IMPLEMENTAÇÃO DO SISTEMA PASCAL CONCORRENTE NA MAQUINA MICRO-BIS

Simão Sirineo Toscani, Thadeu Botteri Corso, Celso Maciel da Costa
Universidade Federal do Rio Grande do Sul
Porto Alegre - *Brasil*

1. INTRODUÇÃO

Este artigo descreve um sistema (hardware e software) adequado para a construção e execução de programas concorrentes. Trata-se de uma máquina denominada MICRO-BIS, com dois processadores e três módulos de memória, que é adequada para a execução de programas escritos em Pascal Concorrente. O sistema aqui descrito foi concebido a partir das sugestões apresentadas por Brinch Hansen /HAN 78/ e está sendo construído no Curso de Pós-Graduação em Ciência da Computação da UFRGS. No restante desta seção serão apresentados conceitos básicos que facilitam o entendimento do artigo.

Em muitas aplicações de computadores, o sistema (hardware e software) deve ter a capacidade de executar uma variedade de tarefas que são requisitadas de forma imprevisível. Programas concorrentes permitem organizar sistemas deste tipo, convenientemente, de modo a habilitá-los ao atendimento "simultâneo" de várias requisições de serviço, conforme será explicado brevemente a seguir.

Um programa concorrente é constituído por vários módulos (processos seqüenciais) que, por serem relativamente independentes, podem ser executados em paralelo. A execução paralela desses módulos pode ser real (física), quando cada processo é executado por um processador próprio, ou virtual (lógica), quando os processos são executados de forma multiplexada por um único processador. Assim, para permitir o atendimento simultâneo de várias requisições de serviço, basta fazer com que cada tarefa seja executada por um dos processos do sistema.

Os sistemas operacionais de computadores são exemplos de sistemas que possibilitam, aos usuários, a execução de diversos tipos de tarefas (edição de arquivos, compilação e execução de programas, por exemplo), as quais podem ser requisitadas de forma imprevisível. De fato pode-se generalizar afirmando que os sistemas operacionais tem por função o atendimento de múltiplos usuários que requisitam a execução de seqüências de tarefas de forma imprevisível. A execução simultânea de múltiplas tarefas permite que os recursos computacionais (hardware e software) sejam utilizados de forma mais econômica e eficiente. Hoje em dia, mesmo em máquinas de pequeno porte, a maioria dos sistemas operacionais já permite a execução simultânea de várias tarefas.

Muitas aplicações importantes de computadores apresentam a mesma característica de ter que atender a uma variedade de requisições de serviço que ocorrem de forma imprevisível (sistemas para controle "on-line" de informações,

como contas bancárias e estoques, sistemas para reservas de passagens, sistemas para controle de processos industriais, etc). Todas essas aplicações podem ser mais eficientemente implementadas com o auxílio de linguagens de programação concorrente.

As linguagens de programação concorrente também estão estimulando a construção de sistemas operacionais orientados para aplicações específicas, os quais, evidentemente, são mais simples, mais econômicos, mais eficientes e mais confiáveis que os sistemas de propósitos gerais (oferecidos pelos fabricantes) capazes de executar uma grande variedade de serviços que nem sempre são utilizados.

Em um programa concorrente, os processos são relativamente autônomos, mas não independentes. Eles necessitam interagir para trocar informações ou competir por recursos. As linguagens de programação concorrente devem, portanto, prover mecanismos que facilitem a programação das interações entre os processos. Pelo fato dos monitores serem os mecanismos de interação utilizados na linguagem Pascal Concorrente, para a qual é orientado o sistema descrito neste artigo, os mesmos serão apresentados, informalmente, a seguir.

O conceito de monitor foi utilizado pela primeira vez por Brinch Hansen /HAN 70/ no projeto de um sistema operacional multiprogramado. Desde que este conceito foi formalizado por Hoare /HOA 74/, o mesmo vem adquirindo importância cada vez maior, já tendo sido incluído, em várias linguagens de programação concorrente (CSP/K, Módula, Pascal Concorrente e Euclid Concorrente, por exemplo). Os monitores são mecanismos de alto nível para sincronização e intercomunicação de processos, de fácil utilização, que impõem uma estruturação aos programas concorrentes que os utilizam. Um monitor é, na realidade, uma estrutura de dados abstrata com facilidades para sincronização de processos. Pode ser visto como um bloco que contém internamente dados locais e procedimentos para manipular esses dados.

