1. Publicação nº	2. Versão	3. Data	5. Distribuição
INPE-2781-PRE/347		*Junho, 1983*	☐ Interna ☒ Externa ☐ Restrita

4. Origem	Programa
DIN	*LANAC/INFORMÁTICA*

6. Palavras chaves - selecionadas pelo(s) autor(es)

LANAC
LINGUAGEM DE PROGRAMAÇÃO
PROGRAMAÇÃO CONCORRENTE

7. C.D.U.: *681.322.0L*

8. Título *INPE-2781-PRE/347*

O DESENVOLVIMENTO DA LINGUAGEM LANAC/II PARA PROGRAMAÇÃO CONCORRENTE

10. Páginas: *22*

11. Última página: *19*

12. Revisada por

Heloisa Galvão V.S.Borges

9. Autoria *Sérgio Donizetti Fischer*
Vânia Aparecida Dinardo Oleinki
Celso de Renna e Souza

13. Autorizada por

Nelson de Jesus Parada
Diretor

Assinatura responsável

14. Resumo/Notas

LANAC é uma linguagem do tipo Algol, desenvolvida no INPE/CNPq, orientada para programação concorrente. Depois de utilizar uma primeira versão da linguagem na aquisição e reprodução de imagens meteorológicas, constatou-se a necessidade de definir uma segunda versão, LANAC/II, com algumas novas estruturas. A principal delas é a possibilidade de definir regiões críticas para compartilhamento de recursos e verificação, em tempo de compilação, do uso correto destes recursos, o que torna os programas concorrentes mais confiáveis. O compilador LANAC/II está sendo escrito em Algol do Burroughs 6800 e deverá gerar código "assembly" simbólico para o minicomputador SISCO MB8000.

15. Observações

Trabalho submetido ao 10º Seminário Integrado de Software e Hardware - SEMISH que se realizará em Campinas de 25 a 29/07/1983.

O DESENVOLVIMENTO DA LINGUAGEM LANAC/II PARA PROGRAMAÇÃO CONCORRENTE.

Sérgio Donizetti Fischer
Vânia Aparecida Dinardo Oleinki
Celso de Renna e Souza

INSTITUTO DE PESQUISAS ESPACIAIS - INPE
CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTÍFICO E TECNOLÓGICO - CNPq
Caixa Postal 515
Fone: (0123) 22-9977 Telex: (011) 33530
12200, São José dos Campos, São Paulo

RESUMO

LANAC é uma linguagem de alto nível que tem sido desenvolvida no INPE/CNPq e orientada para programação concorrente. Uma primeira versão foi implementada para os minicomputadores HP2116B e PDP11/10 e utilizada para escrever alguns programas de aquisição e reprodução de imagens meteorológicas. Depois de constatar a necessidade de enriquecer esta primeira versão, definiu-se a linguagem LANAC/II como uma extensão da versão anterior, também em Português. Nela introduziram-se novos tipos de dados como: variáveis compartilhadas e reais, comandos para definir regiões críticas, mecanismos para atribuir prioridades a processos e outros mecanismos para suspensão destes por um certo intervalo de tempo. O compilador LANAC/II está sendo inicialmente escrito em Algol do Burroughs 6800 e deverá gerar código "assembly" simbólico, a partir de uma forma intermediária do tipo quádrupla, para o minicomputador SISCO MB8000. Atualmente, encontram-se implementadas, com exceção do recupera

dor de erros sintáticos, toda a parte de análise e, também, a que traduz o programa fonte para o código intermediário. O gerador de código teve a sua implementação iniciada, enquanto a definição do cerne que suportará a linguagem está em andamento.

RESUMO ESTENDIDO

Dentre as diversas atividades do INPE, a aquisição, a reprodução e a disseminação, em tempo real, de imagens enviadas por satélites têm grande importância. Estas atividades são normalmente gerenciadas por minicomputadores e apresentam características típicas de programação em tempo real e de concorrência entre processos. Como todas estas atividades vinham sendo implementadas em "assembly", decidiu-se pelo projeto e implementação da linguagem do tipo Algol LANAC (Linguagem de Alto Nível para Aquisição de Dados e Controle de Processos) [3], com comandos em Português e algumas ferramentas que facilitam a programação destas aplicações, tais como semáforos e definição de processos paralelos.

