35'.,
     OUTCOD '        ENDM    FPM'.,
     OUTCOD 'FPD     MACRO   X'.,
     OUTCOD '        LLS     35'.,
     OUTCOD '        FDP     X'.,
     OUTCOD '        ENDM    FPD'.,
     OUTCOD 'FPDI    MACRO   X'.,
     OUTCOD '        LLS     35'.,
     OUTCOD '        FDP*    X'.,
     OUTCOD '        ENDM    FPDI'.,
     OUTCOD 'FPMI    MACRO   X'.,
     OUTCOD '        FMP*    X'.,
     OUTCOD '        LRS     35'.,
     OUTCOD '        ENDM    FPMI'.,
        OUTCOD '        CALL    LRESET'.,
2..           COMPILE.,
        GO TO L2.,
* ------------------------------------------------------------------ */
INVLD.. MACRO INVLD.,
        OUTPUT' *** COMANDO INVALIDO OU CONTINUACAO INVALIDA DE COMA',
              'NDO VALIDO.'.,
ND MINVLD.,
ND COMCOM.,
```

LISTAGEM DAS ROTINAS AUXILIARES

ROTINAS AUXILIARES

Com a finalidade de facilitar a geração de código objeto, o compilador gera, para alguns comandos (ver seção 9), chamadas de rotinas auxiliares que são descritas a seguir.

1 BSCRD(R)

Esta rotina faz a leitura e conversão de um valor numérico perfurado em cartão, e atribui o valor lido à variável R, que é o argumento da rotina. A leitura é feita pela rotina SCAN, e a conversão para a representação binária é feita pela rotina CNVCNS [10] e [12] .

2 AINT(R)

Esta função retorna a parte inteira do argumento R. O compilador BASIC gera uma chamada a esta rotina, para cada referência à função interna INT.

3 TINDEX(IND,MAX)

Esta rotina é usada para testar se o subscrito utilizado para indexar uma variável está dentro dos limites previstos para o susbcrito, isto é, entre 0 (zero) e MAX. MAX é o tamanho especificado no comando DIM, ou então 10, se o comando DIM não foi usado.

4 TOFIX(I,R)

Esta rotina transforma o argumento R, de representação de ponto flutuante, para representação de ponto fixo. Este valor retorna no argumento I. A rotina é usada

em subscritos de variáveis.

5 PORCAR(IC)

Esta rotina é usada para eliminar os characteres brancos de IC, para poder ser transmitido à rotina PORLIN, na compilação de comandos IMPRIMA, onde aparece um literal na lista de impressão.

```
ONTROL DICTIONARY

                    $TEXT   BSCRD

)000
                            ENTRY   BSCRD
                    BSCRD   SAVE    1,2,4
)0 1 00105 10001            LXA     =0,1
                            CALL    SCAN

)001
0 0 20004 10011             CLA     ITIPO       CARREGA INDICADOR DE TIPO.
0 0 00107 10001             SUB     =2          TESTA SE DIGITO.
0 0 00041 10001             TZE     A
0 0 00110 10001             SUB     =3          TESTA SE PONTO.
0 0 00041 10001             TZE     A
0 0 00111 10001             SUB     =6          TESTA SE MAIS.
0 0 00064 10001             TZE     B
0 0 00106 10001             SUB     =1          TESTA SE MENOS.
0 0 00063 10001             TZE     C
                    ERRO    CALL    JOBOU(ADR)  MENSAGEM DE ERRO.
0 0 12000 10011             TRA     S.JXIT
                    A       CALL    CNVCNS      CONVERSAO DO RESTO DO NUMERO.
0 0 20004 10011             CLA     ITIPO       CARREGA INDICADOR DE TIPO.
0 0 00112 10001             SUB     =30         TESTA SE INTEIRO.

)002
0 0 00054 10001             TNZ     R
0 0 20005 10011             CLA     IUNI        FAZ A CONVERSAO DE
0 0 00075 10001             ORA     CHAR              INTEIRO
0 0 00075 10001             FAD     CHAR              PARA
0 0 20005 10011             STO     IUNI              REAL.
0 0 00056 10001             TRA     S
0 0 00106 10001 R           SUB     =1
0 0 00034 10001             TNZ     ERRO
0 0 20005 10011 S           CLA     IUNI
)0 1 00061 10001            TXL     SAI,1,0
)0 0 00002 10000            CHS
50 4 00003 10000 SAI        STO*    3,4
                            RETURN  BSCRD
)0 1 00106 10001 C          LXA     =1,1
                    B       CALL    SCAN
)0 0 20004 10011            CLA     ITIPO
)0 0 00107 10001            SUB     =2

)003
)0 0 00041 10001            TZE     A
)0 0 00110 10001            SUB     =3
)0 0 00041 10001            TZE     A
0 0 00034 10001             TRA     ERRO
000000      10000 CHAR      OCT     233000000000
0 0 00001 10000 ADR         PZE     1
5 0 00100 10001             PZE     MESS,,5
546025      10000 MESS      BCI     5, *** ERRO NO CARTAO DE DADOS.
604546      10000
516321      10000
256024      10000
```

```
623360    10000
                          EXTERN  SCAN,CNVCNS,JOBOU
0 0 00113 00001           USE     S
0 0 00004 00001 NUSED     BSS     4
0 0 00001 00001 ITIPO     BSS     1
0 0 00006 00001 IUNI      BSS     6
0 0 00105 00001           USE     PREVIOUS
                SCN       CONTRL  S
                             LITERALS
000000    10000
000001    10000

004
000002    10000
000003    10000
000006    10000
000036    10000
                          EXTERN      S.JXIT   GENERATED
                          EXTERN      S.SLOC   GENERATED
    00000 01111           END

