# SIMULAÇÃO DE UMA CLASSE DE PROCESSADORES VETORIAIS

*Claudio L. de Amorim*
Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ
Caixa Postal 68511
21945 Rio de Janeiro - RJ

**Sumário**

Um simulador de um modelo parametrizado de processador vetorial baseado na arquitetura do computador CRAY-1 é apresentado. Os parâmetros do modelo permitem a modificação dos elementos da arquitetura, tais como a estrutura do arquivo de registradores, intercalagem e "buffering" da memória principal, composição das unidades funcionais e processamento de instrução. Vinte e seis parâmetros podem ser utilizados para definir os elementos da arquitetura e diretrizes para o simulador.

O simulador aceita programas escritos em linguagem de máquina do CRAY-1 gerados pelo compilador FORTRAN ou manualmente, a configuração da arquitetura do processador e produz relatórios detalhados do desempenho dos programas.

O simulador foi escrito em Pascal e implantado nos computadores da série CYBER 170 da Control Data e Digirede 8000.

## 1. Introdução

Uma alternativa comumente empregada para obter a execução mais rápida de programas é codificá-los eficientemente de modo a ajustá-los à arquitetura do processador em que serão executados. Esse trabalho, geralmente, significa programar em linguagem assembler as partes que mais consomem tempo do processador. Na maioria dos casos, essas partes correspondem às *construções repetitivas* da linguagem, tais como o "Do-loop" do FORTRAN.

Nos computadores de alto desempenho, em particular o computador vetorial CRAY-1, essa tarefa se torna mais difícil, já que o programador agora tem que lidar com a execução concorrente de instruções escalares e vetoriais, além da alocação de um grande número

de registradores, "buffering" da memória principal e controle da alocação dos recursos do processador.

O simulador apresentado neste trabalho serve para apoiar o desenvolvimento e monitorar o desempenho de programas escritos para o CRAY-1 de maneira eficaz. O simulador é capaz de diagnosticar, para cada instrução simulada, o tempo de espera para iniciar execução, as causas das colisões ocorridas e produzir no final, estatísticas das fontes de riscos ocorridas durante a execução do programa.

Embora essa seja uma ferramenta útil e substancial proporcionada pelo simulador, seu principal objetivo é o de propriciar um ambiente para avaliar o desempenho de arquiteturas de processadores derivadas do CRAY-1. Para isto, os elementos da arquitetura podem ser modificados, incluindo o número de unidades funcionais, registradores escalares e vetoriais, "buffers" de instruções, e introduzir outros mecanismos arquiteturais, tais como "cache" de dados e despacho concorrente de instruções.

A sequência de apresentação é a seguinte. Primeiro a arquitetura do CRAY-1 é brevemente revista. Nas seções seguintes, o modelo de processador vetorial e o simulador são descritos. Para concluir, são feitos comentários sobre a utilização do simulador no ensino e pesquisa no Programa de Sistemas e Computação da COPPE.

## 2. A Arquitetura do Processador CRAY-1

A arquitetura do processador CRAY-1 (Figura 1) compreende uma memória principal com palavras de 64 bits, até 1 Mwords com 16 bancos intercalados, uma unidade de controle com facilidade de pré-carregamento de até 256 pacotes (16 bits) de instruções, 8 registradores vetoriais (V), cada um com até 64 elementos, 8 registradors escalares (S), 8 registradores de endereço (A) e dois "buffers" de registradors T e B.

A unidade de controle coordena o despacho de instruções para os três tipo de processamento: vetorial, escalar e de endereçamento. Em geral, cada um dos modos de processamento é apoiado por um conjunto de registradores e um conjunto de unidades funcionais. Uma exceção é que os registradores escalares podem fornecer um dos operandos nas operações vetoriais.

O CRAY-1 possui uma organização de máquina do tipo registrador-a-registrador, ou seja, operações aritméticas só atuam no arquivo primário de registradores (V, S e A). Os dados são transferidos desse arquivo para as unidades funcionais, e destas de volta para o

arquivo.

No caso de processamento escalar e de endereçamento, existem ainda os conjuntos B e T de registradores que servem de memória intermediária para os registradores A e S, respectivamente. Analogamente, as instruções são carregadas em avanço, antes mesmo de serem selecionadas para execução.

A arquitetura é orientada para que tanto as instruções como a maioria de seus operandos sejam transferidos diretamente dessas memórias locais e raramente da memória principal, e quando o acesso a esta for necessário, ele será feito através do carregamento e armazenamento em bloco, utilizando-se o máximo da banda passante da memória. Maiores detalhes da arquitetura do CRAY-1 poderão ser encontrados em [1].

