# Uso da Lógica no desenvolvimento de arquiteturas de hardware e software

Aracele Garcia de Oliveira
aracele.garcia@gmail.com
Universidade Federal de Santa Catarina
Florianópolis – SC – Brasil
Agosto de 2008

## 1 Resumo

Este trabalho baseia-se em uma pesquisa bibliográfica que serviu como fundamentação teórica de um estudo sobre a contribuição da programação em lógica para a construção de arquiteturas de hardware e software, denominadas Máquinas Lógicas ou Máquinas Prolog[1]. A grande vantagem de utilização da lógica na representação e transformação do conhecimento é a sua forte formalização matemática, o que também pode ser melhorado através do uso de técnicas de processamento paralelo, tanto a nível de hardware quanto de software, o que é uma boa alternativa para obter performance. Um projeto financiado pelo governo japonês entre 1982 e 1992, que iniciou este pensamento e deu origem aos chamados "Computadores de Quinta Geração", não atingiu os objetivos esperados mas serviu para impulsionar o uso do Prolog e outros conceitos associados à lógica. As seções a seguir estão organizadas na ordem cronológica nas quais os assuntos e tecnologias ocorreram ou foram desenvolvidos.

## 2 Paradigma Lógico

A lógica é a ciência e Arte do raciocínio [Buchsbaum 2006] e a ciência do pensamento correto [Palazzo 1997]. É necessário esta premissa inicial para entender como a lógica pode ser empregada na programação de computadores e como resumir várias coisas ao seu conceito formal.

O objetivo principal do Paradigma Lógico é trazer para a programação o estilo da lógica matemática.

### 2.1 Raízes

As raízes da lógica podem ser atribuídas a Aristóteles, que sistematizou e codificou o assunto de tal modo que não foi significativamente ultrapassado por mais de dois milênios [Nolt and Rohatyn 1991]. Estudos de Boole(1815-1864) e de De Morgan(1806-1871) remontam o uso da lógica na representação do raciocínio, mas estavam mais próximos da matemática do que da lógica propriamente dita. O relacionamento entre lógica e matemática foi também investigado por Alfred North Whitehead e Georg Cantor, que também produziu a teoria dos números transfinitos e que em "Principia Mathematica" (1910) demonstrou ser a lógica um instrumento adequado para a representação formal de grande parte da matemática. A lógica moderna provém, em grande parte, do trabalho do filósofo alemão Gottlob Fregue no século XIX, com o que hoje denomina-se cálculo dos predicados.

Embora a lógica se desenvolvesse como um ramo da filosofia, o seu progresso explosivo desde Fregue, produziu aplicações em linguística, matemática e ciência da computação. Diversos estudos posteriores foram de grande importância para sua evolução, com destaque para as investigações de Herbrand, Gödel, Tarski, Prawitz, Robinson, Green, Turing e Church.

No início da segunda guerra mundial, por volta de 1939, toda a fundamentação teórica básica da lógica computacional estava pronta. Mas faltava o meio prático para se realizar os procedimentos de prova. E a partir da metade dos anos 50, o desenvolvimento dos computadores conseguiu oferecer aos pesquisadores o potencial computacional necessário para a realização de experiências mais significativas com o cálculo de predicados [Palazzo 1997].

### 2.2 Programação em Lógica

Esta expressão, originária de "Logic programming" em inglês, é devida a Robert Kowalski (1974) e sugere o uso da lógica como linguagem de programação em computadores. Ele identificou que a prova de teoremas poderia ser computada, permitindo assim uma interpretação procedimental da lógica e estabelecendo as condições que nos permitissem entendê-la como linguagem de Programação de uso geral.

Neste paradigma, não se descreve o algoritmo para chegar ao resultado. Descreve-se a informação base (fatos) e as regras de inferência(regras) sobre certo domínio, para se poder deduzir

[1] O Termo *Parallel Prolog Machine também foi utilizado para designar o paralelismo nestas máquinas.*

o resultado através de perguntas que são feitas à esta base. Um procedimento de inferência é a estratégia para a escolha das derivações possíveis em cada estado.

O principal interesse nesse modelo de programação estava na sua capacidade de tratar problemas de alta complexidade computacional e na simplicidade da sua estrutura e principalmente ao rigoroso suporte matemático ao qual a lógica está associada.

## 2.3 Programação em Lógica X Arquitetura de Von Newmann

Programação em lógica é uma teoria que representa um modelo abstrato de computação sem relação direta com o modelo de máquina Von Neumann [Valente 1993].

O paradigma da programação em lógica abandona totalmente o conceito de Von Neumann por não fornecer instruções explícitas para operações e seu seqüenciamento, adotando o modelo dos axiomas lógicos, isto é, proposições verdadeiras que representam o conhecimento que se possui a respeito do problema e regras que permitem deduzir novos conhecimentos. A programação lógica diferencia-se da programação procedimental, fundamentalmente, por requerer a descrição da estrutura lógica do problema ao invés da maneira pela qual o computador deve solucioná-lo [Price and Lisbôa 2003].

