UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM MÉTODO PARA IMPLEMENTAÇÃO DE PROGRAMAS CONCORRENTES NO COMPUTADOR DEC-10

por

Antonio Tobias Silveira

Dissertação submetida como requisito parcial para
a obtenção do grau de Mestre em
Ciência da Computação

Prof. Simão Sirineo Toscani
Orientador

Porto Alegre, março de 1983

Silveira, Antonio Tobias

Um método para implementação de programas concorrentes no computador DEC-10. Porto Alegre, PGCC da UFRGS, 1983.

Diss. (mestr. ci. comp) UFRGS-PGCC, Porto Alegre, BR-RS, 1983.

Dissertação: Programação concorrente
Multiprogramação
Processos
Sincronização
Intercomunicação

AGRADECIMENTOS

Ao Professor Simão Sirineo Toscani, pela orientação e principalmente pelo incentivo.

Ao Professor Pedro Leon da Rosa Filho pela valiosa colaboração prestada através de discussões, críticas e muito estímulo.

Aos colegas e aos professores do CPGCC.

À Universidade Federal do Pará e particularmente ao Serviço de Estatística e Computação (SECOM), pela oportunidade e pelo apoio financeiro.

À Guacira Nascimento e a todos os amigos que contribuiram de alguma maneira para a realização deste trabalho.

A minha mulher, Lucilêda
e meus filhos,
Luciana e Antonio.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Este trabalho descreve um método para a construção de programas concorrentes no sistema DEC-10, utilizando os conceitos de processos, monitores e classes.

Além do método em si, são apresentados uma linguagem de programação, especificada para facilitar a aplicação do método, e o programa tradutor correspondente. A linguagem é um meio termo entre Pascal Concorrente e MACRO-10. O tradutor gera código na linguagem MACRO-10.

ABSTRACT

This paper presents a method for the concurrent programming of DEC-system 10 using process, monitor and class concepts.

In order to ease its application, a programming language and the corresponding translation program were developed. The language is a composition of concurrent Pascal and MACRO-10. The translator generates code for the MACRO-10 language.

1. INTRODUÇÃO

Este trabalho apresenta um método de programação concorrente baseado nos conceitos de processos, monitores e classes para ser usado no sistema DEC-10/TOPS-10.

O método de programação apresentado utiliza um conjunto de normas, de rotinas e de macro-instruções, que visam principalmente a criação de um suporte, baseado nas facilidades oferecidas pelo sistema operacional TOPS-10, para a implementação, em um projeto futuro, de uma linguagem de alto nível com recursos para programação concorrente.

Para tornar mais cômoda a utilização do método em um prazo mais curto, foi também desenvolvida uma linguagem de programação que é um meio termo entre a linguagem MACRO ASSEMBLER (MACRO-10) do DEC-10 e a linguagem PASCAL CONCORRENTE.

O trabalho está dividido em seis capítulos que apresentam o seguinte conteúdo:

Capítulo 1 - esta introdução.

Capítulo 2 - uma breve apresentação do sistema DEC-10/TOPS-10, onde se descreve o conjunto de instruções da máquina, os registradores disponíveis e as facilidades contidas no sistema operacional TOPS-10, para tratar os problemas da programação concorrente.

Capítulo 3 - descreve a organização dos programas concorrentes segundo o método proposto. Trata-se da organização das áreas de dados de processos, monitores e classes, dos procedimentos que tratam esses objetos e das convenções adotadas.

Capítulo 4 - descreve as primitivas de controle de acesso e de sincronização em monitores e as rotinas que implementam essas primitivas.

Capítulo 5 - descreve as ferramentas criadas para possibilitar a aplicação do método, apresentando a linguagem de programação adotada, as macro-instruções implementadas para dar suporte ao método e, finalmente, a maneira de usar essas ferramentas.

Capítulo 6 - descreve um programa de aplicação que está sendo implementado com base no método de programação proposto. O programa é um sistema de "spooling" para o plotter COMPLOT modelo DP-8V, instalado na Universidade Federal do Pará.

2. CARACTERÍSTICAS DO SISTEMA DEC-10/TOPS-10

Neste capítulo será feita uma breve apresentação do Sistema DEC-10 e de seu sistema operacional TOPS-10. Mais precisamente, serão apresentadas as características de hardware e software que são relevantes para a construção de programas concorrentes segundo o método proposto neste trabalho.

2.1 Características do Hardware

O Sistema DEC-10, visto pelo usuário comum ao nível da linguagem assembler, é uma máquina com memória de 256K palavras de 36 bits, um conjunto de dezesseis registradores de uso geral, também de 36 bits, e um conjunto de periféricos - discos magnéticos, fitas magnéticas, impressoras, leitoras de cartões, terminais remotos, etc - aos quais ele só tem acesso indiretamente através do sistema operacional TOPS-10.

O conjunto de instruções do processador, bastante variado, divide-se nos seguintes grupos:

a) Movimentação de palavras completas - são instruções do tipo "load"/"store" que manipulam palavras de 36 bits, duplas palavras de 72 bits, ou blocos de palavras. A movimentação se dá entre registradores, entre registradores e memória, ou mesmo da memória para a memória.

b) Aritmética de ponto fixo - são operadores aritméticos que operam sobre números de 36 bits (precisão simples) ou sobre números de 72 bits (dupla precisão). Os números, tanto em precisão simples como em dupla precisão, são representados na forma complementar de dois. Nessas operações, um dos operandos deve estar em um registrador (ou em um par de registradores, no caso de dupla precisão) e o ou-

tro pode estar na memória ou em um registrador (ou par de registradores). O resultado das operações pode ser salvo em registradores, na memória ou em ambos.

c) Aritmética de ponto flutuante - neste grupo, como no anterior, existem operadores aritméticos de precisão simples e de dupla precisão. Os operandos de precisão simples são representados em 36 bits, onde o primeiro (bit 0) representa o sinal do número, os oito bits seguintes (bits 1-8) contém um expoente de dois representado em código excesso de 128 e os bits restantes (bits 9-35) contém a representação da parte fracionária do número com o ponto decimal ajustado à esquerda do bit 9. Os operandos de dupla precisão utilizam duas palavras, das quais a segunda é idêntica a que foi descrita para operandos de precisão simples e a primeira, contém nos bits de 1 a 35 uma extensão da parte fracionária do número (o bit 0 da primeira palavra não é utilizado e deve ser zero).

d) Funções lógicas - este grupo de operadores lógicos atua sobre palavras de 36 bits, sendo as operações executadas bit a bit. Também incluem-se nesse grupo algumas outras funções, como por exemplo: atribuição de zeros a todos os bits de um registrador ou palavra da memória, ou atribuição de uns a todos os bits de um registrador ou palavra da memória.

e) Deslocamento e rotação - esses operadores, do tipo "shift"/"rotate", operam apenas sobre registradores. O primeiro operando da instrução indica um registrador e o segundo operando, do tipo imediato, dá o número de bits e o sentido (positivo para esquerda e negativo para a direita) do deslocamento ou da rotação. Encontram-se nesse grupo instruções que operam sobre um ou dois registradores e, ainda, as operações podem ser lógicas ou aritméticas (com extensão do bit de sinal).

f) Teste aritmético - neste grupo encontram-se instruções que realizam "skip" ou "jump" dependendo do resultado de um teste aritmético efetuado sobre operandos de 36 bits. Algumas dessas instruções executam, ainda, operações de incremento ou decremento em um registrador ou palavra da memória, antes de fazer o teste aritmético e o "skip" ou "jump", se for o caso.

g) Teste lógico - as instruções desse grupo fazem "skip" se um teste lógico efetuado resultar verdadeiro. Os testes são realizados nos bits de um registrador, indicados por uma máscara, fornecida como segundo operando. São instruções do tipo: salte se todos os bits mascarados forem zeros, ou salte se pelo menos um dos bits mascarados for um. Algumas dessas instruções podem ainda alterar os bits indicados pela máscara: complementando-os, atribuindo-lhes o valor zero ou o valor um.

h) Movimentação de meias palavras - são instruções semelhantes àquelas descritas no primeiro grupo, do tipo "load"/"store", diferenciando-se apenas porque, neste caso, somente a metade direita ou a metade esquerda de uma palavra é movida de um lugar para outro. A outra metade pode permanecer inalterada, ser preenchida com zeros ou com uns.

i) Controle de programa - neste grupo se enquadram os operadores que modificam a seqüência em que as instruções são executadas em um programa. Apesar de já terem sido descritos grupos de instruções com estas características, esse grupo tem sua especificidade porque as instruções nele constantes tratam os "flags" do programa ("overflow", "carry", etc) e executam chamadas de sub-rotinas, operações que não são feitas pelas instruções dos grupos descritos anteriormente. Fazem parte deste grupo a instrução "EXECUTE", instruções para chamadas de sub-rotinas, que salvam o contador de programa em um registrador ou na primeira palavra da

sub-rotina, instruções do tipo "jump" que dependem dos flags do programa ("overflow", "carry", "traps", direitos de acesso, etc) e uma instrução "halt".

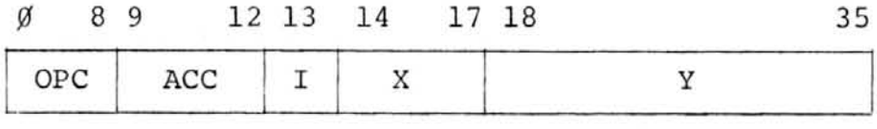
j) Operações sobre uma pilha - existem cinco instruções que possibilitam o uso de pilhas no DEC-10. Nessas instruções, um registrador é designado como apontador de pilha. Com elas se pode fazer as seguintes operações: "push" operando, "pop" operando, desvio para subrotina salvando o contador de programa na pilha, retorno de sub-rotina para um endereço que esteja no topo da pilha e, finalmente, ajuste do registrador que aponta o topo da pilha.

k) Manipulação de bytes - essas instruções possibilitam operações do tipo "load"/"store" de bytes cujo tamanho o programador pode definir (de 1 até 36 bits). Elas facilitam enormemente a manipulação de "strings" de bytes de qualquer tamanho. Conforme pode ser observado, no DEC-10 o conceito de byte é um pouco mais genérico.

l) Operadores estendidos - as instruções deste grupo são especializadas na manipulação de "strings", realizando operações do tipo: mover "string", mover "string" substituindo bytes por outros fornecidos em uma tabela de tradução, conversão decimal, edição de "strings", etc. Essas instruções diferenciam-se bastante das demais porque elas são constituídas de duas palavras: na primeira está o código de operação o qual é sempre o código da operação "EXTENDED" e na segunda palavra, está o código da operação específica que deve ser executada. Os operandos também são bastante diferentes, são apontadores de "strings" e blocos de registradores cujos conteúdos devem ser montados segundo os padrões exigidos por cada operação.

m) Entrada e saída - nesse grupo estão as instruções que permitem realizar as transferências de dados de e para os periféricos, testar ou inicializar os registradores de estado dos periféricos. Essas instruções são instruções privilegiadas que só podem ser executadas pelo sistema operacional, ou por usuários privilegiados executando programas em "real-time".

Excetuando as instruções do grupo de operadores estendidos, as demais ocupam uma palavra de 36 bits e tem o seguinte formato.



O primeiro campo, OPC (bits Ø-8) é o código da operação; o segundo, ACC (bits 9-12) é o campo do primeiro operando, que deve conter um número entre zero e quinze, correspondendo a um dos registradores; o terceiro, I (bit 13) indica endereçamento indireto; o quarto, X (bits 14-17) é usado quando o endereçamento é indexado e neste caso deve conter o número do registrador de índice (qualquer registrador exceto o registrador zero); o último campo, Y (bits 18-35) contém o segundo operando, o seu endereço, ou um valor envolvido no cálculo do operando ou de seu endereço, dependendo, da instrução e do tipo de endereçamento utilizado.

Em instruções que manipulam um operando imediato, esse operando é o próprio endereço efetivo, calculado a partir dos campos I, X e Y da instrução.

O cálculo do endereço efetivo E se processa da seguinte forma:

a) se o conteúdo do campo X for diferente de zero, então E = conteúdo do registrador cujo número está em X + conteúdo de Y; caso contrário, E = conteúdo de Y;

b) se o conteúdo de I for diferente de zero, então, buscar os valores de I, X e Y na posição E da memória e voltar ao passo a); caso contrário o endereço efetivo é aquele já calculado em a).

O que foi apresentado nesta seção dá uma idéia bastante geral das características da máquina em que foi desenvolvido o método de programação concorrente que vai ser apresentado neste trabalho. Na próxima seção será apresentado o sistema operacional, onde ficará esclarecida a forma pela qual o usuário se comunica com ele para efetuar operações de entrada e saída, as quais como foi dito antes, são operações privilegiadas.

2.2 Características do Software

O sistema operacional TOPS-10, é um sistema voltado principalmente para as aplicações de "time-sharing", embora apresente também facilidades para "batch" e "real-time". O usuário se comunica com o sistema operacional, principalmente, através de um amplo repertório de comandos, que digitados ao teclado de um terminal, põe à sua disposição os recursos do sistema. Os recursos disponíveis são aqueles normalmente oferecidos pelos sistemas de grande porte, tais como: um sistema de arquivos em discos magnéticos e fitas magnéticas, "spooling" automático para impressoras, editores de textos, compiladores de diversas linguagens de alto nível, etc. Entretanto, no contexto deste trabalho, o mais importante é mostrar como os programas do usuário se comunicam com o sistema operacional durante a execução e quais são

as facilidades oferecidas pelo TOPS-10 para a implementação de programas concorrentes.

2.2.1 Operadores não implementados

O campo de código de operação em uma instrução do DEC-10 é um campo de nove bits que admite, portanto, códigos com valores desde 0 até 512. Entretanto, nem todos esses códigos são utilizados pelas instruções válidas. O código Ø por exemplo é um código inválido; quando encontrado gera um trap que é interceptado pelo sistema operacional provocando o encerramento do programa com a mensagem: "ILLEGAL UUO at USER PC XXXXXX".

Um processo semelhante ao que foi descrito acima é usado para possibilitar a comunicação de um programa de usuário com o sistema operacional TOPS-10. São as chamadas "UUO's" - "Unimplemented User Operators".

UUO's são instruções com códigos não implementados no hardware e que por isso geram traps interceptados pelo sistema operacional. Assim, o sistema operacional verifica se o código corresponde a alguma das operações implementadas por software transferindo para a rotina correspondente, se for o caso.

Existem dois tipos de UUO's no DEC-10: são as MUUO's e as LUUO's. As MUUO's ou Monitor UUO's são aquelas cujos códigos estão na faixa de 40 octal até 77 octal e que correspondem às funções implementadas pelo próprio TOPS-10 de forma indivisível, como se fossem uma única instrução. As LUUO's ou Local UUO's, são aquelas cujos códigos estão na faixa de 1 octal até 37 octal e que o sistema operacional reconhece como funções implementadas pelo programa do usuário. As LUUO's, ao contrário das MUUO's, não são execu

tadas de forma indivisível.

Assim, é através das MUUO's, também chamadas "monitor calls", que os programas do usuário tem acesso às facilidades de entrada e saída e às facilidades para programação concorrente implementadas no TOPS-10, além de outras, que não serão objeto de análise neste texto.

2.2.2 Facilidades para Programação Concorrente Implementadas no TOPS-10

Nesta seção será abordada a forma específica de solução utilizada no TOPS-10 para os problemas da programação concorrente: criação de processos, sincronização e comunicação entre processos, compartilhamento de "procedures" e compartilhamento de dados.

2.2.2.1 Os processos no TOPS-10

No DEC-10, os programas são vistos pelo sistema operacional TOPS-10, através de uma entidade chamada "job". "Jobs" são como envelopes, a cada programa em execução no sistema em um dado momento, há um "job" associado, identificado, de forma única, por um número no intervalo de 1 a "JOBMAX". "Jobs" não são criados nem destruídos dinamicamente durante a operação do TOPS-10. Eles existem em um número fixo ("JOBMAX"), estabelecido na geração do sistema operacional e podem estar ativos (alocados para usuários) ou inativos (disponíveis para o primeiro usuário que solicitar um "job").

Sendo o DEC-10 orientado para o uso na base de tempo compartilhado, onde normalmente o usuário controla a execução de seus programas através de um terminal, a aloca

ção de "jobs" é feita para atender sessões de trabalho interativo. Ou seja, no momento em que o usuário vai iniciar seu trabalho, a primeira operação que ele deve executar é aquela que lhe vai garantir o acesso ao sistema e um "job", que ficará permanentemente alocado para ele, até o encerramento de sua sessão de trabalho. Essa operação, denominada "log in", é executada digitando-se o comando "LOGIN" em um terminal. Esse comando dispara a execução de um programa do sistema, que verifica os direitos de acesso do usuário, que se identifica por um par de números octais (p,pn = "project, programer number") e uma senha ("password"). A operação termina com a alocação de um "job" ao terminal, que a partir daí estará pronto para aceitar outros comandos do usuário.

Como foi mencionado acima, existe um vínculo entre um "job" (processo envelope), um usuário (p,pn) e o terminal (TTY) onde foi executado o comando "LOGIN", que determinou a alocação do "job". Em algumas circunstâncias é interessante quebrar parte desse vínculo, liberando o terminal sem parar a execução do programa controlado pelo "job". Isto pode ser feito através do comando "DETACH" digitado no terminal, ou pelo próprio programa, através de uma chamada do sistema operacional ("monitor call" ATTACH).

Os "jobs" ativos, executando sem vínculo a um terminal, são ditos "jobs" em "det" ("detached"). Se o usuário desejar, poderá conectar novamente o "job" em "det" a um terminal usando o comando "ATTACH". O próprio programa, dependendo dos privilégios do usuário (p,pn) proprietário, poderá conectar-se a um terminal fazendo uma chamada do sistema operacional ("monitor call" ATTACH).

Não há, no TOPS-10, nenhuma facilidade ("monitor call") explicitamente voltada para permitir que um processo possa iniciar a execução de outros processos. Existe, entretanto, um dispositivo de software, denominado pseudo termi

nal (PTY), no qual um programa pode executar operações de escrita e leitura e que é visto pelo sistema operacional como se fosse um terminal físico. Ou seja, um programa pode funcionar como o operador de um pseudo terminal, que é visto pelo TOPS-10, como se fosse um TTY qualquer. Esse tipo de procedimento é que permite, sob o TOPS-10, que se escreva programas, que disparem e controlem a execução de outros processos, escrevendo em um PTY os comandos necessários e interpretando as respostas que chegam a ele, oriundas do sistema operacional ou do processo controlado. Do ponto de vista do processo controlado é indiferente, se o terminal controlador é um terminal físico com operador humano, ou se é um pseudo terminal operado por um programa.

2.2.2.2 Comunicação entre Processos no TOPS-10

Existe no TOPS-10 um sub-sistema denominado IPCF ("Inter Process Communication Facility"), que é acionado através de chamadas do sistema operacional e que possibilita a comunicação entre processos através de trocas de mensagens.

IPCF é um sistema que permite a um processo: enviar mensagens para um outro processo, receber mensagens a ele destinadas por outros processos, verificar se existem mensagens a ele destinadas, criar uma ou mais caixas postais para recepção de mensagens, associando a elas nomes simbólicos, solicitar e receber como resposta o número da caixa postal associada a um dado nome simbólico, etc.

Os pedidos de serviço de IPCF ao sistema operacional são feitos através de três "monitor calls":

IPCFS. - envia uma mensagem a um outro processo. Opera sobre um bloco de argumentos que deve conter a identificação do processo transmissor, a identificação do processo destinatário, o endereço do bloco de palavras que contém a mensagem a ser transmitida e o número de palavras que compõe a mensagem.

IPCFQ. - retorna ao processo que solicitou o serviço, o número de mensagens destinadas à ele a espera para serem recebidas e um bloco de descrição da primeira mensagem da fila. O bloco de descrição contém basicamente a identificação do processo que transmitiu a mensagem, a identificação do processo que deve recebe-la, o camanho da mensagem e a descrição de algumas propriedades do processo transmissor.

IPCFR. - recebe uma mensagem que esteja na fila aguardando recepção. Opera sobre um bloco de argumentos que deve conter a identificação do processo receptor, o endereço e o tamanho da área onde deverá ser depositada a mensagem recebida.

A identificação dos processos para o sistema IPCF pode ser feita de duas maneiras: a primeira é usando o próprio número do "job" e a outra é usando um código próprio denominado "PID" ("Process Identification"). A utilização do número do "job" para identificar um processo não é muito conveniente porque esse número varia a cada execução do processo. A utilização de "PID's" é mais interessante, uma vez que esses códigos são atribuídos pelo próprio controlador de IPCF, sob solicitação do processo e, ainda, porque a esses códigos, por solicitação do processo, pode ser associado um nome simbólico. Desta forma, um processo que necessite determinar o código de um outro, pode solicitar ao controlador de IPCF o "PID" correspondente ao nome simbólico do outro processo. Esses serviços, do tipo identificação de

processos, não são feitos pelo TOPS-10, eles são executados por um processo chamado "SYSTEM INFORMATION" que se mantém em "det" e que pode ser acionado através de mensagens de IPCF.

As mensagens que podem ser trocadas através do sistema de IPCF são blocos de palavras consecutivas e seu conteúdo pode ser qualquer. Distingue-se, entretanto, pelo tamanho dois tipos de mensagens: mensagens curtas com tamanho entre três e dez palavras e mensagens longas com 512 palavras (uma página). No primeiro caso, as mensagens transmitidas são copiadas para uma área reservada no sistema operacional e posteriormente, quando recebidas, para a área especificada pelo processo receptor. No segundo caso, a mensagem deverá estar contida em uma página de memória que o processo transmissor precisa solicitar ao sistema operacional através da "monitor call PAGE". O processo receptor, para receber uma mensagem longa, deve especificar o endereço de uma página de seu espaço de endereçamento, que não esteja definida; ou seja, que esteja inacessível. A página transmitida será então incorporada ao espaço de endereçamento do processo receptor, efetivando a comunicação.

As mensagens destinadas a um processo são enfileiradas segundo a ordem cronológica de suas ocorrências, até o recebimento. O recebimento das mensagens poderá ser feito na mesma ordem, ou então, poderá ser feito de modo mais seletivo, se o processo receptor tiver definido mais de um "PID" e solicitar a recepção de mensagens de um desses códigos especificamente.

O sistema de comunicação de processos via sub-sistema de IPCF pode também ser utilizado para sincronização de processos, uma vez que o processo receptor pode especificar, em uma chamada de IPCFR., que deve ser bloqueado se não houver mensagens para ele receber. Além disso, a ocor-

rência de mensagens para um processo, pode ser especificada em outras "monitor calls" como causa para que um processo bloqueado por uma dessas chamadas seja desbloqueado. Ou ainda, que a ocorrência de mensagens para um processo, seja a causa para uma interrupção no processo receptor, no caso de estar sendo utilizado o sistema de interrupções de software, o qual pode ser ativado através de outras "monitor calls".

2.2.2.3 Sincronização de Processos no TOPS-10

Além do que foi mencionado anteriormente sobre sincronização baseada exclusivamente na comunicação via IPCF, existem no TOPS-10 outras primitivas que permitem implementar os vários tipos de sincronização comumente requeridos em programação concorrente. Essas primitivas são acessíveis através das "monitor calls": SLEEP, HIBER, WAKE e ENQ./DEC..

A "monitor call" SLEEP, possibilita a um processo, solicitar ao sistema operacional, que ele seja bloqueado por um período de tempo especificado em segundos, até um máximo de 60 segundos.

A "monitor call" HIBER, possibilita a um processo, solicitar ao sistema operacional, que ele seja bloqueado até a ocorrência de pelo menos um dos eventos especificados na chamada. Os eventos que podem ser especificados em uma chamada de HIBER são os seguintes:

a) um intervalo de tempo até um máximo de 68 segundos, especificado em milisegundos (o bloqueio real acontecerá por um número inteiro de jifs, 1/60 de segundo);

b) a chegada de uma mensagem de IPCF;

c) a finalização de uma operação de entrada/saída assíncrona;

d) uma mudança no estado de um pseudo terminal (PTY) controlado pelo processo;

e) uma linha de dados de entrada pronta para ser lida do terminal controlador do processo;

f) um carater de dados de entrada pronto para ser lido do terminal controlador do processo;

g) um outro processo executando em uma conta com mesmo número de projeto (p,pn), executa uma chamada de "WAKE" referente a ele;

h) um outro processo, executando em uma conta com mesmo número de programador (p,pn), executa uma chamada de "WAKE" referente a ele.

Além disso pode ser especificado que o processo deva ser bloqueado a não ser que ele tenha executado previamente um "WAKE" para ele próprio.

Note-se que a especificação das condições listadas acima, em uma chamada de HIBER, funciona como proteção. Ou seja, significa que o processo só deverá ser desbloqueado se pelo menos um dos eventos especificados efetivamente ocorrer. Isto garante, por exemplo, que um processo que se auto-bloqueou, aguardando ser despertado por uma chamada de "WAKE" de outro processo executando na mesma "p,pn", não seja desbloqueado por um outro processo que não atenda a essa condição e que por algum motivo tenha executado "WAKE" referindo-se a ele.

A "monitor call" WAKE, possibilita a um processo, solicitar ao sistema operacional, para sinalizar um outro processo de modo que ele seja desbloqueado. A referência ao processo que deve sofrer a ação de "WAKE" é feita pelo núme

ro do "job" correspondente.

As "monitor calls" ENQ. e DEQ. possibilitam a sincronização de processos no que diz respeito ao acesso a seções críticas e ao controle de recursos compartilhados por mais de um processo. O método consiste em associar nomes ou códigos aos recursos ou às seções que devem ser controladas e passar esses nomes ou códigos ao TOPS-10 que irá tratá-los de forma similar a semáforos. Essas primitivas de sincronização estão implementadas de forma bastante abrangente visando principalmente os problemas de acesso e atualização simultâneas em estruturas de arquivos. Neste texto, entretanto, elas serão tratadas apenas em uma forma limitada, mas que atende completamente a solução dos problemas específicos encontrados durante o desenvolvimento deste trabalho.

A "monitor call" ENQ. é usada para solicitar acesso a um recurso ou seção crítica definida por um código (user code), de modo exclusivo. As solicitações de ENQ. para um mesmo recurso são introduzidas pelo TOPS-10 em uma fila segundo a ordem cronológica dessas solicitações. Os processos são atendidos um por vez e esse atendimento se dá quando não há nenhum processo utilizando o recurso ou quando o recurso é liberado pelo processo que detinha a sua posse. Os processos não atendidos permanecem bloqueados até que seja possível atendê-los.

A "monitor call" DEQ. é usada para liberar um recurso anteriormente adquirido através de ENQ., tornando-o novamente disponível. Neste ponto é interessante ressaltar, a diferença existente entre estas primitivas e as operações P e V sobre variáveis semáforas. Ao contrário do que ocorre com a operação V, uma chamada de DEQ. só pode ser feita por um processo que antes tenha executado ENQ. para o mesmo recurso.

É interessante notar, que do ponto de vista do sistema operacional, os recursos controlados através de ENQ./DEQ. são recursos virtuais. Para o TOPS-10 existem apenas os códigos ou nomes definidos nas chamadas de ENQ./DEQ. e que devem ser respeitados pelos processos que compartilham esses recursos, com base em um protocolo estabelecido na programação do sistema concorrente. Apenas no caso específico de tratamento de arquivos o sistema operacional chega a controlar o próprio recurso e não somente o seu nome ou código.

2.2.2.4 Compartilhamento de Procedimentos e Dados no TOPS-10

O espaço de endereçamento virtual de um processo no DEC-10, compõe-se de dois segmentos chamados "low segment" e "high segment" respectivamente.

O "low segment" deve existir obrigatoriamente em qualquer processo. Iniciando no endereço virtual zero, pode estender-se até o limite de 256K palavras. Os endereços iniciais, de zero a quinze, normalmente não são mapeados na memória principal, mas sim nos dezesseis registradores gerais da máquina.

O "high segment" pode existir, mas não obrigatoriamente. Deve iniciar em um endereço virtual maior que o maior endereço virtual do "low segment" e esse endereço deve ser um múltiplo de 1024.

A soma dos tamanhos dos dois segmentos não pode exceder 256K palavras que é o tamanho máximo do espaço de endereçamento virtual de um processo sob o TOPS-10.

O que diferencia e de certa forma especializa esses segmentos é que o "high segment", existindo, é potencialmente compartilhável por um grupo de processos. O TOPS-10, entretanto, impõe uma proteção do tipo "read/execute only" para esses segmentos de modo, que em princípio, só é possível o compartilhamento de procedimentos e de constantes entre os processos que compartilham um "high segment".

O compartilhamento de dados de uma forma geral, e não apenas de constantes, aparentemente prejudicado, torna-se possível, uma vez que o sistema operacional deixa uma abertura para que um processo modifique a proteção do seu "high segment", se ele tiver privilégio para isso. Esse privilégio é garantido a um processo, se ele estiver sendo executado em uma conta (p,pn), que lhe dê direito de escrita no arquivo de disco a partir do qual se originou o "high segment" compartilhado.

Um outro dado relevante em relação a utilização de um "high segment" compartilhável é que se existe um grupo de processos compartilhando um "high segment", e algum deles, ou qualquer outro processo, executa uma operação que modifique o arquivo que deu origem ao "high segment", a partir daí, só estarão compartilhando esse segmento que já estava na memória, aqueles processos que haviam adquirido o segmento, antes da modificação no arquivo. Qualquer processo que venha solicitar acesso a esse "high segment" posteriormente, irá utilizar um outro "high segment" construído a partir do disco.

Relacionadas à manipulação de "high segments", tendo em vista o compartilhamento de procedimentos e o compartilhamento de dados entre os processos de um programa concorrente, existem no TOPS-10 duas "monitor calls":

GETSEG - possibilita a substituição ou a inclusão de um "high segment" no espaço de endereçamento do processo que a executa. O argumento para essa "monitor call" é um bloco de dados contendo a especificação do arquivo em disco que contém o "high segment". Se o arquivo especificado como argumento tiver sido construído com o comando "SSAVE" (Shareable SAVE), então o "high segment" adquirido pelo processo poderá estar sendo compartilhado por outros processos.

SETUWP - possibilita a mudança do código de proteção do "high segment" de um processo. Como argumento deve ser fornecido o valor zero ("read/write/execute"), ou o valor um ("read/execute"). O TOPS-10 retorna ao processo solicitante o código de proteção que estava em vigor antes da chamada, após ter alterado esse código, ou interrompe a execução do processo se ele não tiver direito de efetuar a operação.

3. ORGANIZAÇÃO DE UM PROGRAMA CONCORRENTE

Os programas concorrentes construídos segundo o método proposto neste trabalho constituem-se de processos, monitores e classes. Um processo inicial tem a função de inicializar e controlar a execução dos demais processos do programa concorrente. Todos os processos são programas independentes (cada um sendo executado sob um "job") que se comunicam e são sincronizados através de "monitores".

3.1 O Espaço de Endereçamento dos Processos

Cada processo em um programa concorrente, inclusive o processo inicial, endereça um espaço virtual dividido em um "low segment" privado e um "high segment" compartilhado com os demais processos.

No "low segment" de um processo estão distribuídos os dados privados do processo e os segmentos de código particulares desse processo. Ele se estende desde o endereço zero até um limite determinado pela soma dos tamanhos das áreas de dados privados com o tamanho do código do programa que o processo executa.

No "high segment" compartilhado estão distribuídas as áreas de dados permanentes dos monitores, os segmentos de código das rotinas dos monitores, os segmentos de código das rotinas compartilhadas de classes que sejam usadas por mais de um processo, ou que sejam usadas por monitores, tabelas de controle do programa concorrente e as rotinas que implementam as primitivas de controle de acesso e sincronização utilizadas em monitores.

O "high segment" compartilhado inicia no endereço virtual 400000 (octal) e pode se estender até o endereço 777777 (octal). Essa divisão, até certo ponto arbitrária, do espaço de endereçamento dos processos, em dois segmentos de 128K palavras, significa apenas a aceitação de uma regra imposta pela prática, uma vez que a maioria dos programas do sistema usam essa divisão.

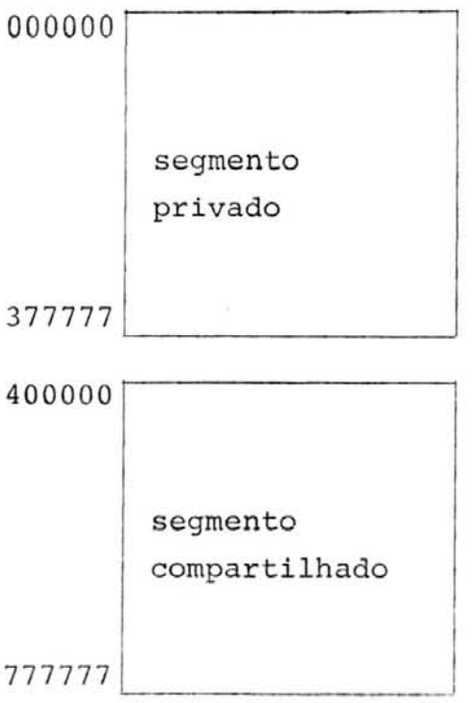


Fig. 3.1 - Espaço de endereçamento de um processo (limites máximos).

3.2 O Processo Inicial

Os programas concorrentes de que trata este texto, possuem um processo inicial que tem a função de inicializar o programa, disparando a execução dos demais processos. Após a inicialização este processo mantém-se em um ciclo, servindo de intermediário na comunicação dos demais processos com o operador do programa.

A execução de um programa concorrente é iniciada executando-se o processo inicial em um terminal, ou mesmo através do sistema de batch, como se fosse um programa qualquer. Ou seja, não há procedimentos especiais que diferenciem, do ponto de vista do usuário que opera o programa, a inicialização de um programa concorrente, da inicialização de um programa ordinário.

Os procedimentos de inicialização executados pelo processo inicial constituem, na maior parte, uma seção crítica, protegida por uma chamada de ENQ. e liberada por uma chamada de DEQ.. Estes procedimentos consistem em:

a) verificar se o usuário (pp,n) tem privilégios suficientes para executar programas desse tipo. Isto é necessário porque, programas implementados segundo este método, utilizam as "monitor call's" ENQ. e DEQ. e para fazê-lo, é necessário estar devidamente autorizado pelo administrador do sistema. Essa verificação é feita examinando uma tabela do sistema operacional através da "monitor call" GETTAB. Se o usuário não estiver habilitado, a execução do programa é interrompida por uma chamada da "monitor call" EXIT, após a emissão de uma mensagem de erro.

b) preparar o "high segment", garantindo que ele será compartilhado apenas pelos processos deste programa. Isto é feito para evitar interferência entre duas ocorrências de um mesmo programa concorrente executadas simultaneamente. Este procedimento é executado em três etapas: na primeira, fazendo uma chamada de ENQ., aplicada sobre um recurso virtual cujo nome é o nome do arquivo que contém o "high segment", o processo inicial garante que apenas uma ocorrência do programa irá continuar executando os passos seguintes que fazem parte da seção crítica; na segunda, o processo inicial adquire o "high segment" que será compartilhado pelos demais processos, através de uma chamada de

GETSEG aplicada sobre um bloco de argumentos que contém a especificação do arquivo que contém o "high segment"; na última, fazendo uma chamada de SETUWP, o processo inicial desprotege o seu segmento alto habilitando-se a escrever nele.

c) inicializar os monitores usados pelo programa. Consiste em depositar nas áreas de parâmetros dos monitores, os endereços de outros monitores acessíveis a cada monitor e, em seguida, executar as rotinas correspondentes aos blocos de inicialização de cada monitor.

d) inicializar os processos componentes do programa. Este procedimento é feito por uma rotina que manipula um pseudo terminal (PTY) onde são escritos os comandos necessários para disparar a execução de cada processo, cada qual em um "job" diferente. Essa rotina será abordada, posteriormente, de modo mais detalhado, em uma seção específica.

e) completar a proteção do "high segment" contra interferência de outras ocorrências do mesmo programa e liberar o acesso à seção crítica. Consiste em executar uma operação de escrita no arquivo que originou o "high segment", sem modificá-lo, entretanto, e em seguida, fazer DEQ. sobre o recurso definido com o nome do arquivo. O efeito de escrita no arquivo sem modificá-lo, é conseguido com a "monitor call" RENAME aplicada sobre um bloco de argumentos contendo a especificação do arquivo que originou o "high segment". Como o argumento de RENAME, especifica o nome original do arquivo, sem modificar nenhum de seus atributos, o arquivo permanece inalterado após a sua execução, mas o efeito é o desejado: daí por diante, o primeiro processo que executar GETSEG, especificando o mesmo arquivo, irá provocar a construção de um novo "high segment" a partir do disco e não compartilhar o segmento já existente na memória.

f) verificar se todos os processos disparados encerraram seus procedimentos de auto-inicialização. Esta verificação é feita através de uma entrada em um monitor de controle, no qual, cada processo que completa a sua própria inicialização, deposita um sinal para avisar ao processo inicial. O processo inicial permanece bloqueado, enquanto o número de sinais for menor que o número de processos disparados.

Encerrada a fase de inicialização, o processo inicial se mantém em um ciclo no qual ele consulta um monitor utilizado pelos demais processos, para enviar e receber mensagens do operador do programa. Cada vez que o processo inicial vai a esse monitor e encontra uma mensagem, ele a exibe na tela do terminal, precedida da identificação do processo que a enviou. Se a mensagem inicia por um ponto de interrogação, ele solicita uma resposta do operador do programa e a envia em seguida, através do mesmo monitor, no qual o processo que enviou a mensagem deverá estar esperando pela resposta. Ao final de cada ciclo, o processo inicial consulta um outro monitor, através do qual os demais processos comunicam o seu encerramento, se todos os processos tiverem terminado, o processo inicial interrompe o ciclo e termina também. Antes de reiniciar o ciclo, o processo inicial libera o processador através de uma chamada da função SLEEP, permanecendo bloqueado durante 1 segundo.

3.2.1 A Inicializacão dos Processos

A inicialização dos processos componentes de um programa concorrente, executada pelo processo inicial, baseia-se na utilização de um pseudo terminal (PTY) e nas estruturas de dados mostradas na figura 3.2.

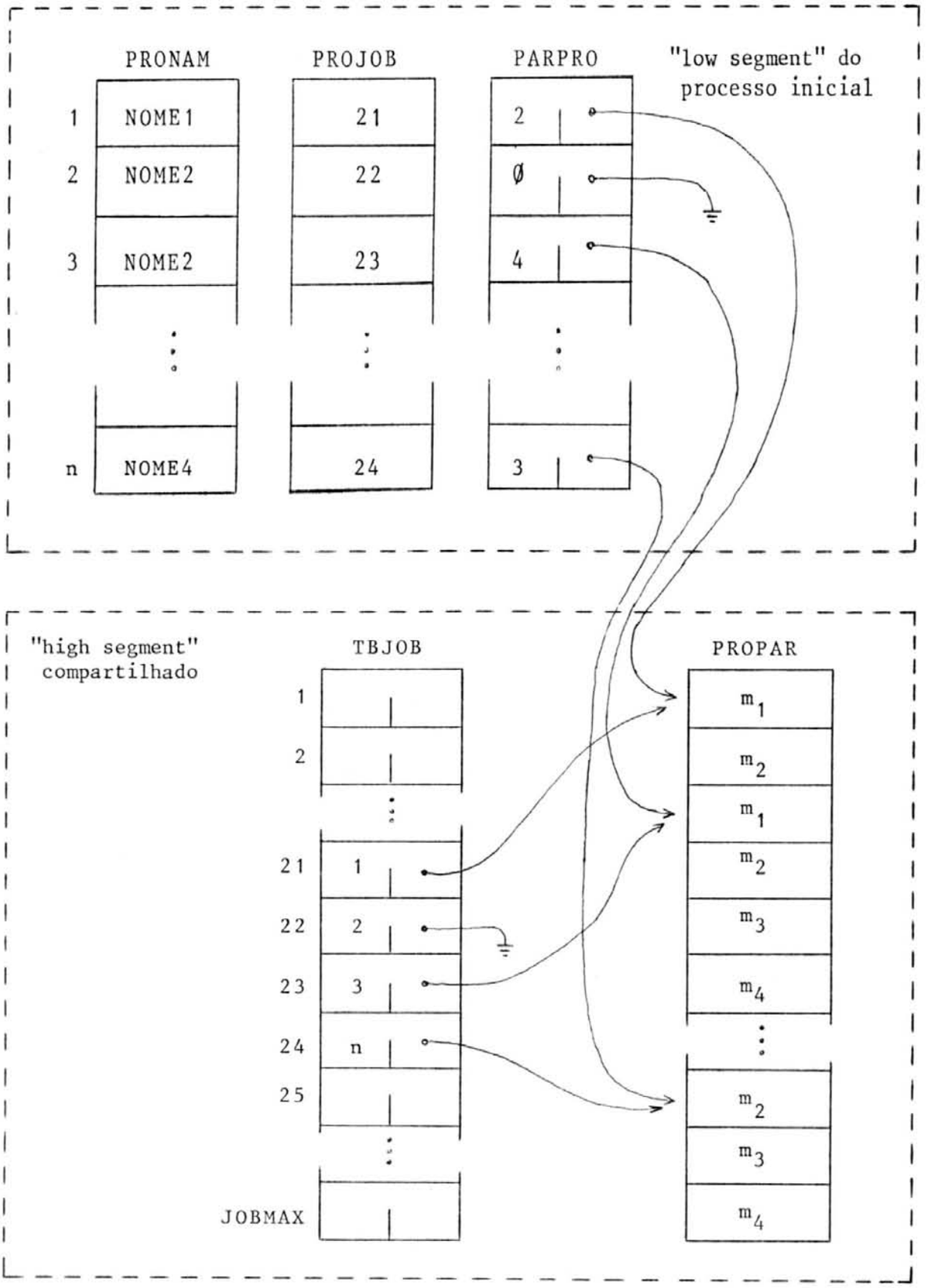


Figura 3.2 - Estrutura de dados usada no controle de disparo dos processos

A tabela PRONAM, montada no "low segment" do processo inicial, contém os nomes dos programas correspondentes a cada processo.

A tabela PROPAR, montada no "high segment" compartilhado, contém as listas de parâmetros de cada processo. Ou seja, a cada processo corresponde uma lista de palavras contendo os endereços dos monitores acessíveis ao mesmo. As listas de parâmetros estão dispostas na mesma ordem dos processos na tabela PRONAM.