## 3. O Modelo de Processador Vetorial (MPV)

Um número significativo de fontes de risco na execução de programas no CRAY-1 foram avaliados em [2] com a conseqüente degradação do tempo de processamento. O MPV e seu simulador permitem investigar alternativas para solucionar tal problema, através do uso de mecanismos arquiteturais. O MPV considera a arquitetura do CRAY-1, seu conjunto de instruções e tempos como ponto de referência, e seu simulador é portanto capaz de executar programas escritos para o CRAY-1. Por outro lado, ele permite expandir os limites da arquitetura através de sua parametrização, de forma a proporcionar diferentes projetos arquiteturais. Isto é obtido ao permitir primeiro, modificações no número de registradores vetoriais, escalares e de endereço, barramentos de acesso, "buffers" de dados e de instruções, bancos de memória e unidades funcionais. Segundo, permitir a substituição dos "buffers" de dados por uma "cache". E terceiro, dar suporte a um mecanismo de processamento de instruções em avanço com distância de avanço variável, de tal forma que instruções possam ser despachadas concorrentemente.

## 4. O Mecanismo de Controle de Instruções

A Figura 2 retrata o fluxo de processamento das instruções no MPV. Pacotes de instruções são retirados dos "buffers" de instruções e mantidos no **registrador da próxima instrução**, onde eles serão examinados pelo **mecanismo de controle de despacho**. De acordo com o código de operação, o controlador de despacho verifica a **informação de controle de acesso aos recursos** do processador e determina o primeiro período de

relógio em que a instrução pode ser despachada. Se o **período de relógio de despacho** for igual ao **período de relógio atual**, então a instrução é entregue ao mecanismo de controle de execução para ser executada. Caso contrário, a instrução é rotulada com o referido período de relógio futuro e inserida na **fila de execução**, onde irá esperar pelo período de relógio rotulado. O primeiro período de relógio na fila de execução é indicado pelo **registrador de rótulo-cabeça**. Sempre que o atual período de relógio for igual ao do relógio do registrador de rótulo-cabeça, a instrução correspondente na fila é transferida para o mecanismo de controle de execução e executada.

Para evitar acessos simultâneos aos recursos do processador e manter compatibilidade com o modo de operação do CRAY-1, somente uma instrução pode ser despachada num período de relógio, o que exige prioridade de instruções pendentes na fila sobre instruções sendo examinadas. Isto também simplifica o processo de modelagem.

O mecanismo de controle de instruções de escrita permite que instruções sejam despachadas concorrentemente. Isto é possível devido ao mecanismo de controle de acesso aos recursos, o qual será descrito a seguir.

## 5. Controle de Acesso aos Recursos

O despacho concorrente de instruções nos leva ao problema de interdependência de instruções [3], que pode ser reduzido à resolução do problema de contenção causado por instruções que competem por recursos. A solução [4] empregada pelo MPV é basicamente a seguinte. Associado a cada recurso, excluindo barramentos de acesso a registradores escalares, existe um indicador que visa definir um período de relógio (**pr**) a partir do qual o recurso estará livre (FREECP). Para os recursos que podem ser lidos e escritos, um outro indicador (LASTWRT) indicará o **pr** no qual ocorreu a última modificação de valor.

Sempre que um recurso é reservado, seus indicadores são atualizados. O indicador FREECP sempre avança, enquanto que LASTWRT só avança numa operação de escrita quando se torna igual a FREECP.

Os recursos do tipo barramento de acesso aos registradores escalares e de endereço registram todos os **pr** reservados em vez de um único indicador FREECP.

Havendo mais de uma cópia de um recurso, a primeira disponível é selecionada.

Os recursos são automaticamente liberados, já que para cada instrução a quantidade de **pr** pela qual cada recurso é reservado pode ser determinada quando a instrução é

despachada.

## 6. O Mecanismo de "Cache" de Dados

O MPV oferece um mecanismo simples de "cache" de dados que manipula automaticamente acesso escalar à memória principal. A "cache" é implementada sobre os "buffers" de dados B e T. A organização da "cache" é do tipo mapeada por setor, e definida através de três parâmetros do simulador: número de setores, número de blocos por setor e número de elementos por bloco. O tamanho mínimo da "cache" é de 64 elementos, podendo ser expandida através da parametrização dos "buffers" B e T.

A "cache" é para ser utilizada em programas puramente escalares. Em programas com instruções escalares e vetoriais o problema de coerência de dados pode ocorrer. Uma solução para este problema e o algoritmo de troca de setores utilizado é discutido em [6].

## 7. Simulação do Modelo de Processador Vetorial [7]

Inicialmente, a configuração do processador é definida, o programa do usuario é carregado na memória do MPV e os recursos do processador a ser simulado são inicializados. Os passos seguintes de 1-5, sumarizados abaixo, são repetidos até a última instrução ser executada. As medidas de desempenho são então impressas. Uma descrição detalhada dos passos é encontrada em [6].