## 2.4 Prolog

O primeiro interpretador experimental foi desenvolvido por um grupo de pesquisadores na Universidade de Aix-Marseille(1972) com o nome de Prolog (acrônimo para Programming in Logic). Implementações mais práticas foram desenvolvidas por David H.D. Warren e outros, na Universidade de Edimburgo (U.K), que definiram o chamado "Prolog de Edimburgo", utilizado como referência na maioria das implementações atuais da linguagem Prolog [Palazzo 1997]. Warren também especificou a chamada "Warren Abstract Machine", um modelo formal utilizado na pesquisa de arquiteturas computacionais baseadas ou orientadas à programação em Lógica.

Foi uma tentativa de fazer uma linguagem de programação que permitisse a expressão de lógica de um programa ao invés de cuidadosamente especificar cada instrução que o computador deveria executar.

Algumas das principais características do Prolog são descritas abaixo:

- Orientada ao processamento Simbólico
- Representa uma implementação da Lógica como Linguagem de Programação
- Permite a obtenção de respostas alternativas
- Representa programas e dados através do mesmo formalismo

A característica que dá ao Prolog vantagens sobre linguagens procedurais[2] é a separação entre componentes lógicos e de controle.

As características herdadas da Lógica o levou a tornar-se uma das duas principais linguagens na área de Inteligencia Artificial, juntamente com o LISP(LISt Processor).

Segue abaixo um quadro comparativo entre características de Programas Convencionais e de Programas em Lógica:

| PROGRAMAS CONVENCIONAIS | PROGRAMAS CONVENCIONAIS |
|---|---|
| Processamento Numérico | Processamento Simbólico |
| Soluções Algorítmicas | Soluções Heurísticas |
| Estruturas de Controle e Conhecimento Integradas | Estruturas de Controle e Conhecimento Separadas |
| Difícil Modificação | Fácil Modificação |
| Somente Respostas Totalmente Corretas | Incluem Respostas Parcialmente Corretas |
| Somente a Melhor Solução Possível | Incluem Todas as Soluções Possíveis |

**Tabela 1- Programas Convencionais X Programas em Lógica**

### 2.4.1 Conceitos essenciais da Linguagem Prolog

Em Prolog, o que se define são objetos e relações entre objetos. Para se referir a objetos usamos "termos".

- Termos: constante, variável ou estrutura;
- Átomos: Valores simbólicos;
- Variável: Inicia-se com uma letra maiúscula;
- Estrutura: meio de especificar fatos em Prolog.

[2] Linguagem de programação na qual o elemento básico é a procedure (uma seqüência de instruções - rotina, sub-rotina ou função - associadas a um nome próprio). As linguagens de alto nível (C, Pascal, Basic, Fortran, Cobol, Ada) são todas linguagens procedurais.

Um programa Prolog consiste basicamente de:

- declaração de fatos sobre objetos e suas relações;
- definições de regras sobre os objetos e suas relações;
- questões que são feitas sobre os objetos e suas relações.

Fatos são estruturas usadas para construir o banco de dados de informações presumidas. Eles são descritos na forma de Horn[3] sem "cabeça". Exemplo:

- homem(João).

Regras são cláusulas Horn com "cabeça" usadas para descrever condições similares a if-them. Exemplo:

- pais(Maria, João):- mãe(Maria, João).

Metas são cláusulas de Horn sem "cabeça" que são avaliadas pelo sistema de inferência do Prolog. Cada meta é decomposta em submetas. O processo de inferência tenta encontrar o encadeamento de regras e fatos que ligam a meta a um ou mais fatos do Banco de Dados

- ? – pais (X, João).
- X=Maria
- Yes

### 2.4.2 A exploração do Paralelismo no Prolog

A busca de regras é realizada através de uma árvore de pesquisa, e é feita, primeiro, em profundidade, implementada por uma máquina de pilha, onde se guardam os objetivos da resolvente.

A execução de Regras é assim descrita: o sistema executa todos os sub-objetivos ou submetas sequencialmente da esquerda para a direita e considera cláusulas candidatas sequencialmente na ordem textual definida no programa, para uma definição precisa da semântica sequencial do Prolog [Beer 1987].

Uma vez que a semântica declarativa do Prolog não impõe qualquer ordem de execução para as metas ou ordem de pesquisa das clausulas, é imediatamente óbvio que programas lógicos podem fazer uso de um inerente paralelismo.