Duas implementações desta primeira versão (LANAC/I) foram realizadas. Na primeira, para o HP2116B, o objetivo principal foi o desenvolvimento de um protótipo do compilador, e, por isso, não se deu muita ênfase ao projeto do cerne que a suporta [1]. Na segunda implementação, o minicomputador escolhido foi o PDP11/10, onde se deu um enfoque maior ao desenvolvimento do núcleo monitor, o qual foi definido de maneira mais elaborada e mais eficiente [1,2]. Nestas duas máquinas, a linguagem LANAC/I foi utilizada na programação de algumas tarefas de aquisição e de reprodução de imagens meteorológicas, obtendo-se bons resultados. Contudo, estes testes mostraram que seria muito útil que a linguagem dispusesse de novos tipos, novas estruturas de dados e comandos de controle mais poderosos.

Assim, definiu-se uma linguagem concebida para uso geral em programação em tempo real, LANAC/II, que é uma extensão da versão anterior, também em Português, na qual se deu mais consistência às estruturas já existentes e introduziram-se novos tipos de dados e outros comandos de controle.

A linguagem LANAC/II permite declarar variáveis inteiras, lógicas e reais; semáforos; rótulos e listas, sendo a listas usadas para imprimir mensagens pela impressora. Os arranjos são unidimensionais

e são do mesmo tipo das variáveis. Tanto as variáveis como os vetores podem ser declarados com o atributo "COMUM", que indica se elas são ou não compartilhadas pelos processos paralelos. Em caso positivo, o compilador irá verificar se elas são usadas somente dentro de seções críticas. A linguagem possibilita, ainda, a declaração de sub-rotinas ou funções, cujos parâmetros são passados por referência ou por valor. Estes procedimentos podem ser externos, ou seja, o seu código é carregado antes da execução do programa na máquina-alvo.

Além dos comandos tradicionais para controle de fluxo. LANAC/II possui outros dirigidos à programação concorrente, que são:

a) comando paralelo: permite a definição de paralelismo entre processos, sendo possível atribuir uma prioridade inicial a cada um deles;

b) comando "PRIORIDADE": facilita a um processo alterar a sua prioridade durante o tempo de execução;

c) comando "ATRASE": causa a suspensão, por um certo intervalo de tempo, do processo que o executou.

d) comando "TEMPO": quando os processos forem escalados por fatias de tempo, este comando obtém o valor inteiro correspondente ao número atual de pulsos do relógio;

e) comando "EVENTEX": permite a um processo verificar e esperar ocorrências de interrupções de dispositivos periféricos;

f) comandos "ESPERE" E "CAUSE": são as primitivas que operam sobre os semáforos e correspondem, respectivamente, às operações "P" e "V" de Dijkstra;

g) comando "CONTROLES": permite o envio de instruções especiais de controle aos dispositivos periféricos;

h) comando "ESTADOES": possibilita a leitura de informações a respeito do estado de funcionamento de um periférico;

i) comando "RESERVE": indica a definição de regiões críticas simples e condicionais, onde as variáveis e vetores compartilhados podem ser usados.

O compilador da linguagem LANAC/II está sendo inicialmente escrito em Algol do computador Burroughs 6800 e deverá gerar código "assembly" para o minicomputador SISCO MB8000. O método de análise sintática utilizado é o ESLL (1) e a tradução é dirigida pela sintaxe, onde o programa fonte é traduzido para uma forma intermediária, do tipo quádrupla.

No estágio atual do trabalho, todos os analisadores estão implementados até a geração do código intermediário, com exceção do recuperador de erros. O gerador de código encontra-se em fase inicial de implementação, enquanto a definição do cerne que suportará a linguagem está em andamento.

