UNIVERSIDADE ESTADUAL DE CAMPINAS
- UNICAMP -

Este exemplar corresponde à redação final da tese defendida por ALFREDO FRANCISCO SILVA JR.
Julgada em 29/07/94
Orientador

Programa de Pós Graduação em Engenharia Elétrica

DEPURADOR DE SISTEMAS DE TEMPO REAL BASEADOS NA NORMA MOSI-DEBUGMOSI

Autor: Alfredo Francisco da Silva Jr.
Orientador: Prof. Dr. Maurício Ferreira Magalhães

Dissertação submetida à Universidade Estadual de Campinas para obtenção do Grau de Mestre em Engenharia Elétrica

Campinas, 29 de julho de 1994

UNICAMP
BIBLIOTECA CENTRAL

Agradeço aos meus amigos Hélio Azevedo, Josué Jr. Guimarães Ramos, Alexandre V. R. de Lara, João Daros, ao meu orientador Maurício F. Magalhães e aos meus sogros João B. Fidelis e Lisimari S. Fidelis, sem os quais não seria possível a realização deste trabalho.

Dedico este trabalho ao meu irmão Eduardo e à minha esposa Rosana.

RESUMO

Neste trabalho é proposto um depurador de sistemas de tempo real, denominado DEBUGMOSI.

O ambiente de depuração prevê duas formas de operação:

- Utilização de duas máquinas distintas, uma "rodando" o depurador e outra a aplicação, interligadas por linha serial para tratar os níveis iniciais de desenvolvimento ou testes, onde o objetivo é verificar a existência de erros graves de projeto e validar algoritmos. Normalmente nestas fases o fator tempo pode ser relegado a um segundo plano.

- Uso de uma única máquina contendo o depurador. A entrada deste último é um arquivo de eventos gerado pelo próprio núcleo de tempo real quando da execução da aplicação em uma sessão anterior, sem a participação do depurador. Dessa forma, é possível conseguir-se um "replay" da execução do aplicativo sob teste. A vantagem do uso desta técnica é a possibilidade de se repetir a execução do arquivo de eventos diversas vezes, garantindo sempre o mesmo comportamento, assim como da não influência do depurador nos resultados obtidos possibilitando seu uso em casos em que as restrições de tempo sejam severas.

ÍNDICE

Pag.

1.0 Introdução..... 01

2.0 Depuração de sistemas de tempo real (T.R.)..... 03

2.1 Sistemas de tempo real..... 03

2.2 Núcleos de tempo real..... 04

2.3 NTR Quasimosi..... 08

2.4 Dificuldades associadas à depuração de sistemas de T.R..... 11

2.5 Tipos de depuradores..... 11

2.6 Técnicas básicas para depuração..... 13

2.7 Técnicas de apresentação dos dados monitorados..... 21

3.0 Implementação de um depurador para tempo real - DEBUGMOSI..... 27

3.1 Histórico..... 27

3.2 Características do depurador DEBUGMOSI..... 27

3.3 Estrutura de dados..... 34

3.4 Ambiente operacional..... 41

3.5 Módulos constituintes para ambas as abordagens..... 42

3.6 Modo de operação do depurador..... 54

4.0 Exemplos de utilização do depurador DEBUGMOSI..... 57

4.1 Modalidades do depurador..... 57

4.2 Estudo de caso..... 60

5.0 Conclusão e perspectivas..... 80

6.0 Bibliografia..... 82

Anexos:

A - Mensagens trocadas entre núcleo e depurador

B - Manual de utilização do DEBUGMOSI

1.0 INTRODUÇÃO.

A depuração de sistemas de tempo real, principalmente aqueles com severas restrições de tempo (HRT), tem sido recentemente motivo de intensa pesquisa, uma vez que o leque de aplicações dessa tecnologia tem crescido significativamente, com aplicações que vão desde equipamentos domésticos até usinas nucleares e armamentos, além da perspectiva de utilização, no futuro, em estações espaciais e robôs de alto desempenho.

O método normalmente utilizado na depuração de programas sequenciais, denominado depuração cíclica, tem sua eficácia associada ao comportamento determinístico deste tipo de programas. Este método não se aplica a Sistemas de Tempo Real (STR) dada a grande dependência temporal dos mesmos, sendo normalmente difícil a repetição de um comportamento do sistema, mesmo quando entradas idênticas são repetidamente apresentadas.

Outro aspecto a ser salientado, é a interferência normalmente causada pelo depurador ao sistema sob observação, podendo mascarar o comportamento deste último. Esse problema é denominado "efeito ponta de prova" [GAI85].

Levando em consideração as dificuldades citadas e a carência de depuradores de tempo real que permitam a apresentação do comportamento das filas de processos, semáforos, etc, ou seja das entidades do sistema do ponto de vista macroscópico, surgiu a proposta do depurador ora apresentado.

O depurador DEBUGMOSI fornece uma interface H/M baseada em janelas sendo constituido de um menu para seleção das opções disponíveis e de uma área para o monitoramento do aplicativo.

Para possibilitar o acompanhamento e testes da aplicação, considerando-se ou não a influência do fator tempo, o depurador apresenta duas propostas:

- máquinas separadas para a execução da aplicação e do depurador interligadas através de linha serial para as fases da depuração onde se deseja acompanhar o comportamento da aplicação em um nível mais algoritmico.
- uma só máquina que se utiliza de um arquivo de eventos provenientes da execução da aplicação e gerado pelo próprio núcleo de tempo real. Este arquivo é utilizado posteriormente em uma sessão de depuração denominada "replay" que possibilitará ao usuário o acompanhamento da

execução de seu sistema sem a influência do depurador, o que torna viável a depuração de aplicações com restrições devidas ao fator tempo.

A maneira como será apresentada a solução do problema de depuração de tempo real através do DEBUGMOSI está estruturada da seguinte forma:

Cap 2.0 - Enfatiza as abordagens e metodologias utilizadas na depuração de Sistemas de Tempo Real, ressaltando as dificuldades associadas à depuração deste tipo de sistema, os tipos de depuradores e as técnicas básicas usadas na construção de depuradores, finalizando por apresentar as características principais do depurador **DEBUGMOSI**.

Cap 3.0 - Detalha a implementação do depurador **DEBUGMOSI**, em diferentes níveis de abstração. Partindo do diagrama de blocos do sistema até alcançar detalhes específicos relativos à estrutura de dados e a utilização de um autômato finito como elemento gerenciador da ferramenta.

Cap 4.0 - Mostra a utilização do depurador em exemplos que envolvem problemas clássicos como inversão de prioridade, e a maneira pela qual a ferramenta fornece suporte para a descoberta e eliminação de erros na aplicação.

Cap 5.0 - Apresenta a conclusão do trabalho e as perspectivas futuras.

2.0 DEPURAÇÃO DE SISTEMAS DE TEMPO REAL (T.R.).

2.1 Sistemas de tempo real.

Sistemas de Tempo Real (STR), estão normalmente associados com controle e/ou supervisão de processos. Nessa área a correção de um sistema, além de estar vinculada à sua estrutura lógica, também necessita que o mesmo cumpra restrições de tempo. Com relação às restrições de tempo, estas estabelecem uma classificação para os STR, sendo chamado de STR "hard" aqueles que exibem um regime de trabalho sob severas restrições de tempo, enquanto os sistemas cujas tarefas devem ser executadas o mais rapidamente possível, porém, sem tempos críticos é denominado STR "soft".

A figura 2.1 apresenta um diagrama típico do ambiente de atuação de um STR, no controle e supervisão de um processo genérico.

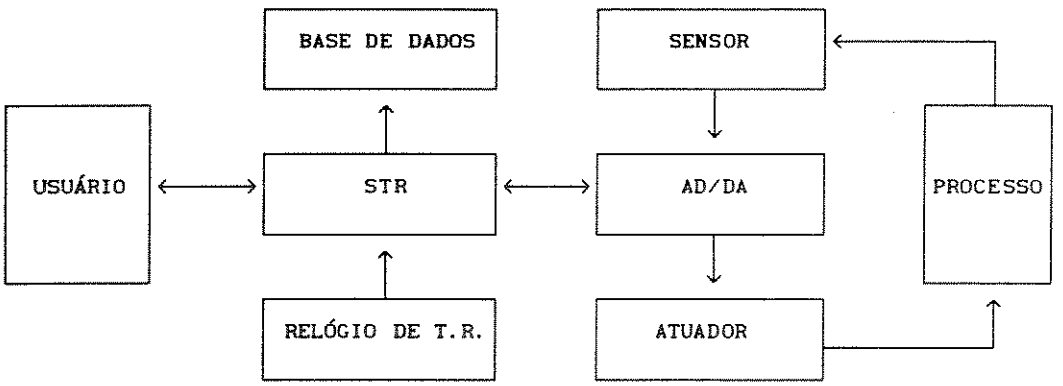


FIG. 2.1 - AMBIENTE DE ATUAÇÃO DE UM STR.

O STR da figura além de interagir com o usuário, através da leitura de teclado, mouse, etc, apresenta o comportamento do processo monitorado via telas, mostradores ou relatórios impressos. As informações adquiridas são armazenadas em uma base de dados, para possibilitar análises posteriores. A aquisição de dados ocorre por intermédio de sensores que acompanham as variações apresentadas pelo processo no decorrer do tempo. Essa informação é manipulada por um conversor analógico digital (AD), para que possa estar disponível para o STR. Por outro lado, se for necessária uma intervenção no

processo, esta é realizada pelos atuadores, sendo o dado previamente tratado por um conversor digital analógico (DA), para adequá-lo ao ambiente do processo.

Todo esse cenário de supervisão e controle, possui uma cadência dada pelo chamado relógio de tempo real, que é um elemento fundamental neste tipo de sistema, seja para registrar o tempo absoluto, como para determinar intervalos, estabelecidos via programa.

Alguns exemplos de STR, podem ser os seguintes:

- Usinas nucleares.
- Sistemas de controle de voô.
- Naves espaciais.
- Robôs.
- Veículos automáticos (AGV).

2.2 Núcleos de tempo real (NTR).

Uma vez que a existência de restrições de tempo é uma das características fundamentais de um STR, é necessária a presença de uma camada de "software" responsável pelo atendimento das mesmas, para que o sistema controlado não entre em colapso. Esta camada é denominada executivo ou núcleo de tempo real (NTR), sendo fundamentalmente um elemento capaz de tratar diversos eventos de modo concorrente. A necessidade da concorrência vem do fato de que na prática, os sistemas de supervisão/controle de processos, são implementados associando a cada evento uma ou mais tarefas, ficando o NTR com a gerência dessas tarefas que deve ser realizada de maneira que aqueles eventos que forem prioritários possam ser tratados com maior urgência.

Segundo [MAG86] um NTR para o desempenho de suas atividades deve possuir as seguintes funções básicas:

- Gerenciar o uso do processador (ou processadores no caso de multiprocessamento) de maneira a permitir o seu compartilhamento pelas tarefas ativas no sistema.
- Suportar que tarefas concorrentes executem sob um esquema de prioridades.

- Prover suporte para comunicação e sincronização entre tarefas de modo a permitir que estas cooperem entre si.

Além dessas funções básicas, alguns requisitos relevantes devem ser observados como:

- O sistema em execução deve ser informado da ocorrência de eventos externos relevantes.
- A memória deve conter mais de uma tarefa simultaneamente.
- Deve ser permitida a interrupção de uma tarefa para a execução de outra bem como o retorno da primeira no ponto em que perdeu a CPU.

2.2.1 Estados das tarefas em um NTR.

Na dinâmica de funcionamento do sistema as tarefas presentes disputam a posse da CPU, gerenciadas pelo NTR. Esse fato gera a existência de diferentes estados possíveis para o conjunto de tarefas, sendo três os estados básicos:

- Execução, referente à tarefa que está de posse da CPU.
- Pronto, quando a tarefa embora apta a executar, não o faz devido à não disponibilidade da CPU.
- Bloqueio, caracterizado pela espera da ocorrência de um evento por parte da tarefa.

A figura 2.2 mostra os estados citados e também as transições entre os mesmos.

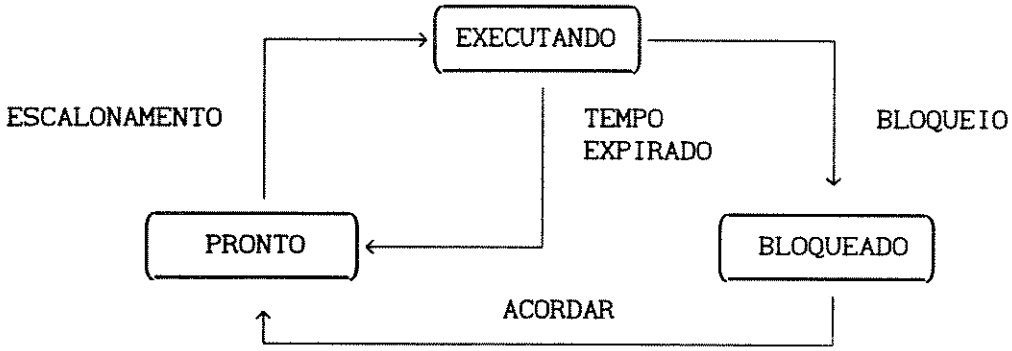


FIG. 2.2 - DIAGRAMA DE ESTADOS DOS PROCESSOS.

2.2.2 Elementos básicos de um NTR.

Dentre os diversos elementos que constituem um NTR aqueles indispensáveis ao seu funcionamento são:

- Escalonador.
- BCT.
- Filas de estado.

a) Escalonador.

A gerência de alocação da CPU para uma dada tarefa, é realizada por um elemento denominado escalonador. Ele pode ser enquadrado em duas categorias: preemptivo e não preemptivo, sendo no primeiro caso permitida a retirada de uma tarefa que estava executando quando outra de maior prioridade estiver requisitando a CPU (ou seja estiver no estado de pronto).

A política de escalonamento escolhida terá um impacto decisivo no tratamento dispensado às tarefas. Existindo escalonadores estáticos ou dinâmicos, caso trabalhe com prioridades fixas ou com valores passíveis de correção conforme a necessidade de momento.

b) BCT.

Para possibilitar a remoção de uma tarefa da CPU e seu posterior retorno, sem risco de perda de informações o que poderia causar graves consequências ao sistema, existe uma estrutura de dados denominada BCT (Bloco de Controle de Tarefa) a qual armazena as informações relevantes do estado do sistema no momento de remoção da mesma, sendo esses dados usados quando do retorno dessa tarefa à CPU. Diversas são as informações salvas no BCT, sendo as principais:

- Identificação da tarefa perante o NTR.
- Prioridade da tarefa.
- Apontador para a área de memória da tarefa.
- Apontadores para recursos alocados.
- Área de salvaguarda de registradores da CPU.

- Campo contendo valor da temporização.

c) Filas de estado das tarefas.

Para um controle efetivo do sistema o NTR necessita conhecer a qualquer instante o estado das tarefas gerenciadas. Com o objetivo de prover essa necessidade são utilizadas filas, sendo uma para cada estado possível das tarefas. No caso de monoprocessamento a fila de execução será unitária, a fila de pronto organizada por ordem de prioridade enquanto as demais não seguem uma ordem explícita, ficando no aguardo da liberação dos recursos de que necessitam.

2.3 NTR Quasimosi.

A apresentação com um maior grau de profundidade desse núcleo em particular, está associada a dois objetivos básicos:

- Mostrar um exemplo de uma implementação onde foi empregada a teoria vista anteriormente.
- Ser o próprio depurador Debugmosi um aplicativo cliente deste núcleo.

A norma MOSI [IEE87] da IEEE, tem por objetivo principal tornar portável aplicativos, entre diferentes ambientes, no que se refere ao acesso dos serviços fornecidos pelo sistema operacional. O núcleo QUASIMOSI implementa um subconjunto dessa norma.

2.3.1 Funções do Quasimosi.

As funções implementadas no núcleo [AZE90], podem ser observadas na figura 2.3.

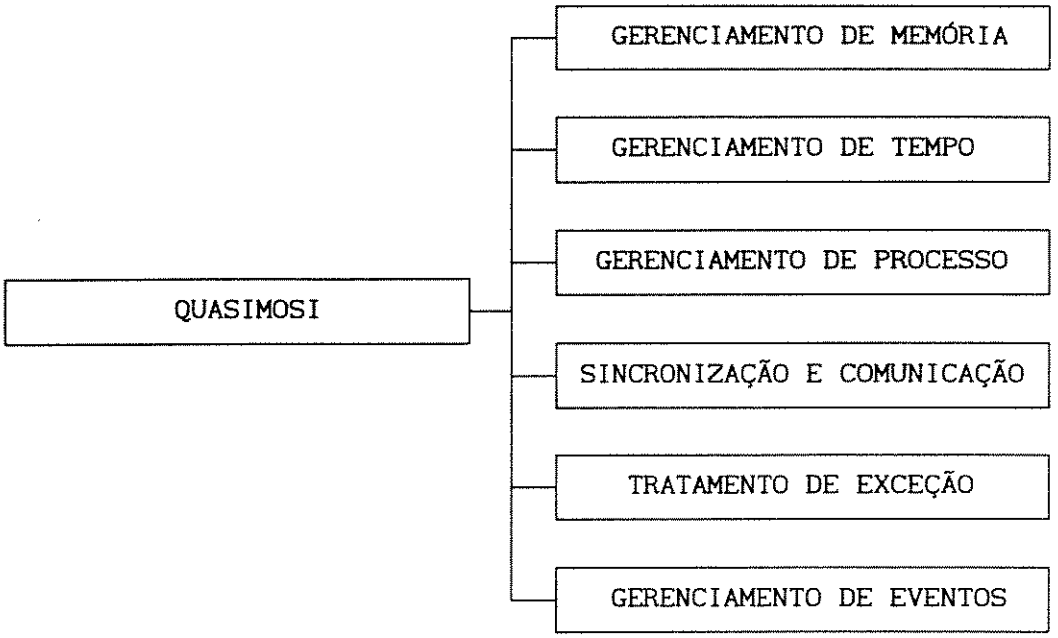


FIG. 2.3 - ESTRUTURA DO NTR QUASIMOSI.

- Gerenciamento de Memória, função responsável pela aquisição e liberação de blocos de memória.
- Gerenciamento de tempo, possibilita o uso de temporizadores aos programas de aplicação.
- Gerenciamento de processo, funções de suporte à criação, destruição e controle dos processos executados de modo concorrente.
- Sincronização e comunicação, permite às tarefas estabelecerem sincronização através de semáforos e troca de mensagens via "mailboxes".
- Tratamento de exceção, trata as exceções (eventos que necessitam de tratamento diferenciado e normalmente urgente) definidas pelo usuário.
- Gerenciamento de eventos, permite à uma tarefa aguardar a ocorrência de um conjunto de eventos.

2.3.2 Estados das tarefas do Quasimosi.

Os estados das tarefas permitidos pelo núcleo podem ser observados na figura 2.4, bem como os eventos que causam a transição.

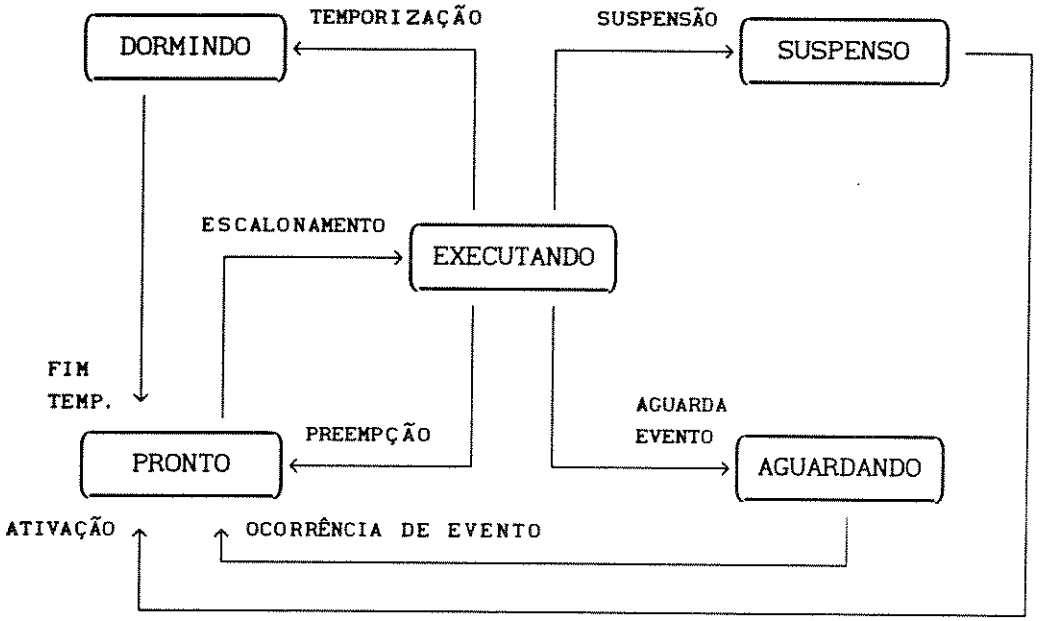


FIG. 2.4 - DIAGRAMA DE ESTADO DAS TAREFAS DO QUASIMOSI.

2.4 Dificuldades associadas à depuração de sistemas de T.R.

O método de depuração cíclica, que consiste em interromper-se repetidamente a execução de um programa para um exame de seu estado, é o mais comum na deteção e correção de erros em programas sequenciais. A eficácia deste método está associada ao comportamento determinístico destes programas. Para sistemas de T.R., no entanto, os métodos convencionais de teste e depuração esbarram nas severas restrições de tempo e no comportamento não determinístico desses sistemas [TSA90], características que praticamente impedem a reprodução de erros.

Além do citado anteriormente, os depuradores de T.R. ainda devem apresentar uma minimização do "efeito ponta de prova" [GAI85] que consiste na interferência causada ao sistema sob monitoramento pelo depurador, podendo provocar alterarações no seu comportamento. Uma forma de minimizar este efeito consiste na adoção de métodos de aquisição e armazenamento dos eventos que permitam a posterior repetição da execução do sistema sob depuração.

Embora alguns pesquisadores façam distinção entre monitores e depuradores [JOY87], entendendo o monitoramento como sendo a atividade de se extrair dados durante a execução de um programa e depuração (segundo o glossário de engenharia de software da ANSI/IEEE) ser o processo de localizar, analisar e corrigir erros, no contexto deste trabalho não será adotada uma diferenciação entre os termos, uma vez que o monitoramento é encarado como um procedimento para a localização de um comportamento incorreto sendo, portanto, uma ferramenta de depuração.

2.5 Tipos de depuradores.

Segundo Haban e Wybranietz [HAB90a], os monitores podem ser classificados em: puramente "hardware", puramente "software" e híbridos. Um monitor puramente "hardware" é um elemento que não está incluido no sistema monitorado. Embora alguns dispositivos sejam projetados para causar um efeito mínimo ou nenhuma interferência no sistema hospedeiro, eles fornecem, geralmente, informações limitadas e de baixo nível. Neste tipo de depurador a

aquisição de dados é realizada pela observação dos barramentos ou pelo uso de pontas conectadas ao processador, memórias ou canais de E/S. Para conseguir informações mais abrangentes é necessário o uso de "hardware" sofisticado, que ainda assim não consegue manipular a criação e destruição dinâmica de processos, a migração de partes de programas na memória e o uso de unidades de gerenciamento de memória. O uso desses monitores é restrito a peritos, sendo exigido, para sua instalação, um profundo conhecimento do sistema.

Plattner [PLA84] optou por um monitor baseado em "hardware". Ele descreve um sistema experimental para monitoramento de T.R., o qual pode ser usado para observar o comportamento de processos, para coletar dados e na identificação de estados ilegais dos processos.

Monitores de "software" apresentam a informação de uma maneira orientada à aplicação. Estes monitores estão normalmente contidos no sistema monitorado, compartilhando o mesmo ambiente de execução causando, portanto, alguma interferência nos tempos de execução e no espaço do sistema não sendo, desta forma, adequados para aplicações de T.R..

Um típico monitor de "software" é o apresentado por Joyce [JOY87]. Seu monitor coleta, interpreta e mostra informações das interações entre processos concorrentes. Para facilitar a deteção dos eventos entre processos, é realizada uma alteração nos processos a serem monitorados carregando-os com um protocolo de comunicações que incorpora facilidades para monitoração. Este procedimento, no entanto, causa uma certa degradação no desempenho do sistema.

Monitores híbridos podem ser projetados para aproveitar as vantagens dos monitores citados anteriormente, com interferência mínima no sistema monitorado. Estes monitores são tipicamente constituidos de um "hardware" independente, o qual recebe dados coletados por uma ferramenta de "software" que está presente no sistema monitorado.

A opção por um monitor híbrido foi feita por Haban [HAB90]. Seu sistema denominado DTM tem seu princípio de monitoramento calcado no fato de que o "software" do monitor presente no sistema monitorado assinala os eventos significativos sendo estes processados e mostrados por um "hardware" dedicado.

2.6 Técnicas básicas para depuração.

Podem ser identificadas basicamente cinco técnicas de depuração:

- Tradicionais.
- Baseadas em eventos.
- Análise estática.
- Baseadas em comportamento.
- Baseadas em Inteligência Artificial (I.A.).

Cada uma dessas técnicas apresenta uma abordagem diferente para o problema de depuração.

Para McDowell e Helmbold [MCD89] a abordagem tradicional apresenta como vantagem principal a facilidade de implementação, porém é uma solução parcial sofrendo de maneira acentuada o "efeito ponta de prova". As técnicas baseadas em eventos permitem um maior nível de abstração além de usarem o recurso da repetição ("replay") para conseguir um comportamento determinístico de um programa que originalmente não apresenta este comportamento tentando, desta forma, minimizar as interferências causadas pelo depurador no sistema monitorado. No caso da análise estática, sua atuação busca encontrar falhas estruturais ao invés de falhas funcionais, uma vez que nesta abordagem o programa não é efetivamente executado. O principal problema associado à análise estática vem do fato de que muitas vezes esta abordagem acaba utilizando algoritmos de complexidade exponencial.

As técnicas restantes, segundo Cheung, Black e Manning [CHE90], realizam uma comparação entre o comportamento esperado e o monitorado, no caso da análise de comportamentos, e na utilização de conceitos e regras de I.A. para a última técnica.

2.6.1 Tradicionais.

Os depuradores cíclicos tradicionais são baseados na inserção de "breakpoints" para permitir a análise do estado do programa. As características principais de parada desses depuradores podem ser agrupadas da seguinte maneira:

- Em um comando do código fonte.
- Na ocorrência de uma interrupção, ou evento detetável pelo usuário.
- Quando uma variável específica é acessada.
- Quando uma expressão condicional é satisfeita.

Para permitir que esse tipo de depurador possa atuar em um ambiente de T.R., seus "breakpoints" passam a ser associados a estados específicos dos processos como: execução, pronto, suspenso, etc, além de possibilitar a parada quando da sincronização (via semáforos), ou comunicação (via "mailbox"), entre os processos.

Com o intuito de possibilitar as características citadas anteriormente, alguns depuradores contam com um suporte fornecido diretamente pelo sistema operacional o que simplifica a implementação de novas funções. Outros modificam o programa fonte, através da inserção de códigos de controle, com o objetivo de evitar alterações no sistema operacional. Neste caso, ocorre uma acentuada perda de desempenho do sistema monitorado, além de apresentar severas restrições de capacidade.

2.6.2 Baseadas em eventos.

Embora utilizado por diversos depuradores, o conceito de evento varia de uma implementação para outra. No sistema DISDEB [LAZ86], um evento é encarado como sendo um acesso à memória; no caso do RADAR [LEB85] cada mensagem enviada ou recebida é um evento e, para o INSTANT REPLAY, [LEB87] o que é visto como evento é o acesso a um objeto.

Com relação ao tratamento dado aos eventos, este varia desde a simples apresentação até o armazenamento dos mesmos. O armazenamento é mais adequado, pois possibilita algumas alternativas importantes como: geração de relatórios, verificação automática dos erros realizada por uma ferramenta adequada e a reprodução da execução do programa. O compromisso associado à opção de armazenamento refere-se ao fato de que, durante a fase de aquisição realizada em tempo de execução, a interferência do depurador no sistema sob monitoramento é acentuada, o que implica que o tempo de aquisição deve ser minimizado limitando, com isso, a coleta dos dados.

Ainda tratando do armazenamento de eventos, alguns aspectos relevantes devem ser ressaltados:

- O nível de detalhamento.
- A forma de aquisição.

a) O nível de detalhamento.

O nível de detalhamento está intimamente relacionado com a quantidade de informação coletada. Com este enfoque, dois níveis com crescente quantidade de armazenamento e complexidade no tratamento da informação podem ser vislumbrados:

- "Browsing".
- Repetição.

a1) "Browsing".

No "browsing" a massa de eventos armazenados é examinada por ferramentas que vão desde simples editores de texto, até programas específicos que realizam animações utilizando os eventos. Apesar de necessitar de uma quantidade mínima de informações sobre os eventos, o principal problema associado ao "browsing" está relacionado com a dificuldade em localizar um evento de interesse em meio à grande quantidade adquirida.

Snodgrass [SNO84] apresenta um mecanismo de captura dos eventos utilizando uma base de dados relacional. Esta base, além de conter os eventos, apresenta também os intervalos de tempo permitindo, dessa forma, a recuperação do comportamento e sua correspondente duração.

a2) Repetição.

Para o mecanismo de repetição, o uso do histórico de eventos é fundamental para permitir a reexecução de um programa. A possibilidade de reexecução viabiliza o uso de técnicas convencionais de depuração como: "breakpoints", exame de estado e execução passo a passo, isso sem o risco de alterar o comportamento do programa. Uma vez que o histórico reflita com fidelidade os eventos na ordem de ocorrência, a repetição produzirá resultados idênticos aos conseguidos durante a execução original.

LeBlanc [LEB87] utiliza a técnica de repetição em seu trabalho. Entretanto, sua abordagem não é adequada para ambientes de T.R. com restrições críticas de tempo, fundamentalmente por dois fatores: a natureza invasiva de seu monitor por ser do tipo "software", interferindo dessa forma no comportamento do programa depurado e pelo não tratamento do fator tempo.

b) A forma de aquisição.

A aquisição dos dados está estreitamente relacionada com a degradação no desempenho e a interferência nos tempos de operação que o depurador pode causar ao sistema monitorado. Com o objetivo de minimizar estes problemas, são propostas diversas maneiras de coletar dados do sistema sob depuração. Três abordagens muito utilizadas são:

- A inserção de comandos diretamente no código fonte, que se encarregam de enviar os dados para o monitor;
- A monitoração passiva dos barramentos, que apresenta a vantagem de ser a menos invasiva;
- A ligação ("link") do programa objeto, proveniente da compilação do programa fonte, com o monitor.

2.6.3 Análise estática.

A análise estática pode ser enquadrada entre as fases de depuração e a de teste. O elemento que a diferencia do teste em si, é o fato dela não necessitar da execução efetiva do programa sob análise. Isso faz com que a análise estática identifique problemas estruturais, ao invés de problemas funcionais, chegando até ao extremo da ferramenta de depuração não ter conhecimento da função realizada pelo programa que está sendo depurado.

O caso típico de aplicação da análise estática ocorre quando a interferência causada pela depuração dinâmica torna-se intolerável, não excluindo a possibilidade de uso da mesma como uma técnica complementar.

Deve-se ressaltar que embora existam semelhanças entre as técnicas de prova de correção de um programa e a análise estática, é o fato desta última não realizar a confrontação entre o programa e sua especificação que as diferencia, além da maneira de atuar no que tange à identificação de duas classes de erros por parte da análise estática. As classes de erros abordadas pela análise estática são:

- Sincronização, que está relacionada com o travamento da execução do programa, denominado "deadlock", e com o problema da espera infinita, normalmente relacionado com a troca de mensagens entre processos ou com o acesso de recursos compartilhados;
- Acesso a dados, onde podem ser identificadas duas vertentes dependendo do trecho do programa que está sendo analisado: se for um fragmento sequencial, o problema concentra-se na leitura de variáveis não inicializadas e, se o trecho for passível de concorrência, os erros estarão restritos a atualizações simultâneas de variáveis compartilhadas.