O Processamento paralelo é uma alternativa muito importante para se obter bons resultados no desempenho de programas. O Paralelismo pode ser explorado implícita ou explicitamente. Os mais conhecidos sistemas paralelos são baseados na programação imperativa e o paralelismo é explorado explicitamente, tornando a programação uma tarefa difícil. Contrastando com o programação imperativa, está a programação declarativa de alto-nível, um modelo onde o programador só se preocupa com o que deve resolver, e não com a forma de resolver um problema. Devido a esta característica declarativa, a lógica de programação oferece algumas oportunidades para explorar o paralelismo implícito. Este claramente reduz o custo do desenvolvimento de software, como o usuário escreve só uma aplicação que irá executar em apenas um processador, assim como em vários[Dutra et al. 1999].

Parcialmente iniciado pelo FGCS, a ser descrito na seção 4, esforços foram consideráveis para definir uma investigação do paralelismo no Prolog e seus dialetos e máquinas paralelas adequadas para execução paralela de grandes projetos Prolog.

As principais formas de paralelismo, a nível de software, que podem ser identificadas são:

- Paralelismo-OR: percorre em paralelo ou simultaneamente, todos os ramos da árvore de pesquisa capazes de resolver um objetivo ou meta corrente.
- Paralelismo-AND: é a execução paralela de todos os sub-objetivos ou submetas no âmbito do corpo da cláusula.
- Paralelismo-Unificação: é a unificação concomitante de todos os argumentos de um dado objetivo e a cláusula original

Implementações iniciais neste sentido foram: Concurrent Prolog, GHC (Guarded Horn Clauses), Parlog e Andorra.

# 3 A Máquina Abstrata de Warren (WAM)

Como foi dito na seção 2.4, David Warren especificou a chamada "Warren Abstract Machine", que consistia de uma arquitetura de memória e conjunto de instruções adaptados ao Prolog. A WAM foi apresentada por ele em uma memorável palestra em U.C. Berkeley em Outubro de 1983. Seria uma máquina abstrata prolog, conhecida como Máquina Abstrada de Warren, e este modelo se tornaria um padrão para a implementação de compiladores e interpretadores prolog [Hassan 1990].

A execução de um programa Prolog é feito em duas etapas, em uma implementação baseada na WAM: [Hanus 1988]

- Compilação do Programa em uma sequência de instruções WAM

[3] Um programa lógico é uma coleção de cláusulas de Horn do tipo $A_0$ if $A_1$ and ... and $A_n$

- Execução do programa WAM por um interpretador ou compilação para uma outra linguagem, por exemplo, linguagem de máquina.

Os principais objetivos do autor quando criou a WAM foram a forma fácil de mapear código Prolog para código WAM e a arquitetura ser baixo-nível suficiente para poder ser implementada/emulada sobre outras arquiteturas.

Alguns nomes de instruções eram put, get, unify, variable, value, execute, proceed, try, retry, and trust.

WAM era simples por fora (um pequeno e limpo conjunto de instruções) e complexa por dentro (suas instruções faziam coisas complexas)

- Objetos de Dados: Em WAM um objeto Prolog é representado por uma palavra contendo uma tag e um valor. A Tag distingue o tipo do objeto. Os principais tipos são variáveis não-ligadas(unbound), variáveis referenciadas, constantes, estruturas e listas.
- Áreas de Memória: A WAM contém cinco áreas de memórias divididas entre o código e o espaço de dados. O espaço de código contêm instruções e outras informações representando o programa. As quatro áreas de espaço de dados são acessadas como pilhas.

O processo de inferência em Prolog inicia com uma consulta, essa consulta é particionada em suas submetas e então o fluxo de execução tenta encontrar o encadeamento de regras e/ou fatos que ligam a meta aos fatos na base de conhecimentos. Essa forma de resolução é capaz de resolver problemas em domínios finitos. Na figura 1 é possível ver a organização de memória da WAM que é composta por áreas de dados lógicos:

