AVALIAÇAO EXPERIMENTAL DO DESEMPENHO DO

SISTEMA MULTIPROCESSADOR ACP

Carla Osthoff Ferreira de Barros

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇAO DOS PROGRAMAS DE POS-GRADUAÇAO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSARIOS PARA A OBTENÇAO DO GRAU DE MESTRE EM CIENCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇAO.

Aprovada por:

Prof.Claudio Luis de Amorim, Ph.D.
(Presidente)

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. Julio Salek Aude, Ph.D.

Prof.Alberto Franco de Sá Santoro, Ph.D.

RIO DE JANEIRO, RJ-BRASIL
ABRIL DE 1989

Barros,CARLA Osthoff Ferreira de

Avaliação Experimental do Desempenho do Sistema Multiprocessador ACP [Rio de Janeiro]1989

X,173p.29,7 cm (COPPE/UFRJ,M.Sc,Engenharia de Sistemas, 1989)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Arquitetura de Computadores, Estudo de Desempenho.

Agradecimentos

A meus orientadores, Prof. Claudio Amorim e Prof. Valmir Barbosa, que por toda a sua experiência, questionamento e incentivo tornaram possivel a realização deste trabalho.

A Alberto Santoro, que com sua garra e dedicação, possibilitou não só ao desenvolvimento do Sistema ACP em colaboração com o CBPF, como também ao nascimento de um novo grupo de Fisica Experimental de Altas Energias.

A Thomas Nash, sem o qual o ADVANCED COMPUTER PROGRAM seria apenas mais um projeto de papel, por toda a confiança que me tem dado.

A Joe Biel, que me ajudou a formular os fundamentos desta pesquisa, e por todo o apoio não só a mim como a toda a equipe do CBPF.

A Moacyr Souza, que com toda a sua experiência e paciência, tanto me ajudou.

A Roberto Valois, pelas discussões que me levaram e entender melhor as minhas questões.

A equipe do Laborátorio de Fisica Experimental de Altas Energias do CBPF; Ignacio, Arthur, Gilvan, Guilherme, Henriette, Mario Vaz, Sandra, Vladimir e outros. Por seu apoio e compreensão durante este periodo.

A meus pais e meus irmãos que tanto amo.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de mestre em Ciências (M.Sc).

AVALIAÇAO EXPERIMENTAL DO DESEMPENHO DO SISTEMA MULTIPROCESSADOR ACP

Carla Osthoff Ferreira de Barros

Abril,1989

Orientador: Prof. Claudio Luis de Amorim

Orientador: Prof. Valmir Carneiro Barbosa

Programa: Engenharia de Sistemas e Computação

Neste trabalho realizamos uma avaliação experimental do desempenho do Sistema Multiprocassador ACP. Num dos ambientes, tarefas são independentes e não se comunicam entre si. Noutro ambiente, as tarefas simulam inter-dependências através de transmissão de mensagens. Os parâmetros de desempenho são baseados nas medidas de "DEGRADAÇAO" e de ACELERAÇAO do sistema para variações do tempo de processamento, do tempo de transmissão, e do número de processadores do Sistema ACP. Baseado neste estudo são fornecidas sugestões para o desenvolvimento de um modelo analitico para o Sistema ACP.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master Science (M.Sc).

A Experimental Performance Evaluation

of the ACP Multiprocessor System

Carla Osthoff Ferreira de Barros

April,1989

Thesis Supervisor: Prof. Claudio Luis de Amorim

Thesis Supervisor: Prof. Valmir Carneiro Barbosa

Department:Systems and Computing Engineering

This work is a experimental performance evaluation of the ACP Multiprocessor System. In an ambient tasks are independent and there is no communication, in another ambient messages are sent to simulate dependence between tasks. The performance parameters are based on the System "OVERHEAD" and "SPEED-UP" which are evaluated by varying processing time, communication time and number of processors. Based on this study we give some suggestions for a ACP Multiprocessor System analytical model.

AVALIAÇAO EXPERIMENTAL DO DESEMPENHO DO SISTEMA MULTIPROCESSADOR ACP

CAPITULO I

INTRODUÇAO

Supercomputadores tem sido definidos como os computadores mais potentes disponiveis em cada época. Essa potência computacional pode ser expressa quantitativamente em termos de capacidade de realização de operações aritméticas por unidade de tempo, da capacidade de memória e da precisão na realização dos cálculos.

Existem atualmente aplicações que requerem máquinas 100 vezes mais rápidas que os atuais **Supercomputadores**.[27]

Assim como em outras áreas de pesquisa, a necessidade computacional em **Fisica de Altas Energias** tem crescido tanto que computadores convencionais não são mais capazes de satisfazê-la. Uma única experiência é capaz de produzir centenas de milhões de eventos, onde cada evento é composto em média por um conjunto de 5Kbytes de dados e necessita para o seu processamento de alguns segundos do tempo de uma Unidade Central de Processamento(UCP). O **Fermi National Laboratory**" criou o **Advanced Computer Program** em colaboração com outras instituições de pesquisa em **Fisica de Altas Energias**, entre elas o CBPF, para o desenvolvimento de computadores capazes de suprir esta necessidade.

Podemos dividir em duas, as linhas de pesquisa para se aumentar o desempenho dos computadores:

O primeiro está em aumentar a velocidade com que uma instrução pode ser executada através do uso de novas tecnologias, tais como Arseneto de Galio, filmes Supercondutores e optoeletrônica. Espera-se que, com as novas tecnologias, se possa atingir um grande fator de redução dos atuais ciclos de máquina, ou seja uma minimização do atraso dos circuitos dos computadores, onde um ciclo de máquina é definido como sendo o tempo necessário para executar a menor fase de processamento de uma instrução.

O segundo caminho está em aumentar o poder computacional introduzindo elementos de multiprocessamento, de forma a termos múltiplos processadores com memória compartilhada e periféricos sob o controle de um sistema operacional integrado, isto é, um Sistema Multiprocessador.

O ADVANCED COMPUTER PROGRAM segue a linha que pesquisa o desenvolvimento de Sistemas Multiprocessadores. O Multiprocessador ACP, primeiro projeto do grupo ADVANCED COMPUTER PROGRAM teve como finalidade satisfazer problemas do tipo "orientado para eventos". Neste tipo de problema, um mesmo processo é executado em dados diferentes centenas de milhões de vêzes, sendo cada resultado de processamento independente dos resultados anteriores. Este tipo de processo é caracterizado por:

1. Necessitar de muito tempo de UCP (Unidade Central de Processamento)

2. Realizar poucas operações de "entrada/saida "

3. Requerer pouca ou nenhuma interação entre os processadores [1]

O objetivo desta tese é o estudo experimental do desempenho do Sistema Multiprocessador Paralelo ACP.

Neste trabalho, após uma rápida descrição das arquiteturas paralelas existentes e de suas classificações (CAPITULO II) é apresentado um estudo do SISTEMA MULTIPROCESSADOR PARALELO ACP (CAPITULO III).

Com base neste estudo é apresentado uma proposta de avaliação experimental do desempenho do SISTEMA ACP para o processamento de problemas para os quais a máquina foi projetada, isto é, para problemas do tipo "orientado para eventos" (CAPITULO IV.I).

Em um segundo passo será avaliado o desempenho da máquina em aplicações onde o processamento de cada processador é uma tarefa que depende do processamento dos processadores vizinhos, isto é, em problemas onde existe comunicação entre os processadores, e cada processador processa uma série de subtarefas que fazem parte de uma única tarefa global (CAPITULO IV.3).

Com base nos resultados dos estudos realizados são fornecidas sugestões para o desenvolvimento de um modelo analitico para o Sistema ACP (CAPITULO IV.2)

Por fim, apresentaremos as conclusões do estudo realizado (CAPITULO V).

CAPITULO II

APRESENTAÇÃO DAS ARQUITETURAS PARALELAS EXISTENTES E DE SUAS CLASSIFICAÇÕES

O termo confluência é muito empregado em Ciência de Computação. Ele se refere a existência de diversas atividades simultâneas ou paralelas que convergem para uma solução conjunta, única. Podemos definir processamento paralelo como sendo uma forma de processamento que procura explorar nos processos a computação de eventos confluentes.

Podemos dividir a implementação do processamento paralelo em 4 niveis [25]:

1. Em um primeiro nivel, um maior grau de processamento paralelo é implementado entre os múltiplos processos através da multiprogramação (seçãoII.1.5), do tempo compartilhado (seçãoII.1.6) e do multiprocessamento (seçãoII.1.7).

2. O segundo nivel é o paralelismo conduzido entre processos ou tarefas (segmentos de um programa) dentro de um mesmo programa. Isto exige a decomposição de um programa em múltiplas tarefas.

3. O terceiro nivel é o estabelecer o paralelismo entre instruções, assim como quebrar sequências como em "laços de instrução "(loop), para revelar o paralelismo entre as instruções.

4. O quarto nivel é o da intrainstrução, que é implementado por meios de hardware. Como exemplo temos o caso do processamento vetorial onde a Unidade Lógica e Aritimética (ULA),é composta por múltiplas unidades funcionais que possibilitam a execução de operações simultâneas em todos os componentes do vetor.

Iremos definir "granularidade" como o nivel de paralelismo de um programa. Podemos determinar que um programa com granularidade fina é um programa cujo grau de paralelismo está entre os niveis 3 e 4; e um programa com granularidade grossa é um programa cujo grau de paralelismo está entre os niveis 1 e 2.

II.1 - IMPLEMENTAÇÃO DO PARALELISMO EM SISTEMAS UNIPROCESSADORES

Um sistema de computação convencional , também conhecido por máquina de Von-Neuman, consiste de cinco unidades: ENTRADA, MEMORIA, UNIDADE LOGICO-ARITMETICA,CONTROLE E SAIDA, relacionadas entre

si segundo a figura2.1[27].

As funções dessas unidades são:

1. ENTRADA; que transmite dados e instruções do ambiente exterior para a memória.

2. MEMORIA; que armazena instruções, dados e resultados intermediários e finais.

3. UNIDADE LOGICO-ARITMETICA; que executa as operações lógico-aritméticas.

4. CONTROLE; que interpreta as instruções e providencia sua execução.

5. SAIDA; que transmite os resultados finais para o ambiente exterior.

Um processador que segue o paradigma de Von-Neumam [27], executa instruções sequencialmente, de acordo com o ciclo mostrado na figura 2.2.

O processamento paralelo pode ser implementado na estrutura acima citada por diversas maneiras:

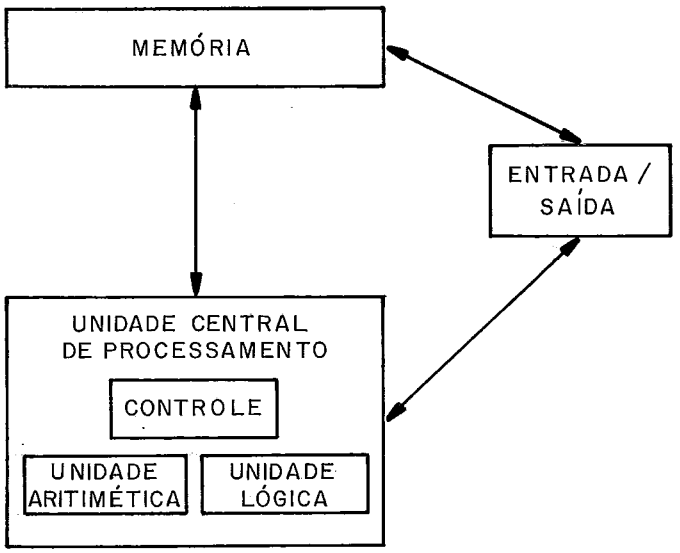


FIG. 2.1 — MÁQUINA DE VON-NEUMAN

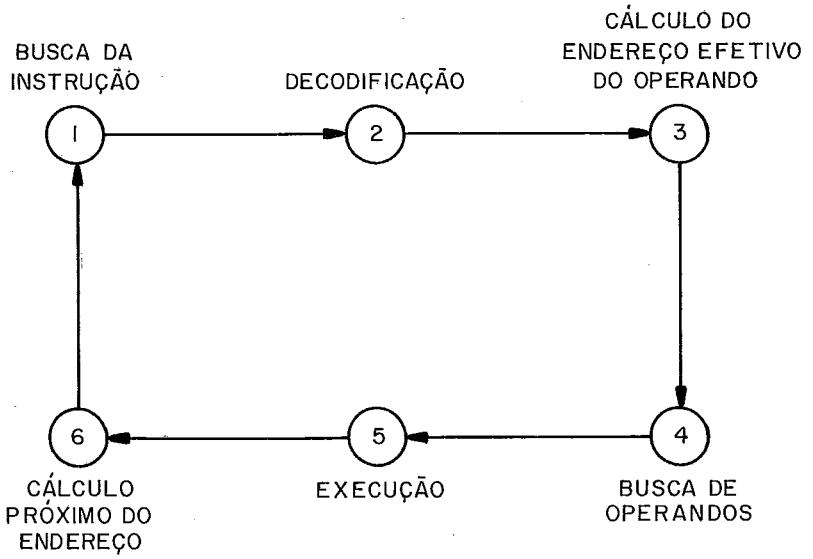


FIG. 2.2 — CICLO DA EXECUÇÃO DE UMA INSTRUÇÃO DE UM PROCESSADOR VON-NEUMAN

II.1.1 - Multiplicidade de Unidades Funcionais

Os primeiros computadores possuiam apenas uma ULA em sua cpu. Desta maneira a ULA podia executar apenas uma instrução por vez. Na prática muitas das funções da ULA podem ser distribuidas em múltiplas unidades funcionais especializadas que podem operar em paralelo.

II.1.2 - Paralelismo e "Pipeline" dentro da Unidade Central de Processamento

Atualmente são construidos em quase todas as Unidades Centrais de Processamento, ou UCPs somadores paralelos utilizando a técnica de somadores de alta velocidade, ("carry lookahead"), técnicas de multiplicadores de alta velocidade e de divisão de convergência para as funções de multiplicação e de divisão.

Várias fases da execução de uma instrução são feitas em "pipeline" , incluindo busca de instrução , decodificação, busca de operandos, execução na unidade lógica e aritmética e armazenamento de resultados.

II.1.3 - Alternância entre as Operações de UNIDADE CENTRAL DE PROCESSAMENTO e as de Entrada e de Saida

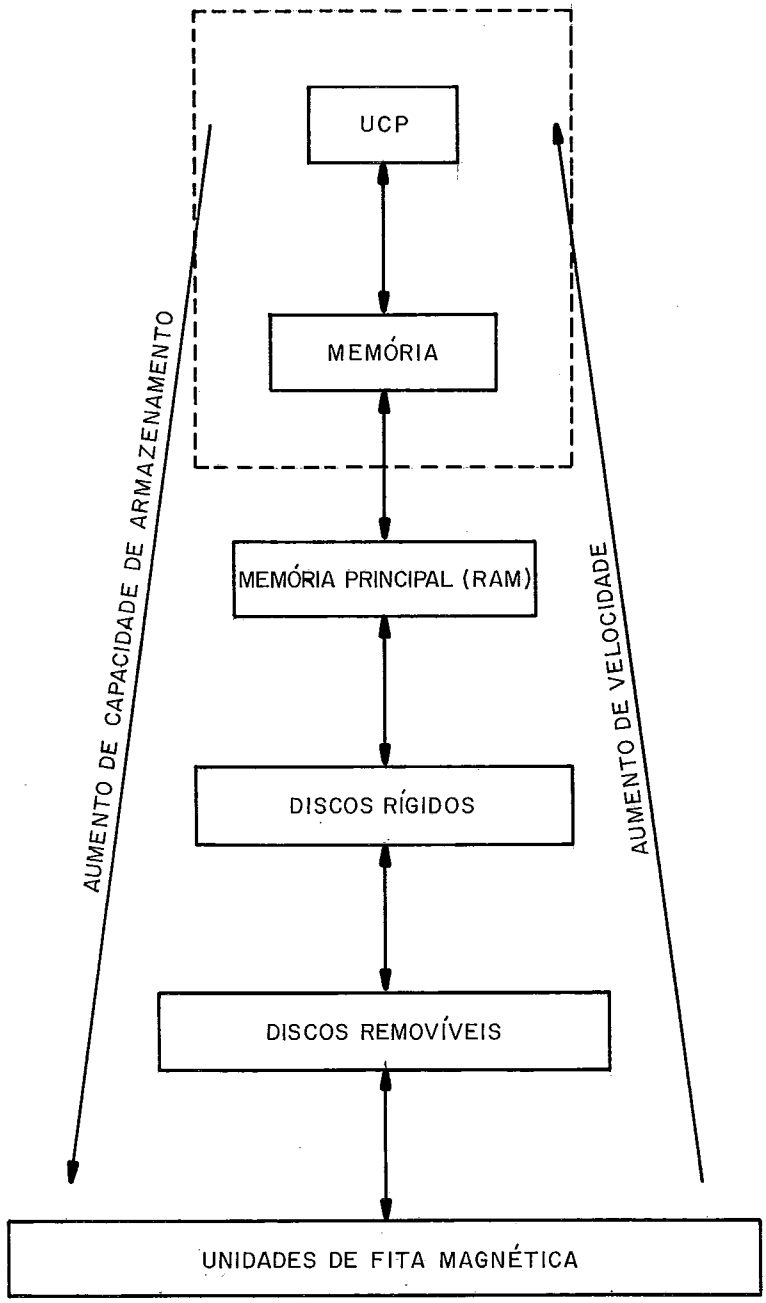


FIG. 2.3 – IMPLEMENTAÇÃO DE UM SISTEMA DE MEMÓRIA HIERÁRQUICO ONDE O NÍVEL MAIS ALTO É O DOS REGISTROS E O NÍVEL MAIS BAIXO SÃO AS UNIDADES DE FITA MAGNÉTICAS

Operações de entrada e de saída (E/S), podem ser realizadas simultâneamente com operações de UCP com a utilização de controladores de E/S, de canais e de processadores de E/S separados. Para possibilitar a transferência direta de dados entre a memória e componentes de E/S utilizando módulos de " Direct Memory Access " (DMA) que funciona no modo " roubo de ciclo de UCP ".

II.1.4 - Utilização de um Sistema de Memória Hierárquico

A existência de um caminho de dados entre o processador e a memória, representado pelas fases de busca de instrução e operandos na memória constitui-se no aspecto mais restritivo do desempenho de processadores convencionais. Essa restrição advém do fato que o ciclo de acesso a memória é muito mais lento que um ciclo do processamento. Este aspecto dos processadores convencionais tornou-se conhecido como o "gargalo de Von-Neuman" , devido ao seu papel limitador no desempenho computacional.

Uma maneira de se contornar este problema está na implementação de um sistema hierárquico de memória(fig. 2.3), onde o nivel mais alto é o dos registros que são diretamente endereçados pela Unidade Lógica e Aritmética, ULA . A seguir uma memória mais rápida que a memória real, uma "memória cache" pode ser utilizada para armazenar

dados mais usados pela UCP. Uma memória virtual pode ser implementada no nivel mais baixo ao serem utilizados memória de massa, discos e unidades de fita.

II.1.5 - Multiprogramação

Em um computador podem existir dentro de um mesmo intervalo de tempo múltiplos processos ativos competindo pela memória, pelas unidades de E/S e pelos recursos da UCP.

A execução simultânea de vários tipos de programas no computador é possivel de ser realizada pelo fato de alguns programas serem CPU-bound (de computação intensiva) e de outros processos serem I/O- bound (E/S intensiva), ocorrendo execuções simultâneas de operações ou de CPU ou de E/S para programas diferentes.

II.1.6 - Tempo Compartilhado

A multiprogramação em um sistema uniprocessador é centrada em torno da alternância da alocação da UCP entre vários programas. Para evitar que um programa com maior prioridade ocupe a UCP por muito tempo é implementado no sistema operacional o conceito de tempo compartilhado. O conceito é uma extensão da multi-programação onde são fornecidas cotas fixas de tempo para múltiplos programas.

II.1.7 - Multiprocessamento

Na multiprogramação os processos compartilham o único processador da máquina. Neste caso, a simultaneidade sómente ocorre quando várias operações de E/S estão sendo executadas pelas unidades de E/S, ao mesmo tempo que um outro tipo de instrução está sendo executado pela UCP.

Em multiprocessamento, os processos são executados em paralelo por uma rede de processadores fortemente acoplados, já em processamento distribuído os processos ficam associados a processadores fracamente conectados por uma rede de comunicação.

Convém observar que as formas de concorrência não são exclusivas, pois é possível existir multiprogramação nos processadores de máquinas que suportam multiprocessamento bem como é possível existir sistemas de processamento distribuído cujos processadores são multiprocessadores e etc.

II.2 - CLASSIFICAÇÕES DAS ESTRUTURAS DE COMPUTAÇÃO PARALELA

Computadores paralelos são definidos como como computadores que "enfatizam o processamento paralelo". Baseado nesta definição foram implementadas várias classificações das arquiteturas dos computadores paralelos existentes.

Apresentaremos três classificações:

A primeira a ser apresentada é a classificação feita por K.Hwang e F. Briggs [25]. Esta classificação divide os computadores paralelos em quatro grupos de arquiteturas: Computadores Pipeline , Array Processor , Sistemas Multiprocessadores e Dataflow .

A segunda a ser apresentada é a classificação de Flynn [25]. Esta classificação se baseia na multiplicidade em sistemas computationais de sequência de instruções (instruction stream) e de sequências de dados (data stream).

A terceira classificação a ser apresentada é a classificação de Hockney[31]. Esta classificação foi feita como consequência do aparecimento de grande variedade de computadores paralelos MIMD(classificação de FLYNN) durante a década de 80. De maneira que esta classificação pode ser considerada como uma complementação da classificação de FLYNN.

Existem ainda outras classificações tais como a de Feng [25] implementada em 1972 que é baseada na comparação entre processamento paralelo e processamento serial, e a classificação de Handler[25] implementada em 1977 e que determina o grau de paralelismo e de pipeline em vários niveis do sistema, mas que não são de interesse para o nosso estudo.

II.2.1 - CLASSIFICAÇÃO DE K.HWANG/F.BRIGGS

Esta classificação divide os computadores paralelos em quatro grupos de arquiteturas:

II.2.1.1 - Computadores Pipeline

II.2.1.2 - Array Computer

II.2.1.3 - Sistemas de Multiprocessadores

II.2.1.4 - DATAFLOW

II.2.1.1 - Computadores Pipeline

Pipelining tem sido a técnica de paralelismo mais utilizada para aumentar o desempenho dos sistemas de computação [25]. Pipeline é uma estrutura que consiste de uma sequência de estágios, através dos quais uma computação flui com a propriedade de que a cada ciclo uma nova computação possa ser iniciada no primeiro

estágio de pipeline, enquanto outras operações evoluem para os estágios seguintes no pipeline.

Normalmente a execução de uma instrução em um processador requer quatro passos principais:

1. Busca da instrução
2. Decodificação da instrução
3. Busca do operando
4. Execução da operação

Um computador sem pipeline necessita de quatro ciclos de pipeline para executar uma instrução.

Em um computador com pipelining de instruções, a execução de uma sequência de instruções pode ser pipelined através da superposição (i.e., concorrência) das fases do ciclo de execução de instrução. O pipeline funciona de forma que enquanto uma instrução está no passo um de execução, uma segunda instrução está no passo dois, uma terceira no três e uma quarta instrução no passo quatro. Isto é, o paralelismo existente no processamento das instruções ocorre a nível das fases do ciclo de execução.

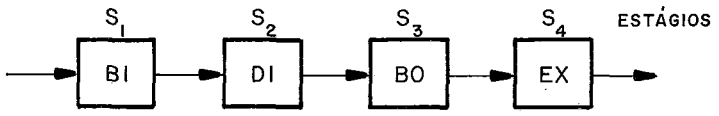
No computador pipeline apresentado na figura 2.4, uma vez inicializado o carregamento das instruções nos estágios de pipeline um resultado de saída é produzido a cada ciclo de pipeline , reduzindo o ciclo original a um quarto. Em um computador sem pipeline , estes quatro passos devem ser completados antes da operação da próxima instrução.

Podemos ter também pipelining de processadores, onde uma sequência de dados é processada por uma cascata de processadores (i.e., vários processadores conectados em linha), na qual cada processador executa uma tarefa especifica sobre os dados.

Com o intuito de aumentar o desempenho, sistemas de computação apresentam pipelining em diversos niveis. No nivel mais alto, existe o pipelining no processamento das instruções. Depois, no segundo nivel, cada uma das fases do processamento das instruções é executada de forma pipelined. Finalmente, num terceiro nivel, encontra-se a execução pipelined das operações aritméticas [27].

II.2.1.2 - "Array Computers"

Um array processor é um computador paralelo



PROCESSADOR "PIPELINE"

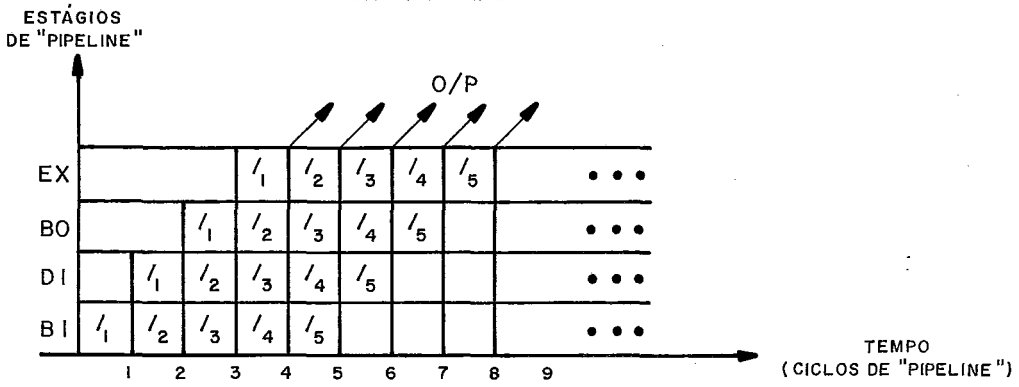


DIAGRAMA DE ESPAÇO/TEMPO
DE UM PROCESSADOR "PIPELINE" DE 3 ESTÁGIOS

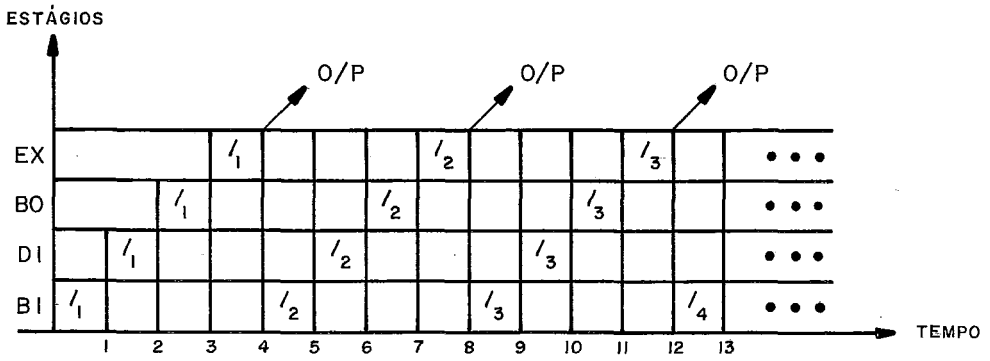


DIAGRAMA DE ESPAÇO/TEMPO
DE UM PROCESSADOR SEM "PIPELINE"

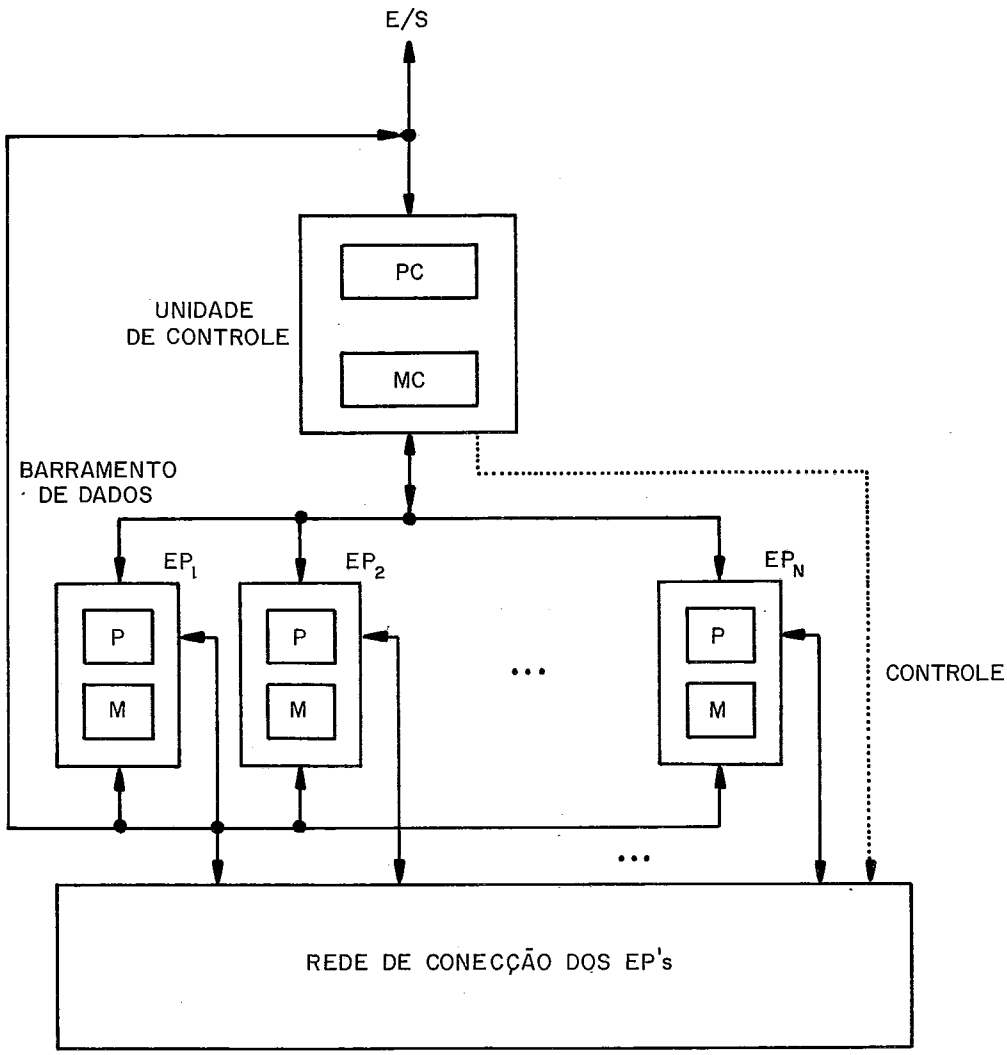
EX – EXECUÇÃO DA OPERAÇÃO
BO – BUSCA DO OPERANDO
DI – DECODIFICAÇÃO DA INSTRUÇÃO
BI – BUSCA DA INSTRUÇÃO

FIG. 2.4 – EXEMPLO DE FUNCIONAMENTO DE UM COMPUTADOR "PIPELINE"

síncrono com múltiplas unidades lógicas aritiméticas chamadas de elementos de processamento (EP),que podem operar em paralelo de uma maneira síncrona [25]. Os EP são sincronizados de maneira a executarem a mesma instrução ao mesmo tempo. Um típico array processor é apresentado na fig 2.5 (a) onde uma unidade de controle dirige as instruções de controle e escalares. Os EPs são unidades passivas sem capacidade de decodificação da instrução.