A pesquisa na área de análise estática pode ser desmembrada em dois ramos: técnicas "dataflow" e a análise da possibilidade de concorrência entre dois elementos de um programa.

a) Técnicas "dataflow".

Taylor e Osterweil [TAY80] apresentam um modelo simples de sincronização de processos. Neste modelo, o controle entre processos é realizado através das primitivas **schedule** e **wait**. A limitação desse modelo está associada com a não possibilidade de recursão.

O algoritmo utilizado apresenta as seguintes possibilidades de deteção:

- Referências a variáveis não inicializadas.
- Variáveis que são referenciadas enquanto estão sendo definidas em paralelo.
- Definição de uma variável que nunca é referenciada.
- Variável que pode apresentar um valor indeterminado.
- Processo aguardando o término de um outro processo não escalonado.
- Processo que está escalonado para executar em paralelo consigo mesmo.

As dificuldades inerentes ao uso das técnicas "dataflow" são apresentadas por Callaham e Subhlok [CAL88].

b) Possibilidade de concorrência entre elementos.

A verificação da possibilidade de concorrência entre elementos pode ser obtida a partir do uso de uma função booleana que tem por parâmetros de entrada os dois elementos candidatos à execução concorrente e como saída uma indicação positiva ou não. Esta função permite a deteção de erros de acesso, seja quando uma variável é escrita e lida simultaneamente, seja quando é escrita por dois processos de forma concorrente. Se o programa puder ser representado por uma rede de Petri [PET77], a função pode examinar os estados atingíveis da rede. O problema associado ao uso da rede é o fato de que o número de estados cresce exponencialmente com o número de nós.

O uso deste tipo de análise pode levar à geração de um relatório de erros que objetiva mostrar ao usuário o local do erro. Dois problemas podem ocorrer: o primeiro se relaciona com os caminhos não viáveis fornecidos pelo relatório que podem ser confundidos com erros pelo usuário, enquanto o segundo refere-se à maneira adequada de apresentar como o estado concorrente errôneo ocorre. A abordagem proposta por Applebe e McDowell [APP85] tenta solucionar o problema da apresentação de como o erro aparece, permitindo ao usuário acessar qualquer estado prévio ou posterior para facilitar o entendimento de como se chegou à situação de erro. A interface com o usuário é realizada através de várias janelas.

2.6.4 Combinação de análise estática e depuração dinâmica.

Algumas maneiras de se utilizar a combinação das duas técnicas são descritas por Taylor [TAY84]. Uma abordagem poderia ser o uso de análise estática para facilitar o desenvolvimento de dados de teste para uso em um depurador dinâmico. Por outro lado, a depuração dinâmica poderia ser usada como guia numa análise estática parcial, quando a análise completa acabasse por gerar um número impraticável de estados. A grande vantagem deste uso híbrido pode ser notada se for enfocado o problema da interferência causada pelos depuradores dinâmicos. No caso da análise estática demonstrar que partes de um programa estão isentas de erros estas, então, estarão livres de monitoramento, o que diminuiria a área a ser monitorada minimizando, dessa forma, os efeitos invasivos causados pelo depurador dinâmico.

2.6.5 Baseadas em comportamento.

Nesta abordagem, o processo de depuração consiste na comparação entre o comportamento do programa durante sua execução com o comportamento esperado, sendo este último definido mediante alguma especificação formal. Os elementos que servem de base para a dedução do comportamento durante a execução são os eventos. As diferenças são notificadas e o usuário tenta encontrar o erro através do estudo do relatório das discrepâncias.

A pesquisa desenvolvida atualmente enfatiza a necessidade de ferramentas que auxiliem no entendimento do comportamento dos programas. O entendimento é alcançado através da possibilidade de se obter diferentes

abordagens do programa, além de permitir analisá-lo sob níveis diversos de abstração.

A abstração de comportamento [BAT88] possibilita diferentes abordagens de um programa a partir de um conjunto de eventos primitivos que constituem o nível mais baixo. Este sistema utiliza uma linguagem denominada EDL. O uso de uma linguagem fornece facilidades na especificação de eventos, comportamento esperado (baseado em eventos primitivos ou definidos) e operações que são acionadas automaticamente quando da deteção de um determinado comportamento.

Com uma linguagem de definição de comportamento é possível impor uma especificação independente do comportamento esperado do programa. Baseado nessa especificação, o depurador verifica se o comportamento real se diferencia do esperado, além de permitir um mecanismo seletivo de relatório, evitando dessa forma a necessidade de se analisar grandes quantidades de dados. É notório que esta abordagem é mais poderosa que as técnicas tradicionais de depuração. Um exemplo é a inserção de "breakpoints", que geralmente é feita de maneira intuitiva e, com a utilização de uma linguagem de definição de comportamento, pode ser feito de um modo formal e seletivo.

2.6.6 Baseadas em I.A.

A depuração baseada em Inteligência Artificial utiliza técnicas de I.A. com três objetivos básicos:

- Ajudar na descoberta de erros.
- Sugerir possíveis causas.
- Propor soluções.

Para possibilitar o atendimento desses objetivos a metodologia utilizada é o sistema perito. Esse sistema consiste basicamente na criação de uma base de conhecimento da área onde se dará sua atuação (no caso o conhecimento de um perito em depuração) e de regras de inferência para a obtenção de informações. O comportamento do sistema pode ser encarado como o de um assistente na formulação de hipóteses, buscando encontrar o erro.

O sistema proposto por Gupta e Seviora [GUP84] utiliza uma base de conhecimento projetada para depurar um ambiente monoprocessado e concorrente, onde os processos interagem através de troca de mensagens. Obtido um conjunto de eventos da execução do programa, o depurador consulta a base para detetar qualquer sequência ilegal de mensagens e suas causas. Deve-se observar que o sistema proposto acusa somente a falha e não a localização do erro. Porém, este trabalho mostra perspectivas de utilização de técnicas de I.A. em depuradores.

2.7 Técnicas de apresentação dos dados monitorados.

Os programas sequenciais apresentam uma maior facilidade de exibição dos dados coletados por um depurador. Essa facilidade reside no fato destes programas possuirem somente um ramo de execução (que pode ser mostrado como um texto sequencial) e por seus dados estarem armazenados em um só local, além de serem facilmente acessados quando necessário. Por seu lado, os programas concorrentes normalmente apresentam diversos ramos de execução e muitas vezes não permitem um acesso fácil aos dados armazenados.

Uma das linhas de pesquisa em depuração concorrente busca encontrar a maneira mais adequada de apresentar os dados coletados durante a monitoração. Podemos identificar atualmente quatro técnicas básicas de apresentação:

- Texto.
- Diagramas tempo x processo.
- Animação.
- Múltiplas janelas.

2.7.1 Texto.

Esta é a forma mais comum e a mais utilizada em todos os sistemas de depuração. A apresentação textual dos dados usa recursos de cor e destaque por contraste para mostrar o fluxo das informações. O sistema Traveler [MAN87] proposto por Manning é baseado em objetos. Ele apresenta uma lista sequencial dos processos na ordem de acesso a um dado objeto compartilhado.

O fator tempo pode ser apresentado de modo não monotônico na tela. No Traveler, os eventos são exibidos em ordem causal ao invés de temporal. Sua aplicação se dá na depuração de programas que utilizam o mecanismo de troca de mensagens. O modelo engloba pares de requisições e respostas - "request" e "reply", denominadas transações.

A figura 2.5 mostra um exemplo de uma execução parcial. Para programas de maior tamanho não é possível uma mostra simultânea de todas as subtransações na tela. A saída é **abrir** e **fechar** transações de modo seletivo, conforme um critério imposto pela necessidade de visualização. A cada vez que uma transação é fechada, todas as subtransações à ela associadas são ocultas.

```
proc Exemplo = reply(request(A) + request(B) + request(C));
proc A = reply(request(X) + request(Y));
proc B = reply(b);
proc C = reply(c);
proc X = reply(x);
proc Y = reply(y);
```

FIG. 2.5 - PROGRAMA DE TRANSAÇÕES ESTILIZADO.

Na figura 2.6 se a transação request(A) for fechada então request(X) e request(Y), bem como as suas correspondentes linhas de resposta, não serão exibidas. As várias linhas "[sem resposta]", serão preenchidas no decorrer do processamento.

request(Exemplo)

request(A)

request(X)

[sem resposta]

request(Y)

reply(y)

[sem resposta]

request(B)

[sem resposta]

request(C)

reply(c)

[sem resposta]

FIG. 2.6 - SAÍDA DO DEPURADOR.

2.7.2 Diagramas tempo x processo.

Referem-se a uma representação bidimensional de um sistema concorrente, onde cada ponto do diagrama relaciona a informação do estado de um dado processo, com o tempo correspondente. A figura 2.7 mostra um exemplo de diagrama tempo x processo, para um sistema hipotético. Neste exemplo cada linha descreve os eventos ocorridos e cada coluna o estado do sistema em um instante de tempo específico.

Processo 1	A		D	F		...
Processo 2		B			G	...
Processo 3		C	E		H	...

FIG. 2.7 - DIAGRAMA TEMPO X PROCESSO.

Dois exemplos de sistemas que utilizam diagramas de tempo x processo são os propostos por Griffin [GRI87] e Harter [HAR85]. O primeiro utiliza o diagrama para mostrar as atividades em um sistema de memória compartilhada,

onde a unidade de tempo é a ocorrência de um evento. Já em Harter, o sistema enfocado usa o mecanismo de troca de mensagens. Uma vez que existe um elevado número de mensagens trocadas, o sistema permite a seleção de vários filtros, para exibir somente o conjunto de mensagens de interesse no momento.

2.7.3 Animação.

Uma alternativa para os diagramas tempo x processo é a utilização de animação. Esta técnica usa a tela inteira para representar o estado do sistema em um dado instante de tempo. A atualização da tela ocorre proporcionalmente com o passar do tempo, criando uma ilusão de continuidade à animação.

O posicionamento dos elementos (processos ou dados) a serem exibidos pode ser feito de três modos:

- Arbitrário.
- Sob o controle do usuário.
- Segundo a estrutura do programa.

No Belvedere [HOU87], o posicionamento dos processos é fundamental e especificado pelo usuário. Este sistema realiza a animação dos eventos primitivos e dos eventos definidos pelos usuários, especificados através da utilização da linguagem EDL. Para uma organização de tela mais adequada o usuário dispõe de três alternativas de exibição:

- processador.
- canais.
- item de texto.

A figura 2.8 mostra uma tela do Belvedere durante a execução do programa que resolve o problema do caixeiro viajante. A atividade é mostrada pelas linhas pontilhadas (canais) e pelos pequenos retângulos preenchidos (portas). A porta é preenchida durante uma recepção e o canal é feito pontilhado no envio. Setas indicam a direção para envios, sendo que múltiplas setas significam mais de uma mensagem no buffer.

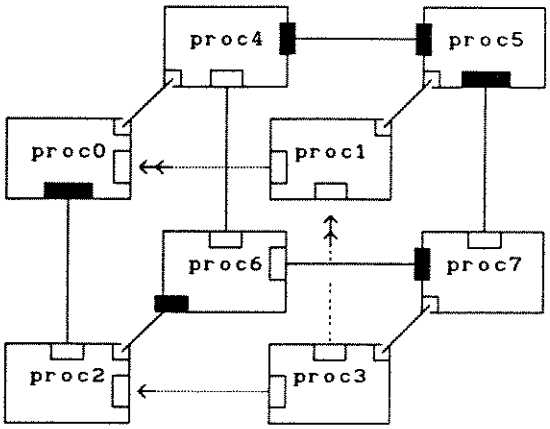


FIG. 2.8 - ANIMAÇÃO USANDO BELVEDERE.

A proposta de Socha [SOC89], com seu sistema Voyer, é simplificar a construção de aplicações que necessitem exibir o comportamento de programas concorrentes. O usuário pode utilizar os modos de exibição disponíveis no sistema, ou gerar novos, conforme sua necessidade. Quatro modos foram implementados na versão original do sistema:

- "Icon", no qual os eventos indicam o posicionamento dos "icons" e o momento de atualização da tela. Através da análise da posição dos "icons" e sua movimentação, diversos tipos de erros podem ser identificados;
- Vetor, que é uma variação do "icon", onde o elemento de visualização utilizado são vetores;
- "Trace", onde são utilizadas diversas janelas, uma para cada processo. Conexões entre processos são mostradas através de linhas interligando as janelas que contém os processos que se comunicam;
- Lista ligada, que é uma variação de "trace", com a diferença de que a a forma de representação utilizada é de uma estrutura de dados tipo lista ligada.

2.7.4 Múltiplas janelas.

Em sistemas de múltiplas janelas normalmente cada processo é associado a uma janela. Canais de E/S de cada processo, comunicações entre processos, além de semáforos e "mailboxes", também podem ser monitoradas através de janelas específicas. Um sistema de gerenciamento permite ao usuário selecionar o conjunto de janelas mais adequado para a depuração de uma dada aplicação.

2.7.5 Comparação entre animação e diagramas tempo x processo.

As técnicas de animação e diagramas tempo x processo não conseguem, quando usadas separadamente, ser eficientes a ponto de atender a todos as situações de depuração. Animação é adequada para o exame do estado instantâneo do sistema. Porém, ela não mostra claramente o comportamento do programa no decorrer do tempo. Diagramas tempo x processo conseguem exibir o padrão de comportamento ao longo do tempo, mas somente uma pequena quantidade de informação está disponível por processo em um dado instante.

Ideal seria a junção dessas técnicas em um único depurador, o que poderia ser conseguido através de duas maneiras. A primeira, a partir da utilização de duas janelas. Em uma delas o diagrama tempo x processo informa os pontos importantes no tempo e na outra estariam presentes informações detalhadas sobre o estado do programa de uma maneira mais abrangente, fornecida pela animação. Enquanto a segunda maneira consiste no uso de diversas janelas de animação simultâneas.

3.0 IMPLEMENTAÇÃO DE UM DEPURADOR PARA TEMPO REAL - DEBUGMOSI.

3.1 Histórico.

Após a implementação da família de núcleos de tempo real **MOSI**, (**QUASIMOSI** [AZE89a], **DEADMOSI** [AZE89b] e **LAXMOSI** [AZE91]), no CTI (Centro Tecnológico para Informática), em março de 1991, surgiu a necessidade de uma ferramenta que oferecesse facilidades para o desenvolvimento e depuração de aplicativos baseados nos mesmos. Dentre as facilidades exigidas figurava: o acompanhamento passo a passo da execução do aplicativo; a possibilidade de inserção de "breakpoints", além da gravação dos eventos ocorridos em disco com o propósito de gerar relatórios e de permitir uma simulação posterior. Esta última, devido ao comportamento não determinístico próprio de ambientes de multitarefa. Estava esboçado o perfil básico do depurador **DEBUGMOSI**, que apresenta como características principais: grande interatividade com o usuário através do uso de menus e janelas de diálogo; baixo "overhead" gerado no núcleo, devido ao uso de máquinas independentes que se comunicam por linha serial, e talvez, sua característica mais marcante, que é o fato de sua atuação se dar a nível de sistema, ou seja, os elementos tratados são: processos, semáforos, "mailboxes", etc, e não linhas de código. Esta abordagem permite uma depuração macroscópica aumentando consideravelmente a flexibilidade da ferramenta. Cabe aqui salientar que o depurador também pode ser utilizado no contexto de sistemas de tempo real crítico, através da opção simulação, usando um arquivo que contem os eventos ocorridos durante a execução do aplicativo. Este arquivo é gerado pelo próprio núcleo. Um maior detalhamento dessas opções será dado nos itens seguintes.

3.2 Características do depurador DEBUGMOSI.

O depurador DEBUGMOSI pode ser considerado como sendo um sistema híbrido. Ele utiliza suporte do núcleo de T.R., sob a forma de mensagens que contém os eventos ocorridos e um "hardware" que monitora o ambiente onde se encontra o aplicativo, através de uma linha serial que interliga sistema monitorado e depurador.

Uma cararcterística específica do DEBUGMOSI é de se prestar a duas abordagens distintas de depuração:

- Depuração com duas máquinas interligadas por linha serial;
- Depuração com uma só máquina.

O primeiro caso se aplica na fase inicial de depuração onde por exemplo se deseja observar erros de comportamento do sistema.

O segundo caso se utiliza de um arquivo contendo o histórico dos eventos ocorridos durante a execução do programa sob depuração. Este arquivo é gerado pelo prório núcleo de T.R. em um "buffer" de memória só passando para disco ao final da execucão do aplicativo sob depuração. Com isso, pretende-se gerar uma saída o mais fiel possível e, sem interferências causadas pelo próprio processo de obtenção de dados.

O arquivo gerado servirá como entrada do depurador que então poderá se utilizar das facilidades fornecidas pela ferramenta como, por exemplo, "breakpoints", supervisão de entidades, etc, sem o risco de com isso interferir na execução do sistema monitorado ou não conseguir reproduzir uma dada sequência de eventos.

Na interface com o usuário a técnica utilizada mescla texto, animação. além do uso de janelas. Os elementos como processos, semáforos e "mailboxes" são apresentados textualmente, porém, a atualização destes elementos na tela é dinâmica e acompanha a chegada dos eventos enviados pelo núcleo. Dessa forma é possível se evitar o uso de técnicas gráficas sofisticadas sem, contudo, comprometer o conteúdo da informação apresentada.

3.2.1 Comandos do depurador.

Para possibilitar a utilização do depurador nas modalidades linha serial e "replay" de eventos podemos identificar cinco grupos principais de comandos:

- Visualização.
- Execução.
- Supervisão.
- Estatística.
- Simulação.

a) Visualização.

A opção visualização engloba as funções responsáveis pela troca da tela a ser apresentada na janela principal. Uma vez que, por "default", o depurador, quando da sua inicialização, assume como tela de fundo as filas dos estados dos processos presentes no núcleo. As outras duas opções de tela disponíveis são: filas de semáforos e filas de "mailbox".

É possível, também, a exibição de uma janela contendo os "ticks" (unidade interna de medida de tempo do núcleo), o que proporciona um acompanhamento temporal dos eventos. A figura 3.1 mostra a estrutura da opção visualização.

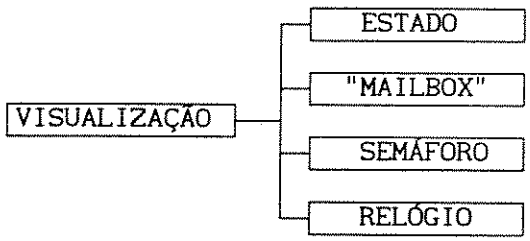


FIG. 3.1 - ESTRUTURA DA OPÇÃO VISUALIZAÇÃO.

b) Execução.

Para controlar o aplicativo sob teste, o depurador fornece a opção execução. Esta opção permite quatro abordagens de depuração:

- Execução simples, objetivando uma verificação preliminar do comportamento do aplicativo;
- Inserção de "breakpoints" nas seguintes situações: mudança no estado de um processo; quando um número específico de mensagens for atingido em uma "mailbox"; no momento em que o contador de um semáforo escolhido atingir determinado valor e, finalmente, quando o número de ativações de um tratador de interrupção ("handler") for atingido.
- Acompanhamento passo a passo da execução do aplicativo sob depuração com possibilidade de observação dos processos presentes nas filas, verificação dos contadores de semáforos, etc.
- Armazenamento em arquivo da execução em qualquer das opções apresentadas anteriormente, o que possibilita a comparação entre diversas execuções diferentes da mesma opção ou a verificação cruzada entre diferentes opções, além de servir de base para a simulação.

A figura 3.2 apresenta a estrutura da opção execução.

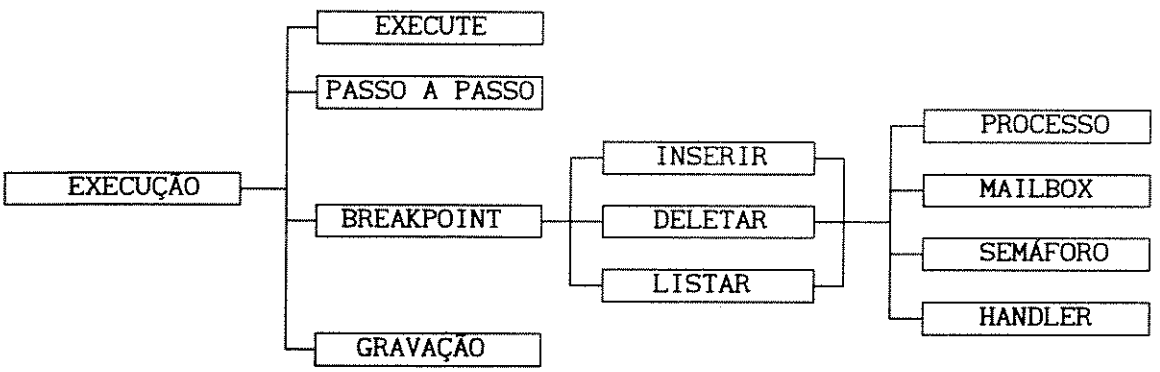


FIG. 3.2 - ESTRUTURA DA OPÇÃO EXECUÇÃO.

c) Supervisão.

Durante a depuração, principalmente no modo passo a passo, dada a sua alta interatividade com o usuário, existe a possibilidade de se obter informações adicionais sobre alguns elementos como processos, semáforos, "mailbox", "handler" e memória. Estes elementos estão englobados na opção supervisão.

Para processos temos os seguintes itens: tipo, quantidade de memória utilizada e estado. No caso de semáforos está disponível o valor do contador do semáforo, enquanto que para "mailbox" é apresentado o número de mensagens presentes na mesma. É fornecida a quantidade de memória livre e a quantidade ocupada no comando memória. No comando "handler" o que aparece é a classe, o nome do handler e o estado do tratador. A figura 3.3 mostra a estrutura da opção supervisão.

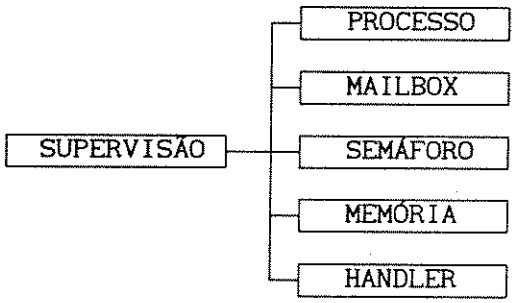


FIG. 3.3 - ESTRUTURA DA OPÇÃO SUPERVISÃO.

d) Estatística.

A opção estatística trata os elementos processo, "mailbox", semáforo, memória e sistema. Com relação a processo teremos os seguintes dados: tempo de execução, número de chamadas a semáforos e tempo médio de espera pela CPU. Nos demais casos são gerados gráficos da seguinte maneira:

- Número de processos x tick e número de mensagens x tick para a opção "mailbox".
- Processos x tick e valor do semáforo x tick para a opção semáforo.
- Memória livre em kbytes x tick e memória ocupada em kbytes x tick para a opção memória.
- Tempo ocioso x tick para a opção sistema.

A figura 3.4 mostra a estrutura da opção estatística.

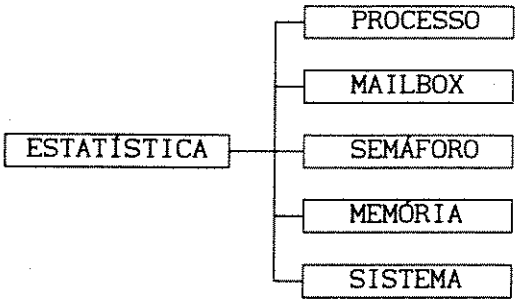


FIG. 3.4 - ESTRUTURA DA OPÇÃO ESTATÍSTICA.

e) Simulação.

Esta opção utiliza o arquivo gerado durante a execução, através da subopção de gravação, ou o arquivo de eventos criado pelo próprio núcleo de tempo real quando o aplicativo estiver utilizando a biblioteca "HRD", que permite a depuração de sistemas de tempo real HRT.

Possuindo as mesmas opções disponíveis na execução, com exceção da gravação, a simulação possibilita a observação do comportamento do aplicativo, sem a necessidade de sucessivas execuções do mesmo.

A figura 3.5 mostra a estrutura da opção simulação.

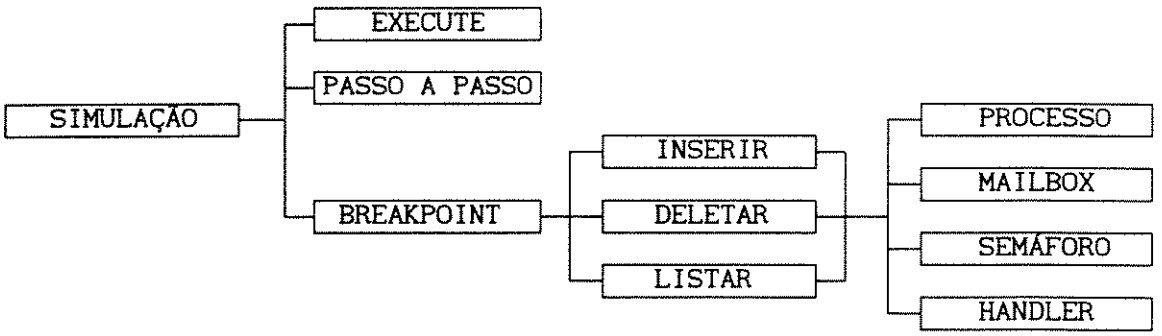


FIG. 3.5 - ESTRUTURA DA OPÇÃO SIMULAÇÃO.

3.3 Estrutura de dados.

A estrutura de dados do depurador foi concebida com o intuito de espelhar aquela utilizada pelos núcleos de tempo real da família **MOSI**. Este compromisso levou à adoção de uma estrutura composta dos seguintes elementos:

- Memória.
- Semáforo.
- "Mailbox".
- "Handler".
- Processo.
- Filas de estado dos processos.

As estruturas de semáforo, "mailbox", "handler" e processo, utilizam vetores para sua implementação. Sendo que, cada célula destes vetores contém informações sobre um único elemento, de acordo com o tipo de estrutura que compõe a célula. O núcleo quando da criação de semáforos, "mailboxes", "handlers" ou processos, estabelece um identificador, sendo este identificador o próprio índice do vetor onde será armazenada a informação, pertinente a esse elemento.

3.3.1 Memória.

A monitoração da quantidade de memória alocada para os processos, e a quantidade ainda disponível, é armazenada na estrutura de memória. A atualização da mesma é realizada a cada evento que faça uso da memória (alocando ou restaurando blocos), mesmo que este uso seja transparente para o usuário, como no caso da manipulação da pilha por parte de um processo.

3.3.2 Semáforo.

O acompanhamento de semáforos, pelo usuário, é realizado a partir do **nome** que foi estabelecido durante sua alocação, sendo este um dos itens da estrutura. O controle de preenchimento de cada célula do vetor é realizado pela variável **socup**, sendo a mesma booleana, com o valor um indicando célula ocupada. A estrutura apresenta ainda as variáveis **cont** e **swait**, sendo a primeira responsável pelo armazenamento do valor corrente do contador do semáforo e a última representando um ponteiro para o início da fila de processos que estão vinculados ao semáforo. As figuras 3.6 a e b mostram a estrutura de memória e a de semáforo, respectivamente.

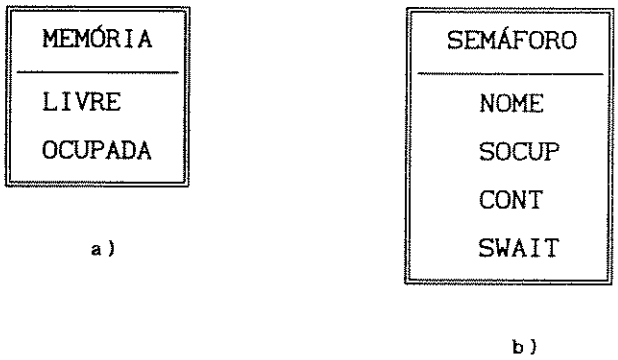


FIG. 3.6 - ESTRUTURA DE MEMÓRIA E SEMÁFORO.

3.3.3 "Mailbox".

No caso da troca de mensagens, realizada via "mailbox", a estrutura de dados é utilizada de maneira análoga àquela apresentada para semáforos, ou seja, um **nome** serve como referência para o usuário. Existe, também, um indicador de célula do vetor ocupada chamado **mocup**, além de um ponteiro (**mwait**) para a fila de elementos que estão relacionados com a "mailbox" e um contador do número de mensagens presentes na "mailbox" (**cont**).

3.3.4 "Handler".

Para o armazenar o estado dos tratadores de interrupções internas ou externas, o depurador utiliza uma estrutura de dados com: o **nome** do tratador, a variável responsável pela indicação de uso de uma célula do vetor **hocup**, a condição **habilitado** ou **desabilitado**, fornecida pela variável **estado** e, finalmente, o elemento usado como chave de acesso pelo usuário a **classe**, que informa qual é o conjunto de exceções que o tratador está capacitado a atender.

3.3.5 Processos.

A estrutura de processos é a mais complexa, englobando características detalhadas de cada processo. Ela está constituida dos seguintes elementos:

- **nome** do processo.
- **tipo** que pode ser prioridade, "deadline" ou periódico.
- **memória** utilizada.
- **estado** atual.
- **período** em "ticks".
- **tempo-comp** é o tempo de computação.
- **chave** de inserção nas filas de estado.
- **length** fornece o tamanho da área de parâmentros.
- **tempo** indica o tempo máximo que o processo ficará dormindo.
- **fila** do estado corrente do processo.
- **rfila** mostra a fila de recursos à qual o processo está vinculado.
- **pprox** aponta para o próximo processo da fila de estado.
- **rprox** aponta para o próximo processo da fila de recursos.
- **pocup** indica o uso de uma das células do vetor.

As figuras 3.7 a, b e c apresentam as estruturas de dados para "mailbox", "handler" e processo, respectivamente.

"MAILBOX"
NOME
MOCUP
CONT
MWAIT

a)

"HANDLER"
NOME
HOCUP
ESTADO
CLASSE

b)

PROCESSO		
NOME	PERÍODO	FILA
TIPO	TEMP-COMP	RFILA
POCUP	CHAVE	PPROX
MEMÓRIA	LENGTH	RPROX
ESTADO	TEMPO	

c)

FIG. 3.7 - ESTRUTURA DE "MAILBOX", "HANDLER" E PROCESSO.

3.3.6 Filas de estado dos processos.

As filas de estado agrupam as variáveis responsáveis pelo controle do estado corrente dos processos. A estrutura de filas contém os seguintes elementos:

- **exec** apresenta o processo que está em execução. Possui valor unitário uma vez que se trata de ambiente monoprocessado;
- **aguardando** é um ponteiro para a fila de processos que esperam pela liberação de um recurso;
- **dormindo** é um ponteiro para a fila de processos que estão em temporização. Esta temporização pode ser causada por um comando

direto, ou estar vinculada a um semáforo ou uma "mailbox";

- **suspenso** é um ponteiro para a fila de processos que foram criados com o "flag" de ativação desligado, ou aqueles processsos que explicitamente utilizaram a função de suspensão do núcleo;
- **pronto** é a única fila do depurador que apresenta dois níveis. Esse fato originou-se da necessidade de atender a todos os núcleos de tempo real da família **MOSI**, uma vez que na versão com escalonador EDF e LLF [AZE91], existe uma duplicação da fila de pronto, face à necessidade em tratar processos com prioridade e com "deadline".