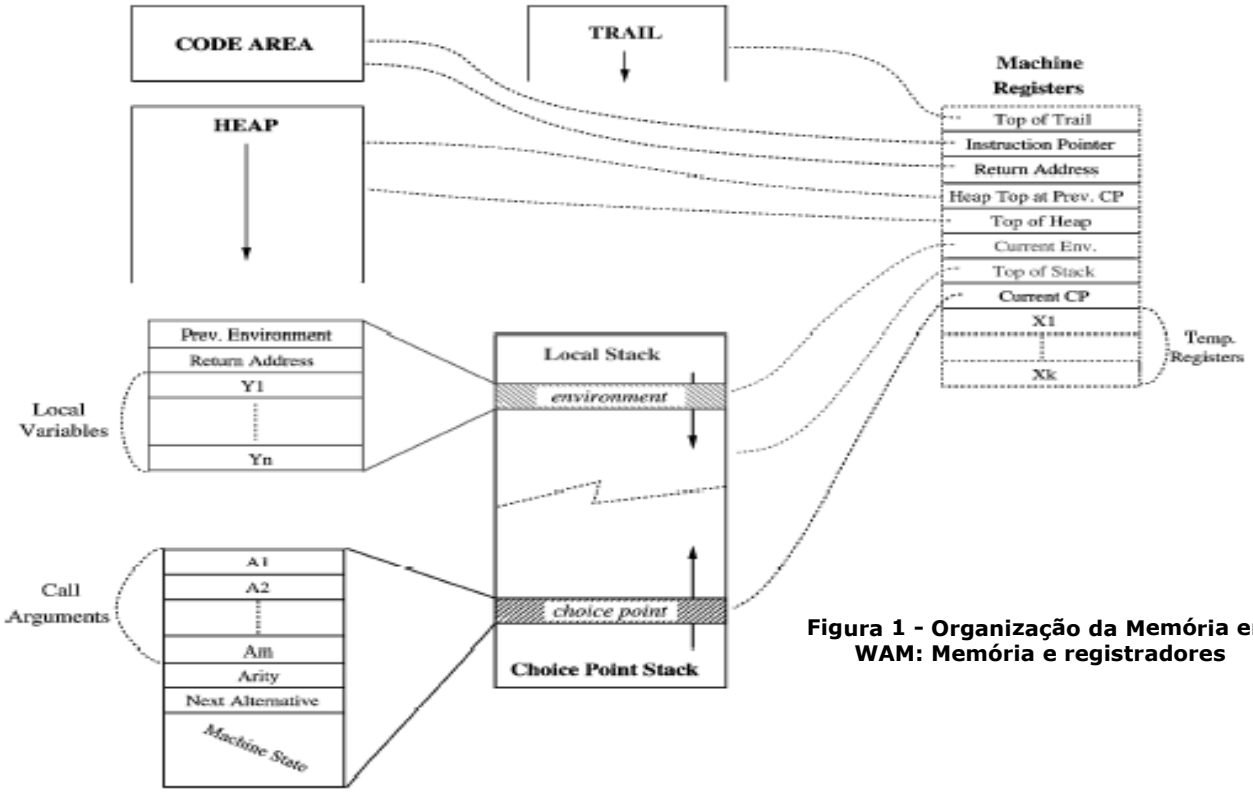


**Figura 1 - Organização da Memória em WAM: Memória e registradores**

- heap onde são alocadas as estruturas complexas em Prolog (listas e termos compostos)
- Pilha local que armazena o environment (semelhante a um registro de ativação em uma máquina von-Neumann) e as variáveis locais, a cláusula objeto de resolução.
- Pilha de pontos de escolha (ou choicepoint-stack) guarda as informações para realizar o backtracking.
- Pilha de rastro (trail-stack) guarda as informações necessárias para desfazer as alterações nos dados em caso de backtracking.

Além disto essa máquina tem alguns registradores globais definidos para o controle das pilhas e da execução.

A execução é realizada da seguinte forma: Argumentos da chamada são preparados e carregados nos registradores temporários através da instrução input. Cláusulas que "casam" com a submeta são detectados e se existe mais de um, então um ponto de escolha é alocado (try). A primeira cláusula é iniciada depois de criar o "ambiente" (allocate), a execução tenta a "unificação de cabeça" usando as instruções get/unify. E então o corpo da cláusula é executado. Caso falhe a unificação, será necessário backtraking, que envolve retorno a um ponto de escolha superior e escolha de uma nova instanciação e execução da submeta.

O sistema completo é mostrado na figura a seguir, quando "pass1" é a implementação do compilador Prolog e o "pass2" é a traduz toda instrução Wam para uma sequencia de instruções de Máquina. Neste caso a especificação baseada no implementação WAM 68000 [Hanus 1988].


**Figura 2 - Esquema de funcionamento do Modelo WAM**

# 4 O projeto de computadores de Quinta Geração

## 4.1 O momento

Ao longo de várias gerações, desde a década de 80, o Japão era um seguidor dos EUA e dos países britânicos no que se refere aos termos de avanço da informática e construção de computadores. Foi então que o Ministério de Comércio Internacional e Indústria (MITI) decidiu a tentativa de quebrar esta situação de "siga o líder". Em meados de 1970 começou a estudar, em pequena escala, o futuro da computação. Eles pediram ao Centro de Processamento e Desenvolvimento de Informações do Japão (JIPDEC) para indicar rumos futuros para serem aprofundados, juntamente com a indústria e a universidade. Foi durante este período que o termo "Quinta geração de computadores" começou a ser utilizado.

Os campos primários da investigação eram:

- Tecnologias de Inferência de computador para processamento do Conhecimento
- Tecnologias de computador para processamento de bases de dados e conhecimento em larga escala.
- Estações de alta performance
- Super-computadores para cálculo científico

## 4.2 O projeto

Na evolução do computador, as mudanças entre uma geração e outra, aconteceram através do aperfeiçoamento e desenvolvimento dos elementos que o compõem. A primeira geração usava válvulas, a segunda usou transistores, a característica principal da terceira foram os circuitos intergrados, chamados de "chips", e a quarta geração possui chips de densidade de integração maior, os VLSI, ou Very Large Scale Integration.

