1. Publicação nº	2. Versão	3. Data	5. Distribuição
INPE-2699-PRE/295		*Abril, 1983*	☐ Interna ☒ Externa ☐ Restrita

4. Origem	Programa
DIN/DPD	*INFORMÁTICA*

6. Palavras chaves - selecionadas pelo(s) autor(es)
LANAC
PROGRAMAÇÃO CONCORRENTE
LINGUAGEM DE PROGRAMAÇÃO

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

8. Título *INPE-2699-PRE/295*

A LINGUAGEM LANAC/II PARA PROGRAMAÇÃO CONCORRENTE E SEU SUPORTE OPERACIONAL

10. Páginas: *9*

11. Última página: *7*

12. Revisada por

Heloísa G.V.S. Borges

9. Autoria *Ladislau José Nave da Silva*
Sérgio Donizetti Fischer
Vânia Aparecida Dinardo Oleinki
Celso de Renna e Souza

Assinatura responsável

13. Autorizada por

Nelson de Jesus Parada
Diretor

14. Resumo/Notas

Descreve-se neste trabalho a linguagem LANAC/II (Linguagem de Alto Nível para Aquisição de Dados e Controle de Processos) do tipo Algol, voltada para sistemas de tempo real, que está sendo desenvolvida no INPE/CNPq, bem como detalhes do cerne (núcleo ou "Kernel") do sistema operacional que a apoia.

15. Observações

Trabalho submetido para apresentação no 1º Congresso Nacional de automação Industrial - CONAI, no Maksoud Plaza Hotel-SP, de 11 a 15/07/83.

ABSTRACT

The Algol-like programming language LANAC/II (Linguagem de Alto Nível para Aquisição de Dados e Controle de Processos) and its Kernel are presented in this work. The language has been developed at INPE/CNPq to be applied in real time systems.

A LINGUAGEM LANAC/II PARA PROGRAMAÇÃO CONCORRENTE E SEU SUPORTE OPERACIONAL

L.J.N. Silva, S.D. Fischer, V.A.D. Oleinki, C.R. Souza

Instituto de Pesquisas Espaciais - INPE
Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq
Caixa Postal 515 - São José dos Campos

RESUMO: Descreve-se neste trabalho a linguagem LANAC/II (Linguagem de Alto Nível para Aquisição de Dados e Controle de Processos) do tipo Algol, voltada para sistemas de tempo real, que está sendo desenvolvida no INPE/CNPq,bem como detalhes do cerne (núcleo ou "Kernel") do sistema operacional que a apoia.

1. INTRODUÇÃO

As atividades de aquisição, processamento e disseminação de imagens e dados enviados por satélites meteorológicos, desenvolvidas pelo Departamento de Meteorologia do INPE/CNPq, vinham sendo programadas através da linguagem "assembly". Como essas tarefas apresentam aspectos de concorrência e de tempo real, a utilização de uma linguagem desse tipo causava dificuldades no entendimento e manutenção de tais programas, além da depuração ser prejudicada pelos erros dependentes do tempo. A partir desse fato e levando em conta a facilidade de desenvolver ferramentas próprias de trabalho, partiu-se para a definição e implementação da linguagem LANAC/I, voltada para a programação concorrente, que oferece aos usuários estruturas para compartilhamento de recursos e mecanismos que possibilitam expressar o paralelismo e a comunicação entre os processos.

A implementação da linguagem LANAC/I (Viola, 1980; Viola et alii, 1981) foi realizada através de um compilador cruzado, escrito na linguagem Algol do sistema Burroughs 6800, gerando código para os minicomputadores HP-2116B e PDP-11/10 da Digital.

Depois de efetuados alguns testes operacionais de utilização da linguagem LANAC/I, sentiu-se a necessidade de enriquecê-la com novos tipos de dados e comandos de controle mais poderosos e partiu-se então para a definição e implementação de uma nova versão da linguagem denominada LANAC/II, na qual se deu mais consistência às estruturas já existentes e introduziram-se novos tipos de dados, tais como variáveis compartilhadas e do tipo real, assim como outros comandos de controle para regiões críticas simples e condicionais, e mecanismos para a atribuição de prioridades a processos.