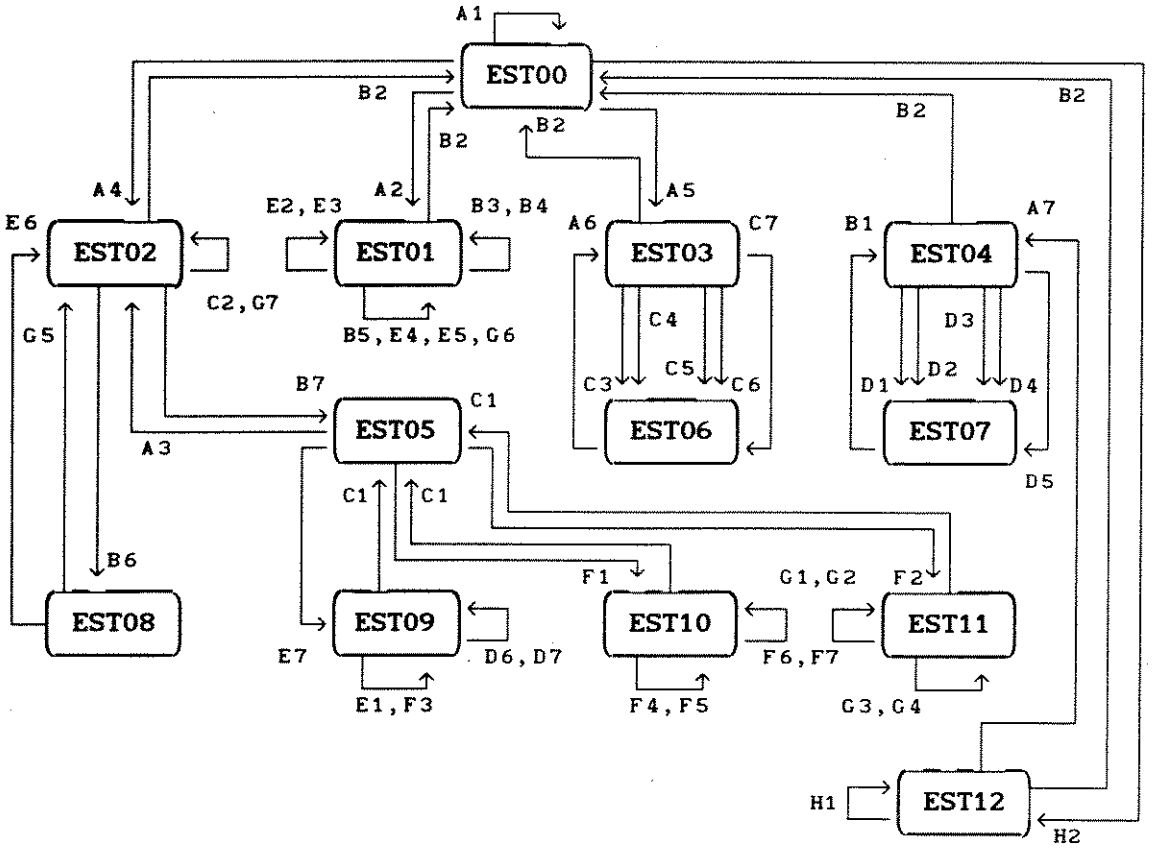
A figura 3.8 mostra a estrutura de dados para as filas de estado dos processos.

FILAS DE ESTADO	
EXEC	SUSPENSO
AGUARDANDO	PRONTO-PRIO
DORMINDO	PRONTO-DEAD

FIG. 3.8 - ESTRUTURA DE FILAS.

3.3.7 Descrição do autômato.

Como já citado no "pacote" de janelas, um autômato finito foi escolhido para implementar a máquina de estado, que coordena as transições entre estados do depurador. Um maior detalhamento do autômato revela que o mesmo é composto de 12 estados e de 46 funções. A figura 3.9a apresenta uma visão completa do modelo utilizado enquanto que a legenda corresponde é mostrada na figura 3.9b.



	1	2	3	4	5	6	7
A	EVSAI F00	EVVIS F01	EVESC F02	EVEXE F02	EVSUP F03	EVESC F03	EVEST F04
B	EVESC F04	EVESC F05	EVESTD F06	EVMAIL F07	EVSEM F08	EVEXCT F09	EVBRK F10
C	EVESC F10	EVPAS F11	EVPSUP F12	EVMSUP F13	EVSSUP F14	EVMMSUP F15	EVHSUP F16
D	EVPEST F17	EVMEST F18	EVSEST F19	EVMMEST F20	EVSIST F21	EVPRBR F22	EVSMBR F23
E	EVMLBR F24	EVUPAR F25	EVDNAR F26	EVPGUP F27	EVPGDN F28	EVESC F31	EVINBR F32
F	EVDLBR F33	EVLTBR F34	EVHDBR F35	EVPRBR F36	EVSMBR F37	EVMLBR F38	EVHDBR F39
G	EVPRBR F40	EVSMBR F41	EVMLBR F42	EVHDBR F43	EVDUM F44	EVREL F45	EVGRAV F46
H	EVSIMU F47	EVMAIS F48					

FIG. 3.9a - MÁQUINA DE ESTADO.

LEGENDA:

- FORMATO GERAL: ESTADOS / EVENTO / FUNÇÃO

ONDE OS EVENTOS SÃO:

EVSAI - EVENTO SAÍDA	EVEST - EVENTO ESTATÍSTICA
EVVIS - EVENTO VISUALIZAÇÃO	EVMAIL - EVENTO MAILBOX
EVESC - EVENTO ESCAPE	EVSEM - EVENTO SEMÁFORO
EVEXE - EVENTO EXECUÇÃO	EVEXCT - EVENTO EXECUÇÃO
EVSUP - EVENTO SUPERVISÃO	EVBRK - EVENTO BREAK
EVPAS - EVENTO PASSO	EVPSUP - EVENTO PROCESSO SUPERVISÃO
EVPEST - EVENTO PROCESSO ESTATÍSTICA	EVMSUP - EVENTO MAILBOX SUPERVISÃO
EVMEST - EVENTO MAILBOX ESTATÍSTICA	EVSSUP - EVENTO SEMÁFORO SUPERVISÃO
EVSEST - EVENTO SEMÁFORO ESTATÍSTICA	EVMMSUP- EVENTO MEMÓRIA SUPERVISÃO
EVMMEST - EVENTO MEMÓRIA ESTATÍSTICA	EVHSUP - EVENTO HANDLER SUPERVISÃO
EVSIST - EVENTO SISTEMA	EVPRBR - EVENTO PROCESSO BREAK
EVMLBR - EVEMTO MAILBOX BREAK	EVSMBR - EVENTO SEMÁFORO BREAK
EVUPAR - EVENTO UP ARROW	EVDNAR - EVENTO DOWN ARROW
EVPGUP - EVENTO PG UP	EVPGDN - EVENTO PG DOWN
EVINBR - EVENTO INSERE BREAK	EVDLBR - EVENTO DELETA BREAK
EVLTBR - EVENTO LISTA BREAK	EVHDBR - EVENTO HANDLER BREAK
EVPRBR - EVENTO PROCESSO BREAK	EVSMBR - EVENTO SEMÁFORO BREAK
EVMLBR - EVENTO MAILBOX BREAK	EVDUM - EVENTO DUMMY
EVREL - EVENTO RELÓGIO	EVGRAV - EVENTO GRAVAÇÃO
EVSIMU - EVENTO SIMULAÇÃO	EVMAIS - EVENTO MAIS
EVESTD - EVENTO ESTADO	

OS ESTADOS SÃO:

EST00 - ESTADO INICIAL	EST01 - VISUALIZAÇÃO	EST02 - EXECUÇÃO
EST03 - SUPERVISÃO	EST04 - ESTATÍSTICA	EST05 - BREAK
EST06 - SUP. ATIVA	EST07 - ESTAT. ATIVA	EST08 - EXECUÇÃO ATIVA
EST09 - INSERIR BREAK	EST10 - DELETAR BREAK	EST11 - LISTAR BREAK
EST12 - SIMULAÇÃO		

E AS FUNÇÕES SIMPLESMENTE IMPLEMENTAM A TRANSIÇÃO ENTRE OS ESTADOS.

FIG. 3.9b - LEGENDA DA MÁQUINA DE ESTADO.

As setas representam as transições entre os estados, enquanto o par letra-dígito é um indexador para a tabela, que faz corresponder a cada entrada na mesma o evento que gerou a transição, juntamente com a função que implementa essa transição. O identificador de cada seta aparece próximo à extremidade da mesma.

3.4 Ambiente operacional.

A adoção da estrutura de depuração descentralizada (aplicativo e depurador em máquinas separadas), teve como objetivo minimizar o efeito invasivo, normalmente causado quando da tentativa de se monitorar sistemas através de um elemento externo, sendo este efeito acentuado por se tratar de um ambiente com características concorrentes.

Para possibilitar a depuração de sistemas em desenvolvimento desde a fase inicial onde o fator tempo não é elemento crítico até as fases em que o tempo representa variável mandatória, duas abordagens são propostas: depuração sem restrição de tempo e com restrição de tempo.

3.4.1 Depuração sem restrição de tempo.

A máquina hospedeira prevista pelo depurador é um microcomputador compatível com IBM/PC. Esta família de microcomputadores possui como interface serial o padrão RS 232C, porém, o modo "default" de obtenção de caracteres utilizado é "pooling", não sendo este o ideal para nossa aplicação uma vez que o mesmo prevê "loops" de espera durante o recebimento e a transmissão de caracteres. Diante disso, foi necessária a reprogramação da interface através de seu "chip" controlador (8250), para o modo de comunicação controlado por interrupção. Neste modo, só existe atividade quando da transmissão e do recebimento efetivo de um caracter. A figura 3.10 ilustra a configuração adotada.

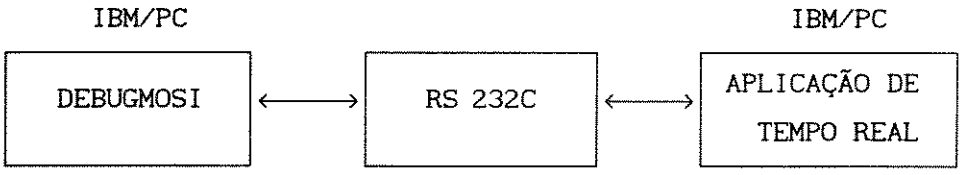


FIG. 3.10 - AMBIENTE OPERACIONAL PARA DEPURAÇÃO SEM RESTRIÇÃO DE TEMPO

3.4.2 Depuração com restrição de tempo.

Esta abordagem pressupõe o uso de somente um IBM/PC onde estará o aplicativo a ser depurado, devendo o mesmo ter sido previamente "linkado" com uma versão modificada dos núcleos de tempo real citados, que cria um arquivo de eventos da execução do aplicativo.

Para minimizar a influência causada ao aplicativo sob teste, os eventos são inicialmente armazenados em um "buffer" de 32K e, somente após o término da execução do aplicativo ou através de uma interrupção no processo causada pelo usuário é que a gravação em disco é realizada.

O arquivo gerado serve como entrada para o depurador que pode utilizar a mesma máquina onde foi realizada a execução do aplicativo.

A figura 3.11 ilustra o processo citado.

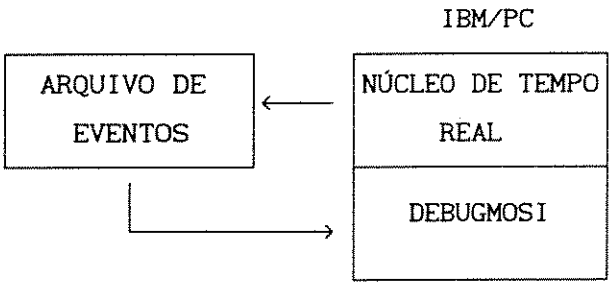


FIG. 3.11 - AMBIENTE OPERACIONAL PARA DEPURAÇÃO COM RESTRIÇÃO DE TEMPO

3.5 Módulos constituintes para ambas as abordagens.

O depurador pode ser estruturalmente particionado em três módulos principais: **GERÊNCIA DO SISTEMA**, **INTERFACE DE JANELAS** e **COMUNICAÇÃO SERIAL**. A figura 3.12 mostra o esquema geral dos módulos.

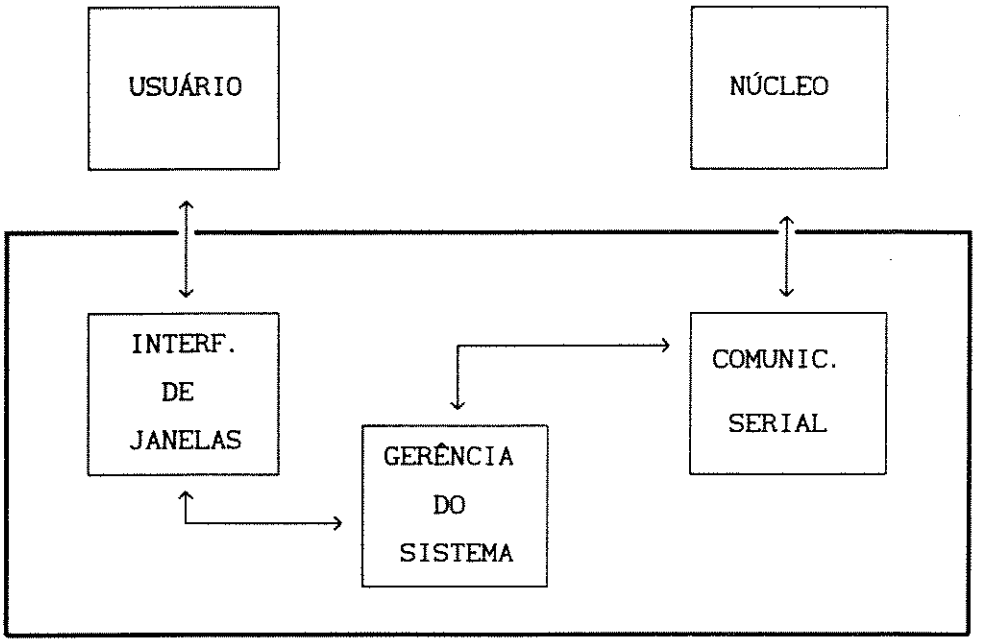


FIG. 3.12 - INTERLIGAÇÃO ENTRE MÓDULOS.

3.5.1 Gerência do sistema.

O módulo gerência do sistema tem por objetivos principais: o tratamento das mensagens (provenientes do módulo de comunicação serial, ou de um arquivo de eventos) e da administração da estrutura de dados, realizando o controle do acesso à mesma. Conflitos no uso dos dados são gerados pela necessidade concorrente de acesso à estrutura, por parte das informações provenientes da comunicação serial (ou do arquivo de simulação) e da interface de janelas. No primeiro caso o objetivo consiste na atualização da estrutura e no segundo na atualização da tela. A figura 3.13 representa a organização desse módulo.

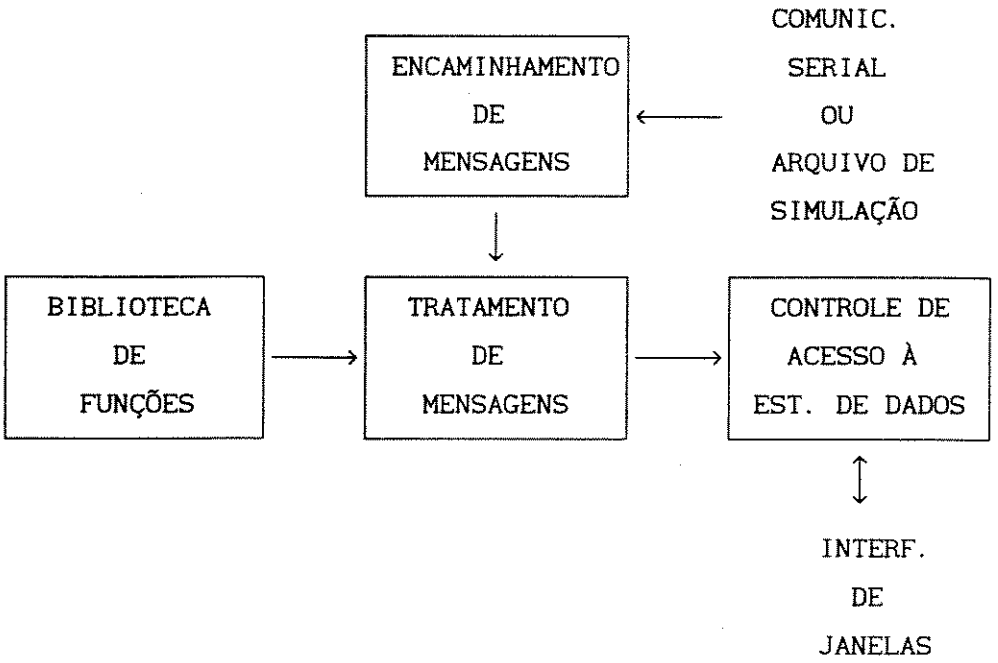


FIG. 3.13 - GERÊNCIA DO SISTEMA.

Para desempenhar suas tarefas, a gerência faz uso de três blocos principais, subdivididos em:

- Encaminhamento de mensagens.
- Tratamento de mensagens.
- Controle de acesso à estrutura de dados.

a) Encaminhamento de mensagens.

Fazendo uso de uma "mailbox" (estrutura de sincronismo e troca de mensagens entre processos), a função de encaminhamento recebe mensagens do módulo de comunicação serial e, pelo exame de seu código, estabelece o primeiro passo no tratamento das mesmas.

Independente de seu conteúdo, as mensagens possuem um conjunto de parâmetros denominados parâmetros básicos, sendo seu código o parâmetro básico inicial, que serve como índice para a localização da função que irá atender especificamente cada mensagem. A mensagem de maior comprimento terá no máximo dez parâmetros, além do código, conforme pode ser visto na figura

3.14. Esta separação em parâmetros tem por objetivo a padronização do tratamento das mensagens.

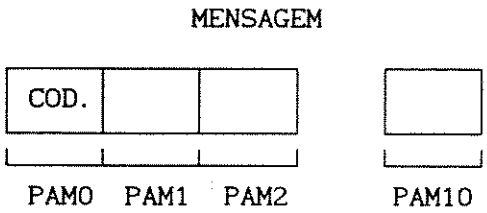


FIG. 3.14 - PARTICIONAMENTO EM PARÂMETROS.

b) Tratamento de mensagens.

O tratamento de mensagens faz uso de uma biblioteca de funções que facilitam a atualização da estrutura de dados do depurador. Entre os diversos procedimentos disponíveis, os mais relevantes são:

- Inserção e retirada de dados, em filas de um e dois níveis.
- Ativação de um processo.

A manipulação de filas está relacionada com o correto posicionamento dos processos na tela objetivando manter a coerência entre as atividades internas do núcleo e a representação fornecida pelo depurador.

c) Controle de acesso à estrutura de dados do depurador.

Uma vez que a estrutura de dados do depurador é um recurso comum, acessado concorrentemente pela interface de janelas e pela gerência do sistema, foi adotado o princípio da exclusão mútua, fazendo com que a estrutura seja vista como uma região crítica [BEN82] e exista, portanto, um protocolo de ingresso e saída da mesma, controlado por um semáforo como pode ser observado na figura 3.15.

```
wait(semaf)
    memoria livre   = pam1 ]
    memoria ocupada = pam2 ]   REGIÃO CRÍTICA
signal(semaf)
```

FIG. 3.15 - ACESSO À ESTRUTURA DE DADOS.

3.5.2 Interface de janelas.

O segundo módulo, chamado de interface de janelas, tem por objetivo facilitar a interação usuário/depurador, através do uso de uma interface "amigável", composta de diversas janelas que se sobrepõem e da escolha de alternativas, realizada através de menus. Este módulo pode ser visto como a junção de um "pacote" de janelas, proposto por Stevens [STE87], e do gerenciamento deste pacote, através de funções adicionais, como mostra a figura 3.16.

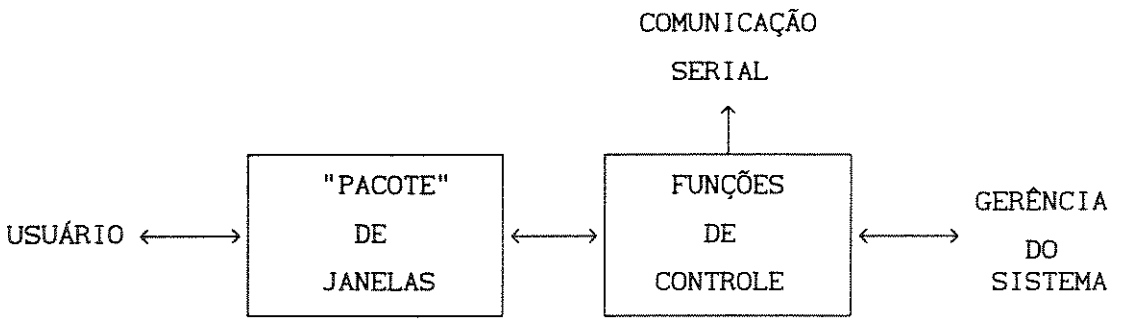


FIG. 3.16 - INTERFACE DE JANELAS.

O "pacote" apresenta cinco grupos principais de funções, relacionadas com as seguintes atividades:

- Acesso à memória RAM de vídeo.
- Manipulação de janelas.
- Manipulação de menus.
- Entrada de dados.
- Gerenciamento de janelas.

a) Acesso à memória RAM de vídeo.

Funções de baixo nível acessam a memória RAM de vídeo para realizar o posicionamento, leitura da posição e mudança de tipo do cursor. Leitura do teclado, escrita de caracteres, além da limpeza da tela, também são tarefas enquadradas neste grupo.

b) Manipulação de janelas.

Os procedimentos relacionados com a manipulação de janelas constituem o núcleo do pacote. Eles permitem a criação das mesmas com diferentes tipos de bordas, várias cores (incluindo variações entre "background" e "foreground"), além da inclusão de títulos. Uma característica útil é a possibilidade de alteração dos parâmetros de visibilidade, conseguindo-se assim janelas ocultas ("hidden window") ou visíveis. Naturalmente, está prevista a destruição de uma dada janela,com a consequente liberação dos recursos a ela alocados, fazendo com que a mesma não possa ser mais acessada.

c) Manipulação de menus.

No caso dos menus, a flexibilidade quanto à sua forma (horizontais ou verticais) e seu posicionamento na tela, além da possibilidade de escolha de um número qualquer de opções (número este limitado pela própria tela), são características oferecidas neste grupo. O depurador utilizou-se de um menu horizontal contendo no máximo cinco opções, selecionadas por uma barra em vídeo reverso e controladas pelas setas do teclado.

d) Entrada de dados.

A entrada de dados é realizada através da exibição de um ou mais campo(s) em vídeo reverso, de comprimento previamente especificado em uma janela, arbitrariamente escolhida, para este fim. Um processo de validação é realizado, a cada entrada, para cada um dos campos, garantindo dessa forma a consistência dos dados. No depurador, em caso de entrada inválida, uma mensagem de advertência é exibida na janela de diálogo. Após o que, é

permitida uma nova entrada, ou o abandono da mesma. A cada entrada válida, os dados tentam acessar uma estrutura compartilhada, cujo controle de acesso é realizado pela gerência do sistema. No caso da entrada de dados relacionar-se com o estabelecimento ou remoção de "breakpoints", será permitida uma conexão direta entre interface de janelas e comunicação serial.

e) Gerenciamento de janelas.

O gerenciamento do "pacote" de janelas pode ser atribuido a um grupo de funções, subdividido da seguinte maneira:

- Autômato.
- Biblioteca de procedimentos executados pelo autômato.
- Gerenciamento de telas/menus.

e1) Autômato.

Optou-se pela utilização de um autômato finito como elemento de controle. Na sua implementação utilizou-se um vetor, sendo cada elemento do mesmo constituido pela estrutura de dados apresentada na figura 3.17.

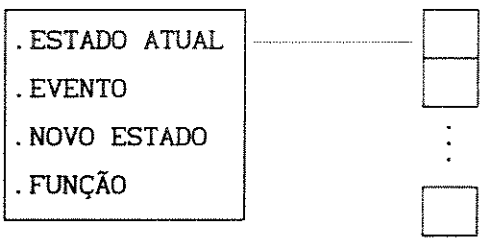


FIG. 3.17 - MODELO GERAL DO AUTÔMATO.

A mecânica de funcionamento baseia-se no conceito de que uma vez atingido um **ESTADO ATUAL**, (que estabelece todo um conjunto de características), para efetuar-se a mudança para um **NOVO ESTADO** é necessária a ocorrência de um **EVENTO** que, através da execução de uma **FUNÇÃO** previamente estabelecida, promove a transição entre estados.

e2) Biblioteca de procedimentos do autômato.

Uma biblioteca contém as funções exigidas para o funcionamento adequado do autômato. As mais relevantes são: inserção, remoção e listagem de "breakpoints"; troca dinâmica dos menus apresentados na tela e controle de "scroll", necessário quando os dados presentes nas filas ultrapassam os limites impostos pela tela.

e3) Gerenciamento de telas/menus.

Para se conseguir a integração e o controle dos elementos apresentados tornou-se necessária a presença de um gerenciador, cujas atividades principais são:

- Inicialização e controle de tela.
- Atualização de janelas.
- Envio de dados teclados ao autômato.

e3.1) Inicialização e controle de tela.

Realiza o posicionamento dos elementos de tela, quando da primeira apresentação da mesma, ao usuário. Estes elementos são: a janela principal, onde são apresentadas as informações colhidas, junto ao núcleo, as janelas de diálogo e de informações sobre a movimentação do cursor, além do menu principal. O controle de tela ocupa-se com o correto posionamento das filas de estado de processos, semáforos e "mailboxes", que refletem o comportamento do aplicativo sob teste. A figura 3.18 é um exemplo da tela exibida ao usuário.

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC      PRT-DEAD   PRT-PRIO   DORMINDO   AGUARDA   SUSP