Em 1979 o governo japonês iniciou estudos para o "Projeto da Quinta Geração" para desenvolver, hardware e software de elevado desempenho (caracterizando assim uma nova geração de computadores) e usando variantes do Prolog e o próprio Prolog como linguagem básica do projeto (assim como o Assembly para as máquinas convencionais) comandando diretamente o hardware. O motivo do uso do prolog é porque se acreditava ser uma verdadeira linguagem paralelizável. Esse projeto, iniciou em 1982 e foi oficialmente concluído em 1992.

O FGCS(Fifth Generation Computer Systems) abrangiu o maior projeto de arquitetura dentro da comunidade de programação em lógica. O objetivo maior do projeto, que era o de produzir e disponibilizar microcomputadores pessoais baseados nestas novas tecnologias desenvolvidas, não foi atingido. Mas os computadores, sistema operacional e os programas produzidos pelo projeto são interessantes e importantes para o nível acadêmico. Diversos protótipos do computador de quinta geração foram construídos e chamados genericamente de PIMs – Parallel Inference Machines e isso serviu para difundir e estimular o uso do Prolog.

A idéia principal desse projeto, que foi considerado um dos mais ambiciosos projetos da história da computação, era construir uma máquina paralela utilizando programação lógica e técnicas de Inteligência Artificial [Rosa 1995].

## 4.3 Linha de Tempo do Projeto

| | |
|---|---|
| **1982** | O FGCS inicia e recebe financiamento para 5 anos. |
| **1985** | é conhecido o primeiro Hardware denominado como Máquina Pessoal de Inferência Sequencial – PSI (Personal Sequential Inference Machine) e a primeira versão do Sistema Operacional para a mesma, o SIMPOS. Ele foi programado em Kernel Language 0 (KL0-Kernel Language Zero), uma variante do prolog com extensões orientadas a objeto. |

| | |
|---|---|
| **1987** | Um protótipo do verdadeiro hardware paralelo chamado Máquina de Inferência Paralela ou *Parallel Inference Machine (PIM*) é construído utilizando vários PSI´s conectados como uma rede. O projeto recebe financiamento para mais 5 anos. Uma nova versão da Linguagem do Kernel, o Kernel Language 1(KL1-Kernel Language 1), é criado, influenciado pela evolução do prolog. O sistema operacional escrito em KL1 recebe um novo nome: *Parallel Inference Machine Operating System* ou PIMOS. |
| **1991** | É produzida a primeira máquina de inferência Paralela-PIM que funciona realmente. |
| **1992** | O programa FCGS é cancelado ou finalizado.O código-fonte do PIMOS é tornado domínio público, mas podendo ser executado somente em Maquinas PIM. Um financiamento adicional é dado para produzir um emulador de UNIX chamado KL1 compilador C (KLIC). |

### 4.4 Máquina de Inferência Paralela (PIM)

Parte das atividades do projeto de quinta geração foi desenvolver protótipos de máquinas paralelas para servir como uma ferramenta de desenvolvimento dos sistemas dos computadores de quinta geração[Galante 2004]. Cinco máquinas de inferência paralela foram criadas: PIM/m, PIM/p, PIM/i, PIM/k, PIM/c.

Nestas novas máquinas o papel da linguagem assembly será desempenhado por um dialeto do Prolog orientado ao processamento paralelo. O KL1 foi um deles. Uma PIM é uma máquina MIMD[4] de propósito geral que executava o KL1 eficientemente.

Tais máquinas possuiam arquitetura massivamente paralela e tinham velocidade de processamento calculada em MLIPS (milhões de inferências lógicas por segundo). A medida MLIPS é análoga àquela usada para os processadores tradicionais (MIPS – Milhões de instruções por segundo)

As principais características foram:

- Computador paralelo
- Bases de Dados em lugar de sistemas de arquivo
- Linguagem de Programação lógica para acessar bases de dados
- Teria entre 100MLIPS(100 megalips) e 1GLIPS (1 gigalips ou 1 bilhão de inferências lógicas por segundo)

As realizações do projeto incluem uma hierarquia de linguagens de programação lógica, o desenvolvimento de programação lógica para sistemas operacionais, e o desenvolvimento de uma família de máquinas de programação em lógica. Tambem serviu para incentivar projetos similares nos Estados Unidos e a criação do interesse em programação lógica ao redor do mundo.

De um modo geral, entretanto, considera-se que o projeto demonstrou ser a tecnologia PIM bem sucedida em novas aplicações envolvendo paralelismo em diversas áreas, especialmente computação não-numérica e inteligência artificial.