REFERÊNCIAS

[1] FISCHER, S.D.; SILVA, L.J.N. da; SOUZA, C.R. *LANAC: uma experiência no desenvolvimento de uma linguagem para controle de processos*. São José dos Campos, INPE, Abr. 1982. 8 p. (INPE-2377-PRE/102).

[2] SILVA, L.J.N. da *Cerne ("kernel") de sistema operacional para suporte de programas em LANAC*. Dissertação de Mestrado em Computação Aplicada. São José dos Campos, INPE. No prelo.

[3] VIOLA, F.E.C. *LANAC - uma linguagem de 'alto nível' para aquisição de dados e controle de processos por minicomputador*. Dissertação de Mestrado em Computação Aplicada. São José dos Campos. 1980. (INPE-1903-TDL/031).

1. INTRODUÇÃO

Nos últimos anos, o uso de computadores, principalmente os minis e micros, está crescendo de maneira bastante intensa, nas atividades que envolvem controle de processos em tempo real. No INPE/CNPq, por exemplo, todo o processo de aquisição, processamento e disseminação de imagens enviadas por satélites meteorológicos é realizado com o controle de minicomputadores. As imagens são adquiridas em tempo real, gravadas em fitas magnéticas, ou reproduzidas diretamente em filme fotográfico, através de equipamentos imageadores.

Para programar aplicações deste tipo, era muito comum o uso de "assembly" ou de extensões do Fortran, com a inclusão de chamadas ao sistema operacional. Devido às características destas atividades, tais como concorrência, sincronização e compartilhamento de recursos por vários processos, ocorrem muitos erros dependentes do tempo, pois as linguagem citadas não provêm estruturas convenientes para representar estas características. Além disso, tanto a manutenção como a depuração dos programas é bastante difícil.

Apesar de ainda se usar "assembly" ou Fortran no desenvolvimento de sistemas para controle de processos e de tempo real, hoje existe uma grande tendência para utilizar linguagens de alto nível, voltadas para programação concorrente, na implementação destas aplicações. Com esta prática, é possível tornar os programas mais confiáveis e inteligíveis, sendo portanto mais fácil a sua depuração e correção.

Com base nesta motivação, a linguagem LANAC (Linguagem de Alto Nível para Aquisição de dados e Controle de processos) está sendo desenvolvida no INPE/CNPq, visando a atender às necessidades do Departamento de Meteorologia do Instituto, já que, devido à inexistência de uma linguagem e de um "software" adequados, todos os programas de aquisição, processamento e reprodução de imagens meteorológicas são escritos em "assembly", trazendo, com isso, alguns problemas decorrentes do uso deste tipo de linguagem.

Nos tópicos seguintes apresentam-se um pequeno histórico do projeto LANAC, uma breve descrição da linguagem LANAC/II, alguns detalhes da implementação do compilador, bem como o estágio atual do trabalho e suas perspectivas futuras.

2. HISTÓRICO DO PROJETO LANAC

O projeto da linguagem LANAC surgiu para atender dois objetivos básicos: o primeiro, desenvolver ferramentas próprias de trabalho, sobre as quais se possa ter total domínio; o segundo, adquirir técnicas mais sofisticadas no projeto de sistemas de tempo real.

A linguagem LANAC, na sua primeira versão (LANAC/I) [11, 12], foi baseada no Algol 60 [6], apresentando comandos e estruturas adicionais adequadas aos seus objetivos. Todos os seus comandos são em Português, ela não apresenta recursividade e permite a definição de processos paralelos. A sincronização e a comunicação entre processos é possível através da estrutura de semáforos [2].

O compilador LANAC/I foi escrito na linguagem Algol do sistema Burroughs 6800, gerando código para os minicomputadores HP 2116B e PDP 11/10. O suporte de execução para o HP2116B apresenta um cerne ("Kernel") bastante simples, pois o objetivo principal era ter, a curto prazo, um protótipo do compilador. Para o PDP11/10 da Digital, o suporte de execução [10] é composto de um núcleo mais aprimorado, onde, além dos recursos necessários ao suporte desta primeira versão, estão incluídas primitivas e estruturas para atender à segunda versão.