A implementação desse compilador está sendo feita dentro da filosofia inicial; ele é escrito também na linguagem Algol do sistema Burroughs 6800 e deverá gerar código "assembly" para o minicomputador SISCO MB 8000, para onde o compilador LANAC/II será futuramente transportado. Esta máquina-alvo foi escolhida principalmente por estar sendo usada pelo Departamento de Meteorologia do INPE no desenvolvimento de suas atividades.

Quanto ao suporte de execução para a máquina-alvo, está sendo implementado, em paralelo ao compilador, um suporte independente constituído de um executivo e de um núcleo ("Kernel") e uma versão (Silva, 1982) já desenvolvida e utilizada no sistema PDP-11/10. Nesta versão, o núcleo monitor, embora tenha as mesmas funções da versão anterior, está sendo elaborado de maneira a atender as novas estruturas e mecanismos incorporados à linguagem. O núcleo será responsável pelo escalonamento das tarefas concorrentes, segundo as prioridades destas, e controlará a alocação de recursos da máquina.

Neste artigo são descritos o histórico do projeto LANAC, a linguagem LANAC/II e o suporte de execução para esta nova versão da linguagem.

2. HISTÓRICO DO PROJETO LANAC

2.1. DESCRIÇÃO DA LINGUAGEM LANAC/I

LANAC/I (Linguagem de Alto Nível para Aquisição de Dados e Controle de Processos), versão I (Viola, 1980; Viola et alii, 1981), foi projetada com base na linguagem Algol 60 (Naur, 1963), com algumas adaptações e extensões convenientes às aplicações previstas. Os seus comandos são todos em português e ela não apresenta recursividade.

Quanto aos tipos de dados, esta permite declarar variáveis inteiras e lógicas, identificadores de rótulos, semáforos, listas de caracteres e vetores inteiros unidimensionais. As sub-rotinas com ou sem tipos especificados podem ser declaradas como internas ou externas ao programa, o que permite que que sub-rotinas codificadas em "assembly" possam ser anexadas ao código.

Os comandos podem ser agrupados em bloco ou comando composto. Essas estruturas diferem entre si pelo fato de que um bloco pode ter um conjunto de declarações seguidas por um ou mais comandos, enquanto os comandos compostos não apresentam declarações. Quando os comandos que compõem um bloco ou um comando composto são executados sequencialmente, estes são delimitados por INICIE e TERMINE, caso estes comandos sejam executados concorrentemente, então os delimitadores devem ser PARINICIE e PARTERMINE (PAR para "EM PARALELO").

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

a). comandos de atribuição

- <ident>:=<ident>:= ··· :=<ident>:= <expressão>
- <var> INV <var>
  (realiza a troca dos conteúdos das variáveis envolvidas)
- <var1> REQ <var2>
  (realiza a rotação de 4 bits para a esquerda de <var1>, armazenando o resultado em <var2>),

onde <ident> denota uma variável, um semáforo ou um idenficador de sub-rotina e <var>,<var1>,<var2> denotam um identificador de váriável.

b). comandos de desvio

- VA <identificador de rótulo>
- SE <expressão lógica> ENTAO<comando1> [SENAO<comando2>
- ENCAIXE <expressão aritmética>
  INICIE
  <comando0>;
  :
  <comandoN>;
  TERMINE
  (De acordo com o valor da expressão será executado um comando de 0 a N).

c). comandos iterativos

- ENQUANTO <expressão lógica> EXECUTE <comando>
- REPITA <comando> ATE <expressão lógica>
- PARA <var>:= <expressão-aritmética1> PASSO <expressão-aritmética2> ATE <expressão-aritmética3> EXECUTE <comando>
  (O comando é executado enquanto o valor da variável for menor ou igual à expressão-aritmética3. A cada execução o valor da variável é acrescido da expressão-aritmética2). Deve-se salientar que um comando pode denotar um bloco ou um comando composto. Quanto aos escopos dos identificadores, eles seguem as mesmas regras da linguagem Algol.