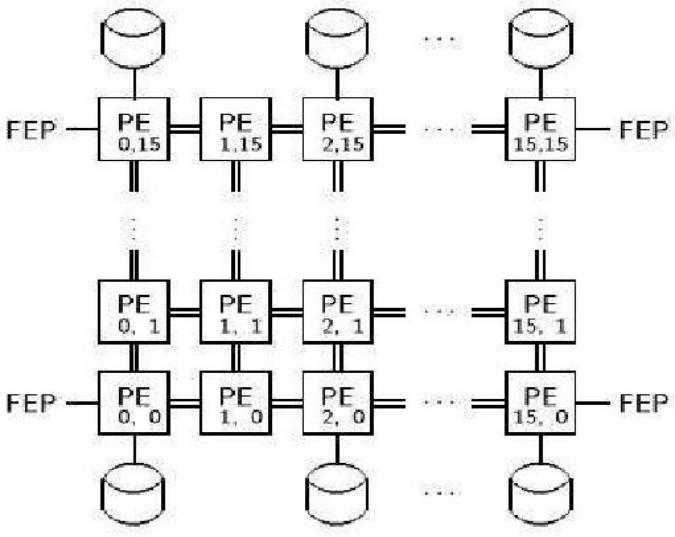


**Figura 3 – Uma modelo de Rede Paralela-PIM de 16x16 elementos de processador (PE) interligados**

Esta foi a primeira máquina de inferência prática. Seu sistema operacional PIMOS provia um ambiente altamente sofisticado para a programação paralela. A terceira máquina PIM/m possuia o

[4] MIMD (Multiple Instruction Multiple Data) são arquiteturas caracterizadas pela execução simultânea de múltiplos fluxos de instruções. Essa capacidade deve-se ao fato de que são construídas a partir de vários processadores operando de forma cooperativa ou concorrente, na execução de um ou vários aplicativos.

aumento de performance como grande diferencial, devido ao aumento de elementos de processador para 256.

A PIM consistia de 8 partes, cada uma com 32 elementos de processador e com quatro discos de 670MB de armazenamento secundário. O que consistia em um total aproximado de 20 GB. Cada disco é ligado com os elementos de processador e os FEP através de 32 portas SCSI. O FEP é uma estação de trabalho IA que provém ambiente de programação para usuários PIM/m. Em cada elemento de processador existem 3 chips VLSI, Unidade de Cache, Unidade de controle de rede e unidade de processamento:[Soler 2006]

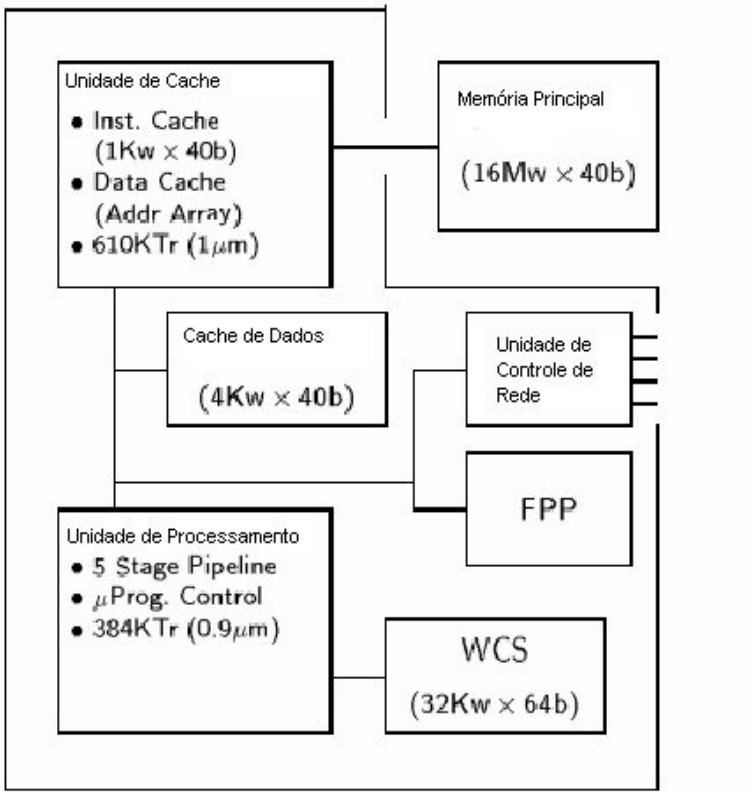


**Figura 4 - Elementos de Processador**

A unidade de cache contêm 1K-Word de instruções e 4K-word de cache de dados. O buffer de translação e endereçamento para instruções e dados também é instalado na unidade de cache. A troca de mensagens entre unidades de processador é feita através de uma tabela RAM chamada de Path Table(PT). A unidade de rede transmite um pacotes de dados independendo da localização física do Elemento de Processador adjacente. Este mecanismo utiliza o mecanismo de comunicação assíncrona usando FIFO buffers de saída.