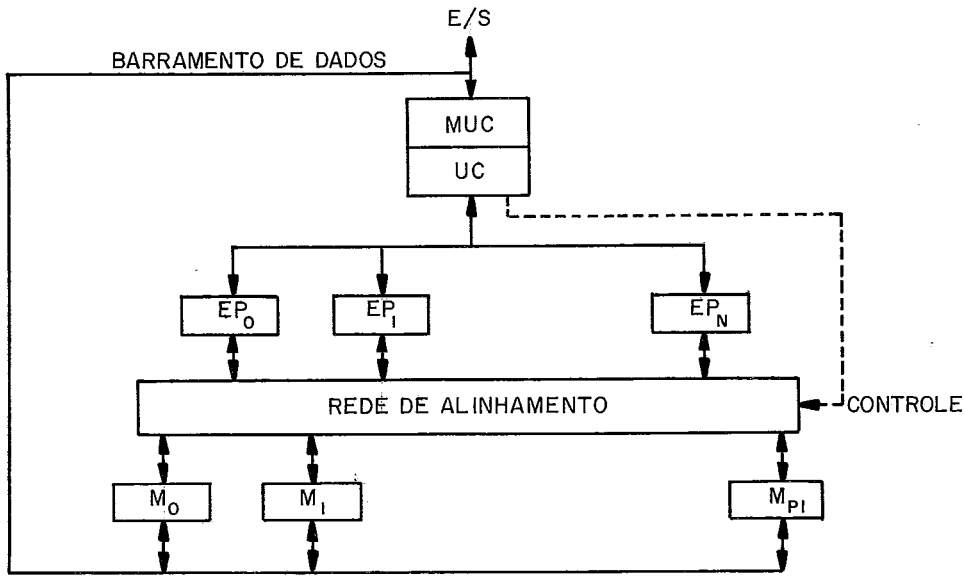
Um array processor é geralmente interfaceado a um computador hospedeiro através da unidade de controle. O computador hospedeiro é geralmente um computador de uso geral que serve como gerenciador de operação" do sistema.Dentre as funções do computador hospedeiro consta a de alocação de recursos e de supervisão de periféricos e de unidades de E/S. A unidade de controle do array processor supervisiona diretamente a execução dos programas, enquanto que o hospedeiro executa as funções de E/S entre o sistema e o mundo externo.

A motivação original para o desenvolvimento de array procesor SIMD (apresentado na secção II.2.2) foi a de realizar a computação paralela



EP – ELEMENTOS DE PROCESSAMENTO
PC – PROCESSADOR DE CONTROLE
CM – MEMÓRIA DE CONTROLE
P – PROCESSADOR
M – MEMÓRIA

FIG. 2.5.a – DIAGRAMA DE UM "ARRAY PROCESSOR" COM REDE DE CONECÇÃO ENTRE OS ELEMENTOS DE PROCESSAMENTO



EP — ELEMENTO DE PROCESSAMENTO

M — MEMÓRIA

UC — UNIDADE DE CONTROLE

MUC — MEMÓRIA DA UNIDADE DE CONTROLE

FIG. 2.5.b — DIAGRAMA DE UM ARRAY PROCESSOR COM REDE DE CHAVEAMENTO ENTRE OS ELEMENTOS DE PROCESSAMENTO E OS MÓDULOS DE MEMÓRIA

em dados do tipo vetoriais e matriciais. Algoritimos de processamento paralelo foram desenvolvidos por muitos pesquisadores de máquinas SIMD. Tais algoritimos são importantes na realização do cálculo de multiplicação de matrizes, Fast Fourier Transform (FFT), transposição de matrizes, somatório de elementos vetoriais, inversão de matrizes, recorrência linear, operações booleanas de matrizes e equações diferenciais parciais.

Geralmente um array processor assume uma das duas configurações apresentadas na fig 2.5

II.2.1.2.1 - Array Processor com Memória Local

A figura 2.5 (a) apresenta uma configuração contendo N EPs, sincronizados sob o controle de uma unidade de controle. Cada EP é essencialmente uma unidade lógica aritimética conectada a registros e contendo uma memória local para o armazenamento dos dados distribuidos. A unidade de controle também possui uma memória própria para o armazenamento dos programas.

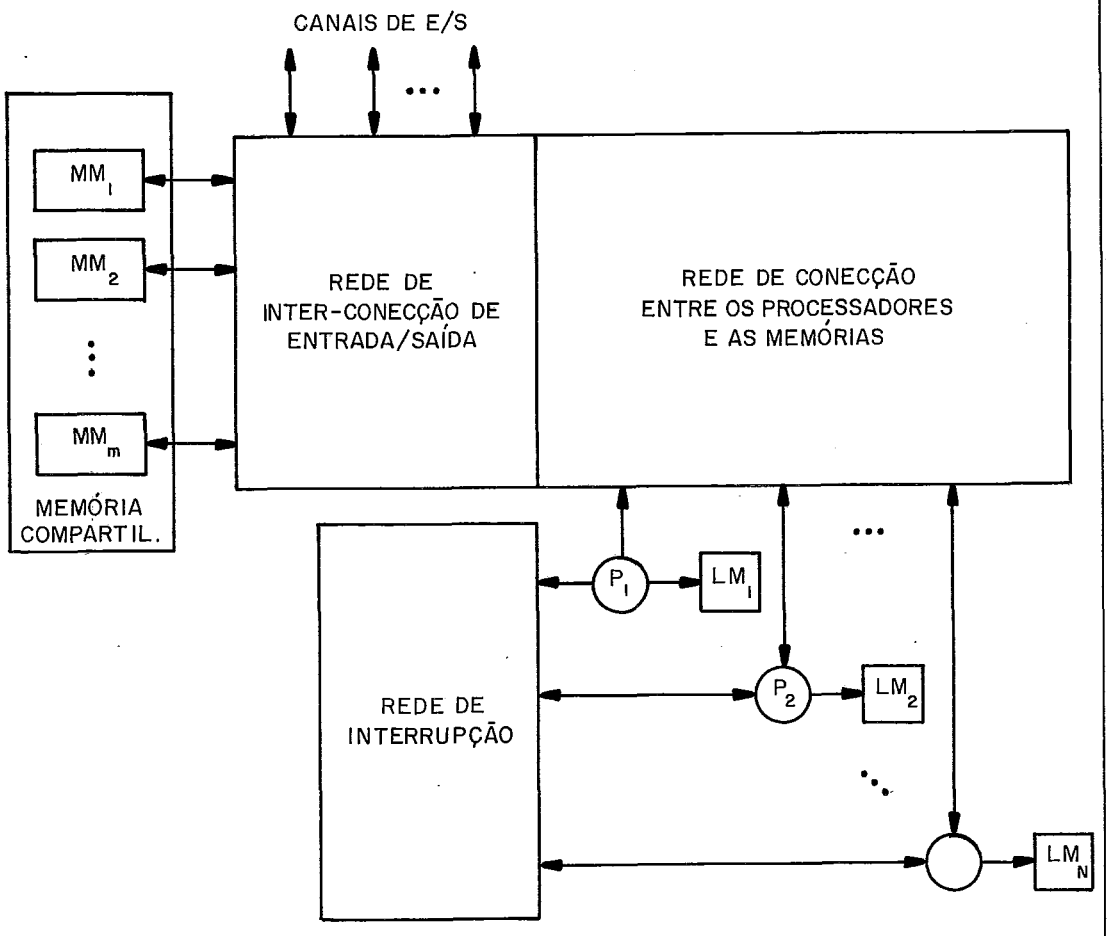
II.2.1.2.2 - Array Processor com Memória Compartilhada

Uma outra configuração que difere da figura 2.5 (a) é apresentada na figura 2.5 (b) ; as memórias locais acopladas a cada EPs são substituídos por módulos paralelos de memória compartilhados por todos os EPs através de uma rede de alinhamento que chaveia as conexões entre os EPs e as memórias.

II.2.1.3 - Sistemas Multiprocessadores

A diferença fundamental entre um sistema array processor e um sistema multiprocessador é que em um array processor os elementos operam de uma maneira sincrona enquanto que um sistema multiprocessador opera de uma maneira assíncrona.

Uma organização básica de um sistema multiprocessador é apresentada na fig 2.6. O sistema contèm dois ou mais processadores de aproximadamente mesma capacidade. Os processadores compartilham o acesso a uma série de módulos de memória, canais de E/S e unidades



MM – MÓDULO DE MEMÓRIA
ML – MEMÓRIA LOCAL
P – PROCESSADOR

FIG. 2.6 – DIAGRAMA DE UM SISTEMA MULTIPROCESSADOR

periféricas. O mais importante é que o sistema global deve ser controlado por um sistema operacional único integrado que possibilita a interação entre os processos e seus programas em vários niveis. Independentemente da memória compartilhada e das unidades de E/S, cada processador possui sua memória local e unidades privadas. A comunicação entre os processos pode ser feita através de uma memória compartilhada ou por uma rede de interconexão.

A organização do hardware de sistemas multiprocessadores é determinada primeiramente pela interconexão estrutural a ser utilizada entre as memórias e os processadores (e entre memórias e canais de e/s caso necessário).

De uma maneira simplificada podemos caracterizar multiprocessadores por dois atributos: em primeiro lugar um multiprocessador é um computador que inclui múltiplos processadores e em segundo lugar os processadores podem se comunicar e cooperar em diferentes niveis para solucionar um determinado problema. A comunicação pode ocorrer seja pelo envio de mensagens entre os processadores, seja pelo compartilhamento de uma memória comum.

Existem algumas similaridades entre sistemas multiprocessadores e sistemas de multicomputadores, já que ambos são motivados pela mesma finalidade que é a de dar suporte para operações concorrentes no sistema. Entretanto existem distinções importantes entre um sistema de computação múltipla e um sistema multiprocessador baseado na implementação do compartilhamento de recursos e no grau de cooperação para a solução de um problema.

Um sistema de computação múltiplo consiste em vários computadores autônomos que podem ou não se comunicar. Um sistema multiprocessador é controlado por um sistema operacional que permite uma interação entre os processadores e seus processos a nivel dos dados e dos conjuntos de dados.
Podemos classificar as arquiteturas de multiprocessadores como sendo fortemente ou fracamente acopladas.

II.2.1.3.1 - ARQUITETURAS DE MULTIPROCESSADORES FORTEMENTE ACOPLADOS

Em uma arquitetura fortemente acoplada (fig 2.11) a comunicação é realizada por uma memória compartilhada, de maneira que a taxa com que um processador pode tranferir um dado depende da taxa de transferência da memória. Neste tipo de arquitetura, os processadores podem apresentar uma memória local de rápido acesso.

Este tipo de conexão pode ser implementado através da inserção de uma rede de interconexão entre os processadores e a memória, ou através de uma memória multiportas. Um dos fatores que limita a expansão do número de processadores nos sistemas fortemente acoplados é a degradação do desempenho devido a contenção de memória que ocorre quando dois ou mais processadores tentam acessar a memória em uma mesma unidade de tempo. O grau de conflito pode ser reduzido com o aumento de intercalagem de memória.

II.2.1.3.2 - ARQUITETURAS DE MULTIPROCESSADORES FRACAMENTE ACOPLADOS

Sistemas fracamente acoplados (fig 2.12) geralmente não apresentam o mesmo grau de

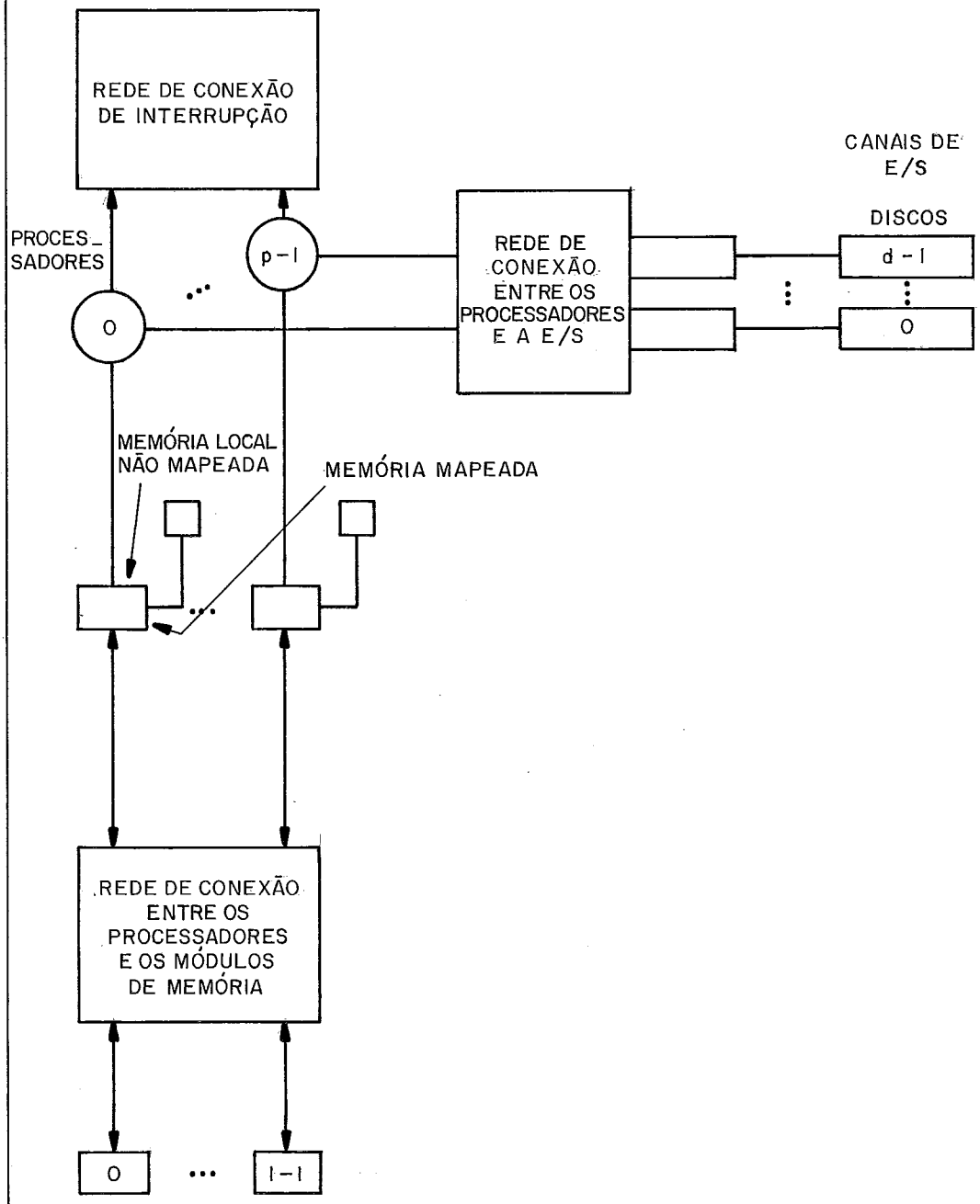


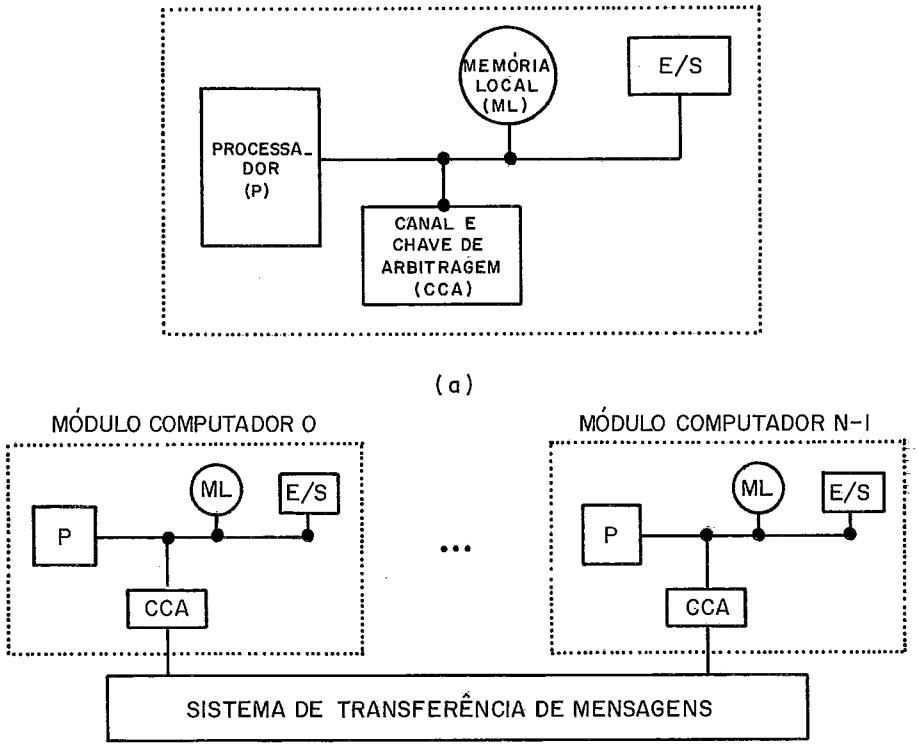
FIG. 2.11 — ARQUITETURA DE UM SISTEMA MULTIPROCESSADOR FORTEMENTE ACOPLADO

conflito presente em sistemas fortemente acoplados. Em tais sistemas cada processador possui uma série de unidades de E/S e a maior parte das instruções e dos dados são acessados em uma memória local. Iremos chamar de um módulo de computador por um sistema composto pelo processador pela sua memória local e pelas interfaces de E/S.

Processos executados em módulos de computadores diferentes se comunicam através de troca de mensagens. O grau de acoplamento em tal sistema é muito fraco, de maneira que é por vezes chamado de sistema distribuido.

Sistemas fracamente acoplados são normalmente eficientes quando as interações entre as tarefas são curtas. Sistema fortemente acoplados podem tolerar um alto grau de interação entre as tarefas sem uma significativa deteriorização do desempenho.

II.2.1.4 - "Dataflow"



(a) UM MÓDULO COMPUTADOR
(b) MÓDULOS DE COMPUTADORES FORMANDO UM SISTEMA MULTIPROCESSADOR FRANCAMENTE ACOPLADO

FIG. 2.12 – ARQUITETURA DE UM SISTEMA MULTIPROCESSADOR FRANCAMENTE ACOPLADO

Existe também o conceito de "dataflow" cujo principio básico é o de permitir a execução de uma operação sempre que operandos se tornam disponiveis de maneira que os "contadores de programa" tornam-se denecessários para direcionar a computação. A iniciação de uma instrução depende da disponibilidade dos dados e é independente da localização física da instrução no programa. Ou seja, as instruções no programa não são ordenadas, e a execução obedece o fluxo dos dados (figura 2.7).

Teóricamente uma concorrência máxima pode ser explorada nesta estrutura sendo limitada apenas pela disponibilidade dos recursos de hardware.

Implementações de processamento paralelo em cada um dos niveis citados não são mutuamente exclusivas. De fato, a maior parte dos computadores atualmente existentes são pipeline e alguns deles assumem tambèm estruturas de array ou de multiprocessadores.

II.2.2 - ClASSIFICAÇÃO DE FLYNN

II.2.2.1 - "SINGLE INSTRUCTION STREAM SINGLE DATA STREAM" (SISD)

Arquiteturas que se baseiam na execução de uma sequência de instruções para uma sequência

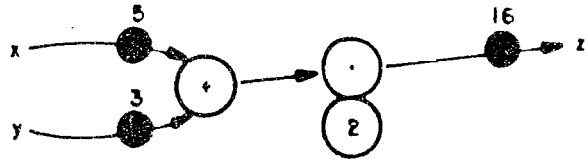
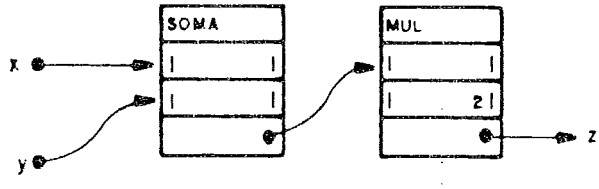
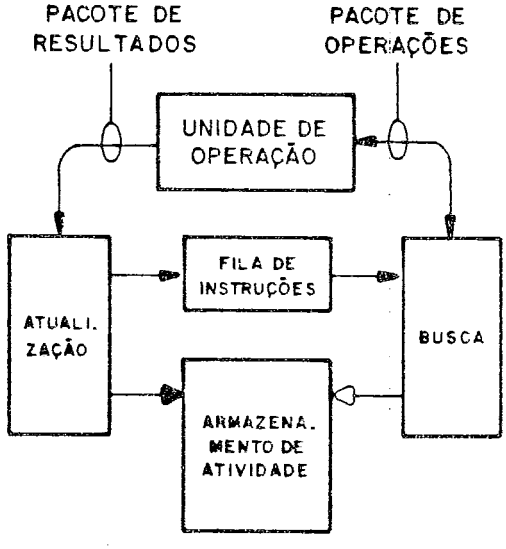


GRÁFICO DE UM PROGRAMA
"DATA FLOW"



IMPLEMENTAÇÃO DO "DATA FLOW"
EM TABELAS



MECANISMO BÁSICO DO
"DATA FLOW"

——— CONECÇÕES PARA TRANSFERÊNCIAS DE MENSAGENS
———● ACESSO À LEITURA/ESCRITA
———○ ACESSO À LEITURA

FIG. 2.7 – CONCEITO DE "DATA FLOW"

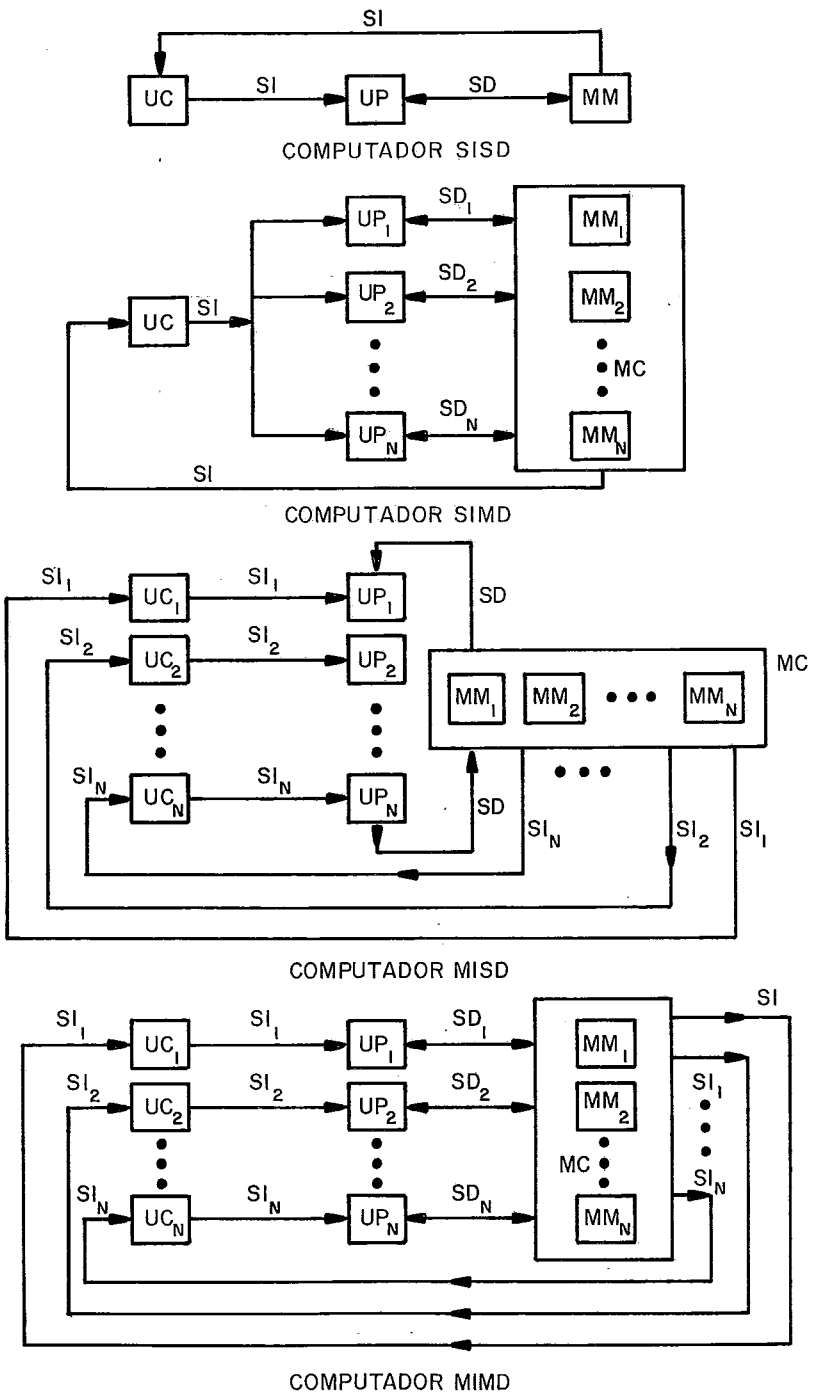
de dados. Esta classificação apresentada na fig 2.8 (a) representa a maioria dos computadores seriais atualmente existentes. As instruções são executadas sequêncialmente podendo ser sobrepostas no estágio de execução. A maioria dos computadores SISD utilizam pipeline . Um computador SISD pode ter mais de uma unidade funcional sob o controle de uma unidade de controle.

II.2.2.2 - "SINGLE INSTRUCTION STREAM MULTIPLE DATA STREAM" (SIMD)

Arquiteturas que se baseiam na execução de uma sequência de instruções para múltiplas sequências de dados. Esta é a classe correspondente aos array processors acima citados (fig 2.8 (b)) onde múltiplos elementos processadores estão sob a supervisão de uma única unidade de controle. Todos os EP recebem a mesma instrução da unidade de controle.

II.2.2.3 - "MULTIPLE INSTRUCTION STREAM SINGLE DATA STREAM" (MISD)

Arquiteturas que se baseiam na execução de múltiplas sequências de instruções para apenas uma sequência de dados. Neste modo de processamento cada um dos n processadores recebe instruções distintas para operar sobre o



UC – UNIDADE DE CONTROLE
UP – UNIDADE DE PROCESSAMENTO
MM – MÓDULO DE MEMÓRIA
SI – SEQUÊNCIA DE INSTRUÇÕES
SD – SEQUÊNCIA DE DADOS
MC – MEMÓRIA COMPARTILHADA

FIG. 2.8 – CLASSIFICAÇÃO DE FLYNN

mesmo data stream (figura 2.8 (c)). Este tipo de estrutura tem recebido muito pouca atenção, sendo considerada impraticável por pesquisadores.

II.2.2.4 - "MULTIPLE INSTRUCTION STREAM MULTIPLE DATA STREAM" (MIMD)

Arquiteturas que se baseiam na execução de múltiplas sequências de instruções para múltiplas sequências de dados. A maior parte dos sistemas multiprocessadores podem ser classificadas nesta categoria. Um computador MIMD intrínseco apresenta interações entre os n processadores onde os dados são obtidos da mesma região de memória, compartilhada por todos os N processadores, (figura 2.8 (d)).

II.2.3 - CLASSIFICAÇAO DE HOCKNEY

Durante a década de 80, surgiu uma grande variedade de computadores paralelos MIMD. Para analisá-la, Hockney [31] propos uma classificação estrutural da classe MIMD em três níveis básicos.

Em um primeiro nivel podemos dividir os sistemas multiprocessadores em dois grupos : sistemas chaveados e sistemas em rede (fig 2.9).

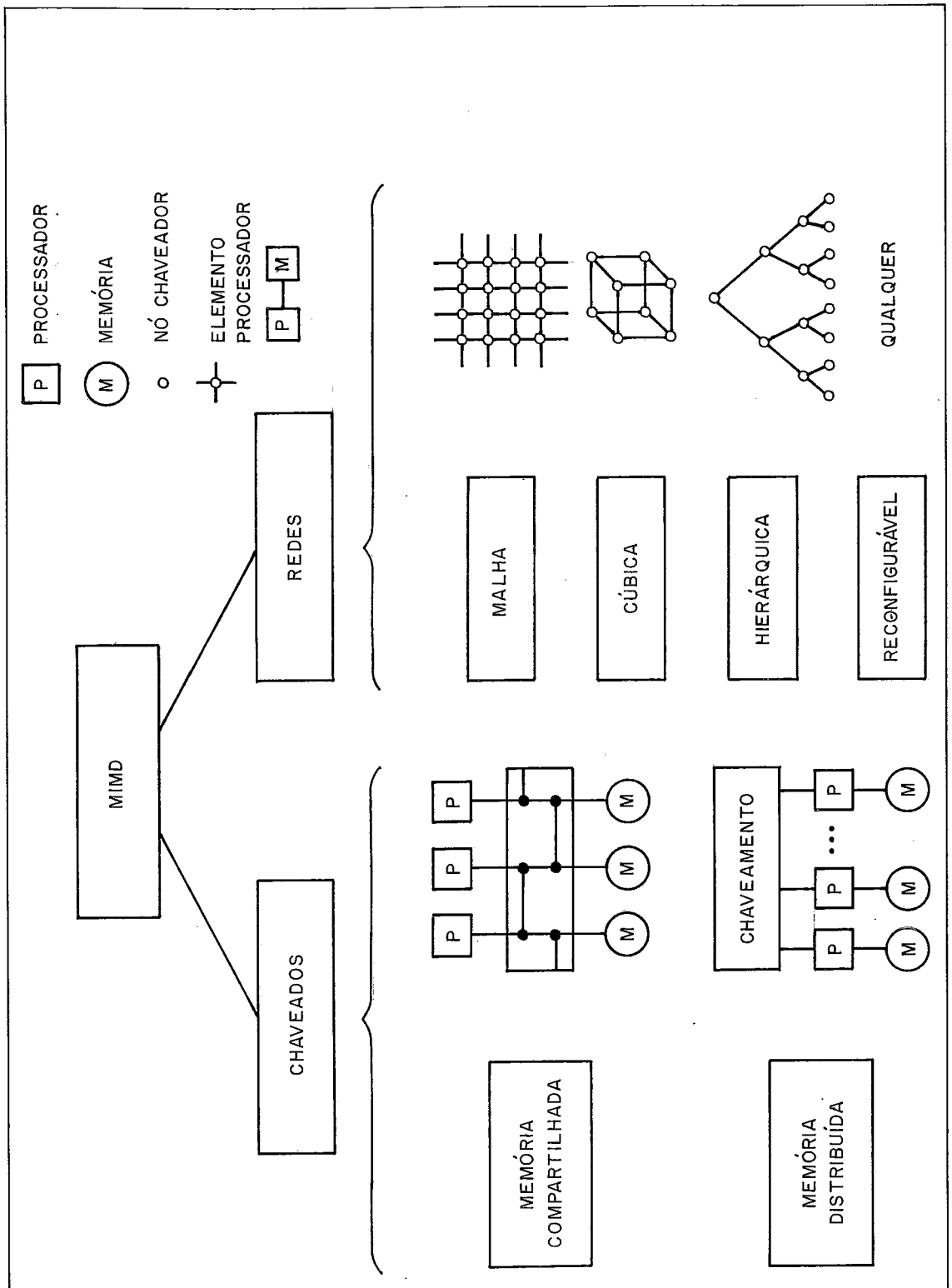


FIG. 2.9 – CLASSIFICAÇÃO DE HOCNEY

II.2.3.1 - SISTEMAS CHAVEADOS

Em sistemas chaveados existe uma unidade de chaveamento que conecta os processadores aos módulos de memória. Computadores paralelos com dezenas e centenas de processadores geralmente utilizam redes de múltiplos estágios.

Para sistemas chaveados a expansão do sistema é limitada apenas pelo tamanho da chave de matriz; ela pode ser expandida modularmente dentro das limitações do projeto. Para a prevenção contra falhas da chave e do sistema, pode ser implementada uma segmentação e/ou redundancia da chave. A adição de unidades geralmente aumenta o desempenho, isto é, um maior potencial para a eficiência do sistema sem reprogramação do sistema operacional.

Dentro desta mesma classe podemos ter um segundo nivel de classificação em que dividimos os sistemas em sistemas de memória compartilhada e sistemas de memória distribuida (fig. 2.10).