d). comandos de entrada/saída e de atitude

Os comandos LEIA e REGISTRE causam, respectivamente, a entrada e saída de dados. Esses comandos permitem que seja especificado, através de parâmetros, se as operações serão feitas com ou sem espera. O comando CONTROLES permite enviar instruções especiais de controle aos dispositivos externos como, por exemplo, rebobinar fita magnética, ligar impressora, etc. A fim de receber informação dos dispositivos externos, a linguagem oferece o comando ESTADOES.

e). outras finalidades da linguagem

As operações sobre os semáforos são realizadas pelas primitivas ESPERE e CAUSE; a primeira atrasa um processo até que ocorra um evento, e a segunda analisa a ocorrência de um evento. Através dessas primitivas, as regiões críticas podem ser implementadas.

2.2. A VERSÃO PARA O HP-2116B

Para dar suporte e permitir a execução de programas em LANAC/1, desenvolveu-se um núcleo monitor ou cerne "Kernel" (Fischer et alii, 1982), responsável pela escalação da execução dos vários processos paralelos e também pelo gerenciamento das atividades de entrada e saída, constituída pelos "drivers" que controlam os periféricos.

O núcleo foi escrito em "assembly" do HP-2116B e sua concepção é bastante simples, uma vez que o objetivo maior desta versão era ter um protótipo do compilador. Neste núcleo, os processos têm a mesma prioridade e podem assumir somente dois estados: ATIVO ou EM EXECUÇÃO. Um processo está ativo quando está aguardando o seu turno de execução numa fila de processos circulares do tipo FIFO ("first-in, first-out"). Um processo em execução é aquele que detém o processa

dor, podendo cedê-lo a outro processo ao esgotar sua fatia de tempo ("time-slice"), ou devido à espera de um sinal da primitiva ESPERE. Em ambos os casos, o processo vai para a fila circular.

Como exemplo da aplicação no HP-2116B, a linguagem LANAC/I foi utilizada na programação da reprodução, em filme fotográfico, de imagens terrestres de alta resolução (AVHRR), enviadas por satélites meteorológicos da série TIROS-N/NOAA. O programa, que usa o modelo clássico do "produtor-consumidor", lê da fita magnética as linhas da imagem adquirida e as envia, através de um conversor digital/analógico, a um imageador, onde o filme fotográfico é sensibilizado.

2.3. A VERSÃO PARA O PDP-11/10

Nesta versão (Silva, 1982), a ênfase maior foi dada ao desenvolvimento do núcleo monitor, que, além dos recursos necessários ao suporte da primeira versão da linguagem LANAC (LANAC/I), inclui outras primitivas e estruturas dirigidas no sentido de atender e oferecer idéias para a segunda versão da linguagem (LANAC/II).

A descrição desta versão do núcleo monitor não será feita aqui, já que, mais adiante, será relatado o suporte de execução da linguagem LANAC/II.

Duas aplicações, que envolvem características de tempo real e concorrência de processos, foram implementadas no PDP-11/10. Uma delas é a aquisição e gravação, em fita magnética, de uma imagem do canal infravermelho enviada pelos satélites meteorológicos da série SMS/GOES; a outra é a leitura, a partir da fita magnética, e reprodução dessas imagens, em filme fotográfico, através de um imageador a "laser". Em ambos os casos, os processos gerenciam as atividades de E/S, e a sincronização entre eles é realizada pelas primitivas dos semáforos.

3. DESCRIÇÃO DA LINGUAGEM LANAC/II

Esta nova versão da linguagem é quase que totalmente compatível com a primeira; com pequenas alterações, um programa escrito em LANAC/I pode ser compilado em LANAC/II. Todas as características da primeira versão foram essencialmente mantidas, até mesmo nas novas estruturas adicionais.

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

3.1. DECLARAÇÕES

Como em LANAC/I, todos os identificadores devem ser declarados. Os identificadores declarados dentro de um bloco podem ser referenciados neste bloco e em todos os blocos internos a estes. Apenas o identificador de rótulo deve ser usado no bloco em que foi declarado.