Ainda segundo [Soler 2006], os principais registradores da unidade de processamento são: PC (program counter), AP (Alternate clause pointer), HP (Heap pointer), An/Xn são implementados como registrador de arquivo. Cada registrador tem o tamanho de 40 bits, incluindo 8 bits para representação de tipo de dados e garbage collection. Na implementação KL1 da PIM/m, foi tomado cuidado com o mecanismo de garbage collection, para isso um garbage collection incremental usando Multiple Reference Bit(MRB) foi introduzido.

## 4.5 Linguagem para o Sistema Operacional PIMOS

Baseados no Parlog e no Concurrent Prolog, uma equipe de pesquisadores desenvolveu a linguagem GHC(Guarden Horn Clauses), que originou a KL0(Kernel Language Zero). Um refinamento desta, produziu em 1987 a linguagem KL1 [Palazzo 1997].

A linguagem denominada KL1 foi utilizada para desenvolver o sistema operacional PIMOS (Parallel Inference Machine Operating System) em 1988. Ela é uma linguagem de alto nível mas, ao mesmo tempo, uma linguagem de máquina, isto significa ser adequada à programação a nível de registradores, posições de memória e portas lógicas

Outras linguagens de programação em lógica foram e ainda são pesquisadas, por exemplo a linguagem de programação em lógica com restrições denominada GDCC. Uma outra, denominada "Quixote" foi produzida para lidar com bases de dados dedutivas e orientadas a objetos. Para o gerenciamento de sistemas paralelos distribuídos foi especificada a linguagem Kappa-P[5].

[5] Knowledge Application-Oriented Advanced Database Management é como um sistema paralelo gerenciador de banco de dados para execução em Máquinas PIM.

### 4.5.1 A linguagem paralela Kernel Language 1 – KL1

A linguagem KL1 possui as seguintes características:

- Shoen[6]: representa um grupo de objetivos. O grupo é utilizado como unidades de controle de execução, nomeados como inicialização, interrupção, recomeço e aborto de execução.
- Prioridade: um objetivo de um programa KL1 é uma unidade de controle prioritário. Cada objetivo tem uma prioridade de valor inteiro associado a mesma. Cada Shoen mantém o máximo e o mínimo de prioridades permitidas para os objetivos descritos acima. A linguagem permite um grande número de níveis prioritários lógicos, os quais são traduzidos para prioridades disponíveis fisicamente, para cada implementação.
- Especificação de Processador: cada objetivo pode ter uma especificação de processador, a qual designa o número de processadores para executar o objetivo. Sem isto, um objetivo é executado no mesmo processador como um objetivo pai.

A máquina de inferência paralela experimental tinha 64 processadores PSI-II conectados em grid, alcançando o pico de desempenho de cerca de 10 MRPS[7] em listas de concatenação.

O sistema operacional PIMOS para Multi-PSI e PIM foi o primeiro desenvolvido utilizando uma implementação KL1.

### 4.5.2 O sistema Operacional PIMOS

Estruturas baseadas em conhecimento têm como característica a existência de uma base de conhecimento, a partir da qual se propõe a criação de um suporte mais inteligente para as aplicações e de melhores ambientes para o usuário. A idéia de sistemas operacionais baseados em conhecimento não é nova. Segundo [Li et al. 1995] eles eram chamados de Sistemas Operacionais Inteligentes e Sistemas Operacionais Baseados em Conhecimento. O sistema operacional PIMOS propunha uma máquina de inferência paralela no sentido de implementar mecanismos de gerenciamento inteligente de sistemas distribuídos.

## 4.6 Resultados do Projeto

Cinco execuções de Máquinas de Inferência Paralela – PIM (parallel inference machines) foram eventualmente desenvolvidas: PIM/m , PIM/p , PIM/i, PIM/k , PIM/c.

O projeto também produziu aplicações para executar nestas máquinas, tais como o Sistema de Gerenciamento de Banco de Dados Kappa, o sistema de raciocínio HELIC-II e o provador de teoremas chamado MGTP.

# 5 Conclusão

Inicialmente acreditava-se que o formalismo matemático poderia ser a solução para o desenvolvimento de máquinas potentes. A máquina abstrata de Warren estimulou os estudos para o desenvolvimento de compiladores prolog que poderiam ser testados em uma máquina lógica abstrata que simulava um hardware construído para melhor se adaptar a realidade da programação em lógica.

Utilizar a lógica no desenvolvimento de hardware e software era o principal objetivo das pesquisas do Projeto de computadores de quinta geração iniciado no Japão e que durou aproximadamente uma década. O projeto não atingiu seus objetivos iniciais, que era comercializar potentes "máquinas lógicas" ou "máquinas prolog". Este potencial seria obtido através do paralelismo que foi aplicado a nível de software utilizando recursos paralelos do prolog e também a nível de hardware através da criação de redes que executavam em paralelo mas que eram formados por elementos de processador que executavam sequencialmente.