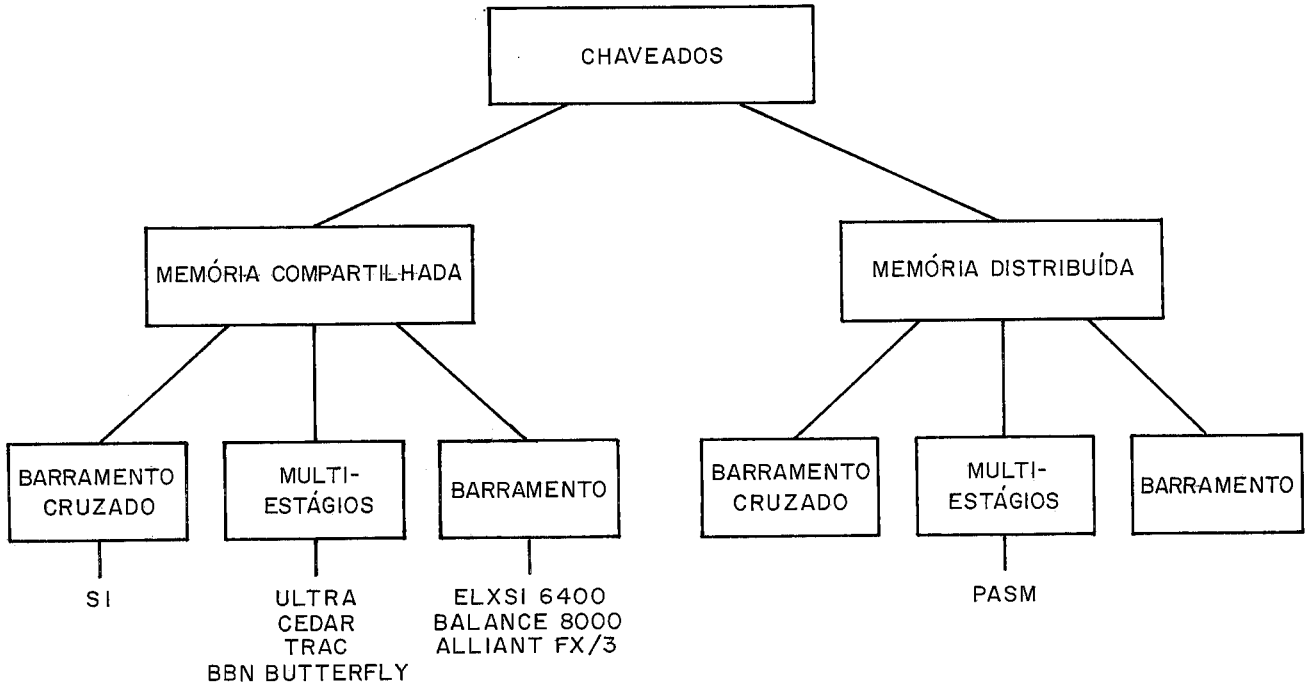


FIG. 2.10 — CLASSIFICAÇÃO DE HOCKNEY
2º NIVEL DE CLASSIFICAÇÃO DOS SISTEMAS CHAVEADOS

II.2.3.1.1 - SISTEMAS DE MEMORIA COMPARTILHADA

Em sistemas de memória compartilhada, processadores são conectados a módulos de memória compartilhada via uma unidade de chaveamento. Desta maneira a memória é compartilhada por todos os processadores de uma maneira equalitária, normalmente contendo um grande espaço de endereçamento. Tal sistema é também chamado de sistema de memória comum. Nestes sistemas a memória é global e seu espaço de endereçamento é único e comum a todos os processadores.

II.2.3.1.2- SISTEMAS DE MEMORIA DISTRIBUIDA

Em sistemas de memória distribuida, cada processador tem sua própria memória e utiliza a rede para se comunicar com os outros processadores via troca de mensagens. São incluidos os multiprocessadores cujos espaços de endereçamento são únicos (virtual), porèm distribuidos pelas memórias dos processadores. A rede é utilizada pelos processadores para acesso remoto a memória.

Existem exemplos de puros sistemas de memória compartilhada e de puros sistemas de memória distribuida. Em grandes sistemas chaveados podemos encontrar ao mesmo tempo memória local (distribuida) e memória global (compartilhada). Tais casos são classificados como sistemas hibridos.

Ambos os sistemas de memória compartilhada e de memória distribuida podem ser subdivididos em : sistemas de "barramento cruzado", sistemas multi-estágios e de barramento único.

II.2.3.2 - SISTEMAS DE REDE

Nos sistemas de rede , a memória é distribuida e os processadores são interconectados atravès de uma topologia definida. Em função da topologia os computadores podem ser classificados em malha,cúbica, reconfigurável e hierárquica (fig 2.9).

Tais sitemas são sistemas de memória distribuida, já que toda a memória é distribuida pelo sistema como memória local. Eles não podem ser definidos como sistemas

chaveados no sentido de que os elementos do sistema não são ligados a uma unidade de chaveamento. Elementos de processamento só podem se comunicar com elementos de processamento que estão diretamente conectados a eles. A comunicação entre EPs distantes entre si só pode ser realizada através da transferencia de dados por EPs intermediários na rede.

Topologias das redes de elementos de processamento :

1. REDES EM MALHAS

   Podemos encontrar malhas quadradas , hexagonais e outras. Em uma malha quadrada de dimensão d, cada EP éconectado à 2 d EPs vizinhos.

2. REDES CUBICAS

   Encontramos nesta classe os cubos multi-dimensão ou hipercúbos binários. O número de EP não pode ser aumentado sem antes aumentar o número de conecções para cada EP.

3. REDES HIERARQUICAS

São redes com topologia recursiva, tais como árvores, piramides, e cachos (cluster).

4. REDES RECONFIGURAVEIS

Comportam todas as classes de redes nas quais as interconexões podem ser trocadas, seja por software a nivel de programação, seja por hardware.

II.3 - AVALIAÇÃO DE DESEMPENHO

Uma das primeiras tentativas de se caracterizar o desempenho de um computador foi realizada em 1960 por J.C.Gibson da IBM [30], que realiza um programa com uma série de instruções dinah^micas e define tempo de execução como sendo o tempo médio de execução das instruções. O inverso desta medida é então denominado de MIPS que é o número médio de instruções realizados por segundo.

O resultado da avaliação do número de MIPS de uma máquina depende da escolha das sequências de instruções escolhidas para a avaliação.

Por exemplo, um computador "1" necessita para a execução da instrução SENO A de 17 ciclos de máquina e para a execução da operação MOV X de 3 ciclos de máquina. Um computador "2" necessita para a execução da instrução SENO A de 10 ciclos de máquina e para a execução da operação MOV X de 6 ciclos de máquina.

Caso a sequência de instruções escolhidas contenha 100 operações MOV A e apenas uma operação SENO A o computador 1 e o computador 2 necessitarão respectivamente de 317 e de 610 ciclos de máquina para executar a sequência .

Caso a sequeh^ncia de instruções escolhidas contenha 20 operações MOV A e 15 operações SENO A o computador 1 e o computador 2 necessitarão respectivamente de 315 e de 210 ciclos de máquina para executar a sequência .

Como demostrado pelo exemplo acima o resultado da avaliação em MIPS de um computador é válido apenas para comparações entre computadores de mesma família, ou seja que possuam a mesma arquitetura(esta é a razão pela qual IBM assim como outras grandes companhias apresentam suas novas máquinas em comparação as suas antigas).

Extendendo o conceito da importância da sequência de instruções utilizadas para a avaliação do desempenho de um computador, podemos concluir que a natureza da aplicação é um dos fatores fundamentais na avaliação do desempenho de um computador. (Por exemplo, o desempenho de um supercomputador operando na modalidade vetorial/paralela pode ser 100 vezes maior que o mesmo sistema operando na modalidade escalar/sequêncial.)

Foram então desenvolvidos **benchmarks** que são programas contendo sequências de instruções que se aproximam do comportamento das tipicas aplicações das linguagens de alto nivel.

Cada **benchmark** tem como finalidade avaliar o desempenho de um computador para uma determinada aplicação, podemos citar como exemplos de **benchmarks**[30]:

1. **WHETSTONE**

   Desenvovido no final dos anos 60 por Curnow e Wichman pelo U. K. National Physical Laboratory in Whetstone, England. E baseado em estatisticas coletadas originalmente em programas **ALGOL** mas é mais popularmente implementado em **FORTRAN**. E em geral utilizado para avaliar aplicações cientificas

e de engenharia. Os resultados são dados em milhões de operações de Whetstone por segundo.

2. DHRYSTONE

Desenvolvido por R.P.Weicker. Sua primeira avaliação foi para programas em ADA. Ele enfatisa os dados e operações encontradas no sistema ao invés de programação número rica. Contèm 100 passos com 15 procedures e function calls. Possui tambem versão para C e Pascal. Os resultados são fornecidos em milhões de Dhrystones por segundo.

3. Linpack

Desenvolvido por J. Dongarrara em 1979. E uma coletânea de subrotinas Fortran para resolver problemas de equações em algebra linear. O Benchmarck consiste na resolução de 100 equações de dupla precição. Os resultados são fornecidos em termos de Linpack MFLOPS(milhões de operações de ponto flutuante por segundo).

OBS:

Ao contrário do que se imagina, não existe nenhum padrão de medida de desempenho para computadores. Mesmo para padrões existentes tais como Whetstones,Dhrystones e Linpack não é possivel de se certificar da validade das medidas de desempenho avaliadas. (Jack Dongarrara do Departamento de

Energia do Laboratorio Argonne em Illinois responsável pela publicação dos resultados do Benchmarks Linpack não garante a autencidade de seus dados)

II.4 - Conclusão

Demonstramos no inicio deste capitulo que o processamento paralelo pode ser implementado em quatro niveis:

Ao nivel do programa,ao nivel da tarefa, ao nivel da inter-instrução e ao nivel da intra-instrução.

Fizemos então um estudo da implementação do paralelismo em sistema uniprocessadores(seção II.1). Extentendo este estudo realizamos um estudo das estruturas de computadores paralelos e de suas classificações. Verificamos que para uma determinada classificação as implementações do processamento paralelo em cada um dos niveis citados não são mutuamente exclusivas. (Como exemplo segundo a classificação de K.HWANG, a maioria dos computadores atualmente existentes são pipeline e alguns deles assumem também estruturas de array ou de multiprocessadores(seção II.2.1))

Para entendermos a influência da arquitetura do computador paralelo na avaliação do seu desempenho definimos o conceito de granularidade.

Definimos um programa de granularidade fina como um programa cujo grau de paralelismo está ao nivel da intraintrução, e um programa de granularidade grossa, como um programa cujo grau de paralelismo está ao nivel do processo.

A seguir, fizemos um estudo das avaliações de desempenho existentes (seção II.3) e concluimos que o estudo do desempenho de computadores é uma função complexa de muitos aspectos interrelacionados, tais como arquitetura da máquina, natureza da aplicação, nivel de vetorização , paralelismo do programa e outros mais.

Podemos então concluir que a determinação do desempenho do sistema computacional ACP deverá ser feita baseada em fatores que levarão em consideração a arquitetura do sistema e a finalidade de sua aplicação.

CAPITULO III

O MULTIPROCESSADOR PARALELO ACP

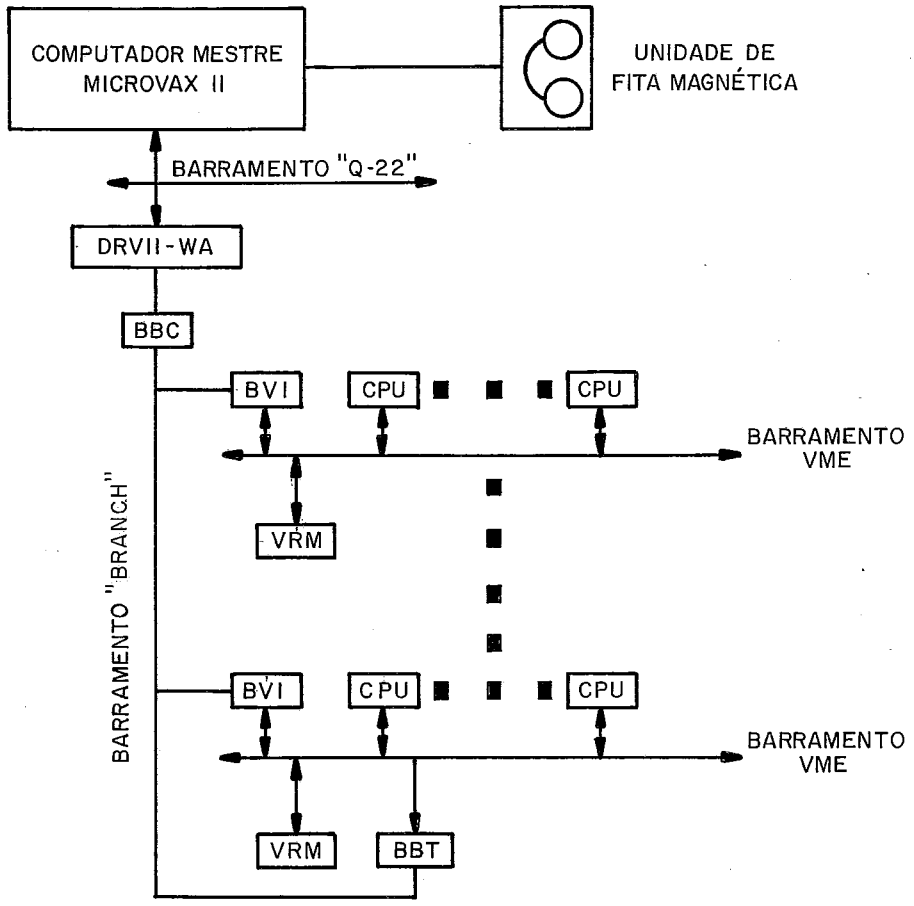
Neste capitulo faremos uma descrição do Sistema ACP, de sua arquitetura, do software desenvolvido para a sua implementação (ACPSYS) e de sua aplicação em sistemas OFF-LINE e em sistemas de tempo real.

III.1 - ARQUITETURA DO SISTEMA ACP

Os "elementos de processamento" do Sistema ACP , ou simplesmente processadores do Sistema ACP , são conectados em bastidores de padrão VME [17] para 20 módulos, formando assim os "galhos" da árvore.

Os bastidores VME são por sua vez interligados via um barramento padrão RS485 , chamado BRANCH [4], formando a raiz do sistema. O barramento BRANCH é um barramento de alta velocidade de transmissão de dados, 20Mbytes/segundo, ele possui 32-bits de dados e pode comportar até 31 bastidores de padrão VME .

Barramentos Branch podem também ser interligados a um barramento chaveado, BUS SWITCH (figura 3.2)[14] que consiste de um bastidor capaz de interligar bi-direcionalmente 16 BRANCH BUS .



BBC – INTERFACE CONTROLADORA DO BARRAMENTO "BRANCH"
BVI – INTERFACE DO BARRAMENTO TRONCO VME
DRVII – UNIDADE DE DMA
CPU – NÓ DE PROCESSAMENTO
BBT – TERMINAL DO BARRAMENTO TRONCO
VRM – ÁRBITRO DO BARRAMENTO VME

FIG. 3.1 – SISTEMA ACP – CONFIGURAÇÃO PADRÃO

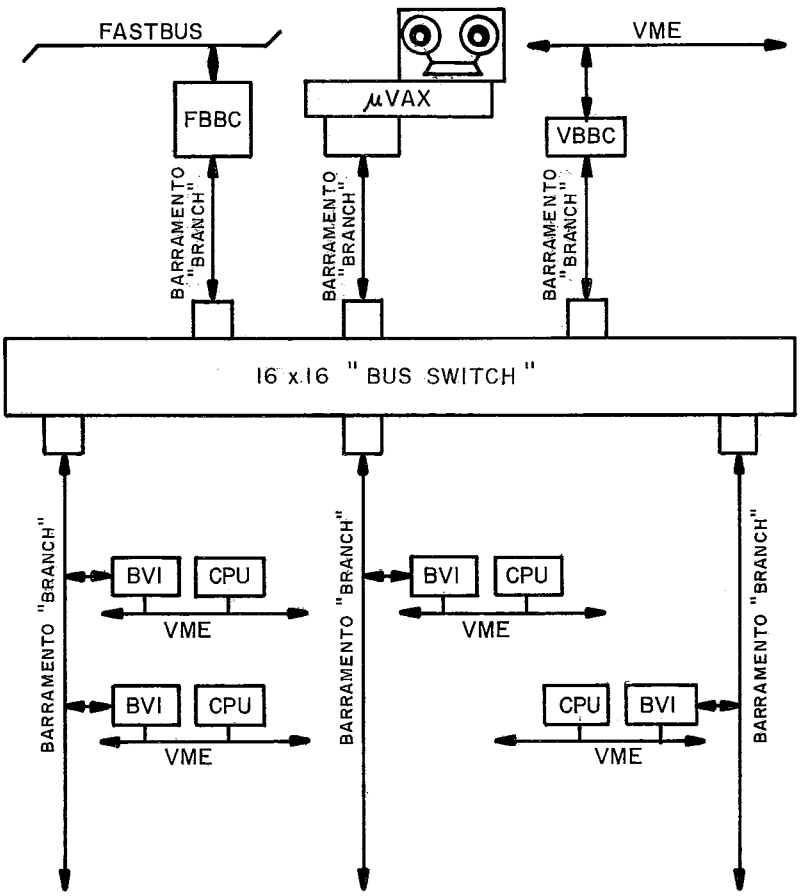


FIG. 3.2 – EXEMPLO DE UM SISTEMA ACP CONSTITUÍDO POR UMA "BUS SWITCH" (CAPAZ INTERLIGAR ATÉ 16 "BRANCH BUS")

Um sistema possuindo 16 BRANCH BUS interligadas pelo BUS SWITCH é capaz de realizar transferências paralelas entre 8 pares de "Branch Bus", ou seja, transferências de dados em até 160 Mbytes/seg. Na figura 3.3 apresentamos o exemplo da implementação de um Sistema ACP com uma alta capacidade de transferência de dados (até 240Mbytes/seg.)

Os elementos de processamento [5], (figura 3.4) são compostos por:

1. um microprocessador comercial de 32 bits/16MHZ, MC68020 ou AT&T32100 , e seu respectivo coprocessador de ponto flutuante,

2. 2Mbytes de memória real RAM, Random Access Memory , podendo ser expandida em até 8 Mbytes. A memória possui tempo de acesso para a leitura de 240ns e para a escrita de 120 ns (0 wait states).

3. saída para a implementação de 64k de memória não volátil PROM , Programable Read Only Memory . Esta saída permite também a implementação de uma interface serial para padrão RS232 e de outras necessidades.

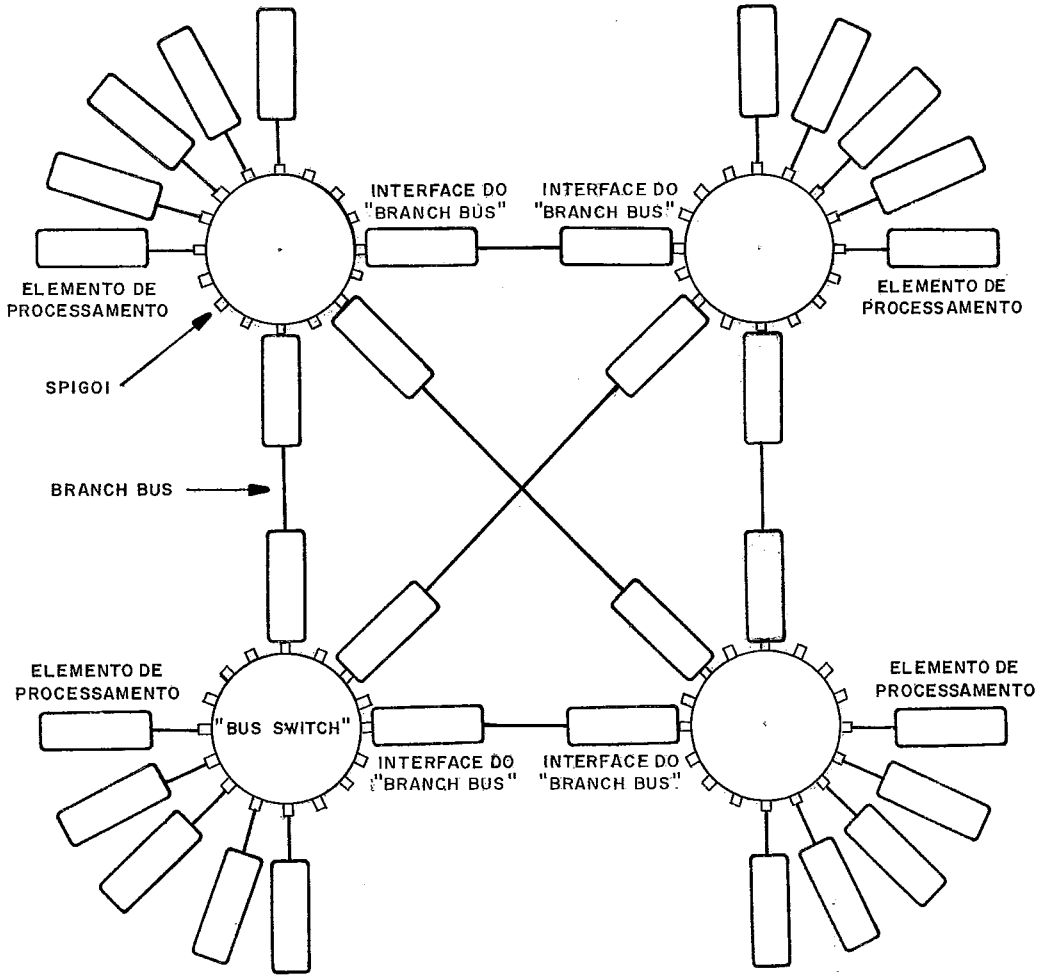


FIG. 3.3 — EXEMPLO DA IMPLEMENTAÇÃO DE UM SISTEMA ACP COM ALTA CAPACIDADE DE TRANSFERÊNCIA DE DADOS (ATÉ 240 MBYTES/SEG)

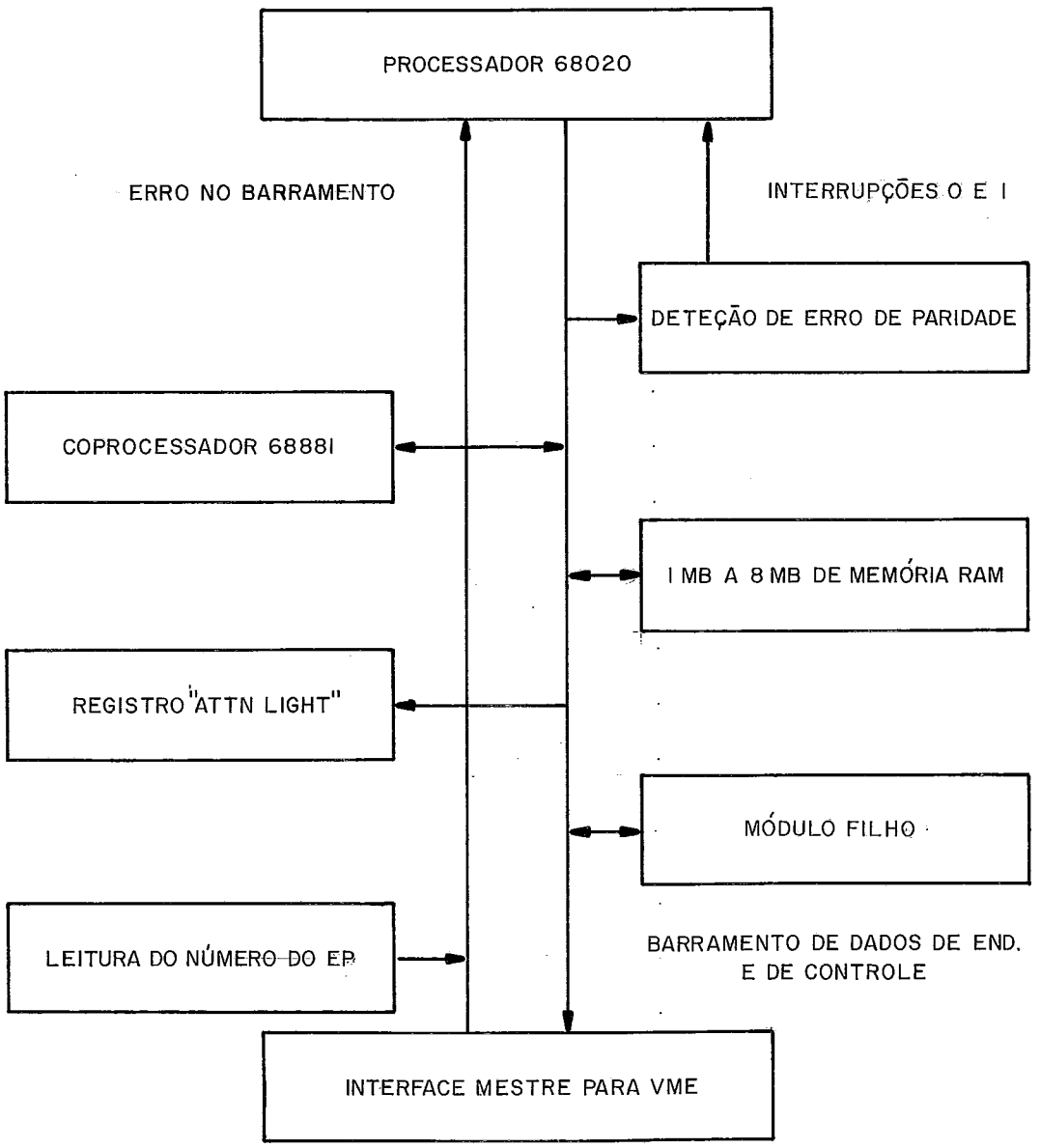


FIG. 3.4 — ESTADOS ASSUMIDOS PELOS ELEMENTOS DE PROCESSAMENTO DO SISTEMA ACP

4. interface para o barramento VME do tipo "mestre/escravo". Esta interface permite que no modo escravo a sua memória possa ser acessada por qualquer módulo no barramento VME, e permite que no modo mestre o processador tenha acesso direto às memórias locais dos processadores do sistema. Cada processador vê as memórias locais dos processadores do sistema como fazendo parte de uma memória remota de 4 Gigabytes (figura 3.5).

III.1.1 - O BARRAMENTO VME

O barramento VME(VERSA MODULE EUROCARD) é um barramento que satisfaz a sistemas de 8,16,e 32 bits. E um barramento multi-processador, que permite o compartilhamento de recursos de memória e de entrada/saida entre vários cortões mestres segundo uma prioridade de acesso paralela ou alternada. A estrutura do barramento VME possui quatro niveis de prioridade de acesso. Ele é construido no conceito "mestre/escravo" onde o mestre detèm o controle do barramento e o escravo depois de decodificar o endereço e de obter o resultado de que é ele quem está sendo acessado, responde ao comando enviado pelo mestre.

MAPEAMENTO DA MEMÓRIA NO VME

BITS MAIS SIGNIFICATIVOS	ENDEREÇO	BITS MENOS SIGNIFICATIVOS	"ADDRESS MODIFIER"	
NNNNNNN	O XXXX XXXX XXXX XXXX XXXX XXXX		00IXIO	RAM – TRANSFERÊNCIA UNITÁRIA
NNNNNNN	O XXXX XXXX XXXX XXXX XXXX XXXX		00IXOI	RAM – TRANSFERÊNCIA UNITÁRIA
NNNNNNN	O XXXX XXXX XXXX XXXX XXXX XXXX		00IXII	RAM – TRANSFERÊNCIA DE BLOCOS
XXXXXXX	O XXXX XXXX XXXX XXXX XXXX XXXX		0IOIII	TRANSFERÊNCIA DE "BROADCAST"
NNNNNNN	I XXXX XXXX XXXX XXXX XXXO OOXX		00IXOI	ATIVA E MANTÉM "RESET" NO PROCESSADOR
NNNNNNN	I XXXX XXXX XXXX XXXX XXXO OIXX		00IXOI	DESATIVA "RESET"
NNNNNNN	I XXXX XXXX XXXX XXXX XXXO IOXX		00IXOI	PERMITE "BROADCAST"
NNNNNNN	I XXXX XXXX XXXX XXXX XXXO IIXX		00IXOI	IMPEDE "BROADCAST"
NNNNNNN	I XXXX XXXX XXXX XXXX XXXI XXXX		00IXOI	NÍVEL I DE INTERRUPÇÃO NO PROCESSADOR

NNNNNNN É A REPRESENTAÇÃO BINÁRIA DO NÚMERO DO PROCESSADOR

FIG. 3.5 – O SISTEMA VÊ CADA ELEMENTO DE PROCESSAMENTO DO SISTEMA ACP COMO FAZENDO PARTE DE UMA MEMÓRIA GLOBAL DE 46 BYTS

A natureza assincrona do barramento permite a cohabitação de processadores, de memórias, e de periféricos apresentando caracteristicas e velocidades diferentes.

O barramento também possui recursos para a implementação de um ciclo indivisivel de "leitura/modificação/escrita", que permite a implementação de semáforos para o compartilhamento de recursos. Sete niveis de interrupção podem ser centralizados ou repartidos entre os processadores do barramento.

Assim como o barramento Multibus é adaptado aos microprocessadores da Intel, o barramento VME foi concebido especialmente para o microprocesador MC68000 [19]. Podemos considerar o barramento VME como sendo um sub-conjunto do Versabus possuindo 96 das 120 linhas deste barramento, o barramento VME foi concebido para permitir a implementação dos formatos simples e dupla Europa, de maneira que todos os sinais necessários para um sistema de 16 bits são localizados em apenas um dos dois conectores de 96 pinos.

III.1.2 - O MICROPROCESSADOR MC68020

O MC68020 é a primeira implementação de 32-bit da família de microprocessadores MC68000 da MOTOROLA.Ele utiliza a tecnologia de VLSI, possui registros de 32-bit, barramento de dados de 32-bit, 32-bit de barramento de endereçamento, possui 18 modos de endereçamento e um rico conjunto de instruções que fornecem suporte para linguagens estruturada de alto nível e para sistemas operacionais sofisticados [18].

A estrutura do barramento do MC68020 é assíncrona, e utiliza um barramento não-multiplexado de 32-bits de dados. O processador suporta um mecanismo que permite a transferência de operandos para unidades externas que possuem tamanhos de barramento dinâmicos,o tamanho da porta da unidade é determinado automaticamente, baseado no método de cycle-by-cycle. A interface de barramento dinâmica permite ao processador uma transferência simples e um acesso eficiente para unidades de diferentes tamanhos de barramento de dados.