    $DKEND BSCRD
```

```
                    $TEXT  TINDEX


000
                           ENTRY  TINDEX
                    TINDEX SAVE   1,2,4
 4 00003 10000             CLA*   3,4
 0 00027 10001             TMI    ERRO
 0 00034 10001             STO    IND
 4 00004 10000             CLA*   4,4

001
 0 00047 10001             SUB    =1
 0 00034 10001             SUB    IND
 0 00027 10001             TMI    ERRO
                           RETURN TINDEX
                    ERRO   CALL   JOBOU(ADR)
 0 14000 10011             TRA    S.JXIT
 0 00001 00001 IND         BSS    1
 0 00001 10000 ADR         PZE    1
 0 00037 10001             PZE    MESS,,8
546062     10000 MESS      BCI    8, *** SUBSCRITO FORA DOS LIMITES ESPECIFICADOS.
235131     10000
264651     10000
266260     10000
315325     10000
524725     10000
312321     10000

002
336060     10000
                           EXTERN JOBOU
                              LITERALS
100001     10000
                           EXTERN     S.JXIT   GENERATED
                           EXTERN     S.SLOC   GENERATED
    00000 01111            END

    $DKEND TINDEX
```

ONTROL DICTIONARY

```
                    $TEXT  AINT

0000
                           ENTRY   AINT
                    AINT   SAVE    1,2,4
60 4 00003 10000           CLA*    3,4
00 0 00026 10001           UFA     MASK
00 0 00012 10000           ALS     10
00 0 00012 10000           ARS     10

0001
00 0 00026 10001           ORA     MASK
00 0 00026 10001           FAD     MASK
                           RETURN  AINT
0000000     10000 MASK     OCT     233000000000
                           EXTERN      S.SLOC   GENERATED
    00000 01111            END

    $DKEND AINT
```

ONTROL DICTIONARY

```
                    $TEXT  TOFIX

0000
                           ENTRY   TOFIX
                    TOFIX  SAVE    1,2,4
60 4 00003 10000           CLA*    3,4
30 0 00025 10001           UFA     MASK
00 0 00012 10000           ALS     10
00 0 00012 10000           ARS     10

0001
60 4 00004 10000           STO*    4,4
                           RETURN  TOFIX
1000000     10000 MASK     OCT     233000000000
                           EXTERN     S.SLOC   GENERATED
     00000 01111           END

     $DKEND TOFIX
```

P R O G R A M A S E X E M P L O S

- Programas fonte
- Códigos gerados
- Resultados

```
$OBCOM          NOLIST
      COM PROGRAMA EXEMPLO 1.
      COM CALCULO DA SOMA DOS NUMEROS NATURAIS DE 1 A 50 (INCLUSIVE).
      ARIT S = 0
      ARIT N = 1
   30 ARIT S = S + N
      ARIT N = N + 1
      SE N MEG 50 ENTAO 30
      IMPRIMA 'SOMA =',S
      FIM

****** INTERPRETACAO COMCOM ******   FIM DE EXECUCAO.
```

CONTROL DICTIONARY

```
                                        $TEXT   BASIC

BINARY CARD 00000
                                        FPM     MACRO   X
                                                FMP     X
                                                LRS     35
                                                ENDM    FPM
                                        FPD     MACRO   X
                                                LLS     35
                                                FDP     X
                                                ENDM    FPD
                                        FPDI    MACRO   X
                                                LLS     35
                                                FDP*    X
                                                ENDM    FPDI
                                        FPMI    MACRO   X
                                                FMP*    X
                                                LRS     35
                                                ENDM    FPMI
                                                CALL    LRESET
                                        *** 0
 00003  0500 00 0 00054 10001                   CLA     =0.0
 00004  0601 00 0 00050 10001                   STO     S
                                        *** 1
 00005  0500 00 0 00055 10001                   CLA     =01.0
 00006  0601 00 0 00051 10001                   STO     N
 00007  2 00000 0 00000 00001           T00030  BSS     0
 00007  0500 00 0 00050 10001                   CLA     S
 00010  0300 00 0 00051 10001                   FAD     N
                                        *** S + N
 00011  0601 00 0 00050 10001                   STO     S
 00012  0500 00 0 00051 10001                   CLA     N
 00013  0300 00 0 00055 10001                   FAD     =01.0
                                        *** N + 1
 00014  0601 00 0 00051 10001                   STO     N
                                        *** N
                                        *** 50
 00015  0500 00 0 00056 10001                   CLA     =050.0
 00016  0302 00 0 00051 10001                   FSB     N
 00017 -0300 00 0 00057 10001                   UFA     =0233000000000
 00020  0767 00 0 00012 10000                   ALS     10
 00021  0771 00 0 00012 10000                   ARS     10

BINARY CARD 00001
 00022  0120 00 0 00007 10001                   TPL     T00030
 00023  0100 00 0 00007 10001                   TZE     T00030
                                                CALL    PORLIN(BRACK)
                                                CALL    PORLIN(=HSOMA =)
                                                CALL    PORLIN(BRACK)
                                                CALL    POREAL(S)
                                                CALL    IMPRME

BINARY CARD 00002
 00047  0020 00 0 34000 10011                   TRA     S.JXIT
 00050  2 00000 0 00001 00001 S                 BSS     1
```

```
00051  2 00000 0 00001 00001 N      BSS     1
                                    EXTERN  IMPRME,PORLIN,POREAL,PORI
00052  363660606060     10000 BRACK OCT     363660606060
                                    EXTERN  LRESET
                                      LITERALS
00053 -224644216013     10000
00054  000000000000     10000
00055  201400000000     10000
00056  206620000000     10000
00057  233000000000     10000
                                    EXTERN      S.JXIT   GENERATED
                  00000 01111       END