Depois de efetuados alguns testes com estas duas implementações, sentiu-se a necessidade de estender a linguagem, introduzindo novas estruturas de controle e novos tipos de dados, de modo a torná-la mais eficiente.

A maior dificuldade encontrada no uso de LANAC/I foi a ausência de comandos adequados à definição de regiões críticas, o que

impossibilita garantir maior confiabilidade aos programas implementados. Embora a definição de uma região crítica através de semáforos seja bastante simples, ela não permite que o compilador verifique se as regiões do programa, onde os recursos compartilhados são usados, estão corretamente estabelecidas. Além disso, como não é possível identificar uma região crítica em tempo de compilação, deve-se permitir que variáveis comuns sejam usadas em qualquer ponto do programa, o que pode acarretar erros dependentes do tempo.

Desta maneira partiu-se para a definição e implementação da segunda versão da linguagem, denominada LANAC/II, na qual foram introduzidos novos tipos de dados, tais como variáveis compartilhadas e do tipo real, outros comandos de controle para definição de regiões críticas simples e condicionais, e mecanismos para atribuição de prioridade a processos.

Todas as características da primeira versão foram essencialmente mantidas, principalmente no que diz respeito à simplicidade da linguagem resultante. Desta forma a linguagem LANAC/II é quase que totalmente compatível com a primeira versão, com exceção de pequenas alterações que a tornaram mais clara.

3. ESPECIFICAÇÃO DA LINGUAGEM LANAC/II

Os comandos da linguagem podem ser agrupados em blocos, que são delimitados pelas palavras reservadas "INICIE" e "TERMINE". Um bloco consiste em um conjunto de declarações, que pode ser omitido, seguido de um ou mais comandos.

Usar-se-á a notação BNF ("Backus Normal Form") para descrever a sintaxe da linguagem.

3.1 - DECLARAÇÕES

Quanto aos tipos de dados, permite-se declarar variáveis inteiras, lógicas e reais, identificadores de rótulos, semáforos, lista de caracteres, vetores unidimensionais, sub-rotinas e funções.

Todos os identificadores devem ser declarados e o seu escopo segue as mesma regras da linguagem Algol.

3.1.1 - DECLARAÇÃO DE VARIÁVEIS

<DECL-VARIÁVEL> :: =

<TIPO> COMUM <SEQ-DE-IDENTIFICADORES> I

<TIPO> <SEQ-DE-IDENTIFICADORES>

<TIPO> :: =

INTEIRO I

LÓGICO I

REAL

<SEQ-DE-IDENTIFICADORES> :: =

<IDENTIFICADOR> I

<IDENTIFICADOR> , <SEQ-DE-IDENTIFICADORES>

onde <IDENTIFICADOR> denota uma cadeia, na qual o primeiro caractere deve ser uma letra e os demais podem ser dígitos, letras ou o caractere "%".

Os identificadores, que na declaração apresentam o atributo "COMUM", são considerados variáveis compartilhadas e só podem ser referenciados dentro de regiões críticas.

3.1.2 - DECLARAÇÃO DE SEMÁFOROS, RÓTULOS E LISTAS

<SEMÁFOROS> :: =

SEMAFORO <SEQ-DE-IDENTIFICADORES>

<LISTA> :: =

LISTA <IDENTIFICADOR> <CADEIA-DE-CARACTERES>

<RÓTULO> :: =

ROTULO <SEQ-DE-IDENTIFICADORES>

onde <CADEIA-DE-CARACTERES> denota um cordão de caracteres delimitados por aspas (").

Semáforos são indicados para a troca de sinais e para a sincronização entre processos. Um semáforo está aberto se o seu valor for positivo, e fechado, se o seu valor for zero.