Um diagrama em blocos do MC68020 é apresentado na figura 3.6. Os maiores blocos operam de maneira independente para maximizar a concorrência das operações enquanto se gerência a sincronização entre a execução da instrução e a operação do barramento. O controlador de barramento carrega instruções do barramento de dados na unidade de decodificação e na

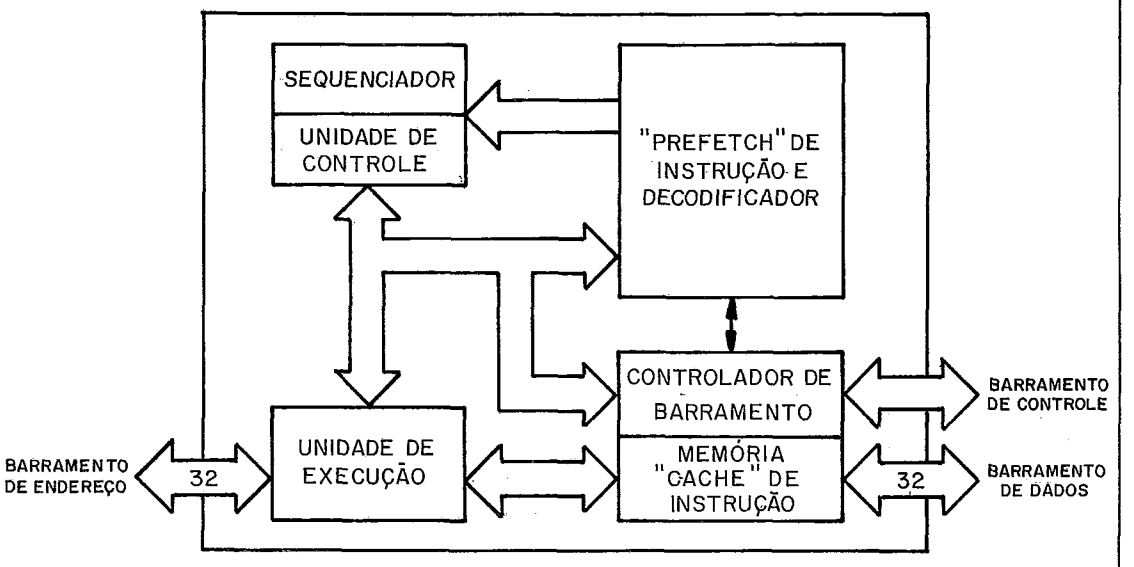

SEQUENCIADOR
UNIDADE DE CONTROLE
"PREFETCH" DE INSTRUÇÃO E DECODIFICADOR
CONTROLADOR DE BARRAMENTO
BARRAMENTO DE CONTROLE
BARRAMENTO DE ENDEREÇO
32
UNIDADE DE EXECUÇÃO
MEMÓRIA "CACHE" DE INSTRUÇÃO
32
BARRAMENTO DE DADOS

FIG. 3.6

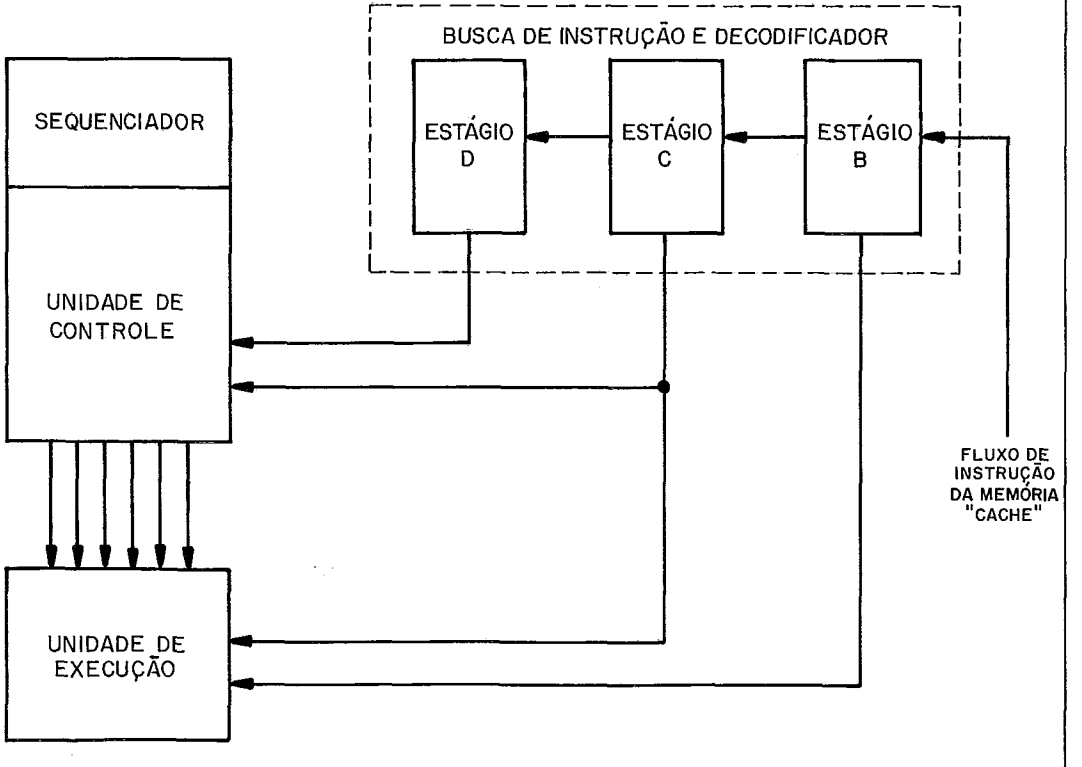

BUSCA DE INSTRUÇÃO E DECODIFICADOR
SEQUENCIADOR
ESTÁGIO D
ESTÁGIO C
ESTÁGIO B
UNIDADE DE CONTROLE
FLUXO DE INSTRUÇÃO DA MEMÓRIA "CACHE"
UNIDADE DE EXECUÇÃO

FIG. 3.7

memória cache. As unidades de sequência e de controle permitem controle do microprocessador, gerenciando o barramento interno, registros, e funções da unidade de execução.

O MC68020 suporta três estágios de instruções em pipe, seção II.2. O pipeline ocorre internamente no microprocessador e permite a operação concorrente de até três palavras de uma única operação ou de até três operações de apenas uma palavra.

III.1.3 - CLASSIFICAÇAO DA ARQUITETURA DO SISTEMA ACP

Segundo a classificação de FLYNN (seção II.2.2) o Sistema ACP, (figura 3.1) pode ser classificado dentro da classe de processadores MIMD, MULTIPLE INSTRUCTION MULTIPLE DATA por ser capaz de processar múltiplas sequências de instruções com múltiplas sequências de dados.

Segundo a classificação de HOKNEY (seção II.2.3) dentro da classe MIMD, em um segundo nivel, ele pode ser classificado como pertencente à classe de Sistemas Multiprocessadores MIMD em Redes, pelo fato de a ligação entre os elementos de processamento do Sistema

ACP não ser efetuada por chaveamento, e sim por barramentos. Dentro da classe de Sistemas em Rede ele pode ser incluido dentro de um terceiro nivel de classificação como fazendo parte da classe Hierárquica, por possuir os elementos de processamento ligados a um barramento VME que por sua vez são ligados a um barramento BRANCH BUS , que são ligados ao computador hospedeiro. E por final, dentro desta classe, ele pode ser incluido em um quarto nivel de classificação, sendo classificado como pertencente à classe de Sistemas de Arquitetura Arvore, devido à topologia da ligação.

Segundo a classificação de K.Hwangg/F.Briggs, ele pode ser classificado no grupo de sistemas Multiprocessadores com Arquitetura Fracamente Acoplada (seção II.2.1.3.2) por possuir "módulos de computação interligados por barramentos. Entretanto, os "módulos de computação" não executam operações de E/S, e têm acesso as memórias locais de todos os módulos de computação do sistema, não existe troca de mensagem, e o sistema funciona como se possuisse uma memória compartilhada de 4Gbytes distribuida entre os "módulos de computação". Logo o sistema pode também ser classificado na classe de Sistemas Multiprocessadores com Arquiteturas Fortemente Acopladas (seção II.2.1.3.1).

III.1.4 - AVALIAÇÃO DO "GARGALO" DO SISTEMA ACP

Iremos identificar o caminho percorrido por um dado ao ser transferido de uma fita para um processador do Sistema ACP de acordo com a figura 3.13, e iremos determinar a taxa de transferência a cada etapa do caminho percorrido. Definiremos "gargalo" como sendo a etapa do caminho que possui a menor taxa de transferência e que consequentemente limita a taxa de transferência de dados do sistema.

1. Leitora de fitas:

   O uVAX II do CBPF possui um leitora de fitas TS05 capaz de ler fitas com densidade de 1600bpi" (bytes por inch) e de transmitir a uma taxa de 25in"/seg ou seja a uma taxa de 40kbytes/segundo[31] e um disco rigido de 160Mbytes com uma taxa de transferência de 625Kbytes/seg [32].

2. DRV11-WA:

   A taxa de transferência do módulo de "DMA" do uVAXII, o DRV11-WA é de 500kbytes/seg no modo single cycle e de 800Kbytes/seg no modo burst cycle [33].

3. BBC:

   O módulo Controlador de Baramento Branch , o BBC [22], converte os dados vindos do DMA para o padrão Branch na mesma taxa que o DMA

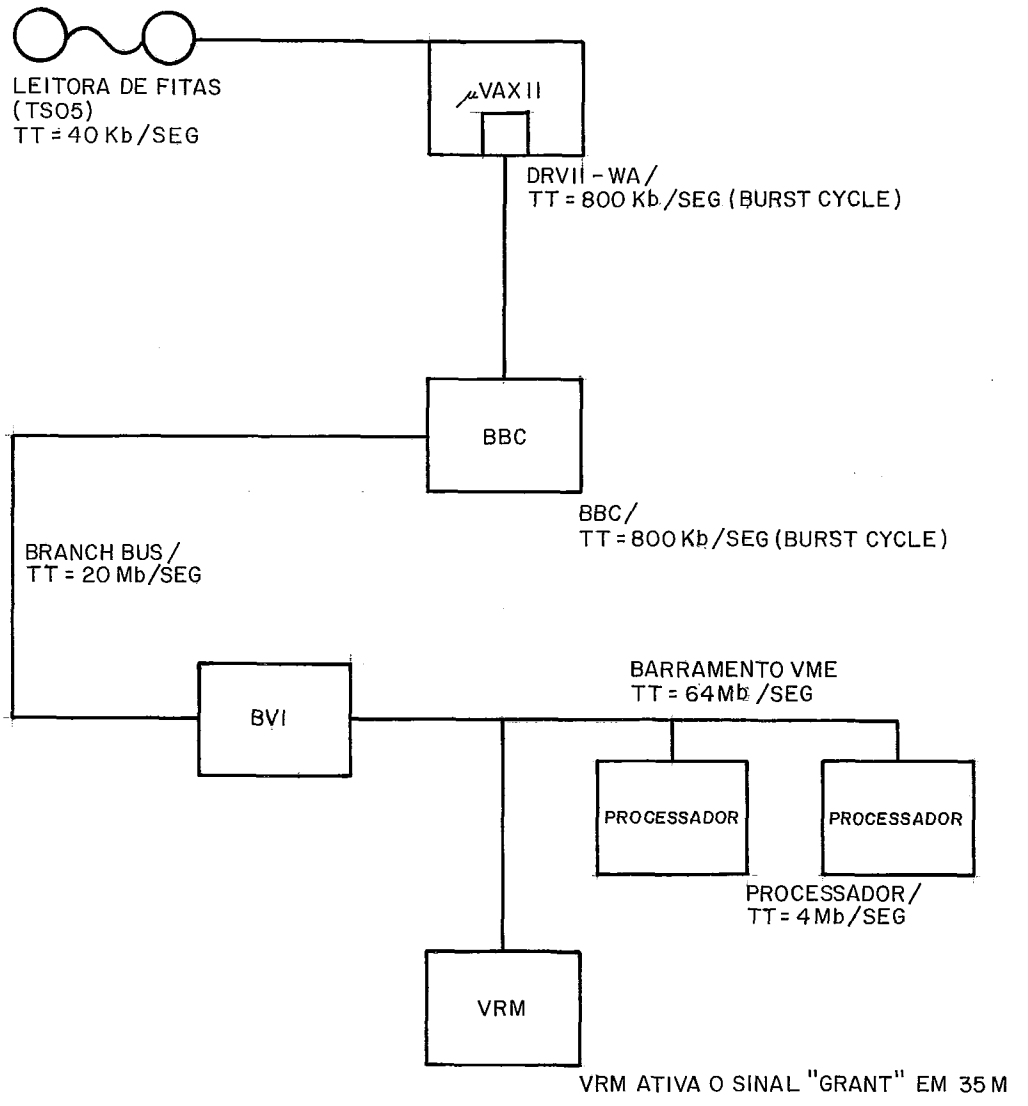


FIG. 3.8 — TAXAS DE TRANSFERÊNCIAS DE DADOS DO SISTEMA ACP

4. Barramento "Branch":

   O barramento Branch tem capacidade de transferir a uma taxa de 20Mbytes/seg [34].

5. BVI:

   módulo de interface entre o barramento Branch e o barramento VME [15]. O BVI, é capaz de transferir dados na mesma taxa em que o barramento "Branch".

6. VRM:

   Um módulo controlador de barramento VME, o VRM [21],é capaz de dar a posse do barramento em 35ns, para o módulo BVI e para o módulo de processamento localizado no primeiro slot, e pode demorar até 1,2us para dar a posse para o processador localizado no vigésimo slot.

7. Barramento "VME":

   O relógio padrão das linhas paralelas do barramento VME é de 16Mhz, logo o barramento é capaz de transmitir á uma taxa de 64Mbytes/seg [16],

8. Processador:

   O tempo de acesso para a leitura de uma memória é de 240ns (40Mbytes/segundo) e o tempo de acesso para uma escrita é de 120ns (8Mbytes/segundo).

O programa de simulação trabalhou com transferências de dados entre o disco rigido e o Sistema ACP para eliminar o "gargalo" causado pela leitora de fitas.

O "gargalo" do sistema será causado pelo disco do uVAX que limita o sistema a uma taxa de transferência de 625 Kbytes/seg.

III.2 - ACPSYS

O Sistema ACP é um sistema multiprocessador escravo. Ele foi configurado para a implementação de um microcomputador uVaxII ou um minicomputador VAX 11/780 como computador hospedeiro.

O computador hospedeiro tem a função de gerenciar todo o acesso de "entrada/saida" dos elementos de processamento do Sistema ACP , assim como outras necessidades do sistema.

Cabe a cada elemento de processamento do Sistema ACP apenas a tarefa de processar.

O pacote de software desenvolvido para o funcionamento do Sistema ACP se chama ACPSYS . Este software só pode ser istalado em um sistema operacional de ambiente VMS .

Dentro deste pacote consta o sistema operacional que é carregado nos processadores do Sistema ACP , o LUNI , ou LITTLE UNIX [24]. O computador hospedeiro carrega o LUNI e o programa a ser executado pelo processador na

fase de inicialização do sistema. O LUNI coloca o processador no estado READY esperando o carregamento do evento a ser processado, ver figura 3.9. Ao receber o evento o processador passa para o estado RUNNING e executa o programa que foi carregado pelo hospedeiro. Ao término da execução do programa o processador passa para o estado DONE e fica esperando a coleta dos resultados para só então passar para o estado READY, reinicializando o ciclo. Caso ocorra uma exceção, o LUNI coloca o processador no estado DEAD esperando uma reinicialização pelo hospedeiro.

O processo que é executado no hospedeiro inicializa um segundo processo, o RMIMOM, cuja função é a de monitorar o estado dos processadores e de executar todos os pedidos de entrada e de saída entre o processo executável do hospedeiro e os processos executados nos processadores do Sistema ACP.

Para a utilização do ACPSYS, o usuário deve gerar um arquivo, o User Parameter File(UPF)[11], contendo as informações sobre o programa que deve ser processado no computador hospedeiro, sobre o programa a ser processado nos elementos de processamento do Sistema ACP, sobre o número de processadores que deseja utilizar , e outras informações, ver figura 3.10. A documentação pode ser encontrada no manual ACP User's Guide for Utilities[11]. Foi desenvolvido um programa

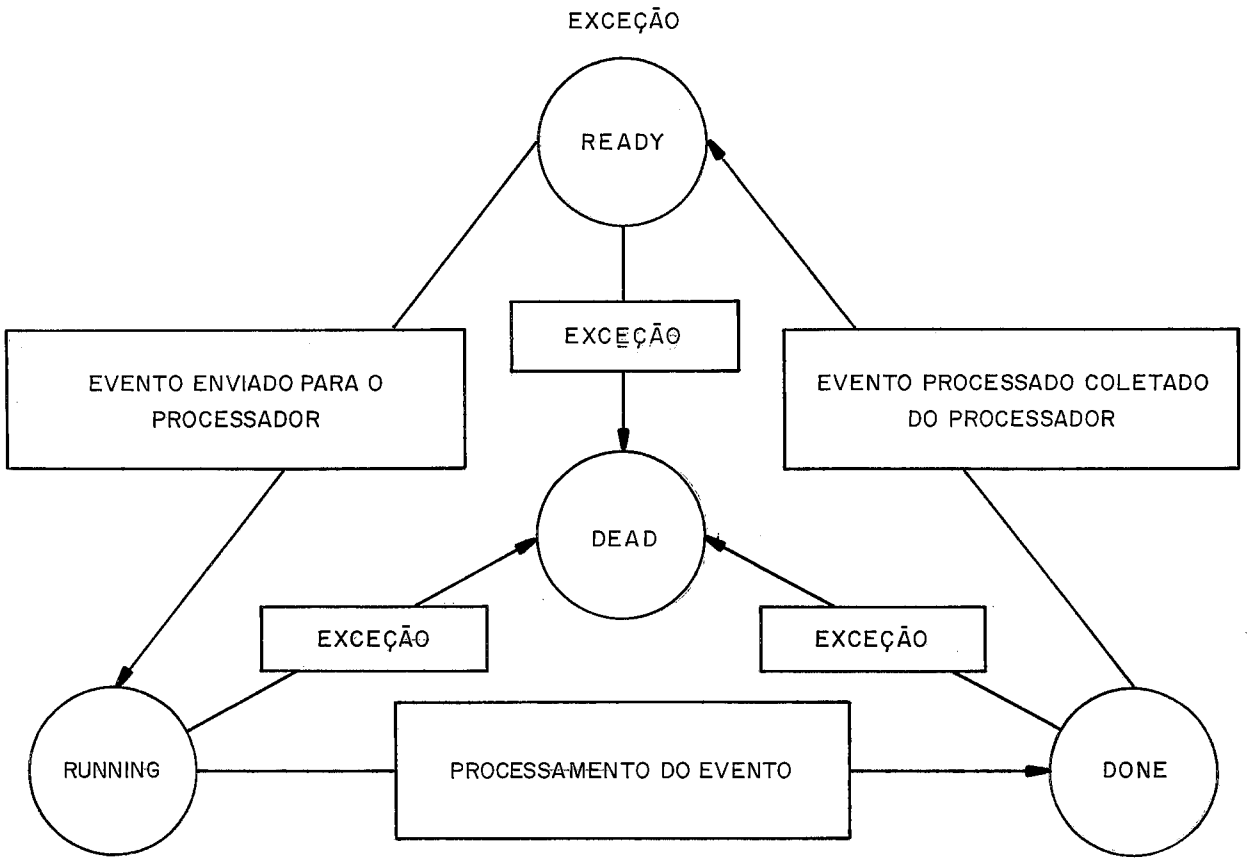


FIG. 3.9 — ESTADOS ASSUMIDOS PELO PROCESSADOR

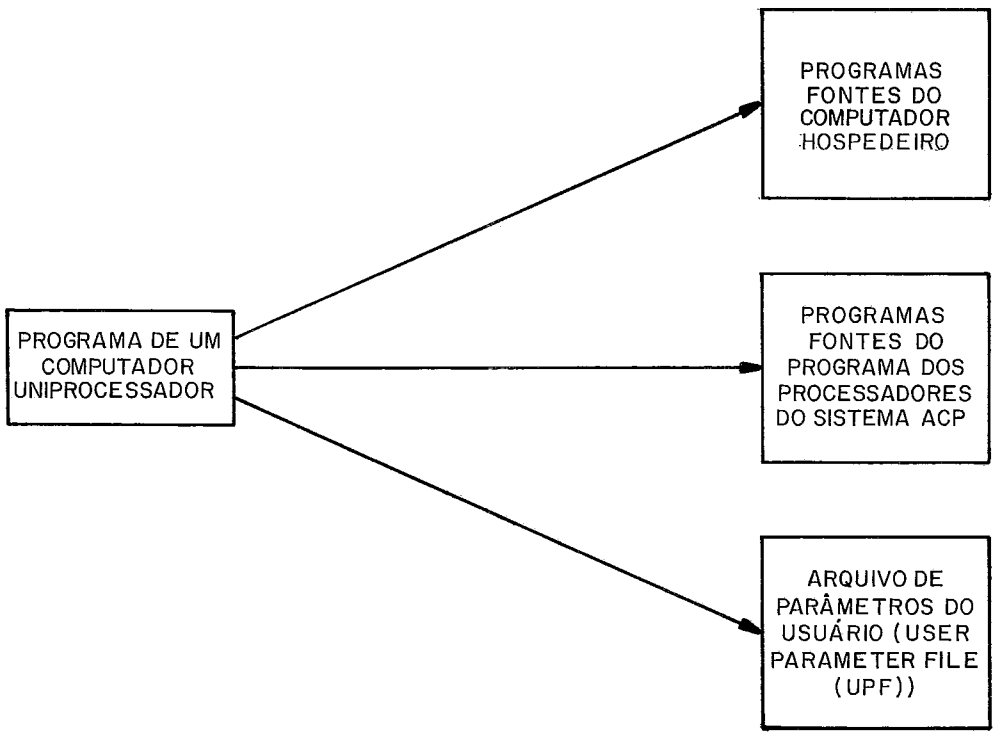


FIG.3.10 – CONVERSÃO DE UM PROGRAMA PARA SER EXECUTADO NO SISTEMA ACP

utilitário, o MULTICOMP, que ao ser executado utiliza os dados fornecidos no arquivo UPF para fazer a compilação dos programas FORTRAN em seus respectivos processadores e para criar os dados necessários para a execução do processo RMIMOM.

O usuário fornece o número de processadores que serão utilizados e quais os processos que serão processados porém não cabe ao usuário a decisão sobre qual o processador que receberá o evento, esta função cabe ao processo RMIMOM. Existem subrotinas de transmissão que possibilitam ao usuário especificar os processadores a serem acessados, caso desejado [12].

O ACPSYS contém uma série de subrotinas FORTRAN[12], fáceis de serem implementadas, e que foram desenvolvidas para realizar toda a transmissão de eventos entre o computador hospedeiro e os processadores do sistema ACP.

O software desenvolvido para o Sistema ACP comporta toda a classe de problemas que requer pouca ou nenhuma intercomunicação entre processadores. Nesta classe podemos incluir muitas aplicações de cálculo em redes, cálculo de integrais de muitas dimensões reconstrução da trajetória de particulas, bem como para problemas fora da fisica tais como cálculo de matrizes,

processamento de sinais, etc [2].

Na figura 3.11 é apresentado um fluxograma de um programa do tipo orientado para eventos onde o programa é dividido em dois programas. O primeiro programa é aquele onde é realizada toda a transmissão e coleta dos eventos para os processadores bem como cálculos estatisticos, etc. O segundo programa é aquele a ser carregado nos processadores onde será realizado o cálculo propriamente dito. Existem três niveis de subrotinas de transmissão entre o computador hospedeiro e o Sistema ACP. O grau de flexibilidade e de complexidade das subrotinas de transmissão aumenta a cada nivel. A documentação pode ser encontrada no manual Software User's Guide[12].

Foi desenvolvido também um simulador do Sistema ACP, o qual pode ser implementado em máquinas de ambiente do sistema operacional VMS.

III.3 - SISTEMA EM TEMPO REAL

Existem também várias possibilidades de se implementar o Sistema ACP em sistemas de tempo real.

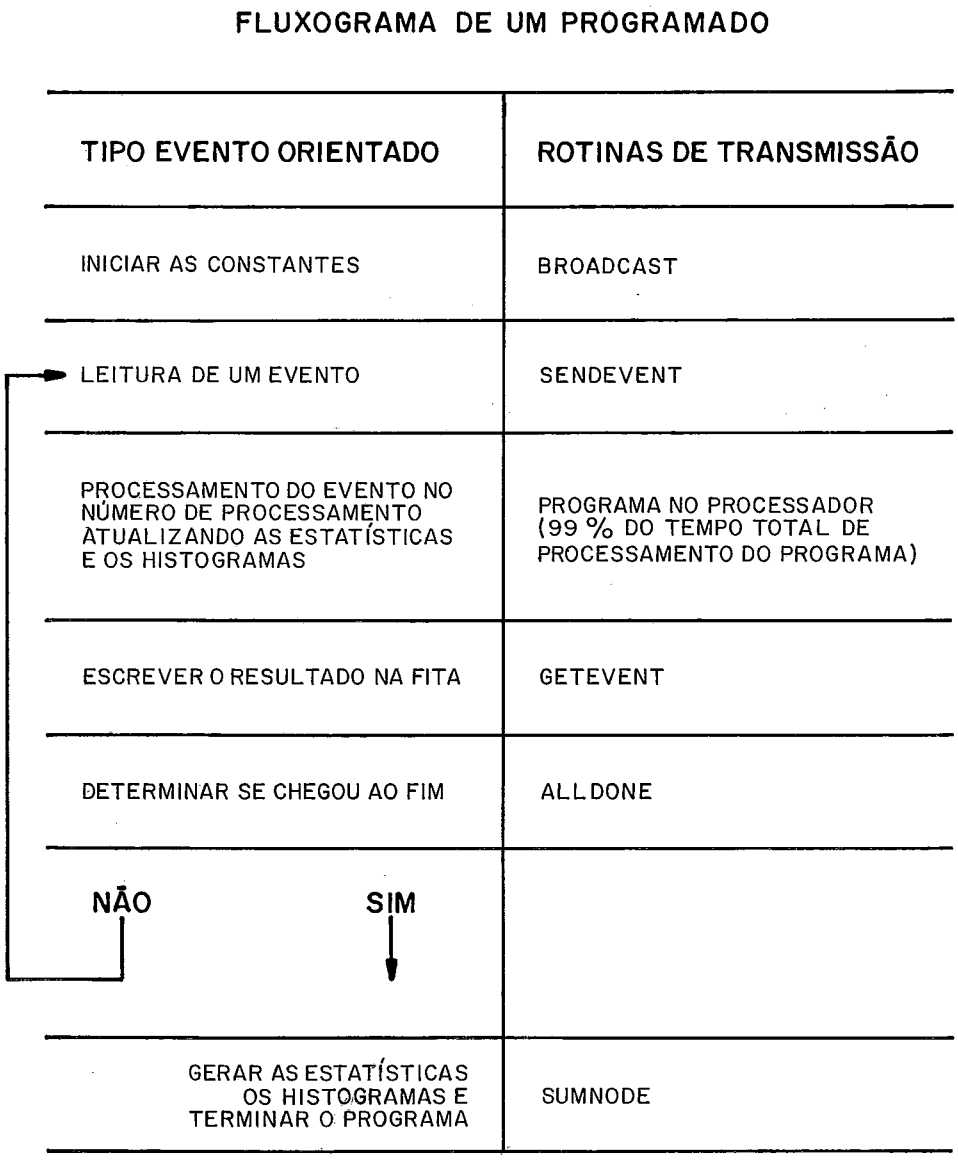


FIG. 3.11 – IMPLEMENTAÇÃO DO FLUXOGRAMA DE UM PROGRAMA DO TIPO "EVENTO ORIENTADO" NO SISTEMA ACP

Um exemplo de implementação é apresentado na figura 3.12, onde um uVaxII exerce funções de gerenciador de recursos. O gerenciador de recursos comunica ao sistema de aquisição de dados o endereço do próximo elemento de processamento livre.

Em um sistema mais complexo a decisão é determinada pelo sistema de aquisição de dados, e o uVaxII é dedicado à monitoração do processamento dos eventos e dos possiveis erros que possam vir a ocorrer no sistema[2].

A fig.9 apresenta o sistema de aquisição de dados da experiência E-769 [32] na qual o CBPF é uma das instituições colaboradoras[6]. Dois módulos foram desenvolvidos para a implementação deste sistema: um controlador de bastidor CAMAC , o SMART CRATE CONTROLER (SCC) e o Read Buffer (RBUFF) . O SCC realiza a coleta dos eventos que chegam no bastidor CAMAC segundo um programa pré-carregado pelo minicomputador VAX 11/780 . O SCC irá transmitir os eventos para o RBUFF via uma interface paralela. Os RBUFFS são módulos de padrão VME e apresentam caracteristicas de buffer de leitura. Os eventos são coletados dos RBUFFS pelos elementos de processamento. Após o processamento,os eventos são enviados para gravação em fita utilizando para isto um módulo padrão VME controlador de fita, o CIPRICO . O gerenciamento da coleta dos eventos e o processamento

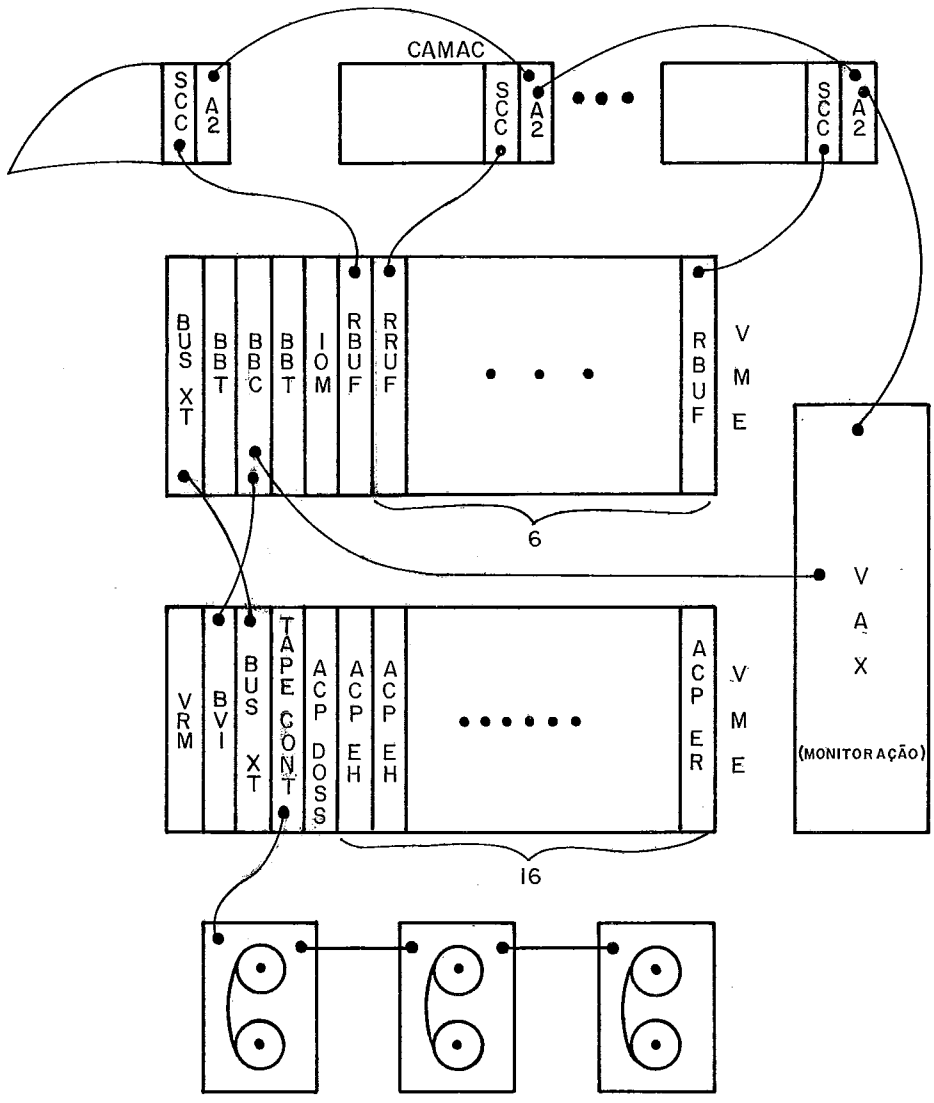


FIG. 3.12 — DIAGRAMA DO SISTEMA DE AQUISIÇÃO DE DADOS DA E-769 (OS EVENTOS SÃO COLETADOS PELOS MÓDULOS SCC DOS BASTIDORES CAMAC, ENVIADOS PARA OS MÓDULOS RBUF DOS BASTIDORES VME, OS PROCESSADORES ACP COLETAM OS EVENTOS, "FILTRAM", "EMPACOTAM" E ARMAZENAM EM FITA MAGNÉTICA)

são realizados por um dos processadores do Sistema ACP.
A monitoração do sistema é realizada pelo VAX 11/780

obs:

Tal sistema coletou uma média de 400 eventos por segundo de aproximadamente 4kbytes cada, produzindo em 8 meses de coleta de dados mais de 10.000 fitas. A análise dos eventos desta experiência necessitaria de 12 anos de processamento no sistema atual do FERMILAB COMPUTER CENTER , mas será realizada em aproximadamente 12 meses em um Sistema ACP OFFLINE de aproximadamente 100 processadores.