PRC1-19   PRC3-17    NULO-0     PRC8-18              PRC4-10
          PRC5-15
          PRC7-13

VISUALIZAÇÃO   EXECUÇÃO   SUPERVISÃO   MAIS...   SAÍDA
```

FIG. 3.18 - EXEMPLO DE TELA.

e3.2) Atualização de janelas.

Quando da depuração do aplicativo, se uma janela específica se encontrar em um plano superior às demais, não existe perda das informações apresentadas nas janelas inferiores, uma vez que uma atualização dinâmica, em todas as janelas visíveis, (mesmo aquelas parcialmente encobertas), é realizada.

e3.3) Envio de dados teclados ao autômato.

O envio de informações ao autômato é realizado com a preocupação em não bloquear o processamento do depurador, fazendo-o permanecer em "loops", até que um caracter seja digitado. Em vez disso, é utilizada uma estratégia na qual um processo de baixa prioridade monitora o teclado e, somente quando um dado está disponível, este é encaminhado ao autômato.

3.5.3 Comunicação serial.

O módulo de comunicação serial, como pode ser observado na figura 3.19, é formado por três camadas que apresentam as seguintes funções:

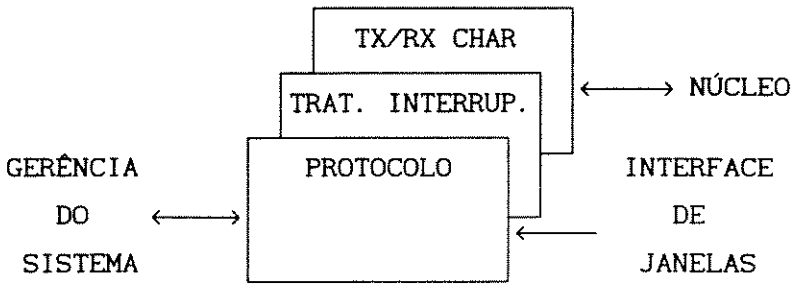


FIG. 3.19 - CAMADAS DA COM. SERIAL.

- Reprogramação do hardware e TX/RX básicos.
- Tratamento de interrupção.
- Protocolo.

a) Reprogramação do "hardware" e TX/RX básicos.

No nível mais baixo, encontram-se funções responsáveis pela reprogramação da interface serial e do controlador de interrupções do PC (8259) (isto devido ao uso de interrupção para o tratamento da comunicação), além de procedimentos básicos para a transmissão e recepção de caracteres. Esta camada é implementada em assembler devido às suas características de estreita interação com o hardware, sendo o único módulo a não utilizar linguagem "C". Os erros de comunicação detetados (overrun, paridade e frame), são enviados à camada superior e, assim, sucessivamente, até atingir o usuário. Esta filosofia foi a adotada em todos os demais módulos durante a implementação.

b) Tratamento de interrupção.

Neste nível ocorre o aparecimento de funções para a inserção e retirada de caracteres de dois "buffers" circulares. Estes "buffers", um para transmissão e outro para recepção, são utilizados com o intuito de prover um armazenamento temporário para os caracteres recebidos e transmitidos, até que a rotina de tratamento de interrupção possa processá-los.

São utilizados semáforos para realizar a sincronização dos processos responsáveis pela comunicação entre depurador e aplicativo visto que o próprio depurador é um aplicativo que utiliza o núcleo **QUASIMOSI**.

c) Protocolo.

A última camada deste módulo realiza a transmissão e a recepção de "pacotes" de caracteres. Estes "pacotes" obedecem a um protocolo cujo formato é ilustrado pela figura 3.20. As funções desta camada permitem o estabelecimento de "timeout", ou seja, a especificação de um tempo máximo de espera pela resposta do receptor da mensagem. A não ocorrência do evento dentro deste prazo, libera os processos envolvidos na comunicação e gera uma sinalização de erro. Outra característica desta camada é o controle do número máximo de retransmissões permitido a cada mensagem. Caso este número seja ultrapassado uma mensagem de advertência é apresentada na janela de diálogo. Uma retransmissão é gerada a cada recebimento de um **NACK** (mensagem não recebida com sucesso), por parte do transmissor. Em caso de comunicação bem sucedida, um **ACK** é obtido como resposta. Além disso, as mensagens enviadas recebem uma ordenação (de **A** a **Z**), permitindo a deteção de erros de perda de sequência entre mensagens.

O anexo A-1 contém uma listagem completa dos tipos de mensagens trocadas entre o depurador e o núcleo.

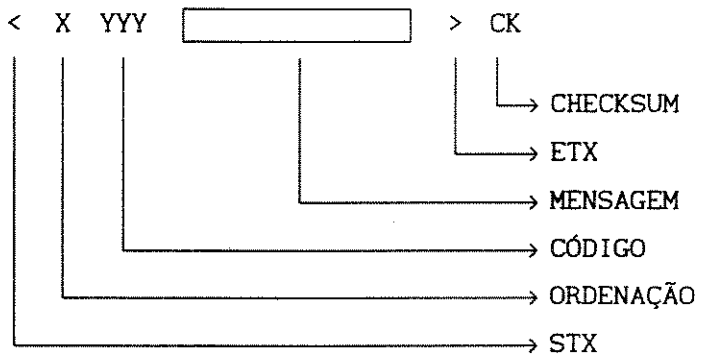


FIG. 3.20 - PROTOCOLO DE COMUNICAÇÃO.

Para promover um maior nível de detalhamento sobre os módulo que constituem o depurador, as figuras 3.21 a, b e c, mostram as funções que implementam as tarefas descritas anteriormente, agrupadas por módulo.

COMUNICAÇÃO SERIAL		
INTER.ASM	-	LÊ/ESC CHAR
COMUNIC.C	-	TRAT. INTERRUP.
PROT.C	-	PROTOCOLO

a)

GERÊNCIA DO SISTEMA		
DEBMOSI.C	-	MOD. PRINCIPAL
MSG.C	-	CTRL ESTRUTURA
BIB.C	-	BIBLIOTECA
TRATAMSG.C	-	TRATAMENTO MSG

b)

INTERFACE DE JANELAS	
"PACOTE"	F. ADICIONAIS
ENTRY.C - ENTRADA DE DADOS	AUT_USER.C - AUTÔMATO
IBMPC.C - ACESSO RAM VÍDEO	FUNC.C - BIBLIOTECA
TWINDOW.C - MANIP. JANELAS	GRAFICO.C - GER. TELA/MENU
TMENU.C - MANIP. MENUS	

c)

FIG. 3.21 - DESCRIÇÃO DAS FUNÇÕES POR MÓDULOS.

3.6 Modo de operação do depurador.

A dinâmica de funcionamento do depurador pode ser vista em função dos processos que o constituem. A cada módulo apresentado anteriormente está associado um processo, com uma prioridade (uma vez que o núcleo utilizado pelo depurador possui um escalonador baseado em prioridade), cujo valor é inversamente proporcional ao grau de importância do mesmo. Os principais processos presentes, em ordem de prioridade, são:

- simula.
- func_rec.
- trata_msg.
- interface.
- grava_disco.

O processo **func_rec** é responsável pelo tratamento dos caracteres recebidos pela comunicação serial. Ele está vinculado ao semáforo **rec**, que é sinalizado, quando do tratamento de interrupção de recepção.

Para possibilitar o direcionamento das mensagens recebidas, e seu correto atendimento, as mesmas são colocadas em uma "mailbox" denominada **rec-com1**. O processo **trata_msg** realiza uma operação de "receive" nesta "mailbox", ficando bloqueado até que uma mensagem esteja disponível. Uma vez que as mensagens recebidas devem atualizar a estrutura de dados, e o acesso a esta última é controlado pelo semáforo **sdados**, o processo também é dependente desse semáforo. Em se utilizando a opção simulação, o processo **simula** é o responsável pelo envio das mensagens presentes no arquivo de eventos para a "mailbox" **rec-com1**.

A atualização de tela é realizada por intermédio do processo **interface**. Este processo necessita para este fim, que haja permissão de acesso à estrutura de dados, para a leitura dos mesmos, e também que o recurso tela esteja disponível, uma vez que este é um recurso compartilhado entre diversas funções de atualização. O controle de acesso à estrutura de dados, como já mencionado, é realizado pelo semáforo **sdados**, enquanto o de tela, pelo semáforo **s_atual**. Estes semáforos são sinalizados pela gerência do sistema e pela última função a atualizar a tela, respectivamente.

A gravação dos eventos ocorridos durante a execução do aplicativo é armazenada em dois "buffers" de 512 bytes cada. A cada vez que um deles é preenchido, o semáforo **sem_dsk** é sinalizado, liberando o processo **grava_disco** para realizar a atualização do arquivo de saída com o conteúdo presente no "buffer" em uso. A presença de dois "buffers" é uma cautela para garantir que os dados presentes em um deles possa ser gravado em disco, sem sofrer sobreposição por novos dados, devido ao fato do processo de gravação ser o menos prioritário, e somente atuar quando nenhum outro estiver ativo. A figura 3.22 mostra a interconexão entre os processos citados anteriormente.

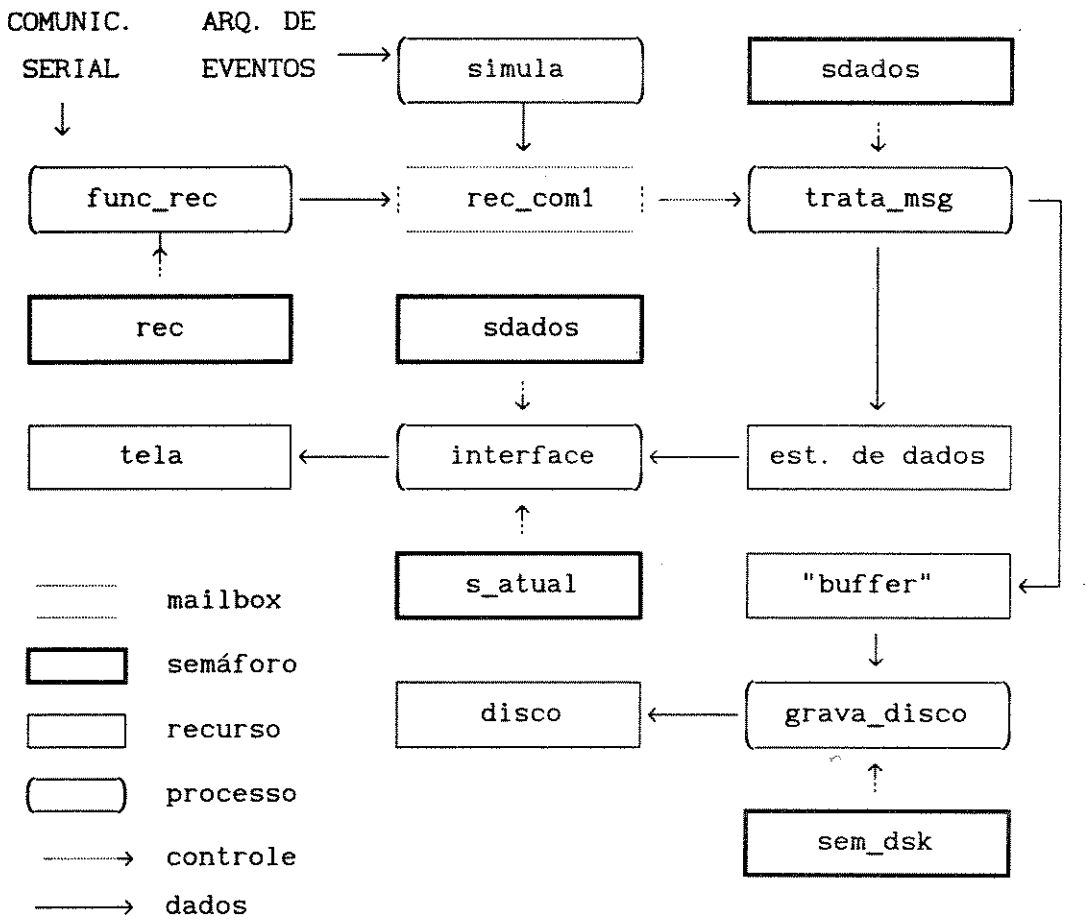


FIG. 3.22 - RELAÇÕES ENTRE PROCESSOS

4.0 EXEMPLOS DE UTILIZAÇÃO DO DEPURADOR DEBUGMOSI.

4.1 Modalidades do depurador.

A estratégia de se adequar o depurador a um sistema sem restrição rígida de tempo (SRT) ou a um sistema com restrição rígida de tempo (HRT), tem sua origem na escolha da biblioteca (lib) gerada pelo configurador dos núcleos de tempo real os quais o DEBUGMOSI suporta. Esta biblioteca será ligada juntamente com a aplicação.

As três bibliotecas disponíveis para os aplicativos do usuário são:

- <nome_da_biblioteca>.sft
- <nome_da_biblioteca>.hrd
- <nome_da_biblioteca>.lib

A biblioteca **.sft** é voltada para a depuração através do uso da linha serial como meio de envio de eventos do núcleo ao depurador. Sendo, portanto, adequada a sistemas SRT ou na fase inicial de sistemas HRT.

Já a biblioteca **.hrd** permite o armazenameto dos eventos gerados pelo núcleo na memória da máquina onde se encontra o aplicativo. Ao término da execução do mesmo é criado um arquivo de simulação. Esse arquivo pode ser utilizado para se conseguir um "replay" da execução do sistema sob depuração, sem a interferência causada à aplicação pelo uso da linha serial.

Duas observações são pertinentes: a primeira se refere ao tamanho do "buffer" utilizado para armazenamento dos eventos, que é da ordem de 32K e pode ter seu tamanho configurado pelo usuário. A segunda enfatiza que a sequência natural na depuração de um sistema HRT deve passar inicialmente por uma depuração com uma biblioteca **.sft** para que os erros mais evidentes possam ser detectados, principalmente aqueles como "deadlock". Só então o usuário deve iniciar a depuração com a biblioteca **.hrd**.

No caso da biblioteca **.lib**, é recomendado que seja utilizada após o término da fase de depuração, uma vez que a mesma não apresenta qualquer tipo de suporte à depuração.

A seguir é dado um detalhamento dos dois tipos de sistemas passíveis de depuração pelo DEBUGMOSI.

4.1.1 Sistemas sem restrição rígida de tempo (SRT).

Esta modalidade de sistemas comporta uma depuração "on line", realizada via linha serial (figura 4.1) . Opcionalmente, pode-se gerar um arquivo dos eventos da execução, através da opção **gravação** (figura 4.2 a), para possibilitar o "replay" da mesma e/ou gerar um relatório de tempos x eventos, através de um utilitário denominado **relmosi** (figura 4.2 b). Neste item também podem ser enquadrados aqueles sistemas HRT que estejam em uma fase inicial de sua implementação, e onde a variável tempo ainda não esteja sendo considerada.

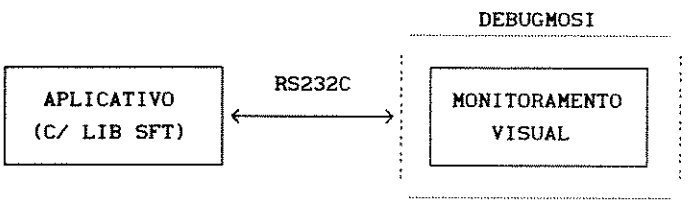


FIG. 4.1 - DEPURAÇÃO "SFT" S/ ARMAZENAMENTO.

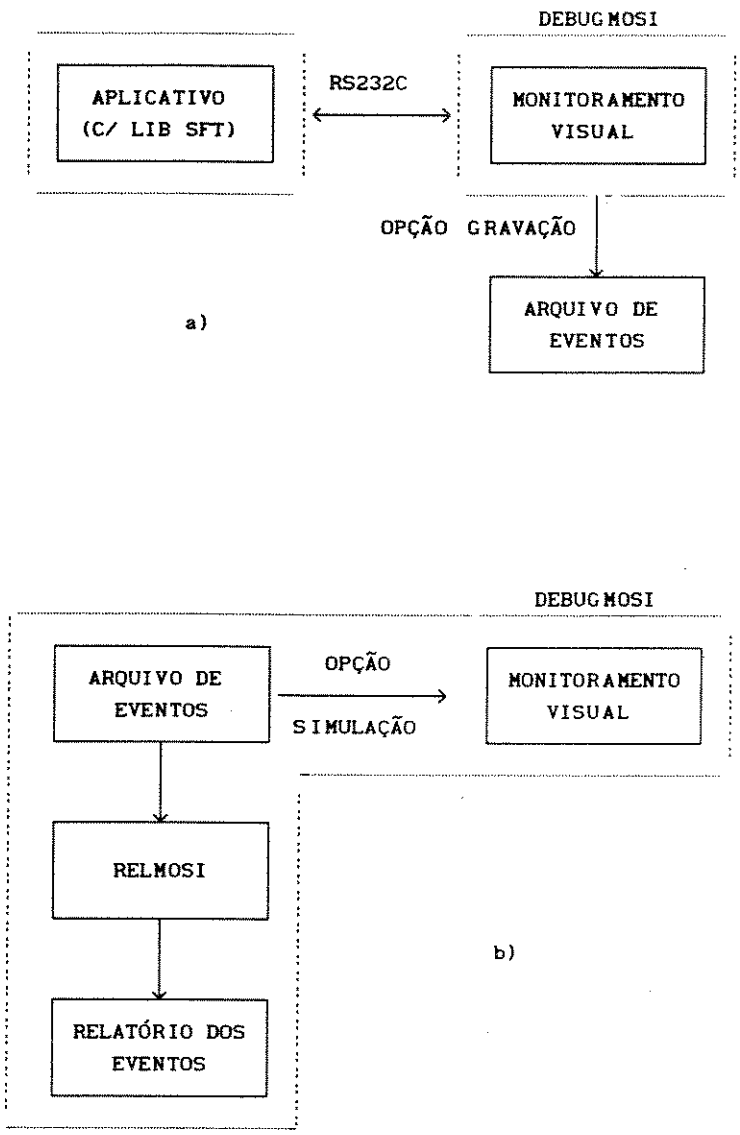


FIG. 4.2 - DEPURAÇÃO "SFT" C/ ARMAZENAMENTO EM ARQUIVO.

4.1.2 Sistemas com restrição rígida de tempo (HRT).

A depuração deste tipo de sistema necessita da geração de um arquivo de eventos de sua execução, o que é realizado pelo próprio núcleo de tempo real (figura 4.3 a). Este arquivo será entrada para o depurador na opção **simulação** e opcionalmente pode ser base para a geração de um relatório de execução. Com o artifício do "replay" a repetição de um dado comportamento (eventualmente errático) se torna viável (figura 4.3 b).

UNICAMP
BIBLIOTECA CENTRAL

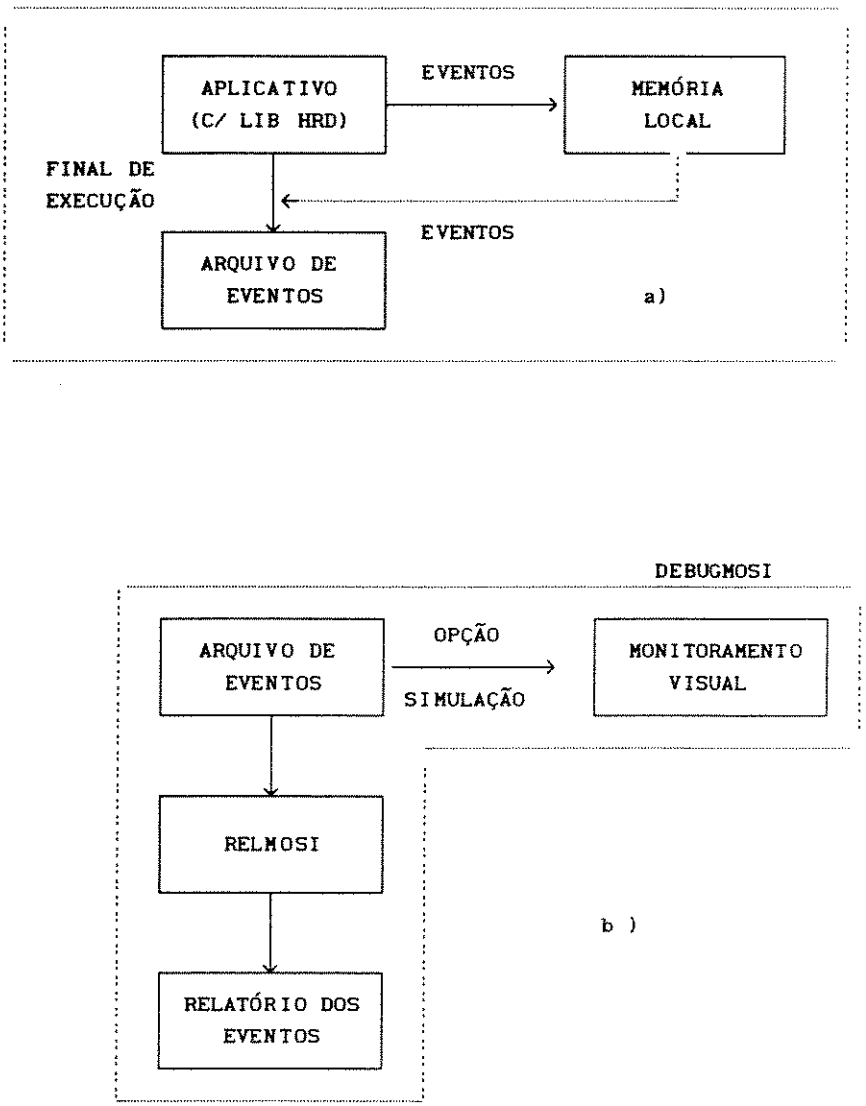


FIG. 4.3 - DEPURAÇÃO "HRT".

4.2 Estudo de caso.

Com o objetivo de ilustrar algumas possibilidades de utilização do depurador, serão apresentados três cenários :

- inversão de prioridade
- protocolo limite de prioridade
- sub e super estimativa dos tempos de computação

Estes cenários terão por base a simulação de um sistema real de controle de nível e temperatura de um líquido, como pode ser observado na figura 4.4.

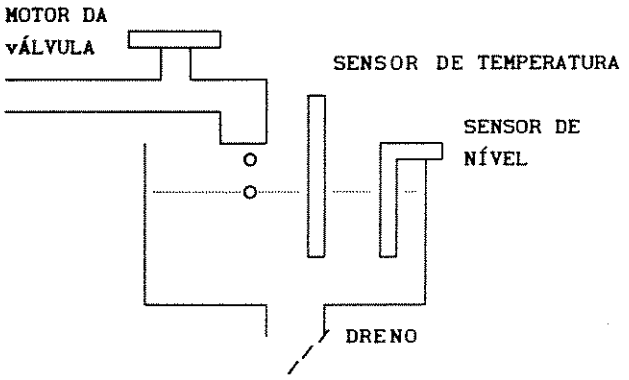


FIG. 4.4 - SISTEMA SIMULADO.

Na simulação serão utilizadas sete tarefas:

- **SIMU** responsável pela variação aleatória nos valores de temperatura e quantidade de fluido presentes no tanque.
- **ALRM** monitora o nível e a temperatura para evitar valores inadequados. Se um valor fora da faixa permitida for encontrado, a tarefa imprime uma mensagem e realiza os procedimentos necessários para a normalização do sistema.
- **CTRL_V** calcula valores para a variável quantidade de fluido ("set-point").
- **CTRL_H** idem ao anterior para a variável temperatura.
- **SUPERV** atualiza os valores das variáveis quantidade de fluido e temperatura.
- **DATA_LOG** imprime informações gerais do andamento da simulação.
- **BACKG** implementa a interface com o usuário.

Esta simulação não lançará mão de cálculos de controle, uma vez que seu objetivo principal é demonstrar a dinâmica de funcionamento do depurador.

4.2.1 Inversão de prioridade.

Neste exemplo as prioridades dos processos são fixas como pode ser observado na figura 4.5. Deve-se notar que quanto menor o valor maior sua prioridade.

Identificação	Valor	Tipo
simu	10	Prioridade
alrm	20	''
ctrl_t	30	''
ctrl_v	35	''
superv	40	''
data_log	50	''
backg	60	''

FIG. 4.5 - PROCESSOS COM PRIORIDADE FIXA.

O ambiente de depuração utilizado é o mostrado na figura 4.1, com uso de biblioteca **.sft** e duas máquinas distintas conectadas por linha serial.

O problema de inversão de prioridade é causado pela necessidade de sincronização dos processos que compartilham recursos lógicos ou físicos, cujo acesso seja controlado por algum mecanismo de sincronização como semáforos. Como resultado, temos um processo de maior prioridade bloqueado pelo de prioridade menor por um intervalo de tempo indefinido.

Com o uso do depurador, na opção **passo a passo**, obteremos a sequência de telas mostradas a partir da figura 4.6.

No instante inicial o processo **DATA_LOG** (data-13), de menor prioridade, é ativado e entra na região crítica controlada pelo semáforo **sem_tela**. Seu objetivo é imprimir na tela o valor atual das variáveis de nível e temperatura.

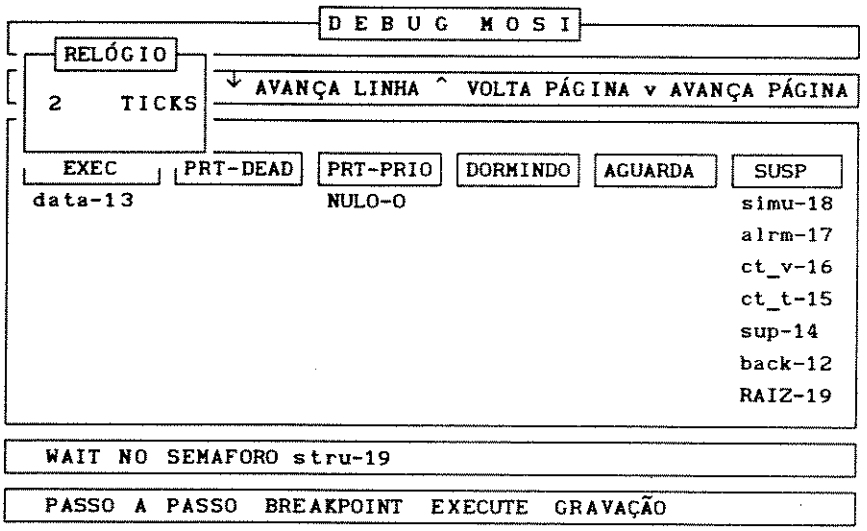


FIG. 4.6 - Processo data acessa RC.

Neste instante o processo **SIMU** (simu-18), o de mais alta prioridade do sistema, é escalonado e tenta acessar a mesma região crítica para poder atualizar os valores das mesmas variáveis.

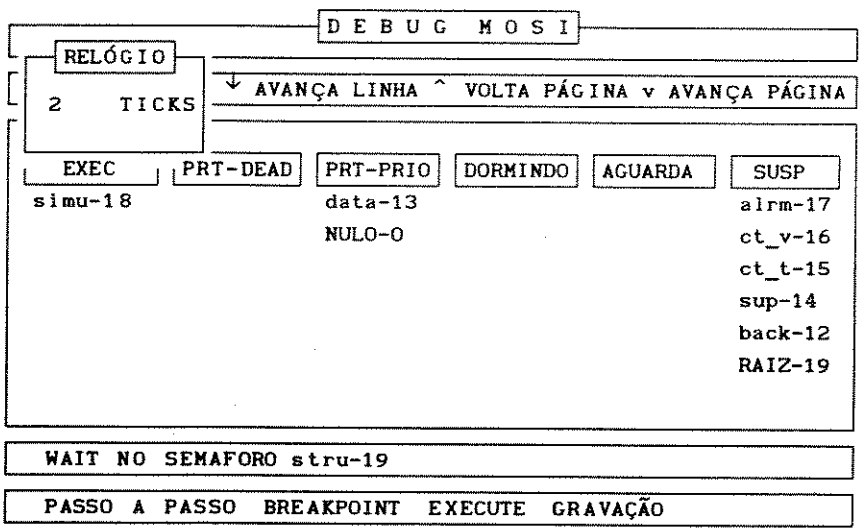


FIG. 4.6 a - Processo simu tenta acessar RC ocupada.

Como consequência o processo **SIMU** (simu-18) é bloqueado. E o processo **data_log** (data-13) volta a executar.

No próximo instante de tempo o processo **ALRM** (alrm-17) é ativado e executa por ter maior prioridade.

Após o término do processo **ALRM** (alrm-17) os demais processos, respeitada suas prioridades, são ativados e executam, uma vez que eles não dependem da região crítica protegida pelo semáforo **sem_struct**.

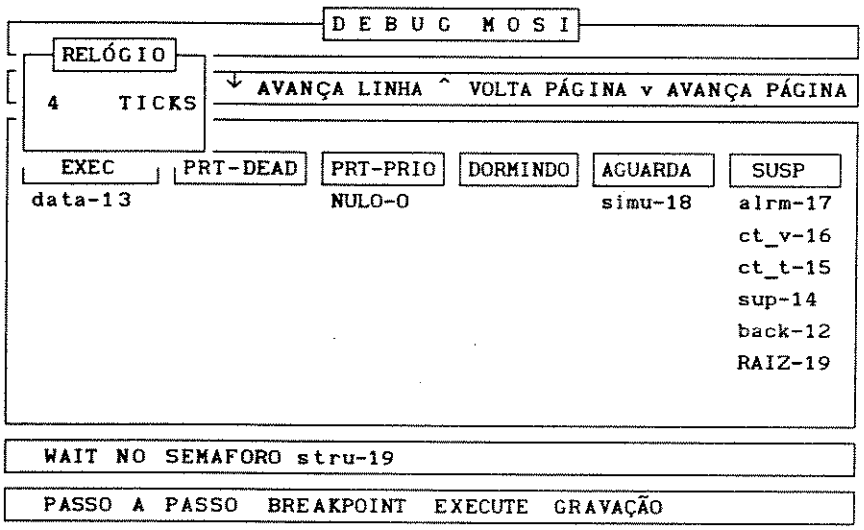


FIG. 4.6 b - Processo simu na fila de sem_struct.

Só então o processo **DATA_LOG** (data-13) retoma a CPU e sai da região crítica. Realizando um **signal** no semáforo **sem_struct**.

```
D E B U G   M O S I
RELÓGIO
664   TICKS
↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC      PRT-DEAD  PRT-PRIO  DORMINDO  AGUARDA  SUSP
data-13             NULO-0    alrm-17   simu-18  RAIZ-19
                              ct_v-16
                              ct_t-15
                              sup-14
                              back-12

SIGNAL NO SEMAFORO stru-19

PASSO A PASSO  BREAKPOINT  EXECUTE  GRAVAÇÃO
```

FIG. 4.6 c - Processo data realiza signal em sem_struct.

O que finalmente permite que **SIMU** (simu-18) termine sua execução.

```
D E B U G   M O S I
RELÓGIO
665   TICKS
↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC      PRT-DEAD  PRT-PRIO  DORMINDO  AGUARDA  SUSP
simu-18             NULO-0    alrm-17             RAIZ-19
                              ct_v-16
                              ct_t-15
                              sup-14
                              data-13
                              back-12

WAIT NO SEMAFORO act-16

PASSO A PASSO  BREAKPOINT  EXECUTE  GRAVAÇÃO
```

FIG. 4.6 d - Processo simu retoma a CPU.

Se um arquivo de eventos da execução do aplicativo for gerado, como é mostrado na figura 4.2, pode-se fazer um gráfico para análise de comportamento. As figuras 4.7 e 4.8 mostram respectivamente: o formato de um arquivo gerado pelo utilitário **relmosi** e um gráfico elaborado a partir dele.

```
                    RELATORIO DE EXECUCAO

                TICK  EVENTO
...

data running
                2     Wait no semaforo stru-19
simu running
                2     Resume processo simu-18
                4     Wait no semaforo stru-19
alrm running
                4     Resume processo alrm-17
                6     Temporizacao do processo alrm-17
ct_v running
                6     Resume processo ct_v-16
                8     Temporizacao do processo ct_v-16
ct_t running
                8     Resume processo ct_t-15
                10    Temporizacao do processo ct_t-15
sup running
                10    Resume processo sup-14
                12    Temporizacao do processo sup-14
data running
                12    Resume processo back-12
                12    Temporizacao do processo data-13
back running
                12    Wait no semaforo tela-18
                12    Signal no semaforo tela-18
                12    Wait no semaforo tela-18
                13    Signal no semaforo tela-18
                13    Temporizacao do processo back-12
...

data running
                664   Wait no semaforo tela-18
                665   Signal no semaforo tela-18
                665   Signal no semaforo stru-19
```

```
simu running
            665  Wait no semaforo act-16
            665  Wait no semaforo tela-18
            665  Signal no semaforo tela-18
            669  Resume processo ct_t-15
            670  Resume processo ct_v-16
            672  Resume processo sup-14
            686  Wait no semaforo tela-18
            686  Signal no semaforo tela-18
            686  Signal no semaforo act-16
            686  Signal no semaforo stru-19
            686  Temporizacao do processo simu-18
```

FIG. 4.7 - Arquivo de eventos - inversão de prioridade.

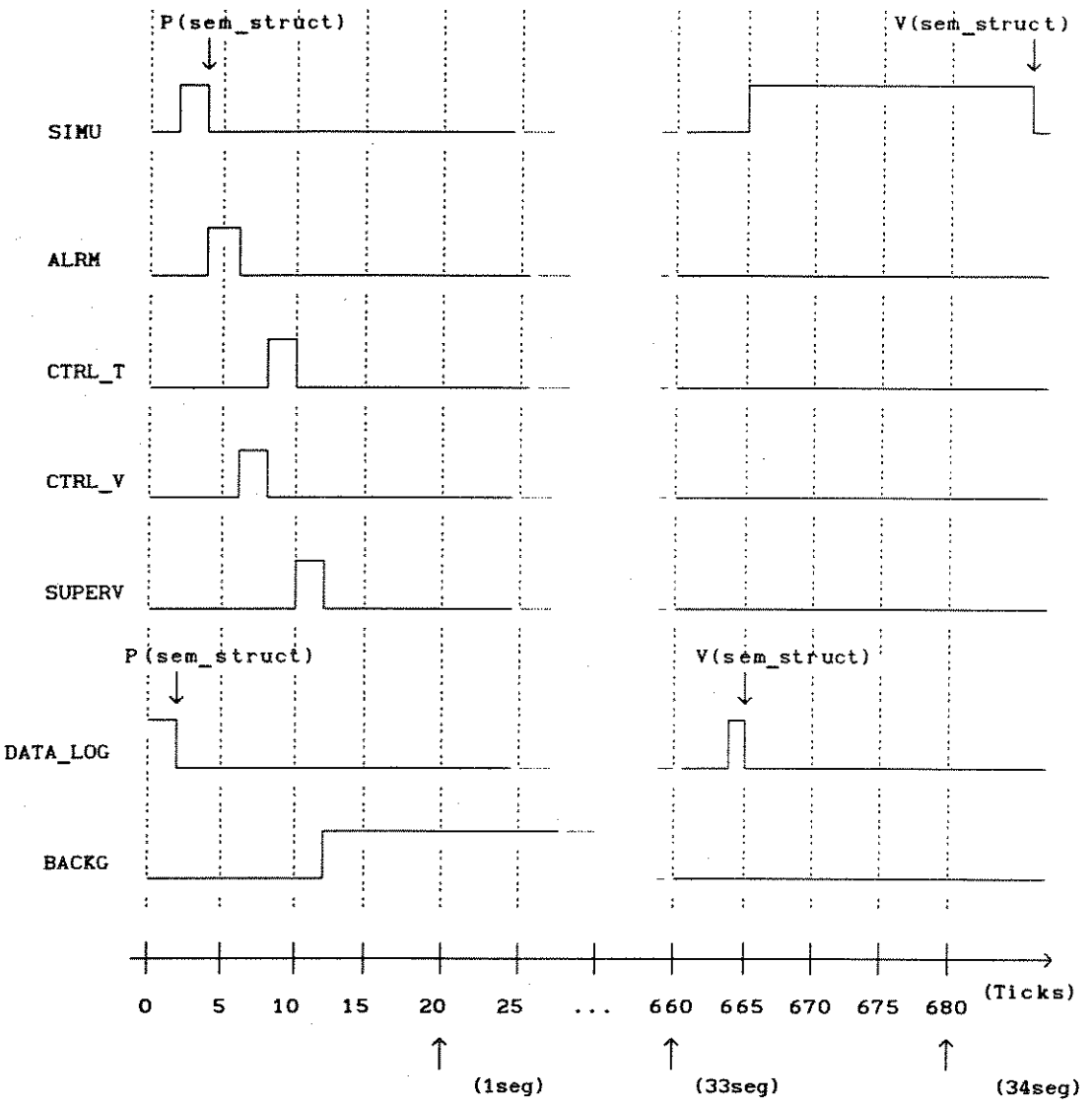


FIG. 4.8 - Comportamento do sistema.

4.2.2 Limite de prioridade.

Segundo [AZE91c] a idéia básica do protocolo limite de prioridade é garantir que, quando uma tarefa Ti realiza preempção sobre a região crítica de outra tarefa e executa sua própria região crítica *zi*, a prioridade na qual a região crítica *zi* será executada é garantidamente maior que o limite de prioridade de todas as regiões críticas que sofreram preempção. Se esta condição não puder ser satisfeita a tarefa Ti é suspensa, e a tarefa que inibiu sua execução herda sua prioridade.

Assim, uma tarefa Ti somente pode iniciar a execução de uma região crítica se sua prioridade for maior que todos os limites de prioridade dos semáforos bloqueados por outras tarefas.

Mantido o mesmo cenário do exemplo anterior e trocando-se somente o escalonador por um que implementa o protocolo limite de prioridade, podemos observar através do depurador a execução do aplicativo sob teste como é ilustrado a partir da figura 4.9.

Podemos observar que o processo **DATA_LOG** (data-13), de menor prioridade, é ativado e entra na região crítica controlada pelo semáforo **sem_tela**. Seu objetivo é imprimir na tela o valor atual das variáveis de nível e temperatura.

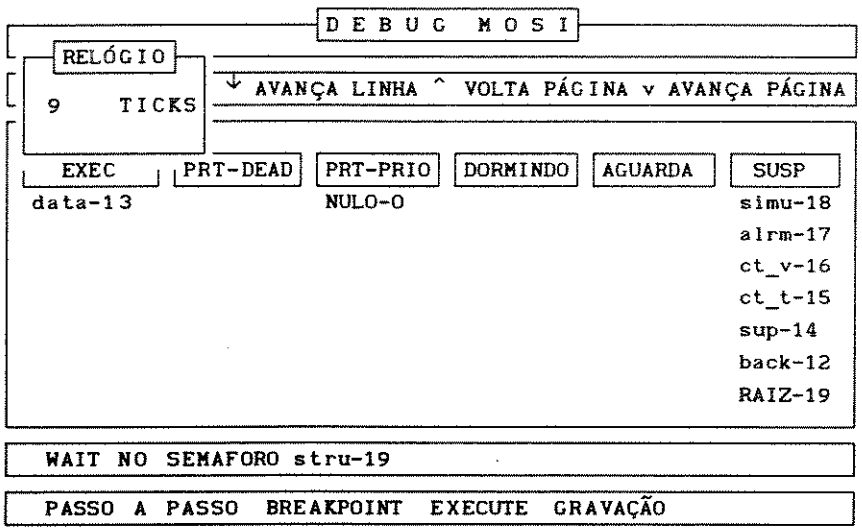


FIG. 4.9 - Processo data acessa RC.

Neste instante o processo **SIMU** (simu-18), o de mais alta prioridade do sistema, é escalonado e tenta acessar a mesma região crítica para poder atualizar os valores das mesmas variáveis.

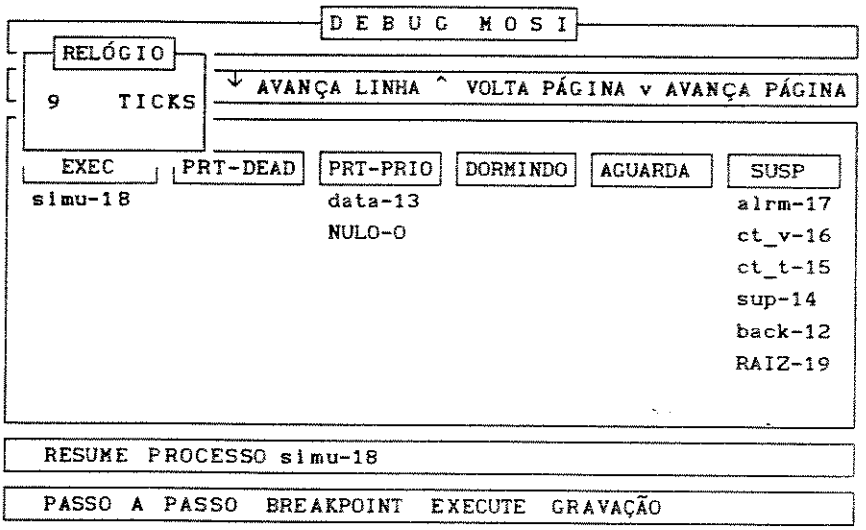


FIG. 4.9 a - Processo simu tenta acessar RC ocupada.

Como consequência o processo **SIMU** (simu-18) é bloqueado. O processo **DATA_LOG** (data-13), por sua vez, herda a prioridade de **SIMU** (simu-18) e continua sua execução.

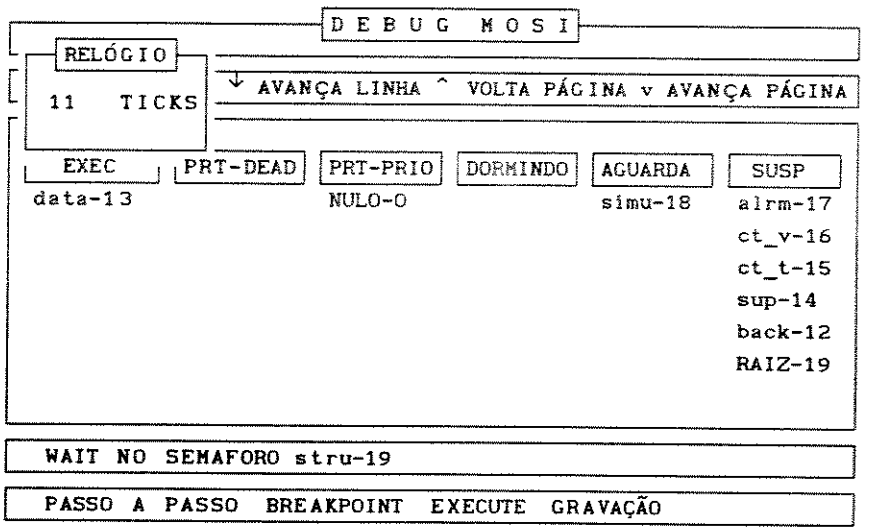


FIG. 4.9 b - Processo simu na fila de sem_struct.

Processo **DATA_LOG** (data-13) realiza um **signal** no semáforo **sem_struct**.