3.1.3 - DECLARAÇÃO DE VETORES

<DECL-VETOR> :: =

<TIPO> VETOR <SEQ-DE-VETORES> I

<TIPO> VETOR COMUM <SEQ-DE-VETORES>

<SEQ-DE-VETORES> ::=

<SEQ-DE-IDENTIFICADORES> <DIMENSÃO> I

<SEQ-DE-IDENTIFICADORES> <DIMENSÃO> , <SEQ-DE-VETORES>

<DIMENSÃO> ::=

[<INTEIRO-DECIMAL> : <INTEIRO-DECIMAL>]

O atributo "COMUM" denota vetores compartilhados por processos paralelos.

3.1.4 - DECLARAÇÃO DE PROCEDIMENTOS

<PROCEDIMENTO> :: =

<TIPO-SUB> SUBROTINA <IDENTIFICADOR> ; <CORPO-SUB-ROTINA> I

<TIPO-SUB> SUBROTINA <IDENTIFICADOR> ; CODIGO I

<TIPO-SUB> SUBROTINA <IDENTIFICADOR>

(<SEQ-DE-IDENTIFICADORES>)

; <DECL-PARÂMETROS> <CORPO-SUB-ROTINA> I

<TIPO-SUB> SUBROTINA <IDENTIFICADOR>

(<SEQ-DE-IDENTIFICADORES>)

;<DECL-PARÂMETRO> CODIGO

<DECL-PARÂMETROS> :: =

VALOR <SEQ-DE-IDENTIFICADORES> ; <TIPO-DOS-PARÂMETROS> ; I

<TIPO-DOS-PARÂMETROS> ;

<TIPO-SUB> :: =

INTEIRO I

LÓGICO I

REAL I

ε

<TIPO-DOS-PARÂMETROS> :: =

<DECLARAÇÃO-DOS-TIPOS> I

<DECLARAÇÃO-DOS-TIPOS> <TIPO-DOS-PARÂMETROS>

<DECLARAÇÃO-DOS-TIPOS> :: =

<DECL-VARIÁVEL> I

<DECL-VETOR>

onde ε denota a cadeia vazia, <CORPO-SUB-ROTINA> denota um bloco e "CODIGO" indica procedimento externo, cujo código será carregado antes da execução na máquina-alvo.

Os parâmetros podem ser passados por referência ou por valor.

3.2 - COMANDOS

Os comandos disponíveis na linguagem podem ser divididos nas seguintes classes:

3.2.1 - COMANDOS DE ATRIBUIÇÃO

<ATRIBUIÇÃO> :: =

<ATRIBUIÇÃO-MÚLTIPLA> I

<VAR-1> INV <VAR-1> I

<VAR-1> REQ <VAR-1> I

<ATRIBUIÇÃO-MÚLTIPLA> :: =

<VAR-2> : = <LADO-DIREITO> I

<VAR-2> : = <ATRIBUIÇÃO-MÚLTIPLA>

<LADO-DIREITO> :: =

SE <EXPRESSÃO> ENTÃO <EXPRESSÃO> SENÃO <LADO-DIREITO> I

<EXPRESSÃO>

onde <VAR-1> indica um elemento de vetor, ou um identificador declarado como inteiro, real ou lógico; <VAR-2>, por sua vez, denota um identificador inteiro, lógico ou real, um identificador de semáforo ou de função, ou um elemento de vetor.

O operador "INV" causa a troca dos conteúdos das variáveis envolvidas, e o operador "REQ" realiza a rotação, da variável do lado direito, de 4 bits à esquerda e atribui o resultado à variável do lado esquerdo.

3.2.2 - COMANDOS CONDICIONAIS

<COMANDOS-CONDICIONAIS> :: =

SE <EXPRESSÃO> ENTAO <COMANDO> SENÃO <COMANDO> I

SE <EXPRESSÃO> ENTAO <COMANDO> I

CASO <EXPRESSÃO> : INICIE <COMANDO> TERMINE

onde <COMANDO> denota um bloco ou um comando simples.