                  $DKEND BASIC
```

OBJECT PROGRAM IS BEING ENTERED INTO STORAGE.
SOMA= .12750000E4

```
$OBCOM        NOLIST
      COM PROGRAMA EXEMPLO 2.
      COM CALCULO DO MAIOR VALOR DOS ELEMENTOS DE UM CONJUNTO.
      LEIA M
      IMPRIMA M
   30 LEIA N
      SE N IGU 0 ENTAO 60
      IMPRIMA N
      SE N MEG M ENTAO 30
      ARIT M = N
      VA PARA 30
   60 IMPRIMA 'MAXIMO=',M
      FIM
```

****** INTERPRETACAO COMCOM ****** FIM DE EXECUCAO.

CONTROL DICTIONARY

```
                                        $TEXT  BASIC

BINARY CARD 00000
                                        FPM    MACRO   X
                                               FMP     X
                                               LRS     35
                                               ENDM    FPM
                                        FPD    MACRO   X
                                               LLS     35
                                               FDP     X
                                               ENDM    FPD
                                        FPDI   MACRO   X
                                               LLS     35
                                               FDP*    X
                                               ENDM    FPDI
                                        FPMI   MACRO   X
                                               FMP*    X
                                               LRS     35
                                               ENDM    FPMI
                                               CALL    LRESET
00003  0534 00 4 00205 10001                   LXA     =1,4
00004  0500 00 0 00013 10001                   CLA     ST.01-2
00005  0402 00 0 00205 10001                   SUB     =1
00006  0621 00 0 00155 10001                   STA     SAVE.-2
00007  0767 00 0 00022 10000                   ALS     18
00010  0622 00 0 00156 10001                   STD     SAVE.-1
00011  0500 00 0 00156 10001                   CLA     SAVE.-1
00012 -1704 00 0 00001 10000                   TMT     1
00013  0020 00 0 00015 10001                   TRA     ST.01
00014  0 00000 0 00151 10001                   PZE     M
00015  2 00000 0 00000 00001 ST.01             BSS     0
                                               CALL    BSCRD(R)
00021  0500 00 0 00154 10001                   CLA     R

BINARY CARD 00001
00022  0601 60 4 00015 10001                   STO*    ST.01,4
00023  1 77777 4 00024 10001                   TXI     *+1,4,-1
00024  3 00000 4 00015 10001                   TXH     ST.01,4,0
00025  0500 00 0 00155 10001                   CLA     SAVE.-2
00026 -1704 00 0 00001 10000                   TMT     1
                                               CALL    PORLIN(BRACK)
                                               CALL    POREAL(M)
                                               CALL    IMPRME
00042  2 00000 0 00000 00001 T00030            BSS     0
00042  0534 00 4 00205 10001                   LXA     =1,4
00043  0500 00 0 00052 10001                   CLA     ST.02-2

BINARY CARD 00002
00044  0402 00 0 00205 10001                   SUB     =1
00045  0621 00 0 00155 10001                   STA     SAVE.-2
00046  0767 00 0 00022 10000                   ALS     18
00047  0622 00 0 00156 10001                   STD     SAVE.-1
00050  0500 00 0 00156 10001                   CLA     SAVE.-1
00051 -1704 00 0 00001 10000                   TMT     1
00052  0020 00 0 00054 10001                   TRA     ST.02
```

```
00053  0 00000 0 00152 10001          PZE     N
00054  2 00000 0 00000 00001 ST.02    BSS     0
                                      CALL    BSCRD(R)
00060  0500 00 0 00154 10001          CLA     R
00061  0601 60 4 00054 10001          STO*    ST.02,4
00062  1 77777 4 00063 10001          TXI     *+1,4,-1
00063  3 00000 4 00054 10001          TXH     ST.02,4,0
00064  0500 00 0 00155 10001          CLA     SAVE.-2
00065 -1704 00 0 00001 10000          TMT     1
                              *** N
                              *** 0

BINARY CARD 00003
00066  0500 00 0 00204 10001          CLA     =0.0
00067  0302 00 0 00152 10001          FSB     N
00070 -0300 00 0 00207 10001          UFA     =0233000000000
00071  0767 00 0 00012 10000          ALS     10
00072  0771 00 0 00012 10000          ARS     10
00073  0100 00 0 00121 10001          TZE     T00060
                                      CALL    PORLIN(BRACK)
                                      CALL    POREAL(N)
                                      CALL    IMPRME
                              *** N
                              *** M
00107  0500 00 0 00151 10001          CLA     M
00110  0302 00 0 00152 10001          FSB     N

BINARY CARD 00004
00111 -0300 00 0 00207 10001          UFA     =0233000000000
00112  0767 00 0 00012 10000          ALS     10
00113  0771 00 0 00012 10000          ARS     10
00114  0120 00 0 00042 10001          TPL     T00030
00115  0100 00 0 00042 10001          TZE     T00030
                              *** N
00116  0500 00 0 00152 10001          CLA     N
00117  0601 00 0 00151 10001          STO     M
00120  0020 00 0 00042 10001          TRA     T00030
00121  2 00000 0 00000 00001 T00060   BSS     0
                                      CALL    PORLIN(BRACK)
                                      CALL    PORLIN(=HMAXIMO)
                                      CALL    PORLIN(=H=     )

BINARY CARD 00005
                                      CALL    PORLIN(BRACK)
                                      CALL    POREAL(M)
                                      CALL    IMPRME
00150  0020 00 0 20000 10011          TRA     S.JXIT
00151  2 00000 0 00001 00001 M        BSS     1
00152  2 00000 0 00001 00001 N        BSS     1
                                      EXTERN  IMPRME,PORLIN,POREAL,PORIN
00153  363660606060     10000 BRACK   OCT     363660606060
                                      EXTERN  LRESET
                                      EXTERN  BSCRD
00154  2 00000 0 00001 00001 R        BSS     1
00155  0 00157 0 00000 10100          PZE     **,,SAVE.
```