Os dados declarados dentro de um monitor só são acessíveis nos procedimentos locais ao monitor, os quais são executados de forma mutuamente exclusiva, quando chamados pelos processos. Um monitor garante, portanto, exclusão mútua na manipulação desses dados. Além dos tipos de dados usuais, um novo tipo pode ser usado: CONDITION. Variáveis desse tipo só podem ser declaradas dentro de monitores e só podem ser utilizadas através de duas operações especiais: WAIT e SIGNAL. Estas operações permitem que processos sejam colocados à espera de condições que serão sinalizadas por outros processos.

2. O SISTEMA PASCAL CONCORRENTE

O Pascal Concorrente é uma linguagem de programação de alto nível, criada com o objetivo de permitir a cons

trução de sistemas operacionais e programas de controle em tempo real, confiáveis, para computadores de pequeno e médio porte, com um mínimo de esforço e tempo. A linguagem permite a programação de processos concorrentes capazes de se comunicar através de monitores. Segundo Brinch Hansen /HAN 77/, o uso do Sistema Pascal Concorrente é capaz de reduzir em uma ordem de grandeza o esforço despendido na implementação de um sistema concorrente, justificando, assim, a política de confecção de sistemas operacionais dedicados, orientados para atendimento às peculiaridades de aplicações específicas.

O Sistema Pascal Concorrente é composto pelos seguintes elementos:

- Um compilador para a linguagem Pascal Seqüencial SPASCAL /HAR 77/;
- Um compilador para a linguagem Pascal Concorrente, denominado CPASCAL /HAR 77/;
- Um interpretador para o código de uma máquina virtual /MED 81/;
- Um núcleo de sistema operacional /MED 81/.

Os compiladores do sistema, encontram-se disponíveis no código da máquina virtual. Isto significa que ao implementar-se o interpretador e o núcleo da máquina virtual, automaticamente se obtém o ambiente de execução necessário para se dispor dos compiladores Pascal Seqüencial e Concorrente e de todos os programas escritos nessas linguagens. Em particular, o sistema operacional SONIX /COS 85/, compatível com o UNIX, escrito em Pascal Concorrente, poderá ser utilizado, sem alteração alguma, como sistema operacional da máquina MICRO-BIS.

2.1 A Linguagem SPASCAL

A linguagem SPASCAL é baseada na definição de Wirth /JEN 75/, com pequenas restrições e variações. É uma linguagem procedural, simples e estruturada, com construções que permitem a definição de tipos de dados, manipulação de cadeias de caracteres, operações entre registros de mesmo tipo, criação dinâmica de variáveis em uma área de trabalho denominada heap, chamada de procedimentos recursivos, além das operações aritméticas e de controle usuais em linguagem de alto nível.

SPASCAL não possui funções intrínsecas da entrada e saída de dados. Estas funções devem ser implementadas pelo sistema operacional e comunicadas ao compilador SPASCAL através de uma porção do programa fonte denominado prefixo. Essa característica facilita o desenvolvimento de novos sistemas operacionais com funções de entrada e saída diferentes, pois os compiladores não estão, à princípio, vinculados com nenhum sistema operacional específico.

2.2 A Linguagem CPASCAL

A linguagem CPASCAL foi projetada especialmente para permtir a escrita de programas concorrentes que cumprem as funções básicas de sistemas operacionais, Foram feitas restrições e extensões a SPASCAL, chegando-se a um modelo de definição de processos e monites semelhante ao formalizado inicialmente por Hoare /HOA 74/.

Um programa concorrente é geralmente composto por um conjunto de processos que compartilham o processador, sob a gerência do núcleo o sistema operacional. Tais processos podem comunicar-se através de monitores, que representam os mecanismos que garantem o acesso exclusivo às seções críticas do programa concorrente /HOL 78/. Os processos possuem também construções que permitem a passagem de controle de execução para programas seqüenciais, escritos em SPASCAL, e carregados em áreas específicas dos processos. Uma vez em execução, os programas seqüenciais podem solicitar serviços aos processos que os supervisionam, através das interfaces contidas no prefixo.

