Prof. Celso Maciel da Costa
Prof. Simão Sirineo Toscani
Prof. Thadeu Botteri Corso

Curso de Pós-Graduação em Ciência da Computação
Universidade Federal do Rio Grande do Sul
Caixa Postal 1501
90.001 - Porto Alegre - RS
Telefone (0512) 21-2161
BRASIL

Este trabalho descreve o projeto MICRO-BIS, em desenvolvimento na Universidade Federal do Rio Grande do Sul, que consiste na construção de uma máquina multiprocessadora simples e na implementação do software para essa máquina. Dá-se detalhes sobre as chamadas do sistema, o tratamento de interrupções, a exclusão mútua no acesso aos dados globais e a inicialização do sistema.

1. INTRODUÇÃO

Alguns anos atrás os grupos de pesquisa em arquitetura de computadores e sistemas operacionais do Curso de Pós-Graduação em Ciência da Computação (CPGCC) da Universidade Federal do Rio Grande do Sul (UFRGS) decidiram construir um sistema multiprocessador que fosse adequado para a execução de programas escritos na linguagem Pascal Concorrente. O projeto, denominado Multimicro /TOS 82/, foi concebido a partir de sugestões apresentadas por Brinch Hansen /HAN 78/ e correspondia a uma arquitetura envolvendo vários processadores e vários módulos de memória. Infelizmente, devido à complexidade do projeto, à inexistência de laboratórios adequados, às dificuldades para se obter os recursos materiais necessários e também ao afastamento de alguns pesquisadores do projeto, o mesmo foi descontinuado após 2 anos de trabalho.

O motivo principal do insucesso do projeto Multimicro foi a grande complexidade do sistema a ser construído. Esta constatação levou parte dos pesquisadores envolvidos no projeto à decisão de construir um novo sistema bem mais simples, mas que pudesse resultar em um protótipo acabado, em um curto espaço de tempo. A existência de um protótipo em funcionamento serviria de estímulo para a elaboração de projetos mais ambiciosos no futuro.

O resultado foi a definição de uma máquina bimicroprocessadora, denominada MICRO-BIS, com arquitetura simples, de fácil implementação, que é adequada para a execução de programas Pascal Concorrente.

A máquina MICRO-BIS, conforme é mostrado na figura 1, é formada por dois microprocessadores MC68000, cada qual com uma memória local de 128 KB e ambos compartilhando uma memória global também de 128 KB. Estes componentes estão dispostos em uma placa ligada a um microcomputador PC/XT que serve como processador de entrada e saída (PES).

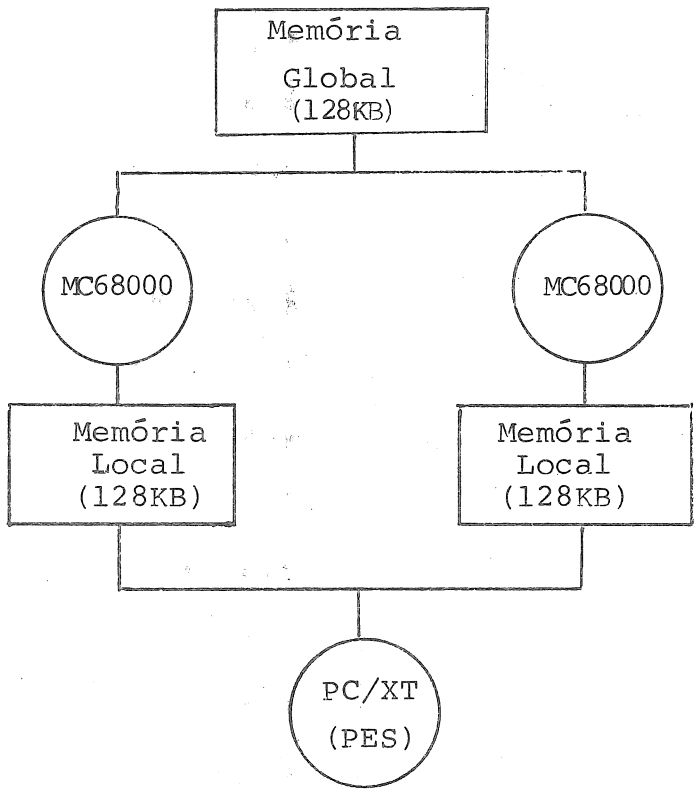

Figura 1. Arquitetura da máquina MICRO-BIS

O PES realiza suas funções de entrada e saída através do acesso às memórias locais dos microprocessadores MC68000. O PES pode interromper a ambos os microprocessadores e esses podem interromper ao PES. Também, os microprocessadores podem interromper-se mutuamente.