```
BINARY CARD 00006
 00156  0 00000 0 00157 10001          PZE      SAVE.,,**
 00157  2 00000 0 00024 00001 SAVE.    BSS      20
                                          LITERALS
 00203 -042167314446     10000
 00204  000000000000     10000
 00205  000000000001     10000
 00206  136060606060     10000
 00207  233000000000     10000
                                      EXTERN     S.JXIT   GENERATED
                    00000 01111       END

                    $DKEND BASIC
```

309043203PAA AZEREDO

5.5 7.5 3.7 2.3 4.2 9.4 8.8 1.6 0.0

.54999999E1
.74999999E1
.37000000E1
.23000000E1
.42000000E1
.94000000E1
.88000000E1
.16000000E1

MAXIMO= .94000000E1

```
$OBCOM          NOLIST
      COM PROGRAMA EXEMPLO 4.
      COM OBTENCAO DOS NUMEROS DE 4 ALGARISMOS (XXYY) QUE SATISFACAM
      COM A CONDICAO DE   XX + YY = N      N**2 = XXYY.
      ARIT N = 999
      PARA I = 10 ATE 99
      PARA J =  0 ATE 99
      ARIT N = N + 1
      ARIT M = ( I + J ) ** 2
      SE M IGU N ENTAO 50
      VA PARA 60
   50 IMPRIMA N
   60 PROXIMO J
      PROXIMO I
      FIM

****** INTERPRETACAO COMCOM ******   FIM DE EXECUCAO.
```

CONTROL DICTIONARY

```
                                        $TEXT   BASIC

BINARY CARD 00000
                                FPM     MACRO   X
                                        FMP     X
                                        LRS     35
                                        ENDM    FPM
                                FPD     MACRO   X
                                        LLS     35
                                        FDP     X
                                        ENDM    FPD
                                FPDI    MACRO   X
                                        LLS     35
                                        FDP*    X
                                        ENDM    FPDI
                                FPMI    MACRO   X
                                        FMP*    X
                                        LRS     35
                                        ENDM    FPMI
                                        CALL    LRESET
                                *** 999
 00003  0500 00 0 00105 10001            CLA     =0999.0
 00004  0601 00 0 00072 10001            STO     N
                                *** 10
 00005  0500 00 0 00103 10001            CLA     =010.0
 00006  0601 00 0 00073 10001            STO     I
 00007  2 00000 0 00000 00001 D00001     BSS     0
 00007  0500 00 0 00104 10001            CLA     =099.0
 00010  0302 00 0 00073 10001            FSB     I
 00011 -0300 00 0 00106 10001            UFA     =0233000000000
 00012  0767 00 0 00012 10000            ALS     10
 00013  0771 00 0 00012 10000            ARS     10
 00014 -0120 00 0 00071 10001            TMI     E00001+1
                                *** 0
 00015  0500 00 0 00100 10001            CLA     =0.0
 00016  0601 00 0 00074 10001            STO     J
 00017  2 00000 0 00000 00001 D00002     BSS     0
 00017  0500 00 0 00104 10001            CLA     =099.0
 00020  0302 00 0 00074 10001            FSB     J

BINARY CARD 00001
 00021 -0300 00 0 00106 10001            UFA     =0233000000000
 00022  0767 00 0 00012 10000            ALS     10
 00023  0771 00 0 00012 10000            ARS     10
 00024 -0120 00 0 00065 10001            TMI     E00002+1
 00025  0500 00 0 00072 10001            CLA     N
 00026  0300 00 0 00101 10001            FAD     =01.0
                                *** N + 1
 00027  0601 00 0 00072 10001            STO     N
 00030  0500 00 0 00073 10001            CLA     I
 00031  0300 00 0 00074 10001            FAD     J
                                *** I + J
 00032  0601 00 0 00076 10001            STO     T+01
 00033  0500 00 0 00076 10001            CLA     T+01
 00034  0560 00 0 00102 10001            LDQ     =02.0
```

```
 00035 -1627 00 0 16000 10011          TSL     .EXP3.
                                  *** ( I + J ) ** 2
 00036  0601 00 0 00075 10001          STO     M
                                  *** M
                                  *** N
 00037  0500 00 0 00072 10001          CLA     N
 00040  0302 00 0 00075 10001          FSB     M
 00041 -0300 00 0 00106 10001          UFA     =0233000000000
 00042  0767 00 0 00012 10000          ALS     10
 00043  0771 00 0 00012 10000          ARS     10

BINARY CARD 00002
 00044  0100 00 0 00046 10001          TZE     T00050
 00045  0020 00 0 00061 10001          TRA     T00060
 00046  2 00000 0 00000 00001 T00050   BSS     0
                                       CALL    PORLIN(BRACK)
                                       CALL    POREAL(N)
                                       CALL    IMPRME
 00061  2 00000 0 00000 00001 T00060   BSS     0
 00061  0500 00 0 00074 10001          CLA     J
 00062  0300 00 0 00101 10001          FAD     =1.0
 00063  0601 00 0 00074 10001          STO     J
 00064  0020 00 0 00017 10001 E00002   TRA     D00002

BINARY CARD 00003
 00065  0500 00 0 00073 10001          CLA     I
 00066  0300 00 0 00101 10001          FAD     =1.0
 00067  0601 00 0 00073 10001          STO     I
 00070  0020 00 0 00007 10001 E00001   TRA     D00001
 00071  0020 00 0 20000 10011          TRA     S.JXIT
 00072  2 00000 0 00001 00001 N        BSS     1
 00073  2 00000 0 00001 00001 I        BSS     1
 00074  2 00000 0 00001 00001 J        BSS     1
 00075  2 00000 0 00001 00001 M        BSS     1
 00076  2 00000 0 00001 00001 T.01     BSS     1
                                       EXTERN  IMPRME,PORLIN,POREAL,PORI
 00077  363660606060     10000 BRACK   OCT     363660606060
                                       EXTERN  LRESET
                                       EXTERN  .EXP3.
                                          LITERALS
 00100  000000000000     10000
 00101  201400000000     10000
 00102  202400000000     10000
 00103  204500000000     10000
 00104  207614000000     10000
 00105  212763400000     10000
 00106  233000000000     10000