CAPITULO IV

AVALIAÇÃO EXPERIMENTAL DO DESEMPENHO DO MULTIPROCESSADOR ACP

Para se fazer a avaliação do desempenho iremos desenvolver programas para a simulação dos problemas a serem estudados, isto é, de problemas "orientado para eventos" (seçãoIV.1) e de problemas com comunicação entre os processadores(seção IV.3). Baseado nos resultados obtidos na seção IV.1 iremos desenvolver um modelo para o Sistema ACP (seçãoIV.2).

O estudo do desempenho será baseado nos resultados obtidos na avaliação da DEGRADAÇÃO e na avaliação da ACELERAÇÃO do Sistema ACP (seção IV.1)

Sabemos que em um sistema paralelo ideal uma tarefa sendo processada em (P) processadores deverá ser (P) vezes mais rápida que no mesmo sistema com apenas um processador.

A diferença entre o tempo de processamento real e o tempo de processamento ideal que o sistema paralelo é capaz de prover será chamada de DEGRADAÇÃO introduzida pelo sistema paralelo.

A relação entre o tempo de processamento em um Sistema ACP com 1 processador e o tempo de processamento em um sistema com (P) processadores será chamada de ACELERAÇÃO do sistema com (P) processadores.

IV.1 - AVALIAÇÃO EXPERIMENTAL DO DESEMPENHO DO MULTIPROCESSADOR ACP EM PROBLEMAS DO TIPO "ORIENTADO PARA EVENTOS"

Na figura 3.11 é apresentado um fluxograma típico de um problema "orientado para eventos". Neste tipo de problema aproximadamente 99% do tempo de cpu é gasto no processamento do evento e aproximadamente 1% é gasto em transferência de dados [25].

Um algoritmo típico para um problema do tipo "orientado para eventos " é descrito a seguir:

1. Ler um evento gravado em fita para ser processado.
2. Processar o evento.
3. Colocar o resultado em um histograma.
4. Verificar o término do processamento de todos os eventos, para então realizar os cálculos estatísticos, etc.
5. Voltar ao passo 1 para a leitura do próximo evento, caso não tenha terminado o processamento de todos as eventos.

Como o processamento de cada evento é independente do processamento de eventos anteriores, este tipo de problema é ideal para uma arquitetura como a do ACP, onde os processadores executam paralelamente o processamento dos eventos ao mesmo tempo em que o uVAXII realiza a

transmissão dos eventos, de maneira que um Sistema ACP ideal com (P) processadores será equivalente a (P) sistemas uniprocessadores (figura 4.2).

A presença de um computador hospedeiro faz com que o sistema multiprocessador seja bastante simples de ser implementado, mas por outro lado, o fato de o hospedeiro ser o gerenciador e o executor das operações de E/S do sistema faz com que o sistema multiprocessador seja dependente do desempenho do hospedeiro e de sua capacidade de implementar as operações de E/S.

Dois casos extremos podem ocorrer:

1. Ao aumentar muito as operações de E/S o uVAXII passará a não mais ser capaz de atender os pedidos na taxa desejada.

2. Ao aumentar muito as operações de processamento dos processadores, a taxa com que os processadores irão solicitar as operações de E/S irá diminuir. Logo o uVAXII ficará ocioso podendo o sistema desta maneira receber mais processadores no sistema.

Podemos então observar que o desempenho do sistema é diretamente dependente da relação entre a taxa de processamento dos processadores e da taxa de comunicação entre os processadores do Sistema ACP e o uVAXII.

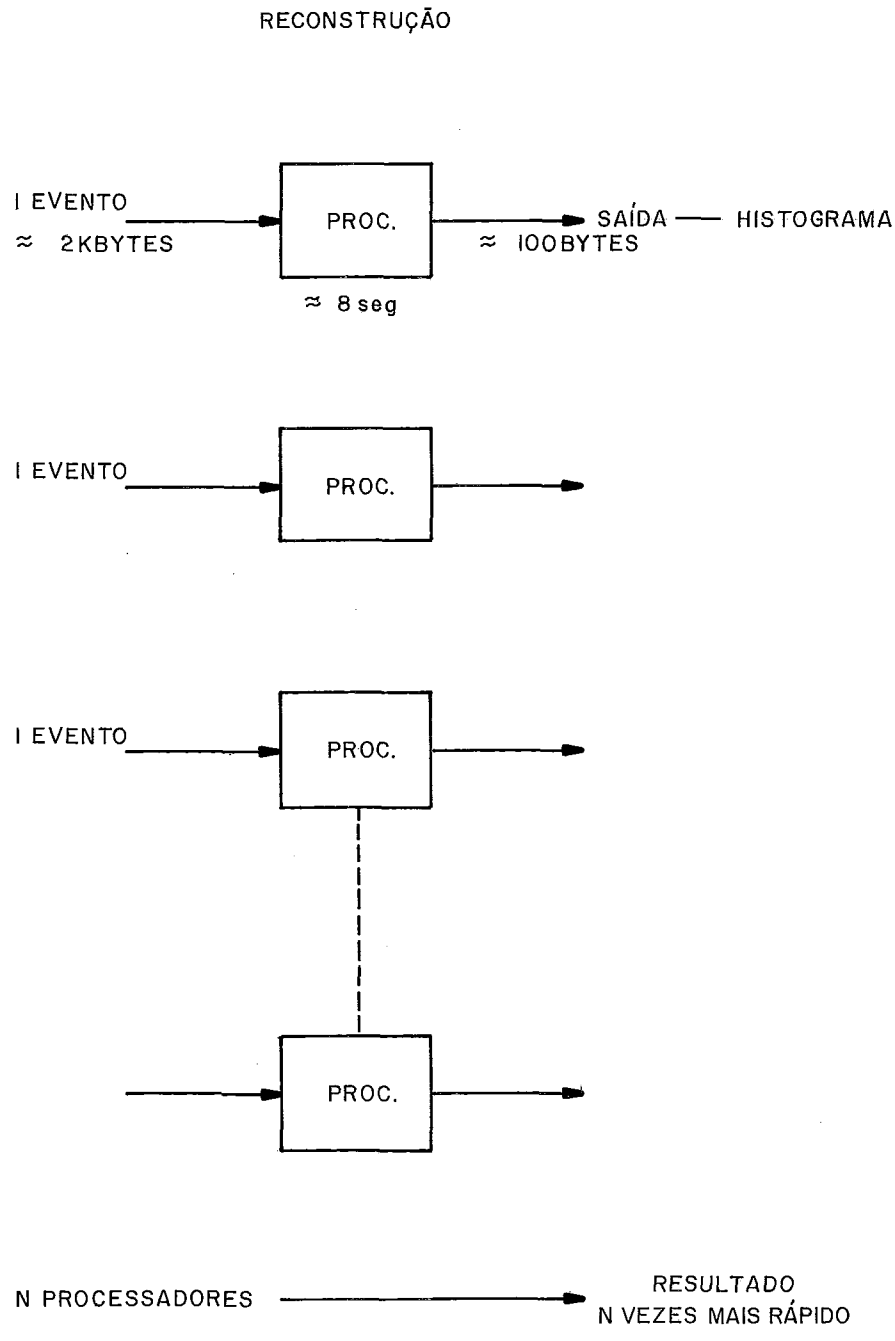


FIG. 4.2 — SISTEMA ACP IDEAL DE N PROCESSADORES EQUIVALE A N SISTEMAS UNIPROCESSADORES

IV.1.1 - DESENVOLVIMENTO DO PROGRAMA DE SIMULAÇÃO

Iremos desenvolver um programa de simulação do processamento de eventos de acordo com o algoritmo de um problema do tipo "orientado para eventos" descrito na seção anterior.

Como visto na seção anterior, os parâmetros que alteram o tempo total de processamento , ou melhor o desempenho do Sistema ACP em um problema do tipo "orientado para eventos" , são:

- (P1) o tamanho do evento sendo transmitido
- (P2) o número de processadores do sistema.
- (P3) o tempo de processamento do evento
- (P4) tamanho do resultado coletado do processador

Iremos nos basear nestes parâmetros para o desenvolvimento de um programa de simulação. Como visto no capitulo III, o Sistema ACP é um sistema escravo de um computador hospedeiro. Cabe ao computador hospedeiro realizar todas as operações de E/S do Sistema ACP , e cabe aos computadores do Sistema ACP a tarefa de "processar". De maneira que para a implementação do programa de simulação será necessário gerar um programa de simulação de transmissão de eventos para ser executado pelo computador hospedeiro, e um programa de simulação de processamento de eventos para ser executado pelos processadores do Sistema ACP .

O Sistema ACP do CBPF , possui no momento apenas 21 processadores. Decidimos realizar a simulação para sistemas ACP de um a 16 processadores ($1 < P2 < 16$), devido ao fato de 16 ser um número bastante utilizado em sistemas multiprocessadores em particular em sistema de arquitetura hipercúbica (seção II.3), o que possibitará futuras comparações.

Um típico programa do tipo "evento orientado" analisa dezenas de milhões de eventos indendentes entre si, o programa é portando executado dezenas de milhões de vezes. Iremos considerar que 100 eventos será um número suficientemente grande para simularmos esta análise em um Sistema ACP com um número máximo de 16 processadores.

A simulação será feita para eventos com tamanhos médios (P1) entre 5000 palavras e 50000 palavras que são tamanho de eventos típicos de tais problemas e (P4) entre 400 e 4000 palavras que são também tamanhos típicos de resultados deste problemas.

Um tempo de processamento típico de um evento é de 20 segundos, para realizarmos o programa de simulação para 100 eventos de 5000 a 50000 palavras necessitaríamos de aproximadamente 20000 segundos de processamento de processadores ACP.

Como o objetivo da tese está em desenvolver um método experimental da avaliação do desempenho do Sistema ACP aplicável a quaisquer conjuntos de parâmetros (P1,P2,P3,P4) decidimos trabalhar com um tempo de

processamento médio de um evento (P3) de 3 segundos que foi pequeno o suficiente para podermos avaliar o desempenho para o processamento de um problema do tipo evento orientado sem termos que alocar o Sistema ACP por muito tempo.

E importante ressaltar que os resultados a serem obtidos para a análise com um tempo de processamento menor não serão proporcionais aos resultados obtidos para um tempo de processamento tipico (seção IV.2), de maneira que para obtermos o comportamento do sistema para problemas com outros parâmetros, como por exemplo de (P3)=20 segundos será necessário aplicar o método a ser apresentado.

IV.1.1.1 - ALGORITMO DO PROGRAMA EXECUTADO PELO COMPUTADOR HOSPEDEIRO

A finalidade deste programa (programa EV.FOR/ APENDICE C)é a de implementar as funções do computador hospedeiro, isto é a de realizar todas as operações de E/S do Sistema ACP. Em primeiro lugar, este programa irá executar o carregamento do sistema operacional dos processadores (o LUNI) e do programa de simulação do processamento dos eventos a ser executado pelos processadores no Sistema ACP, a seguir o programa executará um ciclo de instruções de envio e de coleta dos eventos para o Sistema ACP até que todos os eventos tenham sido processados.

ALGORITMO

1. Carregar o LUNI e o programa executado pelos processadores, e inicializar o programa RMIMOM.

2. Dar "Broadcast" da variável (P3) em todos os processadores do Sistema ACP.

3. Inicializar a contagem do tempo total de processamento dos eventos a serem processados

4. Verificar se todos os eventos já foram processados. Caso negativo continue, caso positivo passe para o item(9).

5. Ler um evento em disco.

6. Carregar um evento em um processador do Sistema ACP que esteja pronto para receber um evento,isto é, que não esteja processando.

7. Verificar a disponibilidade de um resultado a ser coletado, caso positivo, armazenar o resultado em fita.

8. Voltar ao passo 4.

9. Avaliação do tempo total de processamento.

10. Geração de histogramas estatisticos com os resultados coletados.

IV.1.1.2 - ALGORITMO DO PROGRAMA EXECUTADO PELOS PROCESSADORES DO SISTEMA ACP

A finalidade deste programa (programa NODEEV.FOR/APENDICE C) é a de simular o processamento de um evento . Para a simulação do processamento de um evento é executada uma instrução de "retardo".

A determinação da duração do retardo é feita utilizando o valor da variável (P3) fornecida pelo computador hospedeiro. O computador hospedeiro utiliza uma função randômica para gerar uma variável entre os valores 0 e (P3) para cada evento a ser enviado. De maneira que o processamento de cada um dos 100 eventos terá um tempo médio de valor (P3), e estaremos simulando uma situação mais próxima da realidade.

O programa de simulação é carregado pelo computador hospedeiro em conjunto com o LUNI . O LUNI espera o recebimento do evento para dar inicio à execução do programa, após o término da execução do programa o LUNI avisa ao computador hospedeiro que o resultado está pronto para ser coletado. Após o evento ser coletado o LUNI volta ao estado de espera de um evento.

ALGORITMO

1. Determinação do tempo de duração da instrução de retardo.

2. Execução da instrução de retardo

3. Fim (colocar o processador no estado DONE , a ser lido pelo hospedeiro)

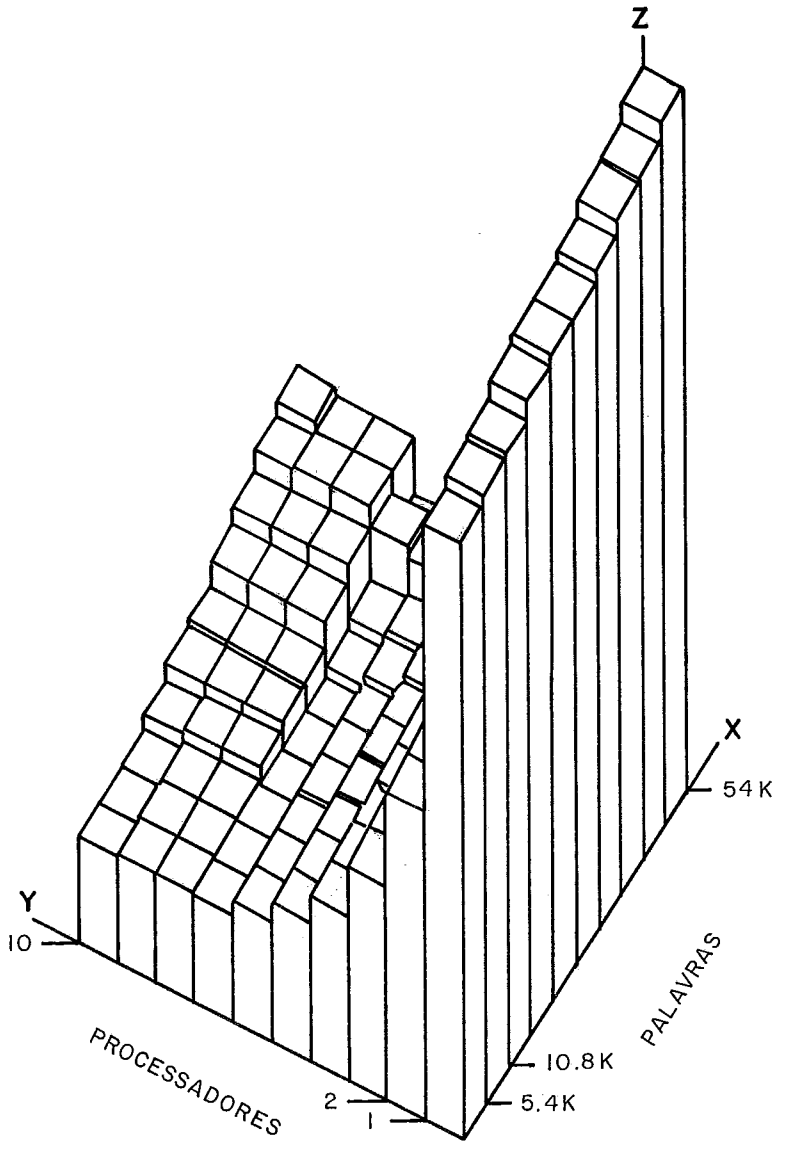
A simulação foi feita para a análise de 100 eventos assumindo tamanhos médios (P1) entre 5000 palavras e 50000 palavras, tamanhos médios de (P4) entre 400 e 4000 palavras, tempo de processamento médio (P3) de 3 segundos e para Sistemas ACPs de 1 a 16 processadores (P2). Os resultados são armazenados na TABELA 1, descrita a seguir.

TABELA 1:

Um gráfico ilustrativo desta tabela encontra-se na figura 4.3. Nesta tabela são armazenados os resultados do tempo de processamento do programa de simulação para tempo de processamento médio de um evento (P3) de 3 segundos. As linhas da tabela assumem os valores do número de processadores no Sistema ACP (P2), assumindo valores de 1 a 16. As colunas da tabela correspondem ao tamanho do bloco de palavras do evento sendo transmitido (P1) assumindo valores entre 5000 palavras e 50000 palavras e tamanho do resultado coletado (P4) assumindo valores entre 400 e 4000 palavras.

PROCESSADORES DO SISTEMA ACP (P2)	TAMANHO MÉDIO DE UM EVENTO (KBYTES) (P1)									
	5	10	15	20	25	30	35	40	45	50
16	.	.	.	.	.	.	.	.	.	.
15	.	.	.	.	.	.	.	.	.	.
14	.	.	.	.	.	.	.	.	.	.
13	.	.	.	.	.	.	.	.	.	.
12	.	.	.	.	.	.	.	.	.	.
11	.	.	.	.	.	.	.	.	.	.
10	45.762	46.078	47.84	55.82	64.152	66.891	81.141	91.91	103.109	111.07
9	48.578	49.5	51.707	59.031	65.207	67.5	81.879	92.82	105.641	105.18
8	48.578	49.5	51.707	59.031	65.207	67.5	81.879	92.82	105.641	105.18
7	46.87	47.85	48.9	50.02	51.52	52.5	57.73	64.73	93.73	74.3
6	54.39	55.11	55.43	58.21	59.34	59.14	64.18	68.8	88.73	79.76
5	63.63	64.93	65.67	68.51	71.07	70.92	74.68	77.76	86.17	83.56
4	78.06	80.586	80.742	85.211	86.24	85.9	90.56	97.17	105.64	101.23
3	105.297	107.539	107.914	110.969	113.281	113.203	120.297	125.156	132.039	129.313
2	155.406	158.57	159.516	165.117	168.406	168.414	175.258	181.898	186.227	187.008
1	308.766	315.688	318.938	329.359	334.719	336.047	342.438	351.906	355.109	366.719

TABELA I
TABELA DO TEMPO TOTAL DE PROCESSAMENTO DE 100 EVENTOS EM UM SISTEMA ACP EM FUNÇÃO DA VARIAÇÃO DO NÚMERO DE PROCESSADORES DO SISTEMA (LINHA) E DO TAMANHO MÉDIO DO EVENTO (KBYTES) SENDO TRANSFERIDO (COLUNA) O TEMPO MÉDIO DE PROCESSAMENTO DE UM EVENTO E 3 SEG (P3)
P4 = 10 % DE (P1)



X = TAMANHO MÉDIO DE UM EVENTO (KBYTS)
Y = NÚMERO DE PROCESSADORES DO SISTEMA APC
Z = TEMPO TOTAL DE PROCESSAMENTO PARA 100 EVENTOS

FIG. 4.3 – GRÁFICO REPRESENTANTE DA TABELA I

IV.1.2 - ESTUDO DA DEGRADAÇÃO DO SISTEMA ACP

Iremos definir tempo de processamento real de 100 eventos, TABELA1(P1,P2,P3,P4), como sendo o tempo de processamento gerado pelo programa de simulação e armazenado na TABELA 1, e tempo de processamento ideal de 100 eventos, T100ideal(P1,P2,P3,P4), como sendo o tempo que se espera obter do Sistema ACP caso ele se comportasse como um sistema multiprocessador ideal.

De posse dos "tempos totais de processamento" correspondentes a cada um dos parâmetros: tempo de processamento médio (P3), de tamanho médio de um evento (P1), do tamanho médio do resultado coletado (P4), e do número de processadores do Sistema ACP (P2), faremos um estudo de quanto o desempenho do Sistema ACP real se distancia do desempenho do Sistema ACP ideal à medida em que se alteram estes parâmetros, ou seja da DEGRADAÇÃO do Sistema ACP segundo cada parâmetro.

Para calcularmos a diferença entre o tempo de processamento real e o tempo de processamento ideal necessitamos determinar o tempo de processamento ideal. O programa de simulação simula o processamento de 100 eventos, logo necessitaremos determinar o tempo de processamento ideal de 100 eventos. O programa de simulação fixa o valor do tempo de processamento médio de um evento (P3) em 3 segundos.

Iremos chamar T100ideal(P1,P2,P3,P4) o tempo médio para um sistema multiprocessador ideal de (P2) processadores processarem 100 eventos de tamanho (P1), de tempo médio de de processamento (P3) = 3 segundos, e de resultado coletado com tamanho médio (P4), isto é:

T100ideal(P1,P2,P3,P4)= ((100*(P3+Tc(P1+P4)))/P2)segundos

onde:

1. P1 : Tamanho médio de um evento, ou seja número de palavras médio de um evento

2. P2: Número de processadores do Sistema ACP

3. P3: Tempo médio de processamento de cada evento utilizado pelos processadores para simularem um tempo médio de processamento de cada evento. Fixado em 3 segundos.

4. TC(P1+P4) : Baseados em testes que verificaram que o tempo de transmissão de um evento de tamanho P1 é aproximadamente igual ao tempo de coleta de um resultado de tamanho P1 (comparar tabela 4 com tabela 5), iremos considerar que o tempo de transmissão de um evento de tamanho P1 é igual ao tempo de coleta de um resultado de tamanho P1. Logo TC(P1+P4) será aproximadamente igual ao tempo de transmissão médio de um evento com um tamanho de P1 palavras somado ao tempo de coleta do seu resultado, de tamanho P4 (seção IV.1.2.2)

5. T1ideal(P1,1,P3,P4): Tempo médio ideal para um sistema uniprocessador processar um evento de tamanho P1 palavras, tempo de processamento P3 segundos e tamanho do resultado coletado P4 palavras. isto é:

   T1ideal(P1,1,P3,P4)=(P3+Tc(P1+P4))segundos

6. T100ideal(P1,1,P3,P4): Tempo médio para um sistema uniprocessador processar 100 eventos de tamanho (P1), de tamanho do resultado (P4) palavras e de tempo de processamneto (P3) segundos.

   T100ideal(P1,1,P3,P4)=100*(P3+Tc(P1+P4))segundos

7. TABELA 1(P1,P2,P3,P4): Resultados obtidos da TABELA 1. Estes resultados correspondem ao tempo total de processamento de 100 eventos de tamanho médio (P1) palavras, de tempo de processamento médio (P3) segundos, de tamanho do resultado coletado médio (P4) palavras e de processadores do sistema (P2), gerado pelo programa de simulação da seção IV.1.

A DEGRADAÇÃO do sistema será então a diferença entre o tempo de processamento real, ou seja TABELA 1(P1,P2,P3,P4), e o tempo de processamento ideal, ou seja T100ideal(P1,P2,P3,P4).

DEGRADAÇÃO(P1,P2,P3,P4)=

TABELA1(P1,P2,P3,P4)-T100ideal(P1,P2,P3,P4)

Os resultados destas diferenças são armazenados na TABELA 3 descrita a seguir.

TABELA 3

Nesta tabela são armazenados os valores dos resultados de DEGRADAÇÃO do Sistema ACP. As colunas da tabela correspondem aos valores do número de processadores no Sistema ACP (P2), variando de 1 a 10. As linhas da tabela correspondem ao tamanho médio do bloco de palavras do evento sendo transmitido para cada processador(P1), variando entre 5000 palavras e 50000 palavras e ao tamanho médio do evento sendo coletado de cada porcessador (P4), variando entre 400 e 4000 palavras. O tempo de processamento médio de cada evento (P3) é de 3 segundos. Um gráfico ilustrativo desta tabela encontra-se na figura 4.4.

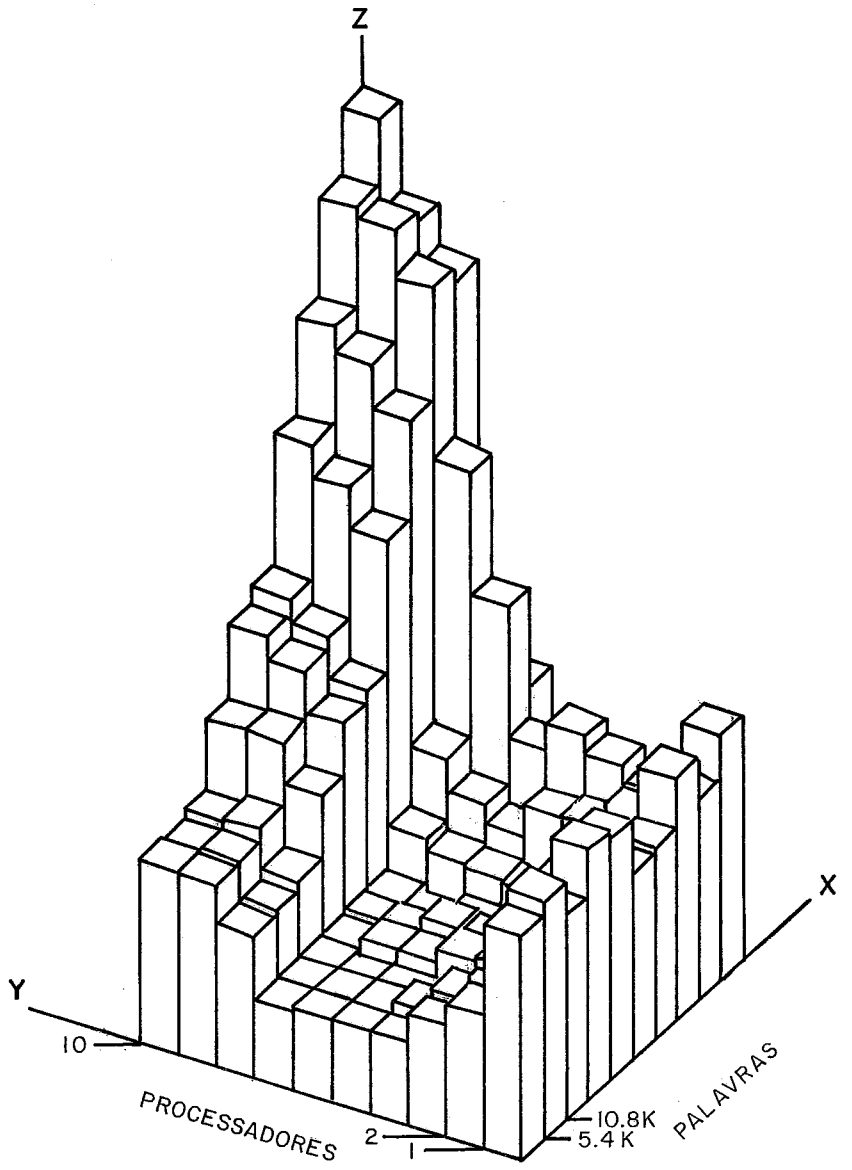
IV.1.2.1 - ANALISE DOS RESULTADOS

Observamos da figura 4.4, ilustrativa da TABELA 3 que é a tabela de DEGRADAÇÃO que ao fixarmos um dos parâmetros e variarmos o segundo obteremos resultados cuja análise é válida para quaisquer outros parâmetros da tabela:

1)Ao fixarmos (P1) em 5000 palavras, (P4) em 400 palavras e variarmos (P2) entre 1 e 16 processadores:

PROCESSADORES DO SISTEMA ACP	TAMANHO MÉDIO DE UM EVENTO (KBYTES)									
	5	10	15	20	25	30	35	40	45	50
16	.	.	.	.	.	.	.	.	.	.
15	.	.	.	.	.	.	.	.	.	.
14	.	.	.	.	.	.	.	.	.	.
13	.	.	.	.	.	.	.	.	.	.
12	.	.	.	.	.	.	.	.	.	.
11	.	.	.	.	.	.	.	.	.	.
10	16.997	16.801	17.856	25.266	32.957	35.012	48.662	59.026	69.453	76.728
9	16.617	16.97	18.391	25.083	30.546	32.078	45.792	56.283	68.246	67.022
8	12.622	12.904	14.226	20.839	26.214	27.651	41.281	51.715	63.572	62.253
7	5.777	6.026	6.065	6.372	6.956	6.958	11.332	17.753	45.651	25.24
6	6.448	6.315	5.456	7.287	7.349	6.008	10.049	13:994	32.637	22.524
5	6.1	6.376	5.701	7.403	8.681	7.161	9.723	11.993	18.859	14.876
4	6.148	7.393	5.781	8.827	8.253	6.202	9.363	14.961	21.501	15.375
3	9.414	9.949	7.966	9.124	9.299	6.938	12.035	15.544	19.854	14.84
2	11.581	12.185	9.594	12.349	12.433	9.017	12.865	17.48	17.949	15.299
1	21.117	22.917	19.093	23.823	22.773	17.253	17.652	23.069	18.553	23.3

TABELA 3
TABELA DOS RESULTADOS DA AVALIAÇÃO "DEGRADAÇÃO" DO SISTEMA ACP PARA O PROCESSAMENTO DE 100 EVENTOS EM FUNÇÃO DO NÚMERO DE PROCESSADORES DO SISTEMA (LINHAS) E DO TAMANHO MÉDIO DE UM EVENTO K PALAVRAS (COLUNA)



X = TAMANHO MÉDIO DE UM EVENTO SENDO TRANSMITIDO
Y = NÚMERO DE PROCESSADORES DO SISTEMA ACP
Z = DEGRADAÇÃO PARA 100 EVENTOS

FIG. 4.4 – GRÁFICO REPRESENTANTE DA TABELA 3

1.1)Para um sistema com apenas um processador (P2=1), a DEGRADAÇÃO é grande:

Para a execução de um programa no Sistema ACP é gerado um processo, o RMIMOM, cuja finalidade principal é a de realizar pooling nos registros de estados dos processadores para gerar uma tabela de estados.

O programa executado pelo computador hospedeiro cuja finalidade é a de realizar as operações de E/S do Sistema ACP solicita os processadores ao processo RMIMOM e transfere os eventos para BUFFERs alocados pelo RMIMOM para que este realize a transferência dos eventos.

O tempo gasto para a comunicação entre o processo executado pelo processador hospedeiro e o processo RMIMOM pode ser considerado independente do número de processadores do sistema o que torna um Sistema ACP com apenas 1 processador bastante ineficaz como podemos ver pela figura 4.4.

1.2) Ao aumentarmos o número de processadores (P2),a DEGRADAÇÃO diminui. O tempo gasto com a comunicação do processo RMIMOM passa a ser proporcionalmente menor que o tempo total de processamento.