A tabela PARPRO, montada no "low segment" do processo inicial, contém o mesmo número de entradas da tabela PRONAM, obedecendo a mesma ordem de processos. Contém em cada entrada um ponteiro para o início da lista de parâmetros correspondentes em PROPAR.

A tabela PROJOB, montada no "low segment" do processo inicial, dispõe de uma entrada para cada processo. Cada vez que um processo da tabela PRONAM é disparado, o número do "job" é salvo pelo processo inicial na entrada de PROJOB correspondente ao processo.

A tabela TBJOB, montada no "high segment" compartilhado, contém JOBMAX entradas, onde JOBMAX é o maior número de "jobs" definido na geração do sistema operacional. Cada vez que um processo da tabela PRONAM é disparado, o número do "job" é usado pelo processo inicial para indexar a tabela TBJOB. Na posição correspondente ao "job" são salvos o número de ordem do processo e uma cópia do ponteiro correspondente obtido na tabela PARPRO, nas metades esquerda e direita da palavra, respectivamente.

O algoritmo da rotina de inicialização dos processos é o seguinte:

a. faz P igual a 1;

b. chama a rotina "LOGIN" que usa um pseudo terminal para executar uma operação de "log in" e retorna o número do "job" obtido na variável JOB;

c. faz PROJOB(P) igual a JOB;

d. faz, a metade esquerda de TBJOB(JOB) igual a P;

e. faz, a metade direita de TBJOB(JOB) igual a PARPRO(P);

f. chama a rotina "RUN", que usa o mesmo pseudo terminal usado por "LOGIN", para executar o comando "RUN" no programa cujo nome está em PRONAM(P);

g. incrementa de um valor de P;

h. se P for menor ou igual a NOPRO (o número de processos a serem inicializados), então, volta ao passo b; caso contrário, termina.

Nessa rotina, um único pseudo terminal é usado para disparar todos os processos. Isto é possível porque, cada processo disparado, executa imediatamente a "monitor call" ATTACH, quebrando o vínculo com o seu terminal controlador, que no caso é o pseudo terminal usado pelas rotinas "LOGIN" e "RUN" mencionadas no algoritmo.

3.3 Organização das Áreas de Dados nos Programas Concorrentes

Os componentes dos programas concorrentes, processos, monitores e classes, possuem áreas de dados permanentes que são organizados segundo o esquema mostrado na figura 3.3. As rotinas dos tipos "procedure" ou "function" definidas nesses componentes, possuem áreas de dados temporá-

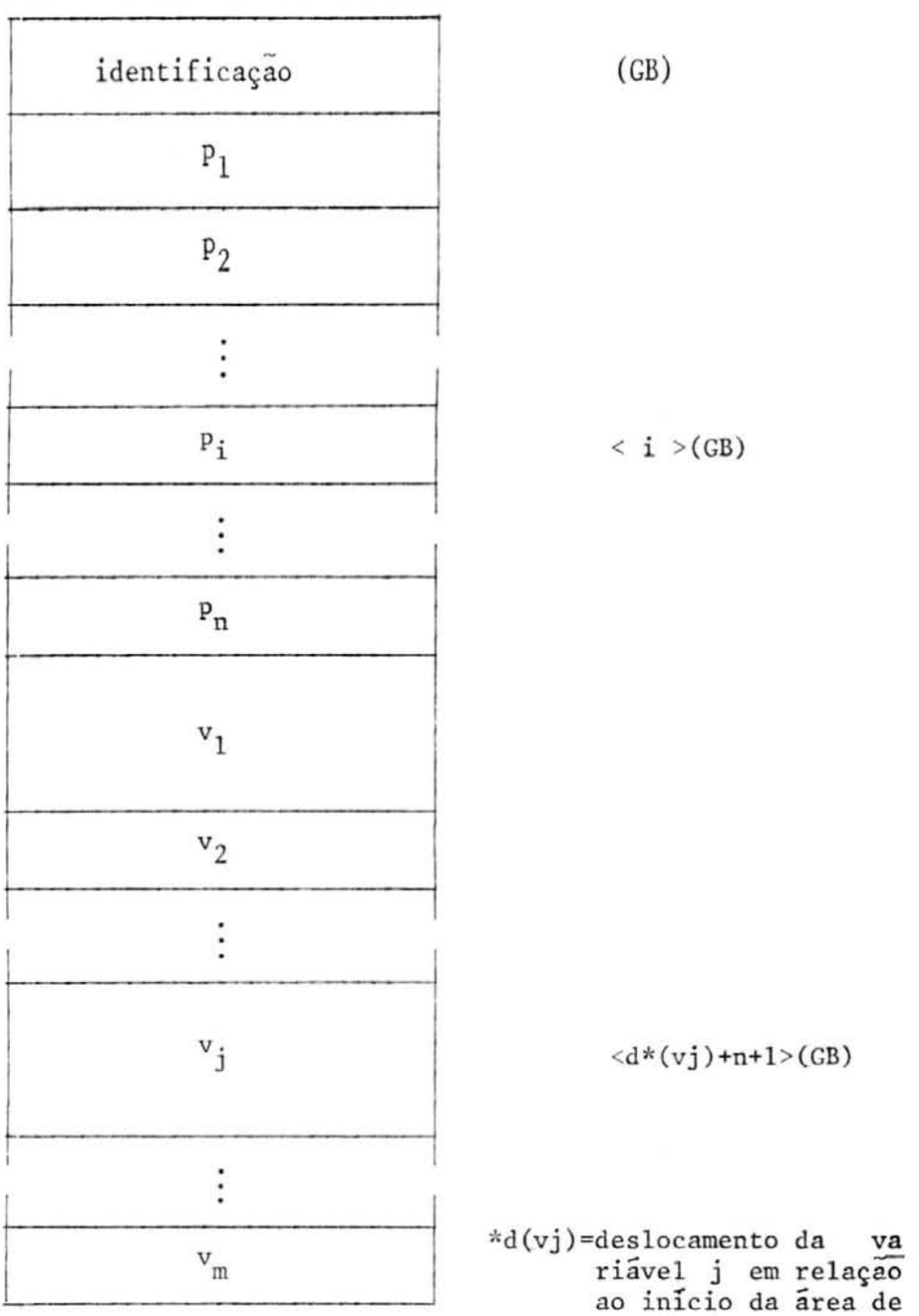


Figura 3.3 - Organização das áreas de dados permanentes.

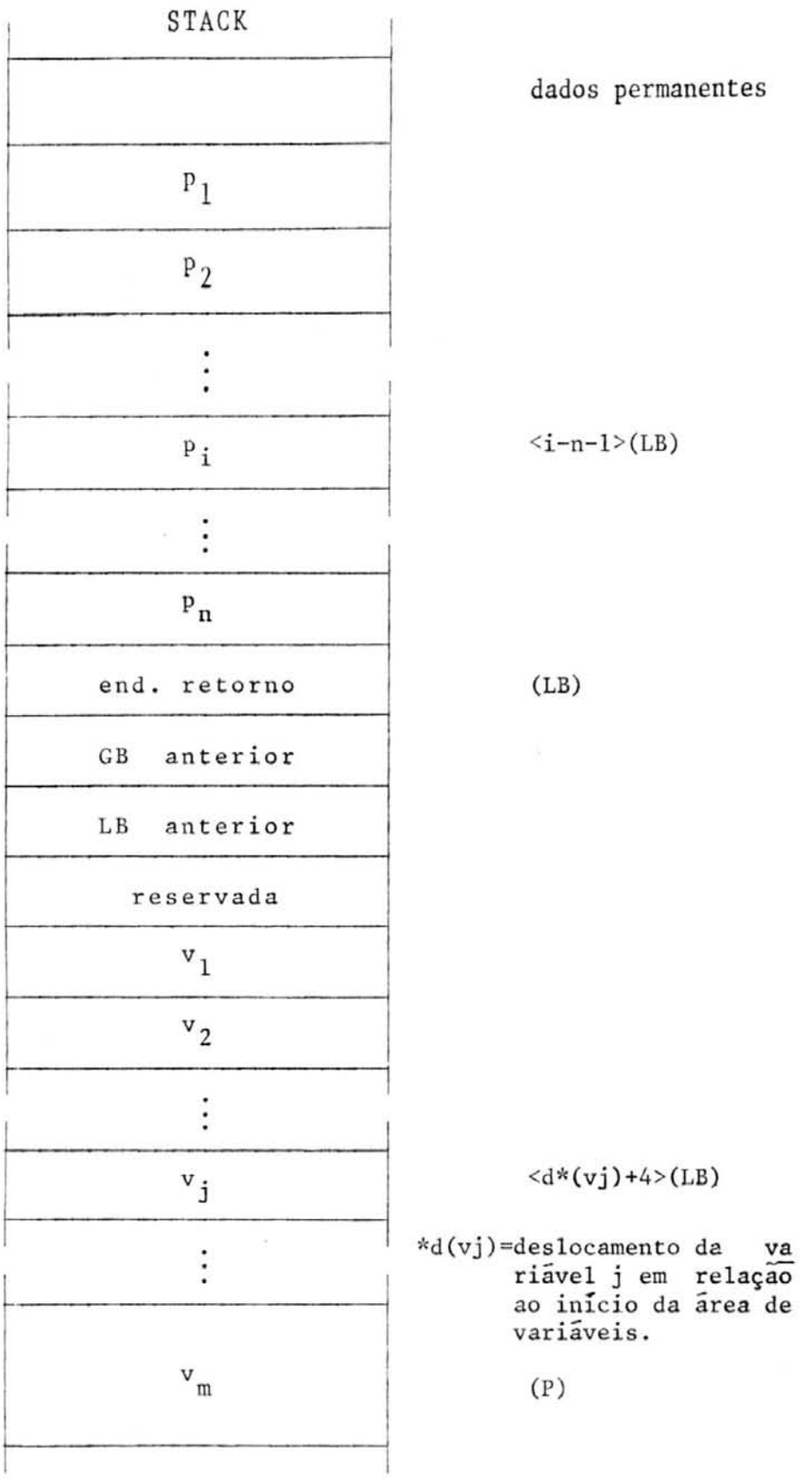


Figura 3.4 - Organização das áreas de dados temporários.

rios, que são alocados dinamicamente nos "stacks" dos processos, que como componentes ativos, determinam as chamadas dessas rotinas. A figura 3.4 mostra a organização de dados temporários na pilha de um processo.

3.3.1 As Áreas de Dados Permanentes

A primeira palavra na área de dados permanente de um componente do tipo processo, monitor ou classe, é reservada para controle e seu uso depende do tipo do componente. Nos processos ela é usada, na fase de auto-inicialização, para salvar o número do processo e o endereço de sua lista de parâmetros, obtidos em TBJOB, indexado pelo número do "job" correspondente ao processo. Nos monitores, ela deve conter um número que identifique de forma única cada monitor (esta identificação é usada pelas primitivas de controle de acesso e sincronização em monitores). Nas classes ela contém um código que identifica de forma única cada classe.

A partir da segunda palavra são reservadas tantas palavras quantos forem os parâmetros do componente. Cada palavra, após a inicialização do componente, conterá o endereço de um outro componente acessível. Componentes sem parâmetros não possuem essa área.

A parte final das áreas de dados permanentes contêm o espaço reservado para as variáveis do componente. Variáveis de organização complexa do tipo classe, embutidas nos componentes, tem suas áreas de dados permanentes definidas dentro desta área.

O endereçamento aos diversos elementos de dados na área de dados permanentes de um componente, é feito com o auxílio de um indexador, que, a cada instante, contém o

endereço do início da área de dados permanentes do componente em que está a execução do processo. Esse registrador, denominado GB (base global), implementado pelo registrador 16 da máquina, é definido em cada processo e seu valor inicial é o endereço da área de dados permanentes do processo. É modificado cada vez que o processo chama uma rotina do tipo "entry" de um monitor ou de uma classe e é restaurado cada vez que a execução retorna ao componente anterior. Assim, o endereçamento dos elementos de dados na área de dados permanentes é feito da seguinte maneira:

a) endereçamento de parâmetros: número de ordem do parâmetro indexado por GB;

b) endereçamento de variáveis: deslocamento da variável em relação ao início da área de variáveis mais o número de parâmetros mais um, indexado por GB.

3.3.2 As Áreas de Dados Temporários

A primeira palavra na área de dados temporários de uma rotina é reservada para um parâmetro especial que só é passado quando a rotina é do tipo "entry" em um monitor ou uma classe. Esse parâmetro é o endereço da área de dados permanentes do monitor ou da classe em questão e é usado no procedimento de entrada da rotina para ajustar o valor do registrador GB mencionado anteriormente. Por uma questão de uniformizar o processo de alocação, de endereçamento e de dealocação do espaço temporário, essa posição é reservada em todas as rotinas.

A partir da segunda palavra, estarão contidos os parâmetros da rotina, um em cada palavra, podendo ser valores ou endereços dos parâmetros atuais. Essa área só existe em rotinas que possuam parâmetros.

As quatro palavras seguintes contém valores responsáveis pelo controle do contexto de execução do programa. Na primeira posição é salvo o endereço de retorno, na segunda posição é salvo o valor de GB, antes de atualizar seu valor para o endereço da área de dados permanentes quando a rotina for do tipo "entry", na terceira posição é salvo o valor do registrador LB (base local, implementado pelo registrador 7 da máquina), antes de atualizar seu valor, que é corrigido para apontar a primeira posição deste bloco de quatro palavras e, finalmente, à última palavra é atribuído o valor zero. Essa última palavra só é usada nas rotinas de monitores embora seja reservada em todas as rotinas.

A última região na área de dados temporários, delimitada pelo registrador P (apontador de pilha, registrador 17 da máquina), contém as variáveis locais da rotina e é inicializada com zeros pelo procedimento de entrada da rotina. A partir daí, a pilha pode ser usada para salvamento de resultados parciais durante a computação, contanto que isto seja feito de forma controlada, para que no momento de retornar da rotina, o registrador P esteja apontando para o final da área de variáveis locais.

O endereçamento dos elementos de dados na área de dados temporários é feito com o auxílio do registrador LB da seguinte maneira:

a) endereçamento de parâmetros: número de ordem do parâmetro menos o número de parâmetros da rotina, menos um, indexado pelo registrador LB. Este endereço dá a posição do parâmetro na área de parâmetros e o valor obtido nessa posição pode ser, o valor do parâmetro atual ou o endereço onde está armazenado esse valor, dependendo, da convenção estabelecida para a passagem do parâmetro.

b) endereçamento de variáveis: deslocamento da variável em relação ao início da área de variáveis locais mais quatro, indexado do registrador LB.

3.4 Os Procedimentos de Início e de Finalização em um Processo

Todos os processos de um programa concorrente, exceto o processo inicial, executam procedimentos introdutórios e procedimentos de finalização padronizados.

São os seguintes os passos executados na introdução de um processo:

a) eliminação do vínculo existente entre o "job" associado ao processo e o seu terminal controlador. Este procedimento é necessário, uma vez, que o terminal controlador desse "job" é o pseudo terminal (PTY), usado pelo processo inicial para disparar a execução de todos os processos. O objetivo é atingido através de uma chamada da "monitor call" ATTACH, especificando um argumento, que indica que o "job" deve continuar sua execução em "det".

b) aquisição do "high segment" compartilhado. Este passo é realizado com uma chamada de GETSEG usando um argumento que especifica o arquivo que contém o "high segment" do programa. Note-se que nesse instante o processo inicial ainda não executou RENAME e por isso o segmento adquirido é o mesmo usado pelo processo inicial e que será compartilhado por todos os processos do programa.

c) habilitação para escrita no "high segment". É feita com uma chamada da "monitor call" SETUWP, com um argumento que indica que o segmento alto do programa deve ser desprotegido.

d) aquisição da identificação do processo e de seus parâmetros. Primeiro, uma chamada da "monitor call" PJOB fornece o número do "job" associado ao processo. Esse número é usado para indexar a tabela TBJOB, de onde o processo recupera a sua identificação (número do processo estabelecido pelo processo inicial) e o endereço de sua lista de parâmetros na tabela PROPAR. Se o processo tiver parâmetros, ele copia a sua lista de parâmetros para a área correspondente na sua área de dados permanentes.

e) inicialização dos registradores responsáveis pelo controle dos contextos de dados. O registrador P é inicializado com um apontador para a base da pilha do processo, o registrador GB é inicializado com um ponteiro para o início da área de dados permanentes do processo e o registrador LB é inicializado com zero.

f) sinalização do processo inicial para indicar que o processo está completamente inicializado. Esta sinalização é feita através da chamada de uma rotina de um monitor onde o processo incrementa um contador de processos inicializados e executa a operação SIGNAL em uma variável condição na qual o processo inicial pode ter executado a operação WAIT.

Encerrada esta fase de auto-inicialização, o processo está pronto para executar as operações específicas que realizem a função que ele deve desempenhar no programa concorrente.

O encerramento de um processo envolve duas etapas:

a) avisar o processo inicial que o processo está sendo encerrado. Isto é feito através de uma chamada de uma rotina em um monitor (o mesmo usado para indicar o início) onde o processo decrementa uma variável que continha ini-

cialmente o número de processos do programa.

b) encerrar a execução do processo liberando o "job" associado. Essa operação é levada a cabo com a execução da "monitor call" RUN aplicada a um bloco de argumentos que especifica o programa "LOGOUT" do sistema. A "monitor call" RUN, encerra a execução de um programa e faz com que o sistema operacional carregue e execute o programa especificado no seu argumento. O programa "LOGOUT" é um programa não residente do sistema TOPS-10 e que interage com ele para encerrar uma sessão de "timesharing". O programa tem a habilidade de verificar se um "job" está sendo executado em "det" e, nesse caso, fazer o encerramento da sessão sem tentar emitir mensagens no terminal controlador do "job".

3.5 As Rotinas nos Programas Concorrentes

O método de programação proposto, admite o uso de rotinas dos tipos "procedure" e "function", especificadas em um único nível (não aninhadas) em componentes de um programa concorrente. Para tanto, foram estabelecidas algumas convenções que estabelecem a forma de utilização dos registradores da máquina, a maneira de se fazer a passagem de parâmetros, a forma de chamar uma "procedure" ou "function" e de receber o valor retornado por uma rotina deste último tipo. Também foram estabelecidos os mecanismos apropriados para executar os procedimentos de entrada e de retorno nessas rotinas, de modo a atender as convenções anteriormente mencionadas.

3.5.1 Convenções para Uso dos Registradores

Além dos registradores GB, LB e P apresentados anteriormente, cada processo pode utilizar dois grupos de registradores designados como: A1, A2, A3, A4, A5 e A6 (registradores de 8 a 13 da máquina) e T1, T2, T3, T4, T5 e T6 (registradores de 1 a 6 da máquina). Além desses, o registrador zero da máquina foi designado como AC, para evidenciar o fato de que ele não pode ser usado como indexador. Todos esses registradores podem ser usados como registradores de trabalho, entretanto, o grupo T1 até T6 tem uma função especial nas chamadas de rotinas. Quando o número de parâmetros da rotina for menor ou igual a seis, então, eles devem ser passados através desses registradores. Nenhum dos registradores de trabalho é automaticamente preservado nas chamadas de rotinas.

3.5.2 Convenções para Passagem de Parâmetros

Os parâmetros passados a uma rotina podem ser os próprios valores dos parâmetros ou os seus endereços. O esquema de organização das áreas de dados temporários nas rotinas não admite a passagem de valores do tipo "array" de palavras. Ou seja, um parâmetro que deve ser passado "by value", deve ter valores que possam estar contidos em uma única palavra.

Quando o número de parâmetros de uma rotina não exceder seis, a passagem dos parâmetros deverá ser feita através dos registradores do grupo T1 até T6. Os parâmetros devem ser carregados nesses registradcres, obedecendo a ordem de definição dos mesmos, ou seja, o primeiro parâmetro deve ser carregado em T1, o segundo em T2 e assim por diante.

Quando o número de parâmetros for maior que seis, eles deverão ser passados diretamente através da pilha. O empilhamento dos parâmetros deve ser feito na ordem de definição dos mesmos e para fazê-lo, deve-se usar a instrução "PUSH" com o registrador P sendo referido como apontador de pilha.

As rotinas do tipo "entry", de monitores ou de classes, possuem um parâmetro especial, que é sempre o primeiro parâmetro em uma rotina desse tipo. O valor a ser passado nesses casos, deve ser o endereço da área de dados permanentes da classe ou do monitor que está sendo invocado. Dessa forma, essas rotinas podem ser usadas como operadores, que se aplicam a mais de uma ocorrência de um monitor ou de uma classe, definidos pelas suas áreas de dados permanentes. No que diz respeito a forma pela qual esse tipo especial de parâmetro deve ser passado, não há nenhuma diferença em relação aos demais.

3.5.3 As Chamadas de Rotinas

As chamadas de rotinas devem ser precedidas pela passagem de seus parâmetros, não se devendo esquecer que rotinas do tipo "entry" tem pelo menos um parâmetro: o endereço da área de dados permanentes da classe ou monitor que está sendo invocado. A chamada da rotina, propriamente, deve ser feita pela instrução "PUSHJ", referindo o registrador P como apontador de pilha. Essa instrução salva na pilha o endereço da instrução seguinte e faz o desvio para o endereço fornecido como segundo operando.

O retorno de uma rotina se dá para a instrução seguinte ao "PUSHJ". Se a rotina for do tipo "function" o registrador T1 conterá o valor computado pela função.

Nenhum dos registradores de trabalho é preservado automaticamente, ficando a cargo do programador a tarefa de fazê-lo, nas situações em que isto for necessário.

3.5.4 Os Procedimentos de Entrada e de Retorno de Rotinas

Os procedimentos de entrada e de retorno nas rotinas são padronizados, de modo a manter as especificações estabelecidas para as áreas de dados temporários, as convenções para passagem de parâmetros e os mecanismos de controle de acesso às áreas de dados permanentes dos componentes onde se inserem as rotinas, facilitando a programação.

Os passos executados na entrada de uma rotina são os seguintes:

a) se o número de parâmetros da rotina (contando com o parâmetro especial das rotinas do tipo "entry"), não excede seis, então, o endereço de retorno, que está no topo da pilha é removido, são empilhados os registradores T1 até Tn, onde n é o número de parâmetros e, finalmente, é re-empilhado o endereço de retorno. Se o número de parâmetros for maior que seis este passo não é executado, pois a pilha já deverá estar convenientemente organizada;

b) salva na pilha o valor do registrador GB;

c) salva na pilha o valor do registrador LB;

d) carrega em LB o endereço da posição da pilha que contém o endereço de retorno (o conteúdo do registrador P menos dois);

e) se a rotina for do tipo "entry", carrega no registrador GB o valor do primeiro parâmetro;

f) empilha o valor zero.

g) inicializa com zero, uma porção da pilha do tamanho da área de variáveis locais, iniciando na posição seguinte à que está sendo apontada pelo registrador P e ajusta o valor do registrador P, para que ele passe a apontar a última posição dessa área inicializada.

No procedimento de retorno de uma rotina, se supõe que se a rotina é do tipo "function", o valor a ser retornado está na primeira palavra da área de variáveis locais na pilha. Assim, são executados os seguintes passos para retornar:

a) se a rotina é do tipo "function", é carregado no registrador T1 o valor contido na posição apontada por LB mais quatro, que é o valor computado pela função;

b) é restaurado o valor do registrador GB com o valor contido na posição apontada por LB mais um;

c) é restaurado o valor do registrador LB com o valor contido na posição apontada por LB mais dois;

d) é ajustado o valor do registrador P, subtraindo-se dele um valor dado pela soma do número de parâmetros da rotina mais o tamanho da sua área de variáveis locais acrescido de cinco. Essa operação é feita com a instrução "ADJSP" que é específica para o ajuste de um registrador usado como apontador de pilha e, neste caso, serve para dealocar toda a área de dados temporários de uma rotina;

e) é feito o retorno através da instrução "JRST" para o endereço contido na posição apontada pelo registrador P acrescido do número de parâmetros da rotina mais dois.

3.6 Implementação de Processos, Monitores e Classes como Tipos Abstratos de Dados

Em função dos procedimentos e convenções propostos por este método de construção de programas concorrentes, torna-se possível a implementação de processos, monitores e classes como tipos abstratos de dados e a utilização posterior de uma ou mais ocorrências de componentes com os tipos especificados.

No caso dos processos de um programa concorrente, consegue-se isto facilmente, escrevendo, para cada tipo de processo, um programa correspondente, segundo as especificações anteriormente enunciadas, definindo-se assim os tipos abstratos correspondentes aos processos do programa. Para criar as ocorrências dos componentes assim definidos, basta montar convenientemente as tabelas de controle utilizadas pelo processo inicial. O processo inicial é, praticamente, um programa padrão, no qual apenas as tabelas e os procedimentos de inicialização dos monitores, são diferentes em cada programa concorrente.

Para os casos de monitores e classes, a especificação dos tipos abstratos corespondentes é feita escrevendo-se as rotinas desses tipos nos pontos convenientes. No caso dos monitores essas rotinas devem ser montadas no "high segment" do programa. No caso das classes, as rotinas poderão ser montadas no "high segment" do programa ou nos "low segments" de processos. Deverão ser montadas no "high segment", as rotinas de classes que sejam usadas em monitores ou que sejam usadas em mais de um processo. Poderão ser montadas no "low segment" de algum processo, as rotinas de classes que sejam usadas apenas nesse processo.

As ocorrências de componentes de um tipo monitor, definido da maneira descrita acima, são criadas quando se

insere no "high segment" uma ou mais ocorrências da área de dados permanentes desse tipo. Além das áreas de dados permanentes, é necessário inserir no processo inicial, as seções de código para introduzir os parâmetros de cada ocorrência e as chamadas das rotinas correspondentes aos blocos de inicialização dos monitores, que devem ser construídas de modo análogo às rotinas do tipo "entry".

A maneira de definir as ocorrências de componentes do tipo classe é semelhante a que foi descrita para o caso de monitores. A diferença está apenas nos lugares em que devem ser criadas as áreas de dados permanentes e os procedimentos de inicialização das classes.

Ocorrências de classes usadas em monitores devem ter suas áreas de dados permanentes embutidas nas áreas de dados dos monitores que as utilizam e os procedimentos para inicializar essas classes devem estar inseridos em alguma rotina desses monitores.

Ocorrências de classes em um processo devem ter suas áreas de dados permanentes embutidas na área de dados permanentes desse processo e os procedimentos para inicializar essas classes devem estar em alguma rotina do processo.

Ocorrências de classes em uma classe devem ter suas áreas de dados permanentes embutidas na área de dados permanente dessa classe e os procedimentos para inicializar essas classes devem estar em alguma rotina da classe que as utiliza.

4. PRIMITIVAS PARA CONTROLE DE ACESSO E SINCRONIZAÇÃO EM MONITORES

Para garantir que a cada instante apenas um processo possa estar acessando um monitor, as rotinas "entry" desses componentes, são guarnecidas por chamadas da primitiva "ENTER_MONITOR". Essa primitiva garante que no caso de mais de um processo tentar acessar um mesmo monitor, através de qualquer de suas "entries", apenas um deles conseguirá fazê-lo efetivamente, permanecendo os demais bloqueados, até que o monitor seja liberado pelo processo que ganhou o acesso. Cada vez que um processo em um monitor executa uma das primitivas: "WAIT", "SIGNAL" ou "EXIT_MONITOR", ele libera o monitor para que um outro processo possa acessá-lo.

A primitiva "EXIT_MONITOR" é executada toda vez que um processo termina a sua execução em uma "entry". Ela faz parte do procedimento de retorno de uma rotina "entry" de um monitor. A sua execução faz com que um dos processos bloqueados por terem executado "ENTER_MONITOR" seja selecionado para entrar no monitor. Se não houver processos bloqueados esperando para ingressar no monitor, esse monitor ficará disponível para o primeiro processo que executar "ENTER_MONITOR" em uma de suas "entries".

A primitiva "WAIT" pode ser executada em uma rotina "entry" de um monitor, por um processo que necessite aguardar uma determinada condição que deva ser criada por um outro processo. Uma chamada de "WAIT" deve fazer referência a uma variável de condição na qual o processo fica bloqueado, liberando o monitor para que outros processos possam acessá-lo. A liberação do monitor ocorre da maneira já descrita para a primitiva "EXIT_MONITOR".

A primitiva "SIGNAL" pode ser executada em uma rotina "entry" de um monitor, por um processo que precise in

dicar a ocorrência de uma dada condição. Essa condição pode estar sendo esperada por um outro processo que tenha executado "WAIT" anteriormente. Uma chamada de "SIGNAL", deve fazer referência a uma variável de condição, na qual pode existir um processo bloqueado a espera do sinal. Se existir, efetivamente, um processo a espera da condição sinalizada, a execução da primitiva "SIGNAL" faz com que esse processo seja continuado, a partir da instrução seguinte a chamada de "WAIT" que o bloqueou (dentro do monitor) e bloqueia o processo que executou a primitiva "SIGNAL". No momento em que o processo sinalizado liberar o monitor, seja pela execução de "EXIT_MONITOR", seja por uma nova chamada de "WAIT", o processo sinalizador voltará a ser executado, na instrução seguinte à chamada de "SIGNAL".

As variáveis de condição mencionadas na descrição das primitivas "WAIT" e "SIGNAL", não são filas de múltiplos processos, ou seja, apenas um processo pode estar bloqueado em uma variável de condição em um dado instante.

As primitivas descritas acima, são implementadas através de rotinas contidas no "high segment" dos programas concorrentes. Elas se baseiam no uso das "monitor call's" ENQ./DEQ., HIBER e WAKE e no uso de estruturas de dados e algoritmos adequados.

4.1 Implementação da Primitiva "ENTER MONITOR"

Para possibilitar a implementação da primitiva "ENTER_MONITOR", cada monitor, ou mais precisamente, cada ocorrência dos diversos tipos de monitores de um programa concorrente, é definida como um recurso virtual, identificado de forma única pelo código existente na primeira palavra da área de dados permanentes que representa essa ocorrência. A primitiva "ENTER_MONITOR" consiste em uma chamada da

"monitor call" ENQ. aplicada sobre um recurso virtual definido da maneira descrita acima.

Toda rotina do tipo "entry" nos monitores, inicia por uma chamada da rotina "ENTMON", logo após os procedimentos de entrada na rotina. Essa rotina recebe como parâmetro, no registrador T1, o código de identificação do monitor, prepara um bloco de argumentos definindo o recurso virtual correspondente e, em seguida, faz a chamada da "monitor call" ENQ.. Quando o sistema operacional devolve o controle, a rotina "ENTMON" termina, retornando à rotina "entry" a partir de onde foi invocada. Nesse momento o processo está com direito de acesso exclusivo ao monitor garantido.

Para evitar interferência entre duas ocorrências de um mesmo programa concorrente sendo executadas simultaneamente, a rotina "ENTMON", no momento da preparação do bloco de argumentos para a chamada de ENQ., inclui na identificação do recurso virtual, o número do "job" correspondente ao processo inicial do programa.

4.2 Implementação da Primitiva "WAIT"

A primitiva "WAIT" está implementada na rotina de mesmo nome. Recebe dois parâmetros: no registrador T1 o código de identificação do monitor a partir do qual está sendo invocada e no registrador T2 o endereço da variável de condição onde o processo deve ser bloqueado. A variável de condição é representada em uma palavra reservada para tal finalidade na área de variáveis permanentes do monitor.

A rotina "WAIT" opera segundo o algoritmo abaixo:

a) Verifica se a variável de condição está desocupada (isto é, se seu conteúdo é zero). Se não, emite uma men

sagem indicativa do erro de programação e encerra a execução do processo fazendo uma chamada da "monitor call" EXIT.

b) Salva o número do "job" correspondente ao processo que está executando "WAIT", obtido através da "monitor call" PJOB, na variável de condição apontada pelo registrador T2.

c) Chama a rotina "XITMON", que implementa a primitiva "EXIT_MONITOR", liberando o acesso ao monitor.

d) Chama a "monitor call" HIBER, bloqueando o processo até que ele seja despertado por um outro processo da mesma "p,pn" que execute a "monitor call" WAKE referindo-se ao número do "job" do processo bloqueado.

e) Este passo é executado no momento em que um outro processo executa "SIGNAL" na mesma variável de condição especificada pelo processo que está em "WAIT". Salva o número do "job" correspondente ao processo sinalizador (que está agora armazenado na variável de condição) na posição reservada na área de dados temporários da rotina "entry" a partir da qual foi feita a chamada de "WAIT". Essa posição é a posição apontada pelo registrador LB acrescido de três e que anteriormente, na inicialização da área de dados temporários havia sido zerada.

f) Zera a variável de condição e retorna à rotina "entry" de onde foi invocada.

Note-se, que o número do "job" correspondente ao processo sinalizador é guardado pelo processo sinalizado, porque, no momento em que este for liberar o acesso ao monitor, deverá fazê-lo em favor do processo que o sinalizou.

4.3 Implementação da primitiva "SIGNAL"

A primitiva "SIGNAL", implementada na rotina de mesmo nome, recebe dois parâmetros: no registrador T1 o código de identificação do monitor a partir do qual está sendo invocada e no registrador T2 o endereço da variável de condição a ser sinalizada.

A rotina "SIGNAL" opera segundo o algoritmo abaixo:

a) Se o valor da variável apontada pelo registrador T2 for zero; então retorna imediatamente.

b) Salva em um registrador de trabalho o número do "job" contido na variável de condição.

c) Salva na variável de condição o número do "job" correspondente ao processo que está executando "SIGNAL", obtido através da "monitor call" PJOB.

d) Executa a "monitor call" WAKE sobre o "job" cujo número foi obtido no passo b, efetivando a sinalização da condição.

e) Executa a "monitor call" HIBER, bloqueando o processo que chamou "SIGNAL", até que ele seja despertado por uma chamada de WAKE feita pelo processo sinalizado, devolvendo o monitor ao processo sinalizador.

f) Retorna a rotina "entry" de onde foi invocada.

É interessante notar que um processo que executa "SIGNAL" não libera efetivamente o monitor, mas apenas o cede por empréstimo ao processo sinalizado, bloqueando-se até que este último lhe devolva o direito de acessar exclusivamente o monitor.

4.4 Implementação da Primitiva "EXIT_MONITOR"

A primitiva "EXIT_MONITOR" é implementada pela rotina "XITMON" e é invocada em duas situações: a primeira havia sido mencionada na apresentação desta primitiva e diz respeito ao procedimento de retorno de uma "entry" de um monitor, a segunda foi mencionada na descrição da implementação da primitiva "WAIT", onde o processo que executa "WAIT" precisa liberar o acesso ao monitor.

Por causa da situação especial criada no caso dos processos que executam "WAIT" que, quando sinalizados, ganham acesso ao monitor através de um direito cedido por empréstimo pelo processo sinalizador, a implementação da primitiva "EXIT_MONITOR", não pode, como se poderia pensar à primeira vista, ser feita pela simples execução da "monitor call" DEQ. aplicada sobre o recurso virtual correspondente ao monitor. Não pode ser assim, porque, um processo sinalizado deve devolver o monitor ao processo que o sinalizou e não pode, também, porque não é permitido fazer DEQ. em um recurso para o qual não tenha sido feito o ENQ. correspondente.

A rotina "XITMON" recebe como parâmetro, no registrador T1, o código do monitor a ser liberado e funciona segundo o algorítmo seguinte:

a) verifica na área de dados temporários o conteúdo da posição apontada por LB acrescido de três; se for zero, vai ao passo seguinte; caso contrário vai para o passo e;

b) prepara um bloco de argumentos definindo o recurso virtual correspondente ao monitor;

c) executa a "monitor call" DEQ. sobre o recurso especificado pelo bloco de argumentos;

d) retorna ao ponto onde foi chamada;

e) executa "monitor call" WAKE usando como argumento o número do "job" contido na posição apontada por LB acrescido de três, que é o "job" correspondente ao processo que sinalizou este processo que agora está executando "EXIT_MONITOR";

f) retorna ao ponto de onde foi chamada.

5. SUPORTE À APLICAÇÃO DO MÉTODO

Pensava-se, inicialmente, usar exclusivamente a linguagem assembler do DEC-10 (MACRO-10), na construção de programas concorrentes baseados no método apresentado neste trabalho. O suporte à aplicação do método se constituiria, basicamente, de um conjunto de macro-instruções que implementassem os procedimentos padronizados descritos anteriormente.

Entretanto, a organização de um programa concorrente, de suas áreas de dados, de suas rotinas e, principalmente, as referências a esses objetos, através da linguagem MACRO-10, mostrou, nas experiências realizadas, ser um trabalho demasiadamente penoso e sujeito a erros.

Para amenizar o esforço requerido na programação e tornar viável a aplicação do método em um prazo mais curto, optou-se pela utilização de uma linguagem de programação, que permitisse ao programador um certo grau de abstração na especificação dos componentes dos programas e, ainda, que lhe desse alguma facilidade para manipular esses componentes. Essa linguagem de programação não existia no DEC-10 e deveria ser implementada da maneira mais simples possível, em função do tempo limitado de que se dispunha.

Foi especificada uma linguagem de programação, que é um meio termo entre as linguagens PASCAL CONCORRENTE e MACRO-10. É uma linguagem que incorpora a estrutura do PASCAL CONCORRENTE, permitindo a declaração de constantes, de tipos abstratos de dados, de variáveis e de rotinas e que, pela necessidade de simplificação, utiliza as instruções do macro assembler do DEC-10 para a manipulação dos objetos declarados em alto nível.

O programa tradutor foi escrito em SNOBOL, pela facilidade que essa linguagem oferece no tratamento de cadeias de caracteres e de algumas estruturas de dados complexas. O produto por ele gerado constitui-se de um conjunto de programas em linguagem MACRO-10 pura e de um conjunto de comandos para dirigir a compilação desses programas e a montagem dos módulos de execução que constituem o programa concorrente.

Os programas traduzidos utilizam macro-instruções pré-definidas, que implementam os procedimentos de controle descritos nos capítulos anteriores. Essas macro - instruções estão armazenados em uma biblioteca de macro-instruções, que é usada durante a compilação dos programas gerados pelo tradutor.

5.1 A Linguagem de Programação

O objetivo da linguagem de programação especificada é facilitar a concepção de objetos como processos, monitores e classes através de tipos abstratos de dados, da mesma maneira que na linguagem PASCAL CONCORRENTE, mantendo entretanto o nível de intimidade com a máquina proporcionado pela linguagem MACRO-10. Não se pretende de forma alguma colocar essa última característica como uma vantagem da linguagem adotada. Ao contrário, ela foi imposta pela necessidade de simplificar o programa tradutor da linguagem, em função do tempo demasiadamente limitado de que se dispunha para a implementação desse programa.

Da linguagem PASCAL CONCORRENTE foi absorvida a estrutura principal de especificação de um programa concorrente, permitindo-se o uso das declarações:

CONST - para a especificação de constantes inteiras (números de ponto fixo representados em 36 bits) ou de cadeias de caracteres alfanuméricos;

TYPE - para a especificação de tipos abstratos como processos, monitores, classes e "arrays" unidimencionais de tipos básicos que podem ser:

WORD - qualquer padrão representável em palavras de 36 bits;

BYTE - bytes de sete bits usados na representação dos caracteres ASCII;

BOOLEAN - usado apenas com finalidade de documentação, pois na realidade corresponde a palavras de 36 bits que podem conter qualquer padrão de bits;

QUEUE - usado para definir variáveis do tipo condição, tem carater apenas de documentação pois, como no caso anterior, utiliza palavras de 36 bits que podem conter qualquer padrão de bits;

VAR - para a especificação de variáveis dos tipos básicos descritos acima, ou dos tipos abstratos especificados através da declaração TYPE;

PROCEDURE - para a especificação de rotinas do tipo "procedure", permitindo o uso da cláusula NETRY nas rotinas desse tipo;

FUNCTION - para a especificação de rotinas do tipo "function", permitindo o uso da cláusula ENTRY nas rotinas desse tipo.

Ainda da linguagem PASCAL, foi mantida a estrutura "BEGIN lista de instruções END", para os blocos do corpo principal dos processos, e para os blocos das rotinas.

As instruções da linguagem, são as instruções permitidas na linguagem MACRO-10, usando-se apenas uma notação um pouco diferente para fazer referências aos objetos declarados em alto nível. Além dessas instruções, foi incorporada ainda do PASCAL, a instrução INIT que se aplica à inicialização de processos, monitores e classes.