```
D E B U G   M O S I
RELÓGIO
71  TICKS
↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD   PRT-PRIO   DORMINDO   AGUARDA    SUSP
data-13                NULO-0     back-12    simu-18    RAIZ-19
                                              alrm-17
                                              ct_v-16
                                              ct_t-15
                                              sup-14

SIGNAL NO SEMAFORO stru-19

PASSO A PASSO  BREAKPOINT  EXECUTE  GRAVAÇÃO
```

FIG. 4.9 c - Processo data realiza signal em sem_struct.

O processo **DATA_LOG** (data-13) é preemptado pelo processo **SIMU** (simu-18) que consegue terminar sua execução. Com um atraso muito menor em relação ao caso anterior.

```
D E B U G   M O S I
RELÓGIO
119  TICKS
↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD   PRT-PRIO   DORMINDO   AGUARDA    SUSP
simu-18                NULO-0     alrm-17                RAIZ-19
                                  ct_v-16
                                  ct_t-15
                                  sup-14
                                  data-13
                                  back-12

WAIT NO SEMAFORO act-16

PASSO A PASSO  BREAKPOINT  EXECUTE  GRAVAÇÃO
```

FIG. 4.9 d - Processo simu retoma a CPU.

Na execução da aplicação com o depurador fica flagrante que o problema da inversão de prioridade foi solucionado pelo utilização do novo escalonador, (o número de "ticks" que o processo **SIMU** é penalizado é bem menor).

As figuras 4.10 e 4.11 mostram respectivamente o arquivo de eventos obtido via **relmosi** e um diagrama do comportamento do aplicativo.

```
                    RELATORIO DE EXECUCAO

              TICK  EVENTO

...

data running
               9    Wait no semaforo stru-19
simu running
               9    Resume processo simu-18
               11   Wait no semaforo stru-19
data running
               11   Resume processo alrm-17
               11   Resume processo ct_v-16
               11   Resume processo ct_t-15
               11   Resume processo sup-14
               11   Resume processo back-12
               11   Temporizacao do processo data-13
alrm running
               13   Temporizacao do processo alrm-17
ct_t running
               15   Temporizacao do processo ct_t-15
ct_v running
               17   Temporizacao do processo ct_v-16
sup running
               19   Temporizacao do processo sup-14
back running
               20   Temporizacao do processo back-12

...

data running
               71   Resume processo data-13
               71   Wait no semaforo tela-18
               71   Signal no semaforo tela-18
               80   Resume processo back-12
               119  Wait no semaforo tela-18
               119  Signal no semaforo tela-18
               119  Signal no semaforo stru-19
```

```
simu running
            119  Wait no semaforo stru-19
            119  Wait no semaforo act-16
            119  Wait no semaforo tela-18
            120  Signal no semaforo tela-18
            139  Wait no semaforo tela-18
            140  Signal no semaforo tela-18
            140  Signal no semaforo act-16
            140  Signal no semaforo stru-19
            140  Temporizacao do processo simu-18
```

FIG. 4.10 - Arquivo de eventos - limite de prioridade.

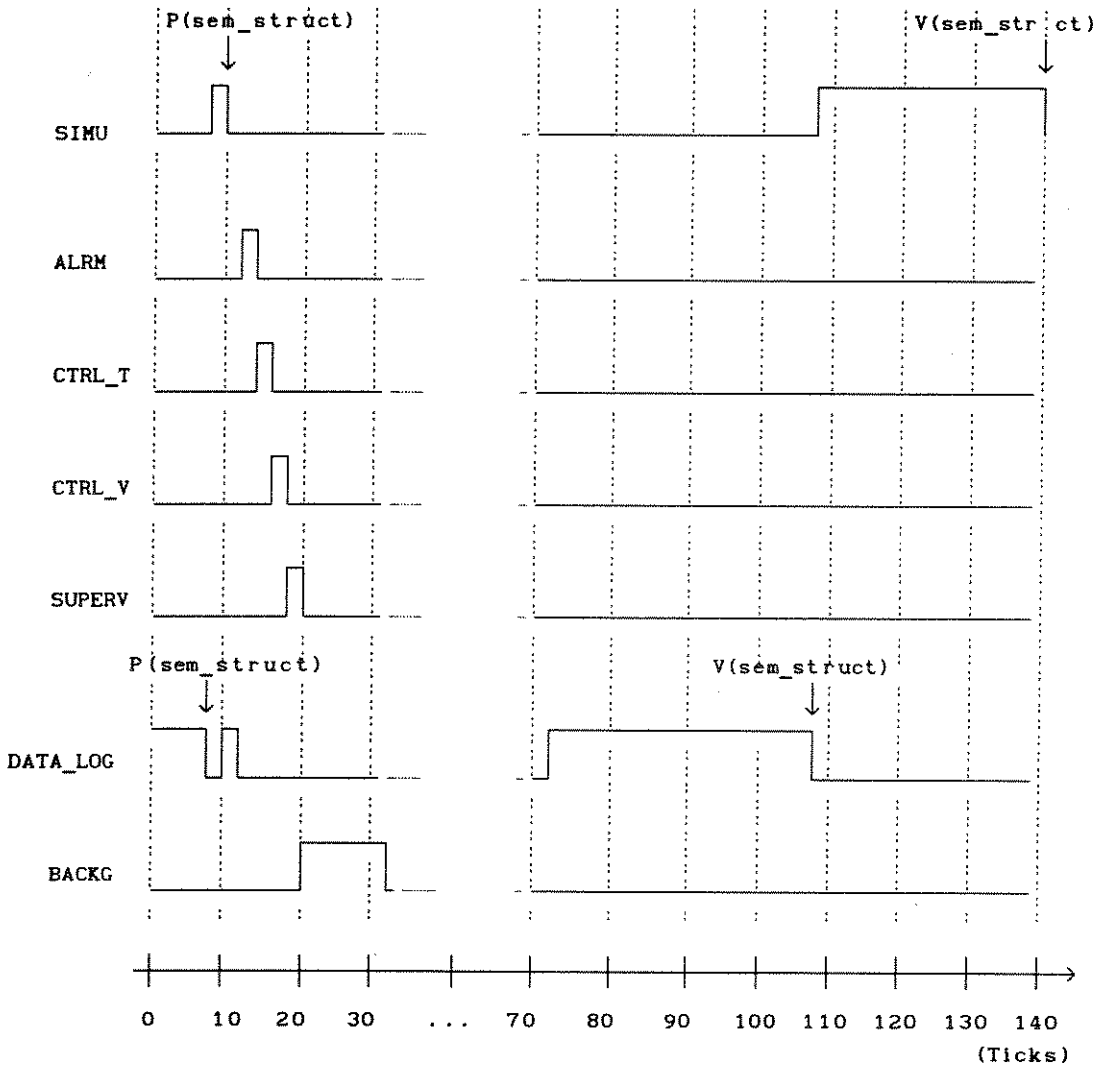


FIG. 4.11 - Comportamento do sistema.

4.2.3 Sub e super estimativa dos tempos de computação.

Os algoritmos de escalonamento são normalmente baseados no pior tempo de execução de cada processo (PTE) [HAB90b]. Em virtude de "loops" e desvios condicionais em cada programa e dos atrasos causados pelo compartilhamento de recursos durante a execução, este PTE é usualmente dificil de determinar e quando estimado pode ser de uma magnitude muito maior ou menor do que a do tempo de execução real. Este fato acaba fazendo com que os processos não consigam cumprir suas restrições de tempo (estouro de "deadline") no caso de uma estimativa a menor ou, numa subutilização de recursos no outro caso. O superdimensionamento do tempo de computação pode até mesmo fazer com que um conjunto de tarefas seja considerado não escalonável a priori, quando na realidade o seria.

Neste terceiro exemplo a aplicação foi desenvolvida sobre uma biblioteca tipo **.hrd**, ou seja, será utilizado o artifício do "replay". A sessão de depuração se dará, conforme o ilustrado na figura 4.3 a e b. O escalonador utilizado será o Earliest Deadline First (EDF).

A política EDF consiste na atribuição de prioridades às tarefas, segundo o "deadline" de sua ativação corrente. Ou seja, a tarefa com prioridade mais crítica possui o "deadline" de ativação corrente mais próximo de "estourar". A CPU será ocupada pelo tarefa com "deadline" mais crítico.

Liu [Liu73] demonstrou que para um conjunto de **n** tarefas periódicas, o algoritmo EDF atende todos os "deadlines se e somente se:

$$U = \sum_{i=1}^{n} (C_i / T_i) \leq 1$$

Onde **C** é o tempo de computação e **T** o período.

Na tabela da figura 4.12 a ordem das tarefas de cima pra baixo, é de prioridade decrescente. A menos do processo **backg** que é do tipo prioridade, todos os demais são periódicos. Por simplicidade, o "deadline" de cada processo foi assumido como sendo o período do mesmo. Nesta tabela **C** é o tempo de computação, **T** é o período e **D** o "deadline". Os valores de **C** irão variar para cada um dos casos do exemplo.

Identificação	T	C	D	Tipo
simu	05	*	05	Periódica
alrm	10	*	10	''
ctrl_t	15	*	15	''
ctrl_v	16	*	16	''
superv	17	*	17	''
data_log	20	*	20	''
backg	—	—	—	Prioridade

FIG. 4.12 - CARACTERÍSCTICAS DOS PROCESSOS.

O exemplo foi dividido em duas partes para abordar os casos de diferentes estimativas do tempo de computação:

- $C_{subestimado}$

- $C_{superestimado}$

4.2.3.1 $C_{subestimado}$.

Para esta estimativa com os tempos de computação da figura 4.13, o conjunto de tarefas não é considerado escalonável, a taxa de ocupação de CPU real é de 103% enquanto a taxa calculada ficou em 29%. Portanto haverá estouro de "deadline" de pelo menos um dos processos.

	simu	alrm	ctrl_t	ctrl_v	superv	data_log	backg
C_{real}	02	16	17	21	15	15	—
C_{sub}	02	02	06	06	09	09	—

FIG. 4.13 - TEMPO DE COMPUTAÇÃO SUBESTIMADO.

O depurador vai detectar o evento de não cumprimento de "deadline" com a tela apresentada na figura 4.14. Nela é apresentado o número de "ticks" transcorridos bem como o processo que teve suas restrições de tempo violadas.

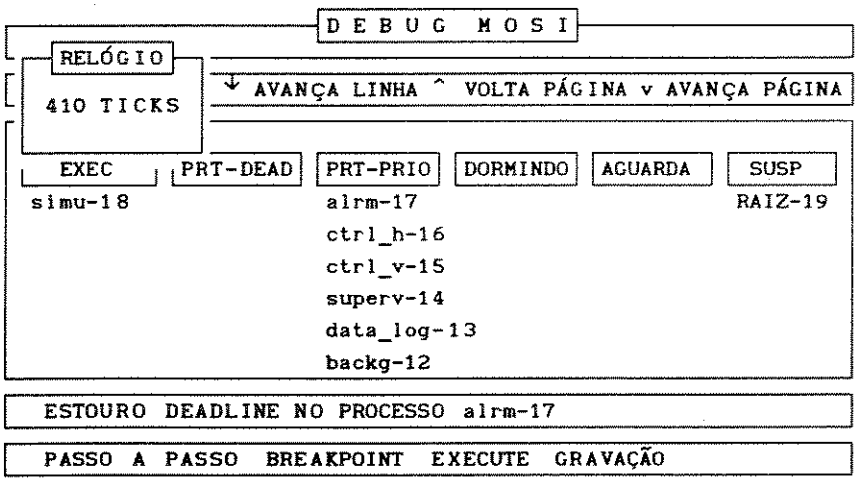


FIG. 4.14 - ESTOURO DE "DEADLINE".

Uma vez que a biblioteca **.hrd** está sendo utilizada, o próprio núcleo de tempo real gera um arquivo com os eventos ocorridos durante a execução do aplicativo, como o ilustrado na figura 4.3. As figuras 4.15 e 4.16 mostram respectivamente o arquivo de eventos obtido via **relmosi** e um diagrama do comportamento do aplicativo elaborado a partir do arquivo.

```
                    RELATORIO DE EXECUCAO

              TICK  EVENTO
...

simu running
              0     Wait no semaforo stru-19
              0     Wait no semaforo act-16
              0     Wait no semaforo tela-18
              0     Signal no semaforo tela-18
              0     Signal no semaforo act-16
              0     Wait no semaforo tela-18
              1     Signal no semaforo tela-18
              1     Signal no semaforo stru-19
              1     Temporizacao do processo simu-18
alrm running
              1     Wait no semaforo stru-19
              1     Wait no semaforo tela-18
              1     Signal no semaforo tela-18
              1     Wait no semaforo tela-18
              2     Signal no semaforo tela-18
              2     Signal no semaforo stru-19
              2     Temporizacao do processo alrm-17
ct_t running
              2     Wait no semaforo set-17
              2     Wait no semaforo act-16
              2     Wait no semaforo tela-18
              2     Signal no semaforo tela-18
simu running
              3     Resume processo simu-18
ct_t running
              3     Signal no semaforo act-16
              3     Wait no semaforo tela-18
              3     Signal no semaforo tela-18
              3     Signal no semaforo set-17
              3     Temporizacao do processo ct_t-15
```

```
ct_v running
                3    Wait no semaforo set-17
                3    Wait no semaforo act-16
                3    Wait no semaforo tela-18
                3    Signal no semaforo tela-18
                3    Signal no semaforo act-16
                3    Wait no semaforo tela-18
                4    Estouro deadline do processo RAIZ-19
                4    Signal no semaforo tela-18
                4    Signal no semaforo set-17
                4    Temporizacao do processo ct_v-16
sup running
                4    Wait no semaforo set-17
                4    Wait no semaforo tela-18
                4    Signal no semaforo tela-18
                4    Wait no semaforo tela-18
                5    Signal no semaforo tela-18
                5    Signal no semaforo set-17
                5    Temporizacao do processo sup-14
data running
                5    Wait no semaforo stru-19
                5    Wait no semaforo tela-18
                5    Signal no semaforo tela-18
                5    Wait no semaforo tela-18
simu running
                6    Resume processo simu-18
                6    Wait no semaforo stru-19
data running
                6    Signal no semaforo tela-18
                6    Signal no semaforo stru-19
simu running
                6    Wait no semaforo act-16
                6    Wait no semaforo tela-18
                7    Signal no semaforo tela-18
                7    Signal no semaforo act-16
                7    Wait no semaforo tela-18
                7    Signal no semaforo tela-18
                7    Signal no semaforo stru-19
                7    Temporizacao do processo simu-18
data running
                7    Temporizacao do processo data-13
back running
                7    Wait no semaforo tela-18
                8    Signal no semaforo tela-18
                8    Wait no semaforo tela-18
                8    Signal no semaforo tela-18
simu running
                9    Resume processo simu-18
                10   Estouro deadline do processo alrm-17
```

FIG. 4.15 - Arquivo de eventos - estimativa subestimada..

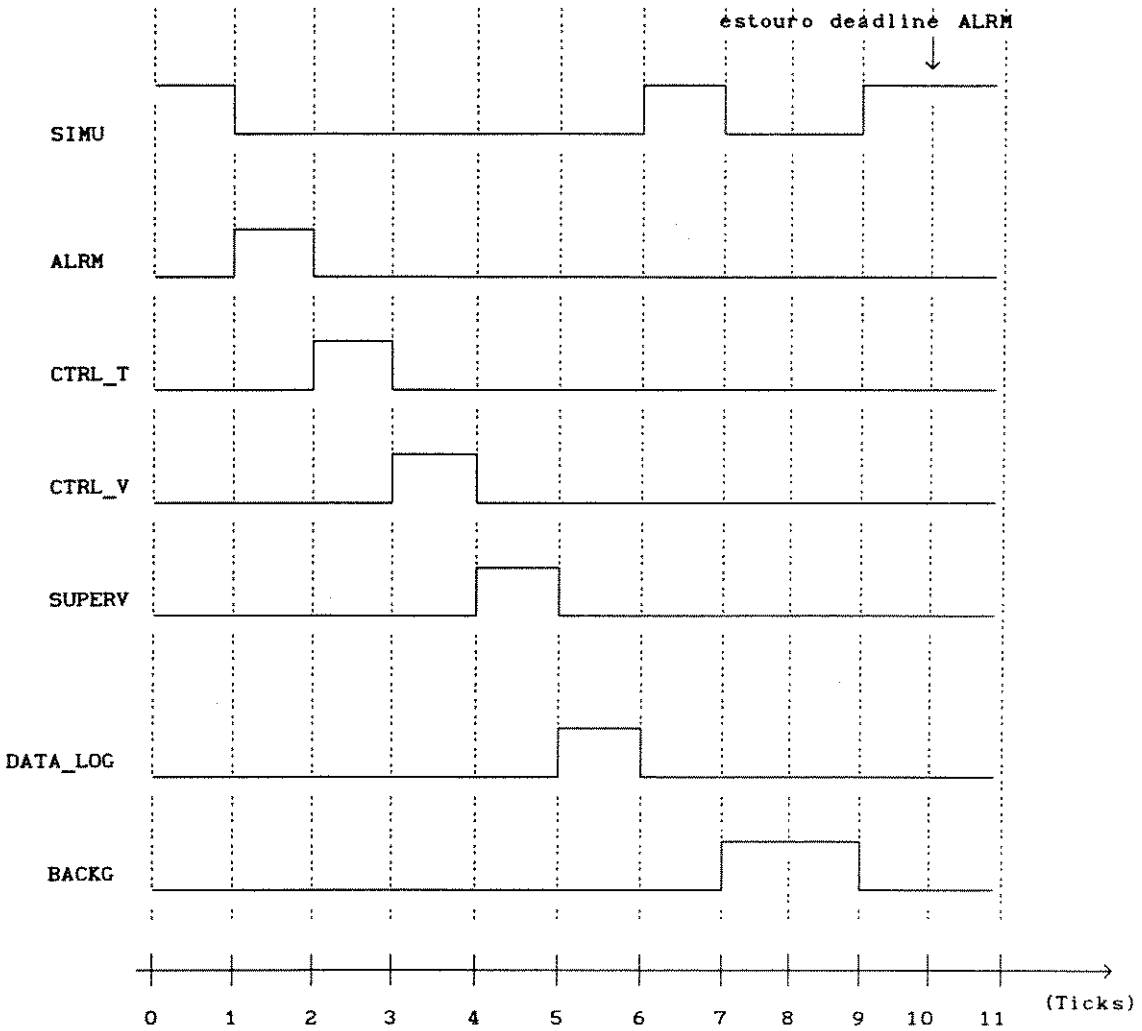


FIG. 4.16 - Comportamento do sistema.

4.2.3.2 $C_{superestimado}$.

A superestimativa dos tempos de computação dos processos pode causar uma subutilização de CPU e levar a conclusões errôneas a respeito da escalonabilidade de um sistema.

A figura 4.17 apresenta valores de um sistema subutilizado onde a taxa real de ocupação de CPU é da ordem de 26% enquanto a estimada de 29%.

	simu	alrm	ctrl_t	ctrl_v	superv	data_log	backg
C_{real}	02	02	06	06	09	09	—
C_{sup}	01	02	04	03	06	06	—

FIG. 4.17 - TEMPO DE COMPUTAÇÃO SUPERESTIMADO.

O suporte oferecido pelo depurador esta relacionada à observação dos processos presentes na fila de execução. Principalmente do processo **NULO-0** cuja "execução" sinaliza que o processador esta ocioso naquele período de tempo.

5.0 CONCLUSÃO E PERSPECTIVAS.

O trabalho apresentado tem por objetivo consolidar os conhecimentos adquiridos na área de sistemas de tempo real com restrições severas de tempo. Procura também suprir a carência de depuradores que forneçam suporte ao desenvolvimento e testes de aplicativos baseados nos núcleos de tempo real da família MOSI.

Além disso, a implementação do depurador procura contornar os principais problemas envolvidos na depuração de STR que são: o não determinismo e a perturbação causada pela presença do depurador, através do uso da opção "replay" e da utilização de máquinas que se comunicam por troca de mensagens de tamanho reduzido.

Importante é ressaltar a realimentação promovida por este trabalho, à nível dos núcleos de tempo real que dão suporte ao depurador. Uma vez que, além das alterações necessárias, para que a cada evento ocorrido, uma mensagem fosse enviada ao depurador pela linha serial, houve também mudanças que permitem que três tipos distintos de bibliotecas possam ser geradas quando da configuração desses núcleos, o que faz com que a faixa de atuação do depurador se estenda desde aplicativos que usam a biblioteca ".sft" que não possuem restrições rígidas de tempo, até aqueles sistemas onde o tempo é um fator extremamente crítico e que utilizam a biblioteca ".hrd", a qual permite o armazenamento das mensagens em memória para um posterior uso, ao invés das mesmas serem enviadas por linha serial, com o objetivo de minimizar a perturbação causada ao núcleo a nível de tempo.

Como sequência do trabalho podemos vislumbrar alguns caminhos interessantes, nos quais, com a experiência adquirida neste trabalho podemos ousar enveredar.

Um desses caminhos é uma tentativa de incrementar o depurador, ora apresentado, através de recursos adicionais de hardware. Com a utilização de uma placa colocada no barramento de um microcomputador compatível com IBM/PC. Este placa deve possuir o software do depurador armazenado em EPROM, e uma grande área para armazenamento na forma de memória RAM. Esta área de memória deve ser de pelo menos 512K. Dessa forma, como o único vínculo entre aplicativo e depurador, será o barramento. Haverá um grande acréscimo no desempenho do sistema e uma considerável minimização na interferência causada pelo processo de depuração.

A figura 5.1 ilustra o modo de implementação dessa técnica.

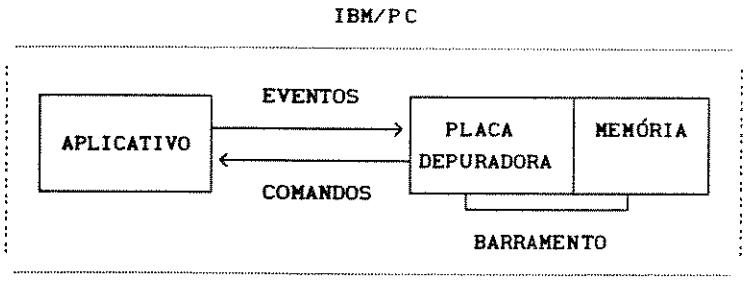


FIG. 5.1 - ESTRATÉGIA UTILIZANDO PLACA DEPURADORA.

Outro caminho passa pela utilização de técnicas de inteligência artificial. Pressupõe-se o uso de uma base de conhecimentos, que poderia ser constantemente atualizada pela "experiência" ganha em novas depurações. Essa base serviria de suporte para a criação de regras que permitissem a detecção de supostos erros e inconsistências em relação à especificação do sistema. A depuração além de acusar erros comportamentais, também deveria ser capaz de perceber erros relacionados com o fator tempo.

O gerador de relatórios **relmosi** também poderia ser incrementado de forma a gerar gráficos de comportamento do sistema depurado a partir das informações da tabela que ele monta com base no arquivo de eventos da execução do aplicativo sob teste.

Apesar das grandes dificuldades e das diversas propostas existentes a nível de depuração na área, o que se tem de pálpavel, é que com o crescente aumento de complexidade das aplicações em tempo real, a presença de depuradores cada vez mais confiáveis e menos intrusivos, será indispensável e terá um papel fundamental para o êxito desses sistemas futuros.

6. BIBLIOGRAFIA.

[APP85] Appelbe W.F., McDowell C.E., Anomaly Reporting: A tool for Debugging Parallel Numerical Algorithms, In Proc. of the 1st Int. Conf. on Supercomputing Systems, IEEE, pp. 386-391, 1985.

[AZE89a] Azevedo H., Executivo de Tempo Real QUASIMOSI - Manual do Usuário, DTIA 189, Instituto de Automação, Centro Tecnológico para Informática, Junho 1989.

[AZE89b] Azevedo H., Executivo de Tempo Real DEADMOSI - Manual do Usuário, DTIA 289, Instituto de Automação, Centro Tecnológico para Informática, Outubro 1989.

[AZE90] Azevedo H., Silva Jr. A. F., Magalhães M. Implementação de um Núcleo "Hard Real-Time" aderente à Norma MOSI da IEEE, I Simpósio de Automação Integrada do CEFET-PR, Curitiba, PR., Julho 1990.

[AZE91] Azevedo H., Tese de Mestrado Técnicas de Escalonamento para Sistemas "Hard Real-Time" Análise e Implementação, FEE UNICAMP, 1991.

[BAT88] Bates P., Distributed Debugging Tools for Heterogeneous Distributed Systems, Proc. 8th Int. Conf. Distributed Computing Systems, CS Press, Los Alamitos, Calif., pp. 308-315, 1988.

[BEN82] Ben-Ari M., Principles of Concurrent Programming, Prentice Hall International, 1982.

[CAL88] Callahan D., Subhlok J., Static Analysis of Low-Level Synchronization. In Proc. of Workshop on Parallel and Distributed Debugging, ACM, pp. 100-111, 1988.

[CHE90] Cheung W.H., Black J.P., Manning E., A Framework for Distributed Debugging, IEEE Trans. Software Eng., vol. 10, nº 6, pp. 106-115, January 1990.

[FAL88] Falk H., CASE Tools emerge to handle Real-Time Systems, Computer Design, January 1988.

[GAI85] Gait J., A Debugger for Concurrent Programs, Softw. Pract. Exper. vol. 15, nº 6, pp. 539-554, 1985.

[GRI87] Griffin J., Parallel Debugging System User's Guide. Tech Rep., Los Alamos National Laboratory, 1987.

[GUP84] Gupta N.K., Seviora R.E., An Expert System Approach to Real-Time System Debugging, Proc. 1st Conf. Artificial-Intelligence Applications, CS Press, Los Alamitos, Calif., pp. 336-343, 1984.

[HAB90a] Haban D., Wybranietz D., A Hybrid Monitor for Behavior and Performance Analysis of Distributed Systems, IEEE Trans. Software Eng., Vol. 16, nº 2, pp. 197-211, February 1990.

[HAB90b] Haban D., Shin K. G., A Application of Real_Time Monitoring to Scheduling Tasks with Random Execution Times, IEEE Trans. Software Eng., Vol. 16, nº 12, pp. 1374-1389, December 1990.

[HAR85] Harter P.K.Jr., Heimbigner D.M., King R., IDD: An Interactive Distributed Debugger, In Proc. of the 5th Int. Conf. on Distributed Computing Systems, IEEE, pp. 498-506, 1985.

[HOU87] Hough A.A., Cuny J., Belvedere: Prototype of a Pattern-Oriented Debugger for Highly Parallel Computation, In Proc. of the Int. Conf. on Parallel Processing, Penn State University, pp. 735-738, 1987.

[IEE87] IEEE Project 855. Draft Standard for Microprocessor Operating System Interfaces, The IEEE Inc., Draft 7, 1987.

[JOY87] Joyce J., Lomow G., Slind K., Unger B., Monitoring Distributed Systems, ACM Trans. Comput. Syst., vol. 5, nº 2, pp. 121-150, May 1987.

[LAZ86] Lazzerini B., Prete C., Disdeb: An Interactive High-Level Debugging System for a Multi-Microprocessor System, Microprocess. Microprogram. nº 18, pp. 401-408, 1986.

[LEB85] LeBlanc T.J., Robbins A.D., Event-Driven monitoring of Distributed Programs, In Proc. of the 5th International Conf. on Distributed Comp. Systems, IEEE, pp. 515-522, 1985.

[LEB87] LeBlanc T.J., Mellor-Crummey J.M., Debugging Parallel Programs with Instant Replay, IEEE Trans. Comput., vol. 36, nº 4, pp. 471-482, April 1987.

[LIU73] Liu C. L., Layland J. W. Scheduling Algorithms for Multiprograming in a Hard Real-Time Environment, Journal of the Association for Computing Machinery, 1(20):46-61, Jan, 1973.

[MAG86] Magalhães M. F. Software para tempo real, Editora da Unicamp, 1986.

[MAN87] Manning C.R., Traveler: The Apiary Observatory. In Proc. of European Conf. on Object Oriented Programming, pp. 97-105, 1987.

[MCD89] McDowell C.E., Helmbold D.P., Debugging Concurrent Programs, ACM Computing Surveys., vol. 21, nº 4, December 1989.

[PET77] Peterson J.L., Petri Nets, ACM Comput. Surv., vol 9, nº 3, pp. 223-252, September 1977.

[PLA84] Plattner B., Real-Time Execution Monitorig, IEEE Trans. Software Eng., vol. 10, nº 6, pp. 756-764, November 1984.

[SOC89] Socha D., Bailey M.L., Notkin D., Voyer: Graphical Views of Parallel Programs, In Proc. of Workshop on Parallel and Distributed Debugging, ACM Published as SIGPLAN Notices vol. 24, nº 1, pp. 206-215, January 1989.

[SNO84] Snodgrass R., Monitoring in a Software Development Environment: A Relational Approach. In Proc. of the Software Eng. Simp. on Practical Software Development Environments. SIGPLAN, ACM SIGSOFT, 1984.

[STE87] Stevens A., Turbo C - Memory Resident Utilities, Screen I/O and Programming Techiniques, 1987.

[TAY80] Taylor R.N., Osterweil L., Anomaly Detection in Concurrent Software by Static Data Flow Analysis, IEEE Trans. Softw. Eng., vol. 6, nº 3, pp. 265-278, May 1980.

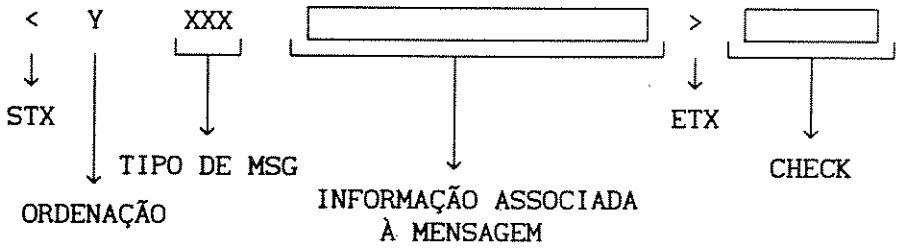
[TAY84] Taylor R.N., Debugging Real-Time Software in a Host-Target Environment, Tech. Rep. 212, Univ. of California at Irvine, 1984.

[TSA90] Tsai J.J.P., Fang K.Y., Chen H.Y., Bi Y.D., A Noninvasive Monitoring and Replay Mechanism for Real-Time Software Testing and Debugging, IEEE Trans. Software Eng., vol. 16, nº 8, pp. 897-916, August 1990.

ANEXO A
MENSAGENS TROCADAS ENTRE NÚCLEO E DEPURADOR.

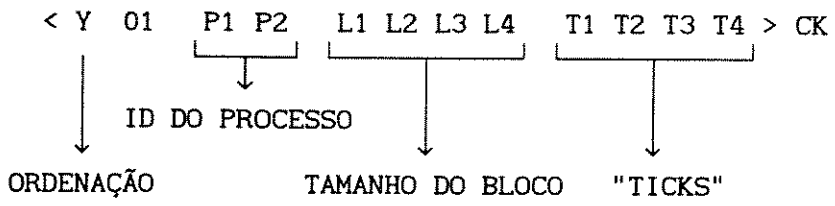
Tipos de mensagens trocadas entre DEBUGMOSI e Núcleo.

A mensagem possui o seguinte formato, utilizando números em hexadecimal.

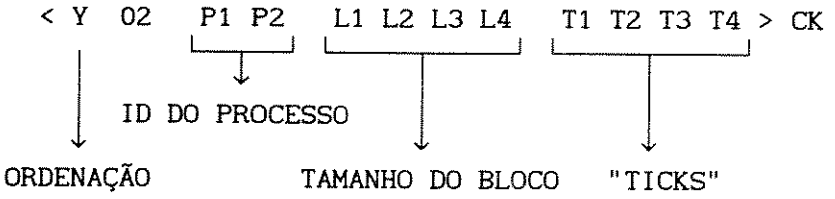


MENSAGENS RELACIONADAS COM A MANIPULAÇÃO DE MEMÓRIA

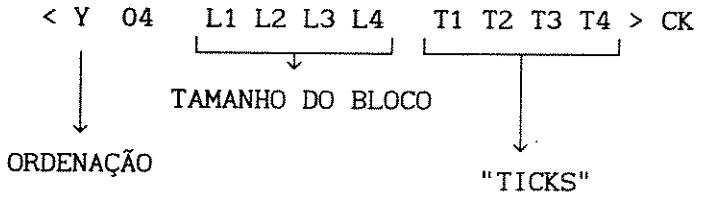
- MSG 01 "ALLOCATE MEMORY".



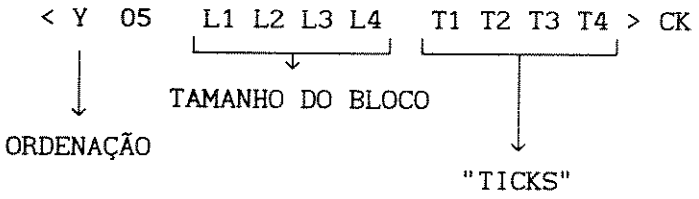
- MSG 02 "FREE MEMORY".



- MSG 04 "ALLOCATE MEMORY FROM KERNEL".



- MSG 05 "FREE MEMORY FROM KERNEL".



- MSG 03 "GET AMOUNT REMAINING".

MENSAGENS RELACIONADAS COM A TEMPORIZAÇÃO

- MSG 10 "SIMPLE DELAY"

< Y 10 P1 P2 D1 D2 D3 D4 T1 T2 T3 T4 N1 N2 N3 N4 > CK

P1 P2 → ID DO PROCESSO

D1 D2 D3 D4 → DELAY EM TICKS DO SISTEMA

T1 T2 T3 T4 → "TICKS"

N1 N2 N3 N4 → NOME DO PROCESSO

Y → ORDENAÇÃO

Obs: Uma mensagem "Resume" é gerada ao término da temporização.

MENSAGENS RELACIONADAS COM A MANIPULAÇÃO DE PROCESSOS

- MSG 20 "INITIALIZE PROGRAM".