A linguagem CPASCAL não permite procedimentos recursivos nem criação dinâmica de variáveis, para evitar crescimento incontrolado dos dados de execução. O conceito de classe introduz a utilização de reentrância de código objeto para monitores e processos concorrentes. A especificação completa da linguagem CPASCAL está descrita em /HAN 77/.

2.3 O Interpretador e o Núcleo da Máquina Pascal Concorrente

Os compiladores produzem código objeto para uma máquina virtual (Máquina Pascal Concorrente - MPC), com 112 isntruções. A palavra da MPC é de 16 bits e cada código de operação ou operando de instrução ocupa uma palavra de memória. O esquema de endereçamento permite acesso a caracteres de 8 bits, sendo, portanto, de 64 Kbytes o espaço de endereçamento virtual.

O núcleo contém funções primitivas, para compartilhamento de tempo de processamento, contagem do tempo real, exclusão mútua no acesso a monitores, comunicação entre processos e operações de entrada e saída.

Para execução de operações de entrada e saída no sistema existe uma única operação, denominada IO, com operandos que especificam o tipo de operação, o dispositivo de entrada e saída envolvido e os dados a serem transferidos pela operação. Uma instrução IO é executada de forma síncrona, isto é, a instrução que a segue no programa objeto só é executada após toda a operação de entrada e saída ter sido completada. Esta característica influiu fortemente na escolha da arquitetura para o MICRO-BIS, especialmente nos aspectos relacionados a entrada e saída dos dados.

O Sistema Pascal Concorrente, conforme definido por Brinch Hansen, pressupõe implementações em instalações com um único processador. Obviamente, para o caso do MICRO-BIS, onde existirão dois processadores, o núcleo deverá ser diferente.

O núcleo implementa também memórias virtuais para os processos. A memória virtual de cada processo é constituída por um segmento comum (igual para todos os processos) e um segmento privativo. O segmento comum contém o interpretador, o núcleo e o código virtual do programa concorrente em execução. O segmento privativo de um processo (segmento de dados) contém o stack e o heap desse processo.

3. ARQUITETURA DA MÁQUINA BIMICROPROCESSADORA

A máquina bimicroprocessadora MICRO-BIS, mostrada na figura 1, é formada por dois microprocessadores MC68000, uma memória local de 128 Kbytes para cada microprocessador e uma memória global, também de 128 Kbytes. Esses componentes estão dispostos em uma placa que deve ser ligada a um minicomputador ED-281 com a seguinte configuração: CPU Z8ØA, 2 terminais, 112 Kbytes de memória RAM, 2 unidades de discos flexíveis de 8", 1 unidade de disco rígido tipo winchester e 1 impressora de 1ØØ caracteres por segundo. O minicomputador ED-281 será o processador de entrada e saída (PES) do sistema.

Cada microprocessador MC68000 pode acessar a sua memória local e a memória global. Dessa forma a intercomunica e troca de mensagens entre processos rodando em processadores diferentes somente pode ser realizada através da memória global.

Para o ED-281 poder realizar a sua função no sistema (execução das operações de entrada e saída) o mesmo deverá poder acessar a memória particular de cada microprocessador MC68000. Cada microprocessador MC68000 poderá interromper o outro e o ED-281, sendo que o último poderá interromper ambos os microprocessadores MC68000.

4. DISTRIBUIÇÃO DOS COMPONENTES DA MPC NA MÁQUINA REAL

A idéia que norteou a definição da divisão dos componentes da MPC na máquina real baseou-se no fato de que cada processador deve operar em sua memória privativa o máximo de tempo possível, com isso minimizando os acessos à memória global, o que conduz a um melhor desempenho do sistema.

Para tanto decidiu-se manter em cada memória privativa dos processadores uma cópia do segmento comum aos processos (núcleo, interpretador e o código virtual gerado pelo compilador Pascal Concorrente) e na memória global o segmento de dados do processo inicial, os dados locais dos

monitores e as estruturas de dados que implementam os monitores (um monitor é implementado por uma variável booleana que indica se o mesmo está livre ou não e uma fila onde ficam os processos a espera para entrar no monitor).

Na memória local de um processador também devem ser alocados os segmentos privativos dos processos que vão ser executados por esse processador. Como em Pascal Concorrente a intercomunicação de processos é feita através de monitores, o acesso à memória global somente é necessário quando um processo deseja acessar um monitor.