5.1.1 A Sintaxe da Linguagem de Programação

A seguir é apresentada a BNF da sintaxe de um programa concorrente.

```
<programa concorrente>::=
        <título do programa>
        <declarações de constantes>
        <declarações de tipos abstratos de dados>
        <declarações de variáveis>
        <bloco do processo inicial>

<título do programa>::=
        PROGRAM <identificador>;

<declarações de constantes>::=
        <vazio> |
        CONST <lista de declarações de constantes>;

<lista de declarações de constantes>::=
        <identificador> = <constante>
        <lista aux. de declarações de constantes>
```

<lista aux. de declarações de constantes>::=

<vazio> |
;<lista de declarações de constantes>

<constante>::= <número> | <cadeia alfanumérica>

<cadeia alfanumérica>::= ' <lista de caracteres> '

<lista de caracteres>::=<carater><lista aux. de caracteres>

<lista aux. de caracteres>::=<vazio>|<lista de caracteres>

<declarações de tipos abstratos de dados>::=

<vazio> |
TYPE <declaração de tipo abstrato de dados>
<declarações de tipos abstratos de dados>

<declaração de tipo abstrato de dados>::=

<identificador> = <especificação de tipo>;
<declaração aux. de tipo abstrato de dados>

<declaração aux. de tipo abstrato de dados>::=

<vazio> |
<declaração de tipo abstrato de dados>

<especificação de tipo>::=

<especificação de array> |
<especificação de componente de sistema>

<especificação de array>::=

ARRAY <número> OF <tipo básico>

<tipo básico>::= WORD | BOOLEAN | BYTE | QUEUE

<especificação de componente de sistema>::=

<especificação de processo> |
<especificação de monitor> |
<especificação de classe>

```
<especificação de processo>::=
            PROCESS <parâmetros>;
           <stacksize>
           <declarações de constantes>
           <declarações de tipos abstratos de dados>
           <declarações de variáveis>
           <declarações de rotinas>
           <bloco>

<stacksize>::=<vazio> | STACK-SIZE  =  <número> ;

<especificação de monitor>::=
            MONITOR <parâmetros> ;
           <declarações de constantes>
           <declarações de tipos abstratos de dados>
           <declarações de variáveis>
           <declarações de rotinas>
           <bloco de inicialização>

<especificação de classe>::=
            CLASS <parâmetros> ;
           <declarações de constantes>
           <declarações de tipos abstratos de dados>
           <declarações de variáveis>
           <declarações de rotinas>
           <bloco de inicialização>

<parâmetros>::= <vazio> | ( <lista de parâmetros> )

<lista de parâmetros>::=
           <lista de identificadores>  :  <identificador>
           <lista auxiliar de parâmetros>

<lista auxiliar de parâmetros>::=
           <vazio> |
            ; <lista de parâmetros>
```

<declarações de rotinas>::=

<vazio> |
<especificação de rotina> ;
<declaração de rotinas>

<especificação de rotina>::=

<declaração de rotina> ;
<corpo de rotina>

<declaração de rotina>::=

PROCEDURE <entry> <identificador>
<parâmetros formais> |
FUNCTION <entry> <identificador>
<parâmetros formais> : <tipo básico>

<parâmetros formais>::=

<vazio> |
(<lista de parâmetros formais>)

<lista de parâmetros formais>::=

<reference> <lista de identificadores>
: <especificação de tipo de parâmetro>
<lista auxiliar de parâmetros formais>

<reference>::= <vazio> | VAR

<especificação de tipo de parâmetro>::=

<tipo básico> |
<identificador>

<lista auxiliar de parâmetros formais>::=

<vazio> |
; <lista de parâmetros formais>

<entry>::= <vazio> | ENTRY

<corpo de rotina>::=

<declarações de constantes>
<declarações de tipos abstratos de dados>
<declarações de variáveis>
<bloco>

<declarações de variáveis>::=

<vazio> |
VAR <lista de declaração de variáveis> ;

<lista de declaração de variáveis>::=

<lista de identificadores> : <tipo de variáveis>
<lista aux. de declaração de variáveis>

<lista aux. de declaração de variáveis>::=

<vazio> |
; <lista de declaração de variáveis>

<tipo de variáveis>::= <tipo básico>|<identificador>

<lista de identificadores>::=

<identificador> |
<identificador> , <lista de identificadores>

<bloco do processo inicial>::= <bloco>

<bloco de inicialização>::= <bloco>

<bloco>::= BEGIN <lista de instruções> END

<lista de instruções>::=

<vazio> |
<instrução> ; <lista de instruções>

<instrução>::=

<instrução init> |
<instrução MACRO-10 modificada>

```
<instrução init>::=
            INIT <lista de variáveis à inicializar>

<lista de variáveis à inicializar>::=
            <identificador> <direitos de acesso> |
            <identificador> <direitos de acesso> ,
            <lista de variáveis à inicializar>

<direitos de acesso>::=
            <vazio> |
             (  <lista de identificadores>  )

<instrução MACRO-10 modificada>::=
            <label> <operador> <operandos>

<label>::=  <vazio> |
            <identificador> :

<operador>::=
            <vazio> |
            <operador implementado em hardware> |
            <operador implementado em software> |
            <pseudo operador MACRO-10> |
            <macro instrução>

<operandos>::=
            <vazio> |
            <operando> |
            <operando> , <operandos>

<operando>::=
            <qualquer operando MACRO-10> |
            <indireto> <expressão1> = <identificador>
                       <expressão2> |
            <cadeia alfanumérica>
```

```
<indireto>::=  <vazio> |

<expressão1>::=

            <vazio> |
            <número>+ |
            <número>-

<expressão2>::=

            <vazio> |
             +  <número> |
             -  <número>

<identificador>::=

            <letra>    <identificador1>

<identificador1>::=

            <vazio> |
            <letra> <identificador1> |
            <dígito> <identificador1>

<número>::= <dígito> |
            <dígito> <número>

<letra>::=   A|B|C|...|X|Y|Z|a|b|c|...|x|y|z|_

<dígito>::=  0|1|2|...|7|8|9
```

A linguagem permite a inserção de comentários que são cadeias de caracteres delimitadas por aspas (").

Foram implementados ainda três comandos de compilação:

a) o comando $LIST - para indicar que deve ser produzida uma listagem do fonte.

b) O comando $NOLIST - para indicar que a listagem do fonte deve ser interrompida.

c) O comando $INCLUDE <especificação de arquivo> indica que o texto contido no arquivo especificado deve ser incluído no programa.

Esses comandos de compilação devem ser colocados nos programas como se fossem comentários: "$<comando>".

5.2 O Programa Tradutor

O tradutor foi escrito em SNOBOL, pelas facilidades que essa linguagem oferece para a manipulação de "strings" de caracteres e para o tratamento de algumas estruturas de dados bastante complexas, necessárias à implementação de um programa deste tipo.

O programa reconhece a linguagem descrita anteriormente e produz como resultado o seguinte conjunto de arquivos: um arquivo de comandos que facilita a compilação dos programas produzidos e a construção dos módulos executáveis correspondentes, um arquivo para cada tipo de processo declarado no programa concorrente, contendo as versões MACRO-10 desses processos, e um arquivo com a versão MACRO-10 do processo inicial e os componentes que devem ser montados no "high segment".

Os programas produzidos pelo tradutor utilizam as definições de símbolos e de macro-instruções armazenados em três bibliotecas. São elas as seguintes:

UUOSYM - arquivo fornecido pelo fabricante da máquina, contém as definições de símbolos usados para identi

ficar as "monitor call's" e símbolos usados na construção dos argumentos desses operadores;

CONSTS - arquivo construído para atender as necessidades deste método de programação, contém as definições de algumas constantes e símbolos usados nos programas gerados pelo tradutor, tais como as definições dos registradores da máquina, segundo a convenção adotada.

MACROS - arquivo construído para atender as necessidades deste método de programação, contém as definições das macro-instruções usadas na implementação dos procedimentos e das convenções descritas nos capítulos anteriores e, ainda, algumas macro-instruções que podem ser usadas pelo programador para facilitar a manipulação dos objetos de dados ou para dar maior clareza a codificação dos programas.

Além dessas bibliotecas, para cada programa concorrente, é gerado uma biblioteca específica, contendo as definições dos símbolos especificados no "high segment" do programa e que precisam estar disponíveis quando da compilação dos programas pelo MACRO-10.

A listagem do programa tradutor encontra-se no Apêndice 1.

5.2.1 A Compilação de Programas

A compilação dos programas concorrentes é feita em duas etapas: na primeira o programa é submetido ao tradutor e na segunda, cada programa gerado pelo tradutor, é compilado, pelo MACRO-10, e transformado em um módulo executável, pelo programa LINK.

Para realizar essas etapas de forma mais confortável, foi utilizada uma facilidade oferecida por um programa do sistema DEC-10 denominado MIC. Esse programa possibilita que se crie "macro-comandos" que atendam necessidades específicas do usuário. Esses "macro-comandos" nada mais são que listas de comandos do TOPS-10 e algumas instruções próprias do programa MIC, que são lidas de um arquivo em disco, interpretadas pelo MIC e passadas ao TOPS-10 através dos "buffers" do terminal do usuário.

Usando o programa MIC foi criado o "macro-comando" C, que é usado para submeter um programa concorrente ao tradutor. A sintaxe do comando que inicia a tradução de um programa concorrente é a seguinte:

DO C <especificação do fonte>[<especificação de listagem>]

A <especificação do fonte> deve ser a especificação do arquivo que contém o programa fonte. A especificação de listagem é opcional e pode ser fornecida de duas maneiras:

a) /LIST ou

b) /LIST:<especificação da saída>

No primeiro caso a listagem do programa é emitida na impressora de linhas do sistema. No último, a listagem é gerada em um arquivo indicado por <especificação da saída>.

As especificações de arquivos tanto no caso de <especificação do fonte>, como em <especificação da saída>, seguem um padrão simplificado estabelecido para o TOPS-10:

[<device>:] [<file name>[.<extension>]]

Além dos novos arquivos fontes produzidos pelo tradutor, é gerado um arquivo com o mesmo nome (<file name>) do arquivo fonte, contendo um "macro-comando" para efetuar a segunda etapa da compilação do programa concorrente. Esta segunda etapa é iniciada pelo comando.

DO <file name> [/LIST]

onde <file name> é o mesmo nome do arquivo fonte. A opção /LIST pode ser usada para produzir listagens da compilação dos programas pelo MACRO-10.

Após a execução dessas duas etapas, o programa concorrente está pronto para ser executado. Para iniciar a sua execução utiliza-se o comando.

RUN <nome do programa>

onde <nome do programa> é formado pelo seis primeiros caracteres (menos de seis se for o caso) do identificador usado na declaração PROGRAM, a primeira do programa concorrente.

5.3 As Macro-Instruções Utilizadas

Nesta seção serão descritas duas categorias de macro-instruções definidas na biblioteca MACROS:

a) as macro-instruções que implementam os diversos procedimentos padronizados que, em suma, constituem o método de programação proposto e

b) macro-instruções que opcionalmente podem ser usadas pelo usuário para facilitar a programação ou para tornar o seu programa mais legível.

As listagens das macros encontram-se no apêndice 2 deste trabalho.

INIHGH - esta macro-instrução é usada para definir as partes fixas do código no "high segment" de um programa concorrente. São as primitivas de controle de acesso e as primitivas de sincronização em monitores, as rotinas e áreas de dados permanentes do monitor usado para controlar a inicialização e finalização dos processos, as rotinas e áreas de dados permanentes do monitor responsável pela comunicação dos processos com o operador do programa e finalmente, a tabela TBJOB descrita no capítulo 3.

INILOW - esta macro-instrução é usada para definir as partes fixas do código no "low segment" do programa que implementa o processo inicial. A expansão dessa macro resulta na geração do código de controle do terminal (TTY) usado para a comunicação com o operador do programa, das rotinas de manipulação do pseudo terminal (PTY) usado para fazer o disparo dos processos e finalmente dos procedimentos introdutórios do processo inicial, tais como:

a) aquisição do "high segment" (usa o único parâmetro da macro-instrução, o nome do arquivo que contém o "high segment").

b) inicialização dos monitores de controle.

c) inicialização da classe responsável pela comunicação entre os processos e o operador do programa.

FINLOW - esta macro-instrução define os procedimentos finais inseridos no "low segment" do processo inicial. Compreende a rotina de disparo dos processos, os procedimentos de proteção do "high segment" contra interferências de outras ocorrências do mesmo programa concorrente, a chamada da rotina que estabelece a comunicação dos processos com o operador do programa através do terminal controlador e o código de encerramento do processo inicial.

INIPRO - esta macro-instrução é usada para gerar os procedimentos iniciais de um processo. Recebe três parâmetros: o nome do arquivo que contém o "high segment" do programa, usado na aquisição do "high segment", o nome simbólico atribuido à área de dados permanentes do processo, usado para inicializar o registrador GB, e o número de parâmetros (direitos de acesso) do processo, usado para a aquisição dos parâmetros.

FINPRO - esta macro-instrução é usada para gerar os procedimentos de finalização de um processo.

EPROCE - esta macro-instrução define os procedimentos introdutórios de uma rotina do tipo "entry". Ela recebe três argumentos:

a) número de parâmetros da rotina;
b) tamanho (número de palavras) da área de variáveis temporárias da rotina;
c) um valor lógico que indica se a rotina é do tipo "function" (1) ou se é do tipo "procedure" (Ø).

O código gerado por esta macro-instrução é responsável pela criação e atribuição dos valores iniciais da área de dados temporários de uma rotina do tipo "entry".

XPROCE - esta macro-instrução define os procedimentos de retorno de uma rotina do tipo "entry". Ela possui três argumentos:

a) número de parâmetros da rotina;
b) tamanho da área de variáveis temporárias da rotina;
c) valor lógico que indica se a rotina é do tipo "function" (1) ou se é do tipo "procedure" (Ø).

EMONIT - esta macro-instrução é usada nas rotinas do tipo "entry" em monitores, logo após os procedimentos introdutórios (EPROCE). Ela define os procedimentos necessários à chamada da primitiva "ENTER_MONITOR".

XMONIT - esta macro-instrução é usada nas rotinas do tipo "entry" em monitores, precedendo os procedimentos de retorno (XPROCE) das rotinas. Ela define os procedimentos necessários à chamada da primitiva "EXIT_ _MONITOR".

EPROC - esta macro-instrução define os procedimentos introdutórios de rotinas que não sejam do tipo "entry". Ela possui três argumentos:

a) número de parâmetros da rotina;
b) tamanho (número de palavras) da área de variáveis temporários da rotina;

c) um valor lógico que indica se a rotina é do tipo "function" (1) ou se é do tipo "procedure" (Ø).

O código gerado por esta macro-instrução é responsável pela criação e atribuição dos valores iniciais da área de dados temporários de uma rotina.

XPROC - esta macro-instrução define os procedimentos de retorno das rotinas que não sejam do tipo "entry". Ela recebe três argumentos:

a) número de parâmetros da rotina;

b) tamanho da área de variáveis temporárias da rotina;

c) um valor lógico que indica se a rotina é do tipo "function" (1) ou se é uma rotina do tipo "procedure" (Ø).

IMPAR - esta macro-instrução é usada no processo inicial para introduzir os parâmetros de um monitor. Ela utiliza dois argumentos:

a) endereço do parâmetro atual;

b) endereço onde deve ser introduzido o parâmetro na área de dados permanentes do monitor.

INIMON - esta macro-instrução é usada no processo inicial para fazer a inicialização de monitores. Ela utiliza dois parâmetros:

a) endereço da área de dados permanentes do monitor a ser inicializado;

b) endereço da rotina correspondente ao bloco de inicialização do monitor.

O código gerado por esta macro-instrução, prepara e faz a chamada da rotina correspondente ao bloco de inicialização do monitor.

ICPAR - esta macro-instrução é usada para introduzir os parâmetros de uma classe. Ela utiliza três argumentos:

a) endereço do parâmetro atual;

b) endereço onde deve ser introduzido o parâmetro na área de dados permanentes da classe;

c) valor lógico que indica se o parâmetro que está sendo introduzido é uma variável definida no componente que está inicializando a classe (1) ou se esse parâmetro é um dos parâmetros do componente que está inicializando a classe (0).

INICLA - esta macro-instrução é usada para inicializar uma classe. Ela utiliza dois parâmetros:

a) endereço da área de dados permanentes da classe;

b) endereço da rotina correspondente ao bloco de inicialização da classe.

O código gerado por esta macro-instrução prepara e faz a chamada da rotina correspondente ao bloco de inicialização da classe.

ITBNAM - esta macro-instrução é usada no processo inicial para montar a tabela PRONAM. Ela utiliza dois parâmetros:

a) nome do programa que implementa o tipo abstrato de dados correspondente ao processo que está sendo introduzido na tabela;

b) número de ordem do processo que está sendo introduzido na tabela.

Esta macro-instrução é chamada no processo inicial, tantas vezes quantos forem os processos do programa concorrente.

ITBJP - esta macro-instrução é usada no processo inicial para montar a tabela PROJOB e iniciar a construção das tabelas PARPRO e PROPAR que estão relacionadas com os direitos de acesso dos processos. Ela possue apenas um parâmetro: o número de processos componentes do programa concorrente (a menos do processo inicial).

ITBPAR - esta macro-instrução é usada no processo inicial para montar a tabela PARPRO. Ela possue apenas um argumento:
o endereço do início da lista de parâmetros de um processo na tabela PROPAR. Esse endereço é introduzido na tabela PARPRO na posição correspondente ao processo.

IPPAR - esta macro-instrução é usada no processo inicial para introduzir, na tabela PROPAR, os parâmetros (direitos de acesso) de um processo. Ela utiliza apenas um argumento:
o endereço de um componente (área de dados permanentes) acessível a um processo.
Para cada processo componente do programa concorrente, essa macro é chamada tantas vezes quantos forem os parâmetros do processo.

CTWSTR - esta macro-instrução é usada nos processos de um programa concorrente para transmitir mensagens ao operador do programa. O código gerado pela expansão desta macro-instrução corresponde aos procedimentos de chamada da rotina REQ.Ø, no monitor que serve de interface entre os processos e o processo inicial.
CTWSTR possue um parâmetro: o endereço de uma área de dados, que deve conter uma cadeia de caracteres ASCII, terminando em um carater nulo. O comprimento dessa cadeia não deve exceder a 80 caracteres e, se o primeiro carater for interrogação, após o retorno do monitor, a área de dados referida conterá uma mensagem enviada pelo operador.

COPYW - esta macro-instrução pode ser usada para copiar um bloco de palavras de uma área de dados para outra. Ela possue quatro parâmetros:

a) um registrador contendo o endereço da área de dados fonte;

b) um registrador contendo o endereço da área de dados destino;

c) número de palavras a serem copiadas;

d) um registrador para ser usado como área de trabalho.

ASCPTR - esta macro-instrução pode ser usada para construir um "byte pointer" para uma cadeia de caracteres ASCII. Ela possue dois parâmetros:

a) um registrador onde é montado o "byte pointer";

b) o endereço do início do bloco de palavras que contém a cadeia de caracteres.

CALL - esta macro-instrução gera a instrução "PUSHJ P,<ROTINA>" e pode ser usada para dar maior clareza a codificação dos programas. Possue apenas um parâmetro que deve ser o endereço da rotina a ser chamada.

RETURN - esta macro-instrução gera a instrução "POPJ P," e pode ser usada para dar maior clareza a codificação dos programas. Não possue parâmetros.

SAVE - esta macro-instrução é usada para salvar na pilha o valor de um registrador. Ela gera a instrução "PUSH P, <registrador>" e possue um parâmetro que deve ser o registrador a ser salvo.

REST - esta macro-instrução deve ser usada para restaurar o valor de um registrador a partir da pilha. Ela gera a instrução "POP P,<registrador>" e possue como argumento a identificação do registrador a ser restaurado.

GOTO - esta macro-instrução gera a instrução "JRST <label>" e pode ser usada para dar maior clareza a codificação dos programas. Possue apenas um parâmetro que deve ser o endereço da instrução para onde deve ser desviado o programa.

SKIPIN - esta macro-instrução gera um procedimento que verifica se o valor de um registrador está contido em um intervalo dado, fazendo "skip" em caso afirmativo. Possue três parâmetros:

a) o registrador cujo conteúdo deve ser testado;

b) limite inferior do intervalo;

c) limite superior do intervalo.

FALSE - esta macro-instrução gera a instrução "SETZ <registrador>," e deve ser usada para atribuir o valor "false" (Ø) a um registrador. Possue apenas um parâmetro, o registrador ao qual se deseja atribuir o valor "false".

FALSEM - esta macro-instrução gera a instrução "SETZM <endereço>" e pode ser usada para atribuir o valor "false" (Ø) a uma palavra da memória. Possue apenas um parâmetro, o endereço da palavra à qual se deseja atribuir o valor "false".

TRUE - esta macro-instrução gera a instrução "SETO <registrador>," e pode ser usada para atribuir o valor "true" (-1) a um registrador. Possue apenas um parâmetro, o registrador ao qual se deseja atribuir o valor "true".

TRUEM - esta macro-instrução gera a instrução "SETOM <endereço>" e pode ser usada para atribuir o valor "true" (-1) a uma palavra da memória. Possue apenas um parâmetro, o endereço da palavra à qual se deseja atribuir o valor "true".

INC - esta macro-instrução gera a instrução "AOJ registrador " e pode ser usada para incrementar de um o valor de um registrador. Possue um único parâmetro, o registrador que se quer incrementar.

INCM - esta macro-instrução gera a instrução "AOS <endereço>" e pode ser usada para incrementar de um o valor contido em uma palavra da memória. Possue um único parâmetro, o endereço da palavra que se deseja incrementar.

DEC - esta macro-instrução gera a instrução "SOJ <registrador>," e pode ser usada para decrementar de um o valor de um registrador. Possue um único parâmetro, o registrador que se deseja decrementar.

DECM - esta macro-instrução gera a instrução "SOS <endereço>" e deve ser usada para decrementar de um o valor contido em uma palavra da memória. Possue apenas um parâmetro, o endereço da palavra que se deseja decrementar.

CASE - esta macro-instrução gera um código que implementa um comando CASE numérico. Ela possue os seguintes parâmetros:

a) um registrador cujo conteúdo é usado para selecionar o desvio a ser executado;

b) o endereço para onde deve ser feito o desvio, caso o valor contido no registrador esteja fora do intervalo [1,n], onde n é o número de casos (desvios) possíveis;

c) uma lista de endereços "$l_1,l_2,\ldots,l_n$" para onde devem ser feitos os desvios. Essa lista deve ser delimitada a esquerda pelo carater "<" e a direita por ">".

Não há restrição quanto ao número de casos selecionáveis.

6. PROJETO DE "SPOOLING" PARA PLOTTER

A Universidade Federal do Pará dispõe de um plotter Complot modelo DP-8V. A partir do diálogo estabelecido na comunidade, decidiu-se, que esse equipamento deveria ser compartilhado pelos usuários e que a melhor maneira de fazê-lo seria através da utilização de um sistema de "spooling". O fato do sistema de "spooling" para plotter oferecido pelo TOPS-10 não se adaptar as características específicas do equipamento adquirido e o fato do fabricante desse equipamento, não fornecer o software para operação nessa modalidade, motivaram o interesse para o desenvolvimento deste projeto.

6.1 A Ligação do Plotter ao DEC-10

O plotter DP-8V é ligado ao DEC-10 através de seu controlador PTC-5A, por uma linha de comunicação assíncrona.

Do ponto de vista do DEC-10, esse conjunto (controlador/plotter) é visto como um terminal (TTY).

O controlador utiliza um protocolo de comunicação bastante simples, para garantir a qualidade da informação que deve chegar ao plotter. Em primeiro lugar, é necessário estabelecer a diferença entre mensagens que devem ser usadas para controlar o plotter e mensagens que devem ser dirigidas a um terminal, que pode ser ligado ao controlador, juntamente com o plotter. Essa diferença é feita iniciando-se as mensagens de controle do plotter pela seqüência de caracteres ";:".

As mensagens de controle do plotter podem conter um máximo de 480 caracteres e devem terminar pelo carater "_". O controlador faz uma avaliação das mensagens, verifi-

cando a ocorrência de caracteres que não pertençam ao conjunto de caracteres de controle do plotter. Quando uma mensagem não contém erros, o controlador responde com a seqüência "3<LF>". Quando uma mensagem contém erros, ele responde com a seqüência "Ø<LF>" e espera que ela seja retransmitida.

Esse protocolo, usado pelo controlador PTC-5A, foi a causa do impedimento para a utilização do programa SPROUT, fornecido pela DEC para fazer "spooling" para plotter, uma vez que ele não preve um sistema de comunicação como o que foi descrito.

6.2 A Estrutura do Sistema para "SPOOLING"

O sistema de "spooling" foi projetado segundo a configuração apresentada no grafo da figura 6.1.

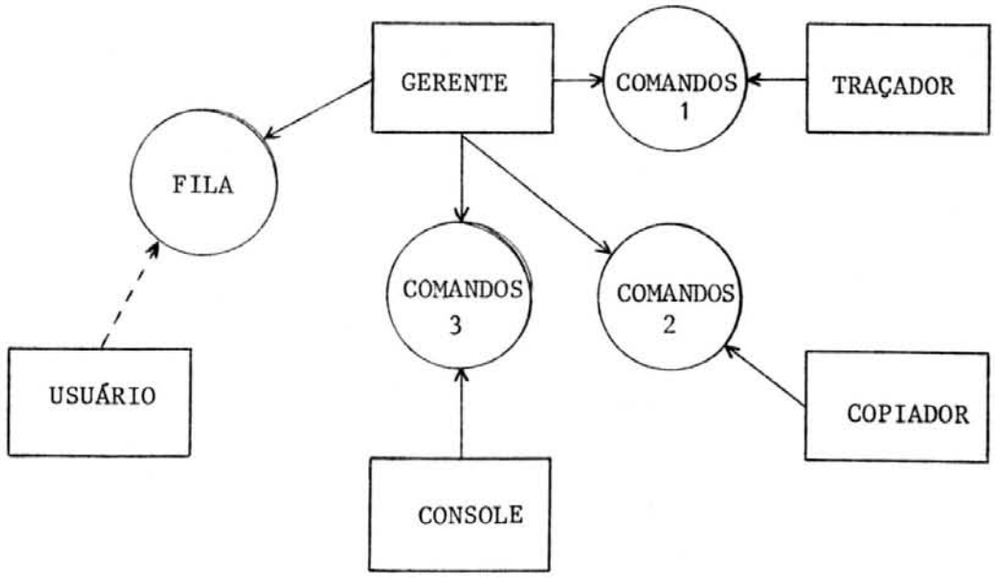


Figura 6.1 - Configuração básica do sistema de "spooling".

Nessa figura, os componentes representados por retângulos são processos e os representados por círculos são monitores. As setas indicam os direitos de acesso dos processos aos monitores.

6.2.1 O Processo Gerente

A função desse processo é distribuir as tarefas dentro do sistema e trocar informações com o operador.

As tarefas podem ter origem em dois tipos de solicitações:

a) as solicitações de serviços dos usuários, que o processo GERENTE seleciona na fila de serviços mantida no monitor FILA;

b) as solicitações do operador que chegam ao processo GERENTE através do monitor COMANDOS 3.

As informações passadas ao operador através do monitor COMANDOS 3, são mensagens que posicionam o operador sobre o andamento das tarefas.

Através do monitor COMANDOS 1, o processo GERENTE envia comandos para o processo TRAÇADOR e tem acesso ao seu estado.

Através do monitor COMANDOS 2, o processo GERENTE tem acesso ao estado do processo COPIADOR e também lhe envia comandos.

O processo GERENTE desempenha sua função, mantendo-se em um ciclo, no qual ele observa os estados dos demais

processos através dos monitores e toma as ações necessárias, em função desses estados.

6.2.2 O Processo Traçador

A função deste processo é ler de um arquivo em disco e transmitir para o controlador PTC-5A, as mensagens de controle que vão produzir o traçado de um gráfico submetido por um usuário.

O processo TRAÇADOR recebe comandos do processo GERENTE através do monitor COMANDOS 1. Os comandos que ele pode receber são os seguintes:

a) DO <especificação de arquivo> - esse comando é depositado no monitor pelo processo GERENTE, quando o processo TRAÇADOR se encontra em WAIT no monitor, refletindo um estado de tarefa concluida, neste caso ele é sinalizado para continuar.

b) STOP - esse comando é depositado no monitor, pelo processo GERENTE, por solicitação do operador.

c) CONTINUE - esse comando é depositado no monitor, pelo processo GERENTE, por solicitação do operador, no caso de antes ter sido depositado um comando STOP.

d) REMOVE - esse comando é depositado no monitor pelo processo GERENTE quando o operador ou o usuário solicita a remoção da tarefa.

Mesmo que o processo TRAÇADOR esteja executando uma tarefa ele vai periodicamente ao monitor COMANDOS 1 verificar se não existem novos comandos que possam modificar a rotina que está sendo executada. Isto é feito sempre entre as

transmissões de mensagens para o plotter. Dessa forma torna-se possível a execução dos comandos STOP e REMOVE apresentados anteriormente.

6.2.3 O Processo Copiador

A função deste processo é criar cópias dos arquivos dos usuários. Isto se faz necessário para impedir que o usuário modifique um arquivo que esteja sendo processado pelo sistema de "spooling". Por isso, cada vez que um usuário submete um arquivo ao sistema de "spooling", o processo COPIADOR é acionado para criar uma cópia do mesmo na área de "spool" do sistema.

O processo COPIADOR recebe comandos do processo GERENTE através do monitor COMANDOS 2 que é do mesmo tipo do monitor COMANDOS 1. Os comandos que ele pode receber são os seguintes:

a) DO <especificação de arquivos> - esse comando é depositado no monitor pelo processo GERENTE, quando o processo COPIADOR se encontra em WAIT no monitor, refletindo um estado de tarefa concluida, neste caso ele é sinalizado para continuar.

b) REMOVE - esse comando é depositado no monitor pelo processo GERENTE quando o operador ou o usuário solicita a remoção da tarefa.

As cópias de arquivos criadas na área de "spool" recebem nomes que são atribuídos pelo processo COPIADOR, nomes esses, que são passados ao processo GERENTE como parte do estado do processo COPIADOR, cada vez que uma tarefa é concluída. Esses nomes são usados pelo processo GERENTE para atualizar as correspondentes solicitações de serviço na

fila mantida no monitor FILA.

Além de realizar as cópias, este processo é responsável pelo restabelecimento dos códigos de proteção dos arquivos dos usuários, que foram modificados por ocasião das solicitações de serviço, para impedir que esses arquivos fossem modificados, até serem copiados para a área de "spool".

6.2.4 O Processo Console

A função deste processo é permitir a comunicação entre o operador do sistema e o processo GERENTE, permitindo o envio de comandos de controle e o recebimento de informações sobre o andamento das tarefas em curso.

A implementação deste processo baseia-se em uma modificação no processo inicial do programa concorrente. Ocorre que o processo inicial, por imposição do método de programação proposto, já é normalmente usado para estabelecer a comunicação entre o operador do terminal controlador do programa e os processos componentes. Entretanto, a iniciativa para o estabelecimento dessa comunicação deve ser sempre de um dos processos e nunca do operador. No caso deste sistema de "spooling", a comunicação entre o processo GERENTE e o operador pode ser iniciada em qualquer um dos dois sentidos.

A modificação introduzida no processo inicial consistiu na substituição da classe a partir da qual se fazia a comunicação por uma outra que atendesse as necessidades particulares deste sistema. Essa nova classe baseia-se em uma rotina, que mantém o processo inicial em um ciclo, no qual ele realiza os seguintes passos:

a) verifica se existe alguma mensagem no "buffer" de entrada do terminal; caso não exista vai para e);

b) interpreta a mensagem do operador e se ela estiver correta vai para d);

c) emite uma mensagem de erro no terminal e vai para e);

d) deposita a mensagem do operador no monitor COMANDOS 3.

e) busca uma mensagem enviada pelo processo GERENTE no monitor COMANDOS 3; se não há mensagens, vai para a);

f) emite a mensagem recebida para o terminal do operador e vai para a).

Os comandos que o operador pode utilizar são os seguintes:

STOP - para temporariamente a execução do gráfico que está sendo traçado;

CONTINUE - reinicia a execução do gráfico interrompida por um comando STOP;

REMOVE: <número de seqüência> - remove a solicitação de serviço identificada pelo número de seqüência, interrompendo definitivamente a tarefa se ela estiver em andamento.

6.2.5 O Processo Usuário

O processo USUÁRIO mostrado na figura 6.1, com acesso ao monitor FILA, tem algumas características particulares, que o diferenciam dos demais processos do sistema.

A função desse processo é possibilitar aos usuários do sistema, introduzir suas solicitações de serviço na fila mantida pelo monitor FILA e, ainda, verificar o andamento das tarefas executadas pelo "spooling". Daí a primeira característica do processo USUÁRIO: ele deve ser disparado pelo usuário e não pelo processo inicial e, como conseqüência, podem existir, em um dado instante, várias, ou nenhuma ocorrência desse processo. Por esse motivo foi necessário modificar a rotina que, no processo inicial, protege o "high segment" do programa contra interferências de outras ocorrências do mesmo programa. Neste caso foi necessário eliminar essa proteção, uma vez que cada processo USUÁRIO disparado precisa acessar o mesmo "high segment" usado pelos demais processos do sistema.

Outra característica do processo USUÁRIO é que, sendo ele executado pelo usuário, torna-se necessário atribuir ao mesmo um privilégio especial, tendo em vista que o usuário não tem direito de acesso ao "high segment" do "spooling", que está sendo executado na conta (p,pn) do operador do sistema. Esse privilégio pode ser atribuído ao processo, incluindo o nome do programa correspondente em uma tabela de processos especiais, mantida no sistema operacional TOPS-10.

O processo USUÁRIO executa suas funções dirigido por comandos fornecidos pelo usuário através do terminal. Esses comandos podem assumir as seguintes formas:

a) <especificação de arquivos>[/COPIES:n] [/AFTER:hh:mm] - a <especificação do arquivo> indica um arquivo que contém um gráfico para ser traçado no plotter. A chave opcional COPIES é usada para indicar o número n de cópias desejadas. A chave opcional AFTER é usada para especificar que o gráfico só deve ser traçado após a hora especificada.

b) <número de seqüência>/KILL - é usado para indicar que a solicitação cujo número de seqüência é fornecido deve ser destruída.

c) o comando nulo é usado quando se quer apenas que seja exibido no terminal o estado da fila de serviço.

No caso dos dois primeiros comandos, o processo USUÁRIO verifica se o usuário tem direito de solicitar os serviços especificados. Ou seja, no primeiro caso é verificado se o arquivo especificado existe e se o usuário tem direito de acesso a esse arquivo. No segundo caso é verificado se a tarefa cujo número de seqüência é referida foi solicitada pelo usuário que está requerendo o cancelamento. Caso a solicitação do usuário esteja correta, o processo USUÁRIO monta um registro descritor da solicitação e vai ao monitor FILA para inserir essa nova solicitação de serviço. A solicitação é inserida na fila levando em conta o tamanho (número de caracteres) do arquivo.

O disparo do processo USUÁRIO é feito com o auxílio de um macro-comando especialmente montado para isso. A forma desse comando é

DO PLOT <comando>

onde <comando> é uma das formas apresentadas anteriormente.

O macro-comando PLOT dispara a execução do processo USUÁRIO e lhe passa o <comando> através do "buffer" de entrada do terminal.

7. CONCLUSÃO

Os testes de aplicação do método de programação apresentado resultaram satisfatórios. Foram implementados alguns programas para resolver problemas clássicos de programação concorrente. Esses programas foram testados no sistema DEC-10 da Universidade Federal do Pará, em ambiente de "time sharing", e com bastante variação na carga do sistema. Não se observou nenhum impacto sobre a performance do sistema, que pudesse ter sido causado pelos programas construídos segundo o método. O apêndice 3 contém as listagens de compilação e execução do problema "Dining Filosophers", de Dijkstra, implementado segundo o método.

Encontra-se em fase de desenvolvimento o sistema de "spooling" para plotter apresentado no capítulo 6. Acredita-se que esse programa constituirá o teste definitivo do método, como ferramenta para a construção de programas que venham a ser usados na produção.

Como uma base para a apresentação de cursos de programação concorrente, acredita-se que o método apresentado seja bastante útil. Esse fato é de particular importância no contexto da UFPa, levando-se em conta que até o momento era muito difícil a apresentação de um curso desse tipo, pela inexistência de ferramentas adequadas para a preparação de exemplos práticos.

Como continuação deste trabalho pretende-se implementar no DEC-10 a linguagem Pascal Concorrente, modificando o código gerado pelo compilador, de maneira a enquadrá-lo no método de programação proposto.