```
< Y  20   P1 P2   PM1 PM2   M1 M2 M3 M4   T1   C1 C2   CH1 CH2  ...
          ID DO PROCESSO    TAMANHO DO PARÂMETRO   MEMÓRIA   TIPO   TEMPO COMPUTAÇÃO   CHAVE DE INSERÇÃO
ORDENAÇÃO

T1 T2 T3 T4   N1 N2 N3 N4 > CK
"TICKS"       NOME DO PROCESSO
```

Obs: Uma mensagem de Alocação de Memória é gerada.

- MSG 21 "TERMINATE PROGRAM".

```
< Y  21   P1 P2   S1 S2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          ID DO PROCESSO   ESTADO TERMINAÇÃO   "TICKS"   NOME DO PROCESSO
ORDENAÇÃO
```

- MSG 22 "CREATE PROCESS".

```
< Y 22  P1 P2  N1 N2 N3 N4  R1 R2  A  M1 M2 M3 M4  T F1 F2  C1 C2  I1 I2
        ID DO PROCESO
                NOME DO PROCESSO
ORDENAÇÃO                TAMANHO DO PARAM.
                                 "FLAG" DE ATIVAÇÃO
                                                 MEMÓRIA
                                                          TIPO
                                                             PERÍODO
                                                                  TEMPO COMPUTAÇÃO
                                                                        CHAVE INS.

T1 T2 T3 T4  N1 N2 N3 N4 > CK
"TICKS"      NOME DO PROCESSO
```

- MSG 23 "DESTROY PROCESS".

```
< Y  23  P1 P2  T1 T2 T3 T4  N1 N2 N3 N4 > CK
         ID DO PROCESSO
ORDENAÇÃO                "TICKS"  NOME DO PROCESSO
```

Obs: Uma mensagem de Liberação de Memória é gerada.

- MSG 24 "GET PROCESS STATUS".

- MSG 25 "GET PROCESS INFO".

- MSG 26 "CHANGE PRIORITY OR CHANGE TYPE".

```
< Y  26   P1 P2   T1    F1 F2   C1 C2   CH1 CH2   T1 T2 T3 T4  ...
          ID DO PROCESSO                CHAVE INSERÇÃO
ORDENAÇÃO         TIPO     TEMPO COMPUTAÇÃO                "TICKS"
                         PERÍODO

N1 N2 N3 N4 > CK
NOME DO PROCESSO
```

- MSG 27 "SUSPEND".

```
< Y  27   P1 P2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          ID DO PROCESÃO
ORDENAÇÃO         "TICKS"     NOME DO PROCESSO
```

- MSG 28 "RESUME".

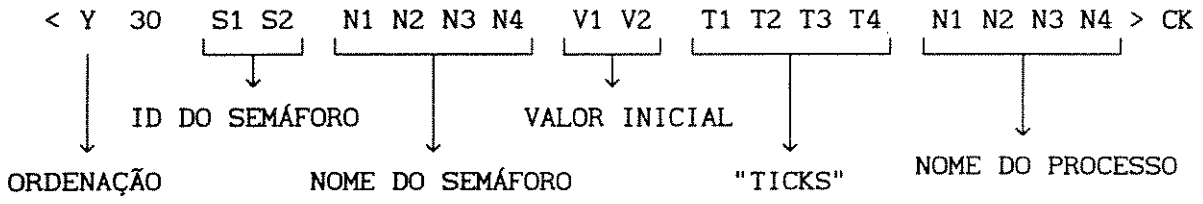
```
< Y  28   P1 P2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          ID DO PROCESÃO
ORDENAÇÃO         "TICKS"     NOME DO PROCESSO
```

- MSG 29 "PERIODIC TASK".

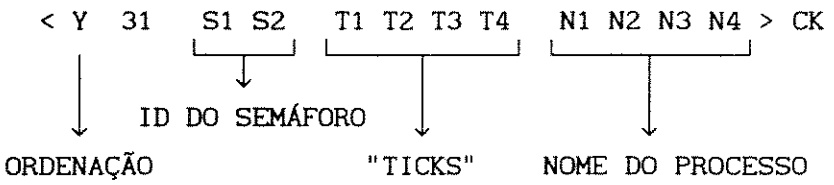
```
< Y  29   P1 P2   T1 T2 T3 T4   C1 C2 C3 C4   T1 T2 T3 T4  ...
          ↓                                   ↓
          ID DO PROCESSO        "TICKS"       "TICKS"
                                RELÓGIO
ORDENAÇÃO

N1 N2 N3 N4 > CK
↓
NOME DO PROCESSO
```

MENSAGEM RELACIONADA COM A MANIPULAÇÃO DE SEMÁFOROS

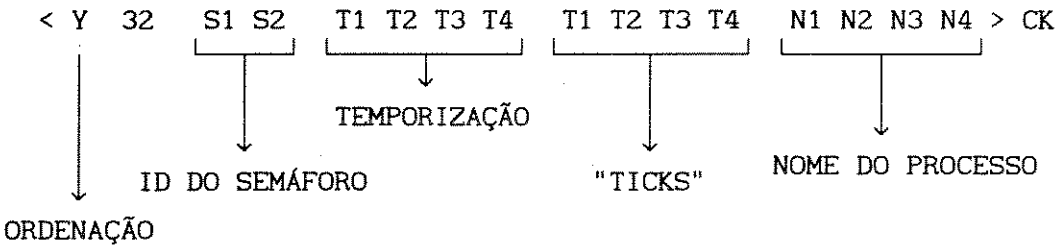
- MSG 30 "ALLOCATE SEMAPHORE".



- MSG 31 "SIGNAL SEMAPHORE".

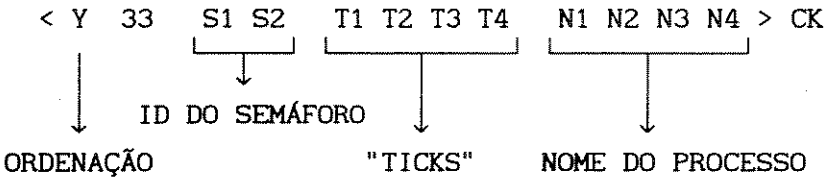


- MSG 32 "WAIT SEMAPHORE".



Obs: Caso a temporização seja diferente de zero, uma mensagem "Resume" será gerada ao fim da temporização.

- MSG 33 "DEALLOCATE SEMAPHORE".



Obs: Esta mensagem só será enviada caso a liberação seja concluida.

- MSG 34 "MULTIPLE SIGNAL SEMAPHORE".

```
< Y  34   S1 S2   C1 C2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
                 ID DO SEMÁFORO          "TICKS"
ORDENAÇÃO        CONTADOR                NOME DO PROCESSO
```

- MSG 35 "MULTIPLE WAIT SEMAPHORE".

```
< Y  35   S1 S2   C1 C2   T1 T2 T3 T4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
                 ID DO SEMÁFORO          TEMPORIZAÇÃO   "TICKS"
ORDENAÇÃO        CONTADOR                                NOME DO PROCESSO
```

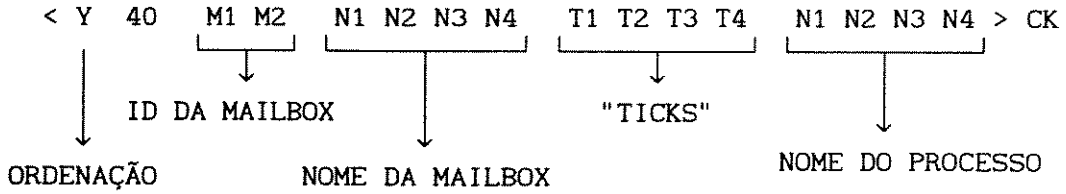
Obs: Caso a temporização seja diferente de zero, uma mensagem "Resume" será gerada ao fim da temporização.

- MSG 36 "RELEASE FROM SEMAPHORE".

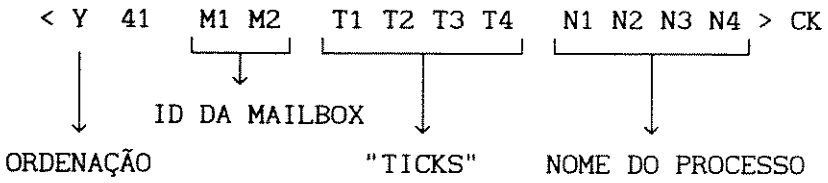
```
< Y  36   P1 P2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          ID DO PROCESSO
ORDENAÇÃO                "TICKS"   NOME DO PROCESSO
```

MENSAGEM RELACIONADA COM A MANIPULAÇÃO DE "MAILBOX"

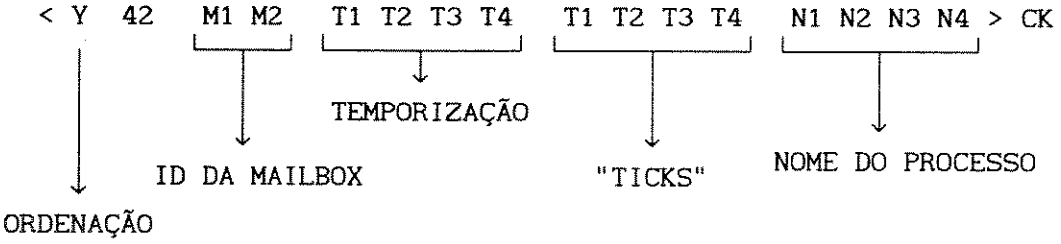
- MSG 40 "ALLOCATE MAILBOX".



- MSG 41 "SEND MESSAGE".

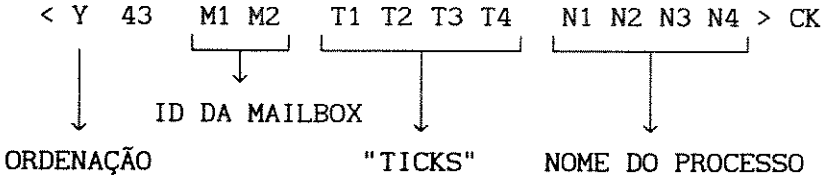


- MSG 42 "RECEIVE MESSAGE".



Obs: Uma mensagem "Resume" será gerada ao fim da temporização.

- MSG 43 "DEALLOCATE MAILBOX".



Obs: Esta mensagem só será enviada caso a liberação seja concluida.

- MSG 44 "RESPOND MESSAGE".

```
< Y  44   P1 P2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          \__/    \_________/   \_________/
            |        |            |
          ID DO PROCESSO
  |                  |            |
ORDENAÇÃO          "TICKS"   NOME DO PROCESSO
```

Obs: Esta mensagem só será enviada caso chegue resposta do processo P1 P2.

- MSG 45 "WAIT RESPONSE".

```
< Y  45   T1 T2 T3 T4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          \_________/   \_________/   \_________/
              |             |             |
          TEMPORIZAÇÃO
  |                         |             |
ORDENAÇÃO                "TICKS"   NOME DO PROCESSO
```

Obs: Caso a temporização seja diferente de zero, uma mensagem "Resume" será gerada ao fim da temporização.

MENSAGEM RELACIONADA COM A MANIPULAÇÃO DE "HANDLERS"

- MSG 50 "DISABLE EXCEPTION CLASS".

```
< Y  50   E1 E2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
 |         |        |            |
 |         v        v            v
 |       CLASSE
 v
ORDENAÇÃO            "TICKS"    NOME DO PROCESSO
```

- MSG 51 "ENABLE EXCEPTION CLASS".

```
< Y  51   E1 E2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
 |         |        |            |
 |         v        v            v
 |       CLASSE
 v
ORDENAÇÃO            "TICKS"    NOME DO PROCESSO
```

- MSG 52 "SET EXCEPTION HANDLER".

```
< Y  52   H1 H2 H3 H4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
 |         |            |            |
 |         v            v            v
 |       NOME DO "HANDLER"
 v
ORDENAÇÃO                "TICKS"    NOME DO PROCESSO
```

- MSG 53 "RESET DEFAULT HANDLER".

```
< Y  53   T1 T2 T3 T4   N1 N2 N3 N4 > CK
 v         v            v
ORDENAÇÃO  "TICKS"
                        v
                      NOME DO PROCESSO
```

- MSG 54 "SET EXCEPTION CLASS HANDLER".

```
< Y  54   L1 L2   E1 E2   H1 H2 H3 H4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
 |         v       |       v            |            |
 |       ID DO "HANDLER"   |            v            v
 |                         v          NOME DO "HANDLER"
 v                       CLASSE
ORDENAÇÃO                            "TICKS"    NOME DO PROCESSO
```

- MSG 55 "IDENTIFY ACTIVATED CLASS".

```
< Y  55   C1 C2   I1 I2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
  |        |        |        |            |
  |        v        |        v            |
  |      CLASSE     |      "TICKS"        |
  v                 v                     v
ORDENAÇÃO         ÍNDICE            NOME DO PROCESSO
```

MENSAGEM RELACIONADA COM A MANIPULAÇÃO DE EVENTOS

- MSG 60 "GET EVENT NAME".

- MSG 61 "DELETE EVENT".

- MSG 62 "EXPRESSION EVENT ADD".

- MSG 63 "EXPRESSION EVENT DELETE".

- MSG 64 "WAIT EVENT".

```
< Y  64   E1 E2   T1 T2 T3 T4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
                  TEMPORIZAÇÃO
          ID DO EVENTO          "TICKS"      NOME DO PROCESSO
ORDENAÇÃO
```

Obs: Caso a temporização seja diferente de zero espera infinita.

- MSG 65 "NOTE".

```
< Y  65   P1 P2   T1 T2 T3 T4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          ID DO PROCESSO        "TICKS"
ORDENAÇÃO         TEMPORIZAÇÃO              NOME DO PROCESSO
```

Obs: Esta mensagem só será enviada se houver a ativação do processo P1 P2.

MENSAGEM RELACIONADA COM A MANIPULAÇÃO DE "BREAKPOINTS"

- MSG 71 "LIBERA".

```
< Y  71  T1 T2 T3 T4   N1 N2 N3 N4 > CK
  ↓      └──────┘      └──────┘
ORDENAÇÃO   "TICKS"     NOME DO PROCESSO
```

- MSG 72 "PASSO A PASSO".

```
< Y  72  T1 T2 T3 T4   N1 N2 N3 N4 > CK
  ↓      └──────┘      └──────┘
ORDENAÇÃO   "TICKS"     NOME DO PROCESSO
```

- MSG 74 "EXECUTE".

```
< Y  74  T1 T2 T3 T4   N1 N2 N3 N4 > CK
  ↓      └──────┘      └──────┘
ORDENAÇÃO   "TICKS"     NOME DO PROCESSO
```

- MSG 75 "INSERE BREAK - PROCESSO".

```
< Y  75  P1 P1   E1 E2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
  │      └─┘      └─┘    └──────┘      └──────┘
  │      ID DO PROC. ESTADO   │         NOME DO PROCESSO
ORDENAÇÃO                   "TICKS"
```

- MSG 76 "INSERE BREAK - SEMÁFORO".

```
< Y  76  S1 S2   C1 C2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
  │      └─┘      └─┘    └──────┘      └──────┘
  │      ID DO SEM CONTADOR   │         NOME DO PROCESSO
ORDENAÇÃO                   "TICKS"
```

- MSG 77 "INSERE BREAK - MAILBOX".

```
< Y 77   M1 M2   N1 N2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
  |        |       |        |              |
  |        v       v        v              v
  |     ID DA MAIL  MSGS                NOME DO PROCESSO
  v                              "TICKS"
ORDENAÇÃO
```

- MSG 78 "INSERE BREAK - HANDLER".

```
< Y 78   C1 C2   A1 A2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
  |        |       |        |              |
  |        v       v        v              v
  |     CLASSE  ATIVAÇÕES                NOME DO PROCESSO
  v                              "TICKS"
ORDENAÇÃO
```

- MSG 85 "DELETAR BREAK - PROCESSO".

```
< Y 85   P1 P2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
  |        |        |              |
  |        v        v              v
  |   POS. TABELA PROC.        NOME DO PROCESSO
  v                  "TICKS"
ORDENAÇÃO
```

- MSG 86 "DELETA BREAK - SEMÁFORO".

```
< Y 86   P1 P2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
  |        |        |              |
  |        v        v              v
  |   POS. TABELA SEM          NOME DO PROCESSO
  v                  "TICKS"
ORDENAÇÃO
```

- MSG 87 "DELETA BREAK - MAILBOX".

```
< Y 87   P1 P2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
  |        |        |              |
  |        v        v              v
  |   POS. TABELA MAIL         NOME DO PROCESSO
  v                  "TICKS"
ORDENAÇÃO
```

- MSG 88 "DELETA BREAK - HANDLER".

```
< Y  88   P1 P2   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          \_/     \_____/       \_____/
           |       |             |
           |       |             |
  POS. TABELA HDL  |             |
  ORDENAÇÃO        "TICKS"       NOME DO PROCESSO
```

- MSG 91 "BREAK PROCESSO".

```
< Y  91   P1 P2   T1 T2 T3 T4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          \_/     \_____/       \_____/       \_____/
           |       |             |             |
           |       |             |             |
  PROC. ATUAL      |             |             NOME DO PROCESSO
  ORDENAÇÃO        TEMPORIZAÇÃO  "TICKS"
```

- MSG 92 "BREAK SEMÁFORO".

```
< Y  92   S1 S2   T1 T2 T3 T4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          \_/     \_____/       \_____/       \_____/
           |       |             |             |
           |       |             |             |
  SEM ATUAL        |             |             NOME DO PROCESSO
  ORDENAÇÃO        TEMPORIZAÇÃO  "TICKS"
```

- MSG 93 "BREAK MAILBOX".

```
< Y  93   M1 M2   T1 T2 T3 T4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          \_/     \_____/       \_____/       \_____/
           |       |             |             |
           |       |             |             |
  MAIL ATUAL       |             |             NOME DO PROCESSO
  ORDENAÇÃO        TEMPORIZAÇÃO  "TICKS"
```

- MSG 94 "BREAK HANDLER".

```
< Y  94   H1 H2   T1 T2 T3 T4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          \_/     \_____/       \_____/       \_____/
           |       |             |             |
           |       |             |             |
  CLASSE ATUAL     |             |             NOME DO PROCESSO
  ORDENAÇÃO        TEMPORIZAÇÃO  "TICKS"
```

MENSAGENS DE USO GERAL

- MSG 70 "INICIALIZAÇÃO".

```
< Y  70   M1 M2 M3 M4   T1   T1 T2 T3 T4   T1 T2 T3 T4 > CK
          MEMÓRIA TOTAL  TIPO  TEMPORIZAÇÃO  "TICKS"
ORDENAÇÃO
                         [ 0 - QUASIMOSI
                           1 - DEADMOSI
                           2 - LAXMOSI
```

- MSG 80 "ERRO DO USUÁRIO".

```
< Y  80   P1 P2   R1 R2   T1 T2 T3 T4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          ID DO PROCESSO  TEMPORIZAÇÃO  NOME DO PROCESSO
ORDENAÇÃO  CÓDIGO DO ERRO  "TICKS"
```

- MSG 90 "ERRO DO SISTEMA".

```
< Y  90   P1 P2   R1   T1 T2 T3 T4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          ID DO PROCESSO  TEMPORIZAÇÃO  NOME DO PROCESSO
ORDENAÇÃO  CÓDIGO DO ERRO  "TICKS"
```

- MSG 97 "TRATA DEADLINE EDF E LLS".

```
< Y  97   E1 E2   T1 T2 T3 T4   T1 T2 T3 T4   N1 N2 N3 N4 > CK
          TEMPORIZAÇÃO  NOME DO PROCESSO
          ID DO EVENTO  "TICKS"
ORDENAÇÃO
```

- MSG 98 "TROCA DINÂMICA DE CONTEXTO".

```
< Y  98  E1 E2  T1 T2 T3 T4  T1 T2 T3 T4  N1 N2 N3 N4 > CK
         ↓      ↓            ↓            ↓
         ID DO  TEMPORIZAÇÃO "TICKS"      NOME DO PROCESSO
         EVENTO
ORDENAÇÃO
```

- MSG 99 "ATUALIZA LLS".