As características básicas dessa máquina, assim como a organização do sistema Pascal Concorrente e do sistema de entrada e saída, já foram apresentadas na XII Conferência Latino Americana de Informática /TOS 86/. O presente trabalho é uma complementação daquele.

2. JUSTIFICATIVA DA ARQUITETURA DA MÁQUINA

Faz parte do projeto MICRO-BIS, além da construção da máquina bimicroprocessadora e da implementação do sistema Pascal Concorrente, também a implementação de um sistema operacional multiusuário e a implementação de um sistema Lisp concorrente. Estes dois últimos sistemas justificam plenamente a arquitetura escolhida para a máquina MICRO-BIS, conforme será mostrado nesta seção.

O sistema Lisp concorrente, denominado COLISP /GEY 86/, é composto por dois módulos concorrentes: um simulador de uma máquina virtual para a interpretação de uma variante da linguagem Lisp e um coletor de lixo. A figura 2 mostra a interação desses módulos através do espaço de memória compartilhado por eles. Esses módulos devem ser executados em processadores diferentes por questões de eficiência. A máquina MICRO-BIS representa, portanto, uma resposta adequada a essa exigência.

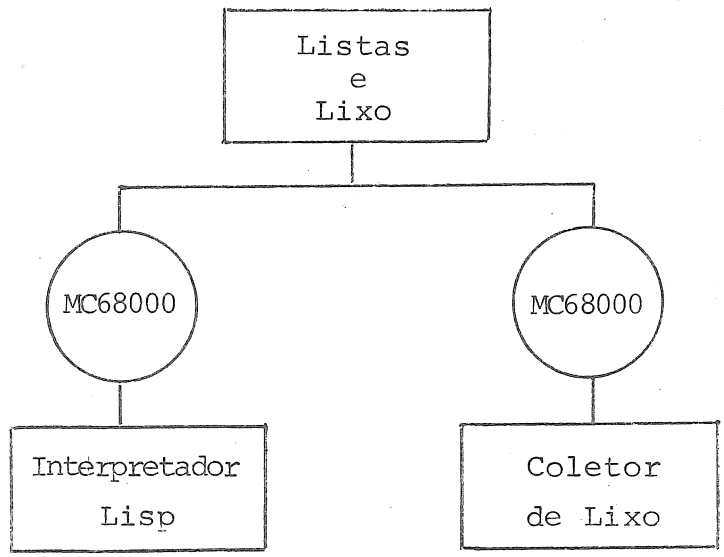

Figura 2. O sistema Lisp concorrente

Por outro lado, a arquitetura é conveniente para a execução de programas concorrentes cujos processos se intercomuniquem através de monitores. Este é o caso de várias linguagens de programação, entre as quais encontra-se o Pascal Concorrente /HAN 77/. Nessas linguagens, os processos executam a maior parte do tempo as suas próprias instruções e somente quando eles precisam se intercomunicar (trocar informações ou acessar dados compartilhados) é que eles executam nos monitores. A figura 3 mostra como ficariam distribuídos os componentes de um programa Pascal Concorrente na máquina MICRO-BIS.