APÊNDICE 1

Listagem do programa tradutor da linguagem de programação descrita no capítulo 5.

```
*
*        PRE-PROCESSADOR DE UMA LINGUAGEM PARA DEFINICAO
*        DAS ESTRUTURAS DE DADOS DE PROGRAMAS CONCORREN-
*        TES ESCRITOS SEGUNDO O METODO DESCRITO NA  DIS-
*        CERTACAO:
*
*        UM METODO PARA CONSTRUCAO DE PROGRAMAS CONCOR-
*        RENTES USANDO OS CONCEITOS DE MONITORES E
*        CLASSES ATRAVES DA LINGUAGEM MACRO-10.
*
*        O PRODUTO DA EXECUCAO DESTE PROGRAMA E UM CON-
*        JUNTO DE ARQUIVOS CONTENDO PROGRAMAS EM MACRO-
*        10 PURA, DEFININDO OS PROCESSOS DO PROGRAMA
*        CONCORRENTE E UM ARQUIVO DE CONTROLE PARA    A
*        COMPILACAO E PREPARACAO DOS MODULOS DE EXECU-
*        CAO DOS PROGRAMAS.
*

         MER001 = 'IDENTIFICADOR ESPERADO'
         MER002 = 'IDENTIFICADOR JA DECLARADO'
         MER003 = 'CONSTANTE INVALIDA'
         MER004 = '; ESPERADO'
         MER005 = '= ESPERADO'
         MER006 = 'TIPO BASICO FALTANDO OU MAL ESCRITO'
         MER007 = ': ESPERADO'
         MER008 = ') ESPERADO'
         MER009 = 'BEGIN ESPERADO'
         MER010 = 'DIMENSAO MAL ESPECIFICADA'
         MER011 = 'IDENTIFICADOR NAO DECLARADO'
         MER012 = '[ ESPERADO'
         MER013 = '] ESPERADO'
         MER014 = 'QUEUE SO E VALIDO EM MONITOR'
         MER015 = 'TIPO ILEGAL'
         MER016 = 'DECLARACAO DE ROTINA MAL ESCRITA'
         MER017 = 'FALTA ESPECIFICACAO DE TIPO NA DECLARACAO DA FUNCAO'
         MER018 = 'ROTINA DO TIPO "PROCEDURE" NAO POSSUE ESPECIFICACAO DE TIPO'
         MER030 = 'IDENTIFICADOR DEVE SER UMA VARIAVEL'
         MER031 = 'VARIAVEL DEVE SER CLASS, MONITOR OU PROCESS'
         MER032 = 'IDENTIFICADOR DEVE SER VARIAVEL OU PARAMETRO DO COMPONENTE'
         MER033 = 'TIPO DO PARAMETRO NAO CONCORDA COM A DEFINICAO PREVIA'
         MER034 = 'PARAMETROS EM EXCESSO'
         MER040 = 'INSTRUCAO INVALIDA'
         MER042 = 'OPERANDO INVALIDO'
         MER045 = 'ESPERAVA , OU )'
         MER050 = 'ARGUMENTO INVALIDO PARA $INCLUDE'
         MER070 = 'STACK_SIZE MAL ESPECIFICADO'
         MER071 = 'STACK_SIZE DEVE SER UM NUMERO'

         &ANCHOR = 1
         LETRA = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_'
         LLETRA = 'abcdefghijklmnopqrstuvwxyz'
         ULETRA = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
         DIGITO = '0123456789'
         MNAME = ANY(LETRA) SPAN(LETRA DIGITO) !
+                ANY(LETRA)
         MNUMB = ANY(DIGITO) SPAN(DIGITO) !
+                ANY(DIGITO)
         OPERATOR = ( '(.' ! '.)' ! ANY('+-*/=><()[]@') )
         PUNCTUAT = ANY(',;:.')
         BREAK = OPERATOR ! PUNCTAUT
         LOUD = ANY(ULETRA) ! ANY(LLETRA) ! ANY(DIGITO)
         ULLL = ANY(ULETRA) ! ANY(LLETRA)
         NAMF = ULLL (LOUD LOUD LOUD LOUD LOUD ! LOUD LOUD LOUD LOUD !
+                    LOUD LOUD LOUD ! LOUD LOUD ! LOUD )
         EXTF = LOUD LOUD LOUD ! LOUD LOUD ! LOUD
         FILENAME = ( (NAMF ! ULLL) ':'
+                   ( (NAMF ! ULLL) '.' EXTF !
+                     (NAMF ! ULLL) '.' !
+                     (NAMF ! ULLL) ) ) !
+                 ( (NAMF ! ULLL) ':' ) !
+                 ( (NAMF ! ULLL) '.' EXTF !
```

```
+                   (NAMF ! ULLL) ',' !
+                   (NAMF ! ULLL) )

        KW = TABLE()
        KW<'CONST'> = 'NOOP'
        KW<'TYPE'> = 'NOOP'
        KW<'VAR'> = 'NOOP'
        KW<'OF'> = 'NOOP'
        KW<'PROCEDURE'> = 'NOOP'
        KW<'FUNCTION'> = 'NOOP'
        KW<'ENTRY'> = 'NOOP'
        KW<'BEGIN'> = 'NOOP'
        KW<'END'> = 'NOOP'
        DATA('SYSTYP(SPN,SVL,STAB,INT)')
        DATA('SPAR(SPTYP,SPVAL)')
        DATA('ARRT(ELTYP,ARRD)')
        DATA('SYMBOL(KIND,VALUE)')
        DATA('CONST(CTYP,CVAL)')
        DATA('VAR(VTYP,VTYPL,VTYPD,VVAL,VLEN,VCOD)')
        DATA('RCUT(RTYP,RPN,RVL,RTAB,RVAL)')
        DATA('PARAM(RPBY,RPTYP,RPVAL,RPDIM)')
        DEFINE('PUSH(A)')
        STACKARQ = 0
        STACKARQ = ARRAY(20)    :(ENDPUSH)
PUSH    STARQ = STARQ + 1
        STACKARQ<STARQ> = A     :(RETURN)
ENDPUSH
        DEFINE('POP(A)')        :(ENDPOP)
POP     $A =
        LE(STARQ,0)             :S(RETURN)
        $A = STACKARQ<STARQ>
        STARQ = STARQ - 1       :(RETURN)
ENDPOP
        DEFINE('GETCH(DUMMY)')          :(ENDGETCH)
GETCH
        ERRLINE = REPLACE(ERRLINE,'%',' ')
        DIFFER(LINE)                    :S(LBLANK)
        LINE = $ENTRADA                 :S(INOK)
        POP('ENTRADA')
        DIFFER(ENTRADA)                 :S(GETCH)F(UEOF)
INOK
        IDENT(ERRORFLAG)                        :S(SKIPERR)
        OUTPUT = BKLIN
        OUTPUT = ERRLINE
        SAIDA = ERRLINE
        I = 1
PRIERR  GT(I,NERROR)                    :S(SKIPERR)
        SAIDA = ERRINLINE<I>
        OUTPUT = ERRINLINE<I>
        I = I + 1                       :(PRIERR)
SKIPERR
        BKLIN = LINE
        LTOK = 0
        ERRORFLAG =
        ERRLINE =
        NERROR = 0
        SAIDA = LINE            :(GETCH)
LBLANK
        LINE SPAN('    ') . P =
        ERRLINE = ERRLINE P
        P =
        IDENT(LINE)                     :S(GETCH)F(RETURN)
ENDGETCH
        DEFINE('GETSYM(DUMMY)')         :(ENDGETSYM)
GETSYM
        GETCH(D)
LNAME
        LINE MNAME . TOKEN =            :F(LNUMB)
        TOKEN = REPLACE(TOKEN,LLETRA,ULETRA)
        IDENT(KW<TOKEN>)                :F(LKW)
        SYM = 'IDENT'                   :(GETSYMRET)
```

```
L..
        SYM = 'RW'                          :(GETSYMRET)
L.:.B
        LINE MNUMB . TOKEN =                :F(LOPR)
        LINE 'B' (NOTANY(LETRA) ! NOTANY(DIGITO)) . XBX = XBX   :F(LNUMB1)
        TOKEN SPAN('01234567')              :S(OCTNUM)
        OUTERROR(MER051)                    :(GETSYM)
OCTNUM  SYM = 'OCTNUM'                      :(GETSYMRET)
LNUMB1
        SYM = 'NUMB'                        :(GETSYMRET)
LOPR
        LINE OPERATOR . TOKEN =             :F(LPUNT)
        TOKEN '(.' = '['
        TOKEN '.)' = ']'
        SYM = 'OPERAT'                      :(GETSYMRET)
LPUNT
        LINE PUNCTUAT . TOKEN =             :F(LSTRING)
        SYM = 'PUNCT'                       :(GETSYMRET)
LSTRING
        LINE "'" ARB . TOKEN "'" =          :F(LCOMENT)
LSTRING1
        LINE "'" ARB . TEMP "'" =           :F(LSTRING2)
        TOKEN = TOKEN "'" TEMP              :(LSTRING1)
LSTRING2
        SYM = 'STRING'                      :(GETSYMRET)
LCOMENT
        LINE '*' =                  :F(LERRO)
        LINE '$LIST' =              :F(UNLIST)
        LIST = '+'                  :(LCOMENT1)
UNLIST  LINE '$NOLIST' =            :F(INCLUDE)
        LIST =                      :(LCOMENT1)
INCLUDE LINE '$INCLUDE' =           :F(LCOMENT1)
        GETCH(D)
        LINE FILENAME . NEWFILE     :S(INCL1)
        OUTERROR(MER050)            :(LCOMENT1)
INCL1   LINE =
        PUSH(ENTRADA)
        KINPUT = KINPUT + 1
        ENTRADA = 'INPUT' KINPUT
        INPUT(ENTRADA,NEWFILE)      :(GETSYM)
LCOMENT1
        LINE ARB '*' =              :S(GETSYM)
        LINE =
        GETCH(D)                    :(LCOMENT1)
LERRO
        LINE ARB (OPERATOR ! PUNCTUAT) . TOKEN =
        LTOK = SIZE(TOKEN)
        TOKEN = 'ERR01'
        SYM = 'ERROR'                       :(RETURN)
GETSYMRET
        LTOK = SIZE(TOKEN)
        ERRLINE = ERRLINE DUPL('%',LTOK)
        :(RETURN)
ENDGETSYM

        DEFINE('CONSTDCL(DUMMY)')                   :(ENDCONSTDCL)
CONSTDCL
        GETSYM(D)
        SYM 'IDENT'                 :F(CERR1)
CIDENT
        IDENT(ITEM(TAB<LEV>,TOKEN))                 :F(CERR2)
        CID = TOKEN
        GETSYM(D)
        TOKEN '='                   :F(CERR5)
        GETSYM(D)
        SYM 'NUMB'                  :F(COCT)
        CNTCOD = CNTCOD + 1
        ITEM(TAB<LEV>,CID) = SYMBOL('CONST',CONST('WORD','CT.' CNTCOD))
        HIGH = 'CT.' CNTCOD ':  EXP                 ^D' TOKEN                   :(CONST1
)
COCT    SYM 'OCTNUM'                :F(CSTR)
        CNTCOD = CNTCOD + 1
```

```
LKW
        SYM = 'RW'
LNUMB                                   :(GETSYMRET)

        LINE NNUMB . TOKEN =            :F(LOPR)
        LINE 'B' (NOTANY(LETRA) ! NOTANY(DIGITO)) . XBX = YBX   :F(LNUMB1)
        TOKEN SPAN('01234567')          :S(OCTNUM)
        OUTERROR(MERO51)                :(GETSYM)
OCTNUM  SYM = 'OCTNUM'                  :(GETSYMRET)
LNUMB1
        SYM = 'NUMB'
LOPR                                    :(GETSYMRET)

        LINE OPERATOR . TOKEN =         :F(LPUNT)
        TOKEN '(' = '['
        TOKEN ')' = ']'
        SYM = 'OPERAT'
LPUNT                                   :(GETSYMRET)

        LINE PUNCTUAT . TOKEN =         :F(LSTRING)
        SYM = 'PUNCT'                   :(GETSYMRET)
LSTRING
        LINE '''' ARB . TOKEN '''' =    :F(LCOMENT)
LSTRING1
        LINE '''' ARB . TEMP '''' =     :F(LSTRING2)
        TOKEN = TOKEN '''' TEMP         :(LSTRING1)
LSTRING2
        SYM = 'STRING'
LCOMENT                                 :(GETSYMRET)

        LINE '*' =              :F(LERRO)
        LINE '$LIST' =          :F(UNLIST)
        LIST = '$'              :(LCOMENT1)
UNLIST  LINE '$NOLIST' =        :F(INCLUDE)
        LIST =                  :(LCOMENT1)
INCLUDE LINE '$INCLUDE' =       :F(LCOMENT1)
        DETCH(D)
        LINE FILENAME . NEWFILE :S(INCL1)
        OUTERROR(MERO50)        :(LCOMENT1)
INCL1   LINE =
        PUSH(ENTRADA)
        NINPUT = NINPUT + 1
        ENTRADA = 'INPUT' NINPUT
        INPUT(ENTRADA,NEWFILE)  :(GETSYM)
LCOMENT1
        LINE ARB '*' =          :S(GETSYM)
        LINE =
        DETCH(D)                :(LCOMENT1)
LERRO
        LINE ARB (OPERATOR ! PUNCTUAT) . TOKEN =
        LTOK = SIZE(TOKEN)
        TOKEN = 'ERRO1'
        SYM = 'ERROR'
GETSYMRET                               :(RETURN)
        LTOK = SIZE(TOKEN)
        ERRLINE = ERRLINE DUPL('X',LTOK)
        :(RETURN)
ENDGETSYM

        DEFINE('CONSTDCL(DUMMY)')               :(ENDCONSTDCL)
CONSTDCL
        GETSYM(D)
        SYM 'IDENT'             :F(CERR1)
CIDENT
        IDENT(ITEM(TAB<LEV>,TOKEN))             :F(CERR2)
        CID = TOKEN
        GETSYM(D)
        TOKEN '='               :F(CERR5)
        GETSYM(D)
        SYM 'NUMB'              :F(COCT)
        CNTCOD = CNTCOD + 1
        ITEM(TAB<LEV>,CID) = SYMBOL('CONST',CONST('WORD','CT.' CNTCOD))
        HIGH = 'CT.' CNTCOD ';' EXP             'D' TOKEN               :(CONST1
)
COCT    SYM 'OCTNUM'            :F(CSTR)
        CNTCOD = CNTCOD + 1
```

```
        OUTERROR(MER001)
        TOKEN = 'NONAME'
TYP06
        IDENT(ITEM(TAB<LEV>,TOKEN))             :S(TYP07)
        OUTERROR(MER002)
TYP07
        ITEM(TAB<LEV>,TOKEN) = SYMBOL('SPARAM',SPAR(,GPN))
        GPN = GPN + 1
        TGPN = TGPN + 1
        TGP<TGPN> = TOKEN
        GETSYM(D)
        TOKEN ','               :S(TSP1)
        TOKEN ';'               :S(TYP08)
        OUTERROR(MER007)
TYP08
        GETSYM(D)
        SYM 'IDENT'             :S(TYP09)
        OUTERROR(MER001)
        TOKEN = 'NONAME'
TYP09
TSP2    GT(TGPN,0)              :F(TSP3)
        SPTYP(VALUE(ITEM(TAB<LEV>,TGP<TGPN>))) = TOKEN
        TGPN = TGPN - 1         :(TSP2)
TSP3    GETSYM(D)
        TOKEN ';'               :S(TSP0)
        TOKEN ')'               :S(TYP10)
        OUTERROR(MER008)
TYP10
        GETSYM(D)
        TOKEN ';'               :S(TSBODY)
        OUTERROR(MER004)
TSBODY
        SPN(VALUE(ITEM(TAB<LEV - 1>,TID))) = GPN        :(TSBODY1)
TSBODY1
        KIND(ITEM(TAB<LEV - 1>,TID)) 'PROCESS'  :F(TSBODY2)
        GETSYM(D);
        TOKEN 'STACK_SIZE'      :F(NOSTACK)
        GETSYM(D);
        TOKEN '='               :S(GSTKSIZ)
        OUTERROR(MER070)        :(NOSTACK)
GSTKSIZ GETSYM(D)
        SYM 'NUMB'              :S(ASTKSIZ)
        OUTERROR(MER071)        :(NOSTACK)
ASTKSIZ PDLSIZ = TOKEN
        GETSYM(D)
        TOKEN ';'               :S(GNEXT)
        OUTERROR(MER004)        :(NOSTACK)
GNEXT   GETSYM(D)
NOSTACK
        PROCESS(TID)            :(TSBODY4)
TSBODY2
        KIND(ITEM(TAB<LEV - 1>,TID)) 'MONITOR'  :F(TSBODY3)
        MONCLA(TID,'HIGH')      :(TSBODY4)
TSBODY3
        MONCLA(TID,ARG)
TSBODY4
        LEV = LEV - 1           :(RETTYPDCL)
TARRDCL
        GETSYM(D)
        TOKEN '['               :S(TAR0)
        OUTERROR(MER012)
TAR0    GETSYM(D)
        SYM 'NUMB'              :F(TAR1)
        AD = TOKEN              :(TAR6)
TAR1    SYM 'IDENT'             :F(TAR3)
        I = LEV
TAR2    GT(I,0)                 :F(TAR5)
        DIFFER(ITEM(TAB<I>,TOKEN))              :F(TAR4)
        KIND(ITEM(TAB<I>,TOKEN)) 'CONST'        :F(TAR3)
        CTYP(VALUE(ITEM(TAB<I>,TOKEN))) 'WORD'  :F(TAR3)
        AD = CVAL(VALUE(ITEM(TAB<I>,TOKEN)))    :(TAR6)
```

```
TAR3    OUTERROR(MER010)
        AD = 1                  :(TAR6)
TAR4    I = I - 1               :(TAR2)
TAR5    OUTERROR(MER011)
        AD = 1                  :(TAR6)
TAR6    GETSYM(D)
        TOKEN ']'               :S(TAR61)
        OUTERROR(MER013)
TAR61   GETSYM(D)
        TOKEN 'OF'              :S(TAR62)
        OUTERROR(MER014)
TAR62   GETSYM(D)
        SYM 'IDENT'             :S(TAR7)
        OUTERROR(MER010)
        TOKEN = 'WORD'
TAR7    TOKEN ( 'WORD' ! 'BOOLEAN' !
+               'BYTE'    ! 'QUEUE'   )         :S(TAR8)
        OUTERROR(MER014)
        TOKEN = 'WORD'
TAR8    DIFFER(TOKEN,'BYTE')            :S(TAR80)
        AD1 = AD / 5
        AD1 = GT(REMDR(AD,5),0) AD1 + 1
        AD = AD1
TAR80
        ITEM(TAB<LEV>,TID) = SYMBOL('ARRAY',ARRT(TOKEN,AD))
        GETSYM(D)
RETTYPDCL
        PDLSIZ = 2000
        TOKEN ';'               :S(TYP14)
        OUTERROR(MER004)
TYP14
        GETSYM(D)
        SYM 'IDENT'             :S(TYPEDCL1)
                                :(RETURN)
ENDTYPEDCL
        DEFINE('VARDCL(TID,PFLEN)GVL')          :(ENDVARDCL)
VARDCL
        GVL = PFLEN
        TGV = ARRAY(20)
        TGVN = 0
VAR1    GETSYM(D)
        SYM 'IDENT'             :S(VAR11)
        OUTERROR(MER001)
        TOKEN = 'NONAME'
VAR11   IDENT(ITEM(TAB<LEV>,TOKEN))             :S(VER02)
        OUTERROR(MER002)
VER02
        ITEM(TAB<LEV>,TOKEN) = SYMBOL('VAR',VAR(,,,,1,))
        TGVN = TGVN + 1
        TGV<TGVN> = TOKEN
        GETSYM(D)
        TOKEN ','               :S(VAR1)
        TOKEN ';'               :S(VER03)
        OUTERROR(MER007)
VER03
        GETSYM(D)
        SYM 'IDENT'             :S(VER04)
        OUTERROR(MER001)
        TOKEN = 'NONAME'
VER04
        TOKEN ( 'WORD' !
+               'BOOLEAN' !
+               'BYTE'    !
+               'QUEUE'         )             :F(VAR2)
        VT = TOKEN
        VTL = 0
        VL = 1
        TOKEN 'QUEUE'           :F(VAR6)
        EQ(LEV,2)               :S(VER09)
        OUTERROR(MER014)
VER09
```

```
        KIND(ITEM(TAB<LEV - 1>,TID)) 'MONITOR'  :S(VAR6)F(VER90)
VER90
        OUTERROR(MER014)                :(VAR6)
VAR2    I = LEV
VAR3    GT(I,0)         :S(VER05)
        OUTERROR(MER011)
        VT = 'WORD'
        VTL = 0
        VL = 1          -  :(VAR6)
VER05
        DIFFER(ITEM(TAB<I>,TOKEN))              :S(VAR4)
        I = I - 1                       :(VAR3)
VAR4    KIND(ITEM(TAB<I>,TOKEN)) 'ARRAY'        :F(VAR5)
        VT = TOKEN
        VTL = I
        VL = ARRD(VALUE(ITEM(TAB<I>,TOKEN)))    :(VAR6)
VAR5    KIND(ITEM(TAB<I>,TOKEN)) ( 'PROCESS' !
+                                  'MONITOR' !
+                                  'CLASS'      ) , VK  :S(VER06)
        OUTERROR(MER015)
        VT = 'WORD'
        VTL = 0
        VL = 1                  :(VAR6)
VER06
        VT = TOKEN
        VTL = I
        VL = SPN(VALUE(ITEM(TAB<I>,TOKEN))) +
+            SVL(VALUE(ITEM(TAB<I>,TOKEN))) + 1
VAR6    GT(TGVN,0)              :F(VAR7)
        VTYP(VALUE(ITEM(TAB<LEV>,TGV<TGVN>))) = VT
        VT ('WORD' ! 'BOOLEAN' ! 'BYTE' ! 'QUEUE')      :S(VAR061)
        VTYPR(VALUE(ITEM(TAB<LEV>,TGV<TGVN>))) = ITEM(TAB<VTL>,VT)
        VTYPL(VALUE(ITEM(TAB<LEV>,TGV<TGVN>))) = VTL
VAR061
        VVAL(VALUE(ITEM(TAB<LEV>,TGV<TGVN>))) = GVL
        VLEN(VALUE(ITEM(TAB<LEV>,TGV<TGVN>))) = VL
        IDENT(VK,'CLASS')       :F(VAR60)
        CLACOD = CLACOD + 1
        VCOM(VALUE(ITEM(TAB<LEV>,TGV<TGVN>))) = CLACOD
VAR60
        GVL = GVL + VL
        TGVN = TGVN - 1         :(VAR6)
VAR7    GETSYM(P)
        TOKEN ';'               :S(VER07)
        OUTERROR(MER004)
VER07
        GETSYM(D)
        TGVN = 0
        SYM 'IDENT'             :S(VAR11)
        SYM 'RW'                :F(VER08)
        VARDCL = GVL            :(RETURN)
VER08
        OUTERROR(MER001)        :(VER07)
ENDVARDCL
        DEFINE('PROCDCL(TID,ARG)')              :(ENDPROCDCL)
PROCDCL
        PF = TOKEN
        PFLEN = 0
        PFENT =
        GETSYM(D)
        TOKEN 'ENTRY'           :F(PF1)
        PFENT = TOKEN
        GETSYM(D)
PF1     SYM 'IDENT'             :S(PF2)
        OUTERROR(MER001)
        TOKEN = 'NONAME'
PF2     PFID = TOKEN
        IDENT(ITEM(TAB<LEV>,PFID))              :S(PF3)
        OUTERROR(MER002)
PF3     ITEM(TAB<LEV>,PFID) = SYMBOL(PF,PFENT,ROUT(0,0,0,TABLE()))
        TAB<LEV + 1> = RTAB(VALUE(ITEM(TAB<LEV>,PFID)))
```

```
        LEV = LEV + 1
PF4     GETSYM(D)
        TOKEN '('                       :S(PF14)
PF400   TOKEN ':'                       :S(PF6)
        TOKEN ';'                       :S(PF5)
        OUTERROR(MER016)                :(PF4)
PF5     FF 'FUNCTION'                   :F(PFBODY)
        OUTERROR(MER017)                :(PFBODY)
PF6     FF 'FUNCTION'                   :S(PF7)
        OUTERROR(MER018)
PF7     GETSYM(D)
        PFLEN = 1
        SYM 'IDENT'                     :S(PF8)
        OUTERROR(MER015)
        TOKEN = 'WORD'  :(PF13)
PF8     TOKEN ('WORD' ! 'BOOLEAN' ! 'BYTE')     :S(PF13)
        I = LEV
PF9     GT(I,0)         :F(PF10)
        DIFFER(ITEM(TAB<I>,TOKEN))      :S(PF11)
        I = I - 1                       :(PF9)
PF10    OUTERROR(MER011)
        TOKEN = 'WORD'  :(PF13)
PF11    KIND(ITEM(TAB<I>,TOKEN)) 'ARRAY'        :F(PF12)
        PFLEN = ARR(VALUE(ITEM(TAB<I>,TOKEN)))
        ELTYP(VALUE(ITEM(TAB<I>,TOKEN))) 'QUEUE'        :F(PF13)
PF12    OUTERROR(MER015)
        TOKEN = 'WORD'  :(PF13)
PF13    RTYP(VALUE(ITEM(TAB<LEV - 1>,PFID))) = TOKEN
        ITEM(TAB<LEV>,PFID) = SYMBOL('VAR',VAR(TOKEN,1,,0,PFLEN,))
        RVL(VALUE(ITEM(TAB<LEV - 1>,PFID))) = PFLEN
        GETSYM(D)
        TOKEN ';'                       :S(PFBODY)
        OUTERROR(MER004)                :(PFBODY)
PF14    LPN = 0
        TPAR = ARRAY(20)
PF15    TLPN = 0
        PARBY = 'VPAR'
        GETSYM(D)
        TOKEN 'VAR'                     :F(PF16)
        PARBY = 'RPAR'
PF150   GETSYM(D)
PF16    SYM 'IDENT'                     :S(PF17)
        OUTERROR(MER001)
        TOKEN = 'NONAME'
PF17    IDENT(ITEM(TAB<LEV>,TOKEN))             :S(PF18)
        OUTERROR(MER002)
PF18    ITEM(TAB<LEV>,TOKEN) = SYMBOL('PARAM',PARAM(PARBY,,LPN,))
        LPN = LPN + 1
        TLPN = TLPN + 1
        TPAR<TLPN> = TOKEN
        GETSYM(D)
        TOKEN ','                       :S(PF150)
        TOKEN ':'                       :S(PF19)
        OUTERROR(MER007)
PF19    PL = 1
        GETSYM(D)
        SYM 'IDENT'                     :S(PF20)
        OUTERROR(MER001)
        TOKEN = 'WORD'
PF20    TOKEN ('WORD' ! 'BOOLEAN' ! 'BYTE')             :S(PF26)
        I = LEV
PF21    GT(I,0)         :S(PF22)
        OUTERROR(MER011)
        TOKEN = 'WORD'  :(PF26)
PF22    DIFFER(ITEM(TAB<I>,TOKEN))              :S(PF23)
        I = I - 1                       :(PF21)
PF23    KIND(ITEM(TAB<I>,TOKEN)) 'ARRAY'                :S(PF24)
        OUTERROR(MER015)
        TOKEN = 'WORD'          :(PF26)
PF24    ELTYP(VALUE(ITEM(TAB<I>,TOKEN)))
+           ('WORD' ! 'BOOLEAN' ! 'BYTE')               :S(PF25)
```

```
        OUTERRGR(MER015)
        TOKEN = 'WORD'          :(PF26)
PF25    PL = ARRD(VALUE(ITEM(TAB<I>,TOKEN)))
PF26
PF260   GT(TLPN,0)              :F(PF27)
        RPTYP(VALUE(ITEM(TAB<LEV>,TPAR<TLPN>))) = TOKEN
        RPDIM(VALUE(ITEM(TAB<LEV>,TPAR<TLPN>))) = PL
        TLPN = TLPN - 1         :(PF260)
PF27    GETSYM(D)
        TOKEN ';'               :S(PF15)
        TOKEN ')'               :S(PF28)
        OUTERROR(MER008)
PF28    RPN(VALUE(ITEM(TAB<LEV - 1>,PFID))) = LPN
        GETSYM(D)               :(PF400)
PFBODY  GETSYM(D)
PFB0    TOKEN 'CONST'           :F(PFB1)
        CONSTDCL(D)             :(PFB0)
PFB1    TOKEN 'TYPE'            :F(PFB2)
        TYPEDCL(D)              :(PFB1)
PFB2    TOKEN 'VAR'             :F(PFB3)
        RVL(VALUE(ITEM(TAB<LEV - 1>,PFID))) = VARDCL(PFID,PFLEN)
PFB3    K = KIND(ITEM(TAB<LEV - 1>,PFID))
        RPN = RPN(VALUE(ITEM(TAB<LEV - 1>,PFID)))
        RVL = RVL(VALUE(ITEM(TAB<LEV - 1>,PFID)))
        RLN = 0
        ROTCOD = ROTCOD + 1
        RVAL(VALUE(ITEM(TAB<LEV - 1>,PFID))) = 'R.' ROTCOD
        $ARG = 'R.' ROTCOD ':'
        RFN = 0
        RFN = IDENT(K,'FUNCTION') RFN + 1
        RFN = IDENT(K,'FUNCTIONENTRY') RFN + 1
        K ('PROCEDUREENTRY' ! 'FUNCTIONENTRY') :S(PFB300)
        $ARG = '        EPROC           ^D'
+       RPN ',^D' RVL ',^D' RLN ',^D' RFN
        BLOCKDCL(PFID,ARG,LEV - 2,SPN(VALUE(ITEM(TAB<LEV - 2>,TID))),RPN)
        $ARG = '        XPROC           ^D'
+       RPN ',^D' RVL ',^D' RFN :(PFB390)
PFB300  $ARG = '        EPROCE          ^D'
+       RPN ',^D' RVL ',^D' RLN ',^D' RFN
        IDENT(KIND(ITEM(TAB<LEV - 2>,TID)),'MONITOR') :F(PFB310)
        $ARG = '        EMONIT'
        BLOCKDCL(PFID,ARG,LEV - 2,SPN(VALUE(ITEM(TAB<LEV - 2>,TID))),RPN)
        $ARG = '        XMONIT'
        $ARG = '        XPROCE          ^D'
+       RPN ',^D' RVL ',^D' RFN :(PFB390)
PFB310  BLOCKDCL(PFID,ARG,LEV - 2,SPN(VALUE(ITEM(TAB<LEV - 2>,TID))),RPN)
        $ARG = '        XPROCE          ^D'
+       RPN ',^D' RVL ',^D' RFN
PFB390
        LEV = LEV - 1
PFB5    TOKEN ';'               :S(RETPROCDCL)
        OUTERROR(MER004)
        GETSYM(D)               :(PFB5)
RETPROCDCL
        GETSYM(D)               :(RETURN)
ENDPROCDCL
        DEFINE('DESCR(N,K,V,L)I')               :(ENDDESCR)
DESCR
        $K =
        I = LEV
DESC1   GT(I,0)         :F(RETURN)
        DIFFER(ITEM(TAB<I>,N))          :S(DESC2)
        I = I - 1               :(DESC1)
DESC2   $K = KIND(ITEM(TAB<I>,N))
        $V = VALUE(ITEM(TAB<I>,N))
        $L = I                          :(RETURN)
ENDDESCR
        DEFINE('GETVAL(OP,VVAL,LEVTYP,SPNTYP,PNROT)TI,DI,LI')
+                       :(ENDGETVAL)
GETVAL  $VVAL = 'NOVAL'
        DESCR(OP,'TI','DI','LI')
```

```
        DIFFER(TI)                          :S(GV100)
        IDENT(REFTYP)                       :S(GV010)
        S = STAB(VALUE(ITEM(TAB<REFLEV>,REFTYP)))
        IDENT(S<OP>)                        :S(GV010)
        REFTYP =
        DI = VALUE(S<OP>)                   :(GV600)
GV010
        CODLAB = CODLAB + 1
        ITEM(TAB<LEV>,OP) = SYMBOL('LABEL','L,' CODLAB)
        $VVAL = 'L,' CODLAB                 :(RETURN)
GV100   LT(LI,LEVTYP)                       :F(GV110)
        TI ('PROCEDUREENTRY' ! 'FUNCTIONENTRY' ! 'PROCEDURE' ! 'FUNCTION')
        :S(GV600)
        TI 'CONST'                          :S(GV400)
        OUTERROR(MER042)                    :(RETURN)
GV110   IDENT(TI,'VAR')                     :F(GV200)
        $VVAL = '<^D' VVAL(DI) + SPNTYP + 1 '>(GB)'
        VTYP(DI) ('WORD' ! 'BOOLEAN' ! 'BYTE' ! 'QUEUE') :S(GV119)
        KIND(VTYPD(DI)) ('CLASS' ! 'MONITOR') :F(GV119)
        REFTYP = VTYP(DI)
        REFLEV = VTYPL(DI)
GV119
        GT(LI,LEVTYP + 1)                   :F(RETURN)
        $VVAL = '<^D' VVAL(DI) + 4 '>(LB)'  :(RETURN)
GV200   IDENT(TI,'PARAM')                   :F(GV300)
        EQ(LI,LEV)                          :S(GV210)
        OUTERROR(MER042)                    :(RETURN)
GV210   $VVAL = '-<^D' PNROT - RPVAL(DI) '>(LB)'        :(RETURN)
GV300   IDENT(TI,'SPARAM')                  :F(GV400)
        EQ(LI,LEVTYP + 1)                   :S(GV310)
        OUTERROR(MER042)                    :(RETURN)
GV310   $VVAL = '<^D' SPVAL(DI) + 1 '>(GB)'
        REFTYP = SPTYP(DI)
        DESCR(SPTYP(DI),'TI','DI','LI')
        REFLEV = LI                         :(RETURN)
GV400   IDENT(TI,'CONST')                   :F(GV500)
        $VVAL = CVAL(DI)                    :(RETURN)
GV500   IDENT(TI,'LABEL')                   :F(GV600)
        $VVAL = DI                          :(RETURN)
GV600   $VVAL = RVAL(DI)                    :(RETURN)
ENDGETVAL
        DEFINE('GETNAM(OP,VNAM,TYPLEV)TI,DI,LI')        :(ENDGETNAM)
GETNAM  $VNAM = 'NONAME'
        DESCR(OP,'TI','DI','LI')
        DIFFER(TI)                          :S(GN100)
        OUTERROR(MER011)                    :(RETURN)
GN100   LT(LI,TYPLEV)                       :F(GN110)
        OUTERROR(MER042)                    :(RETURN)
GN110   IDENT(TI,'VAR')                     :S(GN200)
        OUTERROR(MER042)                    :(RETURN)
GN200   VTYP(DI) ('WORD' ! 'BOOLEAN' ! 'BYTE' ! 'QUEUE')        :F(GN210)
        OUTERROR(MER042)                    :(RETURN)
GN210   KIND(VTYPD(DI)) ('C' , LI 'LASS' ! 'M' , LI 'ONITOR')   :S(GN300)
        OUTERROR(MER042)                    :(RETURN)
GN300   $VNAM = LI ',' VVAL(DI) VCOD(DI)
        REFTYP = VTYP(DI)
        REFLEV = VTYPL(DI)                  :(RETURN)
ENDGETNAM
        DEFINE('BLOCKDCL(TID,ARG,TYPLEV,TYPSPN,ROTPN)ID,NAME,VALUE')    :(ENDBLO
CKDCL)
BLOCKDCL
        INSTR =
B100    GETSYM(D)
B110    IDENT(SYM,'IDENT')              :S(B200)
        IDENT(TOKEN,';')                :F(B111)
        $ARG = INSTR
        INSTR =                         :(B100)
B111
        IDENT(TOKEN,'END')              :S(B120)
        OUTERROR(MER040)                :(B100)
B120    GETSYM(D)
        IDENT(INSTR)                    :S(RETURN)
        $ARG = INSTR                    :(RETURN)
```

```
B200    ID = TOKEN
        GETSYM(D)
        IDENT(TOKEN,';')                    :F(B300)
        IDENT(ITEM(TAB<LEV>,ID))            :S(B210)
        IDENT(KIND(ITEM(TAB<LEV>,ID)),'LABEL')  :S(B201)
        OUTERROR(MER002)                    :(B100)
B201    INSTR = VALUE(ITEM(TAB<LEV>,ID)) ';'                    :(B100)
B210    CODLAB = CODLAB + 1
        ITEM(TAB<LEV>,ID) = SYMBOL('LABEL','L,' CODLAB)
        INSTR = 'L,' CODLAB ';'
B300    IDENT(ID,'INIT')                    :F(B400)                    :(B100)
        INITCHR(ARG)                        :(B110)
B400    INSTR = INSTR ' ' ID '              '
B410    IDENT(TOKEN,';')                    :F(B500)
        $ARG = INSTR
        INSTR =                             :(B100)
B500    IDENT(SYM,'IDENT')                  :F(B600)
        IDENT(TOKEN,'G')                    :F(B630)
        GETSYM(D)
        IDENT(TOKEN,'(')                    :F(B520)
        GETSYM(D)
        IDENT(SYM,'IDENT')                  :S(B510)
        OUTERROR(MER001)
        TOKEN = 'NONAME'
B510    GETNAM(TOKEN,'NAME',TYPLEV)
        INSTR = INSTR NAME
B515    GETSYM(D)
        IDENT(TOKEN,')')                    :S(B800)
        OUTERROR(MER008)                    :(B810)
B520    INSTR = INSTR 'G'                   :(B810)
B530    INSTR = INSTR TOKEN                 :(B800)
B600    IDENT(SYM,'NUMB')                   :F(B610)
        TOKEN = '^D' TOKEN                  :(B530)
B610
        IDENT(SYM,'STRING')                 :S(B530)
        IDENT(TOKEN,'=')                    :F(B630)
        GETSYM(D)
        IDENT(SYM,'IDENT')                  :S(B710)
        OUTERROR(MER001)
        TOKEN = 'NONAME'
B710    GETVAL(TOKEN,'VALUE',TYPLEV,TYPSPC,
        INSTR = INSTR VALUE
B800    GETSYM(D)
B810    IDENT(TOKEN,',')                    :F(B410)
        INSTR = INSTR TOKEN
        GETSYM(D)                           :(B410)
ENDBLOCKDCL
        DEFINE('INITBLOCKDCL(TID,ARG)NB')       :(ENDINITBLOCKDCL)
INITBLOCKDCL
        $ARG = INT(VALUE(ITEM(TAB<LEV - 1>,TID))) ';'
        $ARG = '          EPROCE        ^D0,^D0,^D0,^D0'
        BLOCKDCL(TID,ARG,LEV - 1,SPN(VALUE(ITEM(TAB<LEV - 1>,TID))),0)
        $ARG = '          XPROCE        ^D0,^D0,^D0'      :(RETURN)
ENDINITBLOCKDCL
        DEFINE('XVAR(STABLE,ARG)A,X,B,I,J,K,N')         :(ENDXVAR)
XVAR
        A = CONVERT(STABLE,'ARRAY')
        IDENT(A)        :S(RETURN)
        X = PROTOTYPE(A)
        &ANCHOR = 0
        X ARB , N ','
        &ANCHOR = 1
        B = ARRAY(X)
        J = 0
        I = 1
XVAR1   GT(I,N)         :S(XVAR6)
        IDENT(KIND(A<I,2>),'VAR')       :F(XVAR3)
        K = J
XVAR3   GT(K,0)         :F(XVAR4)
        LT(VVAL(VALUE(A<I,2>)),VVAL(VALUE(B<K,2>)))     :F(XVAR4)
        B<K + 1,1> = B<K,1>
```

```
          B<K + 1,2> = B<K,2>
          K = K - 1                    :(XVAR3)
XVAR4     B<K + 1,1> = A<I,1>
          B<K + 1,2> = A<I,2>
          J = J + 1
XVAR5     I = I + 1                    :(XVAR1)
XVAR6     N = J
          I = 1
XVAR7     GT(I,N)              :S(XVAR110)
          VTYP(VALUE(B<I,2>)) ( 'WORD' ! 'BOOLEAN' !
+                              'BYTE' ! 'QUEUE' )        :F(XVAR8)
          $ARG = '        BLOCK              1'        :(XVAR100)
XVAR8     K = KIND(VTYPD(VALUE(B<I,2>)))
          DIFFER(K,'ARRAY')                  :S(XVAR9)
          J = ARRD(VALUE(VTYPD(VALUE(B<I,2>))))
          $ARG = '        BLOCK          ^D' J             :(XVAR100)
XVAR9     J = SPN(VALUE(VTYPD(VALUE(B<I,2>))))
          A = INT(VALUE(VTYPD(VALUE(B<I,2>))))
          DIFFER(K,'CLASS')          :S(XVAR10)
          $ARG = 'C,' VVAL(VALUE(B<I,2>)) VCOD(VALUE(B<I,2>)) ';'
          $ARG = '        XWD                0,^D' VCOD(VALUE(B<I,2>))
          $ARG = '        BLOCK              ^D' J :(XVAR12)
XVAR10    DIFFER(K,'MONITOR')        :S(XVAR100)
          MONCOD = MONCOD + 1
          $ARG = 'M,' VVAL(VALUE(B<I,2>)) ';'
          $ARG = '        XWD                0,^D' MONCOD
          $ARG = '        BLOCK              ^D' J
XVAR12    XVAR(STAB(VALUE(VTYPD(VALUE(B<I,2>)))),ARG)
XVAR100   I = I + 1                    :(XVAR7)
XVAR110                                :(RETURN)
ENDXVAR
          DEFINE('PROCESS(TID)')               :(ENDPROCESS)
PROCESS
          PRONAM = TID
          TID LEN(4) . PRONAM
          PROCOD = PROCOD + 1
          PRONAM = PRONAM PROCOD
          INT(VALUE(ITEM(TAB<LEV - 1>,TID))) = PRONAM
          OUTPUT('LOW',PRONAM ',MAC')
          LOW = 'TITLE    ' PRONAM
          LOW = 'SEARCH   UUOSYM,CONSTS,MACROS,' HGHNAM
          LOW = 'TWOSEG'
          LOW =
PRS100    TOKEN 'CONST'                  :F(PRS200)
          CONSTDCL(0)                    :(PRS100)
PRS200    TOKEN 'TYPE'                   :F(PRS300)
          TYPEDCL('LOW')                 :(PRS200)
PRS300    TOKEN 'VAR'                    :F(PRS400)
          SVL(VALUE(ITEM(TAB<LEV - 1>,TID))) = VARDCL(TID,0)
PRS400    TOKEN ('PROCEDURE' ! 'FUNCTION')          :F(PRS500)
          PRODCL(TID,'LOW')                    :(PRS400)
PRS500    LOW = ' INIPRO          '' HGHNAM ''',P,' PROCOD
+               ',' SPN(VALUE(ITEM(TAB<LEV - 1>,TID)))
          TOKEN 'BEGIN'                  :S(PRS510)
          OUTERROR(MER009)
PRS510
          BLOCKDCL(TID,'LOW',LEV - 1,SPN(VALUE(ITEM(TAB<LEV - 1>,TID))),0)
          LOW = ' FINPRO'
          LOW = 'P,' PROCOD ';'
          LOW = ' EXP             0'
          LE(SPN(VALUE(ITEM(TAB<LEV - 1>,TID))),0)          :S(PRS600)
          LOW = ' BLOCK           ^D' SPN(VALUE(ITEM(TAB<LEV - 1>,TID)))
PRS600    XVAR(STAB(VALUE(ITEM(TAB<LEV - 1>,TID))),'LOW')
          LOW = 'PDL:     BLOCK          ^D' PDLSIZ
          LOW = 'PDLPTR:  IOWD           ^D' PDLSIZ ',PDL'
          LOW = 'END PROSTA'         :(RETURN)
ENDPROCESS
          DEFINE('MONCLA(TID,ARG)')            :(ENDMONCLA)
MONCLA
          GETSYM(D)
MON100    TOKEN 'CONST'                :F(MON200)
```

```
        CONSTDCL(D)             :(MON100)
MON200  TOKEN 'TYPE'            :F(MON300)
        TYPEDCL(ARG)            :(MON200)
MON300  TOKEN 'VAR'             :F(MON400)
        SVL(VALUE(ITEM(TAB<LEV - 1>,TID))) = VARDCL(TID,0)
MON400  TOKEN ('PROCEDURE' ! 'FUNCTION')        :F(MON500)
        PROCDCL(TID,ARG)                :(MON400)
MON500  INITBLOCKDCL(TID,ARG)   :(RETURN)
ENDMONCLA
        DEFINE('LSTPAR(STABLE,TPAR,SPN)A,N,I,J,K')      :(ENDLSTPAR)
LSTPAR
        A = CONVERT(STABLE,'ARRAY')
        X = PROTOTYPE(A)
        &ANCHOR = 0
        X ARB . N ','
        &ANCHOR = 1
        I = 1
        J = 0
TPAR10  GT(I,N)         :S(RETURN)
        EQ(J,SPN)       :S(RETURN)
        IDENT(KIND(A<I,2>),'SPARAM')    :F(TPAR50)
        K = J
TPAR20  GT(K,0)         :F(TPAR40)
TPAR30  LT(SPVAL(VALUE(A<I,2>)),SPVAL(VALUE(ITEM($TPAR,K,2))))  :F(TPAR40)
        ITEM($TPAR,K + 1,1) = ITEM($TPAR,K,1)
        ITEM($TPAR,K + 1,2) = ITEM($TPAR,K,2>
        K = K - 1               :(TPAR20)
TPAR40  ITEM($TPAR,K + 1,1) = A<I,1>
        ITEM($TPAR,K + 1,2) = A<I,2>
        J = J + 1
TPAR50  I = I + 1       :(TPAR10)
ENDLSTPAR
        DEFINE('INITCLASS(VAR,LPAR,VPN,ARG)I,PNAM')     :(ENDINITCLASS)
INITCLASS
        I = 1
IC100   GT(I,VPN)               :S(IC400)
        IDENT(KIND(LPAR<I,2>),'VAR')    :F(IC200)
        KIND(VTYPD(VALUE(LPAR<I,2>)))
+               ('C' , PNAM 'LASS' ! 'M' , PNAM 'ONITOR' ! 'P' , PNAM 'ROCESS')
        PNAM = PNAM ',' VVAL(VALUE(LPAR<I,2>)) VCOD(VALUE(LPAR<I,2>))
        $ARG = '        ICPAR           ' PNAM ',<C,'
+               VVAL(VAR) VCOD(VAR) '+' I '>,1'  :(IC300)
IC200   $ARG = '        ICPAR           <^D'
+               SPVAL(VALUE(LPAR<I,2>)) '+1>,<C,' VVAL(VAR) VCOD(VAR) '+' I '>,0'
IC300   I = I + 1               :(IC100)
IC400   $ARG = '        INICLA          C,' VVAL(VAR) VCOD(VAR) ','
+               INT(VALUE(VTYPD(VAR))) ''               :(RETURN)
ENDINITCLASS
        DEFINE('INITMONITOR(VAR,LPAR,VPN)I,PNAM')       :(ENDINITMONITOR)
INITMONITOR
        I = 1
IM100   GT(I,VPN)       :S(IM400)
        IDENT(KIND(LPAR<I,2>),'VAR')    :F(IM200)
        KIND(VTYPD(VALUE(LPAR<I,2>)))
+               ('C' , PNAM 'LASS' ! 'M' , PNAM 'ONITOR' ! 'P' , PNAM 'ROCESS')
        PNAM = PNAM ',' VVAL(VALUE(LPAR<I,2>))
        HIGH = '        IMPAR           ' PNAM ',<M,'
+               VVAL(VAR) '+' I '>,1'    :(IM300)
IM200   HIGH = '        IMPAR           <^D'
+               SPVAL(VALUE(LPAR<I,2>)) '+1>,0'
IM300   I = I + 1       :(IM100)
IM400   HIGH = '        INIMON          M,' VVAL(VAR) ','
+               INT(VALUE(VTYPD(VAR))) ''               :(RETURN)
ENDINITMONITOR
        DEFINE('INITPROCESS(VPAR,LPAR,VPN)')    :(ENDINITPROCESS)
INITPROCESS
        PROCONT = PROCONT + 1
        PROTAB<PROCONT,1> = VAR
        PROTAB<PROCONT,2> = LPAR
        PROTAB<PROCONT,3> = VPN :(RETURN)
ENDINITPROCESS
```

```
        :F('INITCMD(ARG)VA..,VARNAM,VARTYP,S.N,V.N,LPAR,XCB') :(ENDINITCMD)
INITCMD
INIT00
        IDENT(SYM,'IDENT')      :S(INT200)
        OUTERROR(MER001)
        TOKEN = 'NONAME'
INT200
        DIFFER(ITEM(TAB<LEV>,TOKEN))    :S(INT210)
        OUTERROR(MER011)
INT210
        IDENT(KIND(ITEM(TAB<LEV>,TOKEN)),'VAR')         :S(INT220)
        OUTERROR(MER030)
INT220
        VARNAM = TOKEN
        VAR = VALUE(ITEM(TAB<LEV>,VARNAM))
        KIND(VTYPD(VAR))
+       ('CLASS' ! 'MONITOR' ! 'PROCESS') . VARTYP      :S(INT230)
        OUTERROR(MER031)
INT230
        SPN = SPN(VALUE(VTYPD(VAR)))
        LE(SPN,0)       :S(INT240)
        XCB = SPN '.2'
        LPAR = ARRAY(XCB)
        LSTPAR(STAB(VALUE(VTYPD(VAR))),'LPAR',SPN)
INT240
        VPN = 0
        GETSYM(D)
        IDENT(TOKEN,'(')        :F(INT400)
INT300
        GETSYM(D)
        IDENT(SYM,'IDENT')      :S(INT310)
        OUTERROR(MER001)
        TOKEN = 'NONAME'
INT310
        VPN = VPN + 1
        DIFFER(ITEM(TAB<LEV>,TOKEN))    :S(INT320)
        OUTERROR(MER011)
INT320
        LE(VPN,SPN)     :S(INT325)
        OUTERROR(MER034)        :(INT350)
INT325
        IDENT(KIND(ITEM(TAB<LEV>,TOKEN)),'VAR') :S(INT330)
        IDENT(KIND(ITEM(TAB<LEV>,TOKEN)),'SPARAM')  :S(INT340)
        OUTERROR(MER032)        :(INT335)
INT330
        IDENT(VTYP(VALUE(ITEM(TAB<LEV>,TOKEN))),
+             SPTYP(VALUE(LPAR<VPN,2>)))        :S(INT335)
        OUTERROR(MER033)
INT335
        LPAR<VPN,1> = TOKEN
        LPAR<VPN,2> = ITEM(TAB<LEV>,TOKEN)      :(INT350)
INT340
        IDENT(SPTYP(VALUE(ITEM(TAB<LEV>,TOKEN))),
+             SPTYP(VALUE(LPAR<VPN,2>)))        :S(INT335)
        OUTERROR(MER033)        :(INT335)
INT350
        GETSYM(D)
        IDENT(TOKEN,',')        :S(INT300)
        IDENT(TOKEN,')')        :S(INT400)
        OUTERROR(MER035)
INT400
        IDENT(VARTYP,'CLASS')   :F(INT410)
        INITCLASS(VAR,LPAR,VPN,ARG)     :(INT500)
INT410
        IDENT(VARTYP,'MONITOR') :F(INT420)
        INITMONITOR(VAR,LPAR,VPN)       :(INT500)
INT420
        INITPROCESS(VAR,LPAR,VPN)
INT500
        EQ(VPN,0)       :S(INT510)
        GETSYM(D)
```

```
INIT510
        IDENT(TOKEN,',')        :F(INIT520)
        GETSYM(D)               :(INIT100)
INIT520 IDENT(TOKEN,'#')        :F(RETURN)
        GETSYM(D)       :(RETURN)
ENDINITCMD
        DEFINE('MAKEPROCESSTAB(DUMMY)I,J,K,PRONAM,LPAR,VPN')
        PROGTAB = ARRAY(10)
        PROCONT = 0             :(ENDMAKEPROCESSTAB)
MAKEPROCESSTAB
        I = 1
P100    GT(I,PROCONT)   :S(P200)
        PRONAM = INT(VALUE(VTYPE(PROGTAB<I,1>)))
P110    LT(SIZE(PRONAM),6)      :F(P120)
        PRONAM = PRONAM ' '     :(P110)
P120
        K = 1
P121    GT(K,PROCONT)           :F(P122)
        PROCONT = K
        PROGTAB<K> = PRONAM     :(P123)
P122    IDENT(PRONAM,PROGTAB<K>)        :S(P123)
        K = K + 1               :(P121)
P123
        HIGH = '        ITBNAM          '' PRONAM ''," I
        I = I + 1       :(P100)
P200
        HIGH = '        ITBJP           "D' PROCONT
        K = 0
        I = 1
P300    GT(I,PROCONT)   :S(RETURN)
        LPAR = PROTAB<I,2>
        VPN = PROTAB<I,3>
        HIGH = '        ITBPAR          <PROPAR+"D' K '>'
        J = 1
P310    GT(J,VPN)       :S(P320)
        HIGH = '        IPPAR           N,' VVAL(VALUE(LPAR<J,2>))
        J = J + 1
        K = K + 1       :(P310)
P320    I = I + 1       :(P300)
ENDMAKEPROCESSTAB
        DEFINE('FINAL(D)I')
        FLAGDEERROS =           :(ENDFINAL)
FINAL
        OUTPUT = '$' MICFILE
        OUTPUT('LOW',MICFILE)
        DIFFER(FLAGDEERROS)             :S(DELETE)
        LOW = 'TE ' HGHNAM '.MAC'
        LOW = 'IUNIVERSAL       ' HGHNAM
        LOW = '<ESC>EX<ESC><ESC>'
        LOW = 'COMP/COMP ' HGHNAM '.MAC'
        LOW = 'TE ' HGHNAM '.MAC'
        LOW = 'FSUNIVERSAL<ESC>TITLE<ESC>EX<ESC><ESC>'
        LOW = 'LOAD/COMP"A ' HGHNAM '.MAC'
        LOW = 'SSAVE'
        I = 1
FINAL1  GT(I,PROCONT)           :S(FINAL2)
        LOW = 'LOAD/COMP"A ' PROGTAB<I> '.MAC'
        LOW = 'SAVE'
        I = I + 1       :(FINAL1)
FINAL2  LOW = 'TE ' HGHNAM '.MAC'
        LOW = 'K<ESC>EX<ESC><ESC>'
        LOW = 'DEL ' HGHNAM '.BAK'      :(RETURN)
DELETE  LOW = 'DEL ' HGHNAM '.MAC'
        I = 1
DEL1    GT(I,PROCONT)           :S(RETURN)
        LOW = 'DEL ' PROGTAB<I> '.MAC'
        I = I + 1       :(DEL1)
ENDFINAL
        DEFINE('OUTERROR(MERR)')        :(ENDOUTERROR)
OUTERROR
        ERRORFLAG = 'TRUE'
```

```
        FLAGDEERROS = 'TRUE'
        ERRLINE = REPLACE(ERRLINE,'%','%')
        GT(NERROR,MAXERROR)             :S(ERRLIMIT)
        NERROR = NERROR + 1
        ERRINLINE<NERROR> = MERR
        LT(NERROR,MAXERROR - 1)         :S(ERRLIMIT)
        NERROR = NERROR + 1
        ERRINLINE<NERROR> = 'LIMITE DE ERROS EXCEDIDO NA LINHA'
ERRLIMIT
                        :(RETURN)
ENDOUTERROR

        NERROR = 0
        MAXERROR = 5
        ERRINLINE = ARRAY(5)
        MONCOD = 0
        CLACOD = 0
        PROCOD = 0
        BATCOD = 0
        IDENTCNT = 0
        PROCNT = 0
        ERRTAB = ARRAY('10,3')
        TAB = ARRAY('0:10')
        RAIZ = SYMBOL('INITPROCESS',SYSTYP(0,0,TABLE()))

        LEV = 1
        TAB<LEV> = STAB(VALUE(RAIZ))
        NEMB = -1
        COMAND = INPUT          :S(COM01)
        OUTPUT = 'FALTOU O PARAMETRO <ESP. FONTE>I/LISTI:<ESP. LISTAGEM>]]'
:(END)
COM01   COMAND FILENAME . NEWFILE = :S(COM02)
        OUTPUT = '<ESP. FONTE> MAL ESCRITA'     :(END)
COM02   COMAND '/LIST' =                :F(COM05)
        COMAND ':' FILENAME . SAIDAF    :S(COM04)
        IDENT(COMAND)   :S(COM03)
        OUTPUT = '<ESP. LISTAGEM> MAL ESCRITA'  :(END)
COM03   OUTPUT('SAIDA',':LPT:')
        LIST = '+'                      :(INICIO)
COM04   OUTPUT('SAIDA',SAIDAF)
        LIST = '+'                      :(INICIO)
COM05   IDENT(COMAND)                   :S(COM06)
        OUTPUT = 'PARAMETRO DEVE SER: <ESP. FONTE>I/LISTI:<ESP. LISTAGEM>]]'
:(END)
COM06   SAIDA =
        LIST =
INICIO  SAIDA = 'PRE-PROCESSADOR PARA CONSTRUCAO DE PROGRAMAS CONCORRENTES
-       V1.0/JANEIRO-83'
        SAIDA =
        OUTPUT =
        OUTPUT = 'PRE-PROCESSADOR PARA CONSTRUCAO DE PROGRAMAS CONCORRENTES
-       V1.0/JANEIRO-83'
        KINPUT = 1
        ENTRADA = 'INPUT' KINPUT
        INPUT(ENTRADA,NEWFILE)
        NEWFILE (NAME ! ULLL) . MICFILE
        MICFILE = MICFILE '.MIC'
        HGHNAM = 'HGHSEG'
        GETSYM(D)
        IDENT(TOKEN,'PROGRAM')  :F(PRG200)
        GETSYM(D)
        IDENT(SYM,'IDENT')      :S(PRG110)
        OUTERROR(MER001)        :(PRG200)
PRG110  HGHNAM = TOKEN
        TOKEN LEN(6) . HGHNAM
        GETSYM(D)
        IDENT(TOKEN,';')        :S(PRG120)
        OUTERROR(MER004)        :(PRG200)
PRG120  GETSYM(D)
PRG200  OUTPUT('HIGH',HGHNAM '.MAC')
        HIGH = '        SEARCH          UUOSYM,CONSYS,MACROS'
        HIGH = '        TWOSEG'
        HIGH = '        RELOC           400000'
        HIGH = '        INIHGH          '' HGHNAM '^'
```

```
START1
        TOKEN 'CONST'                   :F(START2)
        CONSTDCL(0)                     :(START1)
START2
        TOKEN 'TYPE'                    :F(START3)
        TYPEDCL('HIGH')                 :(START2)
START3
        TOKEN 'VAR'                     :F(START4)
        VARDCL('INITPROCESS',0)         :(START3)
START4
        XVAR(STAR(VALUE(RAIZ)),'HIGH')
        TOKEN 'BEGIN'                   :F(START5)
        HIGH = '        INILOW          '' HOHNAM '''
        BLOCKDCL('INITPROCESS','HIGH',LEV,0,0)          :(START6)
START5
        OUTERROR(HER009)
        GETSYM(0)                       :(START4)
START6
        HIGH = '        FINLOW          '' HOHNAM '''
        MAKEPROCESSTAB(0)
        HIGH = 'END START'
        FINAL(0)        :(END)
UEGT
        OUTPUT =
        OUTPUT =
        OUTPUT = '? FIM INESPERADO DO ARQUIVO FONTE'    :(END)
END
```