                                       EXTERN      S.JXIT   GENERATED
                    00000 01111        END

                    $DKEND BASIC
```

```
OBJECT PROGRAM IS BEING ENTERED INTO STORAGE.
 .20250000E4
 .30250000E4
 .98010000E4
```

```
$OBCOM          NOLIST
      COM PROGRAMA EXEMPLO 5.
      COM CALCULO DOS VALORES DE Y = ( X * X - X + 2 ) / ( X + 3 )
      COM X = 0,1,2,3, ... ,10.
      ARIT X = 0
   20 IMPRIMA 'X = ',X,'Y = ', (X * X - X + 2)/(X + 3)
      ARIT X = X + 1
      SE X MEG 10 ENTAO 20
      FIM

****** INTERPRETACAO COMCOM ******   FIM DE EXECUCAO.
```

CONTROL DICTIONARY

```
                                        $TEXT  BASIC

BINARY CARD 00000
                                FPM     MACRO   X
                                        FMP     X
                                        LRS     35
                                        ENDM    FPM
                                FPD     MACRO   X
                                        LLS     35
                                        FDP     X
                                        ENDM    FPD
                                FPDI    MACRO   X
                                        LLS     35
                                        FDP*    X
                                        ENDM    FPDI
                                FPMI    MACRO   X
                                        FMP*    X
                                        LRS     35
                                        ENDM    FPMI
                                        CALL    LRESET
                                *** 0
 00003  0500 00 0 00112 10001           CLA     =0.0
 00004  0601 00 0 00102 10001           STO     X
 00005  2 00000 0 00000 00001 T00020    BSS     0
                                        CALL    PORLIN(BRACK)
                                        CALL    PORLIN(=HX =    )
                                        CALL    PORLIN(BRACK)
                                        CALL    POREAL(X)

BINARY CARD 00001
                                        CALL    PORLIN(BRACK)
                                        CALL    PORLIN(=HY =    )
 00035  0560 00 0 00102 10001           LDQ     X
                                        FPM     X
 00040 -0600 00 0 00103 10001           STQ     T.01
 00041  0500 00 0 00103 10001           CLA     T.01
 00042  0302 00 0 00102 10001           FSB     X
 00043  0300 00 0 00114 10001           FAD     =02.0
                                *** X * X - X + 2
 00044  0601 00 0 00104 10001           STO     T.02

BINARY CARD 00002
 00045  0500 00 0 00102 10001           CLA     X
 00046  0300 00 0 00115 10001           FAD     =03.0
                                *** X + 3
 00047  0601 00 0 00105 10001           STO     T.03
 00050  0560 00 0 00104 10001           LDQ     T.02
                                        FPD     T.03
                                *** ( X * X - X + 2 ) / ( X + 3 )
 00053 -0600 00 0 00106 10001           STQ     T.04
                                        CALL    PORLIN(BRACK)
                                        CALL    POREAL(T.04)
                                        CALL    IMPRME
 00067  0500 00 0 00102 10001           CLA     X
```

```
BINARY CARD 00003
 00070  0300 00 0 00113 10001          FAD     =01.0
                                  *** X + 1
 00071  0601 00 0 00102 10001          STO     X
                                  *** X
                                  *** 10
 00072  0500 00 0 00116 10001          CLA     =010.0
 00073  0302 00 0 00102 10001          FSB     X
 00074 -0300 00 0 00117 10001          UFA     =0233000000000
 00075  0767 00 0 00012 10000          ALS     10
 00076  0771 00 0 00012 10000          ARS     10
 00077  0120 00 0 00005 10001          TPL     T00020
 00100  0100 00 0 00005 10001          TZE     T00020
 00101  0020 00 0 34000 10011          TRA     S.JXIT
 00102  2 00000 0 00001 00001 X        BSS     1
 00103  2 00000 0 00001 00001 T.01     BSS     1
 00104  2 00000 0 00001 00001 T.02     BSS     1
 00105  2 00000 0 00001 00001 T.03     BSS     1
 00106  2 00000 0 00001 00001 T.04     BSS     1
                                       EXTERN  IMPRME,PORLIN,POREAL,PORIN
 00107  363660606060     10000 BRACK   OCT     363660606060
                                       EXTERN  LRESET
                                          LITERALS
 00110 -306013606060     10000
 00111 -276013606060     10000
 00112  000000000000     10000

BINARY CARD 00004
 00113  201400000000     10000
 00114  202400000000     10000
 00115  202600000000     10000
 00116  204500000000     10000
 00117  233000000000     10000
                                       EXTERN     S.JXIT   GENERATED
                   00000 01111         END

                   $DKEND BASIC
```

OBJECT PROGRAM IS BEING ENTERED INTO STORAGE.

```
X=  .0E-38  Y=  .66666666E0
X=  .10000000E1  Y=  .50000000E0
X=  .20000000E1  Y=  .80000000E0
X=  .30000000E1  Y=  .13333333E1
X=  .40000000E1  Y=  .20000000E1
X=  .50000000E1  Y=  .27500000E1
X=  .60000000E1  Y=  .35555556E1
X=  .70000000E1  Y=  .44000000E1
X=  .80000000E1  Y=  .52727273E1
X=  .90000000E1  Y=  .61666666E1
X=  .10000000E2  Y=  .70769231E1
```