A linguagem LANAC/II possui variáveis simples do tipo inteiro, lógico e real (não disponível na primeira versão), semáforos, rótulos e listas.

3.1.1. DECLARAÇÃO DE VARIÁVEIS

```
<VARIÁVEIS>::= <TIPO> COMUM <SEQ. DE IDENTI
               FICADORES>|<TIPO><SEQ. DE
               IDENTIFICADORES>
<TIPO>::= INTEIRO | REAL | LOGICO
<SEQ. DE IDENTIFICADORES>::= <IDENTIFICADOR>|
                             <IDENTIFICADOR>,
                             <SEQ. DE IDENTIFI-
                              CADORES>
```

Os identificadores de tipos inteiro, lógico e real que possuam o atributo "COMUM", são considerados variáveis compartilhadas pelos processos.

3.1.2. DECLARAÇÃO DE SEMÁFOROS, RÓTULOS E LISTAS

```
<SEMÁFOROS>::= SEMAFORO <SEQ. DE INDICADORES>
<LISTA>::= LISTA <IDENTIFICADOR><CADEIA DE
           CARACTERES>
<RÓTULO> ::= ROTULO <SEQ. DE IDENTIFICADO
             RES>,
```

onde IDENTIFICADOR denota uma cadeia de caracteres que deve começar por uma letra, e os demais caracteres podem ser dígitos, letras ou o caractere "%", e CADEIA DE CARACTERES denota um cordão de caracteres delimitados por aspas (").

Semáforos são indicados para a sincronização de processos concorrentes e troca de sinais entre eles. Diz-se que um semáforo está aberto se o seu valor for positivo e fechado, se o valor for zero. Esta estrutura está disponível na primeira versão da linguagem, bem como as estruturas de listas e rótulos.

3.1.3. DECLARAÇÃO DE ARRANJOS

```
<VETORES>::= <TIPO> VETOR <SEQ. DE VETORES>|
             <TIPO> VETOR COMUM <SEQ. DE VE
                                 TORES>
<SEQ. DE VETORES> ::= <SEQ. DE IDENTIFICADO
                      RES> <DIMENSÃO>|<SEQ.
                      DE IDENTIFICADORES>
                     <DIMENSÃO><SEQ. DE VE
                      TORES>
<DIMENSÃO>::= [ <INTEIRO DECIMAL>: <INTEIRO
                DECIMAL>]
```

Os arranjos em LANAC/II são unidimensionais e podem ser do tipo inteiro, real e lógico, sendo que estes dois últimos não eram permitidos na primeira versão da linguagem. Como na declaração de variáveis, o atributo "COMUM" denota vetores compartilhados por processos paralelos.

3.1.4. DECLARAÇÃO DE SUB-ROTINAS

```
<PROCEDIMENTO>::= <TIPO SUB> SUBROTINA<IDEN
                  TIFICADOR>;<CORPO SUBRO
                  TINA>|<TIPO SUB> SUBROTI
                  NA <IDENTIFICADOR>; CODI
```

GO|<TIPO SUB> SUBROTINA <IDENTIFICADOR>(<SEQ. DE INDICADORES>); <DECL. PARÂMETROS><CORPO SUB-ROTINA>|<TIPO SUB> SUBROTINA <IDENTIFICADOR>(<SEQ. DE IDENTIFICADORES>); <DECL. PARÂMETROS> CODIGO

<DECL. PARÂMETROS>::= VALOR <SEQ. DE IDENTIFICADORES>; <TIPOS DOS PARÂMETROS>;|<TIPOS DOS PARÂMETROS>;

<TIPO SUB> ::= INTEIRO|LOGICO|REAL|ε

<TIPOS DOS PARÂMETROS> ::= <DECLARAÇÃO DOS TIPOS>|<DECLARAÇÃO DOS TIPOS> <TIPOS DOS PARÂMETROS>

<DECLARAÇÃO DOS TIPOS>::= <VARIÁVEIS>|<VETORES>

Um procedimento é considerado uma função quando um tipo precede a declaração. Como em LANAC/I, os parâmetros podem ser passados por "REFERÊNCIA e por "VALOR".