```
< Y  99  T1 T2 T3 T4  N1 N2 N3 N4 > CK
↓        ↓            ↓
ORDENAÇÃO "TICKS"     NOME DO PROCESSO
```

ANEXO B
MANUAL DE UTILIZAÇÃO DO DEBUGMOSI.

D E B U G M O S I

VERSÃO 1.1

MANUAL DE UTILIZAÇÃO

CTI - CENTRO TECNOLÓGICO PARA INFORMÁTICA

D E B U G M O S I

DEPURADOR PARA NÚCLEOS DE TEMPO REAL

VERSÃO 1.1

MANUAL DE UTILIZAÇÃO

ALFREDO F. SILVA JR.
ORIENTADOR: HÉLIO AZEVEDO.

INSTITUTO DE AUTOMAÇÃO.
DIVISÃO DE PROCESSOS.
18-04-1991.

ÍNDICE

1.0 INTRODUÇÃO

1.1 OBJETIVO.....05

2.0 GENERALIDADES

2.1 DIAGRAMA ESTRUTURAL.....06

2.2 DESCRIÇÃO FUNCIONAL.....08

3.0 COMANDOS

3.1 INICIALIZAÇÃO.....10

3.2 COMANDO VISUALIZAÇÃO.....11

3.2.1 SUBCOMANDO ESTADO.....12

3.2.2 SUBCOMANDO MAILBOX.....13

3.2.3 SUBCOMANDO SEMÁFORO.....14

3.2.4 SUBCOMANDO RELÓGIO.....15

3.3 COMANDO EXECUÇÃO.....16

3.3.1 SUBCOMANDO EXECUTE.....17

3.3.2 SUBCOMANDO BREAKPOINT.....18

3.3.2.1 OPÇÃO INSERIR.....19

a. SUBOPÇÃO PROCESSO.....20

b. SUBOPÇÃO MAILBOX.....21

c. SUBOPÇÃO SEMÁFORO.....22

d. SUBOPÇÃO HANDLER.....23

3.3.2.2 OPÇÃO DELETAR.....24

a. SUBOPÇÃO PROCESSO.....25

b. SUBOPÇÃO MAILBOX.....26

c. SUBOPÇÃO SEMÁFORO.....27

d. SUBOPÇÃO HANDLER.....28

3.3.2.3 OPÇÃO LISTAR.....29

a. SUBOPÇÃO PROCESSO.....30

b. SUBOPÇÃO MAILBOX.....31

c. SUBOPÇÃO SEMÁFORO.....32

d. SUBOPÇÃO HANDLER.....33

3.3.3 SUBCOMANDO PASSO A PASSO.....34

3.3.4 SUBCOMANDO GRAVAÇÃO.....35

3.4 COMANDO SUPERVISÃO.....36

3.4.1 SUBCOMANDO PROCESSO.....37

3.4.2 SUBCOMANDO MAILBOX.....38

3.4.3 SUBCOMANDO SEMÁFORO.....39

3.4.4 SUBCOMANDO MEMÓRIA.....40

3.4.5 SUBCOMANDO HANDLER.....41

3.5 COMANDO MAIS.....42

3.6 COMANDO ESTATÍSTICA.....43

3.6.1 SUBCOMANDO PROCESSO.....44

3.6.2 SUBCOMANDO MAILBOX.....45

3.6.3 SUBCOMANDO SEMÁFORO.....46

3.6.4 SUBCOMANDO MEMÓRIA.....47

3.6.5 SUBCOMANDO SISTEMA.....48

3.7 COMANDO SIMULAÇÃO.....49

3.8 COMANDO SAÍDA.....50

4.0 CÓDIGOS DE ERRO

4.1 TABELA.....51

1.1 OBJETIVO.

DEBUGMOSI é uma ferramenta de auxílio na depuração de núcleos de tempo real. Esta ferramenta está associada aos núcleos: **QUASIMOSI**, **DEADMOSI** e **LAXMOSI**, desenvolvidos no CTI para ambientes IBM/PC. A proposta do desenvolvimento de um depurador se deve à grande dificuldade de interação entre usuário e núcleo na fase de teste de aplicativos, principalmente devido ao fator de se tratar de um ambiente de multiprogramação. É portanto fundamental a possibilidade de acompanhamento passo a passo do aplicativo, de definição de pontos de parada (breakpoints) para a observação do valor de semáforos e mailbox e análise do estado dos processo que constituem o sistema, além da facilidade de armazenamento da execução do mesmo. Estes e outros recursos existentes serão descritos de maneira mais minuciosa em ítens específicos. Em versões posteriores serão implementados comandos para estatística do sistema em geral e comandos para alteração de valores de semáforos, mailbox, estado de um processo, etc. A interface do depurador com o sistema alvo é realizada através de uma linha serial RS 232C, ou no caso de depuração de sistemas com restrições severas de tempo, através de um arquivo de simulação gerado pelo prório núcleo, sendo sua máquina hospedeira um micro compatível com IBM/PC (fig. 1.0).

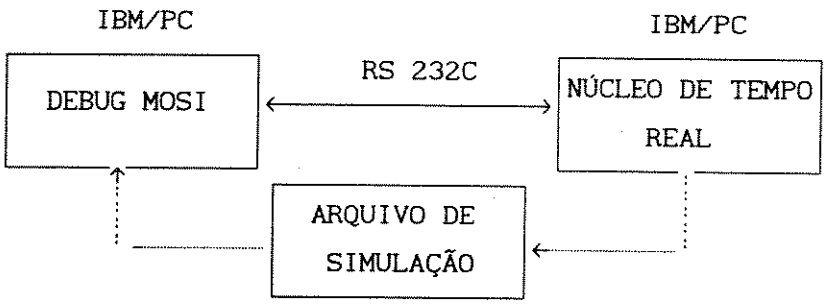
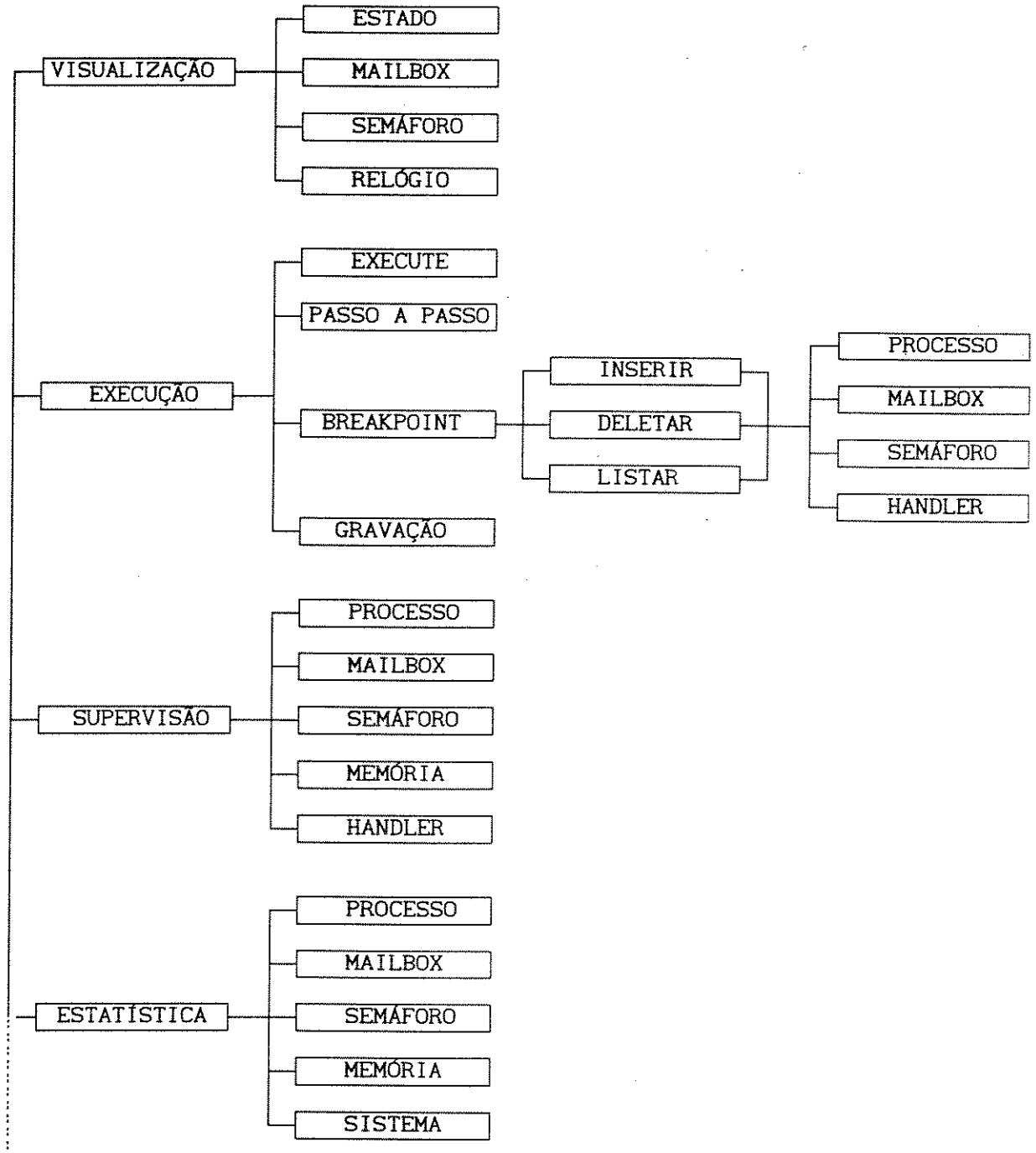
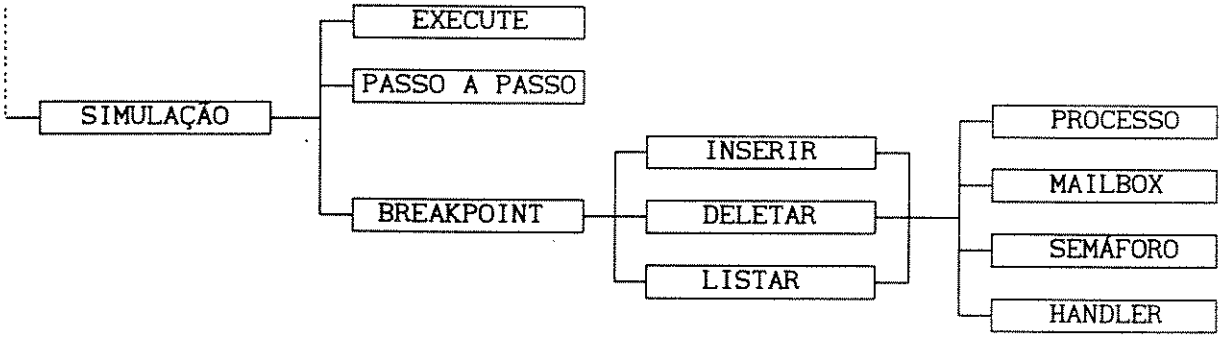


FIG. 1.0 - AMBIENTE OPERACIONAL

2.1 DIAGRAMA ESTRUTURAL.




SIMULAÇÃO
EXECUTE
PASSO A PASSO
BREAKPOINT
INSERIR
DELETAR
LISTAR
PROCESSO
MAILBOX
SEMÁFORO
HANDLER

2.2 DESCRIÇÃO FUNCIONAL.

Os recursos disponíveis são os seguintes:

- Visualização: Visualização gráfica do estado de cada processo (vide fig. 2.0), bem como observação daqueles processos que aguardam recursos (mailbox e semáforos).

ESTADO	DESCRIÇÃO
PRCURR	PROCESSO EM EXECUÇÃO
PRREADY	ESCALONADO PARA EXECUÇÃO
PRSUSP	SUSPENSO
PRWAIT	AGUARDANDO SEMÁFORO
PRWT	SEMÁFORO TEMPORIZADO
PRFREE	PROCESSO TERMINADO
PRSLEEP	PROCESSO ADORMECIDO
PRRECP	AGUARDANDO MENSAGEM
PRRECT	MENSAGEM TEMPORIZADA
PRRESP	AGUARDANDO RESPOSTA
PRREST	RESPOSTA TEMPORIZADA
PREVE	AGUARDANDO EVENTO
PREVT	EVENTO TEMPORIZADO

FIG. 2.0 - ESTADOS POSSÍVEIS DE UM PROCESSO.

- Execução: Execução normal ou passo a passo, com possibilidade de reinício do programa sob teste e inserção de breakpoints no mesmo.

- Supervisão: Supervisão de processos, mailbox, semáforos, handler e memória.

O comando de **SUPERVISÃO** usará janelas para fornecer informações dos

ítens acima citados mediante a entrada de um identificador.

- Estatística: Através de janelas fornece informações de processos, mailbox, semáforos, memória e sistema. Este comando não será implementado nesta versão.

- Simulação: Permite a reexecução do sistema monitorado através de um arquivo que foi previamente gerado, durante a execução.

Em todos os menus a tela de fundo será aquela contendo o estado atual dos processos do sistema.

Esta tela default pode ser alterada através do comando **VISUALIZAÇÃO**. Uma vez escolhida, a nova tela permanecerá até ser novamente alterada pelo comando acima.

O movimento do usuário pela árvore de comandos é realizado através de menus existentes na região inferior da tela.

A escolha de um comando implica na atualização do menu com as novas opções existentes.

O retorno ao nível anterior é realizado através da tecla **ESC**.

3.1 INICIALIZAÇÃO.

Antes de dar início à fase de depuração propriamente dita o usuário deve configurar o núcleo de tempo real utilizado, em relação ao tipo de biblioteca que o mesmo deverá gerar. A biblioteca escolhida depende do tipo da aplicação. Em caso de aplicações sem restrições severas de tempo a biblioteca escolhida deve ser a "SFT", em caso de sistemas com dependência rígida do tempo deve ser a "HRD", já no caso de não se desejar depuração a escolha deve ser a biblioteca "LIB".

Ao se ativado o programa apresentará a tela da fig. 3.0 que mostra o menu dos comandos do **DEBUG MOSI** e como tela de fundo o estado dos processos que estão presentes no núcleo. Esta tela será default para todas os comandos do depurador a não ser que nova escolha seja feita através do comando **VISUALIZAÇÃ**O. As teclas de movimentação do cursor permitem que se percorra a janela principal realizando scroll vertical segundo a opção linha ou página. O processo **NULO**, mostrado, bem como todos os outros, possuem um prefixo numérico, que tem por objetivo, diferenciar, diferentes instâncias, do mesmo processo. A seguir passaremos à descrição de cada um dos comandos presentes na tela principal.

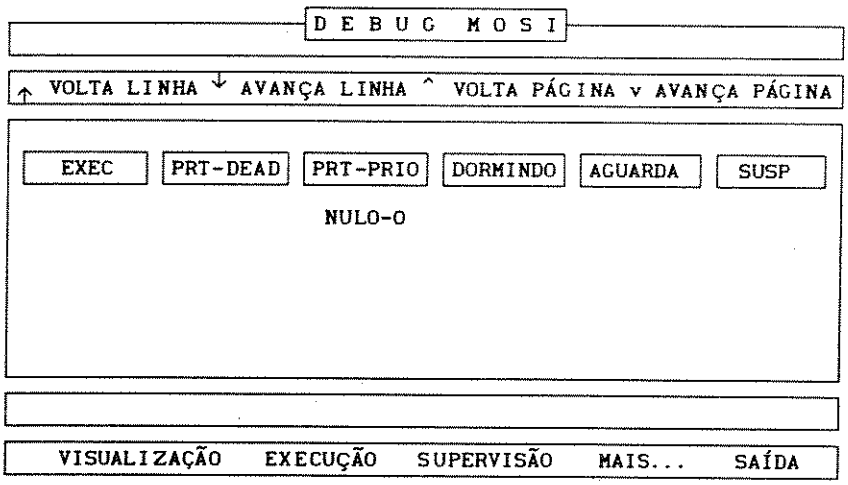


FIG. 3.0 - TELA DE ABERTURA.

3.2 COMANDO VISUALIZAÇÃO.

O comando **VISUALIZAÇÃO** determina qual será a tela de fundo apresentada durante a utilização do **DEBUG**. Esta tela é constantemente atualizada e exprime o estado corrente do sistema.

Ao se escolher o comando **VISUALIZAÇÃO** será mostrada a tela da fig. 3.1 que conterá os subcomandos **ESTADO**, **MAILBOX** e **SEMÁFORO**. Este comando permitirá a troca da tela de fundo segundo o subcomando escolhido, passando esta a ser a nova tela default.

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC      PRT-DEAD   PRT-PRIO   DORMINDO   AGUARDA   SUSP

PRC1-15   PRC3-17    NULO-0     PRC8-19              PRC4-10
          PRC5-13
          PRC7-11

VISUALIZAÇÃO

ESTADO  MAILBOX  SEMÁFORO  RELÓGIO
```

FIG. 3.1 - COMANDO VISUALIZAÇÃO

As figuras 3.2, 3.3, 3.4 e 3.5 mostram o padrão de tela para os subcomados **ESTADO**, **MAILBOX** , **SEMÁFORO** e **RELÓGIO** respectivamente.

3.2.1 SUBCOMANDO ESTADO.

```
                    D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

 EXEC      PRT-DEAD   PRT-PRIO   DORMINDO   AGUARDA   SUSP

 PRC1-15   PRC3-17    NULO-0     PRC8-19              PRC4-10
           PRC5-13
           PRC7-11

VISUALIZAÇÃO - ESTADO
```

FIG. 3.2 - SUBCOMANDO ESTADO

3.2.2 SUBCOMANDO MAILBOX.

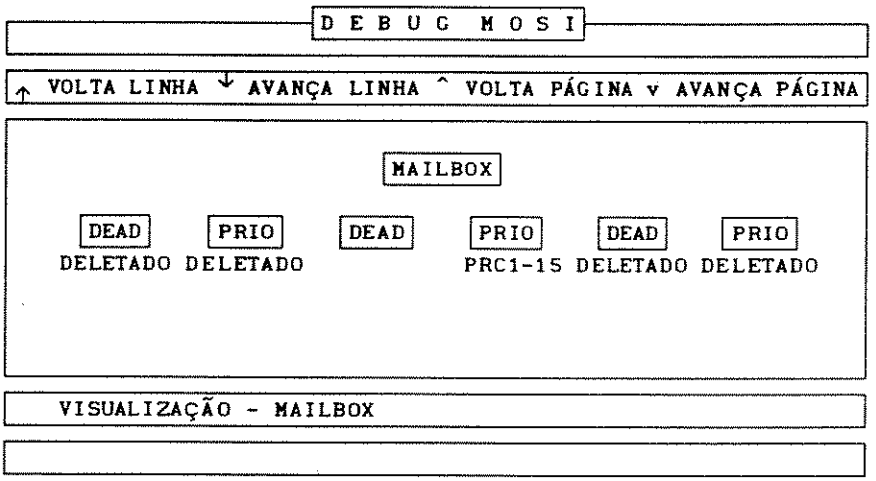


FIG. 3.3 - SUBCOMANDO MAILBOX

3.2.3 SUBCOMANDO SEMÁFORO.

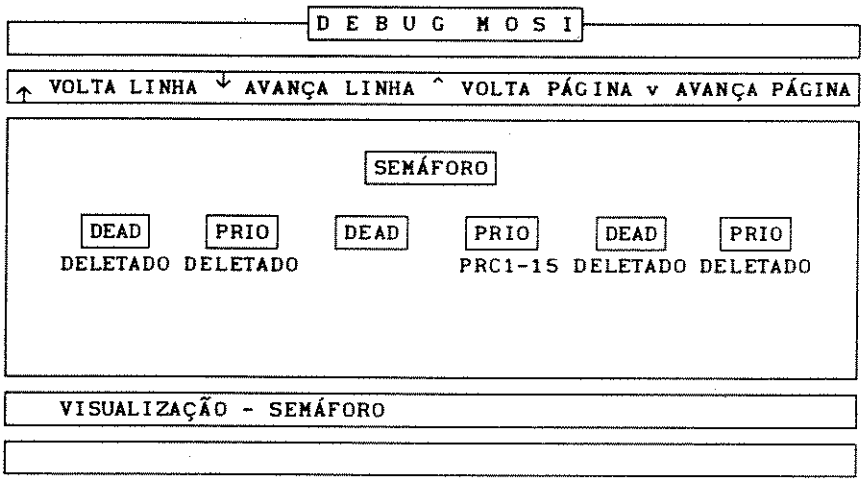


FIG. 3.4 - SUBCOMANDO SEMÁFORO

3.2.4 SUBCOMANDO RELÓGIO.

Este subcomando é do tipo "ON/OFF", permanecendo a opção escolhida a ser válida em todas as telas subsequentes.

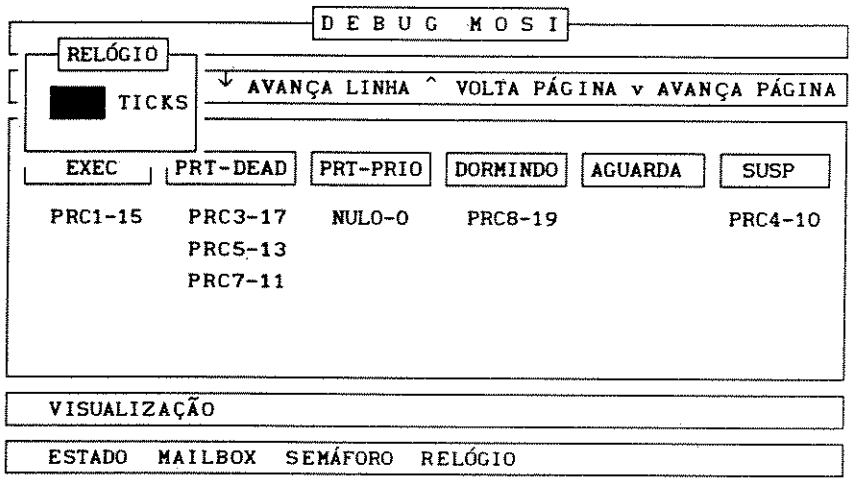


FIG. 3.5 - SUBCOMANDO RELÓGIO.

3.3 COMANDO EXECUÇÃO.

Este comando permite que o usuário mantenha um rígido controle sobre os eventos que caracterizam a execução de um sistema com multiprogramação.

Sua utilização permite basicamente a execução contínua ou passo a passo do sistema associado ao recurso de pontos de parada, além da possibilidade de gravação da execução (passo a passo ou direta), do aplicativo.

Para implementar essas funções ele possui quatro subcomandos: **EXECUTE**, **PASSO A PASSO BREAKPOINT** e **GRAVAÇÃO**. A fig. 3.6 ilustra a tela gerada na chamada desse comando. Uma descrição detalhada dos subcomandos será dada a seguir.

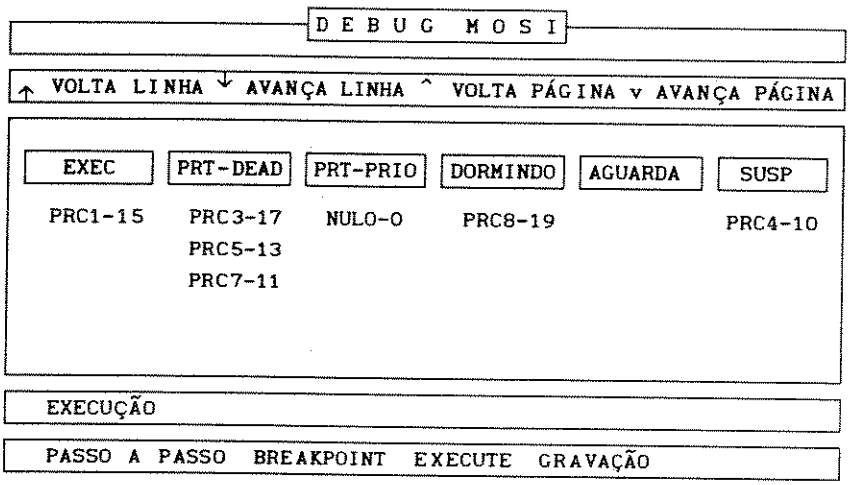


FIG. 3.6 - COMANDO EXECUÇÃO

3.3.1 SUBCOMANDO EXECUTE.

Permite a execução do aplicativo que o usuário deseje depurar. Para finalizar a opção **EXECUTE** deve-se teclar **ESC** o que causará o retorno à tela inicial de execução. Caso ocorra algum ponto de parada definido, durante a execução, o **DEBUG** retornará imediatamente ao usuário, mantendo-se neste nível de comando. A fig. 3.7 mostra a tela do subcomando **EXECUTE**.

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC      PRT-DEAD   PRT-PRIO   DORMINDO   AGUARDA   SUSP

PRC1-15   PRC3-17    NULO-0     PRC8-19              PRC4-10
          PRC5-13
          PRC7-11

EXECUÇÃO - EXECUTE

<ESC> PARA ENCERRAR EXECUÇÃO
```

FIG. 3.7 - SUBCOMANDO EXECUTE

3.3.2 SUBCOMANDO BREAKPOINT.

Este subcomando permite que o usuário defina pontos de parada, retire-os e obtenha uma listagem dos mesmos. As operações citadas serão executadas via as opções **INSERIR**, **DELETAR** e **LISTAR** estando estas vinculadas à quatro subopções: **PROCESSO**, **MAILBOX**, **SEMÁFORO** e **HANDLER**.

O subcomando **BREAKPOINT** fará consistência dos dados digitados em todas as suas opções e retornará em caso de entrada inválida uma mensagem de aviso, permitindo a digitação de uma nova entrada.

A fig. 3.8 mostra a tela gerada pelo subcomando. Nos próximos ítens serão descritas em detalhes cada uma dessas opções.

```
                    D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

 EXEC      PRT-DEAD   PRT-PRIO   DORMINDO   AGUARDA   SUSP

 PRC1-15   PRC3-17    NULO-0     PRC8-19              PRC4-10
           PRC5-13
           PRC7-11

 EXECUÇÃO - BREAKPOINT

 INSERIR  DELETAR  LISTAR
```

FIG. 3.8 - SUBCOMANDO BREAKPOINT

3.3.2.1 OPÇÃO INSERIR.

Esta opção possibilita a inserção de breakpoints, ficando os mesmos vinculados a quatro eventos:

- mudanças no estado de um determinado processo,
- contador de um semáforo atinge um valor escolhido,
- número de mensagens em uma mailbox é atingido,
- número de ativações de um tratador de interrupção (handler), é alcançado.

O número **máximo** de breakpoints permitido pelo depurador, para cada uma das subopções é **quatro**. Caso este limite seja excedido, será gerada uma mensagem de advertência.

A tela gerada pela opção **INSERIR** aparece na fig. 3.9. Serão detalhados na sequência, cada uma das subopções citadas.

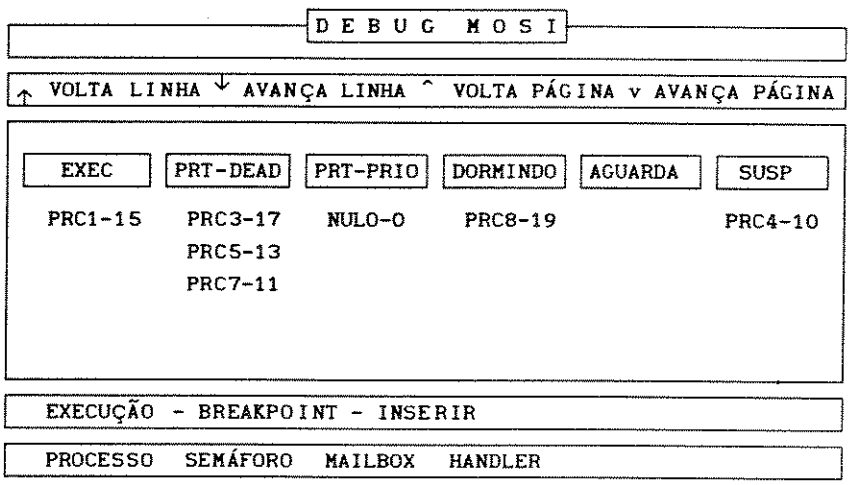


FIG. 3.9 - OPÇÃO INSERIR

a. SUBOPÇÃO PROCESSO.

Esta opção permite associar um BREAKPOINT à mudança de estado de um processo específico. O usuário escolhe o processo através da digitação de seu identificador. Caso o identificador seja válido o usuário é questionado com relação ao estado do processo onde será executada a parada. A fig. 3.10 ilustra a tela da opção PROCESSO bem como as linhas que aparecerão na janela de diálogo.

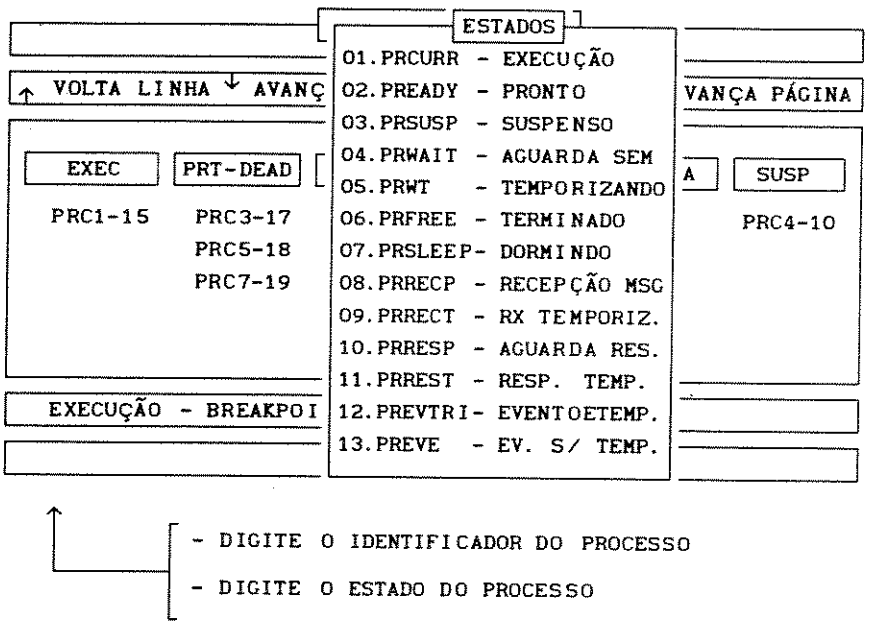


FIG. 3.10 - SUBOPÇÃO PROCESSO

b. SUBOPÇÃO MAILBOX

Esta opção permite que, uma vez fornecido o identificador da mailbox, seja efetuada uma parada no instante em que um número específico de mensagens nessa mailbox for atingido. A fig. 3.11 mostra a tela correspondente a esta opção.

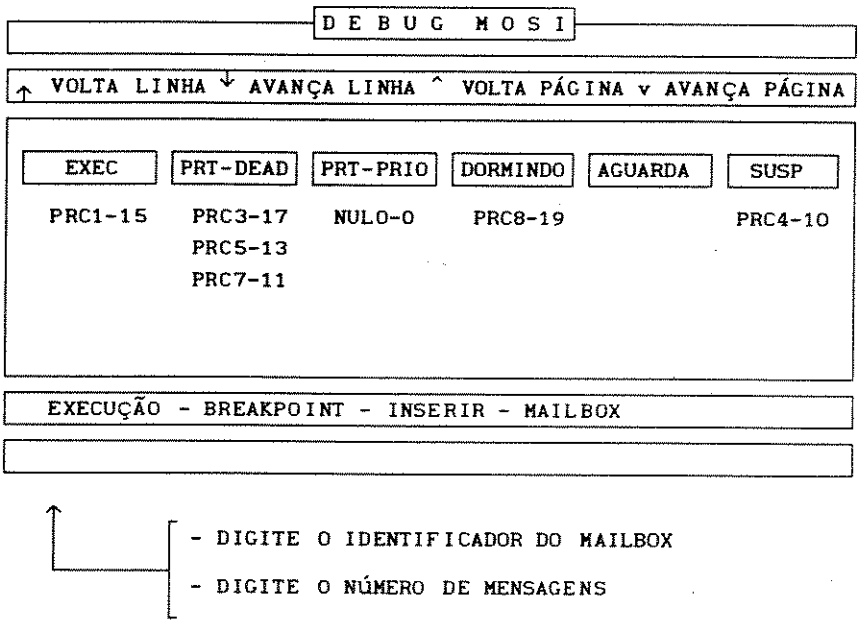


FIG. 3.11 - SUBOPÇÃO MAILBOX

c. SUBOPÇÃO SEMÁFORO

Esta opção permite, uma vez fornecido o identificador do semáforo, que a parada seja executada quando o semáforo atingir um valor previamente estipulado. A fig. 3.12 exibe a tela desta opção e salienta a sequência a ser apresentada na janela de diálogo.

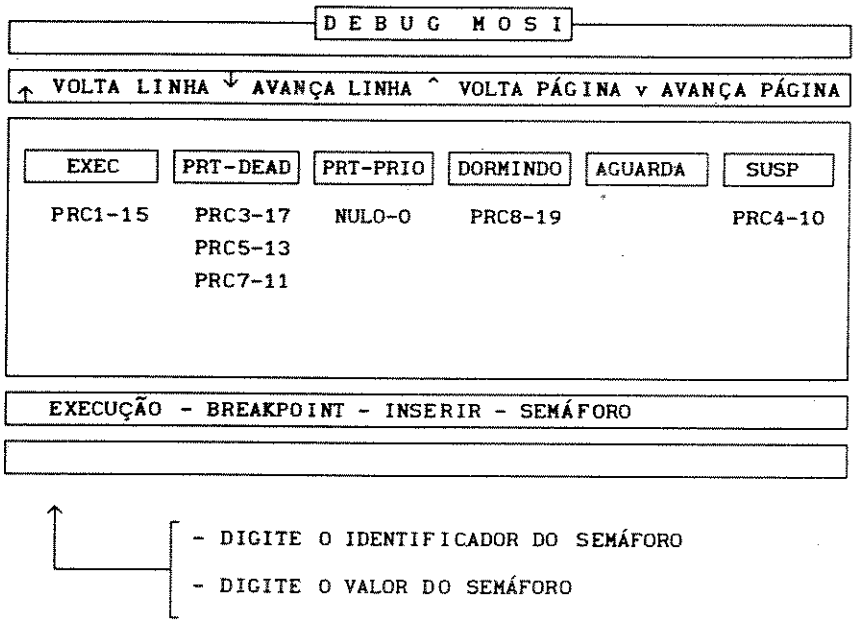


FIG. 3.12 - SUBOPÇÃO SEMÁFORO

d. SUBOPÇÃO HANDLER

Esta opção permite, uma vez fornecida o classe do handler, parar a execução quando o número de ativações do mesmo atingir um valor previamente estipulado. A fig. 3.13 exibe a tela desta subopção e salienta a sequência a ser apresentada na janela de diálogo.

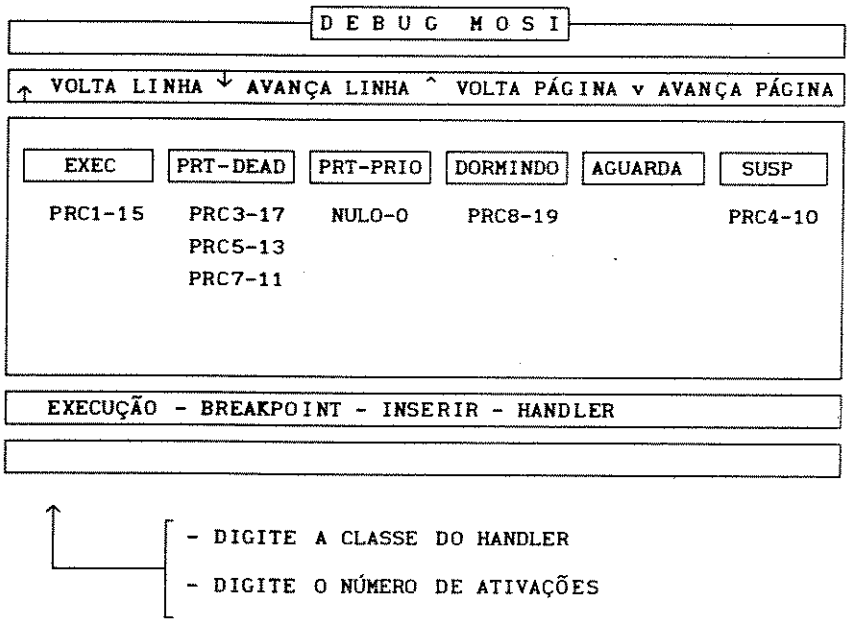


FIG. 3.13 - SUBOPÇÃO HANDLER

3.3.2.2 OPÇÃO DELETAR.

Esta opção possibilita a retirada de breakpoints. Será apresentada uma janela, onde uma tabela numerada segundo a ordem de entrada dos breakpoints na mesma, (sempre respeitando o número máximo permitido), possibilitará a escolha dos ítens a serem excluidos. A tela gerada pela opção **DELETAR** aparece na fig. 3.14. Serão detalhados na sequência, cada uma das subopções citadas.

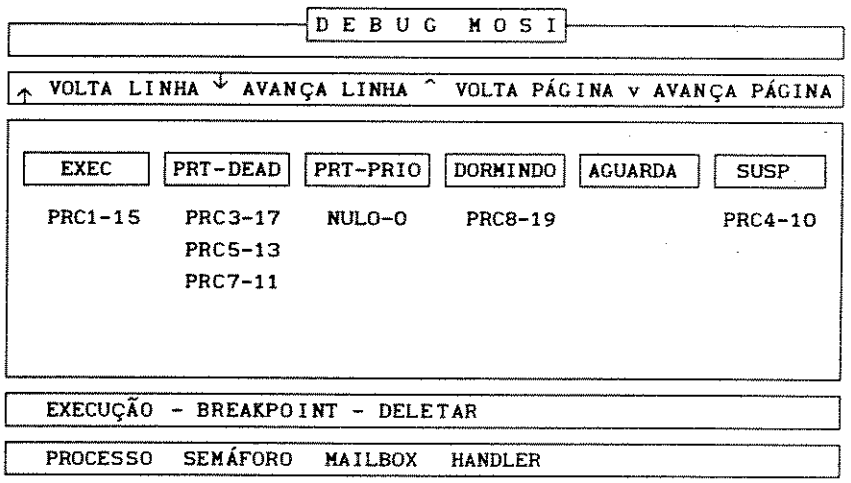


FIG. 3.14 - OPÇÃO DELETAR

a. SUBOPÇÃO PROCESSO.

Esta opção permite a exclusão de breakpoints associados à mudança no estado de um processo específico. A fig. 3.15 ilustra a tela da opção PROCESSO bem como as linhas que aparecerão na janela de diálogo.