APÊNDICE 2

Listagem das macro-instruções constantes da biblioteca MACROS, apresentadas no capítulo 5.

```
universal macros
SEARCH UUSYM:CONSTS
define eproce(np,vl,ln,fn)
;
;       Enter PROCedure Entry - deve ser a primeira macro instrucao em
; uma procedure ou function entry. Tem o objetivo de atualizar o  con-
; texto de modo que ele reflita o componente de sistema do qual    faz
; parte a rotina que esta sendo iniciada.
;
;       np - numero de parametros da rotina
;       vl - tamanho da area de variaveis locais
;       ln - numero da linha fonte onde inicia a definicao da rotina
;       fn - variavel logica que indica se a rotina e function (1)
;            ou procedure (0)
;
;       Esta macro instrucao e tambem utilizada nos blocos de  inicia-
; lizacao de classes e monitores.
;
<XALL
   IFLE NP-5,<
                MOVE            A1,(P)
                HRLI            A2,T1
                HRRI            A2,(P)
                BLT             A2,NP(P)
                ADJSP           P,NP+1
                MOVEM           A1,(P)
   >
                PUSH            P,GB
                PUSH            P,LB
                MOVEI           LB,-2(P)
                MOVE            GB,-<NP+1>(LB)
                MOVEI           A1,LN
                PUSH            P,A1
   IFG VL,<
                MOVSI           A1,-VL
                HRRI            A1,1(P)
                SETZM           (A1)
                BLT             A1,,-1
                ADJSP           P,VL
   >
>
define xproce(np,vl,fn)
;
;       eXit PROCedure Entry - deve ser a ultima macro instrucao    em
; uma procedure ou function entry. Tem o objetivo de restaurar o  con-
; texto para que ele passe a refletir o componente de sistema a partir
; do qual foi chamada a rotina e tambem para fazer o retorno.
;
;       np - numero de parametros da rotina
;       vl - tamanho da area de variaveis locais
;       fn - variavel logica que indica se a rotina e function (1)
;            ou procedure (0)
;
;       Esta macro instrucao e tambem utilizada para finalizar os blo-
; cos de inicializacao de classes e monitores.
;
<XALL
   IFE FN-1,<
                MOVE            T1,4(LB)
   >
   IFE FN-2,<
                MOVEI           T1,4(LB)
   >
                MOVE            GB,1(LB)
                MOVE            LB,2(LB)
                ADJSP           P,-<NP+VL+5>
                JRST            @<NP+2>(P)
>
define emonit
;
;       Enter MONITor - esta macro instrucao deve ser usada no  inicio
```

```
; de uma procedure ou function entry de um monitor, antes de qualquer
; referencia aos dados privados do monitor. Atraves dela e chamada da
; primitiva ENTer MONitor que garante o acesso exclusivo as estruturas
; privadas de um monitor.
;
<XALL
                MOVE            T1,(GB)
                PUSHJ           P,ENTMON
>
define xmonit
;
;       eXit MONITor - esta macro instrucao deve ser usada no final  de
; uma procedure ou function entry de um monitor, antes de retornar  o
; controle. Atraves dela e chamada a primitiva eXIT MONitor que libera
; o acesso as estruturas privadas de um monitor garantindo a    outros
; processos o direito de acesso a esse monitor.
;
<XALL
                MOVE            T1,(GB)
                PUSHJ           P,XITMON
>
define eproc(np,vl,ln,fn)
;
;       Enter PROCedure - esta macro instrucao deve ser a primeira  em
; uma procedure ou function (que nao seja entry). Tem o objetivo de a-
; justar o contexto apenas do ponto de vista local.
;
;       NP - numero de parametros da rotina
;       VL - tamanho da area de variaveis locais
;       LN - numero da linha fonte onde inicia a definicao da rotina
;       FN - variavel logica que indica se a rotina e function (1)
;            ou procedure (0)
;
<XALL
   IFLE NP-6,<
                MOVE            A1,(P)
                HPLI            A2,T1
                HRRI            A2,1(P)
                BLT             A2,NP(P)
                ADJSP           P,NP+1
                MOVEM           A1,(P)
   >
                PUSH            P,QB
                PUSH            P,LB
                MOVEI           LB,-2(P)
                MOVEI           A1,LN
                PUSH            P,A1
   IFG VL,<
                MOVSI           A1,-VL
                HRRI            A1,1(P)
                SETZM           (A1)
                AOBJN           A1,.-1
                ADJSP           P,VL
   >
>
define xproc(np,vl,fn)
;
;       eXit PROCedure - esta macro instrucao deve ser a ultima     em
; uma procedure ou function (que nao seja entry). Tem o objetivo    de
; restaurar o contexto previo e retornar o controle.
;
;       NP - numero de parametros da rotina
;       VL - tamanho da area de variaveis locais
;       FN - variavel logica que indica se a rotina e function (1)
;            ou procedure (0)
;
<XALL
   IFE FN-1,<
                MOVE            T1,4(LB)
   >
   IFE FN-2,<
```

```
                MOVEI           T1,4(L)
    >
                MOVE            GR,1(LE)
                MOVE            LP,2(LE)
                ADJSP           P,-<NP+4L+5>
                JRST            @<NP+2>(P)
>
define inpar(p,m,f)
;
;       INPAR - deve ser usada no processo inicial para inicializar os
; parametros de um monitor.
;
;       p - endereco do parametro atual
;       m - endereco onde deve ser introduzido o parametro atual
;       f - variavel logica que indica se o parametro e uma  variavel
;           do componente que inicializa o monitor (1) ou se o parame-
;           e um dos parametros do proprio componente (0 - nao  devera
;           ocorrer)
;
<XALL
  IFN F,<
                MOVEI           A1,P
  >
  IFE F,<
                MOVE            A1,P(GB)
  >
                MOVEM           A1,M
>
define inikon(m,ib)
;
;       INIKON - deve ser usada no processo inicial para  inicializar
; um monitor.
;
;       m - nome do monitor
;       ib - nome do bloco de inicializacao do monitor
;
<XALL
                RELOC
                MOVEI           T1,M
                PUSHJ           P,ib
                RELOC
>
define i parm(p,c,f)
;
;       INPAR - deve ser usada para inicializar os parametros de  uma
; classe.
;       p - endereco do parametro atual
;       c - endereco onde deve ser introduzido o parametro atual
;       f - variavel logica que indica se o parametro e uma  variavel
;           do componente que inicializa a classe (1) ou um  parametro
;           do proprio componente (0)
;
<XALL
  IFN F,<
                MOVEI           A1,P
  >
  IFE F,<
                MOVE            A1,P(GB)
  >
                MOVEM           A1,C
>
define inicla(c,ib)
;
;       INICLA - deve ser usada para inicializar uma classe.
;
;       c - nome da classe
;       ib - nome do bloco de inicializacao
;
<XALL
                MOVEI           T1,C
                PUSHJ           P,IB
```

```
>
define itbnam(p,np)
;
;       ITBNAM - deve ser usada pelo processo inicial para inicializar
; a tabela de processos, introduzindo nessa tabela os nomes dos  pro-
; gramas correspondentes a cada processo.
;
;       p - nome do programa correspondente ao processo
;       np - numero de ordem do processo
;
<XALL
   IFE NP-1,<
                RELOC
PRONAM:
   >
                ASCIZ           P
>
define itbjp(np)
;
;       ITBJP - deve ser usada pelo processo inicial para  inicializar
; a tabela de processos, criando espaco para os numeros dos jobs     e
; iniciando a construcao da tabela de apontadores de parametros     de
; processos.
;
;       np - numero de processos
;
<XALL
                NOPRO=NP
PROJOB:         BLOCK           NP
PARPRO:
                .
                RELOC
FROPAR:
>
define itbpar(ptr)
;
;       ITBPAR - deve ser usada pelo processo inicial para inicializar
; a tabela de processos, introduzindo os apontadores das areas de  pa-
; rametros dos processos na tabela correspondente.
;
;       ptr - apontador para a area de parametros de um processo
;
<XALL
                RELOC
                EXP             PTR
                RELOC
>
define ippar(p)
;
;       IPPAR - deve ser usada pelo processo inicial para  inicializar
; a area de parametros de um processo.
;
;       p - nome do parametro
;
<XALL
                EXP             P
>
DEFINE INITHGH(HGHNAM)
<XALL
TRGOUT: MOVSI   A1,-4
        MOVE    A3,1(A1)
        MOVEI   A4,14
        SETZ    A2,
        POTC    A2,3
        ADDI    A2,"0"
        OUTCHR  A2
        SOJG    A4,.-4
        MOVEI   A2," "
        OUTCHR  A2
        AOBJN   A1,TRGOUT+1
        POPJ    P,
MONREC: XWD             1,5
```

```
        XWD             0,1
        XWD             200000,-2
        EXP             SB2
        EXP             0
ENTMON: PRLI            T3,MONREC
        HRRI            T3,1(P)
        BLT             T3,5(P)
        HRRM            T1,4(P)
        HRLI            T3,.ENQBL
        HRRI            T3,1(P)
        ENQ.            T3,
          JRST          [ OUTSTR        [ ASCIZ /?ENTMON/]
                          PUSHJ         P,TRGOUT
                          EXIT
                        ]
        POPJ            P,
XITMON: MOVE            T4,3(LB)
        CAIE            T4,0
        JRST            XITM2
        HRLI            T3,MONREC
        HRRI            T3,1(P)
        BLT             T3,5(P)
        HRRM            T1,4(P)
        HRLI            T3,.DEQDR
        HRRI            T3,1(P)
        DEQ.            T3,
          JRST          [ OUTSTR        [ ASCIZ /?XITMON/]
                          PUSHJ         P,TRGOUT
                          EXIT
                        ]
        POPJ            P,
XITM2:  HRRZ            T4,
          JRST          [ OUTSTR        [ ASCIZ /?WAKE/]
                          PUSHJ         P,TRGOUT
                          EXIT
                        ]
        POPJ            P,
WAIT:   SKIPE           (T2)
          JRST          [ OUTSTR        [ ASCIZ /?WAIT EM FILA OCUPADA/]
                          PUSHJ         P,TRGOUT
                          EXIT
                        ]
        PJOB            T3,
        MOVEM           T3,(T2)
        MOVNI           T3,1
        WAKE            T3,
          JRST          [ OUTSTR        [ ASCIZ /?WAKE/]
                          PUSHJ         P,TRGOUT
                          EXIT
                        ]
        HRLZI           T3,(HB.RWP!HB.RWT)
        HIBER           T3,
          JRST          [ OUTSTR        [ ASCIZ /?HIPER/]
                          PUSHJ         P,TRGOUT
                          EXIT
                        ]
        PUSHJ           P,XITMON
        HRLZI           T3,(HB.RWP!HB.RWT)
        HIBER           T3,
          JRST          [ OUTSTR        [ ASCIZ /?HIPER/]
                          PUSHJ         P,TRGOUT
                          EXIT
                        ]
        MOVE            T3,(T2)
        SETZM           (T2)
        MOVEM           T3,3(LB)
        POPJ            P,
```

```
SIGNAL:
        SKIPG           (T2)
        POPJ            P,
        MOVNI           T3,1
        WAKE            T3,
          JRST          [ OUTSTR        [ ASCIZ /?WAKE/]
                          PUSHJ         P,TRGOUT
                          EXIT
                        ]
        HRLZI           T3,(HB.RWP!HB.RWT)
        HIBER           T3,
          JRST          [ OUTSTR        [ ASCIZ /?HIBER/]
                          PUSHJ         P,TRGOUT
                          EXIT
                        ]
        MOVE            T3,(T2)
        PJOB            T4,
        MOVEM           T4,(T2)
        WAKE            T3,
          JRST          [ OUTSTR        [ ASCIZ /?WAKE/]
                          PUSHJ         P,TRGOUT
                          EXIT
                        ]
        HRLZI           T3,(HB.RWP!HB.RWT)
        HIBER           T3,
          JRST          [ OUTSTR        [ ASCIZ /?HIBER/]
                          PUSHJ         P,TRGOUT
                          EXIT
                        ]
        POPJ            P,
REQ.0:
        EPROCE          ^D1,^D0,^D0,^D0
        EMONIT
        PJOB            A1,
        HLRZ            A1,TRJOB(A1)
        MOVEI           A2,<^D131>(GB)
        ADD             A2,-1(A1)
        SKIPN           (A2)
        JRST            REQ.01
        MOVE            T1,(GB)
        MOVEI           T2,<^D181>(GB)
        ADDI            T2,-1(A1)
        PUSHJ           P,WAIT
REQ.01: SETOM           (A2)
        MOVEI           A2,-1(A1)
        IMULI           A2,^D16
        ADDI            A2,<^D1>(GB)
        MOVE            A3,-<^D1>(LB)
        COPYW           A3,A2,^D16,A4
        ADS             <^D192>(GB)
        MOVE            A4,2-<^D1>(LB)
        LSH             A4,-^D29
        CAIE            A4,'?'
        JRST            REQ.02
        MOVE            T1,(GB)
        MOVEI           T2,<^D171>(GB)
        ADDI            T2,-1(A1)
        PUSHJ           P,WAIT
        COPYW           A2,A3,^D16,A4
REQ.02: XMONIT
        XPROCE          ^D1,^D0,^D0
ATN.0:
        EPROCE          ^D3,^D0,^D0,^D0
        EMONIT
        MOVE            A1,<^D191>(GB)
        CAIN            A1,^D10
        SETZ            A1,
ATN.01: ADJ             A1,
        MOVEI           A2,<^D161>(GB)
        ADDI            A2,-1(A1)
        SKIPN           (A2)
```

```
            CAMN        A1,<^D191>(GB)
            JRST        ATN.02
            CAIL        A1,^D10
            SETZ        A1,
            JRST        ATN.01
ATN.02: SETZM       @-<^D1>(LB)
            SKIPN       (A2)
            JRST        ATN.03
            MOVEM       A1,<^D191>(GB)
            SETZM       (A2)
            MOVEM       A1,@-<^D1>(LB)
            MOVEI       A2,-1(A1)
            IMULI       A2,^D16
            ADDI        A2,<^D1>(GB)
            MOVE        A3,-<^D3>(LB)
            COPYW       A2,A3,^D16,A4
            SOS         <^D192>(GB)
            MOVE        T1,(GB)
            MOVEI       T2,<^D181>(GB)
            ADDI        T2,-1(A1)
            PUSHJ       P,SIGNAL
ATN.03: MOVE        A4,<^D192>(GB)
            MOVEM       A4,@-<^D2>(LB)
            XMONIT
            XPROCE      ^D3,^D0,^D0
RPY.0:
            EPROCE      ^D1,^D0,^D0,^D0
            EMONIT
            MOVE        A1,<^D191>(GB)
            MOVE        A2,-<^D1>(LB)
            MOVEI       A3,-1(A1)
            IMULI       A3,^D16
            ADDI        A3,<^D1>(GB)
            COPYW       A2,A3,^D16,A4
            MOVE        T1,(GB)
            MOVEI       T2,<^D171>(GB)
            ADDI        T2,-1(A1)
            PUSHJ       P,SIGNAL
            XMONIT
            XPROCE      ^D1,^D0,^D0
CTY.I:
            EPROCE      ^D0,^D0,^D0,^D0
            MOVEI       A1,^D10
            HRRI        A1,<^D161>(GB)
            SETZM       (A1)
            ADDUN       A1,,-1
            MOVEI       A1,^D10
            MOVEM       A1,<^D191>(GB)
            SETZM       <^D192>(GB)
            XPROCE      ^D0,^D0,^D0
FIN.0:  ASCIZ   /FINALIZANDO/
FIN.1:  ASCIZ   /INICIANDO/
SGF.00: ASCPTR      A2,<^D3>(GB)
            ILDB        A3,A1
            IDPB        A3,A2
            CAIE        A3,0
            JRST        ,-3
            POPJ        P,
SGF.0:
            EPROCE      ^D0,^D0,^D0,^D0
            EMONIT
            ASCPTR      A1,FIN.0
            PUSHJ       P,SGF.00
            MOVE        T1,<^D1>(GB)
            MOVEI       T2,<^D3>(GB)
            PUSHJ       P,REQ.0
            SOS         <^D2>(GB)
            XMONIT
            XPROCE      ^D0,^D0,^D0
SGF.1:
            EPROCE      ^D0,^D1,^D0,^D1
```

```
        EMONIT
        SKIPN           <^D2>(GB)
        SETOM           <^D4>(LB)
        XMONIT
        XPROCE          ^D0,^D1,^D1
SGF.2:
        EPROCE          ^D0,^D1,^D0,^D1
        EMONIT
        MOVE            A1,<^D2>(GB)
        CAIN            A1,NOPRO
        JRST            .+5
        MOVE            T1,(GB)
        MOVEI           T2,<^D19>(GB)
        PUSHJ           P,WAIT
        JRST            .-6
        SETOM           <^D4>(LB)
        XMONIT
        XPROCE          ^D0,^D1,^D1
SGF.3:
        EPROCE          ^D0,^D0,^D0,^D0
        EMONIT
        ASCPTR          A1,FIN.1
        PUSHJ           P,SGF.00
        MOVE            T1,<^D1>(GB)
        MOVEI           T2,<^D3>(GB)
        PUSHJ           P,REQ.0
        AOS             <^D2>(GB)
        MOVE            T1,(GB)
        MOVEI           T2,<^D19>(GB)
        PUSHJ           P,SIGNAL
        XMONIT
        XPROCE          ^D0,^D0,^D0
CTF.1:
        EPROCE          ^D0,^D0,^D0,^D0
        SETZM           <^D2>(GB)
        XPROCE          ^D0,^D0,^D0
CTF.0:
        XWD             0,^D1001
        BLOCK           ^D0
        BLOCK           ^D160
        BLOCK           ^D10
        BLOCK           ^D10
        BLOCK           ^D
        BLOCK           1
        BLOCK           1
CTF.V:
        XWD             0,^D1002
        BLOCK           ^D1
        BLOCK           1
        BLOCK           ^D16
        BLOCK           1
TBJOB:  BLOCK           MAXJBS+1
        LIT
>
define inilow(hghnam)
;
;       INItialize LOWsegment - gera o codigo de inicializacao do
; processo inicial.
;
;       hghnam - e o nome do arquivo que contem o processo inicial
;                e o high segment do programa.
;
<XALL
                RELOC
COUTST: OUTSTR          0
TTW.0:
        EPROCE          ^D2,^D0,^D0,^D0
        ASCPTR          A1,<^D1>(GB)
        MOVE            A2,-<^D1>(LB)
TTW.01: ILDB            A3,-<^D2>(LB)
        IDPB            A3,A1
```

```
            CAIE        A3,0
            SOJG        A2,TTW.01
            SKIPN       A2
            IDPB        A2,A1
            MOVEI       A2,<^D1>(GB)
            IOR         A2,COUTST
            XCT         A2
            XPROCE      ^D2,^D0,^D0
TTW.1:
            EPROCE      ^D1,^D0,^D0,^D0
            SETZ        A3,
            ASCPTR      A4,<^D1>(GB)
            MOVE        A1,-<^D1>(LB)
            JUMPGE      A1,TTW.11
            MOVEI       A2,'-'
            IDPB        A2,A4
            MOVN        A1,A1
TTW.11:     IDIVI       A1,^D10
            PUSH        P,A2
            AOJ         A3,
            JUMPG       A1,TTW.11
TTW.12:     POP         P,A2
            IORI        A2,'0'
            IDPB        A2,A4
            SOJG        A3,TTW.12
            IDPB        A3,A4
            MOVEI       A3,<^D1>(GB)
            IOR         A3,COUTST
            XCT         A3
            XPROCE      ^D1,^D0,^D0
TTW.2:
            EPROCE      ^D1,^D0,^D0,^D0
            MOVE        A1,-<^D1>(LB)
            OUTCHR      A1
            XPROCE      ^D1,^D0,^D0
TTW.3:
            EPROCE      ^D0,^D0,^D0,^D0
            MOVEI       A1,12
            OUTCHR      A1
            MOVEI       A1,15
            OUTCHR      A1
            XPROCE      ^D0,^D0,^D0
TTR.0:
            EPROCE      ^D2,^D0,^D0,^D0
            SETZ        A2,
TTR.01:     INCHRW      A3
            CAIGE       A3,' '
            JRST        TTR.02
            IDPB        A3,-<^D2>(LB)
            AOJ         A2,
            JRST        TTR.01
TTR.02:     MOVEI       A3,0
            IDPB        A3,-<^D2>(LB)
            MOVEM       A2,@-<^D1>(LB)
            XPROCE      ^D2,^D0,^D0
TTY.I:
            EPROCE      ^D0,^D0,^D0,^D0
            XPROCE      ^D0,^D0,^D0
PRS.0:      ASCIZ   /PROCESSO /
OBS.0:
            EPROCE      ^D0,^D0,^D0,^D0
OBS.01:
            MOVE        T1,<^D1>(GB)
            MOVEI       T2,<^D3>(GB)
            MOVEI       T3,<^D22>(GB)
            MOVEI       T4,<^D21>(GB)
            PUSHJ       P,ATN.0
            SKIPN       <^D21>(GB)
            JRST        OBS.03
            MOVE        A1,<^D21>(GB)
            CAMN        A1,<^D20>(GB)
```

```
        JRST    OBS,02
        MOVEM   A1,<^D20>(GB)
        PUTSTR  PRS,0,^D9
        PUTNUM  <^D21>(GB)
        PUTCHR  ")"
        PUTLN
OBS,02: PUTSTR  <^D3>(GB),^D80
        PUTLN
        MOVE    A1,<^D3>(GB)
        LSH     A1,-^D29
        CAIE    A1,"?"
        JRST    OBS,03
        CLRPFI
        PUTCHR  "?"
        GETSTR  <^D3>(GB),<^D19>(GB)
        MOVE    T1,<^D1>(GB)
        MOVEI   T2,<^D3>(GB)
        PUSHJ   P,RPY,0
OBS,03: SAIFE   <^D22>(GB)
        JRST    OBS,01
        MOVE    T1,<^D2>(GB)
        PUSHJ   P,SSF,1
        JUMPE   T1,OBS,01
        XPROCE  ^D0,^D0,^D0
OBS,I:
        EPROCE  ^D0,^D0,^D0,^D0
        INICLA  TTY,V,TTY,I
        SETZM   <^D20>(GB)
        SETZM   <^D21>(GB)
        SETZM   <^D22>(GB)
        XPROCE  ^D0,^D0,^D0
OBS,V:
        XWD     OBS,I,^D1002
        BLOCK   ^D3
        BLOCK   ^D6
        BLOCK   5
        BLOCK   2
        BLOCK   1
TTY,V:
        XWD     TTY,I,^D1001
        BLOCK   ^D9
        BLOCK   ^D7
PTASCZ::
        ILDB    C,T2
        CAIN    C,0
        POPJ    P,
        IDPB    C,PTOBP(T1)
        IFN TSTSW,<OUTCHR               C>
        SOS     PTOBC(T1)
        JRST    PTASCZ
PTCRLF::
        MOVEI   C,2415
        IDPB    C,PTOBP(T1)
        IFN TSTSW,<OUTCHR               C>
        SOS     PTORC(T1)
        LSH     C,-7
        CAIE    C,0
        JRST    PTCRLF+1
        OUTPUT  PTY,
        POPJ    P,
PTCHR::
        IDPB    T2,PTOBP(T1)
        IFN TSTSW,<OUTCHR               T2>
        SOS     PTOBC(T1)
        POPJ    P,
PT.UML::
        INPUT   PTY,
        MOVEI   T2,PTY
        JOBSTS  T2,
          JRST  PT.UML+1
```

```
        TLNE            T2,(JB.UOA)
        JRST            PT.UML
        TLNN            T2,(JB.UOI)
        JRST            PT.UML
        POPJ            P,
PT.TTY::
        INPUT           PTY,
        SOSGE           PTIBC(T1)
        JRST            .+6
        ILDB            C,PTIBP(T1)
        CAIN            C,0
        JRST            .+3
        OUTCHR          C
        JRST            .-6
        MOVEI           T2,PTY
        JOBSTS          T2,
          JRST          .-4
        TLNE            T2,(JB.UOA)
        JRST            .-14
        TLNN            T2,(JB.UOI)
        JRST            .-16
        POPJ            P,
PTOPEN::
        MOVE            T2,.+9
        IOR             T2,T1
        XCT             T2
          JRST          .+4
        PUSHJ           P,PTCRLF
        PUSHJ           P,PT.UML
        POPJ            P,
        OUTSTR          .+3
        EXIT
        OPEN            PTY,0
        ASCIZ           /?PTY OPEN FAIL
/

LOGIN::
        PUSH            P,T2
        MOVEI           T2,PTY
        JOBSTS          T2,
          PUSHJ         P,PTOPEN
        HRLI            T2,440700
        HRRI            T2,.+8
        PUSHJ           P,PTASCZ
        POP             P,T2
        PUSHJ           P,PTASCZ
        PUSHJ           P,PTCRLF
        IFN TSTSW,<PUSHJ                P,PT.TTY>
        IFE TSTSW,<PUSHJ                P,PT.UML>
        ANDI            T2,777
        POPJ            P,
        ASCIZ           /LOG /

RUNPRO::
        PUSH            P,T2
        HRLI            T2,440700
        HRRI            T2,.+16
        PUSHJ           P,PTASCZ
        POP             P,T2
        PUSHJ           P,PTASCZ
        PUSHJ           P,PTCRLF
        IFN TSTSW,<PUSHJ                P,PT.TTY>
        IFE TSTSW,<PUSHJ                P,PT.UML>
        MOVEI           T2,PTY
        JOBSTS          T2,
          JRST          .-2
        TLNN            T2,(JB.ULI)
        POPJ            P,
        MOVEI           T2,1
        SLEEP           T2,
        JRST            .-7
```

```
        ASCIZ           /RUN /

GTPPN::
        GETPPN          T2,
        MOVE            T2,T2
        MOVEI           T4,6
        SETZ            T3,
        ROTC            T2,3
        IORI            T3,'0'
        IDPB            T3,T1
        SOJG            T4,.-4
        JUMPE           T2,.+4
        MOVEI           T3,','
        IDPB            T3,T1
        JRST            .-9
        IDPB            T2,T1
        POPJ            P,
START:
                RESET
                MOVE            P,.PDPTR
;
;       Verifica se o usuario tem privilegio para fazer ENQ./DEQ.
;
                HRLI            A1,-1
                HRRI            A1,.GTPRV
                GETTAB          A1,
                  JRST          .-3
                TLNN            A1,000100
                  JRST          [OUTSTR [ASCIZ '?FALTA PRIVILEGIO DE ENQ./DEQ.']
                                 EXIT]
;
;       Habilita-se a acessar o high segment de modo exclusivo
; fazendo ENQUEUE(hghnam).
;
                MOVEI           A1,[XWD 1,5
                                    XWD 0,1
                                    XWD 0,-2
                                    XWD -1,[ASCIZ HGHNAM]
                                    XWD 0,0]
                ENQ,            A1,
                  JRST          [OUTSTR [ASCIZ '?ENQ, ']
                                 OUTSTR [ASCIZ HGHNAM]
                                 EXIT]
;
;       Adquire o high segment que sera compartilhado pelos
; processos componentes do programa.
;
                MOVEI           A1,[SIXBIT /IS/
                                    SIXBIT HGHNAM
                                    SIXBIT /EXE/
                                    EXP 0
                                    EXP 0
                                    EXP 0]
                GETSEG          A1,
                  JRST          [OUTSTR [ASCIZ '?GETSEG ']
                                 OUTSTR [ASCIZ HGHNAM]
                                 EXIT]
;
;       Habilita-se a escrever no high segment fazendo SETUWP(0)
;
                SETZ            A1,
                SETUWP          A1,
                  JRST          [OUTSTR [ASCIZ '?SETUWP ']
                                 OUTSTR [ASCIZ HGHNAM]
                                 EXIT]
;
;       Inicializa o registro usado para definir os recursos
; virtuais correspondentes aos monitores, de modo que o nu-
; mero do Job do processo inicial faca parte do user code de
; identificacao desses monitores nas chamadas de ENQ./DEQ.
;
```

```
                FJOB            A1,
                HRLZ            A1,A1
                IORM            A1,MONREC+3
;
;       Inicializa os monitores de servico.
;
        INIMON          CTY,V,CTY,I
        IMPAR           CTY,V,<CTF,V+1>,1
        INIMON          CTF,V,CTF,I
;
;       Inicializa as classes fixas do processo inicial.
;
        ICPAR           CTY,V,<OBS,V+1>,1
        ICPAR           CTF,V,<OBS,V+2>,1
        INICLA          OBS,V,OBS,I
                RELOC
>
DEFINE FINLOW(HGHNAM)
<XALL
        RELOC
        MOVE            T1,PPNASC
        PUSHJ           P,GTPPN
        MOVEI           T1,PTYBLK
        MOVSI           A1,-NOPRO
DSP.01: MOVE            T2,PPNASC
        PUSHJ           P,LOGIN
        MOVEM           T2,PROJOB(A1)
        MOVE            A2,PARPRO(A1)
        HRLI            A2,1(A1)
        MOVEM           A2,TBJOB(T2)
        HRRZ            T2,A1
        LSH             T2,1
        MOVEI           T2,PRGNAM(T2)
        HRLI            T2,440700
        PUSHJ           P,RUNPRO
        AOBJN           A1,DSP.01
DSP.02: MOVEI           T1,CTF,V
        PUSHJ           P,SGF,2
        CAIE            T1,0
        JRST            DSP.03
        MOVEI           A1,20
        HIBER           A1,
          JRST          ,-2
        JRST            DSP.02
DSP.03: DSK=15
        GETPPN          A1,
        MOVE            A1,A1
        MOVEM           A1,LUPHGH+3
        MOVEM           A1,RENHGH+3
        OPEN            DSK,[EXP        ,IODMP
                             SIXBIT     /DSK/
                             XWD        0,0]
          JRST          ,-1
        LOOKUP          DSK,LUPHGH
          JRST          ,-1
        RENAME          DSK,RENHGH
          JRST          ,-1
        MOVEI           A1,[XWD 1,5
                            XWD 0,1
                            XWD 0,-2
                            XWD -1,[ASCIZ HGHNAM]
                            XWD 0,0]
        DEQ,            A1,
          JRST          [OUTSTR [ASCIZ /?DEQ. /]
                         OUTSTR [ASCIZ HGHNAM]
                         EXIT]
        MOVEI           T1,OBS,V
        PUSHJ           P,OBS,0
        EXIT
LUPHGH: SIXBIT          HGHNAM
        SIXBIT          /EXE/
```

```
        XWD             0,0
        XWD             0,0
RENHGH: SIXBIT          HGHNAM
        SIXBIT          /EXE/
        EXP             077B8
        XWD             0,0
PPNASC: XWD             440700,,+1
        BLOCK           3
PTYBLK: EXP             0
        SIXBIT          /PTY/
        XWD             PTYBLK+PTOB,PTYBLK+PTIB
        BLOCK           6
.PDL:   BLOCK           200
.PDPTR: IOWD            200,.PDL
        RELOC
>
DEFINE INIPRO(HGHNAM,PNAME,PPNUM)
<XALL
PROSTA: RESET
        MOVSI           A1,-1
        ATTACH          A1,
          JRST          .-2
        MOVEI           A1,[SIXBIT /DSK/
                            SIXBIT HGHNAM
                            SIXBIT /EXE/
                            EXP 0
                            EXP 0
                            EXP 0]
        GETSEG          A1,
          JRST          PROEND
        SETZ            A1,
        SETUWP          A1,
          JRST          PROEND
        PJOB            A1,
        MOVE            A1,TRUSER(A1)
        MOVEM           A1,PNAME
        IFN PPNUM,<
        HRLI            A1,(A1)
        HRRI            A1,PNAME+1
        BLT             A1,PNAME+PPNUM
        >
        MOVE            P,PDLPTR
        MOVEI           SB,PNAME
        MOVEI           L2,0
        MOVEI           T1,CTF,V
        PUSHJ           P,SGF,3
>
DEFINE FINPRO
<XALL
        MOVEI           T1,CTF,V
        PUSHJ           P,SGF,0
PROEND: MOVEI           A1,[SIXBIT /SYS/
                            SIXBIT /LOGOUT/
                            SIXBIT /EXE/
                            EXP 0
                            XWD 1,4
                            EXP 0]
        RUN             A1,
>
DEFINE CTWSTR(STR)
<XALL
        MOVEI           T1,CTY,V
        MOVEI           T2,STR
        PUSHJ           P,REQ,0
>
DEFINE COPYW(AC1,AC2,N,AC)
<
        HRLI            AC,(AC1)
        HRRI            AC,(AC2)
        BLT             AC,<N-1>(AC2)
>
```

```
DEFINE PUTSTR(STR,N)
<
        MOVEI           T1,TTY.V
        ASCPTR          T2,STR
        MOVEI           T3,N
        PUSHJ           P,TTW.0
>
DEFINE ASCPTR(AC,STR)
<
        HRLI            AC,440700
        HRRI            AC,STR
>
DEFINE PUTNUM(N)
<
        MOVEI           T1,TTY.V
        MOVE            T2,N
        PUSHJ           P,TTW.1
>
DEFINE PUTCHR(C)
<
        MOVEI           T1,TTY.V
        MOVEI           T2,C
        PUSHJ           P,TTW.2
>
DEFINE PUTLN
<
        PUSHJ           P,TTW.3
>
DEFINE GETSTR(STR,N)
<
        MOVEI           T1,TTY.V
        ASCPTR          T2,STR
        MOVEI           T3,N
        PUSHJ           P,TTR.0
>
DEFINE CALL(ROT)
<
        PUSHJ           P,ROT
>
DEFINE CASE(AC,OTHERS,CASES,%NCASES)
<
        CAIL    AC,1
        CAILE   AC,%NCASES
        JRST    OTHERS
        JRST    @.(AC)
        IF1 <%NCASES=0>
        IFE     CASES,<
        IF1 <%NCASES=%NCASES+1>
        EXP     CASES>
>
DEFINE RETURN
<
        POPJ            P,
>
DEFINE SAVE(X)
<
        PUSH            P,X
>
DEFINE REST(X)
<
        POP             P,X
>
DEFINE GOTO(LAB)
<
        JRST            LAB
>
DEFINE SKIPIN(AC,MIN,MAX)
<
        CAIL            AC,MIN
        CAILE           AC,MAX
>
```

```
DEFINE FALSE(AC)
<
        SETZ            AC,
>
DEFINE FALSEM(M)
<
        SETZM           M
>
DEFINE TRUE(AC)
<
        SETO            AC,
>
DEFINE TRUEM(M)
<
        SETOM           M
>
DEFINE INC(AC)
<
        AOJ             AC,
>
DEFINE INCM(M)
<
        AOS             M
>
DEFINE DEC(AC)
<
        SOJ             AC,
>
DEFINE DECM(M)
<
        SOS             M
>
END
```