Na descrição sintática acima, CORPO SUB-ROTINA denota um bloco, e a palavra reservada "CODIGO" indica procedimento externo, cujo código será carregado antes da execução na máquina-alvo.

3.2. COMANDOS ADICIONADOS À LINGUAGEM

A maior dificuldade encontrada no uso da linguagem LANAC/I foi a impossibilidade de garantir uma confiabilidade maior aos programas implementados, devido a ausência de comandos adequados à definição de regiões críticas, onde as variáveis compartilhadas pudessem ser usadas de maneira mutuamente exclusiva.

Na versão anterior, a estrutura de semáforos (Dijkstra, 1968) era a única ferramenta disponível para sincronização de processos. Embora a definição de uma região crítica, através de semáforos, seja bastante simples, ela apresenta inconvenientes, tais como a impossibilidade do compilador verificar se as regiões do prógrama em que os recursos compartilhados são usados estão estabelecidas corretamente, e a impossibilidade do compilador identificar uma região crítica; assim, as variáveis comuns podem ser usadas em qualquer ponto do programa, o que causa a não-prevenção de erros dependentes do tempo.

Portanto, a partir desses fatos, sentiu-se a necessidade de criar estruturas mais adequadas à definição de regiões críticas, que permitem assim a realização de algumas consistências em tempo de compilação. Neste sentido, foram estudadas algumas soluções adotadas por vários autores.

A estrutura introduzida combina as idéias de Hoare (1972), de Hansen (1981) e aquelas presentes na linguagem ILIAD (Schutz, 1979). Dois tipos de regiões críticas podem ser definidas: simples e condicional.

3.2.1. REGIÃO CRÍTICA SIMPLES

<REGIÃO CRÍTICA SIMPLES>::= RESERVE EXECUTE <COMANDO>

Na região crítica simples, um processo, ao tentar entrar na seção crítica, verifica se não existe outro processo executando-a; se não existir, reserva-a, executa o comando e logo após o libera. Caso a região crítica já tenha sido reservada, o processo é bloqueado até que tenha condição de entrar na região.

3.2.2. REGIÃO CRÍTICA CONDICIONAL

<REGIÃO CRÍTICA CONDICIONAL>::= RESERVE <EXPRESSÃO> EXECUTE <COMANDO>

Na região crítica condicional, o processo é bloqueado até que não exista nenhum outro executando a seção crítica. Depois da região estar liberada, a expressão é avaliada; caso esta seja verdadeira, o comando é executado liberando em seguida a região para outros processos e dando prioridade aqueles que estão aguardando a expressão ser verdadeira. Porém, se a expressão for falsa, o processo é suspenso e a região é liberada.

Para uma breve constatação da simplicidade destas estruturas de sincronização, pode-se citar a linguagem Edson, projetada e implementada por Hansen (1981), para programação concorrente e sistemas de tempo real. Esta linguagem permite a criação de regiões críticas através da seguinte estrutura:

WHEN <expressão - booleana> DO <LISTA DE COMANDOS> [ELSE <expressão - booleana> DO <LISTA DE COMANDOS>]*

Este comando é executado em duas fases: a primeira de sincronização, onde o processo é suspenso até que nenhum outro esteja executando a fase crítica. A segunda, denominada fase crítica, onde as expressões são avaliadas sequencialmente, sendo executada a lista de comandos correspondente à primeira expressão booleana com valor verdadeiro. Caso todas as expressões sejam falsas, o processo volta à fase de sincronização, caso contrário, termina a execução do comando.

É importante notar que todo o controle de acesso a variáveis compartilhadas pelos processos fica a cargo do programador, pois não há condição do compilador verificar se os recursos comuns são utilizados unicamente dentro de regiões críticas. Também um fator que deve ser levado em conta é o tempo despendido na avaliação das expressões booleanas; caso elas sejam muito grandes, este tipo de estrutura pode ser inviável para ser utilizado em aplicação em sistemas de tempo real, cujo tempo de resposta é crítico.