1.3) A DEGRADAÇAO se estabiliza por volta de 3 processadores. Neste caso o tempo gasto com a cominicação do processo RMIMOM passa a ser desprezivel em comparação com o tempo total de processamento e os

sistemas de 3 e 7 processadores irão apresentar proporcionalmente a mesma DEGRADAÇÃO.

1.4) A partir de 7 processadores a DEGRADAÇÃO volta a aumentar. Isto se deve ao fato de o computador não estar mais sendo capaz de atender todos os pedidos de E/S no tempo desejado, isto gera tempo de espera no sistema e consequentemente um aumento na DEGRADAÇÃO.

2) Ao fixarmos (P2) em 8 processadores e variarmos (P1) de 5000 a 50000 palavras , e (P4) de 400 a 4000 palavras:

2.1) A DEGRADAÇÃO se manterá constante até (P1)=15000 e (P4)=1200. Podemos concluir que para eventos menores que 15000 palavras o uVAXII gasta o mesmo tempo de carregamento. Ou melhor, os processadores demoram mais tempo processando o evento do que o uVAXII carregando os eventos, de maneira que ele fica parte do seu tempo ocioso.

2.2) A medida em que aumentamos o tamanho do evento sendo transmitido (P1) e do resultado sendo coletado (P4), a DEGRADAÇÃO aumenta. Isto se deve ao fato de que medida em que se aumenta o tempo de comunicação de cada evento transmitido o uVAXII começa a não mais ser capaz de atender os processadores no tempo desejado.

2.3) Podemos observar também que o efeito se torna desprezivel para sistemas com poucos processadores (P2<3), porque nesta situação os processadores estão

solicitando atendimento do uVAXII abaixo da sua capacidade.

IV.1.2.2 - Determinação do tempo de transmissão de blocos de dados entre o uVAXII e um processador

Determinação de Tc(P1+P4)

Para a determinação do tempo ideal de processamento tivemos que avaliar o tempo médio de transmissão de um evento do computador hospedeiro para o Sistema ACP , ou seja TC(P1+P4).

Desenvolvemos um programa de simulação (PROGRAMA TC.FOR/APENDICE D) , que avalia o tempo total de processamento da transferência de um grande número de eventos, ou seja de 100 eventos para apenas um processador. O processador simula um processamento de tempo zero para o evento recebido (PROGRAMA NODETC.FOR/APE7NDICE D). O valor obtido do tempo total de transmissão dos 100 eventos é então dividido por 100, fornecendo assim o valor de TC(P1) e armazenado na tabela 4.

Implementamos o mesmo programa de simulação, para avaliar o tempo total de processamento da coleta de um evento, TC(P4), e armazenamos o resultado na TABELA 5.

Verificamos que os resultados das tabelas 4 e 5 são aproximadamente iguais, logo temos que:

TABELA 4	TABELA 5	TABELA 6
tc (5K) = 0. 166094	0. 14	0. 15
tc (10K) = 0. 217305	0.20	0.22
tc (15K) = 0. 268047	0.27	0. 29
tc (20K) = 0. 344961	0.32	0.36
tc (25K) = 0. 409063	0.40	0.42
tc (30K) = 0. 477537	0.45	0. 56
tc (35K) = 0. 537461	0.51	0.58
tc (40K) = 0. 577969	0.57	0.72
tc (45K) = 0. 655156	0.64	0.69
tc (50K) = 0. 723789	0.70	0.80
tc (55K) = 0. 81	0.76	0.91
tc (60K) = 0. 95	0.81	0.96
tc (65K) = 0. 96	0.94	1.07
tc (70K) = 1. 18	0.97	1. 10

TABELA 4 — PARÂMETRO (PI) VARIANDO DE 5000 À 70000 PALAVRAS

TABELA 5 — PARÂMETRO (P4) VARIANDO DE 400 À 5600 PALAVRAS

TABELA 6 — SOMATÓRIO DO PARÂMETRO (PI) COM O PARÂMETRO (P4) (VARIAÇÃO DE 5400 PALAVRAS À 75600 PALAVRAS)

TC(P1) + TC (P4) ~ TC(P1+P4)

Implementamos o programa de simulação para avaliar o tempo total de processamento da transferência de 100 eventos de tamanho (P1) e da coleta de 100 resultados de tamanho (P4)

para apenas um processador. O parâmetro P1 assume valores de 5000 a 50000 palavras e o parâmetro P4 valores de 400 a 4000 palavras. O valor obtido do tempo total de transmissão fornece o valor de TC(P1+P4) e é armazenado na tabela 6.

IV.1.3 - ESTUDO DA "ACELERAÇÃO" DO SISTEMA ACP

Podemos realizar o estudo do desempenho do Sistema ACP por um segundo método. Neste segundo método iremos determinar quantas vezes o Sistema ACP de (P2) processadores é mais rápido que um Sistema ACP com apenas 1 processador ao alterarmos o parâmetro de tamanho médio de um evento (P1), o parâmetro do tamanho médio do resultado coletado (P4) e fixarmos o parâmetro de tempo médio de processamento de um evento (P3).

Definimos anteriormente que a relação entre o tempo de processamento em um Sistema ACP com 1 processador e o tempo de processamento em um sistema com (P2) processadores é chamada de ACELERAÇÃO do sistema com (P2) processadores.

Utilizaremos os valores fornecidos pela TABELA 1, dividiremos o tempo total de processamento em um sistema de P2 processadores, TABELA1(P1,P2,P3,P4), pelo tempo total de processamento no sistema de um processador,

TABELA1(P1,1,P3,P4), para cada um dos valores assumidos pelo número de processadores do Sistema ACP (P2), pelo tamanho médio de um evento(P1), pelo tamanho médio do resultado coletado, e armazenaremos o resultado na TABELA 2.

TABELA 2

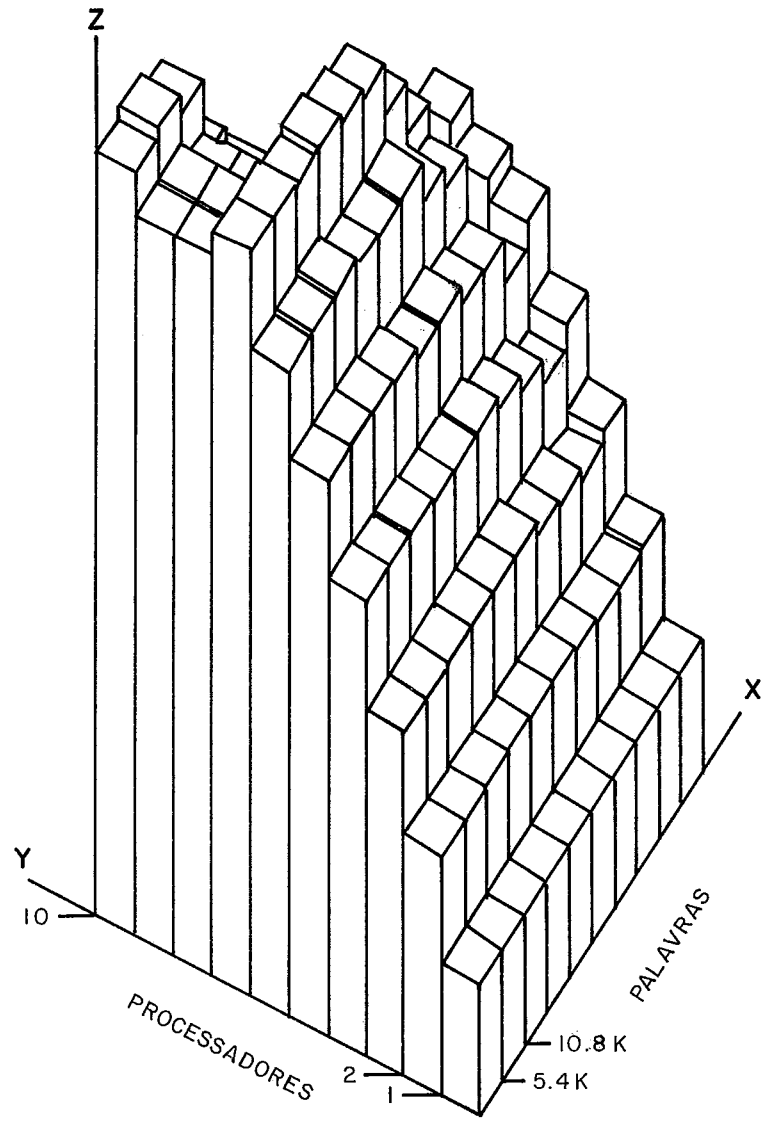
Um gráfico ilustrativo desta tabela encontra-se na figura 4.5. Nesta tabela são armazenados os valores dos resultados de ACELERAÇAO do Sistema ACP . Os valores dos parâmetros (P1), (P2), (P3) e (P4) são os mesmos assumidas para a geração da TABELA 1 pelas mesmas razões apresentadas na seção IV.1.1. As colunas da tabela correspondem aos valores do número de processadores no Sistema ACP (P2), variando de 1 a 16. As linhas da tabela correspondem ao tamanho médio do bloco de palavras do evento sendo transmitido para cada processador (P1), variando entre 5000 palavras e 50000 palavras, e do tamanho médio do resultado sendo coletado (P4), variando de 400 a 4000 palavras e o tempo de processamento médio de um evento (P3) é de 3 segundos.

IV.1.3.1 - ANALISE DOS RESULTADOS

Observamos da figura 4.5, ilustrativa da TABELA 2 , tabela dos valores da ACELERAÇAO do Sistema ACP para o processamento de 100 eventos, que ao fixarmos um dos parâmetros e variarmos o segundo obteremos resultados cuja análise é válida para quaisquer outros parâmetros da tabela:

PROCESSADORES DO SISTEMA ACP	TAMANHO MÉDIO DE UM EVENTO (KBYTES)									
	5	10	15	20	25	30	35	40	45	50
16	.	.	.	.	.	.	.	.	.	.
15	.	.	.	.	.	.	.	.	.	.
14	.	.	.	.	.	.	.	.	.	.
13	.	.	.	.	.	.	.	.	.	.
12	.	.	.	.	.	.	.	.	.	.
11	.	.	.	.	.	.	.	.	.	.
10	6.747	6.851	6.667	5.9	5.218	5.024	4.22	3.829	3.444	3.302
9	6.356	6.378	6.168	5.579	5.133	4.978	4.182	3.791	3.361	3.487
8	6.356	6.378	6.168	5.579	5.133	4.978	4.182	3.791	3.361	3.487
7	6.588	6.597	6.522	6.585	6.497	6.401	5.932	5.437	3.789	4.936
6	5.677	5.728	5.754	5.658	5.641	5.682	5.336	5.115	4.002	4.598
5	4.853	4.862	4.857	4.807	4.71	4.738	4.585	4.526	4.121	4.389
4	3.955	3.917	3.95	3.865	3.881	3.912	3.781	3.622	3.362	3.623
3	2.932	2.936	2.955	2.968	2.955	2.969	2.847	2.812	2.689	2.836
2	1.987	1.991	1.999	1.995	1.988	1.995	1.954	1.935	1.907	1.961
1	1.	1.	1.	1.	1.	1.	1.	1.	1.	1.

TABELA 2
TABELA DOS RESULTADOS DA AVALIAÇÃO DA "ACELERAÇÃO DO SISTEMA ACP PARA O PROCESSAMENTO DE 100 EVENTOS EM FUNÇÃO DO NÚMERO DE PROCESSADORES (LINHAS) E DO TAMANHO MÉDIO DE UM EVENTO SENDO TRANSMITIDO EM K PALAVRAS (COLUNA)



X = TAMANHO MÉDIO DE UM EVENTO EM K PALAVRAS
Y = NÚMERO DE PROCESSADORES DO SISTEMA ACP
Z = ACELERAÇÃO DO SISTEMA ACP PARA 100 EVENTOS

FIG. 4.5 – GRÁFICO REPRESENTANTE DA TABELA 2

1.1)Um sistema com 2 processadores (P2=2) é aproximadamente duas vezes mais rápido que um sistema com 1 processador (P2=1), ou seja possui ACELERAÇÃO de 1.9.

1.2) Um sistema com 3 processadores (P2=3) é também aproximadamente igual a 3 vezes mais rápido que um sistema com 1 processador (P2=1), ACELERAÇÃO de 2.8. Esta relação se mantém constante até sistemas com 5 processadores.

1.3) A partir de 8 processadores a relação acima começa a diminuir. Isto é, um Sistema ACP dede 8 processadores (P2=8) é 6,35 maior que um sistema com (P2=1). Isto se deve ao fato de o computador hospedeiro não atender os pedidos de E/S com a taxa necessária para evitar que os processadores fiquem ociosos.

1.4) A ACELERAÇÃO atinge a saturação para 9 processadores (P2=9). Não adianta aumentarmos o número de processadores. O uVAXII atingiu o seu limite na capacidade de atender as solicitações de E/S do sistema.

2. Ao fixarmos (P2) em 8 e variarmos (P1) de 5000 a 50000 palavras e (P4) de 400 a 4000 palavras:

2.1)O valor da ACELERAÇÃO se manterá constante até o ponto em que P1 for igual a 15000 palavras Podemos concluir que para eventos menores que 15000 palavras o uVAXII gasta o mesmo tempo de carregamento. Ou melhor, os processadores demoram mais tempo processando o evento do que o uVAXII carregando os eventos, de maneira que ele fica parte do seu tempo ocioso.

2.2) A medida em que aumentamos o tamanho do evento sendo transmitido (P1 > 20000) a ACELERAÇÃO diminui. Isto se deve ao fato de que a medida em que se aumenta o tempo de comunicação de cada evento transmitido o uVAXII começa a não mais ser capaz de atender os processadores no tempo desejado. Este efeito se torna bastante acentuado a partir de (P1 > 40000) palavras, que deve ser o ponto em que o uVAXII atinge a saturação para este número de processadores.

2.3) Podemos observar que a medida em que se diminui o número de processadores do sistema (P2), a queda brusca da ACELERAÇÃO caracterizada como ponto de saturação do sistema diminui, isto se deve ao fato de o uVAXII ter menos processadores para atender e consequentemente ser capaz de transferir eventos maiores sem saturar a sua capacidade de

transferência.

2.4) Podemos observar também que o efeito se torna desprezivel para sistemas com poucos processadores (P2<3), porque nesta situação os processadores estão solicitando atendimento do uVAXII abaixo da sua capacidade.

Ao compararmos a análise realizada para o estudo de DEGRADAÇÃO (seção IV.1.2.1) e o estudo de ACELERAÇÃO (seção IV.1.3.1) iremos verificar, como era de se esperar, que são inversamente proporcionais, isto é, que a medida em que aumentamos a DEGRADAÇÃO a ACELERAÇÃO diminui e vice-versa.

IV.2 - OBTENÇÃO DE UM MODELO ANALITICO DE DESEMPENHO PARA O SISTEMA ACP

Para podermos explicar os resultados acima temos primeiro que entender a influência da arquitetura do Sistema ACP no seu desempenho.

Como descrito detalhadamente no capitulo III, no Sistema ACP grupos de processadores ficam ligados a um barramento comum obedecendo a uma estrutura "árvore", onde os barramentos que contém os processadores são ligados a um barramento "tronco" que por sua vez são ligados a um processador hospedeiro (figura 3.4). Toda operação de E/S é realizada pelo computador hospedeiro, o uVAXII , cabe aos processadores apenas a tarefa de "processar". O hospedeiro carrega o programa de processamento do evento nos processadores do sistema, a seguir carrega um evento em

cada processador para ser processado e entra em um ciclo para verificar se tem algum processador no estado para ser coletado o resultado ("ready"). A seguir verifica se tem algum processador no estado pronto para receber um novo evento (" done"), e a seguir volta para verificar se tem algum processador no estado "ready" , reinicializando o ciclo. Este ciclo se mantém até que os resultados de todos os eventos sejam coletados.

O sistema estará sendo utilizado 100% quando todos os processadores estiverem sempre processando e o computador hospedeiro estiver sempre executando as operações de E/S. Dois casos extremos podem surgir para que isto não aconteça:

P3 << TC(P1+P4) :

Tempo de processamento (P3) muito menor que tempo de transmissão onde o tempo de transmissão (TC) está em função do tamanho do evento sendo transmitido (P1) e do tamanho do resultado sendo coletado (P4). Alguns processadores terminam de processar muito antes do computador hospedeiro ter terminado de carregar eventos nos processadores no estado READY e coletar os resultados dos processadores no estado DONE . Os processadores que terminam de processar ficam disponiveis para novos eventos e ficarão ociosos até que o hospedeiro tenha carregado os eventos em todos os processadores do sistema. Tal sistema está sendo sub-utilizado pois os processadores passam parte do seu tempo ociosos.

P3 >> TC(P1+P4) :

Tempo de processamento muito maior do que o tempo de transmissão. O processador termina de processar muito tempo depois que o hospedeiro tenha terminado de transmitir os eventos para os processadores no estado READY e coletar os resultados dos processadores no estado DONE. Neste caso o hospedeiro ficou ocioso, logo o sistema tem a capacidade de suportar mais processadores.

Podemos então deduzir que o sistema estará fornecendo o seu desempenho ótimo quando o tempo de processamento de um evento (P3) for igual ao número de processadores do sistema (P2) vezes o tempo de transmissão de cada evento (TC(P1+P4)). Ou seja:

$$P3 = P2 * TC(P1+P4) \qquad \text{Equação 4.1}$$

IV.2.1 - COMPROVAÇÃO DO MODELO

Para comprovar a validade do modelo, iremos aplicar o modelo nos resultados da tabela 3 que são os resultados obtidos da análise DEGRADAÇÃO do sistema (seção IV.1.2).

obs: podemos fazer a mesma comprovação de maneira análoga com os resultados da tabela 2 que são os resultados obtidos pela análise da ACELERAÇÃO do sistema (seção IV.1.3).

1. Tomaremos como primeiro exemplo o caso em que (P3) = 3 segundos, (P1)= 5000 palavras e (P4)=400 palavras. A tabela 3 apresenta aproximadamente o mesmo resultado a partir de (P2)= 9 processadores, consideramos então como sendo o número de processadores em que o sistema atinge a saturação (seção IV.1.2.1)

   Isto implica que o tempo médio de transmissão de um evento de tamanho médio de (P1)=5000 palavras (TC(5000)), resultado de processamento de (P4)=400 palavras, e de tempo de processamento médio de (P3)=3 segundos no qual o sistema satura (segundo o modelo da equação 4.1) será TC(5k5) = 0.3 segundos.

   A tabela 4 contém os resultados obtidos do programa de avaliação do tempo de transmissão de um evento em função do tamanho do evento (P1) e do tamanho do resultado (P4), segundo esta tabela TC(5000,400)=0,15 segundos.

   Ao compararmos o resultado fornecido pelo modelo da equação 4.1 com o resultado fornecido pela tabela 4 iremos concluir que o modelo fornece um tempo de transmissão duas vezes maior que o tempo de transmissão da tabela 4.

2. Tomemos como segundo exemplo os resultados da tabela 3 para os parâmetros: (P1)=20000 palavras, (P4)=1600 palavras e (P3)=3 segundos. Neste caso os resultados passam a ter aproximadamente o mesmo valor a partir de (P2)=9 processadores. Iremos então considerar como

parâmetros onde o sistema atinge a saturação e aplicar o modelo da equação 4.1 obtendo TC(21k6)=0,3segundos.

Segundo o resultado da tabela 4, teremos Tc(20000,1600)=0,36segundos, que é um resultado bastante próximo do resultado fornecido pelo modelo.

Ao comparar os dois exemplos iremos verificar que o modelo fornece o mesmo tempo de comunicação de para TC(5k5) e para TC(21k6).

Podemos então concluir que o modelo não é satisfatório, e por estar fora dos objetivos desta tese, deixaremos como sugestão para futuros estudos o desenvolvimento de um novo modelo analítico de desempenho do Sistema ACP.

OBS:

O sistema analisado é um Sistema ACP não multi-usuário, utilizando o "device driver" do uVAXII para o módulo de interface " DMA " , o DRV11 , utilizando a interface entre DRV11 e o Barramento Branch , o " Branch Bus Controler " e realizando a leitura de dados em disco.

IV.3 - AVALIAÇÃO EXPERIMENTAL DO MULTIPROCESSADOR ACP EM PROBLEMAS CONTENDO COMUNICAÇÃO ENTRE OS PROCESSADORES

A segunda parte deste capitulo tem como finalidade avaliar o desempenho do Sistema ACP em aplicações onde existe comunicação entre os processadores, isto é onde cada processador processa uma série de sub-tarefas que fazem parte de uma única tarefa global.

Para fazer a simulação de tais problemas, iremos dividir uma tarefa em (N) sub-tarefas para serem distribuidas entre (M) processadores. Iremos supor que para o processamento de cada sub-tarefa será necessário a leitura de um bloco de dados em um processador vizinho, o que corresponde a um acesso ao barramento VME.

IV.3.1 - DESENVOLVIMENTO DO PROGRAMA DE SIMULAÇÃO

Iremos considerar três parâmetros como fundamentais para a determinação do desempenho do sistema: o tempo de comunicação, ou tamanho do bloco de transmissão de cada sub-tarefa (P1), o número de processadores do Sistema ACP (P2) e o tempo de processamento de cada sub-tarefa (P3).

O programa de simulação desenvolvido irá avaliar o tempo total de processamento de um determinado número de sub-tarefas, dado o tempo de processamento (P3), o tamanho do bloco de transmissão de cada sub-tarefa (P1), e o número de processadores do Sistema ACP (P2).

A configuração atual do Sistema ACP do CBPF permite apenas a comunicação entre processadores pertencentes ao mesmo barramento VME, de maneira que o estudo só pode ser realizado com um número máximo de 18 processadores, e devido a este mesmo motivo o sistema atual oferece maior capacidade de comunicação quando configurado com poucos processadores. Iremos realizar o estudo em um sistema com 4 processadores (P2=4) para obter resultados mais próximos de um Sistema ACP atualizado.

A execução de uma sub-tarefa, corresponde a fazer com que o processador alterne entre dois estados: acesso a uma memória remota via o barramento VME, e processamento (com acesso a memória local). Necessitamos fazer com que os processadores alternem entre os dois estados um grande número de vezes para que possamos avaliar como o sistema se comporta dada relação entre estes dois estados. Iremos considerar a execução de 1.000 sub-tarefas por processador, um número grande o suficiente para determinarmos o comportamento do sistema. Consequentemente, a tarefa a ser

processada pelos processadores irá conter 4.000 sub-tarefas para serem processadas em um Sistema ACP de 4 processadores.

Após um série de testes para determinar quais os valares dos parâmetros a serem adotados para o programa de simulação decidimos variar (P1) entre os valores: (5), (5.005), (10.005), (15.005) e (20.005) palavras, e (P3) entre os valores: (0,32), (2), (3,6), (0,52) e (6,8) segundos por fornecerem melhores resultados para análise.

Como visto no CAPITULO III, para a implementação do programa de simulação no Sistema ACP, será necessário gerar um programa de simulação a ser executado pelo computador hospedeiro e um programa de simulação a ser executado pelos processadores do Sistema ACP.

IV.3.1.1 - ALGORITMO DO PROGRAMA EXECUTADO PELO COMPUTADOR HOSPEDEIRO

A finalidade deste programa (PROGRAMA M4.FOR/APENDICE E) é a de implementar as funções do computador hospedeiro, isto é, a de executar as operações de E/S do Sistema ACP. Em primeiro lugar este programa irá executar o carregamento do sistema operacional(o LUNI) e do programa de simulação

do processamento da tarefa a ser executada pelos processadores do Sistema ACP. A seguir irá determinar o valor dos parâmetros de tempo de processamento de cada sub-tarefa (P3)e do tamanho do bloco a ser lido por cada sub-tarefa (P1) para enviar aos processadores, inicializar o processamento das sub-tarefas pelos processadores do Sistema ACP, esperar o término do processamento dos processadores do sistema e voltar a determinar os novos valores dos parâmetros (P1) e (P3).

Para isto iremos fixar o tempo de processamento médio de cada sub-tarefa (P3), e variar o tamanho médio do bloco de comunicação de cada sub-tarefa (P1) entre os valores: (5), (255), (505), (755) e (1055) palavras e a seguir fixar o valor de (P1) e variar (P3) entre os valores: (0,032), (0,039), (0,045), (0,052) e (0,059) segundos

ALGORITMO

1. Inicializar o programa executado pelos processadores do Sistema ACP.

2. Carregar nos processadores do Sistema ACP o valor da variável do tamanho do bloco médio de cada sub-tarefa (P1) e da variável do tempo de processamento médio de cada sub-tarefa (P3).

3. Inicializar a contagem de tempo de processamento

4. Inicializar a execução das sub-tarefas nos processadores

5. Esperar que todos os processadores terminem de processar.

6. Determinar o tempo total de processamento.

7. Armazenar os resultados dos tempos de processamento na TABELA 7:

8. Gerar um novo valor para a variável de tempo médio de processamento (P3) e voltar para o item(1) caso (P3<6,8segundos) ou seguir para o item(9) caso contrário.

9. Gerar um novo valor para a variável de tamanho médio de bloco de comunicação (P1) e retornar ao item(1) caso (P1<20.005 palavras) ou seguir para o item(10) caso contrário.

10. Fim

TABELA 7

Um gráfico representante desta tabela encontra-se na figura 4.6. Nesta tabela são armazenados os resultados do programa de simulação de 4.000 sub-tarefas. As linhas estão em função do número de palavras lidas por sub-tarefa (P1), assumindo os valores: (5), (255), (505), (755) e (1055) palavras e as colunas o tempo de processamento de cada sub-tarefa (P3), assumindo os valores : (0,032), (0,039), (0,045), (0,052) e (0,059) segundos

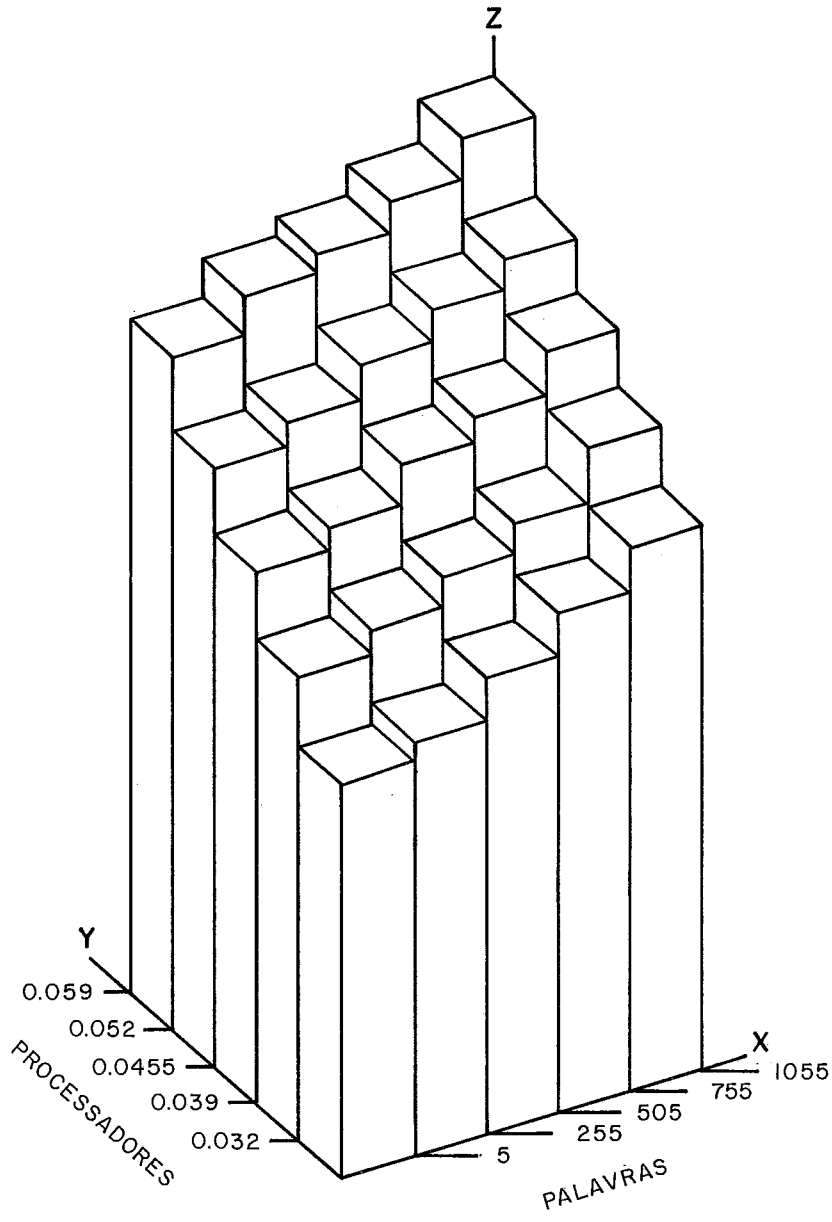
FIGURA 4.6 Esta é a figura dos resultados do tempo total de processamento das 4.000 sub-tarefas em função do número de palavras lidas no vizinho por sub-tarefa, (P1) e do tempo de processamento por sub-tarefa,(P3), onde:

1. x = (P1)= número médio de palavras lidas no processador vizinho por sub-tarefa
   assume os valores: (5), (255), (505), (755) e (1055) palavras

2. y=(P3)= tempo de processamento por sub-tarefa
   assume os valores: (0,032), (0,039), (0,045), (0,052) e (0,059) segundos

TEMPO DE PROCESSAMENTO POR SUB-TAREFA	NÚMERO MÉDIO DE PALAVRAS LIDAS NO PROCESSADOR VIZINHO POR SUB-TAREFA				
	5	255	505	795	1050
0.059	63.52	67.31	68.974	71.88	76.055
0.052	56.24	58.579	62.133	65.066	67.987
0.045	49.74	51.964	55.943	58.404	62.611
0.039	43.099	45.626	48.404	51.363	56.896
0.032	36.466	38.237	42.357	46.419	50.874

TABELA 7 – TABELA DOS RESULTADOS DO TEMPO TOTAL DE PROCESSAMENTO DE UMA TAREFA COMPOSTA POR 4000 SUB-TAREFAS POR UM SISTEMA ACP DE 4 PROCESSADORES EM FUNÇÃO DO NÚMERO MÉDIO DE PALAVRAS LIDAS NO PROCESSADOR VIZINHO POR SUB-TAREFA (COLUNAS) E DO TEMPO DE PROCESSAMENTO MÉDIO POR SUB-TAREFA (LINHAS)



X = NÚMERO MÉDIO DE PALAVRAS LIDAS NO PROCESSADOR VIZINHO POR SUB-TAREFA

Y = TEMPO MÉDIO DE PROCESSAMENTO DE UMA SUB-TAREFA

Z = TEMPO TOTAL DE PROCESSAMENTO DE 4000 SUB-TAREFAS EM UM SISTEMA ACP DE 4 PROCESSADORES

FIG. 4.6 — GRÁFICO REPRESENTANTE DA TABELA 5

3. z = tempo total de processamento de 4.000 sub-tarefa para um sistema ACP de (P2=4) processadores em função de x e y.

4. Número de sub-tarefas por processador = 1.000

5. Número de processadores=4

IV.3.1.2- ALGORITMO DO PROGRAMA EXECUTADO PELOS PROCESSADORES DO SISTEMA ACP

A finalidade deste programa (PROGRAMA NODEM4.FOR/APENDICE E) é a de simular o processamento de uma tarefa composta por 1.000 sub-tarefas a serem processadas em um processador do Sistema ACP. Cada sub-tarefa possui um tempo de processamento e um bloco de palavras para ser lido no processador vizinho.

Para cada sub-tarefa serão gerados os valores do tempo de processamento e do número de palavras a serem lidas na memória remota.