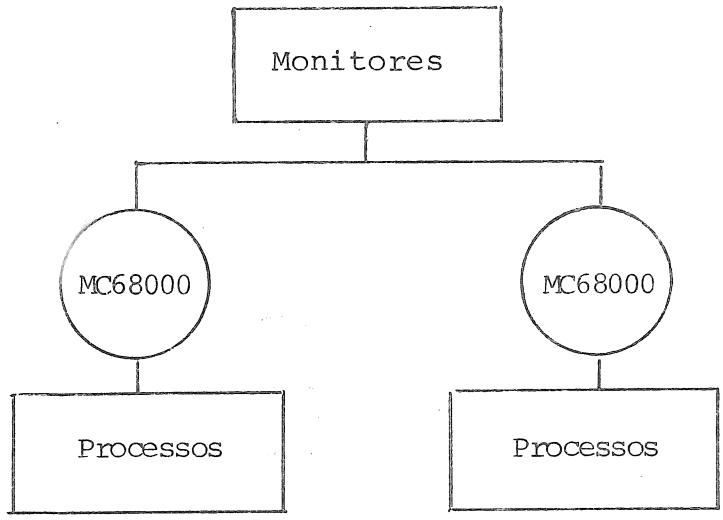

Figura 3. Distribuição dos componentes de um programa Pascal Concorrente

Na realidade, ficam nas memórias locais dos micro processadores cópias do segmento comum aos processos (núcleo, interpretador e código virtual) e na memória global, as estruturas que implementam os monitores e os dados compartilhados. O segmento privativo de um processo é alocado na memória local do microprocessador que for executá-lo. Mais detalhes sobre a distribuição desses componentes na máquina MICRO-BIS podem ser encontrados em /TOS 86/.

O sistema operacional multiusuário para a máquina MICRO-BIS, denominado SOMBRA /OLI 87/, foi escrito em Pascal Concorrente e tem as seguintes características principais:

- multiusuário interativo;
- linguagem de comandos com construções de alto nível;
- sistema de arquivos organizado em árvore;
- acesso controlado ao sistema de arquivos;
- execução de programas em background;
- spooling de saída para impressora;
- controle de acesso ao sistema através de senhas.

3. CHAMADAS DO SISTEMA

Como se sabe, os compiladores do Sistema Pascal Concorrente geram código para uma máquina virtual, a qual é formada por um interpretador e um núcleo.

Um programa de usuário somente pode ser executado quando ele é carregado para a área privativa de algum processo. Quando o programa necessita de alguma função do núcleo, o mesmo invoca um procedimento do processo no qual ele está encapsulado, sendo esse processo o que efetivamente executa a chamada.

Existem, portanto, três entidades envolvidas numa chamada de sistema: o programa de usuário, o processo e a rotina do núcleo.

Os processadores MC68000 da máquina MICRO-BIS possuem dois modos de operação: modo supervisor e modo usuário.

O programa de usuário e o processo são executados em modo usuário, o núcleo em modo supervisor. A mudança de modo usuário para supervisor é feita automaticamente pela instrução TRAP e pelas interrupções do hardware.

A instrução TRAP possui como operando um número entre Ø e 15 que identifica a entrada no vetor de tratamento de exceções (vetor de interrupções) que contém o endereço da rotina do núcleo que deve ser executada.

A implementação das chamadas do sistema foi feita da seguinte forma:

Um programa seqüencial em execução tem seu código executado pelo interpretador da máquina virtual. Eventualmente, o interpretador encontra a invocação de um procedimento do processo e passa então a executá-lo. Esse procedimento pode conter a chamada de uma função do núcleo que, ao ser identificada, ocasiona a execução da instrução TRAP. Com isso, o modo de operação do processador passa para supervisor e a execução prossegue no endereço contido no vetor de tratamento de exceções.

Existem 12 funções do núcleo que podem ser chamadas pelos processos, cada uma implementada por uma rotina específica. Como o vetor de tratamento de exceções do processador possui 16 entradas para TRAPS, sobram ainda quatro entradas desse vetor.

A última instrução de uma rotina do núcleo deve ser RTE (RETURN FROM EXCEPTION). Essa instrução restaura o contexto anterior à chamada da função do núcleo. Deve ser observado que após a execução dessa instrução o processador poderá ficar no modo supervisor ou no modo usuário. Ficará no modo usuário se a chamada do sistema tiver sido feita por um processo.

A passagem dos parâmetros necessários para a execução das diferentes funções do núcleo é feita através de uma área de memória comum ao interpretador e ao núcleo.

4. TRATAMENTO DAS INTERRUPÇÕES

As fontes de interrupção de hardware da máquina MICRO-BIS são três: interrupção de relógio, a cada 3,2 ms, interrupção do outro processador e interrupção do processador de entrada e saída (PES). O PES também pode ser interrompido pelos processadores principais. Para cada interrupção existe uma rotina específica de tratamento.

As interrupções de relógio são usadas para fazer escalonamento de processos e para implementar contagem de tempo. A cada cinco interrupções é atualizada a tabela de contagem de tempo e verificado, em uma fila, se o tempo de espera de algum processo já se esgotou. Em caso afirmativo, o processo é inserido na fila de processos aptos de acordo com sua prioridade. Após isso, é chamada a rotina que faz o escalonamento. O processo selecionado pelo scheduler é o de maior prioridade, podendo ser, inclusive, o que estava executando no momento da interrupção.