3.2.3. COMANDO "ATRASE"

<comando atrase>::= ATRASE (<inteiro decimal>)|
ATRASE (<VARIÁVEL>)|
ATRASE

Permite que um processo seja bloqueado durante o intervalo de tempo especificado. Caso o tempo não seja especificado, o processo é suspenso.

3.2.4. COMANDO "TEMPO"

<comando tempo>::= TEMPO (<VARIÁVEL>)

Este comando retorna à variável especificada um valor inteiro correspondente ao número atual de pulsos dados pelo relógio.

3.2.5. COMANDO PRIORIDADE

<comando prioridade>::= PRIORIDADE (<inteiro decimal>)

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

3.2.6. COMANDO EVENTEX

Na versão I, EVENTEX era definido como uma função aritmética; em LANAC/II, ele passou a ser um procedimento.

<COMANDO EVENTEX>::= EVENTEX (<ARGUMENTO1>, <VARIÁVEL>)|EVENTEX (<ARGUMENTO1>, <VARIÁVEL>,<ARGUMENTO3>)
<ARGUMENTO1>::= <INTEIRO DECIMAL>|
<INTEIRO OCTAL>|
<VARIÁVEL>
<ARGUMENTO 3>::= <INTEIRO DECIMAL>|
<VARIÁVEL>

onde <INTEIRO OCTAL> é UMA CADEIA DE DÍGITOS DE Ø a 7, precedidos pelo símbolo "%".

Este comando pode ser utilizado para verificar e esperar ocorrência de interrupções de dispositivos periféricos.

O primeiro parâmetro especifica o dispositivo periférico, o segundo indica o endereço de retorno do resultado e o terceiro, caso seja especificado, indica o tempo máximo em que o processo ficará esperando pela ocorrência da interrupção.

4. SUPORTE DE EXECUÇÃO DO LANAC/II

Este núcleo é, em parte, uma reprogramação do que foi implementado no PDP-11/10 (Silva, 1982), com o acréscimo de alguns recursos, a destacar a criação de regiões críticas, e a alteração de outros, que foram reconsiderados nesta nova versão da linguagem.

O núcleo está sendo escrito em "assembly" do minicomputador SISCO MB-8000, e é constituído por um conjunto de estruturas de dados e rotinas que operam sobre estas estruturas. Tais rotinas são executadas de forma não-interrompível e podem ser colocadas em três grupos, segundo seu emprego:

- primitivas, que são os procedimentos do cerne chamados diretamente pelos processos;
- atendimento de interrupções;
- rotinas locais, que são chamadas apenas dentro do próprio núcleo pelos demais procedimentos.

Os dois primeiros tipos constituem os inícios de acesso ao núcleo. A responsabilidade do núcleo é a alocação do processador aos processos e não a alocação de recursos.

A definição, de maneira estática, da área de memória ocupada pelos processos é resolvida durante a compilação do programa.

A escalação de processos baseia-se numa política de prioridades atribuídas aos processos. Dentre os processos de igual prioridade, o mais urgente é aquele que aguarda há mais tempo a disponibilidade do processador. Deste modo, ó programador pode influenciar a escalação dos processos.

Há dois modos pelos quais pode-se comandar a multiplexação do processador:

- através de interrupções de periféricos, trocando o processo interrompido por um de maior prioridade;
- através da cessão de fatias de tempo ("time slices") para a execução de cada processo.

Pode-se dizer que sob este aspecto existem dois núcleos: um para os sistemas em que os processos não tendem a ocupar o processador por um período longo, e onde tais períodos estão entre limites conhecidos; e um para os sistemas onde os processos tendem a monopolizar o processador.

Os processos são representados, internamente por uma tabela que contém informações relevantes sobre eles, tais como seu estado, prioridade e condições de retorno à execução. Os estados possíveis são:

- Inativo: o processo ainda não foi iniciado ou já completou sua execução. Inicialmente todos os processos são considerados neste estado.
- Ativo: o processo aguarda o seu turno de execução.

  Existem 4 filas do tipo FIFO ("first-in, first-out") para os processos ativos, uma para cada nível de prioridade.
- Bloqueado: o processo está a espera de alguma condição para continuar e não compete com os outros pelo controle do processador.
- Em execução: o processo detém o processador.