Com a duplicação do segmento comum nas memórias locais existe uma perda em relação a memória que poderia ser destinada aos segmentos privativos dos processos. Porém, se for considerado que cada memória local possui 128 Kbytes, que o núcleo e o interpretador somam juntos aproximadamente, 12 Kbytes e, que o código virtual do sistema operacional SONIX (que será o sistema utilizado na máquina) tem 18 Kbytes, resta em cada memória local 98 Kbytes. Supondo que o sistema operacional seja definido com 10 processos (a presente implementação da MPC suporta até 16), sendo alocados 5 para cada processador, então cada segmento privativo poderia ter aproximadamente 20 Kbytes.

5. SISTEMA DE ENTRADA E SAÍDA

A disparidade de velocidade entre dispositivos de entrada e saída e processador obriga a utilização de processadores de entrada e saída, isto é, dispositivos que visam liberar a CPU de executar tais operações.

Conforme já foi mencionado anteriormente, o sistema de entrada e saída será implementado no microcomputador ED-281 (o qual será referido como PES), sob o sistema operacional MP/M(Multi-programming monitor control program).

As rotinas de processamento de entrada e saída são freqüentemente as mais complexas num sistema operacional. Isto é devido, em parte, a natureza assíncrona dos dispositivos de entrada e saída e aos problemas de sincronização causados pelo paralelismo inerente da sua operação. Na solução adotada optou-se por implementar atendimento seqüencial das requisições de entrada e saída geradas pelos processos.

A seguir o ambiente do sistema é descrito caracterizando-se inicialmente as estruturas de dados e posteriormente sua filosofia de funcionamento.

Conforme a figura 2, em cada memória particular dos processadores MC68000 haverá uma fila de requisições de entrada e saída e uma fila de operações já executadas pelo PES. Os elementos dessas filas são estruturados da seguinte maneira:

```
type REQUISIÇÃO = record
                      NÚMERO-PROCESSO
                      PERIFÉRICO
                      TICKET
                      OPERAÇÃO
                      ENDEREÇO-LEITURA
                      ENDEREÇO-ESCRITA
                      STATUS
                  end;
```

O campo ticket foi utilizado para garantir que o atendimento das requisições de entrada e saída seja realizado na ordem em que as mesmas foram solicitadas pelos processos, seqüencializando desta forma as operações.

A estrutura que armazenará as requisições à medida em que forem geradas, é definida da seguinte maneira:

```
type FILA = array(Ø..15) of REQUISIÇÃO
```

À medida que as requisições forem sendo atendidas elas irão passando para a fila de operações executadas, representada na mesma estrutura acima.

A capacidade do array FILA foi estabelecida em 16 considerando-se que o número máximo de processos do sistema operacional a ser implementado está definido em 16 e que pode haver uma situação em que todos os processos sejam alocados em um único processador (a alocação é determinada em tempo de carga conforme será explicado na seção 6).

Na memória global aos processadores MC68000 haverá uma variável chamada TICKET que, inicializada com Ø, é incrementada pelo núcleo a cada requisição de entrada e saída realizada pelos processos. Com isto pode-se implementar ordem seqüencial de atendimento.

No PES não é necessária a definição de uma fila, pois a ordem de atendimento das requisições de entrada e saída será determinada pelo campo ticket que o processo "toma" (conforme acima citado) ao requisitar uma operação de entrada e saída. Dessa forma, ao ser interrompido, o PES verifica qual será a próxima requisição a ser atendida, obtendo os parâmetros para tratar a operação da fila na memória local do processador que causou a interrupção, garantindo assim uma política justa de atendimento.

A figura 3 mostra as diversas seções em que é dividido o sistema, as quais são a seguir comentadas.