Um processador interrompe o outro através de um registrador de comandos programável. O processador interrompido desvia para uma rotina que acessa uma estrutura de dados, na memória global, usada para troca de informações.

De maneira análoga um processador interrompe o PES. Esse, quando interrompido desvia para uma rotina que acessa uma estrutura de dados na memória local do processador que o interrompeu, a qual contém a fila de requisições de entrada e saída.

Finalmente, o PES pode interromper a qualquer um dos dois processadores principais, o que de fato faz ao término de uma operação de entrada e saída. Um processador principal, ao ser interrompido, pelo PES, desvia para uma rotina que acessa a estrutura de dados em sua memória local, a qual contém o resultado da operação. Se a operação foi bem sucedida, o processo que a tinha solicitado é inserido na fila de aptos, de acordo com sua prioridade.

5. EXCLUSÃO MÚTUA NO ACESSO AOS DADOS COMPARTILHADOS

Conforme já foi visto, na máquina MICRO-BIS os dados compartilhados ficam na memória global. Existem três estruturas de dados compartilhadas nessa memória: um buffer que serve para troca de mensagens entre os processadores, um contador de requisições de entrada e saída, usado para que o atendimento das requisições seja feito na ordem da solicitação, e uma tabela com os registros descritores dos processos. O registro descritor de um processo contém, entre outras coisas, a identificação do processador que executa esse processo.

O acesso a essas estruturas deve ser feito de forma exclusiva pelos processadores. Para o controle do acesso

foram utilizados três semáforos: um para a tabela de descritores de processos, outro para o buffer de troca de mensagens e o terceiro para o contador de requisições de entrada e saída.

Existe ainda uma área destinada à conter 25 outros semáforos, os quais são utilizados para controlar o acesso aos monitores que, como se sabe, são os mecanismos usados pelos processos para sincronização e troca de mensagens.

A implementação dos semáforos é feita com a instrução TAS (Test And Set). Esta instrução permite testar e modificar o valor do operando de forma indivisível (em uma única operação).

Para obter acesso a uma estrutura de dados compartilhada, o processador executa a operação P. A liberação do acesso é feita através da operação V. Supondo um semáforo S inicializado com 1, as operações P e V são implementadas da seguinte maneira:

```
P:   TAS   S
     BNE   P
     RTS

V:   MOVE  $1, S
     RTS
```

6. SISTEMA DE ENTRADA E SAÍDA

A organização e o funcionamento do sistema de entrada e saída já foram apresentados na XII Conferência Latino Americana de Informática /TOS 86/. Entretanto, houve uma pequena mudança em relação ao que foi lá apresentado. Essa mudança foi a troca do PES, motivada pela intenção de dar maior portabilidade para o sistema multiprocessador. O papel do PES passou a ser desempenhado por um microprocomputador PC, mais universal que o minicomputador ED-281 previsto originalmente. Assim, a placa com o sistema multiprocessador (2 microprocessadores MC68000 e 3 módulos de memória) poderá ser instalada facilmente em qualquer microcomputador PC, no futuro.

7. INICIALIZAÇÃO DO SISTEMA

O processo de inicialização do sistema acontece em duas fases distintas. Na primeira delas, o PES carrega, a partir do byte 1024 de cada memória privativa, o núcleo do sistema operacional (o núcleo é carregado logo após o vetor de interrupções que ocupa os primeiros 1024 bytes). Carrega ainda, na entrada do vetor de interrupções correspondente à interrupção do PES, o valor do stack pointer do sistema e

o endereço da rotina de inicialização (valor inicial do PC) correspondente ao processador. A seguir, através de seu registrador de comandos, o PES executa a operação RESET nos processadores principais.

A partir daí inicia a segunda fase do processo de inicialização. Cada processador acessa seu vetor de interrupções para buscar o valor do stack pointer e do PC e iniciar sua execução.

A rotina inicial de um dos processadores inicializa o restante do seu vetor de interrupções (colocando o endereço da rotina de tratamento de cada TRAP e de cada interrupção), inicializa os dados globais e, por último, faz a chamada da rotina do núcleo que cria o processo ancestral. A partir daí, o processo ancestral, ganha o processador e passa a ser executado.

Simultaneamente, o outro processador executa uma rotina que inicializa o restante do seu vetor de interrupções e executa a instrução STOP. Esta instrução faz com que o processador fique em estado de espera até ser interrompido pelo outro processador.