APÊNDICE 3

Listagens da compilação e execução do programa que implementa uma solução para o problema "Dining Philosofers" de Djikstra, construido segundo o método de programação apresentado.

```
18:04:18 BAJOB   BATCON version 102(2057) running TST6 sequence 219 in stream 1
18:04:18 BAFIL   Input from DSKC:TST6.CTL[72540,4034]
18:04:18 BAFIL   Output to  DSKC:TST6.LOG[72540,4034]
18:04:18 BASUM   Job parameters
                 Time:00:01:40   Unique:YES   Restart:NO   Output:LOG

18:04:18 MONTR
18:04:18 MONTR   .LOGIN 72540/4034 /DEFER/SPOOL:ALL/TIME:100/LOCATE:10/NAME:'SUPISC - ATS'
18:04:18 USER    JOB 10 UFPA SECOM DEC-1091 G-06 TTY53
18:04:19 USER    [LGNJSP Other Jobs same PPN:8,12]
18:04:17 USER    1804    12-Feb-83       Sat
18:04:20 MONTR
18:04:20 MONTR   .DO C TST6.PRG/LIST
18:04:21 MONTR
18:04:21 MONTR   .ERROR $
18:04:21 MONTR
18:04:21 MONTR   .MIC RESPONSE B(15)
18:04:21 MONTR
18:04:21 MONTR   .R SNOBOL
18:04:21 USER
18:04:21 USER    *C,TTY:
18:04:24 USER    TST6.PRG/LIST
18:04:24 USER
18:04:24 USER    PRE-PROCESSADOR PARA CONSTRUCAO DE PROGRAMAS CONCORRENTES      -      V1.0/JANEIRO-83
18:04:31 USER    *TST6.MIC
19:04:31 USER    *^C
18:04:31 MONTR
18:04:31 MONTR   .LET B=$B.[2,15]
18:04:31 MONTR
18:04:31 MONTR   .TE TST6.MIC
18:04:32 USER
18:04:32 USER    [2K Core]
18:04:32 USER    *
18:04:32 USER    J<TS<ESC>$^R^R$$;>$EX$$
18:04:33 MONTR
18:04:33 MONTR   .
18:04:33 MONTR   .LET B=$B.[1,','J+'.BAK'
18:04:33 MONTR
18:04:33 MONTR   .DEL TST6.BAK
18:04:33 USER    Files deleted:
18:04:34 USER    DSKC:TST6.BAK
18:04:34 USER    01 Blocks freed
18:04:34 MONTR
18:04:34 MONTR   .^Z
18:04:34 MONTR   .
18:04:34 MONTR   .DO TST6 /LIST
18:04:49 MONTR
18:04:49 MONTR   .TE FIVEPH.MAC
18:04:49 USER
18:04:49 USER    [2K Core]
18:04:49 USER    *IUNIVERSAL     FIVEPH
18:04:50 USER    $EX$$
18:04:51 MONTR
18:04:51 MONTR   .
18:04:51 MONTR   .COMP/COMP FIVEPH.MAC
18:04:51 USER    MACRO:  FIVEPH
18:04:58 MONTR
18:04:58 MONTR   EXIT
18:04:58 MONTR
18:04:58 MONTR   .TE FIVEPH.MAC
18:04:59 USER
18:04:59 USER    [2K Core]
18:04:59 USER    *FSUNIVERSAL$TITLE$EX$$
18:05:00 MONTR
18:05:00 MONTR   .
18:05:00 MONTR   .LOAD/COMP/LIST FIVEPH.MAC
18:05:00 USER    MACRO:  FIVEPH
18:05:07 USER    LINK:   Loading
18:05:08 MONTR
18:05:08 MONTR   EXIT
18:05:08 MONTR
```

```
18:05:06 MONTR  .SSAVE
18:05:09 MONTR  FIVEPH saved
18:05:09 MONTR
18:05:09 MONTR  .LOAD/COMP/LIST FIL01 .MAC
18:05:09 USER   MACRO:  FIL01
18:05:13 USER   LINK:   Loading
18:05:13 MONTR
18:05:13 MONTR  EXIT
18:05:13 MONTR
18:05:13 MONTR  .SAVE
18:05:14 MONTR  FIL01 saved
18:05:14 MONTR
18:05:14 MONTR  .TE FIVEPH.MAC
18:05:14 USER
18:05:14 USER   [2k Core]
18:05:14 USER   ^K$EX$$
18:05:15 MONTR
18:05:15 MONTR  .
18:05:15 MONTR  .DEL FIVEPH.BAK
18:05:16 USER   Files deleted:
18:05:16 USER   DSKC:FIVEPH.MAC
18:05:16 USER   04 Blocks freed
18:05:16 MONTR
18:05:16 MONTR  .RUN FIVEPH
18:05:31 USER
18:05:43 USER   PROCESSO 1)
18:05:43 USER   INICIANDO
18:05:43 USER   PROCESSO 2)
18:05:43 USER   INICIANDO
18:05:43 USER   PROCESSO 3)
18:05:43 USER   INICIANDO
18:05:43 USER   PROCESSO 4)
18:05:43 USER   INICIANDO
18:05:43 USER   PROCESSO 5)
18:05:43 USER   INICIANDO
18:05:43 USER   PROCESSO 1)
18:05:43 USER   VOU TENTAR PEGAR OS GARFOS
18:05:43 USER   PROCESSO 2)
18:05:43 USER   VOU TENTAR PEGAR OS GARFOS
18:05:43 USER   PROCESSO 1)
18:05:43 USER   CONSEGUI OS GARFOS
18:05:43 USER   PROCESSO 2)
18:05:43 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:43 USER   PROCESSO 3)
18:05:43 USER   VOU TENTAR PEGAR OS GARFOS
18:05:43 USER   PROCESSO 4)
18:05:43 USER   VOU TENTAR PEGAR OS GARFOS
18:05:43 USER   PROCESSO 1)
18:05:43 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:43 USER   PROCESSO 3)
18:05:43 USER   CONSEGUI OS GARFOS
18:05:43 USER   PROCESSO 4)
18:05:43 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:43 USER   PROCESSO 5)
18:05:43 USER   VOU TENTAR PEGAR OS GARFOS
18:05:43 USER   PROCESSO 3)
18:05:43 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:43 USER   PROCESSO 5)
18:05:43 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:43 USER   VOU TENTAR PEGAR OS GARFOS
18:05:43 USER   CONSEGUI OS GARFOS
18:05:43 USER   PROCESSO 2)
18:05:43 USER   VOU TENTAR PEGAR OS GARFOS
18:05:43 USER   PROCESSO 5)
18:05:43 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:43 USER   PROCESSO 1)
18:05:43 USER   NAO ABORRECA ESTOU PENSANDO
18:05:43 USER   PROCESSO 2)
18:05:43 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:43 USER   VOU TENTAR PEGAR OS GARFOS
18:05:43 USER   CONSEGUI OS GARFOS
```

```
18:05:43 USER   PROCESSO 4)
18:05:43 USER   VOU TENTAR PEGAR OS GARFOS
18:05:43 USER   PROCESSO 2)
18:05:43 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:43 USER   PROCESSO 3)
18:05:43 USER   NAO ABORRECA ESTOU PENSANDO
18:05:43 USER   PROCESSO 4)
18:05:43 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:43 USER   VOU TENTAR PEGAR OS GARFOS
18:05:43 USER   PROCESSO 5)
18:05:43 USER   NAO ABORRECA ESTOU PENSANDO
18:05:43 USER   PROCESSO 1)
18:05:43 USER   VOU TENTAR PEGAR OS GARFOS
18:05:43 USER   PROCESSO 4)
18:05:44 USER   CONSEGUI OS GARFOS
18:05:44 USER   PROCESSO 1)
18:05:44 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:44 USER   PROCESSO 4)
18:05:44 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:44 USER   PROCESSO 1)
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   PROCESSO 2)
18:05:44 USER   NAO ABORRECA ESTOU PENSANDO
18:05:44 USER   PROCESSO 3)
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   PROCESSO 1)
18:05:44 USER   CONSEGUI OS GARFOS
18:05:44 USER   PROCESSO 3)
18:05:44 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:44 USER   PROCESSO 5)
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   PROCESSO 1)
18:05:44 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:44 USER   PROCESSO 5)
18:05:44 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:44 USER   PROCESSO 3)
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   CONSEGUI OS GARFOS
18:05:44 USER   PROCESSO 5)
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   PROCESSO 3)
18:05:44 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:44 USER   PROCESSO 4)
18:05:44 USER   NAO ABORRECA ESTOU PENSANDO
18:05:44 USER   PROCESSO 5)
18:05:44 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:44 USER   PROCESSO 2)
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:44 USER   PROCESSO 5)
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   CONSEGUI OS GARFOS
18:05:44 USER   PROCESSO 2)
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   PROCESSO 5)
18:05:44 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:44 USER   PROCESSO 1)
18:05:44 USER   NAO ABORRECA ESTOU PENSANDO
18:05:44 USER   PROCESSO 2)
18:05:44 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   PROCESSO 3)
18:05:44 USER   NAO ABORRECA ESTOU PENSANDO
18:05:44 USER   PROCESSO 4)
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   PROCESSO 2)
18:05:44 USER   CONSEGUI OS GARFOS
18:05:44 USER   PROCESSO 4)
18:05:44 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:44 USER   PROCESSO 2)
18:05:44 USER   ESTOU OCUPADO COMENDO MACARRAO
```

```
18:05:44 USER   PROCESSO 4)
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   PROCESSO 5)
18:05:44 USER   NAO ABORRECA ESTOU PENSANDO
18:05:44 USER   PROCESSO 1)
18:05:44 USER   VOU TENTAR PEGAR OS GARFOS
18:05:44 USER   PROCESSO 4)
18:05:45 USER   CONSEGUI OS GARFOS
18:05:45 USER   PROCESSO 1)
18:05:45 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:45 USER   PROCESSO 3)
18:05:45 USER   VOU TENTAR PEGAR OS GARFOS
18:05:45 USER   PROCESSO 4)
18:05:45 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:45 USER   PROCESSO 3)
18:05:45 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:45 USER   PROCESSO 1)
18:05:45 USER   VOU TENTAR PEGAR OS GARFOS
18:05:45 USER   CONSEGUI OS GARFOS
18:05:45 USER   PROCESSO 3)
18:05:45 USER   VOU TENTAR PEGAR OS GARFOS
18:05:45 USER   PROCESSO 1)
18:05:45 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:45 USER   PROCESSO 2)
18:05:45 USER   NAO ABORRECA ESTOU PENSANDO
18:05:45 USER   PROCESSO 3)
18:05:45 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:45 USER   PROCESSO 5)
18:05:45 USER   VOU TENTAR PEGAR OS GARFOS
18:05:45 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:45 USER   PROCESSO 2)
18:05:45 USER   VOU TENTAR PEGAR OS GARFOS
18:05:45 USER   CONSEGUI OS GARFOS
18:05:45 USER   PROCESSO 4)
18:05:45 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:45 USER   PROCESSO 5)
18:05:45 USER   NAO ABORRECA ESTOU PENSANDO
18:05:45 USER   PROCESSO 2)
18:05:45 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:45 USER   PROCESSO 4)
18:05:45 USER   VOU TENTAR PEGAR OS GARFOS
18:05:45 USER   PROCESSO 5)
18:05:45 USER   CONSEGUI OS GARFOS
18:05:45 USER   PROCESSO 2)
18:05:45 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:45 USER   PROCESSO 5)
18:05:45 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:45 USER   PROCESSO 2)
18:05:45 USER   VOU TENTAR PEGAR OS GARFOS
18:05:45 USER   PROCESSO 3)
18:05:45 USER   NAO ABORRECA ESTOU PENSANDO
18:05:45 USER   PROCESSO 4)
18:05:45 USER   VOU TENTAR PEGAR OS GARFOS
18:05:45 USER   PROCESSO 2)
18:05:45 USER   CONSEGUI OS GARFOS
18:05:45 USER   PROCESSO 4)
18:05:45 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:45 USER   PROCESSO 1)
18:05:45 USER   VOU TENTAR PEGAR OS GARFOS
18:05:45 USER   PROCESSO 2)
18:05:45 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:45 USER   PROCESSO 1)
18:05:45 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:45 USER   PROCESSO 4)
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   CONSEGUI OS GARFOS
```

```
18:05:46 USER   PROCESSO 1)
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   PROCESSO 4)
18:05:46 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:46 USER   PROCESSO 5)
18:05:46 USER   NAO ABORRECA ESTOU PENSANDO
18:05:46 USER   PROCESSO 1)
18:05:46 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:46 USER   PROCESSO 3)
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:46 USER   PROCESSO 1)
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   CONSEGUI OS GARFOS
18:05:46 USER   PROCESSO 3)
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   PROCESSO 1)
18:05:46 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:46 USER   PROCESSO 2)
18:05:46 USER   NAO ABORRECA ESTOU PENSANDO
18:05:46 USER   PROCESSO 3)
18:05:46 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   PROCESSO 4)
18:05:46 USER   NAO ABORRECA ESTOU PENSANDO
18:05:46 USER   PROCESSO 5)
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   PROCESSO 3)
18:05:46 USER   CONSEGUI OS GARFOS
18:05:46 USER   PROCESSO 5)
18:05:46 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:46 USER   PROCESSO 3)
18:05:46 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:46 USER   PROCESSO 5)
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   PROCESSO 1)
18:05:46 USER   NAO ABORRECA ESTOU PENSANDO
18:05:46 USER   PROCESSO 3)
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   PROCESSO 5)
18:05:46 USER   CONSEGUI OS GARFOS
18:05:46 USER   PROCESSO 2)
18:05:46 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:46 USER   PROCESSO 4)
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   PROCESSO 5)
18:05:46 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:46 USER   PROCESSO 4)
18:05:46 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:46 USER   PROCESSO 2)
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   CONSEGUI OS GARFOS
18:05:46 USER   PROCESSO 4)
18:05:46 USER   VOU TENTAR PEGAR OS GARFOS
18:05:46 USER   PROCESSO 2)
18:05:46 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:46 USER   PROCESSO 3)
18:05:46 USER   NAO ABORRECA ESTOU PENSANDO
18:05:46 USER   PROCESSO 4)
18:05:46 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:46 USER   PROCESSO 1)
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:47 USER   PROCESSO 4)
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   CONSEGUI OS GARFOS
18:05:47 USER   PROCESSO 1)
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   PROCESSO 4)
18:05:47 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:47 USER   PROCESSO 5)
```

```
18:05:47 USER   NAO ABORRECA ESTOU PENSANDO
18:05:47 USER   PROCESSO 1)
18:05:47 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   PROCESSO 2)
18:05:47 USER   NAO ABORRECA ESTOU PENSANDO
18:05:47 USER   PROCESSO 3)
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   PROCESSO 1)
18:05:47 USER   CONSEGUI OS GARFOS
18:05:47 USER   PROCESSO 3)
18:05:47 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:47 USER   PROCESSO 1)
18:05:47 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:47 USER   PROCESSO 3)
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   PROCESSO 4)
18:05:47 USER   NAO ABORRECA ESTOU PENSANDO
18:05:47 USER   PROCESSO 5)
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   PROCESSO 3)
18:05:47 USER   CONSEGUI OS GARFOS
18:05:47 USER   PROCESSO 5)
18:05:47 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:47 USER   PROCESSO 2)
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   PROCESSO 3)
18:05:47 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:47 USER   PROCESSO 2)
18:05:47 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:47 USER   PROCESSO 5)
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   CONSEGUI OS GARFOS
18:05:47 USER   PROCESSO 2)
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   PROCESSO 5)
18:05:47 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:47 USER   PROCESSO 1)
18:05:47 USER   NAO ABORRECA ESTOU PENSANDO
18:05:47 USER   PROCESSO 2)
18:05:47 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:47 USER   PROCESSO 4)
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:47 USER   PROCESSO 2)
18:05:47 USER   VOU TENTAR PEGAR OS GARFOS
18:05:47 USER   CONSEGUI OS GARFOS
18:05:48 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:48 USER   PROCESSO 4)
18:05:48 USER   VOU TENTAR PEGAR OS GARFOS
18:05:48 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:48 USER   PROCESSO 3)
18:05:48 USER   NAO ABORRECA ESTOU PENSANDO
18:05:48 USER   PROCESSO 4)
18:05:48 USER   VOU TENTAR PEGAR OS GARFOS
18:05:48 USER   CONSEGUI OS GARFOS
18:05:48 USER   PROCESSO 5)
18:05:48 USER   NAO ABORRECA ESTOU PENSANDO
18:05:48 USER   PROCESSO 1)
18:05:48 USER   VOU TENTAR PEGAR OS GARFOS
18:05:48 USER   PROCESSO 4)
18:05:48 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:48 USER   PROCESSO 1)
18:05:48 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:48 USER   VOU TENTAR PEGAR OS GARFOS
18:05:48 USER   CONSEGUI OS GARFOS
18:05:48 USER   PROCESSO 2)
18:05:48 USER   NAO ABORRECA ESTOU PENSANDO
18:05:48 USER   PROCESSO 3)
18:05:48 USER   VOU TENTAR PEGAR OS GARFOS
18:05:48 USER   PROCESSO 1)
```

```
18:05:48 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:48 USER   PROCESSO 3)
18:05:48 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:48 USER   VOU TENTAR PEGAR OS GARFOS
18:05:48 USER   CONSEGUI OS GARFOS
18:05:48 USER   PROCESSO 4)
18:05:48 USER   NAO ABORRECA ESTOU PENSANDO
18:05:48 USER   PROCESSO 5)
18:05:48 USER   VOU TENTAR PEGAR OS GARFOS
18:05:48 USER   PROCESSO 3)
18:05:48 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:48 USER   PROCESSO 5)
18:05:48 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:48 USER   VOU TENTAR PEGAR OS GARFOS
18:05:48 USER   PROCESSO 1)
18:05:48 USER   NAO ABORRECA ESTOU PENSANDO
18:05:48 USER   PROCESSO 2)
18:05:49 USER   VOU TENTAR PEGAR OS GARFOS
18:05:48 USER   PROCESSO 5)
18:05:48 USER   CONSEGUI OS GARFOS
18:05:48 USER   PROCESSO 2)
18:05:48 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:48 USER   PROCESSO 5)
18:05:48 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:48 USER   PROCESSO 2)
18:05:48 USER   VOU TENTAR PEGAR OS GARFOS
18:05:48 USER   PROCESSO 3)
18:05:48 USER   NAO ABORRECA ESTOU PENSANDO
18:05:48 USER   PROCESSO 4)
18:05:48 USER   VOU TENTAR PEGAR OS GARFOS
18:05:48 USER   PROCESSO 2)
18:05:49 USER   CONSEGUI OS GARFOS
18:05:49 USER   PROCESSO 4)
18:05:49 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:49 USER   PROCESSO 1)
18:05:49 USER   VOU TENTAR PEGAR OS GARFOS
18:05:49 USER   PROCESSO 2)
18:05:49 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:49 USER   PROCESSO 1)
18:05:49 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:49 USER   PROCESSO 4)
18:05:49 USER   VOU TENTAR PEGAR OS GARFOS
18:05:49 USER   CONSEGUI OS GARFOS
18:05:49 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:49 USER   PROCESSO 1)
18:05:49 USER   VOU TENTAR PEGAR OS GARFOS
18:05:49 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:49 USER   PROCESSO 5)
18:05:49 USER   NAO ABORRECA ESTOU PENSANDO
18:05:49 USER   PROCESSO 3)
18:05:49 USER   VOU TENTAR PEGAR OS GARFOS
18:05:49 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:49 USER   PROCESSO 1)
18:05:49 USER   VOU TENTAR PEGAR OS GARFOS
18:05:49 USER   CONSEGUI OS GARFOS
18:05:49 USER   PROCESSO 3)
18:05:49 USER   VOU TENTAR PEGAR OS GARFOS
18:05:49 USER   PROCESSO 1)
18:05:49 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:49 USER   PROCESSO 2)
18:05:49 USER   NAO ABORRECA ESTOU PENSANDO
18:05:49 USER   PROCESSO 3)
18:05:49 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:49 USER   VOU TENTAR PEGAR OS GARFOS
18:05:49 USER   PROCESSO 4)
18:05:49 USER   NAO ABORRECA ESTOU PENSANDO
18:05:49 USER   PROCESSO 5)
18:05:49 USER   VOU TENTAR PEGAR OS GARFOS
18:05:49 USER   PROCESSO 3)
18:05:49 USER   CONSEGUI OS GARFOS
18:05:49 USER   PROCESSO 5)
```

```
18:05:49 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:49 USER   PROCESSO 3)
18:05:49 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:49 USER   PROCESSO 5)
18:05:49 USER   VOU TENTAR PEGAR OS GARFOS
18:05:49 USER   PROCESSO 1)
18:05:49 USER   NAO ABORRECA ESTOU PENSANDO
18:05:49 USER   PROCESSO 2)
18:05:49 USER   VOU TENTAR PEGAR OS GARFOS
18:05:49 USER   PROCESSO 5)
18:05:49 USER   CONSEGUI OS GARFOS
18:05:49 USER   PROCESSO 2)
18:05:49 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:49 USER   PROCESSO 4)
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   PROCESSO 5)
18:05:50 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:50 USER   PROCESSO 4)
18:05:50 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:50 USER   PROCESSO 2)
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   CONSEGUI OS GARFOS
18:05:50 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:50 USER   PROCESSO 4)
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:50 USER   PROCESSO 3)
18:05:50 USER   NAO ABORRECA ESTOU PENSANDO
18:05:50 USER   PROCESSO 1)
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:50 USER   PROCESSO 4)
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   CONSEGUI OS GARFOS
18:05:50 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:50 USER   PROCESSO 1)
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:50 USER   PROCESSO 5)
18:05:50 USER   NAO ABORRECA ESTOU PENSANDO
18:05:50 USER   PROCESSO 1)
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   CONSEGUI OS GARFOS
18:05:50 USER   PROCESSO 2)
18:05:50 USER   NAO ABORRECA ESTOU PENSANDO
18:05:50 USER   PROCESSO 3)
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   PROCESSO 1)
18:05:50 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:50 USER   PROCESSO 3)
18:05:50 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   PROCESSO 4)
18:05:50 USER   NAO ABORRECA ESTOU PENSANDO
18:05:50 USER   PROCESSO 5)
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   PROCESSO 3)
18:05:50 USER   CONSEGUI OS GARFOS
18:05:50 USER   PROCESSO 5)
18:05:50 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:50 USER   PROCESSO 3)
18:05:50 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:50 USER   PROCESSO 5)
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   PROCESSO 1)
18:05:50 USER   NAO ABORRECA ESTOU PENSANDO
18:05:50 USER   PROCESSO 2)
18:05:50 USER   VOU TENTAR PEGAR OS GARFOS
18:05:50 USER   PROCESSO 5)
18:05:50 USER   CONSEGUI OS GARFOS
18:05:50 USER   PROCESSO 2)
```

```
18:05:50 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:50 USER   PROCESSO 4)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   PROCESSO 5)
18:05:51 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:51 USER   PROCESSO 4)
18:05:51 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:51 USER   PROCESSO 2)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   CONSEGUI OS GARFOS
18:05:51 USER   PROCESSO 4)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   PROCESSO 2)
18:05:51 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:51 USER   PROCESSO 3)
18:05:51 USER   NAO ABORRECA ESTOU PENSANDO
18:05:51 USER   PROCESSO 4)
18:05:51 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:51 USER   PROCESSO 1)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:51 USER   PROCESSO 4)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   CONSEGUI OS GARFOS
18:05:51 USER   PROCESSO 1)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   PROCESSO 4)
18:05:51 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:51 USER   PROCESSO 5)
18:05:51 USER   NAO ABORRECA ESTOU PENSANDO
18:05:51 USER   PROCESSO 1)
18:05:51 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   PROCESSO 2)
18:05:51 USER   NAO ABORRECA ESTOU PENSANDO
18:05:51 USER   PROCESSO 3)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   PROCESSO 1)
18:05:51 USER   CONSEGUI OS GARFOS
18:05:51 USER   PROCESSO 3)
18:05:51 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:51 USER   PROCESSO 1)
18:05:51 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:51 USER   PROCESSO 3)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   PROCESSO 4)
18:05:51 USER   NAO ABORRECA ESTOU PENSANDO
18:05:51 USER   PROCESSO 5)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   PROCESSO 3)
18:05:51 USER   CONSEGUI OS GARFOS
18:05:51 USER   PROCESSO 5)
18:05:51 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:51 USER   PROCESSO 2)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   PROCESSO 3)
18:05:51 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:51 USER   PROCESSO 2)
18:05:51 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:51 USER   PROCESSO 5)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   CONSEGUI OS GARFOS
18:05:51 USER   PROCESSO 3)
18:05:51 USER   VOU TENTAR PEGAR OS GARFOS
18:05:51 USER   PROCESSO 5)
18:05:51 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:51 USER   PROCESSO 1)
18:05:51 USER   NAO ABORRECA ESTOU PENSANDO
18:05:51 USER   PROCESSO 2)
18:05:51 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:51 USER   PROCESSO 4)
```

```
18:05:52 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:52 USER   PROCESSO 2)
18:05:52 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   CONSEGUI OS GARFOS
18:05:52 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:52 USER   PROCESSO 4)
18:05:52 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:52 USER   PROCESSO 3)
18:05:52 USER   NAO ABORRECA ESTOU PENSANDO
18:05:52 USER   PROCESSO 4)
18:05:52 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   CONSEGUI OS GARFOS
18:05:52 USER   PROCESSO 5)
18:05:52 USER   NAO ABORRECA ESTOU PENSANDO
18:05:52 USER   PROCESSO 1)
18:05:52 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   PROCESSO 4)
18:05:52 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:52 USER   PROCESSO 1)
18:05:52 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:52 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   PROCESSO 2)
18:05:52 USER   NAO ABORRECA ESTOU PENSANDO
18:05:52 USER   PROCESSO 3)
18:05:52 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   PROCESSO 1)
18:05:52 USER   CONSEGUI OS GARFOS
18:05:52 USER   PROCESSO 3)
18:05:52 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:52 USER   PROCESSO 1)
18:05:52 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:52 USER   PROCESSO 3)
18:05:52 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   PROCESSO 4)
18:05:52 USER   NAO ABORRECA ESTOU PENSANDO
18:05:52 USER   PROCESSO 5)
18:05:52 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   PROCESSO 3)
18:05:52 USER   CONSEGUI OS GARFOS
18:05:52 USER   PROCESSO 5)
18:05:52 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:52 USER   PROCESSO 2)
18:05:52 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   PROCESSO 3)
18:05:52 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:52 USER   PROCESSO 2)
18:05:52 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:52 USER   PROCESSO 5)
18:05:52 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   CONSEGUI OS GARFOS
18:05:52 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:53 USER   PROCESSO 2)
18:05:53 USER   VOU TENTAR PEGAR OS GARFOS
18:05:53 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:53 USER   PROCESSO 1)
18:05:53 USER   NAO ABORRECA ESTOU PENSANDO
18:05:53 USER   PROCESSO 4)
18:05:53 USER   VOU TENTAR PEGAR OS GARFOS
18:05:53 USER   PROCESSO 1)
18:05:53 USER   FINALIZANDO
18:05:53 USER   PROCESSO 4)
18:05:53 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:53 USER   PROCESSO 2)
18:05:53 USER   VOU TENTAR PEGAR OS GARFOS
18:05:53 USER   CONSEGUI OS GARFOS
18:05:53 USER   PROCESSO 4)
18:05:53 USER   VOU TENTAR PEGAR OS GARFOS
18:05:53 USER   PROCESSO 2)
18:05:53 USER   ESTOU OCUPADO COMENDO MACARRAO
```

```
18:05:53 USER   PROCESSO 3)
18:05:52 USER   NAO ABORRECA ESTOU PENSANDO
18:05:53 USER   PROCESSO 4)
18:05:53 USER   NAO TENHO GARFOS PARA COMER, ENTAO DURMO
18:05:53 USER   PROCESSO 3)
18:05:53 USER   FINALIZANDO
18:05:53 USER   PROCESSO 4)
18:05:53 USER   VOU TENTAR PEGAR OS GARFOS
18:05:52 USER   PROCESSO 5)
18:05:53 USER   NAO ABORRECA ESTOU PENSANDO
18:05:53 USER   PROCESSO 2)
18:05:53 USER   NAO ABORRECA ESTOU PENSANDO
18:05:53 USER   PROCESSO 4)
18:05:53 USER   CONSEGUI OS GARFOS
18:05:53 USER   PROCESSO 5)
18:05:54 USER   FINALIZANDO
18:05:54 USER   PROCESSO 2)
18:05:54 USER   FINALIZANDO
18:05:54 USER   PROCESSO 4)
18:05:54 USER   ESTOU OCUPADO COMENDO MACARRAO
18:05:54 USER   NAO ABORRECA ESTOU PENSANDO
18:05:54 USER   FINALIZANDO
18:05:54 USER
18:05:54 MONTR  EXIT
18:05:54 MONTR
18:05:54 MONTR  .
18:05:54 TRUE   .IF(NOERROR)
18:05:54 BATCH  .GOTO FIM
18:05:54 BLABL  FIM::
18:05:54 MONTR  .KJOB/BATCH
18:05:54 USER   8:05:54 USER     TAJL Another job is still logged-in under [72540,4034]]
18:05:54 MONTR  Job 10  User SUPTEC - ATS [72540,4034]
18:05:54 MONTR  Logged-off TTY53  at 18:05:54  on 12-Feb-83
18:05:54 MONTR  Runtime: 0:00:28, KCS:490, Connect time: 0:01:36
18:05:54 MONTR  Disk Reads:980, Writes:341
18:11:30 LPNAT  [LPTLSJ  LPTSPL version 102(2353) running on LPT101, 12-Feb-83 18:11:38]
18:11:33 LPMSG  [LPTSJS  Starting Job TST6, Seq #219, request created at 12-Feb-83 18:05:54]
18:11:36 LPMSG  [LPTSTF  Starting File DSKC:009YY0.LPT<077>[3,3]]
18:11:44 LPMSG  [LPTFPF  Finished Printing File DSKC:009YY0.LPT<077>[3,3]]
18:11:47 LPMSG  [LPTSTF  Starting File DSKC:009Z1B.LPT<077>[3,3](FIVEPH)]
18:13:19 LPMSG  [LPTFPF  Finished Printing File DSKC:009Z1B.LPT<077>[3,3](FIVEPH)]
18:13:19 LPMSG  [LPTSTF  Starting File DSKC:009Z1W.LPT<077>[3,3](FILO1)]
18:13:24 LPMSG  [LPTFPF  Finished Printing File DSKC:009Z1W.LPT<077>[3,3](FILO1)]
```

```
PRE-PROCESSADOR PARA CONSTRUCAO DE PROGRAMAS CONCORRENTES          V1.0/JANEIRO-83

PROGRAM FIVEPH;

TYPE CONTROLE_GARFOS=MONITOR;

     CONST NAO='NAO TENHO GARFOS PARA COMER, ENTAO DURMO';
           VOU='VOU TENTAR PEGAR OS GARFOS';
           OBTIVE='CONSEGUI OS GARFOS';

     TYPE
          TALHER_REC=ARRAY (.5.) OF WORD;
          ESPERA_REC=ARRAY (.5.) OF QUEUE;

     VAR
          TALHERES:TALHER_REC;
          AGUARDE:ESPERA_REC;

     PROCEDURE ENTRY DE_ME_GARFOS(EU:WORD);
     BEGIN
          TENTATIVA:
               CTWSTR  =VOU;
               MOVE    A1,=EU;
               ADDI    A1,=TALHERES;
               MOVE    A2,(A1);
               CAIL    A2,2;
               GOTO    =ENDIF;
                       CTWSTR  =NAO;
                       MOVE    T1,(GB);
                       MOVE    T2,=EU;
                       ADDI    T2,=AGUARDE;
                       CALL    WAIT;
                       GOTO    =TENTATIVA;
               ENDIF;
          MOVE    A1,=EU;
          DEC     A1;
          CAIGE   A1,0;
           MOVEI  A1,4;
          ADDI    A1,=TALHERES;
          DECM    0(A1);
          MOVE    A1,=EU;
          INC     A1;
          CAILE   A1,4;
           MOVEI  A1,0;
          ADDI    A1,=TALHERES;
          DECM    0(A1);
          CTWSTR  =OBTIVE;
     END 'DE_ME_GARFOS';

     PROCEDURE ENTRY TOME_GARFOS(EU:WORD);
     VAR ESQ,DIR:WORD;
     BEGIN
          MOVE    A1,=EU;
          DEC     A1;
          CAIGE   A1,0;
           MOVEI  A1,4;
          MOVEM   A1,=ESQ;
          ADDI    A1,=TALHERES;
          INCM    0(A1);
          MOVE    T1,(GB);
          MOVE    T2,=ESQ;
          ADDI    T2,=AGUARDE;
          CALL    SIGNAL;
          MOVE    A1,=EU;
          INC     A1;
          CAILE   A1,4;
           MOVEI  A1,0;
          MOVEM   A1,=DIR;
          ADDI    A1,=TALHERES;
          INCM    0(A1);
          MOVE    T1,(GB);
```

```
            MOVE    T2,=DIR;
            ADDI    T2,=AGUARDE;
            CALL    SIGNAL;
        END 'TOME_GARFOS';

        BEGIN
            MOVSI   A1,-5;
            HRRI    A1,=TALHERES;
            MOVEI   A2,2;
            MOVEM   A2,(A1);
            AOBJN   A1,',-1';
        END 'CONTROLE_GARFOS';

TYPE FILOSOFO=CLASS(GARFOS:CONTROLE_GARFOS);

        CONST
            COMENDO='ESTOU OCUPADO COMENDO MACARRAO';
            PENSANDO='NAO ABORRECA ESTOU PENSANDO';

        VAR
            EU,
            CONT:WORD;

        PROCEDURE ENTRY LOOP;
        BEGIN
            PJOB    A1,;
            MOVE    A1,TBJOB(A1);
            HLRZM   A1,=EU;
            DECM    =EU;
            REPEAT;
                MOVE    T1,=GARFOS;
                MOVE    T2,=EU;
                CALL    =DE_ME_GARFOS;
                CTWSTR  =COMENDO;
                MOVE    T1,=GARFOS;
                MOVE    T2,=EU;
                CALL    =TOME_GARFOS;
                CTWSTR  =PENSANDO;
                SOSE    =CONT;
                GOTO    =REPEAT;
        END 'LOOP';

        BEGIN
            MOVEI   A1,10;
            MOVEM   A1,=CONT;
        END 'FILOSOFO';

TYPE FILOSOFOS=PROCESS(GARFOS:CONTROLE_GARFOS);

        STACK_SIZE = 1000;

        VAR MAINLOOP:FILOSOFO;

        BEGIN
            INIT MAINLOOP(GARFOS);
            MOVEI   T1,G(MAINLOOP);
            CALL    =LOOP;
        END 'FILOSOFOS';

VAR
        GARFOS:CONTROLE_GARFOS;
        FILOSOFO_1,
        FILOSOFO_2,
        FILOSOFO_3,
        FILOSOFO_4,
        FILOSOFO_5:FILOSOFOS;

BEGIN
        INIT GARFOS,
             FILOSOFO_1(GARFOS),
             FILOSOFO_2(GARFOS),
```

```
FILOSOFO_3(GARFOS),
FILOSOFO_4(GARFOS),
FILOSOFO_5(GARFOS);
END.
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1
FIVEPH  MAC     6-May-83 13:41


                                        TITLE   FIVEPH
                                                SEARCH          UUOSYM,CONSTS,MACROS
400000'                                         TWOSEG
400000'                                         RELOC           400000
                                                INIHGH          'FIVEPH'^XALL
400000' 205 10 0 00 777774      TRGOUT: MOVSI   A1,-4
400001' 200 12 0 10 000001              MOVE    A3,1(A1)
400002' 201 13 0 00 000014              MOVEI   A4,14
400003' 400 11 0 00 000000              SETZ    A2,
400004' 245 11 0 00 000003              ROTC    A2,3
400005' 271 11 0 00 000060              ADDI    A2,"0"
400006' 051 01 0 00 000011              OUTCHR  A2
400007' 367 13 0 10 400003'             SOJG    A4,-4
400010' 201 11 0 00 000040              MOVEI   A2," "
400011' 051 01 0 00 000011              OUTCHR  A2
400012' 253 10 0 00 400001'             AOBJN   A1,TRGOUT+1
400013' 263 17 0 00 000000              POPJ    P,
400014' 000001  000005          MONREC: XWD             1,5
400015' 000000  000001                  XWD             0,1
400016' 200000  777776                  XWD             200000,-2
400017' 500000  000000                  EXP             500000
400020' 000000  000000                  EXP             0
400021' 505 03 0 00 400014'             HRLI            T3,MONREC
400022' 541 03 0 17 000001              HRRI            T3,1(P)
400023' 251 03 0 17 000005              BLT             T3,5(P)
400024' 542 01 0 17 000004              HRRM            T1,4(P)
400025' 505 03 0 00 000000              HRLI            T3,.ENQPL
400026' 541 03 0 17 000001              HRRI            T3,1(P)
400027' 047 03 0 00 000151              ENQ,            T3,
                                          JRST          [ OUTSTR        [ ASCIZ /ENQ-MONREC/
                                                        PUSHJ           P,TRGOUT
                                                        EXIT
                                                        ]
400030' 254 00 0 00 401207'                             JRST    .+1
400031' 263 17 0 00 000000              POPJ    P,
400032' 200 04 0 17 000003              MOVE    T4,3(LP)
400033' 302 04 0 00 000000              DATE    T4,0
400034' 254 00 0 00 400045'             JRST    XITM2
400035' 505 03 0 00 400014'             HRLI    T3,MONREC
400036' 541 03 0 17 000001              HRRI    T3,1(P)
400037' 251 03 0 17 000005              BLT     T3,5(P)
400040' 542 01 0 17 000004              HRRM    T1,4(P)
400041' 505 03 0 00 000000              HRLI    T3,.DEQDR
400042' 541 03 0 17 000001              HRRI    T3,1(P)
400043' 047 03 0 00 000152              DEQ,    T3,
                                          JRST          [ OUTSTR        [ ASCIZ /DEQ-MONRE/
                                                        PUSHJ           P,TRGOUT
                                                        EXIT
                                                        ]
400044' 254 00 0 00 401214'                             JRST    .+1
400045' 263 17 0 00 000000              POPJ    P,
400046' 047 04 0 00 000073              WAKE    T4,
                                          JRST          [ OUTSTR        [ ASCIZ /WAKE/
                                                        PUSHJ           P,TRGOUT
                                                        EXIT
                                                        ]
400047' 254 00 0 00 401221'                             JRST    .+1
400050' 263 17 0 00 000000              POPJ    P,
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-1
FIVEPH  MAC     6-May-83 13:41

        400051' 332 00 0 02 000000          SKIPE       (T2)
                                              JRST      [ OUTSTR      [ ASCIZ /?WAIT EM FILA OCUPADA
/]
                                                        PUSHJ       P,TRGOUT
                                                        EXIT
                                                      ]
        400052' 254 00 0 00 401231'                   T3,
        400053' 047 03 0 00 000030          PJOB      T3,(T2)
        400054' 202 03 0 02 000000          MOVEM     T3,1
        400055' 211 03 0 00 000001          MOVNI     T3,
        400056' 047 03 0 00 000073          WAKE      [ OUTSTR      [ ASCIZ /?WAKE/]
                                                        PUSHJ       P,TRGOUT
                                                        EXIT
                                                      ]
        400057' 254 00 0 00 401221'                   T3,(HB,RWP!HB,RWT)
        400060' 515 03 0 00 000003          HRLZI     T3,
        400061' 047 03 0 00 000072          HIBER     [ OUTSTR      [ ASCIZ /?HIBER/]
                                                        PUSHJ       P,TRGOUT
                                                        EXIT
                                                      ]
        400062' 254 00 0 00 401236'                   P,XITMON
        400063' 260 17 0 00 400032'         PUSHJ     T3,(HB,RWP!HB,RWT)
        400064' 515 03 0 00 000003          HRLZI     T3,
        400065' 047 03 0 00 000072          HIBER     [ OUTSTR      [ ASCIZ /?HIBER/]
                                                        PUSHJ       P,TRGOUT
                                                        EXIT
                                                      ]
        400066' 254 00 0 00 401236'                   T3,(T2)
        400067' 200 03 0 02 000000          MOVE      (T2)
        400070' 402 00 0 02 000000          SETZM     T3,3(LP)
        400071' 202 03 0 07 000003          MOVEM     P,
        400072' 263 17 0 00 000000          POPJ      (T2)
        400073' 337 00 0 02 000000          SKIPG     P,
        400074' 263 17 0 00 000000          POPJ      T3,1
        400075' 211 03 0 00 000001          MOVNI     T3,
        400076' 047 03 0 00 000073          WAKE      [ OUTSTR      [ ASCIZ /?WAKE/]
                                                        PUSHJ       P,TRGOUT
                                                        EXIT
                                                      ]
        400077' 254 00 0 00 401221'                   T3,(HB,RWP!HB,RWT)
        400100' 515 03 0 00 000003          HRLZI     T3,
        400101' 047 03 0 00 000072          HIBER     [ OUTSTR      [ ASCIZ /?HIBER/]
                                                        PUSHJ       P,TRGOUT
                                                        EXIT
                                                      ]
        400102' 254 00 0 00 401236'                   T3,(T2)
        400103' 200 03 0 02 000000          MOVE      T4,
        400104' 047 04 0 00 000030          PJOB      T4,(T2)
        400105' 202 04 0 02 000000          MOVEM     T3,
        400106' 047 03 0 00 000073          WAKE      [ OUTSTR      [ ASCIZ /?WAKE/]
                                                        PUSHJ       P,TRGOUT
                                                        EXIT
                                                      ]
        400107' 254 00 0 00 401221'                   T3,(HB,RWP!HB,RWT)
        400110' 515 03 0 00 000003          HRLZI     T3,
        400111' 047 03 0 00 000072          HIBER     [ OUTSTR      [ ASCIZ /?HIBER/]
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-2
FIVEPH  MAC     6-May-83 13:41


                                                        PUSHJ       P,TRGOUT
                                                        EXIT
400112' 254 00 0 00 401236'                         ]
400113' 263 17 0 00 000000          POPJ            P,
400114' 200 10 0 17 000000                  MOVE        A1,(P)
400115' 505 11 0 00 000001                  HRLI        A2,T1
400116' 541 11 0 17 000000                  HRRI        A2,(P)
400117' 251 11 0 17 000001                  BLT         A2,^D1(P)
400120' 105 17 0 00 000002                  ADJSP       P,^D1+1
400121' 202 10 0 17 000000                  MOVEM       A1,(P)
400122' 261 17 0 00 000016                  PUSH        P,GB
400123' 261 17 0 00 000007                  PUSH        P,LB
400124' 201 07 0 17 777776                  MOVEI       LB,-2(P)
400125' 200 16 0 07 777776                  MOVE        GB,-<^D1+1>(LB)
400126' 201 10 0 00 000000                  MOVEI       A1,^D0
400127' 261 17 0 00 000010                  PUSH        P,A1
400130' 200 01 0 16 000000                  MOVE        T1,(GB)
400131' 260 17 0 00 400021'                 PUSHJ       P,EXITMON
400132' 047 10 0 00 000030          PJOB            A1,
400133' 554 10 0 10 401122'                 HLRZ        A1,TRJOB(A1)
400134' 201 11 0 16 000241                  MOVEI       A2,<^D161>(GB)
400135' 271 11 0 10 777777                  ADDI        A2,-1(A1)
400136' 336 00 0 11 000000                  SKIPN       (A2)
400137' 254 00 0 00 400144'                 JRST        REQ.01
400140' 200 01 0 16 000000                  MOVE        T1,(GB)
400141' 201 02 0 16 000265                  MOVEI       T2,<^D181>(GB)
400142' 271 02 0 10 777777                  ADDI        T2,-1(A1)
400143' 260 17 0 00 400051'                 PUSHJ       P,WAIT
400144' 476 00 0 11 000000          REQ.01: SETOM       (A2)
400145' 201 11 0 10 777777                  MOVEI       A2,-1(A1)
400146' 224 11 0 00 000020                  IMULI       A2,20
400147' 271 11 0 16 000001                  ADDI        A2,1(GB)
400150' 200 12 0 07 777777                  MOVE        A3,-<^D11*LB>
400151' 505 12 0 12 000000                  HRLI        A4,(A3)
400152' 541 13 0 11 000000                  HRRI        A4,(A2)
400153' 251 13 0 11 000017                  BLT         A4,<^D16-1>(A2)
400154' 350 00 0 16 000300                  AOS         <^D192>(GB)
400155' 200 13 1 07 777777                  MOVE        A4,@-<^D1>(LB)
400156' 242 13 0 00 777743                  LSH         A4,-^D29
400157' 302 13 0 00 000077                  CAIE        A4,"?"
400160' 254 00 0 00 400170'                 JRST        REQ.02
400161' 200 01 0 16 000000                  MOVE        T1,(GB)
400162' 201 02 0 16 000253                  MOVEI       T2,<^D171>(GB)
400163' 271 02 0 10 777777                  ADDI        T2,-1(A1)
400164' 260 17 0 00 400051'                 PUSHJ       P,WAIT
400165' 505 13 0 11 000000                  HRLI        A4,(A2)
400166' 541 13 0 12 000000                  HRRI        A4,(A3)
400167' 251 13 0 12 000017                  BLT         A4,<^D16-1>(A3)
400170' 200 01 0 16 000000          REQ.02: MOVE        T1,(GB)
400171' 260 17 0 00 400032'                 PUSHJ       P,XITMON
400172' 200 16 0 07 000001                  MOVE        GB,1(LB)
400173' 200 07 0 07 000002                  MOVE        LB,2(LB)
400174' 105 17 0 00 777772                  ADJSP       P,-<^D1+^D0+5>
400175' 254 00 1 17 000003                  JRST        @<^D1+2>(P)
400176' 200 10 0 17 000000                  MOVE        A1,(P)
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-3
FIVEPH  MAC     6-May-83 13:41

400177' 505 11 0 00 000001                      HRLI            A2,T1
400200' 541 11 0 17 000000                      HRRI            A2,(P)
400201' 251 11 0 17 000003                      BLT             A2,^D3(P)
400202' 105 17 0 00 000004                      ADJSP           P,^D3+1
400203' 202 10 0 17 000000                      MOVEM           A1,(P)
400204' 261 17 0 00 000016                      PUSH            P,GB
400205' 261 17 0 00 000007                      PUSH            P,LB
400206' 201 07 0 17 777776                      MOVEI           LB,-2(P)
400207' 200 16 0 07 777774                      MOVE            GB,-<^D3+1>(LB)
400210' 201 10 0 00 000000                      MOVEI           A1,^D0
400211' 261 17 0 00 000010                      PUSH            P,A1
400212' 200 01 0 16 000000                      MOVE            T1,(GB)
400213' 260 17 0 00 400021'                     PUSHJ           P,ENTMON
400214' 200 10 0 16 000277              MOVE    A1,<^D191>(GB)
400215' 306 10 0 00 000012              CAIN    A1,^D10
400216' 400 10 0 00 000000              SETZ    A1,
400217' 340 10 0 00 000000      ATN.01: ADJ     A1,
400220' 201 11 0 16 000241              MOVEI   A2,<^D161>(GB)
400221' 271 11 0 10 777777              ADDI    A2,-1(A1)
400222' 336 00 0 11 000000              SKIPN   (A2)
400223' 316 10 0 16 000277              CAMN    A1,<^D191>(GB)
400224' 254 00 0 00 400230'             JRST    ATN.02
400225' 301 10 0 00 000012              CAIL    A1,^D10
400226' 400 10 0 00 000000              SETZ    A1,
400227' 254 00 0 00 400217'             JRST    ATN.01
400230' 402 00 1 07 777777      ATN.02: SETZM   @-<^D1>(LB)
400231' 336 00 0 11 000000              SKIPN   (A2)
400232' 254 00 0 00 400252'             JRST    ATN.03
400233' 202 10 0 16 000277              MOVEM   A1,<^D191>(GB)
400234' 402 00 0 11 000000              SETZM   (A2)
400235' 202 10 1 07 777777              MOVEM   A1,@-<^D1>(LB)
400236' 201 11 0 10 777777              MOVEI   A2,-1(A1)
400237' 221 11 0 00 000020              IMULI   A2,^D16
400240' 271 11 0 16 000001              ADDI    A2,<^D1>(GB)
400241' 200 12 0 07 777775              MOVE    A3,-<^D3>(LB)
400242' 505 13 0 11 000000              HRLI    A4,(A3)
400243' 541 13 0 12 000000              HRRI    A4,<^D16-1>(A3)
400244' 251 13 0 12 000017              BLT     <^D192>(GB)
400245' 370 00 0 16 000300              SOS     T1,(GB)
400246' 200 01 0 16 000000              MOVE    T2,<^D191>(GB)
400247' 201 02 0 16 000265              MOVEI   T2,-1(A1)
400250' 271 02 0 10 777777              ADDI    P,SIGNAL
400251' 260 17 0 00 400073'             PUSHJ   A4,<^D192>(GB)
400252' 200 13 0 16 000300      ATN.03: MOVE    A4,3-<^D32>(LB)
400253' 202 13 1 07 777776              MOVEM   T1,(GB)
400254' 200 01 0 16 000000              MOVE    P,XITMON
400255' 260 17 0 00 400032'             PUSHJ   GB,1(LB)
400256' 200 16 0 07 000001              MOVE    LB,2(LB)
400257' 200 07 0 07 000002              MOVE    P,-<^D3+^D0+5>
400260' 105 17 0 00 777770              ADJSP   @<^D3+2>(P)
400261' 254 00 1 17 000005              JRST    A1,(P)
400262' 200 10 0 17 000000              MOVE    A2,T1
400263' 505 11 0 00 000001              HRLI    A2,(P)
400264' 541 11 0 17 000000              HRRI    A2,^D1(P)
400265' 251 11 0 17 000001              BLT
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-4
FIVEPH  MAC      6-May-83 12:41

400266' 105 17 0 00 000002                      ADJSP           P,^D1+1
400267' 202 10 0 17 000000                      MOVEM           A1,(P)
400270' 261 17 0 00 000016                      PUSH            P,GB
400271' 261 17 0 00 000007                      PUSH            P,LB
400272' 201 07 0 17 777776                      MOVEI           LB,-2(P)
400273' 200 16 0 07 777776                      MOVE            GB,-<^D1+1>(LB)
400274' 201 10 0 00 000000                      MOVEI           A1,^D0
400275' 261 17 0 00 000010                      PUSH            P,A1
400276' 200 01 0 16 000300                      MOVE            T1,(GB)
400277' 260 17 0 00 400021'                     PUSHJ           P,ENTMON
400300' 200 10 0 16 000277              MOVE            A1,<^D191>(GB)
400301' 200 11 0 07 777777              MOVE            A2,-<^D1>(LB)
400302' 201 12 0 10 777777              MOVEI           A3,-1(A1)
400303' 221 12 0 00 000020              IMULI           A3,^D16
400304' 271 12 0 16 000001              ADDI            A3,<^D1>(GB)
400305' 505 13 0 11 000000              HRLI            A4,(A2)
400306' 541 13 0 12 000000              HRRI            A4,(A3)
400307' 251 13 0 12 000017              BLT             A4,<^D16-1>(A3)
400310' 200 01 0 16 000000              MOVE            T1,(GB)
400311' 201 02 0 16 000253              MOVEI           T2,<^D171>(GB)
400312' 271 02 0 10 777777              ADDI            T2,-1(A1)
400313' 260 17 0 00 400073'             PUSHJ           P,SIGNAL
400314' 200 01 0 16 000000                      MOVE            T1,(GB)
400315' 260 17 0 00 400032'                     PUSHJ           P,XITMON
400316' 200 16 0 07 000001                      MOVE            GB,1(LB)
400317' 200 07 0 07 000002                      MOVE            LB,2(LB)
400320' 105 17 0 00 777772                      ADJSP           P,-<^D1+^D0+5>
400321' 254 00 1 17 000003                      JRST            @<^D1+2>(P)
400322' 200 10 0 16 000060                      MOVE            A1,(P)
400323' 505 11 0 00 000001                      HRLI            A2,T1
400324' 541 11 0 10 000000                      HRRI            A2,(P)
400325' 251 11 0 10 000000                      BLT             A2,^D0(P)
400326' 105 17 0 00 000001                      ADJSP           P,^D0+1
400327' 202 10 0 17 000000                      MOVEM           A1,(P)
400330' 261 17 0 00 000016                      PUSH            P,GB
400331' 261 17 0 00 000007                      PUSH            P,LB
400332' 201 07 0 17 777776                      MOVEI           LB,-2(P)
400333' 200 16 0 07 777777                      MOVE            GB,-<^D0+1>(LB)
400334' 201 10 0 00 000000                      MOVEI           A1,^D0
400335' 261 17 0 00 000010                      PUSH            P,A1
400336' 205 10 0 00 777766              MOVSI           A1,-^D10
400337' 541 10 0 16 000241              HRRI            A1,<^D161>(GB)
400340' 402 00 0 10 000000              SETZM           (A1)
400341' 253 10 0 00 400340'             AOBJN           A1,.-1
400342' 201 10 0 00 000012              MOVEI           A1,^D10
400343' 202 10 0 16 000277              MOVEM           A1,<^D191>(GB)
400344' 402 00 0 16 000300              SETZM           <^D192>(GB)
400345' 200 16 0 07 000001                      MOVE            GB,1(LB)
400346' 200 07 0 07 000002                      MOVE            LB,2(LB)
400347' 105 17 0 00 777773                      ADJSP           P,-<^D0+^D0+5>
400350' 254 00 1 17 000002                      JRST            @<^D0+2>(P)
400351' 106 111 116 101 114    FIN.0:  ASCIZ   /FINALIZANDO/
400352' 111 132 101 116 104
400353' 117 000 000 000 000
400354' 111 116 111 103 111    FIN.1:  ASCIZ   /INICIANDO/
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-5
FIVEPH  MAC     6-May-83 13:41

400355' 101 116 104 117 003
400356' 505 11 0 00 440700          HRLI    A2,440700
400357' 541 11 0 16 000003          HRRI    A2,^D3(GB)
400360' 134 12 0 00 000010          ILDB    A3,A1
400361' 136 12 0 00 000011          IDPB    A3,A2
400362' 302 12 0 00 000000          CAIE    A3,0
400363' 254 00 0 00 400360'         JRST    .-3
400364' 263 17 0 00 000000          POPJ    P,
400365' 200 10 0 17 000000                  MOVE    A1,(P)
400366' 505 11 0 00 000001                  HRLI    A2,T1
400367' 541 11 0 17 000000                  HRRI    A2,(P)
400370' 251 11 0 17 000000                  BLT     A2,^D0(P)
400371' 105 17 0 00 000001                  ADJSP   P,^D0+1
400372' 202 10 0 17 000000                  MOVEM   A1,(P)
400373' 261 17 0 00 000016                  PUSH    P,GB
400374' 261 17 0 00 000007                  PUSH    P,LB
400375' 201 07 0 17 777776                  MOVEI   LB,-2(P)
400376' 200 16 0 07 777777                  MOVE    GB,-^D0+1>(LB)
400377' 201 10 0 00 000000                  MOVEI   A1,^D0
400400' 261 17 0 00 000010                  PUSH    P,A1
400401' 200 01 0 16 000000                  MOVE    T1,(GB)
400402' 260 17 0 00 400021'                 PUSHJ   P,ENTMON
400403' 505 10 0 00 440700          HRLI    A1,440700
400404' 541 10 0 00 400351'         HRRI    A1,FIN.0
400405' 260 17 0 00 400356'         PUSHJ   P,SOF.00
400406' 200 01 0 16 000001          MOVE    T1,<^D1>(GB)
400407' 201 02 0 16 000003          MOVEI   T2,<^D3>(GB)
400410' 260 17 0 00 400114'         PUSHJ   P,REQ.0
400411' 370 00 0 16 000002          SOS     <^D2>(GB)
400412' 200 01 0 16 000000                  MOVE    T1,(GB)
400413' 260 17 0 00 400032'                 PUSHJ   P,XITMON
400414' 200 16 0 07 000001                  MOVE    GB,1(LB)
400415' 200 07 0 07 000002                  MOVE    LB,2(LB)
400416' 105 17 0 00 777773                  ADJSP   P,-<^D0+^D0+5>
400417' 254 00 1 17 000002                  JRST    @<^D0+2>(P)
400420' 200 10 0 17 000000                  MOVE    A1,(P)
400421' 505 11 0 00 000001                  HRLI    A2,T1
400422' 541 11 0 17 000000                  HRRI    A2,(P)
400423' 251 11 0 17 000000                  BLT     A2,^D0(P)
400424' 105 17 0 00 000001                  ADJSP   P,^D0+1
400425' 202 10 0 17 000000                  MOVEM   A1,(P)
400426' 261 17 0 00 000016                  PUSH    P,GB
400427' 261 17 0 00 000007                  PUSH    P,LB
400430' 201 07 0 17 777776                  MOVEI   LB,-2(P)
400431' 200 16 0 07 777777                  MOVE    GB,-<^D0+1>(LB)
400432' 201 10 0 00 000000                  MOVEI   A1,^D0
400433' 261 17 0 00 000010                  PUSH    P,A1
400434' 205 10 0 00 777777                  MOVSI   A1,-^D1
400435' 541 10 0 17 000001                  HRRI    A1,1(P)
400436' 402 00 0 10 000000                  SETZM   (A1)
400437' 253 10 0 00 400436'                 AOBJN   A1,.-1
400440' 105 17 0 00 000001                  ADJSP   P,^D1
400441' 200 01 0 16 000000                  MOVE    T1,(GB)
400442' 260 17 0 00 400021'                 PUSHJ   P,ENTMON
400443' 336 00 0 16 000002          SKIPN   <^D2>(GB)
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-6
FIVEPH  MAC     6-May-83 13:41

400444' 476 00 0 07 000004          SETOM           <^D4>(LB)
400445' 200 01 0 16 000000                  MOVE            T1,(GB)
400446' 260 17 0 00 400032'                 PUSHJ           P,XITMON
400447' 200 01 0 07 000004                  MOVE            T1,4(LB)
400450' 200 16 0 07 000001                  MOVE            GB,1(LB)
400451' 200 07 0 07 000002                  MOVE            LB,2(LB)
400452' 105 17 0 00 777772                  ADJSP           P,-<^D0+^D1+5>
400453' 254 00 1 17 000002                  JRST            @<^D0+2>(P)
400454' 200 10 0 17 000000                  MOVE            A1,(P)
400455' 505 11 0 00 000001                  HRLI            A2,T1
400456' 541 11 0 17 000000                  HRRI            A2,(P)
400457' 251 11 0 17 000000                  BLT             A2,^D0(P)
400460' 105 17 0 00 000001                  ADJSP           P,^D0+1
400461' 202 10 0 17 000000                  MOVEM           A1,(P)
400462' 261 17 0 00 000016                  PUSH            P,GB
400463' 261 17 0 00 000007                  PUSH            P,LB
400464' 201 07 0 17 777776                  MOVEI           LB,-2(P)
400465' 200 16 0 07 777777                  MOVE            GB,-<^D0+1>(LB)
400466' 201 10 0 00 000000                  MOVEI           A1,^D0
400467' 261 17 0 00 000010                  PUSH            P,A1
400470' 205 10 0 00 777777                  MOVSI           A1,-^D1
400471' 541 10 0 17 000001                  HRRI            A1,1(P)
400472' 402 00 0 10 000000                  SETZM           (A1)
400473' 253 10 0 00 400472'                 AOBJN           A1,.-1
400474' 105 17 0 00 000001                  ADJSP           P,^D1
400475' 200 01 0 16 000000                  MOVE            T1,(GB)
400476' 260 17 0 00 400021'                 PUSHJ           P,ENTMON
400477' 200 10 0 16 000002          MOVE            A1,<^D2>(GB)
400500' 306 10 0 00 000005          CAIN            A1,^D5
400501' 254 00 0 00 400506'         JRST            .+5
400502' 200 01 0 16 000000          MOVE            T1,(GB)
400503' 201 02 0 16 000023          MOVEI           T2,<^D19>(GB)
400504' 260 17 0 00 400051'         PUSHJ           P,WAIT
400505' 254 00 0 00 400477'         JRST            .-6
400506' 476 00 0 07 000004          SETOM           <^D4>(LB)
400507' 200 01 0 16 000000                  MOVE            T1,(GB)
400510' 260 17 0 00 400032'                 PUSHJ           P,XITMON
400511' 200 01 0 07 000004                  MOVE            T1,4(LB)
400512' 200 16 0 07 000001                  MOVE            GB,1(LB)
400513' 200 07 0 07 000002                  MOVE            LB,2(LB)
400514' 105 17 0 00 777772                  ADJSP           P,-<^D0+^D1+5>
400515' 254 00 1 17 000002                  JRST            @<^D0+2>(P)
400516' 200 10 0 17 000000                  MOVE            A1,(P)
400517' 505 11 0 00 000001                  HRLI            A2,T1
400520' 541 11 0 17 000000                  HRRI            A2,(P)
400521' 251 11 0 17 000000                  BLT             A2,^D0(P)
400522' 105 17 0 00 000001                  ADJSP           P,^D0+1
400523' 202 10 0 17 000000                  MOVEM           A1,(P)
400524' 261 17 0 00 000016                  PUSH            P,GB
400525' 261 17 0 00 000007                  PUSH            P,LB
400526' 201 07 0 17 777776                  MOVEI           LB,-2(P)
400527' 200 16 0 07 777777                  MOVE            GB,-<^D0+1>(LB)
400530' 201 10 0 00 000000                  MOVEI           A1,^D0
400531' 261 17 0 00 000010                  PUSH            P,A1
400532' 200 01 0 16 000000                  MOVE            T1,(GB)
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-7
FIVEPH  MAC     6-May-83 13:41

400533' 260 17 0 00 400021'                     PUSHJ   P,ENTMON
400534' 505 10 0 00 440700              HRLI    A1,440700
400535' 541 10 0 00 400354'             HRRI    A1,FIN.1
400536' 260 17 0 00 400356'             PUSHJ   P,SGF.00
400537' 200 01 0 16 000001              MOVE    T1,(TB)
400540' 201 02 0 16 000002              MOVEI   T2,2(TB)
400541' 260 17 0 00 400114'             PUSHJ   P,REG.0
400542' 350 00 0 16 000002              AOS     (TP)
400543' 200 01 0 16 000000              MOVE    T1,(TP)
400544' 201 02 0 16 000023              MOVEI   T2,23(TB)
400545' 260 17 0 00 400071'             PUSHJ   P,SIGNAL
400546' 200 01 0 16 000000              MOVE    T1,(TB)
400547' 260 17 0 00 400332'             PUSHJ   P,(T1)
400550' 200 16 0 07 000001              MOVE    TB,1(LB)
400551' 200 07 0 07 000002              MOVE    LB,2(LB)
400552' 105 17 0 00 777773              ADJSP   P,-5
400553' 254 00 1 17 000002              JRST    2(T4)
400554' 200 10 0 17 000000              MOVE    A1,(P)
400555' 505 11 0 00 000001              HRLI    A2,1
400556' 541 11 0 17 000000              HRRI    A2,(P)
400557' 251 11 0 17 000000              SKIP    A2,0(P)
400560' 105 17 0 00 000001              ADJSP   P,1
400561' 202 10 0 17 000000              MOVEM   A1,(P)
400562' 261 17 0 00 000016              PUSH    P,TB
400563' 261 17 0 00 000007              PUSH    P,LB
400564' 201 07 0 17 777776              MOVEI   LB,-2(P)
400565' 200 16 0 07 777777              MOVE    TB,-1(LB)
400566' 201 10 0 00 000000              MOVEI   A1,0
400567' 261 17 0 00 000010              PUSH    P,A1
400570' 402 00 0 16 000002      SETZM   @(TB)(TB)
400571' 200 16 0 07 000001              MOVE    TB,1(LB)
400572' 200 07 0 07 000000              MOVE    LB,0(LB)
400573' 105 17 0 00 777775              ADJSP   P,-3
400574' 254 00 1 17 000002              JRST    @(T4)(TB)
400575' 000000  001751          XWD     0,1001
401076' 000000  001752          XWD     0,1002
401205' 077 105 116 124 115
401206' 117 116 000 000 000
401207' 051 03 0 00 401205'
401210' 260 17 0 00 400000'
401211' 047 00 0 00 000012