```
$OBCOM          NOLIST
      COM PROGRAMA EXEMPLO 6.
      COM SOLUCAO DE UM SISTEMA DE DUAS EQUACOES A DUAS INCOGNITAS.
      COM A1 * X + B1 * Y = C1
      COM A2 * X + B2 * Y = C2
      ARIT N = 0
   50 ARIT N = N + 1
      SE N MAI 5 ENTAO 30
      LEIA A1,B1,C1,A2,B2,C2
      IMPRIMA A1,B1,C1,A2,B2,C2
      ARIT D = A1*B2-A2*B1
      SE D IGU 0 ENTAO 20
      ARIT X = (C1*B2-C2*B1)/D
      ARIT Y = (A1*C2-A2*C1)/D
      IMPRIMA 'SOLUCAO DO SISTEMA','X =',X,'Y =',Y
      VA PARA 50
   20 IMPRIMA 'SIST.INDETERMINADO'
      VA PARA 50
   30 IMPRIMA 'FIM'
      FIM

****** INTERPRETACAO COMCOM ******   FIM DE EXECUCAO.
```

CONTROL DICTIONARY

```
                                        $TEXT  BASIC

BINARY CARD 00000

                                  FPM   MACRO  X
                                        FMP    X
                                        LRS    35
                                        ENDM   FPM
                                  FPD   MACRO  X
                                        LLS    35
                                        FDP    X
                                        ENDM   FPD
                                  FPDI  MACRO  X
                                        LLS    35
                                        FDP*   X
                                        ENDM   FPDI
                                  FPMI  MACRO  X
                                        FMP*   X
                                        LRS    35
                                        ENDM   FPMI
                                        CALL   LRESET
                                  *** 0
 00003  0500 00 0 00411 10001           CLA    =0.0
 00004  0601 00 0 00335 10001           STO    N
 00005  2 00000 0 00000 00001 T00050    BSS    0
 00005  0500 00 0 00335 10001           CLA    N
 00006  0300 00 0 00413 10001           FAD    =01.0
                                  *** N + 1
 00007  0601 00 0 00335 10001           STO    N
                                  *** N
                                  *** 5
 00010  0500 00 0 00414 10001           CLA    =05.0
 00011  0302 00 0 00335 10001           FSB    N
 00012 -0300 00 0 00415 10001           UFA    =0233000000000
 00013  0767 00 0 00012 10000           ALS    10
 00014  0771 00 0 00012 10000           ARS    10
 00015 -0120 00 0 00021 10001           TMI    100030
 00016  0534 00 4 00412 10001           LXA    =6,4
 00017  0500 00 0 00026 10001           CLA    ST.01-7
 00020  0402 00 0 00412 10001           SUB    =6
 00021  0621 00 0 00354 10001           STA    SAVE.-2

BINARY CARD 00001
 00022  0767 00 0 00022 10000           ALS    18
 00023  0622 00 0 00355 10001           STD    SAVE.-1
 00024  0500 00 0 00355 10001           CLA    SAVE.-1
 00025 -1704 00 0 00006 10000           TMT    6
 00026  0020 00 0 00035 10001           TRA    ST.01
 00027  0 00000 0 00336 10001           PZE    A1
 00030  0 00000 0 00337 10001           PZE    B1
 00031  0 00000 0 00340 10001           PZE    C1
 00032  0 00000 0 00341 10001           PZE    A2
 00033  0 00000 0 00342 10001           PZE    B2
 00034  0 00000 0 00343 10001           PZE    C2
 00035  2 00000 0 00000 00001 ST.01     BSS    0
                                        CALL   BSCRD(R)
```

```
 00041  0500 00 0 00353 10001          CLA     R
 00042  0601 60 4 00035 10001          STO*    ST.01,4
 00043  1 77777 4 00044 10001          TXI     **+1,4,-1

BINARY CARD 00002
 00044  3 00000 4 00035 10001          TXH     ST.01,4,0
 00045  0500 00 0 00354 10001          CLA     SAVE.-2
 00046 -1704 00 0 00006 10000          TMT     6
                                       CALL    PORLIN(BRACK)
                                       CALL    POREAL(A1)
                                       CALL    PORLIN(BRACK)
                                       CALL    POREAL(B1)
                                       CALL    PORLIN(BRACK)

BINARY CARD 00003.
                                       CALL    POREAL(C1)
                                       CALL    PORLIN(BRACK)
                                       CALL    POREAL(A2)
                                       CALL    PORLIN(BRACK)

BINARY CARD 00004
                                       CALL    POREAL(B2)
                                       CALL    PORLIN(BRACK)
                                       CALL    POREAL(C2)
                                       CALL    IMPRME
 00132  0560 00 0 00336 10001          LDQ     A1
                                       FPM     B2

BINARY CARD 00005
 00135 -0600 00 0 00345 10001          STQ     T.01
 00136  0560 00 0 00341 10001          LDQ     A2
                                       FPM     B1
 00141 -0600 00 0 00346 10001          STQ     T.02
 00142  0500 00 0 00345 10001          CLA     T.01
 00143  0302 00 0 00346 10001          FSB     T.02
                                *** A1 * B2 - A2 * B1
 00144  0601 00 0 00344 10001          STO     D
                                *** D
                                *** 0
 00145  0500 00 0 00411 10001          CLA     =0.0
 00146  0302 00 0 00344 10001          FSB     D
 00147 -0300 00 0 00415 10001          UFA     =0233000000000
 00150  0767 00 0 00012 10000          ALS     10
 00151  0771 00 0 00012 10000          ARS     10
 00152  0100 00 0 00275 10001          TZE     T00020
 00153  0560 00 0 00340 10001          LDQ     C1.
                                       FPM     B2
 00156 -0600 00 0 00345 10001          STQ     T.01
 00157  0560 00 0 00343 10001          LDQ     C2
                                       FPM     B1

BINARY CARD 00006
 00162 -0600 00 0 00346 10001          STQ     T.02
 00163  0500 00 0 00345 10001          CLA     T.01
 00164  0302 00 0 00346 10001          FSB     T.02
```