A seguir são descritas as primitivas que implementam os recursos fornecidos pelo núcleo.

Alteração de prioridade

A prioridade de um processo é dinâmica. Ela pode ser modificada pelo próprio processo, o que permite que ele, ao entrar em alguns trechos críticos em termos de execução, eleve sua prioridade e depois, ao sair, volte a uma prioridade menor.

Espera por tempo determinado e consulta ao relógio do sistema

A primitiva ATRASE (tempo) permite o bloqueio de um processo durante o intervalo de tempo especificado; ou, caso este tempo de espera não seja fornecido ou o relógio não esteja ativado no sistema, ela permite que o processo libere o processador em favor de outro, continuando porém no estado ativo. Os processos assim bloqueados são introduzidos numa fila especial ("filatempo"). No instante de tempo em que houver processos a serem reativados, esta fila é pesquisada e dela retiram-se tais processos.

Outra primitiva, TEMPO, provê a consulta ao relógio do sistema para a obtenção de informação sobre o tempo. Se o relógio não estiver ativo, esta primitiva não tem efeito.

Troca de sinais entre processos

As primitivas ESPERE e CAUSE operam sobre variáveis do tipo semáforo (Dijkstra, 1968; Guimarães, 1980; Hansen, 1973), correspondentes às operações P e V, respectivamente, segundo o modelo proposto por Dijkstra (1968). A cada semáforo estão associados um contador e uma fila de espera. Em ESPERE, se o contador for maior que zero, ele é decrementado e o processo continua; caso contrário, o processo é bloqueado e colocado no fim da fila de espera. A primitiva CAUSE reativa o primeiro processo na fila do semáforo; se ela estiver vazia, o contador é incrementado.

Regiões críticas

Em LANAC/II quando um processo executa uma região crítica (condicional ou não), os demais ficam impedidos de ter acesso a quaisquer das regiões críticas declaradas.

Há estruturas de dados especiais para a implementação de regiões críticas: uma porta ("gate") que indica se elas estão livres ou não; uma fila de processos a espera de liberação das regiões críticas; e uma fila de processos a espera de que suas condições de execução da região crítica sejam satisfeitas. Três primitivas operam sobre estas estruturas: entrada da região crítica, saída da região crítica e espera por condição.

Na entrada da região crítica verifica-se o estado da variável porta. Se ela estiver "aberta" o processo continua e a porta é "fechada"; senão, o processo é introduzido na fila de espera pela liberação.

A seguir, dependendo do caso, avaliam-se as condições de execução da região. Se for válida, o processo continua; caso contrário, a primitiva de espera por condição é chamada para colocar o processo na fila correspondente e liberar a região para o primeiro processo a espera de liberação, ou, se ela estiver vazia, "abrir" a porta.

Após a execução da região crítica, a primitiva de saída é invocada para transferir todos os processos da fila de espera por condição para o início da fila de espera por liberação, e reservar a região para o primeiro processo desta fila, ou "abrir" a porta se ela estiver vazia.

Início e término de processos

Em LANAC/II processos paralelos são delimitados por PARINICIE e PARTERMINE, tais processo são chamados "filhos" daquele que os definiu - o "pai". PARINICIE gera uma sequência de chamadas a uma primitiva que inicia um processo, (uma para cada filho). Esta primitiva aloca um descritor ao novo processo, inicializa este descritor e incrementa o número de filhos indicado no descritor do pai.

Após a ativação de todos os filhos, a primitiva "suspende pai" é chamada e bloqueia o processo "pai" até que todos os "filhos" criados tenham terminado.

Após o último comando de um processo, tem-se a primitiva "termine processo" que libera o descritor utilizado, decrementa o número de filhos do processo pai e, dependendo do momento, reativa-o.

Eventos externos e entrada e saída

O núcleo é responsável pelas operações de E/S, aciona os "drivers" dos periféricos e atende às suas interrupções.