401212' 077 130 111 124 115
401213' 117 116 020 000 000
401214' 051 03 0 00 401213'
401215' 260 17 0 00 400005'
401216' 047 00 0 00 000012
401217' 077 127 101 113 105
401220' 000 000 000 000 000
401221' 051 03 0 00 401217'
401222' 260 17 0 00 400000'
401223' 047 00 0 00 000012
401224' 077 127 101 111 124
401225' 040 105 115 040 106
401226' 111 114 101 040 117
401227' 103 125 120 101 104
```

```
FIVEPH  MACRO %534(1152) 13:43  6-May-83 Page 1-8
FIVEPH  MAC     6-May-83 13:41

        401230' 101 000 000 000 000
        401231' 051 03 0 00 401224'
        401232' 260 17 0 00 400000'
        401233' 047 00 0 00 000012
        401234' 077 110 111 102 105
        401235' 122 000 000 000 000
        401236' 051 03 0 00 401234'
        401237' 260 17 0 00 400000'
        401240' 047 00 0 00 000012
        401241' 116 101 117 040 124     CT.1:   ASCIZ   /NAO TENHO GARFOS PARA COMER, ENTAO DUSMO/
        401242' 105 114 110 117 040
        401243' 107 101 122 106 117
        401244' 123 040 120 101 122
        401245' 101 040 103 117 115
        401246' 105 122 054 040 105
        401247' 116 124 101 117 040
        401250' 104 125 122 115 117
        401251' 000 000 000 000 000
        401252' 126 117 125 040 124     CT.2:   ASCIZ   /VOU TENTAR PEGAR OS GARFOS/
        401253' 105 116 124 101 122
        401254' 040 120 105 107 101
        401255' 122 040 117 123 040
        401256' 107 101 122 106 117
        401257' 123 000 000 000 000
        401260' 103 117 116 123 105     CT.3:   ASCIZ   /CONSEGUI OS GARFOS/
        401261' 107 125 111 040 117
        401262' 123 040 107 101 122
        401263' 106 117 123 000 000
        401264'                         R.1:
                                                EPROCE          ^D1,^D0,^D0,^D0^XALL
                                                        MOVE            A1,(P)
        401264' 200 10 0 17 000000              HRLI            A2,T1
        401265' 505 11 0 00 000001              HRRI            A2,(P)
        401266' 541 11 0 17 000000              BLT             A2,^D1(P)
        401267' 251 11 0 17 000001              ADJSP           P,^D1+1
        401270' 105 17 0 00 000002              MOVEM           A1,(P)
        401271' 202 10 0 17 000000              PUSH            P,GB
        401272' 261 17 0 00 000016              PUSH            P,LB
        401273' 261 17 0 00 000007              MOVEI           LB,-2(P)
        401274' 201 07 0 17 777776              MOVE            GB,-<^D1+1>(LB)
        401275' 200 16 0 07 777776              MOVEI           A1,^D0
        401276' 201 10 0 00 000000              PUSH            P,A1
        401277' 261 17 0 00 000010      EMONIT^XALL
                                                        MOVE            T1,(GB)
        401300' 200 01 0 16 000000              PUSHJ           P,ENTMON
        401301' 260 17 0 00 400021'
        401302'                         L.1:    CTWSTR          CT.2^XALL
        401303' 201 01 0 00 400575'             MOVEI           T1,CTY.V
        401304' 201 02 0 00 401274'             MOVEI           T2,CT.2
        401305' 260 17 0 00 400114'             PUSHJ           P,REQ.0
        401306' 200 10 0 07 777777              MOVE            A1,-<^D1>(LB)
        401307' 271 10 0 16 000001              ADDI            A1,<^D1>(GB)
        401310' 200 11 0 10 000000              MOVE            A2,(A1)
        401311' 301 11 0 00 000002              CAIL            A2,^D2
                                                GOTO            L.2^
        401312' 254 00 0 00 401344'             JRST            L.2
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-9
FIVEPH  MAC     6-May-83 13:41


401334' 201 01 0 00 400575'                     CTWSTR  CT,1^XALL
401335' 201 02 0 00 401263'                     MOVEI   T1,CTY.V
401336' 260 17 0 00 400114'                     MOVEI   T2,CT.1
401337' 200 01 0 16 000000                      PUSHJ   P,REQ.0
401340' 200 02 0 07 777777                      MOVE    T1,(GB)
401341' 271 02 0 16 000006                      MOVE    T2,-^D1(LB)
                                                ADDI    T2,^D6(GB)
401342' 260 17 0 00 400051'                     CALL    WAIT
                                                PUSHJ   P,WAIT
401343' 254 00 0 00 401324'                     GOTO    L.1^
401344' 200 10 0 07 777777      L.2:            JRST    L.1
                                                MOVE    A1,-^D1(LB)
401345' 360 10 0 00 000000                      DEC     A1^
401346' 305 10 0 00 000000                      SOJ     A1,
401347' 201 10 0 00 000004                      CAIGE   A1,^D0
401350' 271 10 0 16 000001                      MOVEI   A1,^D4
                                                ADDI    A1,^D1(GB)
401351' 370 00 0 10 000000                      DECM    ^D0(A1)^
401352' 200 10 0 07 777777                      SOS     ^D0(A1)
                                                MOVE    A1,-^D1(LB)
401353' 340 10 0 00 000000                      INC     A1^
401354' 303 10 0 00 000004                      AOJ     A1,
401355' 201 10 0 00 000000                      CAILE   A1,^D4
401356' 271 10 0 16 000001                      MOVEI   A1,^D0
                                                ADDI    A1,^D1(GB)
401357' 370 00 0 10 000000                      DECM    ^D0(A1)^
                                                SOS     ^D0(A1)
401360' 201 01 0 00 400575'                     CTWSTR  CT,3^XALL
401361' 201 02 0 00 401302'                     MOVEI   T1,CTY.V
401362' 260 17 0 00 400114'                     MOVEI   T2,CT.1
                                                PUSHJ   P,REQ.0
401363' 200 01 0 16 000000      XWAIT^XALL:     MOVE    T1,(GB)
401364' 260 17 0 00 400032'                     PUSHJ   P,XPROCE

401365' 200 16 0 07 000001      XPROCE: ^D1,^D0,^D00^XALL
401366' 200 07 0 07 000002                      MOVE    GB,1(LB)
401367' 105 17 0 00 777770                      MOVE    LB,2(LB)
401370' 254 00 1 17 000003                      ADJSP   P,-(^D1+^D10+5)
401371'                                         JRST    @(^D1+2)(6)
                                R.2:
401371' 200 10 0 17 000000      EPROCE: ^D1,^D2,^D0,^D00^XALL
401372' 505 11 0 00 000001                      MOVE    A1,(P)
401373' 541 11 0 17 000000                      HRLI    A2,T1
401374' 251 11 0 17 000001                      HRRI    A2,(P)
401375' 105 17 0 00 000002                      BLT     A2,^D1(P)
401376' 202 10 0 17 000000                      ADJSP   P,^D1+1
401377' 261 17 0 00 000011                      MOVEM   A1,(P)
401400' 261 17 0 00 000007                      PUSH    P,GB
401401' 201 07 0 17 777776                      PUSH    P,LB
401402' 200 16 0 07 777776                      MOVEI   LB,-2(P)
401403' 201 10 0 00 000000                      MOVE    GB,-(^D1+1)(LB)
401404' 261 17 0 00 000010                      MOVEI   A1,^D0
401405' 205 10 0 00 777776                      PUSH    P,A1
401406' 541 10 0 17 000001                      MOVSI   A1,-^D2
                                                HRRI    A1,1(P)
```

```
FIVEPH  MACRO %53A(1152) 13:47  6-May-83 Page 1-10
FIVEPH  MAC     6-May-83 13:41

401407' 402 00 0 10 000000                      SETZM           (A1)
401410' 253 10 0 00 401407'                     ACBJN           A1,,-1
401411' 105 17 0 00 000002                      ADJSP           P,^D2
                                        EMONIT^XALL
401412' 200 01 0 16 000000                              MOVE            T1,(GB)
401413' 260 17 0 00 400021'                             PUSHJ           P,ENTMON
401414' 200 10 0 07 777777              MOVE            A1,-<^D1>(LB)
                                        DEC             A1^
401415' 360 10 0 00 000000              SOJ             A1,
401416' 305 10 0 00 000000              CAIGE           A1,^D0
401417' 201 10 0 00 000004              MOVEI           A1,^D4
401420' 202 10 0 07 000005              MOVEM           A1,<^D5>(LB)
401421' 271 10 0 16 000001              ADDI            A1,<^D1>(GB)
                                        INCM            ^D0(A1)^
401422' 350 00 0 10 000000              AOS             ^D0(A1)
401423' 200 01 0 16 000000              MOVE            T1,(GB)
401424' 200 02 0 07 000005              MOVE            T2,<^D5>(LB)
401425' 271 02 0 16 000006              ADDI            T2,<^D6>(GB)
                                        CALL            SIGNAL^
401426' 260 17 0 00 400073'             PUSHJ           P,SIGNAL
401427' 200 10 0 07 777777              MOVE            A1,-<^D1>(LB)
                                        INC             A1^
401430' 340 10 0 00 000000              AOJ             A1,
401431' 303 10 0 00 000004              CAILE           A1,^D4
401432' 201 10 0 00 000000              MOVEI           A1,^D0
401433' 202 10 0 07 000004              MOVEM           A1,<^D4>(LB)
401434' 271 10 0 16 000001              ADDI            A1,<^D1>(GB)
                                        INCM            ^D0(A1)^
401435' 350 00 0 10 000000              AOS             ^D0(A1)
401436' 200 01 0 16 000000              MOVE            T1,(GB)
401437' 200 02 0 07 000004              MOVE            T2,<^D4>(LB)
401440' 271 02 0 16 000006              ADDI            T2,<^D6>(GB)
                                        CALL            SIGNAL^
401441' 260 17 0 00 400073'             PUSHJ           P,SIGNAL
                                        XMONIT^XALL
401442' 200 01 0 16 000000                              MOVE            T1,(GB)
401443' 260 17 0 00 400032'                             PUSHJ           P,XITMON
                                        XPROCE          ^D1,^D2,^D0^XALL
401444' 200 16 0 07 000001                              MOVE            GB,1(LB)
401445' 200 07 0 07 000002                              MOVE            LB,2(LB)
401446' 105 17 0 00 777770                              ADJSP           P,-<^D1+^D2+5>
401447' 254 00 1 17 000003                              JRST            @<^D1+2>(P)
401450'                                 I.B.1:
                                        EPROCE          ^D0,^D0,^D0,^D0^XALL
401450' 200 10 0 17 000000                              MOVE            A1,(P)
401451' 505 11 0 00 000001                              HRLI            A2,T1
401452' 541 11 0 17 000000                              HRRI            A2,(P)
401453' 251 11 0 17 000000                              BLT             A2,^D0(P)
401454' 105 17 0 00 000001                              ADJSP           P,^D0+1
401455' 202 10 0 17 000000                              MOVEM           A1,(P)
401456' 261 17 0 00 000016                              PUSH            P,GB
401457' 261 17 0 00 000007                              PUSH            P,LB
401460' 201 07 0 17 777776                              MOVEI           LB,-2(P)
401461' 200 16 0 07 777777                              MOVE            GB,-<^D0+1>(LB)
401462' 201 10 0 00 000000                              MOVEI           A1,^D0
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-11
FIVEPH  MAC     6-May-83 13:41

401463' 261 17 0 00 000010                      PUSH    P,A1
401464' 205 10 0 00 777773              MOVSI   A1,-^D5
401465' 541 10 0 16 000001              HRRI    A1,<^D1>(GB)
401466' 201 11 0 00 000002              MOVEI   A2,^D2
401467' 202 11 0 10 000000              MOVEM   A2,(A1)
401470' 253 10 0 00 401467'             AOBJN   A1,.-1
                                        XPROCE  ^D0,^D0,^D0^XALL
401471' 200 16 0 07 000001                      MOVE    GB,1(LB)
401472' 200 07 0 07 000002                      MOVE    LB,2(LB)
401473' 105 17 0 00 777773                      ADJSP   P,-<^D0+^D0+5>
401474' 254 00 1 17 000002                      JRST    @<^D0+2>(P)
401475' 105 123 124 117 125    CT.4:    ASCIZ   /ESTOU OCUPADO COMENDO MACARRAO/
401476' 040 117 103 125 120
401477' 101 104 117 040 103
401500' 117 115 105 116 104
401501' 117 040 115 101 103
401502' 101 122 122 101 117
401503' 000 000 000 000 000
401504' 116 101 117 040 101    CT.5:    ASCIZ   /NAO ABORRECA ESTOU PENSANDO/
401505' 102 117 122 122 105
401506' 103 101 040 105 123
401507' 124 117 125 040 120
401510' 105 116 123 101 116
401511' 104 117 000 000 000
401512'                         R.3:
                                        XPROCE  ^D0,^D0,^D0,^D0^XALL
401512' 200 10 0 17 000000              MOVE    A1,(P)
401513' 505 11 0 00 000001              HLLI    A2,T1
401514' 541 11 0 17 000000              HRRI    A2,(P)
401515' 251 11 0 17 000000              BLT     A2,^D0-1
401516' 105 17 0 00 000001              ADJSP   P,^D0+1
401517' 202 10 0 17 000000              MOVEM   A1,(P)
401520' 261 17 0 00 000016              PUSH    P,GB
401521' 261 17 0 00 000007              PUSH    P,LB
401522' 201 07 0 17 777776              MOVEI   LB,-2(P)
401523' 200 16 0 07 777777              MOVE    GB,-<^D0+1>(LB)
401524' 201 10 0 00 000000              MOVEI   A1,^D0
401525' 261 17 0 00 000010              PUSH    P,A1
401526' 047 10 0 00 000030      PJOB    A1,
401527' 200 10 0 10 401122'             MOVE    A1,TBJOB(A1)
401530' 556 10 0 16 000003              HLLZM   A1,<^D3>(GB)
                                        DECM    <^D3>(GB)
401531' 370 00 0 16 000003              SOS     ^D3(GB)
401532' 200 01 0 16 000001      L.3:    MOVE    T1,<^D1>(GB)
401533' 200 02 0 16 000003              MOVE    T2,<^D3>(GB)
401534' 260 17 0 00 401306'             CALL    R,1^
                                        PUSHJ   P,R.1
401535' 201 01 0 00 400575'             CTLSTR  CT.4^XALL
401536' 201 02 0 00 401475'             MOVEI   T1,CTY.U
401537' 260 17 0 00 400114'             MOVEI   T2,CT.4
401540' 200 01 0 16 000001              PUSHJ   P,REQ.0
401541' 200 02 0 16 000003              MOVE    T1,<^D1>(GB)
                                        MOVE    T2,<^D3>(GB)
401542' 260 17 0 00 401371'             CALL    R,2^
                                        PUSHJ   P,R.2
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-12
FIVEPH  MAC     6-May-83 13:41

                                        CTWSTR          CT,5^XALL
401543' 201 01 0 00 400575'             MOVEI           T1,CTY,V
401544' 201 02 0 00 401504'             MOVEI           T2,CT,5
401545' 260 17 0 00 400114'             PUSHJ           P,REQ,0
401546' 372 00 0 16 000002              SOSE            <^D2>(GB)
                                        GOTO            L,3^
401547' 254 00 0 00 401532'             JRST            L,2
                                        XPROCE          ^D0,^D0,^D0^XALL
401550' 200 16 0 07 000001                      MOVE            GB,1(LB)
401551' 200 07 0 07 000002                      MOVE            LB,2(LB)
401552' 105 17 0 00 777773                      ADJSP           P,-<^D0+2>(P)
401553' 254 00 1 17 000002                      JRST            @<^D0+2>(P)
401554'                                 I.P.2:
                                        EPROCE          ^D0,^D0,^D0,^D0^XALL
401554' 200 10 0 17 000000                      MOVE            A1,(P)
401555' 505 11 0 00 000001                      HRLI            A2,T1
401556' 541 11 0 17 000000                      HRRI            A2,(P)
401557' 251 11 0 17 000000                      BLT             A2,^D0(P)
401560' 105 17 0 00 000001                      ADJSP           P,^D0+1
401561' 202 10 0 17 000000                      MOVEM           A1,(P)
401562' 261 17 0 00 000016                      PUSH            P,GB
401563' 261 17 0 00 000007                      PUSH            P,LB
401564' 201 07 0 17 777776                      MOVEI           LB,-2(P)
401565' 200 16 0 07 777777                      MOVE            GB,-<^D0+1>(LB)
401566' 201 10 0 00 000000                      MOVEI           A1,^D0
401567' 261 17 0 00 000010                      PUSH            P,A1
401570' 201 10 0 00 000012              MOVEI           A1,^D10
401571' 202 10 0 16 000002              MOVEM           A1,<^D2>(GB)
                                        XPROCE          ^D0,^D0,^D0^XALL
401572' 200 16 0 07 000001                      MOVE            GB,1(LB)
401573' 200 07 0 07 000002                      MOVE            LB,2(LB)
401574' 105 17 0 00 777773                      ADJSP           P,-<^D0+^D0+5>
401575' 254 00 1 17 000002                      JRST            @<^D0-2>(P)
401576'                                 M.0:
401576' 000000  000001                  XWD             0,^D1
401577'                                 BLOCK           ^D0
401577'                                 BLOCK           ^D5
401604'                                 BLOCK           ^D5
                                        INILOW          'FIVEPH'^XALL
000000' 051 03 0 00 000000      COUTST: OUTSTR          0
000001' 200 10 0 17 000000                      MOVE            A1,(P)
000002' 505 11 0 00 000001                      HRLI            A2,T1
000003' 541 11 0 17 000000                      HRRI            A2,(P)
000004' 251 11 0 17 000002                      BLT             A2,^D2(P)
000005' 105 17 0 00 000003                      ADJSP           P,^D2+1
000006' 202 10 0 17 000000                      MOVEM           A1,(P)
000007' 261 17 0 00 000016                      PUSH            P,GB
000010' 261 17 0 00 000007                      PUSH            P,LB
000011' 201 07 0 17 777776                      MOVEI           LB,-2(P)
000012' 200 16 0 07 777775                      MOVE            GB,-<^D2+1>(LB)
000013' 201 10 0 00 000000                      MOVEI           A1,^D0
000014' 261 17 0 00 000010                      PUSH            P,A1
000015' 505 10 0 00 440700              HRLI            A1,440700
000016' 541 10 0 16 000001              HRRI            A1,^D1(GB)
000017' 200 11 0 07 777777              MOVE            A2,-<^D1>(LB)
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-13
FIVEPH  MAC     6-May-83 13:41

000020' 134 12 0 07 777776      TTW.01: ILDB    A3,-<^D2>(LB)
000021' 136 12 0 00 000010              IDPB    A3,A1
000022' 302 12 0 00 000000              CAIE    A3,0
000023' 367 11 0 00 000020'             SOJG    A2,TTW.01
000024' 336 00 0 00 000011              SKIPN   A2
000025' 136 11 0 00 000010              IDPB    A2,A1
000026' 201 11 0 16 000001              MOVEI   A2,<^D1>(GB)
000027' 434 11 0 00 000000'             IOR     A2,COLTST
000030' 256 00 0 00 000011              XCT     A2
000031' 200 16 0 07 000001                      MOVE    GB,1(LB)
000032' 200 07 0 07 000002                      MOVE    LB,2(LB)
000033' 105 17 0 00 777771                      ADJSP   P,-<^D2+^D0+5>
000034' 254 00 1 17 000004                      JRST    @<^D1+2>(P)
000035' 200 10 0 17 000000                      MOVE    A1,(P)
000036' 505 11 0 00 000001                      HRLI    A2,T1
000037' 541 11 0 17 000000                      HRRI    A2,(P)
000040' 251 11 0 17 000001                      BLT     A2,^D1(P)
000041' 105 17 0 00 000002                      ADJSP   P,^D1+1
000042' 202 10 0 17 000000                      MOVEM   A1,(P)
000043' 261 17 0 00 000005                      PUSH    P,GB
000044' 261 17 0 00 000007                      PUSH    P,LB
000045' 201 07 0 17 777776                      MOVEI   LB,-2(P)
000046' 200 16 0 07 777776                      MOVE    GB,-<^D1+1>(LB)
000047' 201 10 0 00 000000                      MOVEI   A1,^D0
000050' 261 17 0 00 000010                      PUSH    P,A1
000051' 400 12 0 00 000000              SETZ    A3,
000052' 505 13 0 00 440700              HRLI    A4,440700
000053' 541 13 0 16 000001              HRRI    A4,^D1(GB)
000054' 200 10 0 07 777777              MOVE    A1,-<^D1>(LB)
000055' 325 10 0 00 000061'             JUMPGE  A1,TTW.11
000056' 201 11 0 00 000055              MOVEI   A2,"-"
000057' 136 11 0 00 000013              IDPB    A2,A4
000060' 210 10 0 00 000010              MOVN    A1,A1
000061' 231 10 0 00 000012      TTW.11: IDIVI   A1,^D10
000062' 261 17 0 00 000011              PUSH    P,A2
000063' 340 12 0 00 000000              ADJ     A3,
000064' 327 10 0 00 000061'             JUMPG   A1,TTW.11
000065' 262 17 0 00 000011      TTW.12: POP     P,A2
000066' 435 11 0 00 000060              IORI    A2,"0"
000067' 136 11 0 00 000013              IDPB    A2,A4
000070' 367 12 0 00 000065'             SOJG    A3,TTW.12
000071' 136 12 0 00 000013              IDPB    A3,A4
000072' 201 12 0 16 000001              MOVEI   A3,<^D1>(GB)
000073' 434 12 0 00 000000'             IOR     A3,COLTST
000074' 256 00 0 00 000012              XCT     A3
000075' 200 16 0 07 000001                      MOVE    GB,1(LB)
000076' 200 07 0 07 000002                      MOVE    LB,2(LB)
000077' 105 17 0 00 777772                      ADJSP   P,-<^D1+^D0+5>
000100' 254 00 1 17 000003                      JRST    @<^D1+2>(P)
000101' 200 10 0 17 000000                      MOVE    A1,(P)
000102' 505 11 0 00 000001                      HRLI    A2,T1
000103' 541 11 0 17 000000                      HRRI    A2,(P)
000104' 251 11 0 17 000001                      BLT     A2,^D1(P)
000105' 105 17 0 00 000002                      ADJSP   P,^D1+1
000106' 202 10 0 17 000000                      MOVEM   A1,(P)
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-14
FIVEPH  MAC     6-May-83 13:41

000107' 261 17 0 00 000016                      PUSH            P,GB
000110' 261 17 0 00 000007                      PUSH            P,LB
000111' 201 07 0 17 777776                      MOVEI           LB,-2(P)
000112' 200 16 0 07 777776                      MOVE            GB,-<^D1+1>(LB)
000113' 201 10 0 00 000000                      MOVEI           A1,^D0
000114' 261 17 0 00 000010                      PUSH            P,A1
000115' 200 10 0 07 777777              MOVE            A1,-<^D1>(LB)
000116' 051 01 0 00 000010              OUTCHR          A1
000117' 200 16 0 07 000001                      MOVE            GB,1(LB)
000120' 200 07 0 07 000002                      MOVE            LB,2(LB)
000121' 105 17 0 00 777772                      ADJSP           P,-<^D1+^D0+5>
000122' 254 00 1 17 000003                      JRST            @<^D1+^D2>(P)
000123' 200 10 0 17 000000                      MOVE            A1,(P)
000124' 505 11 0 00 000001                      HRLI            A2,T1
000125' 541 11 0 17 000000                      HRRI            A2,(P)
000126' 251 11 0 17 000000                      BLT             A2,^D0(P)
000127' 105 17 0 00 000001                      ADJSP           P,^D0+1
000130' 202 10 0 17 000000                      MOVEM           A1,(P)
000131' 261 17 0 00 000016                      PUSH            P,GB
000132' 261 17 0 00 000007                      PUSH            P,LB
000133' 201 07 0 17 777776                      MOVEI           LB,-2(P)
000134' 200 16 0 07 777777                      MOVE            GB,-<^D0+1>(LB)
000135' 201 10 0 00 000000                      MOVEI           A1,^D0
000136' 261 17 0 00 000010                      PUSH            P,A1
000137' 201 10 0 00 000012              MOVEI           A1,12
000140' 051 01 0 00 000010              OUTCHR          A1
000141' 201 10 0 00 000015              MOVEI           A1,15
000142' 051 01 0 00 000010              OUTCHR          A1
000143' 200 16 0 07 000001                      MOVE            GB,1(LB)
000144' 200 07 0 07 000002                      MOVE            LB,2(LB)
000145' 105 17 0 00 777773                      ADJSP           P,-<^D0+^D2+5>
000146' 254 00 1 17 000002                      JRST            @<^D0+^D2>(P)
000147' 200 10 0 17 000000                      MOVE            A1,(P)
000150' 505 11 0 00 000001                      HRLI            A2,T1
000151' 541 11 0 17 000000                      HRRI            A2,(P)
000152' 251 11 0 17 000002                      BLT             A2,^D2(P)
000153' 105 17 0 00 000003                      ADJSP           P,^D2+1
000154' 202 10 0 17 000000                      MOVEM           A1,(P)
000155' 261 17 0 00 000016                      PUSH            P,GB
000156' 261 17 0 00 000007                      PUSH            P,LB
000157' 201 07 0 17 777776                      MOVEI           LB,-2(P)
000160' 200 16 0 07 777775                      MOVE            GB,-<^D2+1>(LB)
000161' 201 10 0 00 000000                      MOVEI           A1,^D0
000162' 261 17 0 00 000010                      PUSH            P,A1
000163' 400 11 0 00 000000              SETZ            A2,
000164' 051 00 0 00 000012      TTR.01: INCHRW          A3
000165' 305 12 0 00 000040              CAIGE           A3,' '
000166' 254 00 0 00 000172'             JRST            TTR.02
000167' 136 12 0 07 777776              IDPB            A3,-<^D2>(LB)
000170' 340 11 0 00 000000              AOJ             A2,
000171' 254 00 0 00 000164'             JRST            TTR.01
000172' 201 12 0 00 000000      TTR.02: MOVEI           A3,0
000173' 136 12 0 07 777776              IDPB            A3,-<^D2>(LB)
000174' 202 11 1 07 777777              MOVEM           A2,@-<^D1>(LB)
000175' 200 16 0 07 000001                      MOVE            GB,1(LB)
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-15
FIVEPH  MAC     6-May-83 13:41

000176' 200 07 0 07 000002                      MOVE            LP,2(LP)
000177' 105 17 0 00 777771                      ADJSP           P,-(%D2+%D45)
000200' 254 00 1 17 000004                      JRST            @(%D2+%D45)(P)
000201' 200 10 0 17 000000                      MOVE            A1,(P)
000202' 505 11 0 00 000001                      HRLI            A2,1
000203' 541 11 0 17 000000                      HRRI            A2,(P)
000204' 251 11 0 17 000000                      BLT             A2,%D2(P)
000205' 105 17 0 00 000041                      ADJSP           P,%D41
000206' 202 10 0 17 000000                      MOVEM           A1,(P)
000207' 261 17 0 00 000016                      PUSH            P,%D8
000210' 261 17 0 00 000007                      PUSH            P,%D3
000211' 201 07 0 17 777774                      MOVEI           LP,-2(P)
000212' 200 16 0 07 777777                      MOVE            %D8,-1(LP)
000213' 201 10 0 00 000000                      MOVEI           A1,0
000214' 261 17 0 00 000010                      PUSH            P,A1
000215' 200 16 0 07 000001                      MOVE            %D8,1(LP)
000216' 200 07 0 07 000002                      MOVE            LP,2(LP)
000217' 105 17 0 00 777773                      ADJSP           P,-5
000220' 254 00 1 17 000002                      JRST            @2(P)
000221' 120 122 117 103 105     PRG.0:  ASCIZ   /PROCESS/
000222' 123 123 117 040 000
000223' 200 10 0 17 000000                      MOVE            A1,(P)
000224' 505 11 0 00 000001                      HRLI            A2,1
000225' 541 11 0 17 000000                      HRRI            A2,(P)
000226' 251 11 0 17 000000                      BLT             A2,%D2(P)
000227' 105 17 0 00 000041                      ADJSP           P,%D41
000230' 202 10 0 17 000000                      MOVEM           A1,(P)
000231' 261 17 0 00 000016                      PUSH            P,%D8
000232' 261 17 0 00 000007                      PUSH            P,%D3
000233' 201 07 0 17 777776                      MOVEI           LP,-2(P)
000234' 200 16 0 07 777777                      MOVE            %D8,-1(LP)
000235' 201 10 0 00 000001                      MOVEI           A1,1
000236' 261 17 0 00 000010                      PUSH            P,A1
000237' 200 01 0 16 000001              MOVE    T1,%D1(%D8)
000240' 201 02 0 16 000003              MOVEI   T2,%D3(%D8)
000241' 201 03 0 16 000026              MOVEI   T3,%D22(%D8)
000242' 201 04 0 16 000025              MOVEI   T4,%D21(%D8)
000243' 260 17 0 00 400174'             PUSHJ   P,@ATN.0
000244' 336 00 0 16 000025              SKIPN   %D21(%D8)
000245' 254 00 0 00 000314'             JRST    OBS.03
000246' 200 10 0 16 000025              MOVE    A1,%D21(%D8)
000247' 316 10 0 16 000024              CAIN    A1,%D20(%D8)
000250' 254 00 0 00 000265'             JRST    OBS.0
000251' 202 10 0 16 000024              MOVEM   A1,%D20(%D8)
000252' 201 01 0 00 000471'             MOVEI   T1,TTY.
000253' 505 02 0 00 440700              HLRZ    T2,440700
000254' 541 02 0 00 000221'             HRRI    T2,FRS.0
000255' 201 03 0 00 000011              MOVEI   T3,9
000256' 260 17 0 00 000401'             PUSHJ   P,TT4.2
000257' 201 01 0 00 000401'             MOVEI   T1,TTY.V
000260' 200 02 0 16 000025              MOVE    T2,%D21(%D8)
000261' 260 17 0 00 000035'             PUSHJ   P,TT4.1
000262' 201 01 0 00 000401'             MOVEI   T1,TTY.V
000263' 201 02 0 00 000051              MOVEI   T2,")"
000264' 260 17 0 00 000101'             PUSHJ   P,TT4.2
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-16
FIVEPH  MAC     6-May-83 13:41

000265' 260 17 0 00 000123'             PUSHJ   P,TTW.3
000266' 201 01 0 00 000401'             MOVEI   T1,TTY.V
000267' 505 02 0 00 440700              HRLI    T2,440700
000270' 541 02 0 16 000003              HRRI    T2,^D3(SB)
000271' 201 03 0 00 000100              MOVEI   T3,^D0
000272' 260 17 0 00 000001'             PUSHJ   P,TTW.0
000273' 260 17 0 00 000123'             PUSHJ   P,TTW.3
000274' 200 10 0 16 000003              MOVE    A1,^D3(SB)
000275' 242 10 0 00 777743              LSH     A1,-^D3
000276' 302 10 0 00 000077              CAIE    A1,"?"
000277' 254 00 0 00 000314'             JRST    OBS.03
000300' 051 11 0 00 000000              [UNSPEC]
000301' 201 01 0 00 000401'             MOVEI   T1,TTY.V
000302' 201 02 0 00 000077              MOVEI   T2,"?"
000303' 260 17 0 00 000101'             PUSHJ   P,TTW.2
000304' 201 01 0 00 000421'             MOVEI   T1,TTY.D
000305' 505 02 0 00 440700              HRLI    T2,440700
000306' 541 02 0 16 000003              HRRI    T2,^D3(SB)
000307' 201 03 0 16 000023              MOVEI   T3,^D19(SB)
000310' 260 17 0 00 000147'             PUSHJ   P,TTW.6
000311' 200 01 0 16 000021              MOVE    T1,^D17(SB)
000312' 201 02 0 16 000033              MOVEI   T2,^D27(SB)
000313' 260 17 0 00 400262'             PUSHJ   P,FFY.0
000314' 332 00 0 16 000026      OBS.03: SKIPE   <^D22>(SB)
000315' 254 00 0 00 000337'             JRST    OBS.01
000316' 200 01 0 16 000002              MOVE    T1,^D2(SB)
000317' 260 17 0 00 400420'             PUSHJ   P,SSF.I
000320' 322 01 0 00 000237'             JUMPE   T1,OBS.01
000321' 200 16 0 07 000001                      MOVE    SB,1(LB)
000322' 200 07 0 07 000002                      MOVE    LB,2(LB)
000323' 105 17 0 00 777773                      ADJSP   P,-^D4+^D0+5
000324' 254 00 1 17 000002                      JRST    2(P)
000325' 200 10 0 17 000000                      MOVE    A1,0(P)
000326' 505 11 0 00 000001                      HRLI    B2,1
000327' 541 11 0 17 000000                      HRRI    B2,0(P)
000330' 251 11 0 17 000000                      BLT     A2,^D0(P)
000331' 105 17 0 00 000001                      ADJSP   P,^D0+1
000332' 202 10 0 17 000000                      MOVEM   A1,0(P)
000333' 261 17 0 00 000016                      PUSH    P,GP
000334' 261 17 0 00 000007                      PUSH    P,LB
000335' 201 07 0 17 777774                      MOVEI   LB,-3(P)
000336' 200 16 0 07 777777                      MOVE    GB,-^D0+1(LB)
000337' 201 10 0 00 000000                      MOVEI   A1,^D0
000340' 261 17 0 00 000010                      PUSH    P,A1
000341' 201 01 0 00 000401'                     MOVEI   T1,TTY.V
000342' 260 17 0 00 000201'                     PUSHJ   P,TTY.I
000343' 402 00 0 16 000024      SETZM   <^D20>(SB)
000344' 402 00 0 16 000025      SETZM   <^D21>(SB)
000345' 402 00 0 16 000026      SETZM   <^D22>(SB)
000346' 200 16 0 07 000001                      MOVE    SB,1(LB)
000347' 200 07 0 07 000002                      MOVE    LB,2(LB)
000350' 105 17 0 00 777773                      ADJSP   P,-<^D0+^D0+5>
000351' 254 00 1 17 000002                      JRST    @<^D0+2>(P)
000352' 000325' 001752          XWD     OBS.I,^D1002
000401' 000201' 001751          XWD     TTY.I,^D1001
```

```
FIVEPH  MACRO Z53A(1152) 13:43  6-May-83 Page 1-17
FIVEPH  MAC      6-May-83 13:41

000423' 134 06 0 00 000002          ILDB        C,T2
000424' 306 06 0 00 000000          CAIN        C,0
000425' 263 17 0 00 000000          POPJ        P,
000426' 136 06 0 01 000004          IDPB        C,PTOBP(T1)
000427' 370 00 0 01 000005          SOS         PTOBC(T1)
000430' 254 00 0 00 000423'         JRST        PTASCZ
000431' 201 06 0 00 002415          MOVEI       C,2415
000432' 136 06 0 01 000004          IDPB        C,PTOBP(T1)
000433' 370 00 0 01 000005          SOS         PTOBC(T1)
000434' 242 06 0 00 777771          LSH         C,-7
000435' 302 06 0 00 000000          CAIE        C,0
000436' 254 00 0 00 000432'         JRST        PTCRLF+1
000437' 067 01 0 00 000000          OUTPUT      PTY,
000440' 263 17 0 00 000000          POPJ        P,
000441' 136 02 0 01 000004          IDPB        T2,PTOBP(T1)
000442' 370 00 0 01 000005          SOS         PTOBC(T1)
000443' 263 17 0 00 000000          POPJ        P,
000444' 066 01 0 00 000000          INPUT       PTY,
000445' 201 02 0 00 000001          MOVEI       T2,PTY
000446' 047 02 0 00 000061          JOBSTS      T2,
000447' 254 00 0 00 000445'           JRST      PT.UML+1
000450' 603 02 0 00 040000          TLNE        T2,(JB.UOA)
000451' 254 00 0 00 000444'         JRST        PT.UML
000452' 607 02 0 00 020000          TLNN        T2,(JB.UDI)
000453' 254 00 0 00 000444'         JRST        PT.UML
000454' 263 17 0 00 000000          POPJ        P,
000455' 066 01 0 00 000000          INPUT       PTY,
000456' 375 00 0 01 000010          SOSGE       PTIBC(T1)
000457' 254 00 0 00 000465'         JRST        .+6
000460' 134 06 0 01 000007          ILDB        C,PTIBP(T1)
000461' 306 06 0 00 000000          CAIN        C,0
000462' 254 00 0 00 000465'         JRST        .+3
000463' 051 01 0 00 000006          OUTCHR      C
000464' 254 00 0 00 000456'         JRST        .-6
000465' 201 02 0 00 000001          MOVEI       T2,PTY
000466' 047 02 0 00 000061          JOBSTS      T2,
000467' 254 00 0 00 000465'           JRST      .-2
000470' 603 02 0 00 040000          TLNE        T2,(JB.UOA)
000471' 254 00 0 00 000455'         JRST        .-14
000472' 607 02 0 00 020000          TLNN        T2,(JB.UDI)
000473' 254 00 0 00 000455'         JRST        .-16
000474' 263 17 0 00 000000          POPJ        P,
000475' 200 02 0 00 000506'         MOVE        T2,.+9
000476' 434 02 0 00 000001          IOR         T2,T1
000477' 256 00 0 00 000002          XCT         T2
000500' 254 00 0 00 000504'           JRST      .+4
000501' 260 17 0 00 000431'         PUSHJ       P,PTCRLF
000502' 260 17 0 00 000444'         PUSHJ       P,PT.UML
000503' 263 17 0 00 000000          POPJ        P,
000504' 051 03 0 00 000507'         OUTSTR      .+3
000505' 047 00 0 00 000012          EXIT
000506' 050 01 0 00 000000          OPEN        PTY,0
000507' 077 120 124 131 040         ASCIZ       /?PTY OPEN FAIL
000510' 117 120 105 116 040
000511' 106 101 111 114 015
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-18
FIVEPH  MAC     6-May-83 13:41

        000512' 012 000 000 000 000     /
        000513' 261 17 0 00 000002              PUSH            P,T2
        000514' 201 02 0 00 000001              MOVEI           T2,PTY
        000515' 047 02 0 00 000061              JOBSTS          T2,
        000516' 260 17 0 00 000475'               PUSHJ         P,PTOPEN
        000517' 505 02 0 00 440700              HRLI            T2,440700
        000520' 541 02 0 00 000530'             HRRI            T2,.+8
        000521' 260 17 0 00 000423'             PUSHJ           P,PTASCZ
        000522' 262 17 0 00 000002              POP             P,T2
        000523' 260 17 0 00 000423'             PUSHJ           P,PTASCZ
        000524' 260 17 0 00 000431'             PUSHJ           P,PTCRLF
        000525' 260 17 0 00 000444'             IFE TSTSW,<PUSHJ                        P,PT.UML>
        000526' 405 02 0 00 000777              ANDI            T2,777
        000527' 263 17 0 00 000000              POPJ            P,
        000530' 114 117 107 040 000             ASCIZ           /LOG /
        000531' 261 17 0 00 000002              PUSH            P,T2
        000532' 505 02 0 00 440700              HRLI            T2,440700
        000533' 541 02 0 00 000551'             HRRI            T2,.+15
        000534' 260 17 0 00 000423'             PUSHJ           P,PTASCZ
        000535' 262 17 0 00 000002              POP             P,T2
        000536' 260 17 0 00 000423'             PUSHJ           P,PTASCZ
        000537' 260 17 0 00 000431'             PUSHJ           P,PTCRLF
        000540' 260 17 0 00 000444'             IFE TSTSW,<PUSHJ                        P,PT.UML>
        000541' 201 02 0 00 000001              MOVEI           T2,PTY
        000542' 047 02 0 00 000061              JOBSTS          T2,
        000543' 254 00 0 00 000541'               JRST          .-2
        000544' 607 02 0 00 200000              TLNN            T2,(JB.ULI)
        000545' 263 17 0 00 000000              POPJ            P,
        000546' 201 02 0 00 000001              MOVEI           T2,1
        000547' 047 02 0 00 000031              SLEEP           T2,
        000550' 254 00 0 00 000541'             JRST            .-7
        000551' 122 125 116 040 000             ASCIZ           /RUN /
        000552' 047 02 0 00 000024              GETPPN          T2,
        000553' 200 02 0 00 000002              MOVE            T2,T2
        000554' 201 04 0 00 000006              MOVEI           T4,6
        000555' 400 03 0 00 000000              SETZ            T3,
        000556' 245 02 0 00 000003              ROTC            T2,3
        000557' 435 03 0 00 000060              IORI            T3,"0"
        000560' 136 03 0 00 000001              IDPB            T3,T1
        000561' 367 04 0 00 000555'             SOJG            T4,.-4
        000562' 322 02 0 00 000566'             JUMPE           T2,.+4
        000563' 201 03 0 00 000054              MOVEI           T3,","
        000564' 136 03 0 00 000001              IDPB            T3,T1
        000565' 254 00 0 00 000554'             JRST            .-9
        000566' 136 02 0 00 000001              IDPB            T2,T1
        000567' 263 17 0 00 000000              POPJ            P,
        000570' 047 00 0 00 000000                      RESET
        000571' 200 17 0 00 001123'                     MOVE            P,.PDPTR
        000572' 505 10 0 00 777777                      HRLI            A1,-1
        000573' 541 10 0 00 000006                      HRRI            A1,.GTPRV
        000574' 047 10 0 00 000041                      GETTAB          A1,
        000575' 254 00 0 00 000572'                       JRST          .-3
        000576' 607 10 0 00 000100                      TLNN            A1,000100
                                                          JRST          [OUTSTR [ASCIZ "?FALTA PRIVILEGIO DE E
NQ./DEQ."]
        000577' 254 00 0 00 401631'                                      EXIT]
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-19
FIVEPH  MAC     6-May-83 13:41


                                        MOVEI           A1,[XWD 1,5
                                                            XWD 0,1
                                                            XWD 0,-2
                                                            XWD -1,[ASCIZ 'FIVEPH']
000600' 201 10 0 00 401635'                                 XWD 0,0]
000601' 047 10 0 00 000151              ENQ.            A1,
                                          JRST          [OUTSTR [ASCIZ '?ENQ. ']
                                                         OUTSTR [ASCIZ 'FIVEPH']
000602' 254 00 0 00 401644'                              EXIT]
                                        MOVEI           A1,[SIXBIT /DSK/
                                                            SIXBIT 'FIVEPH'
                                                            SIXBIT /ERE/
                                                            EXP 0
                                                            EXP 0
000603' 201 10 0 00 401647'                                 EXP 0]
000604' 047 10 0 00 000040              GETCEG          A1,
                                          JRST          [OUTSTR [ASCIZ '?GETSEG ']
                                                         OUTSTR [ASCIZ 'FIVEPH']
000605' 254 00 0 00 401657'                              EXIT]
000606' 400 10 0 00 000000              SETZ            A1,
000607' 047 10 0 00 000036              SETUWP          A1,
                                          JRST          [OUTSTR [ASCIZ '?SETUWP ']
                                                         OUTSTR [ASCIZ 'FIVEPH']
000610' 254 00 0 00 401664'                              EXIT]
000611' 047 10 0 00 000030              PJOB            A1,
000612' 514 10 0 00 000010              HRLZ            A1,A1
000613' 436 10 0 00 400017'             IORM            A1,MONREC+3
401611' 201 01 0 00 400575'             MOVEI           T1,CTY.V
401612' 260 17 0 00 400322'             PUSHJ           P,CTY.I
000614' 201 10 0 00 400575'             MOVEI           A1,CTY.V
000615' 202 10 0 00 401077'             MOVEM           A1,CTF.V+1
401613' 201 01 0 00 401076'             MOVEI           T1,CTF.V
401614' 260 17 0 00 400554'             PUSHJ           P,CTF.I
000616' 201 10 0 00 400575'             MOVEI           A1,CTY.V
000617' 202 10 0 00 000353'             MOVEM           A1,OBS.V+1
000620' 201 10 0 00 401076'             MOVEI           A1,CTF.V
000621' 202 10 0 00 000354'             MOVEM           A1,OBS.V+2
000622' 201 01 0 00 000352'             MOVEI           T1,OBS.V
000623' 260 17 0 00 000325'             PUSHJ           P,OBS.I
                                INIMON          M.O,I.B.1^XALL
000624' 201 01 0 00 401576'             MOVEI           T1,M.O
000625' 260 17 0 00 401450'             PUSHJ           P,I.B.1
                                FINLOW          'FIVEPH'^XALL
000626' 200 01 0 00 000706'             MOVE            T1,PPNASC
000627' 260 17 0 00 000552'             PUSHJ           P,GTPPN
000630' 201 01 0 00 000712'             MOVEI           T1,PTYBLK
000631' 205 10 0 00 777773              MOVSI           A1,-NOPRO
000632' 200 02 0 00 000706'     DSP.01: MOVE            T2,PPNASC
000633' 260 17 0 00 000513'             PUSHJ           P,LOGIN
000634' 202 02 0 10 001136'             MOVEM           T2,PROJOB(A1)
000635' 200 11 0 10 001143'             MOVE            A2,PARPRO(A1)
000636' 505 11 0 10 000001              HRLI            A2,1(A1)
000637' 202 11 0 02 401122'             MOVEM           A2,TBJOB(T2)
000640' 550 02 0 00 000010              HRRZ            T2,A1
000641' 242 02 0 00 000001              LSH             T2,1
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-20
FIVEPH  MAC     6-May-83 13:41

000642' 201 02 0 02 001124'             MOVEI           T2,PRONAM(T2)
000643' 505 02 0 00 440700              HRLI            T2,440700
000644' 260 17 0 00 000531'             PUSHJ           P,RUNPRO
000645' 253 10 0 00 000632'             AOBJN           A1,DSP,01
000646' 201 01 0 00 401076'     DSP,02: MOVEI           T1,CTF,V
000647' 260 17 0 00 400454'             PUSHJ           P,SGF,2
000650' 302 01 0 00 000000              CAIE            T1,0
000651' 254 00 0 00 000656'             JRST            DSP,C3
000652' 201 10 0 00 000020              MOVEI           A1,20
000653' 047 10 0 00 000072              HIBER           A1,
000654' 254 00 0 00 000652'               JRST          .-2
000655' 254 00 0 00 000646'             JRST            DSP,02
000656' 047 10 0 00 000024              GETPPN          A1,
000657' 200 10 0 00 000010              MOVE            A1,A1
000660' 202 10 0 00 000701'             MOVEM           A1,LUPHGH+3
000661' 202 10 0 00 000705'             MOVEM           A1,RENHGH+3
                                        OPEN            DSK,[EXP        ,,PPNASC
                                                             SIXBIT     /DSK/
                                                             XWD        0,0]
000662' 050 15 0 00 401667'
000663' 254 00 0 00 000662'               JRST          ,-1
000664' 076 15 0 00 000676'             LOOKUP          DSK,LUPHGH
000665' 254 00 0 00 000664'               JRST          ,-1
000666' 055 15 0 00 000702'             RENAME          DSK,RENHGH
000667' 254 00 0 00 000666'               JRST          ,-1
                                        MOVEI           A1,[XWD 1,5
                                                            XWD 0,1
                                                            XWD 0,-2
                                                            XWD -1,[ASCIZ 'FIVEPH']
000670' 201 10 0 00 401635'                                 XWD 0,0]
000671' 047 10 0 00 000152              DEQ,            A1,
                                          JRST          [OUTSTR [ASCIZ /?DEQ, /]
                                                         OUTSTR [ASCIZ 'FIVEPH']
                                                         EXIT]
000672' 254 00 0 00 401674'
000673' 201 01 0 00 000352'             MOVEI           T1,OBS,V
000674' 260 17 0 00 000223'             PUSHJ           P,OBS,0
000675' 047 00 0 00 000012              EXIT
000676' 46 51 66 45 60 50       LUPHGH: SIXBIT          'FIVEPH'
000677' 45 70 45 00 00 00               SIXBIT          /EXE/
000700' 000000  000000                  XWD             0,0
000701' 000000  000000                  XWD             0,0
000702' 46 51 66 45 60 50       RENHGH: SIXBIT          'FIVEPH'
000703' 45 70 45 00 00 00               SIXBIT          /EXE/
000704' 077000  000000                  EXP             077B8
000705' 000000  000000                  XWD             0,0
000706' 440700  000707'         PPNASC: XWD             440700,.+1
000712' 000000  000000          PTYBLK: EXP             0
000713' 60 64 71 00 00 00               SIXBIT          /PTY/
000714' 000715' 000720'                 XWD             PTYBLK+PTOB,PTYBLK+PTIB
001123' 777600  000722'         .PDPTR: IOWD            200,.PDL
                                        ITBNAM          'FIL01 ',1^XALL
001124' 106 111 114 117 061                     ASCIZ           'FIL01 '
001125' 040 000 000 000 000
                                        ITBNAM          'FIL01 ',2^XALL
001126' 106 111 114 117 061                     ASCIZ           'FIL01 '
001127' 040 000 000 000 000
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page 1-21
FIVEPH  MAC     6-May-83 13:41


                                                ITBNAM          'FILO1 ',3^XALL
001130' 106 111 114 117 061                             ASCIZ           'FILO1 '
001131' 040 000 000 000 000

                                                ITBNAM          'FILO1 ',4^XALL
001132' 106 111 114 117 061                             ASCIZ           'FILO1 '
001133' 040 000 000 000 000

                                                ITBNAM          'FILO1 ',5^XALL
001134' 106 111 114 117 061                             ASCIZ           'FILO1 '
001135' 040 000 000 000 000

                                                ITBJP           ^D5^XALL
                                                ITBPAR          <PROPAR+^D0>^XALL
001143' 000000  401615'                                 EXP             PROPAR+^D0

                                                IPPAR           M.O^XALL
401615' 000000  401576'                                 EXP             M.O

                                                ITBPAR          <PROPAR+^D1>^XALL
001144' 000000  401616'                                 EXP             PROPAR+^D1

                                                IPPAR           M.O^XALL
401616' 000000  401576'                                 EXP             M.O

                                                ITBPAR          <PROPAR+^D2>^XALL
001145' 000000  401617'                                 EXP             PROPAR+^D2

                                                IPPAR           M.O^XALL
401617' 000000  401576'                                 EXP             M.O

                                                ITBPAR          <PROPAR+^D3>^XALL
001146' 000000  401620'                                 EXP             PROPAR+^D3

                                                IPPAR           M.O^XALL
401620' 000000  401576'                                 EXP             M.O

                                                ITBPAR          <PROPAR+^D4>^XALL
001147' 000000  401621'                                 EXP             PROPAR+^D4

                                                IPPAR           M.O^XALL
401621' 000000  401576'                                 EXP             M.O
                000E70'         END START

NO ERRORS DETECTED

HI-SEG. BREAK IS 401677
PROGRAM BREAK IS 001150
CPU TIME USED 00:03.821

34P CORE USED
```

```
FIVEPH  MACRO %53A(1152) 13:43  6-May-83 Page S-1
FIVEPH  MAC     6-May-83 13:41          SYMBOL TABLE

A1              000010  int     MOVREC          400014'         T2              000002  int
A2              000011  int     NCERC           000005          T3              000003  int
A3              000012  int     OBS.0           000223'         T4              000004  int
A4              000013  int     OBS.01          000237'         TBJOB           401122'
ATN.0           400176'         OBS.02          000266'         TRSOUT          400000'
ATN.01          400217'         OBS.03          000314'         TSTSW           000000  int
ATN.02          400230'         OBS.I           000325'         TTR.0           000147'
ATN.03          400252'         OBS.V           000352'         TTR.01          000164'
C               000006  int     OPEN    050000  000000          TTR.02          000172'
CLRFI   051440  000000          OUTCHR  051040  000000          TTW.0           000001'
COUTST          000000'         OUTPUT  067000  000000          TTW.01          000020'
CT.1            401263'         OUTSTR  051140  000000          TTW.1           000035'
CT.2            401274'         P               000017  int     TTW.11          000061'
CT.3            401302'         PARPRO          001143'         TTW.12          000065'
CT.4            401475'         PJOB    047000  000030          TTW.2           000101'
CT.5            401504'         PPNASC          000706'         TTW.3           000123'
CTF.I           400554'         PROJOB          001136'         TTY.I           000201'
CTF.V           401075'         PROGRM          001124'         TTY.V           000241'
CTY.I           400300'         PROPAR          401615'         WAIT            000061'
CTY.V           400575'         PRS.0           000221'         WAKE    047000  000073
DEQ.    047000  000152          PT.TTY          000455' int     XITM2           400046'
DSK             000015          PT.UML          000444' int     XITMCN          400032'
DSP.01          000632'         PTASCZ          000423' int     .DE30R          000000  SPD
DSP.02          000646'         PTCHR           000441' int     .EN30L          000000  SPD
DSP.03          000656'         PTCRLF          000431' int     .GTPRV          000006  SPD
ENQ.    047000  000151          PTIB            000006  int     .IOIMF          000017  SPD
ENTMON          400021'         PTIBC           000010  int     .RTL            000723'
EXIT    047000  000012          PTIBP           000007  int     .PDFTR          001123'
FIN.0           400351'         PTOB            000003  int
FIN.1           400354'         PTOBC           000005  int
GB              000016  int     PTOBP           000004  int
GETPPN  047000  000024          PTOPEN          000475' int
GETSEG  047000  000040          PTY             000001  int
GETTAB  047000  000041          PTYBLK          000712'
GTPPN           000552' int     R.1             401306'
HB.RWP  000002  000000  SPD     R.2             401371'
HB.RWT  000001  000000  SPD     R.3             401512'
HIBER   047000  000072          RENAME  055000  000000
I.B.1           401450'         RENHGH          000752'
I.B.2           401554'         REQ.0           400114'
INCHRW  051000  000000          REQ.01          400144'
INPUT   066000  000000          REQ.02          400170'
JB.UDI  020000  000000  SPD     RESET   047000  000000
JB.ULI  200000  000000  SPD     RPV.0           400262'
JB.UOA  040000  000000  SPD     RUNPRO          000531' int
JOBSTS  047000  000061          SETUWP  047000  000035
L.1             401324'         SGF.0           400365'
L.2             401344'         SGF.00          400356'
L.3             401532'         SGF.1           400420'
LB              000007  int     SGF.2           400454'
LOGIN           000513' int     SGF.3           400516'
LOOKUP  076000  000000          SIGNAL          400073'
LUFHGH          000676'         SLEEP   047000  000031
M.0             401576'         START           000570'
MAXJBS          000062  int     T1              000001  int
```

```
FIL01   MACRO %53A(1152) 13:45  12-Feb-83 Page 1
FIL01   MAC      12-Feb-83 13:44


                                        TITLE   FIL01
                                        SEARCH  UUOSYM,CONSTS,MACROS,FIVEPH
        400000'                         TWOSEG

                                                INIPRO          'FIVEPH',P,1,1^XALL

000000' 047 00 0 00 000000      PROSTA: RESET
000001' 205 10 0 00 777777              MOVSI           A1,-1
000002' 047 10 0 00 000104              ATTACH          A1,
000003' 254 00 0 00 000011'                     JRST            ,.+2
                                                MOVEI           A1,[SIXBIT /DSK/
                                                                    SIXBIT 'FIVEPH'
                                                                    SIXBIT /EXE/
                                                                    EXP 0
                                                                    EXP 0
                                                                    EXP 0]
000004' 201 10 0 00 002015'
000005' 047 10 0 00 000040              GETSEG          A1,
000006' 254 00 0 00 000035'                     JRST            PROEND
000007' 400 10 0 00 000000              SETZ            A1,
000010' 047 10 7 00 000010              [illegible]     A1,
000011' 254 00 0 00 000035'                     JRST            PROEND
000012' 047 10 0 00 000030              PJOB            A1,
000013' 200 10 0 10 401122'             MOVE            A1,TBJOB(A1)
000014' 202 10 0 00 000037'             MOVEM           A1,P.1
000015' 505 10 0 10 000000              HRLI            A1,(A1)
000016' 541 10 0 00 000040'             HRRI            A1,P.1+1
000017' 251 10 0 00 000040'             BLT             A1,P.1+1
000020' 200 17 0 00 002015'             MOVE            P,PDLPTR
000021' 201 16 0 00 000037'             MOVEI           GB,P.1
000022' 201 07 0 00 000000              MOVEI           LB,0
000023' 201 01 0 00 401076'             MOVEI           T1,CTF.V
000024' 260 17 0 00 400516'             PUSHJ           P,SGF.3
                                        ICPAR           <[illegible]>,<C.01+1>,0^XALL

000025' 200 10 0 16 000001                      MOVE            A1,[illegible]
000026' 202 10 0 00 000042'                     MOVEM           A1,C.01+1

                                        INICLA          C.01,I.B,2^XALL

000027' 201 01 0 00 000041'                     MOVEI           T1,C.01
000030' 260 17 0 00 401554'                     PUSHJ           P,I.B.2
000031' 201 01 0 00 000041'             MOVEI           T1,C.01
                                        CALL            R,3^
000032' 260 17 0 00 401512'             PUSHJ           P,R.3

                                        FINPRO^XALL

000033' 201 01 0 00 401076'             MOVEI           T1,CTF.V
000034' 260 17 0 00 400365'             PUSHJ           P,SGF.0
                                PROEND: MOVEI           A1,[SIXBIT /SYS/
                                                            SIXBIT /LOGOUT/
                                                            SIXBIT /EXE/
                                                            EXP 0
                                                            XWD 1,4
                                                            EXP 0]

000035' 201 10 0 00 002024'             RUN             A1,
000036' 047 10 0 00 000035
000037'                         P.1:
000037' 000000  000000                  EXP             0
000040'                                 BLOCK           ^D1
000041'                         C.01:
000041' 000000  000001                  XWD             0,^D1
```

```
FIL01   MACRO %5.A(1150) 13:45  12-Feb-83 Page 1-1
FIL01   MAC     12-Feb-83 13:44

       000042'                              BLOCK           ^D1
       000043'                              BLOCK           1
       000044'                              BLOCK           1
       000045'                      PDL:    BLOCK           ^D1000
       002015' 776030  000044'      PDLPTR: IOWD            ^D1000,PDL
                       000000'      END PROSTA

NO ERRORS DETECTED

HI-SEG, BREAK IS 400000
PROGRAM BREAK IS 002032
CPU TIME USED 00:00.410

40P CORE USED
```

```
FILO1   MACRO %53A(1152) 13:45  12-Feb 83 Page S-1
FILO1   MAC      12-Feb-83 13:44              SYMBOL TABLE

A1              000010  int
ATTACH  047000  000104
C.01            000041'
CTF.V           401076'
GB              000016  int
GETSEG  047000  000040
I.9.2           401554'
LB              000007  int
P               000017  int
P.1             000037'
PDL             000045'
PDLPTR          002015'
PJOB    047000  000030
PRDEND          000035'
PROSTA          000000'
R.7             401512'
RESET   047000  000000
RUN     047000  000035
SETUWP  047000  000036
SGF.0           400365'
SGF.3           400516'
T1              000001  int
TSJOB           401122'
```