1. **Simula o tráfego de instruções do CRAY-1** levando em consideração o número e o tamanho dos "buffers" de instruções. Para carregar um "buffer" com instruções é preciso que a memória esteja livre, o que será indicado calculando-se

$$\max(FREECP), \quad \forall\, FREECP \in \text{ bancos de memória .}$$

2. **Avanca o relógio e inicia instruções pendentes.** Instruções pendentes durante o tráfego de instruções são despachadas.
3. **Carrega a próxima instrução e avança o apontador de programa.** O carregamento do registrador de instruções é simulado e a instrução é rotulada com uma etiqueta de atividade a ser utilizada no relatório de conflitos.
4. **A atividade reivindica recursos e os tempos de despacho e execução são definidos.** As principais funções deste passo são as seguintes:
   (i) Determinar os atributos dinâmicos das instruções, se for o caso.

(ii) Determinar a disponibilidade (FREECP) de cada recurso requisitado.

(iii) Definir o tempo de despacho (pr) através da disponibilidade de recursos produzidos por (ii) e os pr's das instruções pendentes.

(iv) Determinar o tempo de execução da instrução e atualizar os indicadores de controle de cada recurso envolvido na instrução.

(v) Preparar os pacotes de execução da instrução, inclusive a cópia de operando para execução posterior. Define se a instrução será ou não colocada na fila.

5. **Testa disponibilidade na fila de execução.** Se não houver fila de execução (distância de avanço = 0), o relógio irá avançar o suficiente para igualar o tempo de despacho e/ou tempo de execução, dependendo da instrução. Caso haja uma fila, a instrução poderá iniciar a execução (**pr** de despacho = **pr** atual) ou ser enfileirada. Verifica se a fila está cheia e outras situações especiais.

Durante os passos, as medidas de desempenho são coletadas para que, ao final da simulação, elas sejam avaliadas e estatísticas sobre o programa sejam produzidas.

## 8. Conclusões

O simulador foi implantado no super-micro da Digirede e tem sido utilizado pelos alunos de pós-graduação dentro da disciplina de processamento paralelo. Algoritmos numéricos clássicos são programados pelos alunos e testados no simulador, variando a configuração do processador de acordo com o desempenho obtido. Com o auxílio das estatísticas de conflito, os alunos são também capazes de reprogramar os códigos para obter maior eficiência. O simulador mostra-se um instrumento bastante eficaz para uma introdução a problemas de supercomputação.

Como instrumento de apoio à pesquisa em processamento vetorial, o simulador está sendo utilizado na avaliação de um compilador vetorizado [8], desenvolvimento de um compilador ACTUS [9], avaliação de técnicas de vetorização e uma versão para simular configurações multiprocessadoras do CRAY-XMP [10].

## Referências


[1] R. L. Sites, "An Analysis of the CRAY-1 Computer", *IEEE-ACM Proc. of the 5th Annual Symp. on Comp. Architectures*, SIGARC News, vol. 10, no. 3, Abril de 1982.

[2] C. L. Amorim, "Avaliação de Conflitos no Desempenho do CRAY-1", *Anais do VI Congresso da Soc. Bras. de Computação*, pp. 175–184, Julho de 1986.

[3] R. M. Keller, "Lookahead Processors", *Computing Surveys*, vol. 7, no. 4, Dezembro de 1975.

[4] C. L. Amorim, "Simulated Performance of a Class of Vector Processors", *II Int. Conf. on Supercomputing*, Santa Clara, CA (USA), Maio de 1987. A ser impresso.

[5] A. J. Smith, "Cache Memories", *Computing Surveys*, vol. 14, no. 3, Setembro de 1982.

[6] C. L. Amorim, "Simulated Performance of a Class of Vector Processors", Tese de Ph.D., Computing Dept., Imperial College, 1984.

[7] C. L. Amorim, "The Vector Processor Simulator User's Guide", Relatório Técnico, Programa de Sistemas, COPPE/UFRJ, 1986.

[8] N. O. Alvarez *et al.*, "Um Compilador Vetorizado para Maquinas Vetoriais", *Anais do VI Congresso da Soc. Bras. de Computação*, pp. 616–620, Julho de 1986.

[9] R. H. Perrot *et al.*, "The Programming Language ACTUS", *Software — Practice and Experience*, vol. 13, pp. 305–322, 1982.

[10] S. Chen, "Large-Scale and High-Speed Multiprocessor Systems for Scientific Applications: CRAY XMP-2 Series", *Tutorial on Supercomputers: Design and Applications*, Kai Hwang (ed.), IEEE Computer Society, 1984.