Um processo quando requisita uma operação de entrada e saída perde o processador e é inserido na fila de processos bloqueados até a operação ter sido completada, quando então é colocado fila de processos prontos para novamente concorrer pelo processador. Entretanto, a ação de bloquear

os processos não faz parte do procedimento de requisição de entrada e saída. O interpretador ao reconhecer uma instrução deste tipo, realiza uma chamada ao núcleo do sistema operacional. Este por sua vez coloca os parâmetros na fila de requisição na sua respectiva memória local e interrompe o PES, executando a seguinte instrução:

```
MOVE $2,$4ØØØØ ; carrega no endereço 4ØØØØH o comando de
               ; interrupção do PES
```

Essa instrução faz com que o hardware transfira o valor armazenado na posição 4ØØØH (registrador de comandos do MC68000) para o registrador de status (E4H) do PES, onde a requisição é indicada pelo nível lógico 1 do bit correspondente ao processador (bit Ø corresponde a PØ e bit 1 corresponde a P1). Essa ação ocasiona uma interrupção IRQ14 que no PES é controlada por dois elementos PIC 8259 (Programmable Interrupt Controller), ligados na configuração mestre-escravo. A figura 4 mostra a arquitetura do sistema de forma mais detalhada, o que permite um melhor entendimento do funcionamento do mesmo.

A rotina de atendimento de interrupção, ao ser disparada, consulta o registrador de status e liga uma variável de controle indicando assim ao PES a existência de uma requisição a ser atendida.

O PES, através de uma rotina específica, transfere os parâmetros da fila na memória local do processador que o interrompeu para o seu registro de operação (na memória), que apresenta a seguinte estrutura:

```
type REGISTRO-DE-OPERAÇÃO = record
                              NÚMERO-DO-PROCESSO
                              PERIFÉRICO
                              OPERAÇÃO
                              ENDEREÇO-LEITURA
                              ENDEREÇO-ESCRITA
                              PROCESSADOR
                              STATUS
                              VARIÁVEL-CONTROLE
                            end;
```

Caso o PES esteja realizando uma operação de entrada e saída, esta será completada para após ser consultada a variável de controle. Se esta estiver ligada, serão consultadas as filas para verificar qual a próxima requisição a ser tratada. Note que esta variável permanecerá ligada enquanto houver requisições nas filas.

O endereçamento para transferência de dados de/para as memórias locais dos MC68000 é feito utilizando-se dois registradores de endereçamento (EØH e E1H), cujas estruturas são mostradas na figura 5. Como as memórias estão divididas em 8 K páginas de 16 bytes, são necessários 13 bits pa

ra endereçá-las (as páginas). Esses bits são obtidos através da concatenação dos 5 bits menos significativos do registrador EØH com os 8 bits do registrador ElH. O deslocamento dentro da página, ØH a FH, é obtido a partir dos 4 bits menos significativos das portas de endereço FØH a FFH do PES. O bit 5 do registrador EØH permite selecionar a memória local que se deseja referir.

Realizada a operação de entrada e saída, será devolvido uma indicação do resultado da mesma fila de status (ocupando o mesmo nodo em que estava representada a requisição, na estrutura da fila mostrada na figura 2) na memória local do processador que requisitou a operação. Esse processador será avisado do término da operação através de uma interrupção gerada pelo PES via registrador de comandos (E2H). Isto é feito através das seguintes instruções:

```
LD A, XH      ; carrega o acumulador com o comando desejado
OUT (ØE2H), A; escreve no registrador de comandos o comando
              ; armazenado no acumulador
```

onde X pode assumir os valores Ø ou 1 conforme o processador em questão.

Com isto o processador interrompido devolve o status da operação realizada para o processo que a requisitou, colocando o mesmo na fila de processos aptos.

Além da interrupção, o registrador de comandos do PES permite a execução de operações HALT e RESET nos processadores PØ e Pl (figura 6). Isto é feito de maneira semelhante ao procedimento de interrupção, utilizando-se os mesmos comandos acima descritos, mudando contudo o valor de X, conforme o comando desejado. Assim para executar uma operação HALT, X deverá assumir os valores 4H ou 8H, respectivamente, para PØ ou Pl. Da mesma forma, para executar a operação RESET, X assume os valores 1ØH (PØ) ou 20H (Pl).

O driver de entrada e saída é formado pelas rotinas de atendimento das requisições para cada periférico disponível no sistema, sendo sua estrutura básica definida da seguinte maneira:

```
CARGA-DO-SISTEMA;

loop
  if VARIÁVEL-CONTROLE
    then begin
         CARREGA-REGISTRO-DE-OPERAÇÃO;
         case REGISTRO-DE-OPERAÇÃO.PERIFÉRICO of
             "terminal"    : ...
             "disquete"    : ...
             "winchester"  : ...
             "impressora"  : ...
              else         : nada;
          end case;
          end
forever;
```

O sistema operacional é carregado nas memórias locais dos processadores através da rotina CARGA-DO-SISTEMA. Feito isto, é enviado um sinal de reset para ambos os processadores, fazendo com que estes comecem a executar. Realizado o procedimento inicial de carga, o PES entra em estado de busy-wait até ser interrompido por algum dos processadores, para atender uma requisição de entrada e saída.