BIBLIOGRAFIA

[DIG 78a] DIGITAL EQUIPMENT CORPORATION. DEC system-10/DEC system-20 hardware reference manual, Feb.1978.

[DIG 78b] DIGITAL EQUIPMENT CORPORATION. DEC system-10 macro assembler reference manual. Apr. 1978.

[DIG 78c] DIGITAL EQUIPMENT CORPORATION. DEC system-10 link reference manual. Jun. 1978.

[DIG 80] DIGITAL EQUIPMENT CORPORATION. TOPS-10 monitor calls. Aug. 1980.

[HAN 77] HANSEN, Per Brinch. The architecture of concurrent programs. Englewood Cliffs, Prentice-Hall, 1977.

[HOL 78] HOLT, R.C. Structured concurrent programming with operating systems applications. Toronto, Addison-Wesley, 1978.

[MED 81] MEDEIROS, Gil Carlos Rodrigues. Implementação do sistema Pascal Concorrente no computador LABO-8034. Porto Alegre, PGCC da UFRGS, 1981.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Pós-Graduação em Ciência da Computação

Um Método para Implementação de Programas Concorrentes no Computador DEC-10

Dissertação apresentada aos Srs.

Visto e permitida a impressão.
Porto Alegre, ..01/.10./.84...

Flávio Rech Wagner
Coordenador do Curso de Pós-Graduação em Ciência da Computação