3.2.3 - COMANDOS ITERATIVOS

<COMANDOS-ITERATIVOS> :: =

REPITA <COMANDO> SEMPRE I

REPITA <COMANDO> ATE <EXPRESSÃO> I

ENQUANTO <EXPRESSÃO> EXECUTE <COMANDO> I

PARA <VAR-1> : = <EXPRESSÃO> PASSO <EXPRESSÃO> ATE <EXPRESSÃO>
EXECUTE <COMANDO> I

PARA <VAR-1> : = <EXPRESSÃO> ATE <EXPRESSÃO> EXECUTE <COMANDO>

3.2.4 - COMANDOS INCONDICIONAIS

<COMANDO-INCONDICIONAIS> :: =

VA <IDENTIFICADOR> I

<IDENTIFICADOR> I

<IDENTIFICADOR> (<SEQ-DE-IDENTIFICADORES>) I

<COMANDO-PARE> I

<COMANDO-CONTROLES-ESTADOES> I

<COMANDO-ESPERE-CAUSE> I

<COMANDO-ATRASE> I

<COMANDO-TEMPO> i

<COMANDO-PRIORIDADE> I

<COMANDO-EVENTEX>

A) <COMANDO-PARE > :: =

PARE

Este comando interrompe, incondicionalmente, a execução de um programa.

B) <COMANDOS-CONTROLES-ESTADOES> :: =

CONTROLES (<DISPOSITIVO> , <EXPRESSÃO>) I

ESTADOES (<DISPOSITIVO> , EXPRESSÃO)

<DISPOSITIVO> :: =

<INTEIRO-DECIMAL> I

<INTEIRO-OCTAL> I

<VAR-3>

onde <INTEIRO-OCTAL> é uma cadeia de dígitos de 0 a 7, precedida pelo símbolo "@" e <VAR-3> denota um identificador ou um elemento de vetor inteiros.

O comando "CONTROLES" permite enviar instruções especiais de controle aos dispositivos externos, enquanto "ESTADOES" possibilita a leitura de informações a respeito do estado de funcionamento de um periférico.

C) <COMANDOS-ESPERE-CAUSE> :: =

ESPERE (<IDENTIFICADOR>) I

CAUSE (<IDENTIFICADOR>)

Estes dois comandos são definidos como as primitivas que operam sobre os semáforos, correspondendo às operações "p" e "V" de Dijkstra [2].

D) <COMANDO-ATRASE> :: =

ATRASE (<INTEIRO-DECIMAL>) I

ATRASE (<VAR-3>) I

ATRASE

Este comando permite atrasar a execução de um processo de duas maneiras: na primeira, quando o modo de escalação é por fatias de tempo cedidas ao processo, aquele que está em execução é bloqueado, voltando a ser executado após ter ocorrido o número de pulsos de relógio especificado entre parênteses; na segunda, quando o modo de escalação for por interrupção de dispositivos, o processo será suspenso e colocado na fila de processos ativos, onde competirá com outros pelos recursos da máquina. O mesmo ocorrerá quando o tempo não for especificado, independente do modo de escalação dos processos.

E) <COMANDO-TEMPO> :: =

TEMPO (<VAR-3>)

Este comando retorna, sobre a variável especificada, um valor inteiro correspondente ao número de pulsos dados pelo relógio. Caso o modo de escalação dos processos não seja por fatias de tempo, este comando não terá efeito.

F) <COMANDO-PRIORIDADE> :: =

PRIORIDADE (<INTEIRO-DECIMAL>)

Permite a um processo alterar a sua própria prioridade durante o tempo de execução. Este comando só deve aparecer dentro de um comando paralelo.