6. INICIALIZAÇÃO DO SISTEMA

A inicialização do sistema é realizada a partir do PES. Incialmente deve ser executado o programa que implementa o sistema de entrada e saída. Este programa dispara o procedimento inicial de carga cuja função é transferir de disco para as memórias locais, em posições físicas pré-definidas, os códigos que implementam a MPC e o programa concorrente. Ao término da transferência da MPC e do sistema operacional para as memórias locais, cada processador recebe um comando reset que o habilita a entrar em estado de processamento.

Ao iniciar a execução, o processador acessa uma posição de memória que contém o endereço de um procedimento que estabelece o protocolo de conversação com o usuário, através do qual será definido quais os processos que deverão ser executados pelo processador. A identificação do processo deve ser feita pelo número que indica a ordem de execução da instrução INIT para esse processo no programa concorrente.

Ao término da associação de processos a processadores, o sistema estará apto a operar normalmente o interpretador passará a executar as instruções virtuais do programa concorrente.

7. CONCLUSÕES

O trabalho descrito neste artigo faz parte de um projeto em andamento. A arquitetura apresentada já tem seu projeto lógico concluído. Atualmente está sendo feita a fixação dos componentes na placa. O tempo estimado para a conclusão desse trabalho (ligação da placa no computador ED-281 e testes da mesma) é de 4 meses.

O sistema de entrada e saída está todo definido, estando em fase final de programação. As alterações que devem sofrer a MPC de modo a funcionar com 2 processadores já foram todas definidas, estando a mesma em fase de programação. O tempo previso para término da implementação do software é de seis meses, após a conclusão do hardware.

O desenvolvimento do software está sendo levado em paralelo com a construção do hardware. Isto é possível porque se dispõe de um computador ED680 cujo processador principal é um MC68000. Foi escolhida a linguagem C para a programação do sistema, em função da sua disponibilidade no

ED680 e da sua adequação para a produção de software básico.

É importante salientar que o sistema em construção é um protótipo. Todas as soluções adotadas levaram em conta principalmente a simplicidade de implementação para que o tempo de produção do sistema fosse mínimo. Pretende-se com isso, no menor prazo possível, ter o sistema operando, o que servirá de estímulo e incentivo para elaborar um projeto mais ambicioso de construção de uma máquina multiprocessadora.

BIBLIOGRAFIA

/COS 85/ COSTA, C.M. Um sistema operacional compatível com o UNIX para a Máquina Pascal Concorrente. In: XVIII CNI, São Paulo, setembro 1985. Anais. São Paulo, SUCESU, 1985. p.728-732.

/HAN 70/ HANSEN, P.B. The nucleus of a multiprogramming system. Comm. ACM 13(4):238-50, Apr. 1970.

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

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

/HAR 77/ HARTMAN, A.C. A concurrent pascal compiler for minicomputers. Berlin, Springer-Verlag, 1977. (Lectures Notes in Computer Science 50).

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

/HOL 78/ HOLT, R.C. et alii. Structured concurrent programming with operating systems applications. Reading, Addison-Wesley, 1978.

/JEN 75/ JENSEN, K. & WIRTH, N. PASCAL: user manual and report. New York, Springer-Verlag, 1975.

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

/THO 78/ THOMPSON, K. Unix implementation. The Bell System Technical Journal, 57(6):1931-46, July--August, 1978.

/WEB 82/ WEBER, T.S. & ROCHA, A.C. Descrição da arquitetura da máquina Pascal Concorrente. Porto Alegre, CPGCC-UFRGS, 1982. (relatório interno).