```
                                        *** C1 * B2 - C2 * B1
 00165  0601 00 0 00350 10001                   STO     T.03
 00166  0560 00 0 00350 10001                   LDQ     T.03
                                               FPD     D
                                        *** ( C1 * B2 - C2 * B1 ) / D
 00171 -0600 00 0 00347 10001                   STO     X
 00172  0560 00 0 00336 10001                   LDQ     A1
                                               FPM     C2
 00175 -0600 00 0 00345 10001                   STO     T.01
 00176  0560 00 0 00341 10001                   LDQ     A2
                                               FPM     C1
 00201 -0600 00 0 00346 10001                   STO     T.02
 00202  0500 00 0 00345 10001                   GLA     T.01

BINARY CARD 00007
 00203  0302 00 0 00346 10001                   FSB     T.02
                                        *** A1 * C2 - A2 * C1
 00204  0601 00 0 00350 10001                   STO     T.03
 00205  0560 00 0 00350 10001                   LDQ     T.03
                                               FPD     D
                                        *** ( A1 * C2 - A2 * C1 ) / D
 00210 -0600 00 0 00351 10001                   STO     Y
                                               CALL    PORLIN(BRACK)
                                               CALL    PORLIN(=HSOLUCA)
                                               CALL    PORLIN(=HO DO S)
                                               CALL    PORLIN(=HISTEMA)

BINARY CARD 00010
                                               CALL    PORLIN(BRACK)
                                               CALL    PORLIN(=HX =    )
                                               CALL    PORLIN(BRACK)
                                               CALL    POREAL(X)
                                               CALL    PORLIN(BRACK)

BINARY CARD 00011
                                               CALL    PORLIN(=HY =    )
                                               CALL    PORLIN(BRACK)
                                               CALL    POREAL(Y)
                                               CALL    IMPRME

BINARY CARD 00012
 00274  0020 00 0 00005 10001                   TRA     T00050
 00275  2 00000 0 00000 00001 T00020 BSS       0
                                               CALL    PORLIN(BRACK)
                                               CALL    PORLIN(=HSIST.I)
                                               CALL    PORLIN(=HNDETER)
                                               CALL    PORLIN(=HMINADO)
                                               CALL    IMPRME

BINARY CARD 00013
 00320  0020 00 0 00005 10001                   TRA     T00050
 00321  2 00000 0 00000 00001 T00030 BSS       0
                                               CALL    PORLIN(BRACK)
                                               CALL    PORLIN(=HFIM    )
                                               CALL    IMPRME
```

```
 00334  0020 00 0 20000 10011          TRA     S.JXIT
 00335  2 00000 0 00001 00001 N        BSS     1
 00336  2 00000 0 00001 00001 A1       BSS     1
 00337  2 00000 0 00001 00001 B1       BSS     1

BINARY CARD 00014
 00340  2 00000 0 00001 00001 C1       BSS     1
 00341  2 00000 0 00001 00001 A2       BSS     1
 00342  2 00000 0 00001 00001 B2       BSS     1
 00343  2 00000 0 00001 00001 C2       BSS     1
 00344  2 00000 0 00001 00001 D        BSS     1
 00345  2 00000 0 00001 00001 T.01     BSS     1
 00346  2 00000 0 00001 00001 T.02     BSS     1
 00347  2 00000 0 00001 00001 X        BSS     1
 00350  2 00000 0 00001 00001 T.03     BSS     1
 00351  2 00000 0 00001 00001 Y        BSS     1
                                       EXTERN  IMPRME,PORLIN,POREAL,PORI
 00352  363660606060     10000 BRACK   OCT     363660606060
                                       EXTERN  LRESET
                                       EXTERN  BSCRD
 00353  2 00000 0 00001 00001 R        BSS     1
 00354  0 00356 0 00000 10100          PZE     **,,SAVE.
 00355  0 00000 0 00356 10001          PZE     SAVE.,,**
 00356  2 00000 0 00024 00001 SAVE.    BSS     20
                                          LITERALS
 00402 -306013606060     10000
 00403 -276013606060     10000
 00404 -224643642321     10000
 00405 -223162633331     10000

BINARY CARD 00015
 00406 -066024466062     10000
 00407 -052425632551     10000
 00410 -043145212446     10000
 00411  000000000000     10000
 00412  000000000006     10000
 00413  201400000000     10000
 00414  203500000000     10000
 00415  233000000000     10000
 00416  263144606060     10000
 00417  316263254421     10000
                                       EXTERN     S.JXIT   GENERATED
                    00000 01111        END

                    $DKEND BASIC
```

```
 1.0   2.0   3.0   3.0   4.0   7.0
  1.0   2.0   3.0   2.0   4.0   6.0
 .10000000E1  .20000000E1  .30000000E1  .30000000E1  .40000000E1  .70000000
 SOLUCAODOSISTEMA  X=  .10000000E1  Y=  .10000000E1
  4.7   2.5   6.25   1.8   5.5   1.0
 .10000000E1  .20000000E1  .30000000E1  .20000000E1  .40000000E1  .60000000
 SIST.INDETERMINADO
  1.5   2.5   3.5   5.0   7.5   9.0
 .47000000E1  .25000000E1  .62499999E1  .18000000E1  .54999999E1  .10000000
 SOLUCAODOSISTEMA  X=  .14929742E1  Y=  -.30679157E-0
  20.0   45.0   9.5   32.5   42.00   2.5
 .15000000E1  .25000000E1  .35000000E1  .50000000E1  .74999999E1  .90000000
 SOLUCAODOSISTEMA  X=  -.30000000E1  Y=  .32000000E1
 .20000000E2  .45000000E2  .94999999E1  .32500000E2  .42000000E2  .25000000
 SOLUCAODOSISTEMA  X=  -.46024097E-0  Y=  .41566265E-0
 FIM
```

```
$OBCOM          NOLIST
      COM   PROGRAMA EXEMPLO 8.
      COM   PROGRAMA COM ERRO DE COMPILACAO.
      DIM A(15),A(12)
      PARA I=1 ATE 15
  ***  VARIAVEL REDEFINIDA. A ( 12 ) IGNORADA.
      LEIA A(I)
      PROXIMO J
  ***  AGRUPAMENTO INCORRETO DE COMANDOS 'PARA E 'PROXIMO'.
      FIM