G) <COMANDO-EVENTEX> :: =

EVENTEX (<DISPOSITIVO> , <VAR-3> , <TEMPO-ESPERA>) I

EVENTEX (<DISPOSITIVO> , <VAR-3>)

3.2.6 - COMANDO PARALELO

<COMANDO-PARALELO> :: =

PARINICIE <PROCESSOS> PARTERMINE

<PROCESSOS> :: =

PRIORIDADE <INTEIRO-DECIMAL> : <COMANDO> I

<COMANDO> I

PRIORIDADE <INTEIRO-DECIMAL> : <COMANDO> ; <PROCESSOS> I

<COMANDO> ; <PROCESSOS>

Este comando define os comandos que serão executados de maneira concorrente. Cada comando especificado é denominado processo. A inicialização de cada processo é feita na entrada, quando é atribuída a ele a prioridade definida pelo inteiro decimal. Se esta parte for omitida, o processos assumirá a menor prioridade existente.

3.2.7 - ESPECIFICAÇÃO DE REGIÕES CRÍTICAS

<REGIÕES-CRÍTICAS> :: =

<REGIÕES-SIMPLES> I

<REGIÕES-CONDICIONAIS>

<REGIÕES-SIMPLES> :: =

RESERVE EXECUTE <COMANDO>

<REGIÕES-CONDICIONAIS> :: =

RESERVE <EXPRESSÃO> EXECUTE <COMANDO>

Esta estrutura combina as idéias de Hoare [4], de Hansen [3] e aquelas presentes na linguagem Iliad [7].

Na região crítica simples, um processo, ao tentar entrar na seção crítica, verifica se ela está livre, isto é, se não existe

outro processo executando-a. Se não existir, reserva-a para si, executa o comando e, no final, a libera. Caso a região crítica já tenha sido reservada, o processo é bloqueado até que tenha condições de entrar na região.

Quando se trata de uma região crítica condicional, o processo é bloqueado até que não exista um outro executando a seção crítica. Quando a região estiver livre, a expressão é avaliada e, caso seja verdadeira, o comando é executado, liberando em seguida a região para outros processos e dando prioridade àqueles que estão aguardando a expressão ser verdadeira. Porém, se a expressão for falsa, o processo é suspenso e a região é liberada.

4 - A IMPLEMENTAÇÃO DO COMPILADOR

O compilador da linguagem LANAC/II está sendo escrito em Algol do sistema Burroughs 6800 e deverá gerar código "Assembly" simbólico para o minicomputador SISCO MB8000. Esta máquina-alvo foi escolhida por estar sendo adotada pelo Departamento de Meteorologia do INPE/CNPq, no processo de aquisição, processamento e reprodução de imagens enviadas por satélites meteorológicos. É importante salientar que, como se pretende adquirir um compilador Algol que possa ser executado no SISCO MB8000, algumas decisões foram tomadas quanto ao uso do Algol do B6800, visando facilitar o futuro transporte do compilador.

O analisador léxico foi implementado de uma maneira clássica, através de um Autômato Finito Determinístico. Sua função é ler o programa fonte, ignorando os comentários, e identificar, dentro dele, as entidades léxicas, convertendo-as numa forma interna própria. A sua invocação ocorre toda vez que o analisador sintático necessita de um novo símbolo para continuar a análise.

A análise sintática da linguagem LANAC/II é realizada pelo método ESLL (1) ("Extended Simple LL (1)") [8,9]. Trata-se de um método descendente, que consiste em um analisador que aceita a linguagem

LANAC/II gerada a partir de uma gramática ESLL (1). Este método foi adotado por ser bastante simples, geral, eficiente e por facilitar a inclusão de recuperação de erros sintáticos. Outra vantagem deste método é a possibilidade de conseguir uma modularização do compilador, o que facilita a introdução de alterações ou extensões na linguagem e, igualmente a correção de erros de implementação.

Como os acessos à tabela de símbolos consomem muito tempo de compilação, realizou-se, com base nos estudos de Mckeeman [5], uma análise comparativa entre 4 métodos de acesso: Linear, "Hash", "Sort" e "Tree". Neste estudo, pôde-se observar que o método "Hash", para tabelas não muito pequenas, é o que sobrecarrega menos o mecanismo de acesso à memória. Deste modo, decidiu-se utilizá-lo na organização da tabela de símbolos, onde tanto os identificadores como as constantes são colocadas numa única tabela, o que simplifica bastante o mecanismo de acesso.