O processo ancestral é o responsável pela criação dos demais processos do sistema. As informações que indicam quais processos devem ser executados por cada processador estão contidas em uma tabela. No momento em que o processo ancestral identifica um processo que deve rodar no outro processador, ele coloca todas as informações necessárias para a criação desse processo no buffer de comunicação dos processadores e interrompe o outro processador.

A rotina de tratamento dessa interrupção acessa o buffer de comunicação e, de posse das informações alí existentes, chama o procedimento do núcleo que faz a inicialização de um processo.

Passada a fase de inicialização, cada processador terá a seu encargo a execução de um conjunto de processos.

8. CONCLUSÃO

O projeto MICRO-BIS deverá terminar ainda este ano. O hardware da máquina multiprocessadora foi concluído e testado no ED-281 (antigo PES) e atualmente a placa está sendo adaptada para ser ligada ao microcomputador PC/XT (novo PES).

O sistema Lisp concorrente /GEY 86/ foi todo escrito em C, tendo sido construído um programa Pascal Turbo para fazer a carga dos códigos do interpretador e do coletor de lixo nas memórias locais dos processadores MC68000 (códigos de máquina para o MC68000 gerados pelo compilador C). O sistema foi desenvolvido e testado num computador

ED-680, esperando-se que com pouco esforço o mesmo possa ser colocado em funcionamento na máquina MICRO-BIS.

O sistema operacional SOMBRA /OLI 87/ foi todo escrito em Pascal Concorrente, tendo sido testado no minicomputador LABO 8034, que dispõe de um sistema Pascal Concorrente /MED 81/. A próxima etapa será a transferência e o teste do SOMBRA na máquina MICRO-BIS.

O núcleo do sistema operacional descrito neste trabalho foi programado na linguagem assembly do MC68000 e está em fase de depuração. O sistema de entrada e saída a ser executado pelo PES está sendo desenvolvido como dissertação de mestrado por um aluno do CPGCC, devendo ser concluído até o final do corrente ano. Este sistema deverá permitir a execução simultânea de várias operações de entrada e saída (funcionamento simutâneo de vários periféricos).

Espera-se que em um novo projeto os pesquisadores dos grupos de arquitetura de computadores e sistemas operacionais venham a se envolver na construção de uma máquina mais poderosa, com um número maior de processadores, o que permitirá o desenvolvimento de outras pesquisas em arquiteturas multiprocessadoras e em novos softwares para essas arquiteturas.

9. BIBLIOGRAFIA

/GEY 86/ GEYER, C.F.R. Implementação de um interpretador para uma linguagem funcional com coletor de lixo concorrente. Porto Alegre, CPGCC/UFRGS, 1986. (Dissertação de mestrado).

/HAN 77/ HANSEN, P.B. The architecture of concurrent programs. Englewood Cliffs, N.J. Prentice-Hall, 1977.

/HAN 78/ HANSEN, P.B. Multiprocessor architectures for concurrent programs. In: ACM 78, Washington, D.C., Dec. 1978. Proceedings. New Yor, ACM, 1978. p.317-323.

/HOA 74/ HOARE, C.A.R. Monitors: an operating systems structuring concept. ACM, 17(10):549-557, Oct. 1974.

/MED 81/ MEDEIROS, G.C.R. Implementação do sistema PASCAL CONCORRENTE no Labo 8034. Porto Alegre, CPGCC da UFRGS, 1981. (Dissertação de mestrado).

/OLI 87/ OLIVEIRA, R.S. SOMBRA: um sistema operacional multiusuário. Porto Alegre, CPGCC/UFRGS, 1986. (Dissertação de mestrado).

/TOS 82/ TOSCANI, S.S. et alii. Multimicro Pioneiro: protótipo de sistema multiprocessador orientado para execução de programas Pascal Concorrente. In: CONGRESSO NACIONAL DE INFORMÁTICA, 15., Rio de Janeiro, 1982. Anais. Rio de Janeiro, SUCESU/ /RJ, 1982.

/TOS 86/ TOSCANI, S.S. et alii. Implementação do sistema Pascal Concorrente na máquina MICRO-BIS. In: CONFERÊNCIA LATINO AMERICANA DE INFORMÁTICA - PANEL' 86, 12., Montevidéu, 1986. Anais. Montevidéu, Uruguai, Grafiservice S.R.L., 1986.