Os valores são gerados por uma funçào randômica. A função gera um valor entre os limites zero e (P), onde (P) é um dos parâmetros fornecido pelo computador hospedeiro. De maneira que no final da execução das 1000 sub-tarefas cada processador terá processado em média um

um tempo de (1000* P3) segundos e terá lido na memória remota uma média de (1000 * P1)palavras, que são situações mais próximas de problemas reais.

ALGORITMO

1. Gerar a variável de tempo de processamento.
2. Gerar a variável de número de palavras lidas na memória remota.
3. Gerar um "loop" de processamento com o valor da variável fornecida pelo ítem (1).
4. Realizar uma leitura de um bloco de dados do processador vizinho cujo tamanho do bloco é fornecido pelo variável gerada pelo ítem (2).
5. Voltar ao ítem (1) até completar 1.000 voltas.
6. Fim

IV.3.1.3 - ANALISE DOS RESULTADOS

Observe que cada processador irá processar 1.000 sub-tarefas de maneira que ao tomarmos como exemplo os primeiros valores dos "loops" teremos cada processador realizando uma transferência média de 5.000 palavras em 320 seg, ou seja em uma taxa de transferência de 156,25

palavras por segundo, e um sistema de 4 processadores estará realizando uma taxa de transferência de 625 palavras por segundo.

Dentre os testes realizados com outros parâmetros para (P1) e (P3), obtivemos dois casos extremos:
1)Caso em que o valor médio do tamanho do bloco assume o valor 5 palavras e o valor médio do tempo de processamento assume o valor 6,8seg. Cada processador transfere em média 0,7 palavras por segundo. E a taxa de transferência média de um sistema com 4 processadores será de de 2,9 palavras por segundo.

2)Caso em que o valor médio do tamanho do bloco assume o valor 102.000 e o valor médio do tempo de processamento assume o valor 0,032seg. Cada processador tranfere em média 3.138.462 palavras por segundo, ou 12,55 Mbytes/segundo.

Para avaliarmos esses resultados precisamos conhecer melhor a velocidade de comunicação no Sistema ACP através do barramento VME:

A frequência do relógio do barramento VME(SYSCLK) é de no máximo 16Mhz para as linhas paralelas e de no máximo 32Mhz para as linhas seriais (SERCLK). Caso seja feita uma transferência de dados a cada ciclo do relogio, ela poderá atingir a uma taxa de até 64Mbytes por segundo.

Os processadores do Sistema ACP foram configurados para serem capazes de realizar a escrita de uma palavra em sua memória local em 180ns, ou seja a uma taxa de 5M palavras/segundo ou 20Mbytes/seg. Para realizar o acesso à memória remota localizada em um processador vizinho o processador tem que passar por circuitos de arbitragem e de interface ao barramento VME, estes circuitos geram um acréscimo no tempo de transmissão. Se considerarmos o acréscimo como sendo de mais um ciclo de relógio teremos um tempo de acesso minimo de 240ns para a leitura e de 300ns para a escrita, ou seja de 16Mbytes para a escrita e de 13Mbytes para a leitura na memória remota.

E importante também ressaltar que para a execução de uma instrução o microprocessador MC68020 necessita de vários ciclos de relógio. Por exemplo, para a execução uma instrução (MOV D4,(A1)) ele necessita, no melhor caso, de 4 ciclos de relógio [17], de maneira que a taxa de transmissão deve ser menor que 16Mbytes/segundo.

Obtivemos uma taxa máxima de 12Mbyte/seg por processador em um sistema de 4 processadores. Estes resultados são bastantes satisfatórios se levarmos em consideração que a taxa máxima de leitura por processador é de 16Mbytes/seg se tomarmos como referência apenas a capacidade de transmissão com que os processadores podem transmitir no barramento VME.

Na realidade os processadores irão transmitir a uma taxa muito menor porque para a execução de uma leitura ou escrita na memória de um processsador vizinho é necessário executar várias instruções (tais como a chamada a subrotina "acpsup" para que o processador entre no modo supervisor e a subrotina "acpusr" para que o processador volte ao modo usuário depois de efetuar o acesso a memória vizinha) e para a execução de uma instrução de leitura ou de escrita são necessários vários ciclos de relógio.

IV.3.2 - ESTUDO DA "DEGRADAÇAO" DO SISTEMA ACP

A diferença entre um tempo de processamento da TABELA 7 e o tempo de processamento ideal de uma tarefa, dados o valor médio do tempo de processamento e o valor médio do tamanho do bloco de transmissão, é armazenada na TABELA 8:

TABELA 8

Um gráfico representante desta tabela encontra-se na figura4.7 Nesta tabela são armazenados os valores da DEGRADAÇAO do Sistema ACP em função do número de palavras lidas por sub-tarefa (linhas) e do tempo de processamento de cada sub-tarefa (colunas), para um Sistema ACP com 4 processadores.

FIGURA 4.7

Esta é a figura da diferença entre o tempo de processamento real das 4.000 sub-tarefas e o tempo de processamento ideal em um Sistema ACP de 4 processadores em função de (P1) e de (P3) onde :

1. x=(P1) = número médio de palavras lidas no processador vizinho por sub-tarefas.
   assume os valores: (5), (255), (505), (755) e (1055) palavras

2. y=(P3) = tempo de processamento por sub-tarefas
   assume os valores: (0,032), (0,039), (0,0455), (0,052) e (0,059) segundos

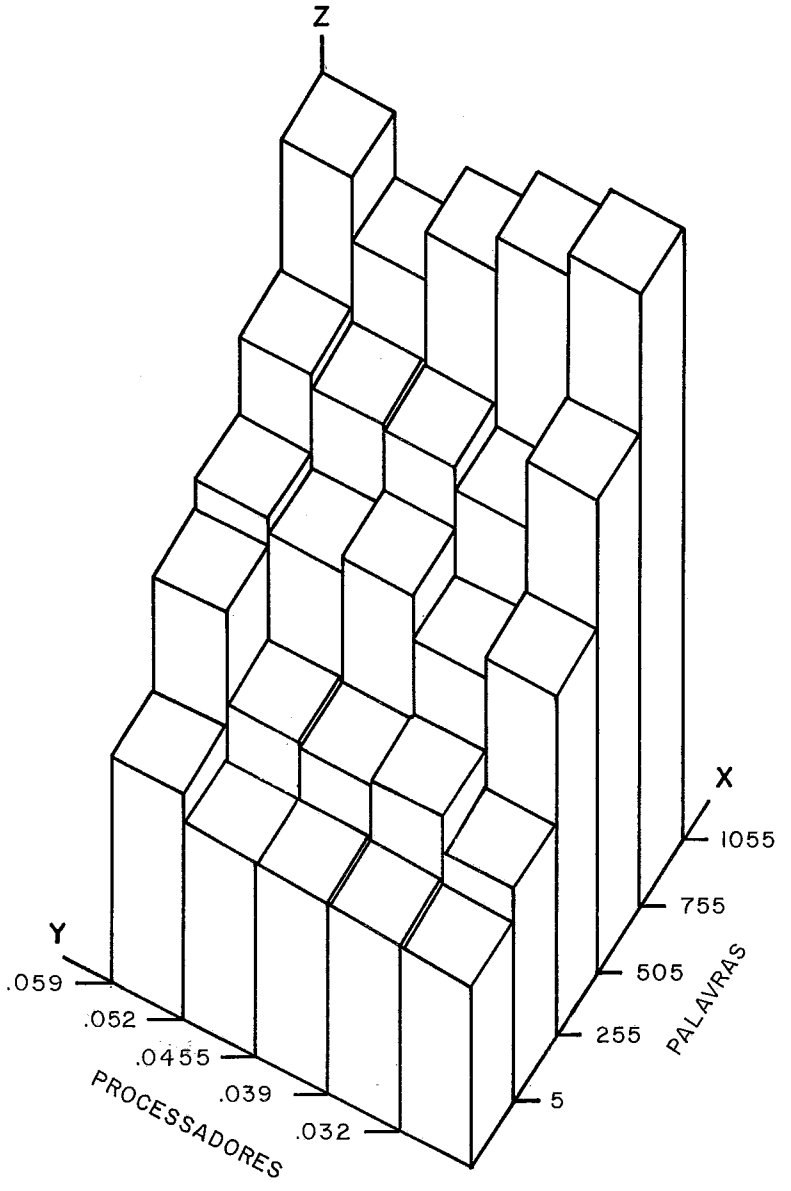
3. z =temp(x,y)-tp-tc
   ou seja, DEGRADAÇÃO do Sistema ACP de 4 processadores para processar 4.000 sub-tarefass em função de x e y.
   onde:

   - temp(P1,P3)=tempo total de processamento obtido da TABELA 7

   - tp= (P3) * 1.000
     (tempo de processamento médio ideal de 1.000 sub-tarefas)

TEMPO DE PROCESSAMENTO POR SUB-TAREFA	NÚMERO MÉDIO DE PALAVRAS LIDAS NO PROCESSADOR VIZINHO POR SUB-TAREFA				
	5	255	505	795	1005
0.059	5	7.79	8.454	10.36	13.535
0.045	4.22	5.559	8.113	10.046	11.967
0.039	4.22	5.444	8.423	9.884	13.091
0.037	4.079	5.606	7.384	9.343	13.876
0.032	3.946	4.717	7.837	10.899	14.354

TABELA 8 – TABELA DOS RESULTADOS DA AVALIAÇÃO DA DEGRADAÇÃO DO SISTEMA ACP DE 4 PROCESSADORES PARA PROCESSAR 4000 SUB-TAREFAS EM FUNÇÃO DO NÚMERO DE PALAVRAS LIDAS NO PROCESSADOR VIZINHO POR SUB-TAREFA (COLUNAS) E DO TEMPO MÉDIO DE PROCESSAMENTO POR SUB-TAREFA



X = NÚMERO MÉDIO DE PALAVRAS LIDAS NO PROCESSADOR VIZINHO POR SUB-TAREFA

Y = TEMPO MÉDIO DE PROCESSAMENTO DE UMA SUB-TAREFA

Z = "DEGRADAÇÃO" DE UM SISTEMA ACP DE 4 PROCESSADORES PARA O PROCESSAMENTO DE 4000 SUB-TAREFAS

FIG. 4.7 – GRÁFICO REPRESENTANTE DA TABELA 6

- tc=(P1) *1.000*0,0000004
  (tempo de transmissão médio ideal de 1.000 tarefas, onde
  0,000004 seg= Tempo médio de transmissão de uma palavra (avaliado pelos programas T.FOR e NODET.FOR/ APENDICE G)

4. Número de sub-tarefas por processador = 1.000

5. Número de processadores=4

OBS:

Os sulcos observados na figura são causados pelo fato de a fator erro utilizada para o cálculo do tempo ideal de processamento não ser uniforme.(APENDICE E)

IV.3.2.1 - ANALISE DOS RESULTADOS

Podemos observar que:

1. Ao fixarmos o tempo de processamento e aumentarmos o tempo de comunicação a DEGRADAÇÃO aumenta.

2. Se fixarmos o tempo de comunicação e aumentarmos o tempo de processamento a DEGRADAÇÃO diminui.

Para explicarmos a causa da influência dos parâmetros de tempo de processamento de cada sub-tarefa e do tamanho bloco lido de cada sub-tarefa no resultado da DEGRADAÇÃO do Sistema ACP, temos que entender o mecanismo de comunicação entre os processadores do sistema:

O barramento de comunicação entre os processadores é tal que apenas um processador pode ter acesso por vez. Desta maneira parte do tempo de comunicação é gasto na espera da obtenção do barramento, e parte do tempo é gasto na transferência do bloco. Duas situações extremas podem ocorrer para alterar o desempenho do sistema:

Na primeira situação extrema o tamanho dos bloco lido pelos processadores em cada sub-tarefa é muito grande, de maneira que eles gastam a maior parte do seu tempo de processamento esperando a obtenção do barramento, isto é, eles ficam ociosos durante parte do tempo de processamento da sub-tarefa; isto aumenta a DEGRADAÇÃO do sistema.

Na segunda situação, o tamanho dos blocos lidos não é suficientemente grande para que ocorra uma disputa pelo barramento entre os processadores, e ao contrário, é tão pequeno que o barramento fica parte do seu tempo ocioso, logo o sistema está sendo utilizado com um menor número de processadores que a sua capacidade máxima de atendimento. Neste caso ao sistema pode ser acrescentado mais

Na segunda situação, o tamanho dos blocos lidos não é suficientemente grande para que ocorra uma disputa pelo barramento entre os processadores, e ao contrário, é tão pequeno que o barramento fica parte do seu tempo ocioso, logo o sistema está sendo utilizado com um menor número de processadores que a sua capacidade máxima de atendimento. Neste caso ao sistema pode ser acrescentado mais processadores, podendo ter uma diminuição na DEGRADAÇÃO. Este é o mesmo efeito gerado com o aumento o tempo de processamento de cada sub-tarefas, pois cada processador passa a solicitar o barramento com uma menor frequência e o barramento fica parte do seu tempo ocioso.

O sistema é utilizado no melhor do seu desempenho quando os processadores recebem o barramento prontamente ao solicitarem, isto é, em um sistema no qual os processadores processam 100% do seu tempo. Por outro lado o sistema também é utilizado no melhor do seu desempenho quando o seu barramento é utilizado com um número de processadores igual a sua capacidade máxima de atendimento.

IV.3.3 - ESTUDO DA "ACELERAÇÃO" DO SISTEMA ACP

Assim como na seção IV.1, podemos avaliar o desempenho do sistema por um segundo método, determinando a a ACELERAÇÃO do sistema segundo os parâmetros (P1,P2,P3):

processadores do sistema (P2), e iremos armazenar o resultado na TABELA 9 (PROGRAMA M.FOR/ APENDICE F).

Devido às mesmas razões fornecidas na seção IV.1 iremos realizar o estudo para sistemas de até 16 processadores.

Para fornecermos 1.000 subtarefas para um sistema de 16 processadores teremos que implementar uma tarefa composta por 16.000 sub-tarefas.

Iremos variar o parâmetro (P1) de acordo com os valores da seção IV.3.1.

Como iremos variar os parâmetros (P1,P2) para termos uma figura tri-dimensional, fixaremos o valor do parâmetro (P3). Iremos adotar o valor P3=0.02segundos por se tratar de um tempo de processamento por sub-tarefa suficientemente pequeno para observarmos alterações em um sistema de 16 processadores.

TABELA 9

Um gráfico que representa esta tabela encontra-se na figura 4.9. Nesta tabela são armazenados os valores de um Sistema ACP com y processadores (coluna) para processar 16.000 sub-tarefas em função do número de palavras lidas por sub-tarefa (linhas).

FIGURA 4.9

Esta é a figura dos valores da divisão entre o tempo de processamento de 16.000 sub-tarefas em um Sistema ACP com (P2) processadores, número de palavras transmitidas por sub-tarefa igual a (P1) palavras e tempo de processamento por sub-tarefa igual a (P3=0,02) segundos em um Sistema ACP com 1 processador, onde:

1. x= (P1) = número médio de palavras lidas no processador vizinho por sub-tarefa.
   assume os valores: (1), (10), (20) palavras

2. y=(P2)= número de processadores
   (P2) assume valores entre 1 e 16

3. z = ACELERAÇÃO do Sistema ACP de y processadores para o processamento de 16.000 sub-tarefas em função de x.

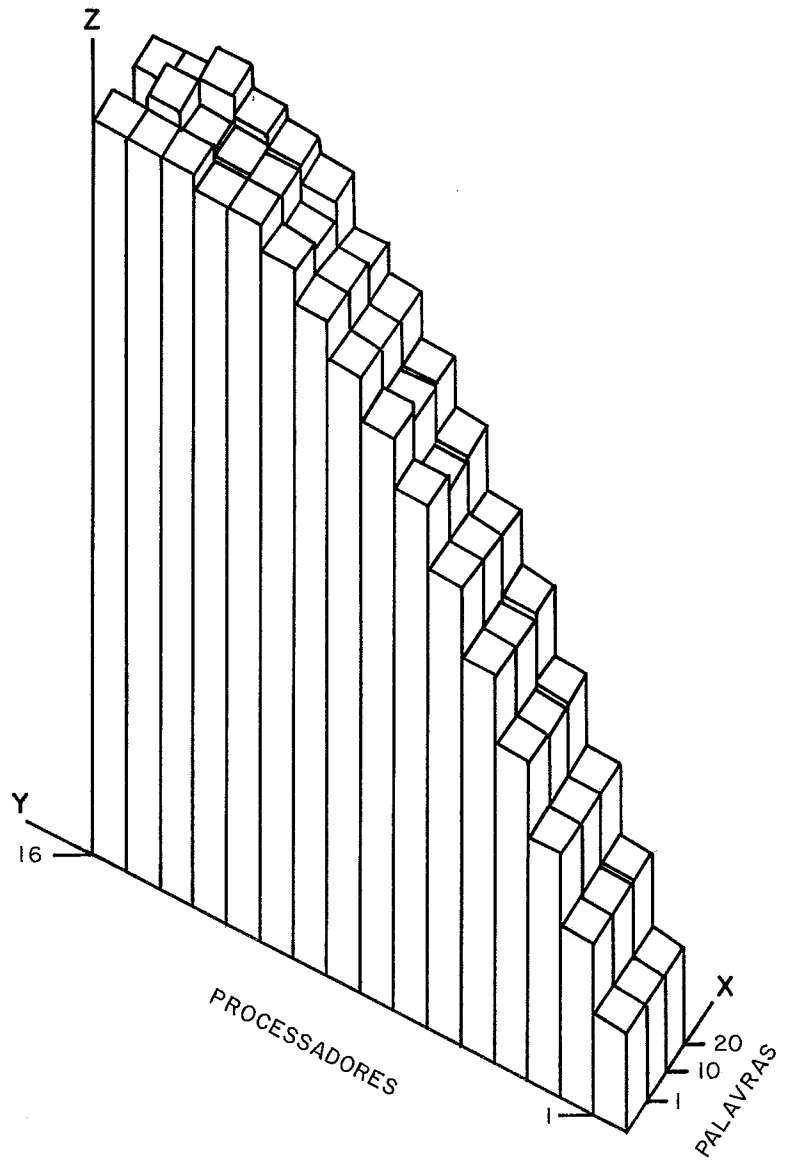
4. Número total de sub-tarefas = 16.000

5. (P3)= Tempo de processamento por sub-tarefa = 0,02 segundos

6. Tempo médio de transmissão de uma palavra=0,000004seg (apêndice A)

NÚMERO DE PROCESSADORES	NÚMERO DE PALAVRAS LIDAS NO PROCESSADOR VIZINHO POR SUB-TAREFA		
	1	10	20
16	9.302	8.717	9.198
15	9.268	9.475	9.236
14	9.268	9.146	9.52
13	9.04	9.068	9.228
12	9.045	8.996	9.042
11	8.709	8.582	8.794
10	8.264	8.251	8.108
9	7.773	7.715	7.716
8	7.23	7.059	7.102
7	6.557	6.338	6.412
6	5.7	5.655	5.582
5	4.765	4.753	4.82
4	3.872	3.81	3.859
3	2.896	2.903	2.919
2	1.95	1.942	1.984
1	1	1	1

TABELA 9 — ACELERAÇÃO DE UM SISTEMA ACP PARA O PROCESSAMENTO DE 1600 SUB-TAREFAS DE TEMPO DE PROCESSAMENTO MÉDIO DE 0.02 SEG. POR SUB-TAREFA EM FUNÇÃO DO NÚMERO DE PROCESSADORES (LINHAS) E DO NÚMERO DE PALAVRAS LIDAS NO PROCESSADOR VIZINHO (COLUNAS)



X = NÚMERO DE PALAVRAS LIDAS NO PROCESSADOR VIZINHO POR SUB-TAREFA

Y = NÚMERO DE PROCESSADORES DO SISTEMA APC

Z = ACELERAÇÃO DO SISTEMA ACP PARA O PROCESSAMENTO DE 16 000 SUB-TAREFAS DE TEMPO MÉDIO DE PROCESSAMENTO 0.02 SEG

FIG. 4.9 — GRÁFICO REPRESENTANTE DA TABELA 7

IV.3.3.1 - ANALISE DOS RESULTADOS

As causas da influência dos parâmetros: tempo de processamento por sub-tarefa, tamanho do bloco de leitura por sub-tarefa e número de processadores do sistema no resultado da ACELERAÇÃO do Sistema ACP foram apresentadas na seção IV.3.2.1.

Baseados na análise da influência do sistema de comunicação entre os processadores do Sistema ACP, podemos observar pelo gráfico da ACELERAÇÃO, figura4.9 que ao fixarmos o tempo de processamento e o tempo de comunicação e aumentarmos o número de processadores, a ACELERAÇÃO do sistema irá aumentar até atingir um patamar, que corresponde a saturação do barramento segundo o tempo de processamento isto é, o ponto em que o barramento está sendo utilizado 100% e que os processadores estão processando 100%.

A partir deste ponto ao aumentarmos o número de processadores a velocidade do sistema não irá aumentar pois o barramento não poderá atender os processadores com a taxa solicitada e os processadores passarão a ficar ociosos a espera do barramento. O barramento continuará sendo utilizado 100% mas os processadores serão subutilizados a medida em que se aumentar o número de processadores.

Podemos observar pela tabela da ACELERAÇÃO, tabela 9 que:

1. 1 processador processa 16.000 sub-tarefas em 21 segundos, isto é,a uma taxa de 0,0013segundos por tarefa.

2. 16 processadores processam as 16000 sub-tarefas em 2,17 segundos. Atribuindo 1000 sub-tarefas para cada um temos que cada processador processa uma sub-tarefa em 0,00217segundos

Podemos então observar que uma sub-tarefa em um sistema com 16 processadores demora o dobro do tempo de processamento se comparado a um sistema com apenas 1 processador. Por outro lado, o sistema torna-se aproximadamente 10 vezes mais rápido.

CAPITULO V

CONCLUSÕES

Iremos apresentar as conclusões do "Estudo Experimental do Multiprocessador ACP" na avaliação do seu desempenho para problemas para os quais ele foi projetado, ou seja, para problemas do tipo "orientado para eventos" e para a avaliação do seu desempenho para problemas com comunicação entre os processadores. Iremos comparar e explicar os resultados obtidos e a seguir apresentar outras conclusões obtidas ao longo do estudo desenvolvido.

1. Problemas do tipo "orientado para eventos"

Implementamos um programa de simulação de um problema do tipo "orientado para eventos" e obtivemos uma taxa de transferência máxima de 70kbytes/segundos, ou seja aproximadamente 9% da capacidade limite que é de 800kbytes/seg.

Avaliamos do estudo de "gargalos" do Sistema ACP , que a taxa máxima de transferência entre ele e o computador hospedeiro, é de 625Kbytes/segundo.

Podemos supor então que o "gargalo " do sistema está sendo o "software" de desenvolvimento, o ACPSYS . (OBS: De fato, nova versão do ACPSYS que ainda não pôde ser implementada no CBPF deverá aumentar a taxa de transferência do sistema em 30%, ou seja, para 100kbytes/segundo.)

2. Problemas com comunicação entre os processadores

Implementamos um programa de simulação para um problema contendo comunicação entre os processadores, e obtivemos uma taxa de transferência entre os processadores do sistema de 1Mbyte/seg.

Avaliamos do mesmo estudo de "gargalos" do Sistema ACP que a taxa de transferência máxima entre os processadores é de 4Mbytes/seg.

Concluímos que para problemas onde existe comunicação entre os processadores o sistema apresenta uma taxa de transferência de dados 25% da sua taxa de transferência máxima.

3. Comparação entre os resultados obtidos

O Sistema ACP apresenta um melhor aproveitamento da sua capacidade de taxa de transferência de dados em problemas onde ocorre comunicação entre os processadores.

Estes resultados podem ser explicados pelo fato de que as subrotinas que executam toda a comunicação entre o computador hospedeiro e o Sistema ACP , fazem parte de um pacote de software de alto nivel, o ACPSYS (seção III.2), cuja finalidade é a de tornar o mais "amigável" possivel o desenvolvimento de programas em FORTRAN IV.

Para problemas onde existe comunicação entre os processadores não foram desenvolvidas rotinas de comunicação, de forma que o usuário tem que trabalhar em um nível mais baixo de programação, tendo que por exemplo implementar endereçamento absoluto para acessar a memória remota, e implementar seus próprios semáforos [apêndice c].

Podemos concluir que ao ter sido implementado um software de alto nível, se ganhou na facilidade e na rapidez de desenvolvimento de programas, mas se perdeu no seu desempenho.

Estão sendo desenvolvidas melhorias no software e no hardware do Sistema ACP [apêndice A], com a finalidade de melhorar o seu desempenho. Para sabermos o quanto elas serão eficientes será necessário a implementação dos programas de simulação desenvolvidos.

5. OUTRAS CONCLUSÕES

5.1) Baseado nos resultados obtidos, na avaliação da DEGRADAÇÃO e da ACELERAÇÃO do Sistema ACP , conforme definidos no capitulo IV, verificamos que os parâmetros que alteram o desempenho do Sistema ACP em um problema do tipo "orientado para eventos" são :

1. O número de palavras sendo transmitidas

2. O tempo de processamento do evento

3. O número de processadores do sistema

5.2) Comprovamos que os programas de simulação são eficientes para a avaliação do desempenho do Sistema ACP .

5.3) Baseado nos resultados do estudo de desempenho, desenvolvemos um modelo analítico para o Sistema ACP , e verificamos que o modelo não era válido para alguns resultados obtidos pelo programa de simulação.

Podemos concluir que este modelo analítico desenvolvido, está deficiente para a avaliação do desempenho do Sistema ACP . E por estar fora dos objetivos desta tese, deixaremos como sugestão para futuros estudos, o desenvolvimento de um novo modelo analítico para o Sistema ACP .

E importante ressaltar que os resultados obtidos para a análise do desempenho do Sistema ACP não serão proporcionais a resultados obtidos caso sejam adotados outros parâmetros.

Consequentemente, para obtermos o comportamento do sistema para problemas com parâmetros diferentes dos adotados para o estudo da tese, como por exemplo para determinarmos o comportamento do sistema para um tempo de processamento médio de vinte segundos, (P3=20segundos), teremos que aplicar o método apresentado no capitulo IV.

REFERENCIAS

[1] THE ACP MULTIPROCESSOR SYSTEM AT FERMILAB /I.GAINES, H.ARETI, R.ATAC, J.BIEL, A.COOK, M.FISCHLER, R.HANCE, D.HUSNY, T.NASH and T.ZMUDA, Fermilab-Conf-86/132

[2] THE FERMILAB ADVANCED COMPUTER PROGRAM MULTI-PICROPROCESSOR PROJECT/ T.NASH et al., Proc. Conf. Computing in High Energy Physics, Amsterdam, June 1985 (North-Holland,Amsterdam,1986) and references therein.

[3] SOFTWARE FOR EVENT ORIENTED PROCESSING ON MULTIPROCESSOR SYSTEMS/ M.FISHCLER, H.ARETI, J.BIEL, S.BRACKER, G.CASE, I.GAINES, D.HUSBY and T.NASH, Processor and software research for High-Energy Physics, Guanajuato, Mexico (1984) p.175.

[4] Bus Switch Specification,BSS. R.ATAC/ACP/FERMILAB

[5] Acp68020 CPU module User's Manual. R.ATAC/ACP/FERMILAB

[6] E-769 Acquisition System. C.GAY/Toronto University

[7] Projeto MSPARALLEL, Colaboração IFT , Colaboração COPPE [APENDICE B]

[8] Extensões do Sistema ACP/Roberto Valois/ CBPF

[9] Segunda Geração de Processadores Paralelos do "Advanced Computer Program" (ACP)/ B.Schuze, R. Valois /CBPF

[10] "ACP User's Guide for LUNI"/Advanced Computer Program-FERMILAB/Revision 3.1 /December 1,1987.

[11] "ACP User's guide for Utilities"/Advanced Computer Program-FERMILAB/Revision 3.1 /December 1,1987.

[12] "ACP Software User's guide for Event Oriented Processing"/Advanced Computer Program-FERMILAB/Revision 2.0 /april 27,1987.

[13] BBBWI,"Branch Bus to Bus Switch Interface"/ADVANCED COMPUTER PROGRAM- FERMILAB/April 1,1987

[14] BSS,"Bus Switch Specification"/ADVANCED COMPUTER PROGRAM-FERMILAB/April 2,1987.

[15] "Branch Bus to VME Interface(BVI)"/ADVANCED COMPUTER PROGRAM-FERMILAB/March 11,1987.

[16] The VMEbus Specification/MOTOROLA/Revision C.1-October 1985

[17] "MC68020 32-Bit Microprocessor User's Manual/MOTOROLA

[18] "Le bus VME: description du fonctionement dynamique"/Minis et Micros numero 162/Dominique Girod

[19] "Q-bus to VME Interface (QVI)"/ADVANCED COMPUTER PROGRAM-FERMILAB/March 11,1987.

[20] "VMEvbus Branch Bus Controler(VBBC)"/ADVANCED COMPUTER PROGRAM-FERMILAB/March 11,1987.

[21] "VMEbus resource module(VRM)"/ADVANCED COMPUTER PROGRAM-FERMILAB/March 11,1987.

[22] "Q-bus Branch Bus Controler"/ADVANCED COMPUTER PROGRAM-FERMILAB/March 06,1987.

[23] "ACP User's guide for LUNI"/ADVANCED COMPUTER PROGRAM-FERMILAB/Revision 2.0-April 14,1987

[24] Colaboração FERMILAB [APENDICE B]

[25] Computer Architecture and Parallel Processing/ Kai Hwang Fayé, A. Briggs

[26] Uma Introdução à Computação Paralela e Distribuida /Claudio Amorim,Valmir Barbosa e Edil Fernandes

[27] VII Congresso da Sociedade Brasileira de Computação /Introdução aos Supercomputadores /Virgilio Almeida

[28] Máquinas de Cálculo Coordenado/Claudio Amorim, Ronald Shellard

[29] Operating System Concepts/James Petrson, Abraham Silberschatz

[30] Mips Dhrystones, and other tales /Datamation /july87.

[31] MIMD Computing in the USA-1984/ R.W.HOCKNEY/ Parallel Computing (1985)119-136

[32] Experiência E-769 realizada no periodo de 06/87 a 03/88 no FERMILAB/U.S.A.,

[33] Tape transport Subsystem User's Guide , TSV05 DIGITAL EK-TSV05-UG-003

[34] Maxtor Product Specification and OEM Manual / XT-2000 PIONEER

[35] DRV11-WA General Purpose DMA Interface User's Guide, DIGITAL EK-DRWA-UG-002

[36] Branch Bus Specification, BB, Advanced Computer Program/ March 6,1986

[37] Fastbus do Branch Bus Controler (FBBC)/ Advanced Computer Program 14 oct.1985/Marcus Larwill.

[38] ACP32100CPU MODULE / Advanced Computer Program oct.23,1986

[39] VME DISPLAY MODULE Advanced Computer Program jan.1986/Carla Osthoff

APÊNDICE A

DESENVOLVIMENTOS NO SISTEMA ACP

Para melhorar o Sistema ACP estão sendo conduzidas duas linhas de pesquisa. A primeira tem como finalidade desenvolver melhorias no software e no hardware do atual sistema (seção V.1) e a segunda tem como finalidade o desenvolvimento de uma nova geração compatível com a primeira mas com um poder computacional muito maior, o Sistema ACP II [35] (seção V.2).

I - MELHORIAS NO ATUAL SISTEMA

1. A última atualização do ACPSYS é a versão 1.13

2. Implementação de uma extensão de Linguangem Fortran para programação concorrente.

3. Maior comunicação entre processadores:

   - Desenvolvimento de um módulo que permite que um processador mestre no barramento VME possa ser mestre no Barramento Branch, o VBBC [20].

   - Implementação de Sistemas ACP possuindo mais de um Barramento Branch acessando um mesmo bastidor VME.