****** INTERPRETACAO COMCOM ******   FIM DE EXECUCAO.
```

```
$OBCOM          NOLIST
      COM PROGRAMA EXEMPLO 9.
      COM   PROGRAMA COM ERRO DE EXECUCAO.
      PARA I=0 ATE 15
      ARIT X(I) = I
      IMPRIMA X(I)
      PROXIMO I
      FIM

****** INTERPRETACAO COMCOM ******   FIM DE EXECUCAO.
```

```
3090432038AA  AZEREDO                                  IBMAP ASSEMBLY BASIC
              CONTROL DICTIONARY

                                        $TEXT  BASIC


BINARY CARD 00000
                                        FPM    MACRO   X
                                               FMP     X
                                               LRS     35
                                               ENDM    FPM
                                        FPD    MACRO   X
                                               LLS     35
                                               FDP     X
                                               ENDM    FPD
                                        FPDI   MACRO   X
                                               LLS     35
                                               FDP*    X
                                               ENDM    FPDI
                                        FPMI   MACRO   X
                                               FMP*    X
                                               LRS     35
                                               ENDM    FPMI
                                               CALL    LRESET
                                        *** 0
 00003   0500 00 0 00104 10001                 CLA     =0.0
 00004   0601 00 0 00065 10001                 STO     I
 00005   2 00000 0 00000 00001 D00001  BSS     0
 00005   0500 00 0 00107 10001                 CLA     =015.0
 00006   0302 00 0 00065 10001                 FSB     I
 00007  -0300 00 0 00110 10001                 UFA     =0233000000000
 00010   0767 00 0 00012 10000                 ALS     10
 00011   0771 00 0 00012 10000                 ARS     10
 00012  -0120 00 0 00064 10001                 TMI     E00001+1
                                        *** I
                                               CALL    TOFIX(I,T.01)
                                               CALL    TINDEX(T.01,=11)

BINARY CARD 00001
                                        *** I
 00025   0535 00 1 00101 10001                 LAC     T.01,1
 00026   0500 00 0 00065 10001                 CLA     I
 00027   0601 00 1 00066 10001                 STO     X,1
                                        *** I
                                               CALL    TOFIX(I,T.01)
                                               CALL    TINDEX(T.01,=11)
 00042   0535 00 1 00101 10001                 LAC     T.01,1
 00043   0500 00 1 00066 10001                 CLA     X,1
                                        *** X ( I )
 00044   0601 00 0 00102 10001                 STO     T.02
                                               CALL    PORLIN(BRACK)

BINARY CARD 00002
                                               CALL    POREAL(T.02)
                                               CALL    IMPRME
 00060   0500 00 0 00065 10001                 CLA     I
 00061   0300 00 0 00106 10001                 FAD     =1.0
 00062   0601 00 0 00065 10001                 STO     I
 00063   0020 00 0 00005 10001 E00001  TRA     D00001
```

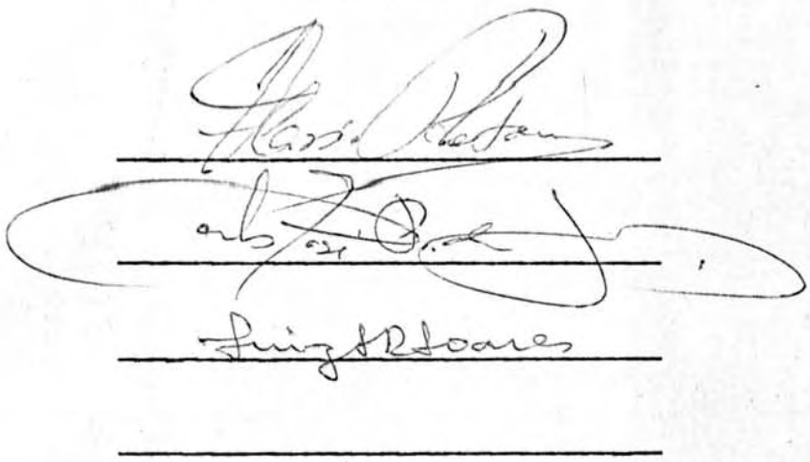
```
 00064  0020 00 0 22000 10011          TRA     S.JXIT
 00065  2 00000 0 00001 00001 I        BSS     1
 00066  2 00000 0 00013 00001 X        BSS     11
 00101  2 00000 0 00001 00001 T.01     BSS     1

BINARY CARD 00003
 00102  2 00000 0 00001 00001 T.02     BSS     1
                                       EXTERN  IMPRME,PORLIN,POREAL,PORIN
 00103  363660606060     10000 BRACK   OCT     363660606060
                                       EXTERN  LRESET
                                       EXTERN  TOFIX
                                       EXTERN  TINDEX
                                          LITERALS
 00104  000000000000     10000
 00105  000000000013     10000
 00106  201400000000     10000
 00107  204740000000     10000
 00110  233000000000     10000
                                       EXTERN     S.JXIT   GENERATED
                    00000 01111        END

                    $DKEND BASIC
```

```
OBJECT PROGRAM IS BEING ENTERED INTO STORAGE.
 .0E-38
 .10000000E1
 .20000000E1
 .30000000E1
 .40000000E1
 .50000000E1
 .60000000E1
 .70000000E1
 .80000000E1
 .90000000E1
 .10000000E2
```

309043203PAA AZEREDO

*** SUBSCRITO FORA DOS LIMITES ESPECIFICADOS.

TESE APRESENTADA AOS SRS.



Visto e permitida a impressão
Rio de Janeiro, .../...../.....

Coordenador dos Programas de Pós-Graduação e
Pesquisas do Centro Técnico Científico.