A geração de código objeto pelo compilador LANAC/II será realizada a partir de uma forma intermédiária, do tipo quádrupla. Com esta técnica, pode-se otimizar mais o código gerado, pois a linguagem intermediária está bem mais próxima do código objeto. Além disso, o compilador torna-se mais portátil, pois, numa implementação de um gerador de código para outra máquina-alvo, basta reescrever o módulo de geração de código, sem que seja necessário fazer modificações significativas nos analisadores. O esquema de tradução implementado é o dirigido pela sintaxe [1], onde as rotinas semânticas são introduzidas, em pontos adequados, nas produções da gramática e, ao serem chamadas pelo analisador sintático, geram o código intermediário.

5 - CONCLUSÕES

Atualmente, encontram-se implementadas as partes referentes às análises léxica e sintática, bem como o analisador semântico que traduz o programa fonte para o código intermediário. O gerador de código está em fase de desenvolvimento e, num futuro próximo, dever-se-á

iniciar a implementação do recuperador de erros sintáticos. Em paralelo a estas tarefas, está sendo definido o núcleo que dará suporte à execução dessa segunda versão da linguagem, o qual deverá ser programada em "Assembly" do minicomputador SISCO MB8000.

Com o produto final, poder-se-á obter um "Software", a curto prazo, bastante confiável devido à experiência obtida com o desenvolvimento das versões anteriores.

6 - REFERÊNCIAS

[1] AHO, A.V.; ULLMAN, J.D. *Principles of compiler design.* 3.ed. Reading. MA, Addison-Wesley, 1979.

[2] DIJKSTRA, E.W. The structure of T.H.E. multiprogramming system. *Communications of the ACM*, 11(5):341-346, May 1968.

[3] HANSEN, P.B. *Operating systems principles.* Englewood Clifs. NJ, Prentice-Hall, 1973.

[4] HOARE, C.A.R. Towards a Theory of parallel programming. In: HOARE, C.A.R.; PERROT, R.H., ed. *Operating systems techniques.* New York, NY, Academic Press, 1972. p. 61-71.

[5] MCKEEMAN, W.M. Symbol table access. In: BAUER, F.L.; EICKEL, J., ed. *Compiler construction - an advanced course.* 2. ed. Berlin, Springer-Verlag, 1976. cap. 3.D, p. 253-301.

[6] NAUR, P. Revised report on the algorithmic language ALGOL 60. *Communications of the ACM*, 6(1): 1-23, Jan. 1963.

[7] SCHUTZ, H.A. On the design of a language for programming real-time concurrent processes. *IEEE Transactions on Software Engineering*, 5(3):248-255, May 1979.

[8] SETZER, V.W.; MELLO, I.S.H. de *A construção de um compilador - parte I.* Campinas, Segunda Escola de Computação. IMECC-UNICAMP, 1981.

[9] ——— *A construção de um compilador - parte II.* Campinas, Segunda Escola de Computação, IMECC-UNICAMP, 1981.

[10] SILVA, L.J.N. da *Cerne ("kernel") de sistema operacional para suporte de programas em LANAC.* Dissertação de Mestrado em Computação Aplicada. São José dos Campos, INPE. No prelo.

[11] VIOLA, F.E.C. *LANAC - Uma linguagem de alto nível para aquisição de dados e controle de processos por minicomputador.* Dissertação de Mestrado em Computação Aplicada. São José dos Campos, 1980. (INPE-1903-TDL/031).

[12] VIOLA, F.E.C.; FISCHER, S.D.; SOUZA, C.R. LANAC - uma Linguagem de Alto Nível para Aquisição de dados e Controle de processos. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 8., Florianópolis, 1981. *Anais.* Florianópolis, Sociedade Brasileira de Computação, 1981, p. 619-627.