   - Desenvolvimento de um estrutura de chaveamento do Barramento Branch, o Bus Switch [4]:

- o 16 portas bi-direcionais
- o Banda de 20 Mhz para cada porta
- o A ROM de endereçamento é capaz de endereçar até 2048 processadores

- Implementação de leitoras de fita com maior taxa de transferência e de armazenamento de dados.

II - SEGUNDA GERAÇÃO DO SISTEMA ACP

1. Operação com o sistema operacional **UNIX**
2. Eliminação da dependência do computador hospedeiro:

   - Qualquer processador do sistema poderá assumir a função de mestre e gerenciar a ditribuição das tarefas de E/S.
   - As tarefas de E/S serão realizadas por vários processadores simultaneamente, através de periféricos distribuidos no sistema.
   - Os processadores do sistema poderão se comunicar sem a necessidade de intervenção do "mestre".

- Os processadores serão logicamente configurados em grupos, com dados fluindo entre os grupos.

- Qualquer processador poderá ser um "nó" conectado ao barramento Branch ou a um rede Ethernet.

3. Novas unidades de processamento:

- Os novos processadores do Sistema ACP utilzam o microprocessador MIPS de 32 bits com arquitetura RISC [39], que deverá atingir um desempenho computacional dez vezes superior ao desempenho dos atuais microprocessadores MC68020/MC68881.

- Eles obedecerão o padrão VME e poderão gerenciar interrupções externas.

- 4Mbytes de memória RAM

- Relógio de 8 Mhz e desempenho previsto de 8 MIPS.

APENDICE B

1. Projeto MSPARALLEL.Tem como objetivo o desenvolvimento de um programa para a resolução de problemas de contorno em equaçõoes diferenciais ordinárias. O algoritimo básico é um "multiple Shooting" adotado para aproveitar o processamento paralelo do computador ACP do CBPF. Este programa será incorporado a um pacote para resolução de equações diferenciais parciais.(projeto em colaboração com a Universidade Simon Bolivar,IM/UFRJ e CBPF.

2. A colaboração com o IFT visa a pesquisa em algoritimos paralelos e aplicações à física e matemática.

3. A colaboração com a COPPE está desenvolvendo dois trabalhos de pesquisa : Na área de Sistemas de Computação desenvolvendo a tese de mestrado "Uma Avaliação Experimental do desempenho do Multiprocessador ACP. por Valmir Barbosa, Claudio Amorim e Carla Osthoff.

   Na área de Engenharia Elétrica desenvolvendo a tese de mestrado "Cálculo do Fluxo de Potencia em sistemas de energia elétrica utilizando decomposição e cálculo paralelo a partir de métodos iterativos." por Eugenius Kaskurewick e por Nestor Roqueiro,

Na área **Engenharia Civil** desenvolvendo a tese de mestrado "Analise de Confiabilidade de Estruturas Reticuladas" por Luis Volnei Sudati e Edson Lima, e a tese de mestrado "Analise de grandes estruturas por particao de dominio aplicada a computacao paralela"por Marcos Siqueira e Edson Lima Sagrillo

4. No periodo de desenvolvimento inicial deste projeto,participou, pelo CBPF a engenheira CARLA OSTHOFF.

No periodo de implementação do sistema participaram pelo CBPF os engenheiros CARLA OSTHOFF e BRUNO SCHULZE.

Atualmente participa do desenvolvimento de novos projetos o engenheiro MARIANO MIRANDA pelo CBPF e, no CBPF os engenheiros CARLA OSTHOFF, BRUNO SCHULZE e VLADIMIR SALGADO o analista de sistemas ROBERTO VALOIS.

APENDICE C

```
!================================================
!     PROGRAMA  EV.UPF
!================================================
!
system=CBPF
host source=EV.for
host library=DUA0:[lib.hplot.hpldigs]hpldigs.olb
host library=DUA0:[lib.digs]gralib.olb
host library=DUA0:[lib.hbook]packlib.olb
host library=DUA0:[lib.cernlib]kernlib.olb
node source=node_ev.for
node main subroutine = wait
node blocks = 1:input, 2:output, 3:calib
class 1: node = 16 68020



c*************************************************************
c     PROGRAMA EV.for
c*************************************************************
c       este programa avalia o tempo total de proces
c       de um evento de tamanho (P1), de tempo de pr
c       (P3) em um  Sistema ACP de (P2) processadore
*************************************************************
c
c     comando necessário para a utilização das subro
c     ==============================================
c
      include 'acp$area:user_common.inc'
c
c     declaração local:
c     =====================
c
integer*4 in,out,status,n(13),iseed,iiseed,cmd
integer*4 num_in,num_out,event,node,byte_count
common/input/in(300000)
common/output/out(32000)

common/calib/wat(2)
logical nodeready,nodedone,lastevent,alldone
logical read_done,response_node
real time_host(10000),start_time,num_inl,speed(11,17
real A(5),SIG(5),CH12,rnode,wat,soma,media(11,17)
real m(11,17),s(11,17),o(11,17),overhead(11,17)
real tp,tc(10)
COMMON // HMEM(10000)
c
c       Inicializar o processo RMIMOM, carregar o LU
c       e o programa NODEEVENTOS.FOR nos processador
c       do Sistema ACP.
c      ==========================================
c
```

```
      call hplint(0)
      call acp_init
c
c
c       Abrir os arquivos de entrada e de saida
c      ==========================================
c
      call acp_input_tape_open_file(status)
      call acp_output_tape_open_file(status)
c

c
c       inicializar as tabelas para serem armazenadc
c       os resultados
c      ==========================================

call htable(1,'tp=8s $',11,1.,11.,17,1.,17.,0.)

call htable(2,'speed up $',11,1.,11.,17,1.,17.,0.)

call htable(3,'overhead $',11,1.,11.,17,1.,17.,0.)

call hbook2(4,'tp=8s $',11,1.,11.,17,1.,17.,0.)

call hbook2(5,'speed up $',11,1.,11.,17,1.,17.,0.)

call hbook2(6,'overhead $',11,1.,11.,17,1.,17.,0.)


c       Inicializar as variáveis
c      ================================

c       obs:o processamento médio
c       de um evento (P3)será: 4.48 * 0.67 =3 seg
c       onde (P3)=wat(1)
        wat(1)=4.48

c       obs:tc(1)=5k,tc(2)=10k...tc(10)=50k
c       ===================================

tc(1)=0.166094
tc(2)=0.217305
tc(3)=0.288047
tc(4)=0.344961
tc(5)=0.409063
tc(6)=0.477539
tc(7)=0.537461
tc(8)=0.577969
tc(9)=0.655156
tc(10)=0.723789
```

```
        event=100
        write(*,1)
1       format(1x,'entre o numero de proc. >>',$)
        ACCEPT 2,node
2       FORMAT (I)
        iseed=secnds(0.0)
        iiseed=secnds(0.0)*2

c        Inicializar o "loop" de variação
c       do tamanho do evento  (P1=num_in) e do tamar
c       do resultado coletado (P4= num_out)
c      ========================================

        do 400 i=1,10
       soma=0.

c      lembrar que num_out max=8000
c      ============================

        DO 3 mm=1,1

        num_in=rndm(iseed)*i*10000

        num_out=rndm(iiseed)*i*800


    read_done = .true.
    nodeready = .true.
    nodedone = .false.
    lastevent = .false.
    L = 0.

c       inicializar a contagem do
c       tempo total de processamento
c        ========================
c
start_time = secnds(0.0)


**************************************************
*
*       BROADCAST
*
**************************************************
c       carregar nos processadores o valor de (P3)

 call acp_broadcast (3,wat,1,acp_r_4)
C      start of event loop
c    ======================
10      continue
```

```
*****************************************************
*
*         READ
*
*****************************************************

 CALL ACP_READTAPE(IN,NUM_IN,READ_DONE,byte_count)

*****************************************************
*
*         SEND
*
*****************************************************
if(.not.lastevent) then
 if(nodeready) then
  l= l+1
 num_in=rndm(iseed)*i*10000
end if
 call acp_sendevent(in,num_in,nodeready)
 else
 if(acp_alldone(response_node))go to 20
end if
*****************************************************
*
*         GET
*
*****************************************************


call acp_getevent(out,num_out,nodedone)


*****************************************************
*
*         WRITE
*
*****************************************************
 if(nodedone)call acp_writetape(out,num_out)

  go to 10
*****************************************************
*       Fim do "loop" de envio de eventos
*
*****************************************************
20        continue
C     determinação do tempo total de procesamento
C     ===========================================
time_host(mm)   = secnds (start_time)
```

```
      soma=time_host(mm)+soma

3      continue



      if (node.ne.1)then

      media(i,1)=hij(1,i,1)
      end if


c     determinação da média do tempo total
c     de processamento
c     ====================================
      media(i,node)=soma/1.


c     determinação da aceleração
c     ==========================

      speed(i,node)=media(i,1)/media(i,node)


c     determinação da degradação
c     ==========================

       tp=wat(1)*0.65*event
       tc(i)=tc(i)*event
       overhead(i,1)=media(i,1)-tp-tc(i)

      if (node.ne.1)then

      overhead(i,node)=media(i,node)-overhead(i,1)

      end if
c     armazenar os resultados
c     =======================
      call hfill(1,float(i),float(node),media(i,nc

      call hfill(2,float(i),float(node),speed(i,nc

      call hfill(3,float(i),float(node),overhead(:
c     voltar ao loop
c     ==============
400   continue
```

```
c       fechar os arquivos de saida
c      ======================
c
      call acp_output_tape_close_file(status)
      call acp_input_tape_close_file(status)

c       plotar
c       ======
c
        call hplego(4,60.,60.)
        call hplot(4)

        call hplego(5,60.,60.)
        call hplot(5)

        call hplego(6,60.,60.)
        call hplot(6)

        call hplend

c       imprimir
c       ========

        call houtpu(14)

        call hprint(1)

        call hprint(2)

        call hprint(3)


c os procedimentos de saida do
c       Sistema ACP
c     ========================================
C
      call acp_exit

c     fim do programa:
c     ================
c
       stop
       end
```

```
c ========================================
c      PROGRAMA    NODE_EV.FOR
c ========================================

      subroutine wait

C     declaração local
c     ==================

      common/INPUT/in(300000)
        common/OUTPUT/out(32000)
      common/calib/wat(2)
      integer*4 in,out
      REAL*4 wat(2)

c       inicializar o loop
c     ================


          call acpwat(wat(1))


          return
          end
```

APENDICE D

```
!================================================
!     PROGRAMA  TC.UPF
!================================================
!
system=CBPF
host source=TC.for
node source=node_tc.for
node main subroutine = wait
node blocks = 1:input, 2:output, 3:calib
class 1: node = 1 68020


c*****************************************************************
C     PROGRAMA TC.FOR
c*****************************************************************
c Este programa avalia o tempo de comunicação entre o uVAXII
c              um processador do Sistema ACP
*****************************************************************

      include 'acp$area:user_common.inc'

c     local variables:
c     ================

      integer*4 in,out,status
      integer*4 num_in,num_out,event,node
      common/input/in(100000)
      common/output/out(100000)
      logical nodeready,nodedone,lastevent,alldone
      logical read_done,response_node
      real time_host(10000),start_time
      real wat,soma,media

c       Inicializar o processo RMIMOM, o
C       carregamento do LUNI e o proograma nos
c       processadores do Sistema ACP.
c     ========================================

      call acp_init

c     initializar as variáveis
c     ================================

       event=100
       num_out = 1.
       num_in = 1.

c      Inicializar o "loop" de variação do
c      tamanho do bloco.
c      ========================================

      DO 3 M=1,15
```

```
c        Inicializar o "loop" de valor médio
c       ======================================

        soma=0.

       do 2 k=1,3

c     Inicializar o "loop" de variação
c       do tamanho do evento
c      ============================================

             nodeready = .true.
             nodedone = .false.
             lastevent = .false.
             L = 0.

c        Inicializar o "loop" de envio de eventos
C       ==========================================


         start_time = secnds(0.0)


10        continue

          IF(L.Gt.EVENT)LASTEVENT=.TRUE.
*****************************************************
*                                                   *
*         SEND                                      *
*                                                   *
*****************************************************
         if(.not.lastevent) then
          if(nodeready) then
           l= l+1
          end if
          call acp_sendevent(in,num_in,nodeready)
          else
          if(acp_alldone(response_node))go to 20
         end if

*****************************************************
*                                                   *
*         GET                                       *
*                                                   *
*****************************************************
         call acp_getevent(out,num_out,nodedone)

          go to 10
*****************************************************
```

```
20        continue

C         detrminação do tempo total de processamento
C       ==================================================

        time_host(k)   = secnds (start_time)


         soma=time_host(k)+soma

2         continue

C         Avaliacao do valor médio
c       ===================================

         media=soma/3.

         WRITE(6,55)media
55       FORMAT(/,10x,' Processing time (sec) =',F8.2,/)

          NUM_OUT=NUM_OUT+5000
3         CONTINUE

C         Executar a saída do Sistema ACP
c       ============================================

        call acp_exit

c       End of the program:
c       ==================

         stop
         end



c ========================================
c          PROGRAMA NODE_TC.FOR
c ========================================

        subroutine wait

c       local declaration:
c       ==================

        common/INPUT/in(100000)



        common/OUTPUT/out(100000)
        integer*4 in,out



            return
            end
```

APENDICE E

```
!*************************************************
!          PROGRAMA M4.UPF
!*************************************************
system=CBPF
host source=m4.for
host library=dua0:[lib.hplot.hpldigs]hpldigs.olb
host library=dua0:[lib.digs]gralib.olb
host library=dua0:[lib.hbook]packlib.olb
host library=dua0:[lib.cernlib]kernlib.olb
node source=nodem4.for
node main subroutine = wait
node subroutine = wait
node blocks = 1:input, 2:output, 3:calib,4:va
class 1: nodes = 4  68020
```

```
c*************************************************************
c      PROGRAMA M4.FOR
c*************************************************************
c     Programa para determinar o tempo total de processamento
c     de uma tarefa composta de 16.000 sub-tarefas, onde cada
c     sub-tarefa porcessa por um tempo (P3) e acessa a memori
c     remota de um processador vizinho para ler um bloco
c     de tamanho (P1)
c
*************************************************************

      include 'acp$area:user_common.inc'

c     local variables:
c     ================
         common//hmem(10000)
         common/input/in(10)
         common/output/out(10)
         common/calib/addr(100)
         common/va/vec(100000)
         logical nodeready,nodedone,lastevent,alldone
         logical read_done,response_node,send_done
         real*4 time_host(10000),start_time,x1,x2,y1,y2
         real*4 soma,media(10000,17),tr(10000,17),np,tc,t
         integer*4 node_address1,addr
         integer*4 in,out,status,vec,cont
         integer*4 num_in,num_out,event,nodes

c        Inicializar o processo RMIMOM, carregar
c        o LUNI e o porgrama NODEMASTER4.FOR
c        nos processadores do Sistema ACP.
```

```
c       ==========================================
          call acp_init
          call hplint(0)

c         determinação do endereço  de VME
c         do bloco a ser lido
c     ==========================================

          node_address1=acp_node_vme_address(1)
          node_address2=acp_node_vme_address(2)
          node_address3=acp_node_vme_address(3)
          addr(1)=acp_node_block_offset(1,4)
          addr(2)=node_address1+addr(1)
          addr(3)=node_address2+addr(1)
          addr(6)=node_address3+addr(1)
c
c         inicializar as variáveis
c       ================================

          num_in = 1
          addr(5)=2
          addr(9)=1
          addr(8)=1000
          addr(7)=10

          call hbook2(CONT+1,'master$',6,1.,6.,1,6.,0.)

          call hbook2(CONT+2,'tr table $',6,1.,6.,6,1.,6.,0.)

          call htable(CONT+3,'master table $',6,1.,6.,6,1.,6.,

          call htable(CONT+4,'tr table $',6,1.,6.,6,1.,6.,0.)

c         inicializar o "loop" de variacao do tempo
c         de processamento de uma tarefa (P3)
c       ============================================

          DO 5 JJ=1,5


               np=float(addr(7))*float(addr(8))*0.005

c         inicializar o "loop" de variação
c         do número de palavras lidas (P1)
c       ========================================

           addr(4)=1
          DO 3 M=1,5

              mm=addr(4)*addr(8)*5
*****************************************************
*                                                   *
```

```
*        BROADCAST                                                    *
*                                                                     *
**********************************************************************
c        carregar os valores dos endereços de VME dos
c        blocos

         call acp_broadcast (3,addr,9,ACP_I_4)

c        Inicializar o valor médio
c       ================================

         soma=0.

         DO 2 K=1,3

c       inicializar o "loop" de envio de eventos
C       ========================================


             nodeready = .true.
             nodedone = .false.
             lastevent = .false.
             L = 0.
             n=0

c        Inicializar a contagem do tempo total de
c        processamento
c        ========================================

         start_time = secnds(0.0)


10        continue

          IF(L.eq.nodes)LASTEVENT=.TRUE.
**********************************************************************
*                                                                    *
*         SEND                                                       *
*                                                                    *
**********************************************************************
         if(.not.lastevent) then
          if(nodeready) then
           l= l+1
           n=n+1
            if(n.gt.nodes)n=1
c             write(6,66)n
66           format(/,2x,' n =',i8)
          end if
          call acp_runnode(1,in,0,n,1,'wait',nodeready)
              addr(5)=addr(5)+1
```

```
          else
          if(acp_alldone(response_node))go to 20
         end if
*****************************************************
*                                                   *
*         GET                                       *
*                                                   *
*****************************************************

         call acp_getevent(out,1,nodedone)


          go to 10


*****************************************************
*                                                   *
*         end of event loop                         *
*                                                   *
*****************************************************

20        continue

c        determinar o tempo total de processamento
C        ========================================


       time_host(1)   = secnds (start_time)


        soma=time_host(1)+soma


2        continue

c        determinar o valor médio
c       ==================================


        media(m,jj)=soma/3.

c     addr(8)= numero de tarefas por processador
c     0.4*0.0001 =tempo necessário para uma transmissão
c     add(4)*5= número médio de transmissões por tarefa
c     tc = tempo de comunicação por processador

        tc=float(addr(4))*5*0.4*0.00001*float(addr(8))

c     0.005 =tempo médio de processamento por tarefa
c     addr(7)= cte que multiplica a função randomica
```

```
c     0.65 = valor de correção da função secnds
c     np= tempo de processamento por processador

         np=float(addr(7))*0.005*float(addr(8))*0.65

c        determinar o valor de degradação
c        ================================
          tr(m,jj)=media(m,jj)-tc-np

c       armazenar o valor de deradação (tr(m,jj))
c       e o tempo total de processamento medio(m,jj))
c       =============================================

        call hfill(cont+1,float(m),float(jj),media(m,jj))

        call hfill(cont+2,float(m),float(jj),tr(m,jj))

        call hfill(cont+3,float(m),float(jj),media(m,jj))

        call hfill(cont+4,float(m),float(jj),tr(m,jj))

             addr(4)=addr(4)+5000

c         somar 5000 no valor de (P1) e voltar ao "loop"
c         ==============================================

3          continue


c        somar 20 ao valor da constante que multiplica
c        (P3) e voltar ao "loop"
c        ==============================================

          addr(7)=addr(7)+20

5          continue

c         plotar
c         =========

        call hplego(cont+1,60.,60.)
        call hplot(cont+1)
        call hplego(cont+2,60.,60.)
        call hplot(cont+2)
        call hplend

c       imprimir os histogramas no arquivo for015.dat
c       =============================================
```

```
          call houtpu(16)
          call hprint(cont+3)
          call hprint(cont+4)
            cont=cont+2
4           continue
c         Executar os procedimentos de saida do
c         Sistema ACP.
c       ========================================
        call acp_exit
         stop
         end

c ========================================
c         PROGRAMA   NODEM4.FOR
c ========================================

          subroutine wait

c         declaração local
c          ==================

          common/input/in(10)
          common/output/out(10)
          common/calib/addr(100)
          common/va/vec(10000)
          integer*4 out,va,vec,in,v,n,nodes,ph,soma,i
          integer*4 vector_address,dst,addr,a,d,flag
          real*4 secnds,iseed,time,d

          call acpndi('PHYSICAL NUMBER',ph)

c         inicializar a variavel randomica
c       ============================

          iseed = 1001*ph*addr(5)

          seed = 1001*ph*addr(9)

c         (n) é
c         o  número de acessos a memória remota(P1)
c         e uma variável randomica cujo limite
c         e fornecido pelo hospedeiro
c         ===================================

          n=acpran(seed)*addr(4)*10

           time=0.

c         inicializar o loop para executar
c         1.000 sub-tarefas
c         ============================
```

```
          DO 1 J=1,addr(8)

c         (secnds) é
c         o tempo de porcessamento e o resultado
c         de uma variavel randomica cujo limite(P3)
c         foi fornecido pelo hospedeiro
c         ======================================

          secnds=acpran(iseed)*float(addr(7))*0.01

c
c         executar a rotina de retardo
c         ====================================

          call acpwat(secnds)

c         caso seja o processador mestre, escreva
c         na memoria local
c       ====================================

          if (ph.eq.3) then
                        call acpsup
                      do 40 i=1,n
                        flag=long(addr(3))
40                  continue
                        call acpusr

c         caso seja processador escravo escreva
c         na memória remota
c       ============================================

              else


                    call acpsup
                  do 10 i=1,n
                    vec(1)=long(addr(2))
10                continue
                   call acpusr

          end if

1         CONTINUE

          return
          end
```

APENDICE F

```
!*************************************************
!          PROGRAMA M.UPF
!*************************************************
system=CBPF
host source=m.for
host library=dua0:[lib.hplot.hpldigs]hpldigs.olb
host library=dua0:[lib.digs]gralib.olb
host library=dua0:[lib.hbook]packlib.olb
host library=dua0:[lib.cernlib]kernlib.olb
node source=nodem4.for
node main subroutine = wait
node subroutine = wait
node blocks = 1:input, 2:output, 3:calib,4:va
class 1: nodes = 16  68020
```

```
c*************************************************************
c     PROGRAMA M.FOR
c*************************************************************
c  Este  programa determina a aceleração de um Sistema ACP d
c  processadores para o processamento de uma tarefa composta
c  16.000 sub-tarefas onde em cada sub-tarefa um processador
c  processa por (P3) segundos e acessa a mamória
c  remota de um processador vizinho (P1) vezes.
c
*************************************************************

      include 'acp$area:user_common.inc'

c       variáveis locais
c        ================
         common//hmem(10000)
         common/input/in(10)
         common/output/out(10)
         common/calib/a(10),addr(10)
         common/va/vec(100000)
         logical nodeready,nodedone,lastevent,alldone
         logical read_done,response_node,send_done
         real*4 time_host(10000),start_time,x1,x2,y1,y2,a
         real*4 soma,media(10000,17),speed(10000,17),np
         integer*4 node_address1,addr
         integer*4 in,out,status,vec,cont
         integer*4 num_in,num_out,event,nodes

c       Inicializar o processo RMIMOM, carregar
c        o LUNI nos processadores do Sistema ACP
c        e o programa nodemaster.for
c       ============================================
         call acp_init
```

```
          call hplint(0)
c         determinar o endereço de VME dos blocos
c         comuns dos processadores
c    ========================================

          node_address1=acp_node_vme_address(1)
          node_address2=acp_node_vme_address(2)
          node_address3=acp_node_vme_address(3)
          addr(1)=acp_node_block_offset(1,4)
          addr(2)=node_address1+addr(1)
          addr(3)=node_address2+addr(1)
          addr(6)=node_address3+addr(1)


c         inicializar as variáveis
c       ================================

          num_in = 1
          addr(5)=2


c         inicializar o "loop" de variação
c         do número de processadores (P2)
c         ========================

              CONT=0
              nodes=0

              do 4 ii=1,16


                NODES=ii

c.      abrir os arquivos para os graficos
c      ==================================

              addr(8)=1600/nodes
              addr(7)=1

          call hbook2(CONT+1,'master$',3.,1.,3.,16,1.,16.,0.)

          call hbook2(CONT+2,'speed up $',3,1.,3.,16,1.,16.,0.

          call htable(CONT+3,'master table $',3,1.,3.,16,1.,16

          call htable(CONT+4,'speed up/tp=1s $',3,1.,3.,16,1.,


c         inicializar o loop de processamento do numero de lei
c       ======================================================
```

```
          addr(4)=1

          DO 3 M=1,3

*****************************************************
*                                                   *
*         BROADCAST section                         *
*                                                   *
*****************************************************
c         carregar nos processadores o endereco de
c         VME do bloco comum

           call acp_broadcast (3,addr,8,ACP_I_4)
          a(1)=0.0001
          a(2)=0.0002
          a(3)=0.0003

           call acp_broadcast (3,a,8,ACP_r_4)

c         inicialzar o valor médio
c        ================================

          soma=0.

          do 2 k=1,3


c         inicializar o loop do evento
C        ====================


              nodeready = .true.
              nodedone = .false.
              lastevent = .false.
              L = 0.
              n=0
c         inicializar a contagem de tempo
c         ================================

          start_time = secnds(0.0)
10         continue
           IF(L.eq.nodes)LASTEVENT=.TRUE.
*****************************************************
*                                                   *
*          SEND section                             *
*                                                   *
*****************************************************
```

```
          if(.not.lastevent) then
           if(nodeready) then
            l= l+1
            n=n+1
             if(n.gt.nodes)n=1
           end if
           call acp_runnode(1,in,0,n,1,'wait',nodeready)
               addr(5)=addr(5)+1
           else
           if(acp_alldone(response_node))go to 20
          end if

*****************************************************
*                                                   *
*         GET section                               *
*                                                   *
*****************************************************
         call acp_getevent(out,1,nodedone)


          go to 10

*****************************************************
*                                                   *
*         end of event loop                         *
*                                                   *
*****************************************************

20        continue


c         determinação do tempo total de processamento
C         ============================================


       time_host(1)   = secnds (start_time)


        soma=time_host(1)+soma


2        continue

c       determinar o valor medio
c       ========================


         MEDIA(M,JJ)=SOMA/3.
```

```
c        determinar o valor da aceleração
c        ========================================

       speed(m,jj)=media(m,1)/media(m,jj)


c        armazenar o resultado
c        =====================

       call hfill(cont+1,float(m),float(jj),media(m,jj))

       call hfill(cont+2,float(m),float(jj),speed(m,jj))

       call hfill(cont+3,float(m),float(jj),media(m,jj))

       call hfill(cont+4,float(m),float(jj),speed(m,jj))


c        incrementar o valor de (P1) e voltar ao loop
c        ============================================

            addr(4)=addr(4)+10


3         continue


5         continue

c         plotar
c         =========
         call hplego(cont+1,30.,30.)
         call hplot(cont+1)
         call hplego(cont+2,60.,60.)
         call hplot(cont+2)
         call hplend
c        imprimir os histogramas no arquivo for015.dat
c        =============================================
           call houtpu(16)
         call hprint(cont+3)
         call hprint(cont+4)
           cont=cont+2
4          continue
c        executar os procedementos de saida do
c        sistema ACP
c      ========================================
       call acp_exit
        stop
        end
```

APENDICE G

```
|**********************************************
|          PROGRAMA T.UPF
|**********************************************
system=CBPF
host source=T.for
node source=node_T.for
node main subroutine = wait
node subroutine = wait
node blocks = 1:input, 2:output, 3:calib,4:va
class 1: nodes = 1  68020
```

```
c**************************************************************
c     PROGRAMA T.FOR
c**************************************************************
c  programa para deteminar o tempo de um processador realiza
c  uma leitura de uma palavra em
c  uma memoria remota via barramento VME
c
**************************************************************

      include 'acp$area:user_common.inc'

c     local variables:
c     ================
         common//hmem(10000)
         common/input/in(10)
         common/output/out(10)
         common/calib/a(10),addr(10)
         common/va/vec(100000)
         logical nodeready,nodedone,lastevent,alldone
         logical read_done,response_node,send_done
         real*4 time_host(10000),start_time,x1,x2,y1,y2,a
         real*4 soma,media(10000,17),speed(10000,17),np
         integer*4 node_address1,addr
         integer*4 in,out,status,vec,cont
         integer*4 num_in,num_out,event,nodes


c      Inicialização  do processo RMIMOM,
c      carregamento do processo nodemaster.for
c      e do LUNI nos processadores do Sistema ACP
c      ==========================================

        call acp_init
        call hplint(0)
```

```
c       detrminar o endereço de VME do
c       bloco comum dos processadores
c    ========================================

         node_address1=acp_node_vme_address(1)
         node_address2=acp_node_vme_address(2)
         node_address3=acp_node_vme_address(3)
         addr(1)=acp_node_block_offset(1,4)
         addr(2)=node_address1+addr(1)
         addr(3)=node_address2+addr(1)
         addr(6)=node_address3+addr(1)


c       inicializar as variáveis
c     ================================

        num_in = 1
        addr(5)=2
        cont=0
        NODES=1



c        executar apenas uma sub-tarefa
c       ================================
        ADDR(8)=1

            addr(7)=1

c       ler 100.000 palavras na memória remota
c       ======================================

         addr(4)=100000


*****************************************************************
*                                                               *
*       BROADCAST section                                       *
*                                                               *
*****************************************************************
c       carregar nos processadores os endereços
c       de VME dos blocos comuns

         call acp_broadcast (3,addr,8,ACP_I_4)
        a(1)=0.0001
        a(2)=0.0002
        a(3)=0.0003

         call acp_broadcast (3,a,8,ACP_r_4)
```

```
c        inicializar o valor médio
c       ==============================

         soma=0.

         DO 2 K=1,100


c        inicializar o loop do evento
C       ====================


             nodeready = .true.
             nodedone = .false.
             lastevent = .false.
             L = 0.
             n=0

c        inicializar a contagem do tempo total de
c        processamento
c        ========================================

         start_time = secnds(0.0)


10        continue

          IF(L.eq.nodes)LASTEVENT=.TRUE.
*****************************************************
*                                                   *
*         SEND section                              *
*                                                   *
*****************************************************
         if(.not.lastevent) then
          if(nodeready) then
           l= l+1
           n=n+1
          call acp_runnode(1,in,0,n,1,'wait',nodeready)
              addr(5)=addr(5)+1
          else
          if(acp_alldone(response_node))go to 20
         end if


*****************************************************
*                                                   *
*         GET section                               *
*                                                   *
*****************************************************
         call acp_getevent(out,1,nodedone)
```

```
          go to 10

*****************************************************
*                                                   *
*         end of event loop                         *
*                                                   *
*****************************************************

20        continue

c        determinar o tempo total de processamento
C     ==========================================

       time_host(1)   = secnds (start_time)

        soma=time_host(1)+soma

2        continue

c        determinar o valor medio
c       ==================================

         MEDIA(M,JJ)=SOMA/100.



c        executar os procedementos de saida do
c        Sistema ACP
c      ========================================

       call acp_exit

        stop
        end
```

```
c ========================================
c      PROGRAMA    NODE_T.FOR
c ========================================

      subroutine wait

c     declaração local
c      ==================

      common/input/in(10)
      common/output/out(10)
      common/calib/addr(100)
      common/va/vec(10000)
      integer*4 out,va,vec,in,v,n,nodes,ph,soma,i
      integer*4 vector_address,dst,addr,a,d,flag
      real*4 secnds,iseed,time,d



c     o  número de acessos a memória remota(P1)
c     é fornecido pelo hospedeiro
c     ====================================


      time=0.

c     inicializar o loop para executar
c     1.000 sub-tarefas
c     ==============================


      DO 1 J=1,addr(8)



c     escreva
c     na memória remota
c    ============================================



                call acpsup
              do 10 i=1,ADDR(4)
                vec(1)=long(addr(2))
10            continue
               call acpusr

      end if

1     CONTINUE

      return
      end
```