Mas, apesar disso, várias idéias seguintes as que foram lançadas neste projeto hoje são realidade, por exemplo a programação lógica distribuída sobre bases de dados usado em OWL[8] – Web Ontology Language e a Computação Paralela[9]. Além dos vários estudos multidisciplinares: Lógica Jurídica, Linguagem natural, Lógicas Paraconsistentes, redes neurais artificiais paraconsistentes, Lógica aplicada à Engenharia de Software, dentre outros.

---

[6] Shoen é a palavra Japonese correspondente a "manor" em Inglês.

[7] MRPS significa Mega reduções por segundo o que corresponde a aproximadamente ao MPLIS (mega inferencias lógicas por segundo) do Prolog

[8] OWL é uma linguagem para definir e instanciar ontologias na Web. É uma tecnologia importante para a futura implementação da Web Semântica. Lógica descritiva e lógica menos expressiva são as mais utilizadas.

[9] Computação paralela é caracterizada pelo uso de várias unidades de processamento ou processadores para executar uma computação de forma mais rápida.

# 6 Bibliografia


Beer, J. (1987). Concepts, Design and Performance Analysis of a Parallel Prolog Machine, Ph.D. Thesis, Tech. Uni. of Berlin, W. Germany.

Buchsbaum, A. (2006). Lógica Geral. Disponível em http://wwww.inf.ufsc.br/~athur/index.php?page=material_didatico&lang=pt. Acesso em 10/07/2008

Dutra, I.C., Geyer C.F.R., Vargas,P.K.(1999). Parallelism in Logic Programming, Intl. School on Advanced Techniques for Parallel Computation with Applications, Natal, RN, Brasil, Sep, 35 pages.

Galante,G. (2004). Programação em Lógica: Prolog. Programa de Pós-Graduação em Computação. UFRGS, Porto Alegre.

Hanus, M. (1988). Formal Specification of a Prolog Compiler. In Proc.Int.Workshop on Programming Language Implementation and Logic Programming, Orléans, May. Springer LNCS 348, pp.273-282

Hassan Aït-Kaci. (1990). "Warren's Abstract Machine -- A Tutorial Reconstruction", MIT Press, Cambridge, Mass.

Beer, J., Giloi, W.K. (1987). POPE — a parallel-operating prolog engine. In Future Generation Computer Systems. Volume 3, Issue 2, May, Pages 83-92.

Kowalski, R. (1973). Predicate Logic as a Programming Language Memo 70, Department of Artificial Intelligence, Edinburgh University.

LI,X., Xing, D., Jun, C., Yuhua, Z., Zhongxus, S. (1995). An Introduction to Intelligent Operating System KZ2. ACM Operating Systems Review, Vol.29 no.1.

Nolt, J., Rohatyn, D. (1991). Lógica. McGraw & Makron Books. São Paulo

Palazzo, L.A.M. (1997). Introdução a programação prolog. EDUCAR Editora da Universidade Católica de Pelotas. Pelotas-RS.

Price, A.M., Lisbôa, M.L. (2003). Linguagens de programação em Lógica. Notas de Aula. Disponível em http://allanedgard.sites.uol.com.br/MAT052/LOGICAN.pdf

Rosa, J. L. G. (1995). A Quinta Geração, *Caderno de Informática*, *Jornal Diário do Povo*, Campinas, SP, 25/05.

Schultz,M.R.O.(2003). Metodologias para o ensino de lógica de programação de computadores. Dissertação apresentada à Universidade Federal de Santa Catarina, Novembro. Acesso em 11 de Agosto de 2008.

Silva, A.C.S., Silveira, C. Prolog. Disponível em http://anacarol.blog.br/old/aulas/artigos_uteis/prolog.pdf

Soler,L. (2006). Arquitetura PIM/m. Instituto de informática, Universidade Federal do Rio Grande do Sul-UFRGS, Rio Grande do Sul. Disponível em http://www.inf.ufrgs.br/procpar/disc/cmp135/trabs/soler/pim.pdf. Acesso em 21 de Agosto de 2008.

Ueda,K. Chikayama, T. (1990). Design of the Kernel Language for the Parallel Inference Machine. The Computer Journal, Vol.33, No.6(Dec), pp. 494-500)

Vargas, Patrícia Kayser. (1997). Exploração de Paralelismo OU em uma Linguagem em Lógica com Restrições. In: SEMANA ACADÊMICA DO CPGC, Porto Alegre, RS. Anais. Porto Alegre: CPGCC/UFRGS. p.31-34

Valente, J.A.(1993). Computadores e Conhecimento: Repensando a educação. UNICAMP, SP: Ed. NIED.