Os comandos LEIA, REGISTRE, CONTROLES e ESTADOES correspondem às chamadas dos procedimentos do núcleo que acionam o "driver" do periférico envolvido, e bloqueiam o processo até que a operação termine. LEIA e REGISTRE admitem também, opcionalmente, que o processo continue paralelamente à operação de E/S por ele solicitada, através da especificação do parâmetro SESP ("sem espera"). Neste caso o processo fica responsável pelo reconhecimento do término da E/S: ou pela verificação do estado do periférico (ESTADOES), ou pela espera de ocorrência de evento externo (interrupção) usando o comando EVENTEX.

EVENTEX bloqueia um processo até que uma interrupção do periférico especificado tenha ocorrido, ou que o tempo máximo de espera, opcionalmente definido, tenha se esgotado. A

causa de sua reativação é devolvida ao processo se: o tempo se esgotar; o evento efetivamente ocorrer após a chamada, ou após o evento ter ocorrido previamente. Nesta última hipótese o processo não chegará a ser bloqueado.

Para que se realize o bloqueio e reativação de processos devido a eventos externos, a cada dispositivo de interrupção estão associadas duas variáveis: "processo a espera" e "estado" do evento. Neste último, é indicado se o evento já ocorreu ou não.

Quando um tempo máximo de espera é especificado, o processo é colocado na "filatempo". Se este tempo se esgotar, o processo será reativado do mesmo modo que os bloqueados temporariamente. Se o evento ocorrer antes, o "processo a espera" será retirado da fila e reativado.

5. CONCLUSÕES

A utilização de uma linguagem de alto nível como o LANAC facilita enormemente a programação, depuração, manutenção e entendimento das aplicações de sistemas de tempo-real. Ela permite a construção de programas modulares, onde as tarefas são explicitamente delimitadas e podem comunicar-se entre si através de estruturas de sincronização bem definidas.

Embora a orientação principal da linguagem fosse para sistemas de tempo-real, procurou-se também dotar o núcleo de recursos para desenvolvimento de sistemas de tempo-compartilhado.

Para o núcleo implementado no PDP-11/10 (Silva, 1982) em termos de memória, foram usados aproximadamente 1,5 Kbytes, e os tempos de execução das primitivas, em geral, situam-se na faixa de 200 a 400 microsegundos.

A experiência obtida com as versões anteriores da linguagem, para o HP-2116 e o PDP-11/10, permitiu que a versão II, para o SISCO MB-8000, fosse implementada com maior confiabilidade e menor tempo. O sucesso do projeto demonstra, mais uma vez, que equipes brasileiras estão amplamente capacitadas a projetar e implementar "software" de base e de suporte - e, certamente, de aplicação - para máquinas nacionais ou importadas.

REFERÊNCIAS BIBLIOGRÁFICAS

1. DIJKSTRA, E.W. The Structure of T.H.E. Multiprogramming System. *Communications of the ACM*, 11(5):341-346, May 1968.

2. FISCHER, S.D.; SILVA, L.J.N.; SOUZA, C.R. *Lanac: Uma Experiência no Desenvolvimento de uma Linguagem para Controle de Processos*. São José dos Campos, INPE, 1982. (INPE-2377-PRE/102).

3. GUIMARÃES, C.C. *Princípios de Sistemas Operacionais*. Rio de Janeiro, Campus, 1980.

4. HANSEN, P.B. *Operating Systems principles*. Englewood Cliffs, NJ, Prentice-Hall, 1973.

5. HANSEN, P.B. Edison - A multiprocessor Language. *Software: Practice and Experience*, 11(4):325-361, Oct. 1981.

6. HOARE, C.A.R. Towards a Theory of Parallel Programming. In: *Operating Systems Techniques*. New York, NY, Academic Press, 1972. p. 61-71.

7. NAUR, P. Revised Report on the Algorithmic language Algol 60. *Communications of the ACM*, 6(1):1-23, Jan. 1963.

8. 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.

9. 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, 1982. No prelo.

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

11. VIOLA, F.E.C.; FISCHER, S.D.; SOUZA, C.R. LANAC - Uma Linguagem de Alto Nivel 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.