```
                    D E B U G   M O S I
↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD    [ BREAKPOINTS          A    SUSP
                          PROCESSO   ESTADO
PRC1-15     PRC3-11                                   PRC4-10
            PRC5-17       1.PROC1    SUSPENSO
            PRC7-18       2.PROC2    DORMINDO

EXECUÇÃO - BREAKPOINT - DELETAR - PROCESSO

DIGITE A POSIÇÃO DO PROCESSO NA TABELA
```

FIG. 3.15 - SUBOPÇÃO PROCESSO

b. SUBOPÇÃO MAILBOX

Esta opção permite a exclusão de breakpoints associados a um mailbox específico. A fig. 3.16 mostra a tela correspondente a esta opção.

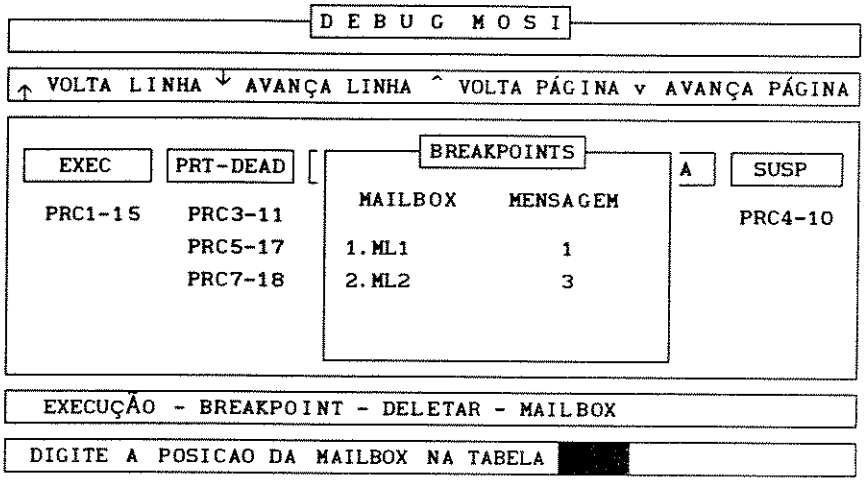


FIG. 3.16 - SUBOPÇÃO MAILBOX

c. SUBOPÇÃO SEMÁFORO

Esta opção permite a exclusão de breakpoints associados a um semáforo específico. A fig. 3.17 exibe a tela desta opção e salienta a sequência apresentada na janela de diálogo.

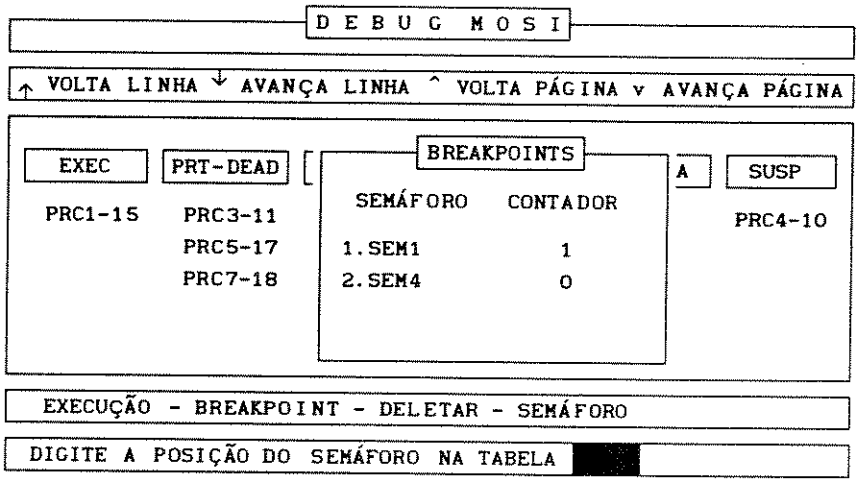


FIG. 3.17 - SUBOPÇÃO SEMÁFORO

d. SUBOPÇÃO HANDLER

Esta opção permite a exclusão de breakpoints associados a uma classe específica de um handler. A fig. 3.18 exibe a tela desta subopção e salienta a sequência apresentada na janela de diálogo.

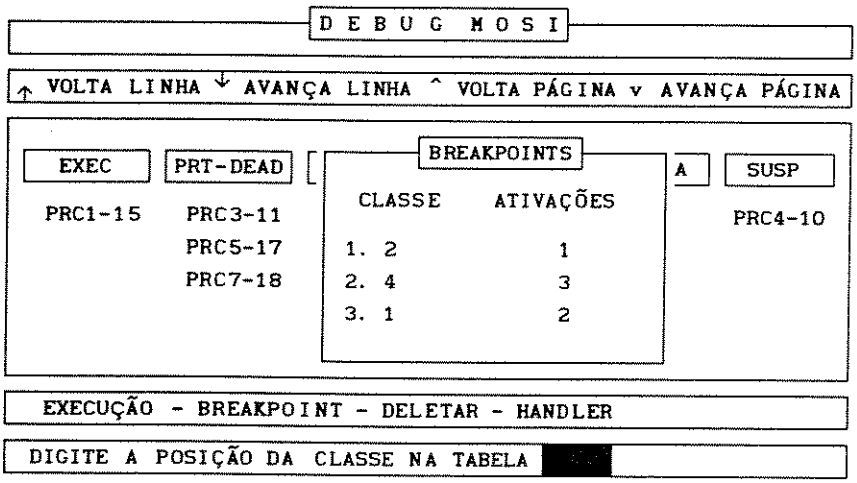


FIG. 3.18 - SUBOPÇÃO HANDLER

3.3.2.3 OPÇÃO LISTAR.

Esta opção possibilita a listagem dos breakpoints inseridos, por subopção. A tela gerada pela opção **LISTAR** aparece na fig. 3.19. Serão detalhados na sequência, cada uma das subopções citadas.

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD    PRT-PRIO    DORMINDO    AGUARDA     SUSP

PRC1-15     PRC3-17     NULO-0      PRC8-19                 PRC4-10
            PRC5-13
            PRC7-11

EXECUÇÃO - BREAKPOINT - LISTAR

PROCESSO  SEMÁFORO  MAILBOX  HANDLER
```

FIG. 3.19 - OPÇÃO LISTAR

a. SUBOPÇÃO PROCESSO.

Esta opção permite a listagem de breakpoints associados à mudança no estado de um processo específico. A fig. 3.20 ilustra a tela da opção PROCESSO bem como as linhas que aparecerão na janela de diálogo.

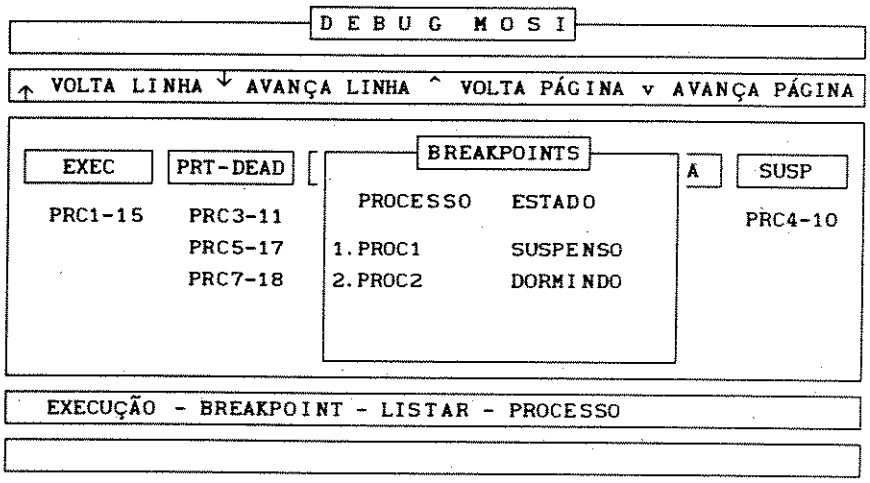


FIG. 3.20 - SUBOPÇÃO PROCESSO

b. SUBOPÇÃO MAILBOX

Esta opção permite a listagem de breakpoints associados a um mailbox específico. A fig. 3.21 mostra a tela correspondente a esta opção.

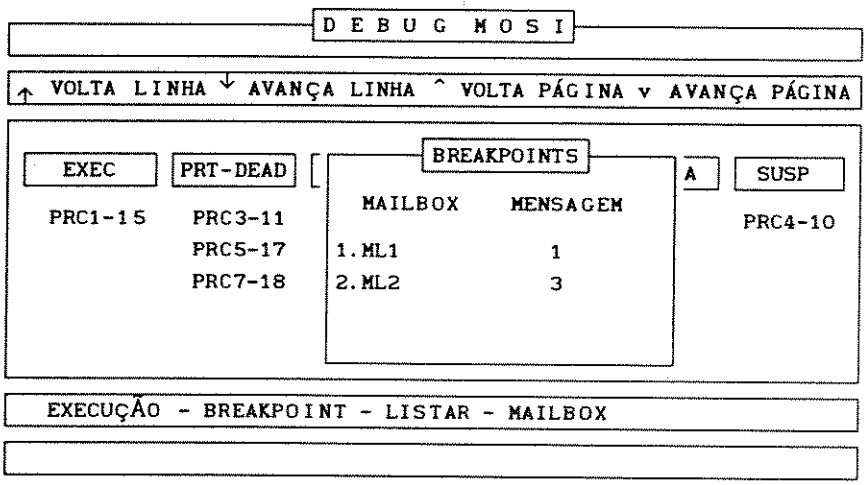


FIG. 3.21 - SUBOPÇÃO MAILBOX

c. SUBOPÇÃO SEMÁFORO

Esta opção permite a listagem de breakpoints associados a um semáforo específico. A fig. 3.22 exibe a tela desta opção e salienta a sequência a ser apresentada na janela de diálogo.

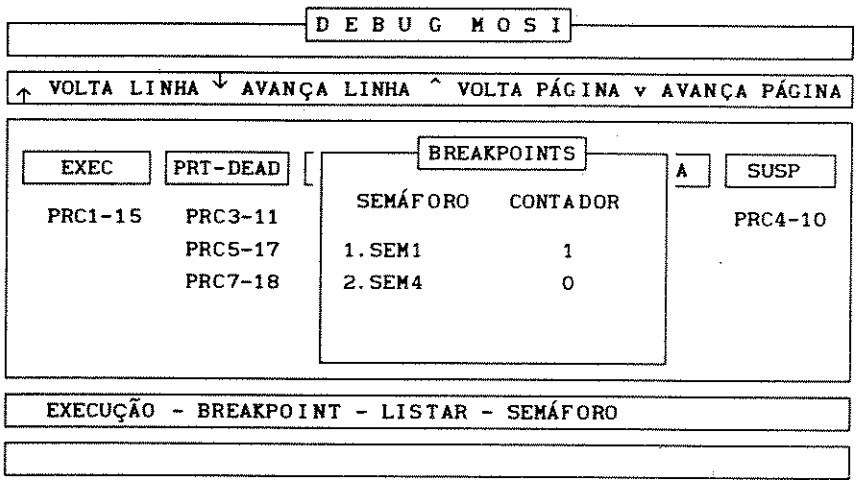


FIG. 3.22 - SUBOPÇÃO SEMÁFORO

d. SUBOPÇÃO HANDLER

Esta opção permite a listagem de breakpoints associados a uma classe específica de um handler. A fig. 3.23 exibe a tela desta subopção e salienta a sequência a ser apresentada na janela de diálogo.

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD        [ BREAKPOINTS        A    SUSP
                              CLASSE   ATIVAÇÕES
PRC1-15     PRC3-11                                    PRC4-10
            PRC5-17           1. 2        1
            PRC7-18           2. 4        3
                              3. 1        2

EXECUÇÃO - BREAKPOINT - LISTAR - HANDLER
```

FIG. 3.23 - SUBOPÇÃO HANDLER

3.3.3 SUBCOMANDO PASSO A PASSO.

Este subcomando permite a execução gradual e controlada do sistema. A execução será realizada em etapas onde "passo" é definido como qualquer evento que altere o estado global do sistema. Cada evento ocorrido no núcleo será reportado na forma de uma mensagem, que descreverá o tipo de evento e a entidade associada com o mesmo. A fig. 3.24 mostra a tela gerada pelo comando (na verdade, a tela de **EXECUÇÃO** será mantida).

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC      PRT-DEAD   PRT-PRIO   DORMINDO   AGUARDA   SUSP

PRC1-15   PRC3-17    NULO-0     PRC8-19              PRC4-10
          PRC5-13
          PRC7-11

CRIAÇÃO DO PROCESSO PROC4

PASSO A PASSO  BREAKPOINT  EXECUTE  REINICIALIZA
```

FIG. 3.24 - OPÇÃO PASSO A PASSO.

3.3.4 SUBCOMANDO GRAVAÇÃO.

Este subcomando permite o armazenamento de uma execução direta ou passo a passo de um aplicativo. A gravação de partes relevantes da depuração também é possível. Quando de sua chamada o subcomando questiona sobre o local e nome onde será gravado o arquivo de execução. Para a gravação de partes a delimitação é realizada pressionando-se novamente a subopção gravação ("ON/OFF"). Os dados gravados, encontram-se em formato "bruto", sendo seu refinamento realizado por um utilitário externo ao depurador chamado **RELMOSI**. Este utilitário gera como saída um relatório formatado relacionando tempo x evento, além de destacar o processo que está executando a cada evento. A sua utilização deve ser realizada na forma: **RELMOSI ARQ. ENTRADA ARQ. SAÍDA**. A figura 3.25 ilustra o uso do subcomando.

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC      PRT-DEAD   PRT-PRIO   DORMINDO   AGUARDA   SUSP

PRC1-15   PRC3-17    NULO-0     PRC8-19              PRC4-10
          PRC5-13
          PRC7-11

EXECUÇÃO - GRAVAÇÃO

DIGITE O PATH E NOME DO ARQUIVO
```

FIG. 3.25 - SUBCOMANDO GRAVAÇÃO

3.4 COMANDO SUPERVISÃO.

Este comando possibilita a verificação de algumas entidades associadas ao núcleo como **PROCESSOS**, **MAILBOX**, **SEMÁFORO**, **MEMÓRIA** e **HANDLER**. A interface com o usuário é realizada através de janelas que se sobrepõem à tela de fundo corrente. São realizadas consistências dos dados digitados em todos os subcomandos e em caso de erro a mensagem **IDENTIFICADOR INCORRENTO-REDIGITE** será mostrada. A fig. 3.26 mostra a tela gerada pelo comando.

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC      PRT-DEAD   PRT-PRIO   DORMINDO   AGUARDA   SUSP

PRC1-15   PRC3-17    NULO-0     PRC8-19              PRC4-10
          PRC5-13
          PRC7-11

SUPERVISÃO

PROCESSO  MAILBOX  SEMÁFORO  MEMÓRIA  HANDLER
```

FIG. 3.26 - COMANDO SUPERVISÃO

3.4.1 SUBCOMANDO PROCESSO.

Uma vez dado o identificador do processo e verificada sua validade as seguintes informações estão disponíveis: **IDENTIFICAÇÃO, PRIORIDADE, QUANTIDADE DE MEMÓRIA, PARÂMETROS** recebidos pelo processo quando de sua criação e **ESTADO** segundo a tabela mostrada na fig. 2.0. Caso o processo esteja aguardando um semáforo/mailbox será também apresentado o identificador do elemento bem como o tempo máximo de espera. A fig. 3.27 mostra a tela gerada pelo subcomando.

```
                    D E B U G   M O S I
↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD        PROCESSO        A    SUSP

PRC1-15     PRC3-11     IDENTIFICAÇÃO:  PROC1      PRC4-10
            PRC5-17     PRIORIDADE:     2
            PRC7-18     QUANT. MEMÓRIA: 300
                        ESTADO: EXECUTANDO

SUPERVISÃO - PROCESSO

DIGITE O IDENTIFICADOR DO PROCESSO
```

FIG. 3.27 - OPÇÃO PROCESSO

3.4.2 SUBCOMANDO MAILBOX.

Uma vez fornecida uma identificação correta da mailbox a janela gerada pelo subcomando conterá as seguintes informações: **IDENTIFICAÇÃO** e **NUMERO DE MENSAGENS**, como ilustrado pela fig. 3.28.

```
D E B U G   M O S I
↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD    P [ MAILBOX ]    SUSP

PRC1-15     PRC3-11     IDENTIFICAÇÃO: ML1     PRC4-10
            PRC5-17     NÚMERO DE MSG: 4
            PRC7-18

SUPERVISÃO - MAILBOX

DIGITE O IDENTIFICADOR DO MAILBOX
```

FIG. 3.28 - OPÇÃO MAILBOX

3.4.3 SUBCOMANDO SEMÁFORO.

Este subcomando, de posse de uma identificação válida do semáforo, apresenta as seguintes informações: **IDENTIFICAÇÃO** e **VALOR ATUAL**, como pode ser observado na fig. 3.29.

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD    P   SEMÁFORO    SUSP

PRC1-15     PRC3-11         IDENTIFICAÇÃO: SEM1     PRC4-10
            PRC5-17         VALOR ATUAL  : 1
            PRC7-18

SUPERVISÃO - SEMÁFORO

DIGITE O IDENTIFICADOR DO SEMÁFORO
```

FIG. 3.29 - OPÇÃO SEMÁFORO

3.4.4 SUBCOMANDO MEMÓRIA.

Apresenta a quantidade de memória livre e ocupada pelo sistema. A fig. 3.30 mostra a tela gerada por este subcomando.

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD    P   MEMÓRIA                    SUSP

PRC1-15     PRC3-11         MEMÓRIA LIVRE  : 21346     PRC4-10
            PRC5-17
            PRC7-18         MEMÓRIA OCUPADA: 3233

SUPERVISÃO - MEMÓRIA
```

FIG. 3.30 - OPÇÃO MEMÓRIA

3.4.5 SUBCOMANDO HANDLER.

Uma vez fornecido um identificador válido para a classe do handler estarão disponíveis as seguintes informações: **CLASSE**, **NOME DO HANDLER** e **ESTADO** conforme mostra a fig. 3.31.

```
D E B U G   M O S I
↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD    P [ HANDLER ]                SUSP
PRC1-15     PRC3-11       CLASSE:            2       PRC4-10
            PRC5-17       NOME DO HANDLER:   DFLT
            PRC7-18       ESTADO:            HAB

SUPERVISÃO - HANDLER
DIGITE O IDENTIFICADOR DO HANDLER
```

FIG. 3.31 - OPÇÃO HANDLER

3.5 COMANDO MAIS.

Pseudo comando que tem por função permitir a ampliação do número de opções do menu. Através dele são acessados os comandos estatística e simulação.

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD    PRT-PRIO    DORMINDO    AGUARDA     SUSP
PRC1-15     PRC3-17     NULO-0      PRC8-19                 PRC4-10
            PRC5-13
            PRC7-11

ESTATÍSTICA   SIMULAÇÃO
```

FIG. 3.32 - COMANDO MAIS.

3.6 COMANDO ESTATÍSTICA

Apresenta estatística das entidades associadas ao núcleo como: **PROCESSO**, **MAILBOX**, **SEMÁFORO**, **HANDLER** e a atuação geral do **SISTEMA**. A interface com o usuário será realizada através de janelas que conterão basicamente gráficos do comportamento de cada entidade. A implementação dos gráficos utilizados nas janelas em todos os subcomandos será feita através de texto, e não utilizando caracteres gráficos devido à problemas de velocidade. A unidade de tempo utilizada em todos os gráficos é TICK que representa o número de períodos do relógio interno da máquina. A fig. 3.33 mostra a tela inicial deste comando. Nesta versão este comando não será implementado.

```
                    [D E B U G   M O S I]
[↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA]

[EXEC] [PRT-DEAD] [PRT-PRIO] [DORMINDO] [AGUARDA] [SUSP]
PRC1-15  PRC3-17   NULO-0    PRC8-19            PRC4-10
         PRC5-13
         PRC7-11

[ESTATÍSTICA]

[PROCESSO  MAILBOX  SEMÁFORO  MEMÓRIA  SISTEMA]
```

FIG. 3.33 - COMANDO ESTATÍSTICA

3.6.1 SUBCOMANDO PROCESSO.

Este subcomando fornecerá dados como: **TEMPO DE EXECUÇÃO, NÚMERO DE CHAMADAS AOS SEMÁFOROS, NÚMERO DE CHAMADAS A MAILBOX TEMPO MÉDIO DE ESPERA PELA CPU**. A fig. 3.34 mostra a tela inicial deste subcomando.

```
D E B U G   M O S I

↑ VOLTA LINHA ↓ AVANÇA LINHA ^ VOLTA PÁGINA v AVANÇA PÁGINA

EXEC        PRT-DEAD    P [ PROCESSO ]                SUSP
                          TEMPO DE EXECUÇÃO: 10ms
PRC1-15     PRC3-11       NUM. CHAMADAS SEM: 4        PRC4-10
            PRC5-17       NUM. CHAMADAS MAIL: 7
            PRC7-18       TEMPO MÉDIO DE ESPERA
                                PELA CPU: 1ms

ESTATÍSTICA - PROCESSO
```

FIG. 3.34 - SUBCOMANDO PROCESSO

3.6.2 SUBCOMANDO MAILBOX.

Este subcomando mostra uma janela contendo dois gráficos. O primeiro relacionando **NÚMERO DE PROCESSOS x TICK** e o segundo **NÚMERO DE MENSAGENS x TICK**. A fig. 3.35 mostra a tela gerada pelo subcomando. O primeiro gráfico representa o número de processos aguardando na mailbox por uma mensagem no período de tempo compreendido desde o início da simulação até o presente momento. O segundo gráfico representa o número de mensagens existentes na mailbox no período de tempo compreendido entre o início da simulação e o presente momento.

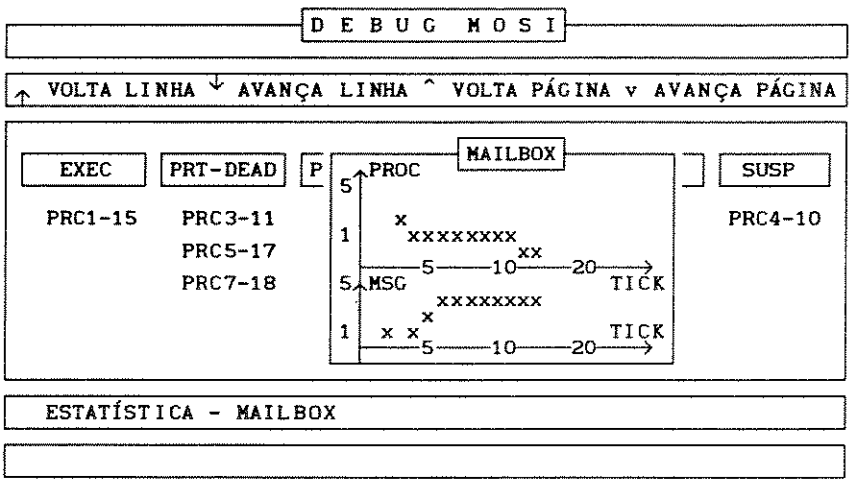


FIG. 3.35 - SUBCOMANDO MAILBOX

3.6.3 SUBCOMANDO SEMÁFORO.

Este subcomando apresenta um gráfico relacionando **PROCESSOS x TICK** e outro com **VALOR SEMÁFORO x TICK**. A fig. 3.36 mostra a tela gerada pelo subcomando. O primeiro gráfico representa o número de processos aguardando por uma liberação do semáforo no período de tempo compreendido desde o início da simulação até o presente momento. O segundo gráfico representa o valor do contador do semáforo no período de tempo compreendido desde o início da simulação até o presente momento.

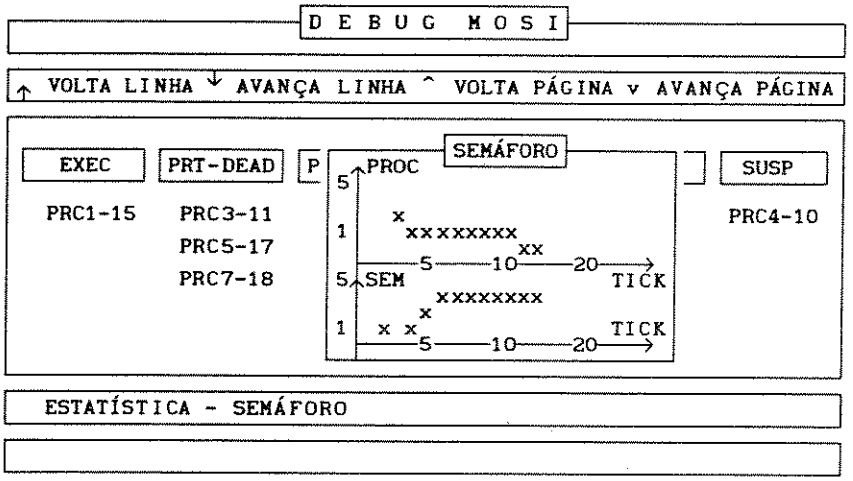


FIG. 3.36 - SUBCOMANDO SEMÁFORO

3.6.4 SUBCOMANDO MEMÓRIA.

Este subcomando apresenta uma janela com um gráfico que relaciona **MEMÓRIA LIVRE** em **KBYTES x TICK** e outro. com **MEMÓRIA OCUPADA** em **KBYTES x TICK**. A fig. 3.37 mostra a tela gerada pelo subcomando.

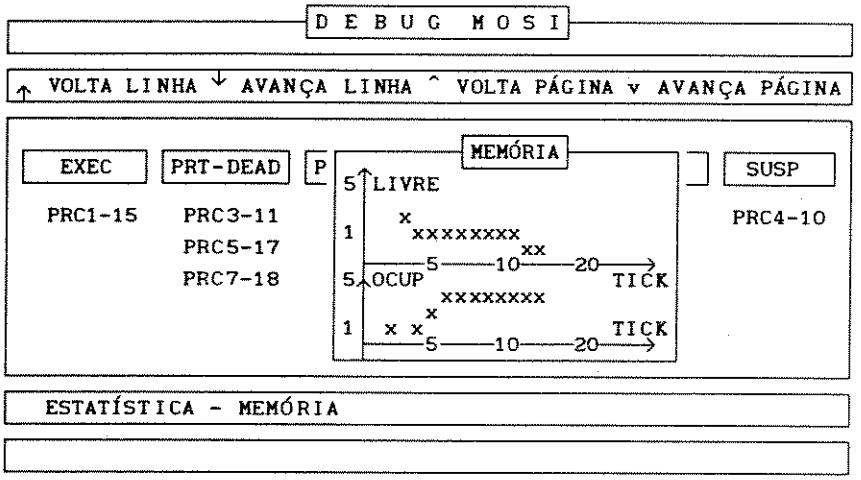


FIG. 3.37 - SUBCOMANDO MEMÓRIA

3.6.5 SUBCOMANDO SISTEMA.

Este subcomando procura dar uma visão do **TEMPO OCIOSO** do sistema pelo número de **TICK** decorridos. A fig. 3.38 mostra a tela gerada pelo subcomando.

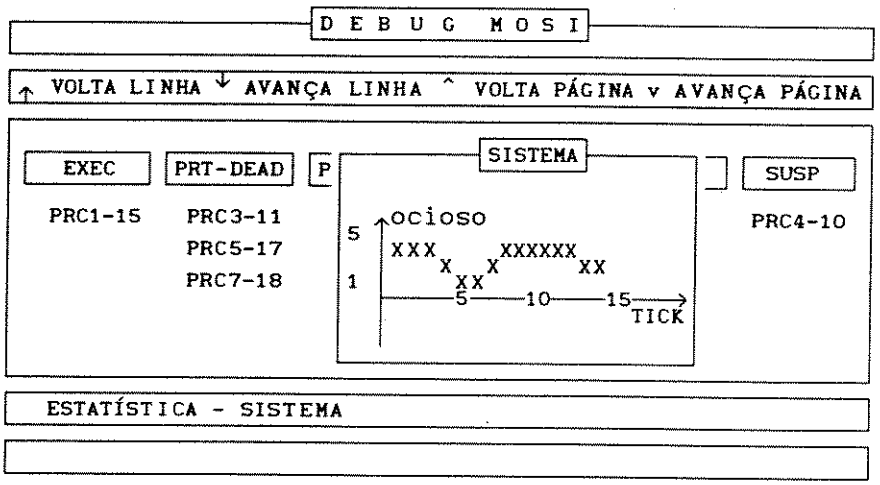


FIG. 3.38 - SUBCOMANDO SISTEMA

3.7 COMANDO SIMULAÇÃO.

Este comando permite a reexecução do sistema sob depuração, desde que o mesmo seja previamente armazenado em um arquivo pelo subcomando gravação. Durante a simulação as mesmas opções presentes durante a execução normal, estarão disponíveis, com exceção do subcomando de gravação. É possível o abandono do modo de simulação, selecionando-se novamente simulação no menu.

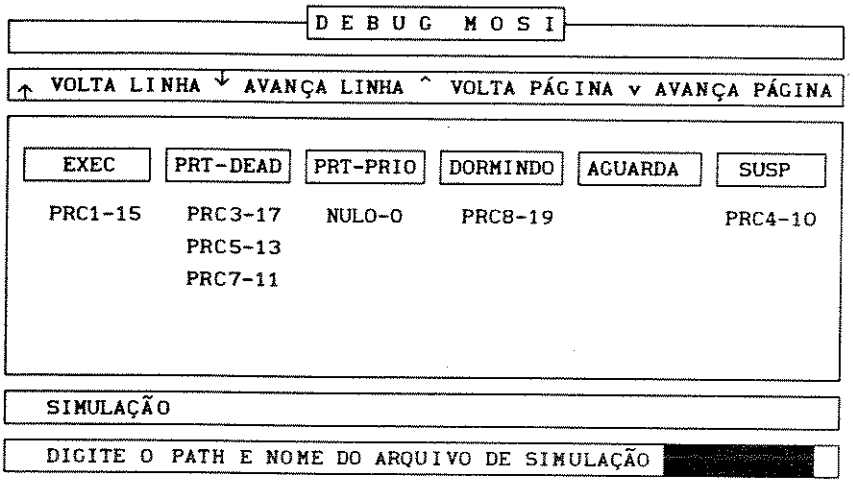


FIG. 3.39 - COMANDO SIMULAÇÃO.

3.8 COMANDO SAÍDA.

A utilização deste comando faz com que o depurador seja encerrado. O sistema sob supervisão será avisado, e também encerrará sua execução.

4.1 TABELA.

MÓDULO	CÓDIGO
GERÊNCIA	100 - ERRO NA MENSAGEM → ANÁLISE 101 - ERRO NA MENSAGEM → CÓDIGO INVÁLIDO 102 - IDENTIFICADOR DE PROCESSO INVÁLIDO 103 - IDENTIFICADOR DE SEMÁFORO INVÁLIDO 104 - IDENTIFICADOR DE "MAILBOX" INVÁLIDO 105 - IDENTIFICADOR DE "HANDLER" INVÁLIDO 106 - FILA COM PROBLEMA → PID NÃO ENCONTRADO 107 - LISTA COM PROBLEMA → PID NÃO ENCONTRADO
PROTOCOLO	201 - "OVERFLOW BUFFER" DE TRANSMISSÃO 202 - "OVERFLOW" Nº DE RETRAMSMISSÕES 203 - "OVERFLOW BUFFER" DE RECEPÇÃO
COMUNICAÇÃO	300 - "UNDERFLOW BUFFER" DE RECEPÇÃO 301 - "OVERRUN" 302 - PARIDADE 303 - "FRAMING" 304 - "OVERFLOW BUFFER" DE RECEPÇÃO 305 - "OVERFLOW BUFFER" DE ENVIO 306 - "UNDERFLOW BUFFER" DE ENVIO
DISCO	400 - GRAVAÇÃO EM DISCO