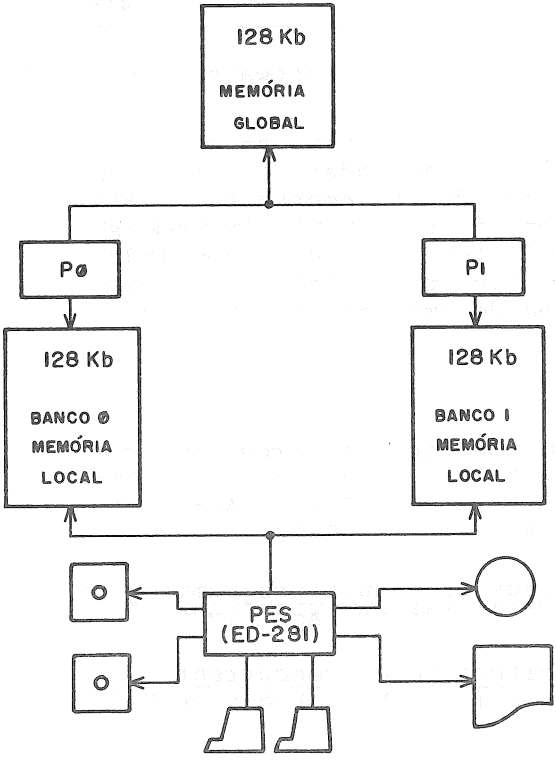
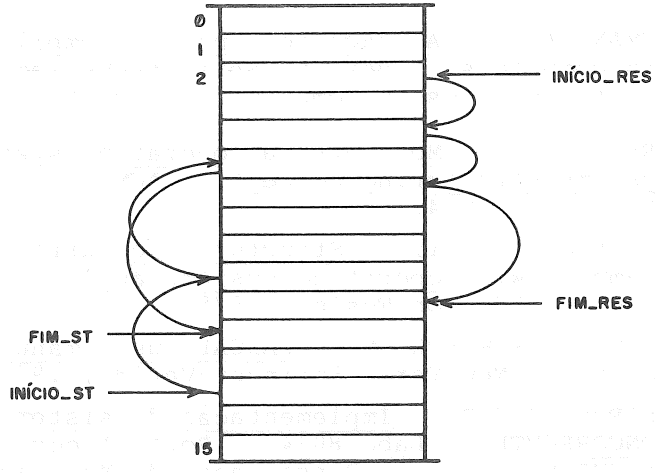


Figura 1. - Arquitetura do Sistema.

Figura 3. - Estrutura Lógica do Sistema de Entrada e Saída.



INÍCIO-RES: APONTADOR PARA INÍCIO DA FILA DE REQUISIÇÕES DE ENTRADA E SAÍDA;
FIM-RES: APONTADOR PARA FIM DA FILA DE REQUISIÇÕES DE ENTRADA E SAÍDA;
INÍCIO-ST: APONTADOR PARA INÍCIO DA FILA DE STATUS;
FIM-ST: APONTADOR PARA FIM DA FILA DE STATUS.

Figura 2. - Filas de Requisição de E/S e de Status nas Memórias Locais nos Processadores 68000.

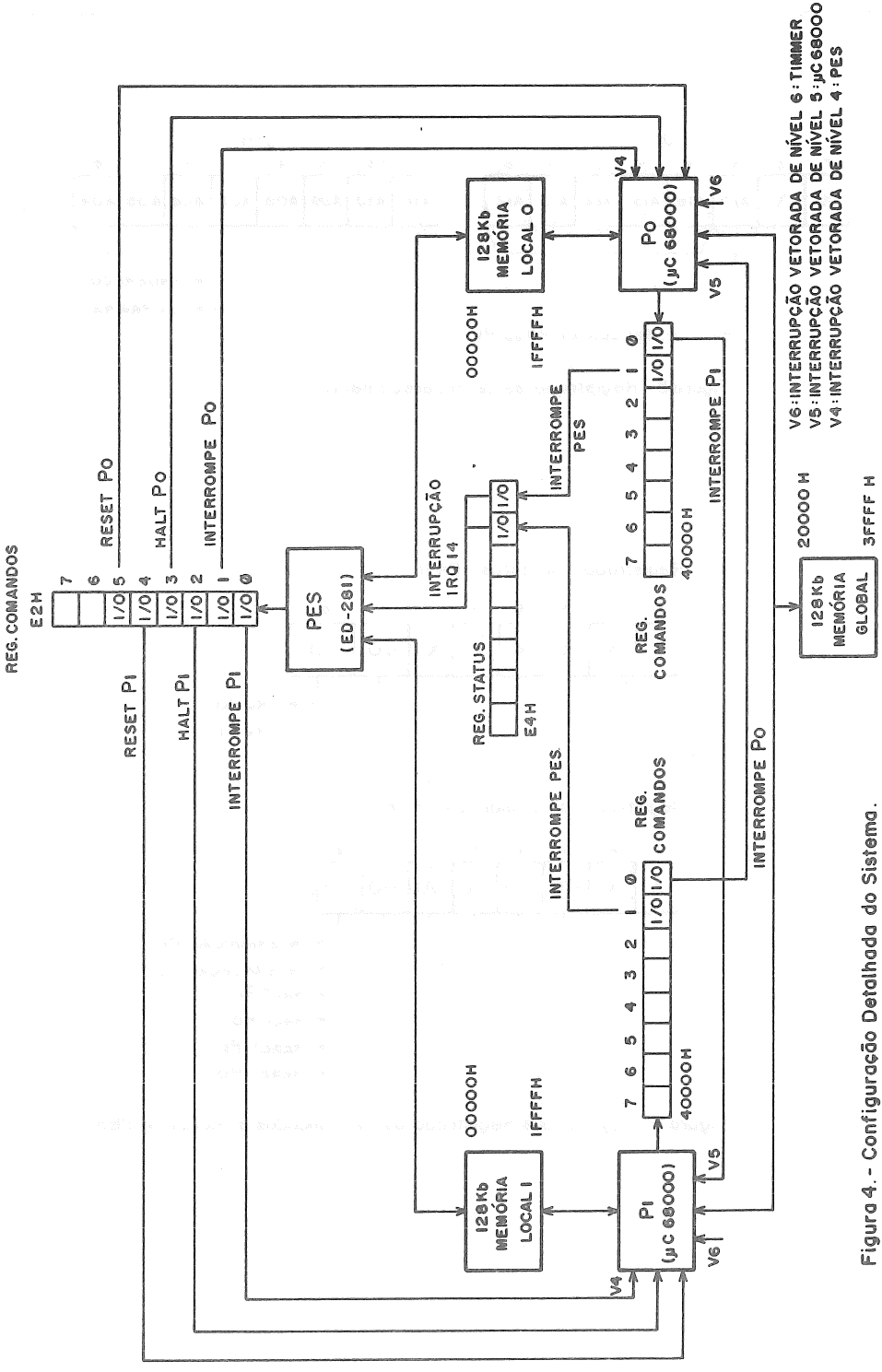


Figura 4. - Configuração Detalhada do Sistema.

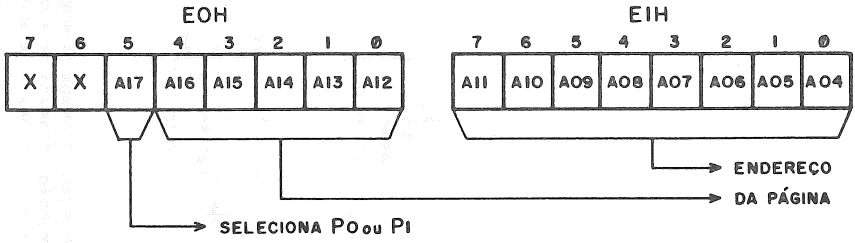
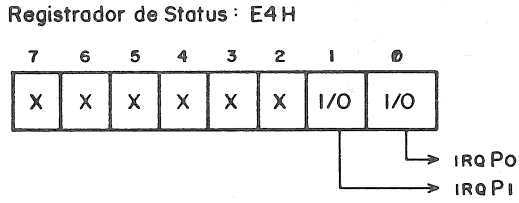


Figura 5- Registradores de Endereçamento.



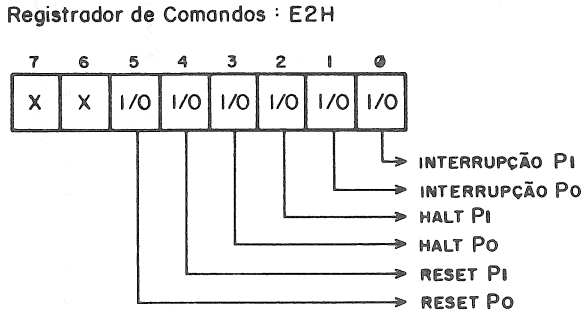


Figura 6 - Lay-out dos Registradores de Comandos e Status do PES.